UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Universidad Internacional de La Rioja
Máster universitario en Seguridad Informática

Herramienta para la automatización de análisis forenses en sistemas iOS

Trabajo Fin de Máster

presentado por: Fernández Valero, Diego

Director/a: Alarcos Alcázar, Bernardo

Ciudad: Zaragoza
Fecha: 22 de Julio de 2014

ÍNDICE

1. Introducción 1
2. Estructura de un Backup de iOS 3
3. Estado del arte 7
4. Identificación de requisitos 11
5. Arquitectura y diseño de la herramienta 14
   5.1 Detalles de la Herramienta 14
   5.2 Componente de recolección de evidencias 15
   5.3 Componente de informe de evidencias 18
   5.4 Códigos de retorno de la aplicación 21
   5.5 Opciones y argumentos de la herramienta 22
6. Esquema de la Base de Datos 24
7. Sistema de Plugins 25
   7.1 Plugin para el análisis de las llamadas 26
   7.2 Plugin para el análisis de los mensajes de texto 27
   7.3 Plugin para el análisis de los contactos 28
   7.4 Plugin para el análisis del historial web 31
   7.5 Plugin para el análisis del Calendario y los Eventos 32
   7.6 Plugin para el análisis de las Notas 34
   7.7 Plugin para el análisis de las conversaciones de WhatsApp 35
8. Estudio de diferentes formatos de informe 38
   8.1 Formato de informe HTML 38
   8.2 Modelos de informe PDF 43
9. Modulo de Reporting 45
   9.1 Generar informe HTML 47
   9.2 Generar informe PDF 49
   9.3 Estructuras HTML 51

10. Diseño técnico de la herramienta....53
10.1 Diagramas UML de clases....53
10.1.1 Paquete db....54
10.1.1.1 Diagrama UML de la clase database....54
10.1.2 Paquete modules....57
10.1.2.1 Diagrama UML de la clase manifest....57
10.1.2.2 Diagrama UML de la clase Info....58
10.1.2.3 Diagrama UML de la clase DateUtils....59
10.1.2.4 Diagrama UML de la clase Plist_converter....60
10.1.2.5 Diagrama UML de la clase dto....61
10.1.2.6 Diagrama UML de la clase Reporting....62
10.1.2.7 Diagrama UML de la clase pdf_report....65
10.1.2.8 Diagrama UML de la clase PluginBase....67
10.1.3 Paquete plugins....69
10.1.3.1 Diagrama UML de la clase Plugin....69
11. Manual de Usuario....70
11.1 Filtrando por palabras clave....73
11.2 Filtrando por fechas....76
11.3 Filtrando por tipos....80
11.4 Obteniendo toda la información....83
12. Conclusiones....87
13. Trabajo futuro....89
14. Bibliografía....90
15. Referencias....92

ÍNDICE DE FIGURAS

Figura 1 - Archivos de una copia de seguridad de iOS 4
Figura 2 - Contenido de Domains.plist 5
Figura 3 - Licencia de Oxygen Forensic 7
Figura 4 - GUI de iPhone Analyzer 8
Figura 5 - Diagrama de flujo de datos de iBAFT 15
Figura 6 - Diagrama de Flujo de Datos de ibaft.py 17
Figura 7 - Diagrama de Flujo de Datos de ibaft_report.py 20
Figura 8 - Esquema de la base de datos de la herramienta 24
Figura 9 - Estructura de call_history.db 26
Figura 10 - Estructura de sms.db 27
Figura 11 - Estructura de AddressBook.sqlitedb 30
Figura 12 - Formato de cada visita en el archivo History.plist 32
Figura 13 - Estructura de Calendar.sqlitedb 33
Figura 14 - Estructura de notes.sqlitedb 34
Figura 15 - Estructura de ChatStorage.sqlite (WhatsApp) 36
Figura 16 - Cuadro de búsqueda avanzada de un modelo de informe 38
Figura 17 - Busqueda por fecha en el informe HTML 38
Figura 18 - Ejemplo de tabla con DataTables 40
Figura 19 - Ejemplo de una conversación de WhatsApp en el informe HTML 41
Figura 20 - Ejemplo de Timeline en el informe HTML 42
Figura 21 - Ejemplo de informe PDF 43
Figura 22 - Flujo de Datos del componente de reporting 46
Figura 23 - Flujo de ejecución para generar el informe HTML 48
Figura 24 - Flujo de ejecución para generar el informe PDF 50
Figura 25 - Clases de la herramienta 53
Figura 26 - Diagrama UML de la clase database 55
Figura 27 - Diagrama UML de la clase manifest 57
Figura 28 - Diagrama UML de la clase Info 58
Figura 29 - Diagrama UML de la clase DateUtils 59
Figura 30 - Diagrama UML de la clase Plist_converter 60
Figura 31 - Diagrama UML de la clase dto 61
Figura 32 - Diagrama UML de la clase Reporting 63
Figura 33 - Diagrama UML de la clase pdf_report 65
Figura 34 - Diagrama UML de la clase PluginBase 67

Figura 35 - Diagrama UML de la clase Plugin 69
Figura 36 - Directorios de ibaft y el backup de iOS en Debian....70
Figura 37 - Sintaxis de ibaft.py 70
Figura 38 - Texto de ayuda de ibaft.py 71
Figura 39 - Reconstrucción del backup 71
Figura 40 - Directorio destino 71
Figura 41 - tree de /usr/local/backup/backup (I)....72
Figura 42 - tree de /usr/local/backup/backup (II)....72
Figura 43 - tree de /usr/local/backup/backup (III)....72
Figura 44 - Sintaxis de ibaft_report.py 73
Figura 45 - Texto de ayuda de ibaft_report.py 73
Figura 46 - Ejecución de ibaft_report.py filtrando por palabras clave....73
Figura 47 - Timeline filtrado por palabras clave 74
Figura 48 - Conversaciones Whatsapp filtradas por palabra clave 74
Figura 49 - Comunicaciones sin datos....75
Figura 50 - Contactos sin datos....75
Figura 51 - Agenda sin datos 75
Figura 52 - Historial web sin datos 75
Figura 53 - Informe PDF filtrado por palabras clave 76
Figura 54 - Ejecución de ibaft_report.py filtrando por fechas 76
Figura 55 - Timeline filtrado por fechas 77
Figura 56 - Conversaciones de WhatsApp filtradas por fecha 77
Figura 57 - Comunicaciones filtradas por fecha....78
Figura 58 - Contactos filtrados por fechas 78
Figura 59 - Agenda filtrada por fechas....78
Figura 60 - Historial web filtrado por fechas 78
Figura 61 - Informe PDF filtrado por fechas (Timeline) 79
Figura 62 - Informe PDF filtrado por fechas (WhatsApp) 79
Figura 63 - Informe PDF filtrado por fechas (Comunicaciones) 80
Figura 64 - Informe PDF filtrado por fechas (Agenda) 80
Figura 65 - Ejecución de ibaft_report.py 80
Figura 66 - Timeline filtrado por tipos 81
Figura 67 - Contactos filtrados por tipo....81
Figura 68 - Comunicaciones filtradas por tipo....82
Figura 69 - Comunicaciones filtradas por tipo (25 por página)....82
Figura 70 - Búsqueda dentro de la tabla de Comunicaciones....83
Figura 71 - Ejecución de ibaft_report.py sin filtros 83

Figura 72 - Parte del timeline sin filtrar 84
Figura 73 - Conversaciones Whatsapp.....85
Figura 74 - Agenda.....85
Figura 75 - Historial web 85

ÍNDICE DE TABLAS

Tabla 1 - Rutas donde se almacenan los backups de iOS en cada sistema operativo 3
Tabla 2 - Atributos de seguridad de los archivos en iOS 5
Tabla 3 - Estructura del fichero Manifest.mbdb 6
Tabla 4 - Comparación de las herramientas 10
Tabla 5 - Códigos de retorno de ibaft.py 21
Tabla 6 - Códigos de retorno de ibaft_report.py 22
Tabla 7 - Opciones de ibaft.py 22
Tabla 8 - Opciones de ibaft_report.py 23
Tabla 9 - Documentación de la clase database 56
Tabla 10 - Documentación de la clase manifest 58
Tabla 11 - Documentación de la clase Info 59
Tabla 12 - Documentación de la clase DateUtils 60
Tabla 13 - Documentación de la clase Plist_converter 61
Tabla 14 - Documentación de la clase dto 62
Tabla 15 - Documentación de la clase Reporting 63
Tabla 16 - Documentación de la clase pdf_report 66
Tabla 17 - Documentación de la clase PluginBase 68

RESUMEN

El presente trabajo pretende mostrar el diseño y desarrollo de una herramienta que permite analizar una copia de seguridad de iTunes de dispositivos iOS para poder obtener un *timeline* de las acciones realizadas, y con ello, poder demostrar o rebatir los hechos pertinentes en una investigación o en un procedimiento judicial.

La herramienta está desarrollada en Python y funciona bajo línea de comandos, analizando la información del backup y generando un informe con las evidencias coincidentes a los criterios de búsqueda del investigador.

Palabras clave: forense, iOS, análisis forense, investigación, software

ABSTRACT

This paper aims to show the design and development of a tool to analyze an iTunes backup of iOS devices to obtain a timeline of actions taken and thus able to prove or disprove the relevant facts in an investigation or judicial proceedings.

The tool is developed in Python and works on command line. Analyzes information backup and generates a report with evidence matching the search criteria of the investigator.

Keywords: forensics, iOS, investigation, software, phone forensic

AGRADECIMIENTOS

En primer lugar, quisiera agradecer a todas las personas, profesores y amigos, que hicieron crecer mi entusiasmo por las nuevas tecnologías, lo que me ha ayudado a convertirme en el profesional que hoy en día soy.

Quiero dar las gracias también a Raquel, por aguantarme todo este tiempo y apoyarme en todas las decisiones importantes de mi vida. A mi familia, porque siempre me han animado a continuar con mi formación, y a María, por prestarme una copia de seguridad de iPhone, que sin ella no podría haber desarrollado este trabajo.

Por último, quiero dar las gracias también a Raúl y a Pilar, por haberme ayudado durante todo este tiempo en todo lo que he necesitado.

1. INTRODUCCIÓN

En la actualidad, el uso de dispositivos móviles está creciendo a un ritmo imparable. Según un estudio de Cisco[2], a finales del año 2012 el número de *smartphones* superó el número de habitantes del planeta, y para el año 2018, se prevé que este número ascienda hasta la cifra de diez mil millones.

Es una realidad que los hábitos de las personas han cambiado significativamente en los últimos años, priorizando la portabilidad de los dispositivos y dejando de lado esos equipos de sobremesa. Es por eso que la cantidad de información almacenada en los *smartphones* y *tablets* es cada vez mayor.

El incremento del uso de estos dispositivos ha hecho que la información contenida en los mismos sea una evidencia necesaria en cualquier investigador forense. Pero la adquisición de evidencias digitales en los terminales móviles no siempre es sencilla. Si el dispositivo no tiene permisos de *superusuario* (root), puede ser tedioso acceder a su información, especialmente cuando el sistema del terminal es el diseñado por Apple, iOS.

Si no se dispone de un dispositivo hardware para poder adquirir un clonado forense del terminal, que normalmente son dispositivos con un coste muy elevado, se puede obtener información relevante a través del backup que se realiza con iTunes.

El objetivo de este trabajo es desarrollar una herramienta que analice el backup de iTunes, en busca de la información que puede ser relevante para un investigador forense, y la almacene de una forma centralizada para poder generar un informe con los resultados de dicho análisis, permitiendo filtrar por fechas y palabras clave, facilitando así la tarea de estudio de las evidencias.

Una de las contribuciones de la herramienta es el poder disponer de ella de forma portable y libre, ya que se trata de una herramienta Open Source, que se distingue por ello de otras herramientas del mismo sector, como se detalla en el Capítulo 3.

Además de ser libre y portable, presenta unos formatos de informe que hacen más visual la investigación, con su correspondiente aumento de productividad, a través de maquetación personalizadas para la línea temporal y las conversaciones de mensajería instantánea.

Recapitulando, las características de la herramienta que constituyen las principales contribuciones serán:

- Es una herramienta Gratuita
- Es una herramienta Open Source[1]
- Es una herramienta Multiplataforma[2]
- Es una herramienta Portable[3]
- Es una herramienta Flexible[4]
- Presenta informes en múltiples formatos
- En el informe HTML, se presenta un diseño novedoso[5] para la visualización de los datos.

Aunque algunas de las herramientas detalladas en el Capítulo 3 (Estado del arte) cumplan con alguna de las características anteriormente enumeradas, la verdadera contribución es el desarrollo de una herramienta que cumpla con todas ellas.

El resto de la memoria se estructura de la siguiente forma. El capítulo dos detalla la estructura de una copia de seguridad de iOS realizada a través de iTunes, el capítulo tres presenta un estudio del estado del arte, en el capítulo cuatro se identifican los requisitos de la herramienta a desarrollar. En el capítulo cinco se detalla el diseño y la arquitectura de la herramienta, mientras que en el capítulo seis se muestra el esquema de base de datos interno de la herramienta.

El capítulo siete está dedicado al análisis del sistema de los plugins, y cada plugin que viene con la primera versión de la herramienta. El capítulo ocho realiza un estudio de los diferentes formatos de informe, para acabar escogiendo el más adecuado. En el capítulo nueve se detalla el funcionamiento del módulo de reporting, encargado de generar los informes.

El capítulo diez contiene el diseño técnico de la herramienta, con diagramas UML de las clases que componen el software. EL capítulo once muestra un manual de usuario, dónde se explica el uso de la herramienta. Por último, los capítulos doce, trece y catorce, contienen las conclusiones, trabajo futuro y referencias, respectivamente.

[1] El código fuente es accesible y se puede modificar para cualquier fin
[2] Se puede ejecutar en cualquier sistema operativo
[3] No necesita instalación
[4] Posibilidad de aumentar funcionalidades mediante plugins
[5] Conjunto de propiedades CSS que dan un aspecto visual que hace más fácil el seguimiento para grandes cantidades de datos.

2. ESTRUCTURA DE UN BACKUP DE iOS

Cuando se realiza una copia de seguridad del sistema operativo de Apple, a través de su aplicación de escritorio iTunes, se crea y almacena un directorio que contiene los ficheros que conforman la copia de seguridad del dispositivo iOS.

Esta ruta cambia en los diferentes sistemas operativos[1], que se resumen en la siguiente tabla:

Tabla 1 - Rutas donde se almacenan los backups de iOS en cada sistema operativo

OS	Ruta Backup iOS
Mac	`~/Librería/Application Support/MobileSync/Backup/`
Windows XP	`C:\Documents and Settings\[USER]\Application Data\Apple Computer\MobileSync\Backup\`
Windows Vista	`C:\Users\[USER]\AppData\Roaming\Apple Computer\MobileSync\Backup\`
Windows 7	`C:\Users\[USER]\AppData\Roaming\Apple Computer\MobileSync\Backup\`
Windows 8	`C:\Users\[USER]\AppData\Roaming\Apple Computer\MobileSync\Backup\`

Como se puede observar, en Windows Vista, Windows 7 y Windows 8, la ruta es la misma, siendo `[USER]` el nombre de usuario que realiza la copia de seguridad del dispositivo.

Cada copia de seguridad se crea en la ruta correspondiente, dentro de una carpeta que tiene por nombre el UDID (Unique Device ID) del dispositivo, que está formado por 40 caracteres hexadecimales, que identifican de manera unívoca el dispositivo.

Dentro de este directorio se encuentran los ficheros de la copia de seguridad del dispositivo, pero no están en un formato cómodo para el ojo humano. Al abrir el directorio se observa una gran cantidad de ficheros con nombres un tanto crípticos, que no son amigables para el ojo humano, tal y como se muestra en la Figura 1. Estos nombres se corresponden con el resultado de un algoritmo de hash[4], concretamente SHA1, de la concatenación del nombre de dominio y la ruta del archivo en el dispositivo, anexados con el símbolo '-'.

En iOS se clasifican los datos en 12 dominios (11 dominios de sistema y dominio de aplicación). Estos dominios se encuentran definidos en /Library/Backup/Domains.plist [4], que contiene los datos que se muestran en la Figura 2.

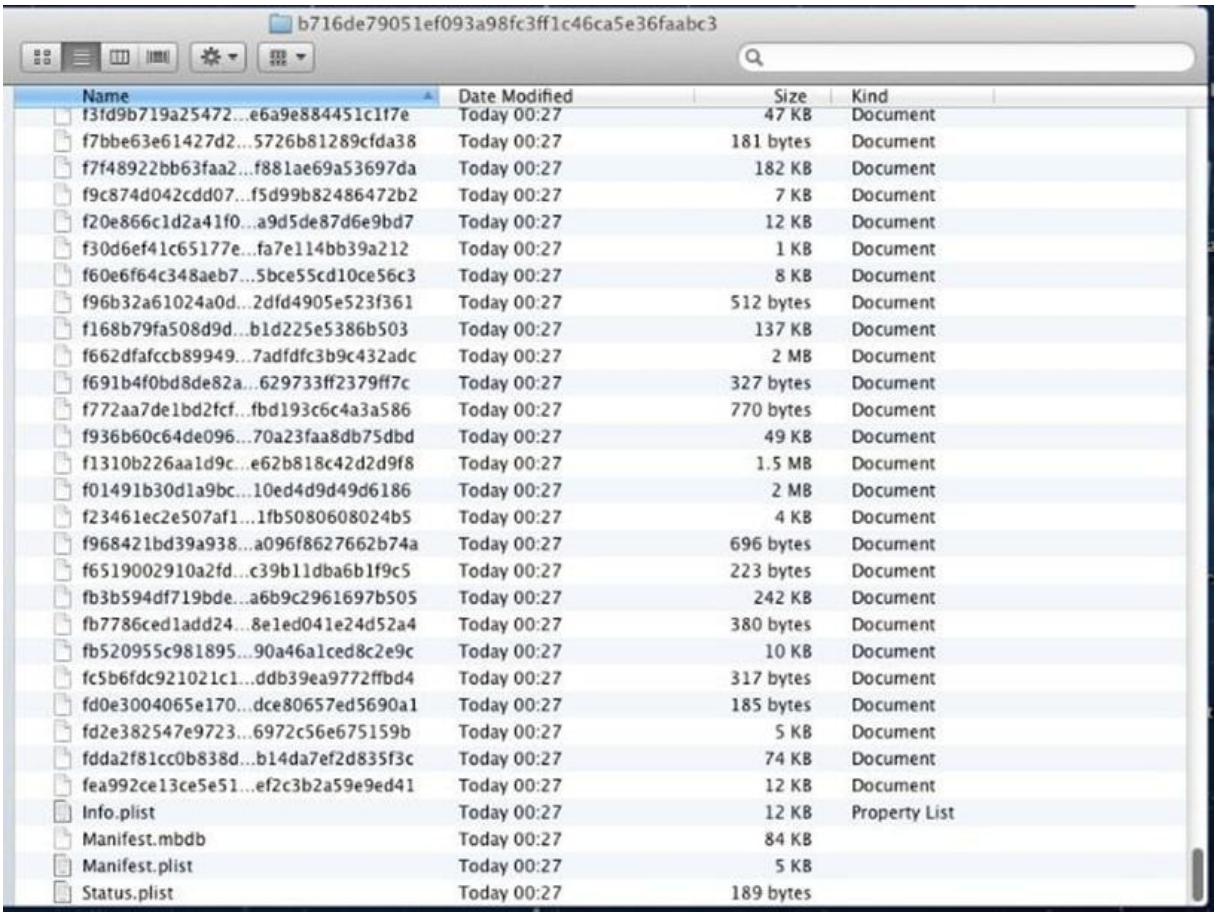
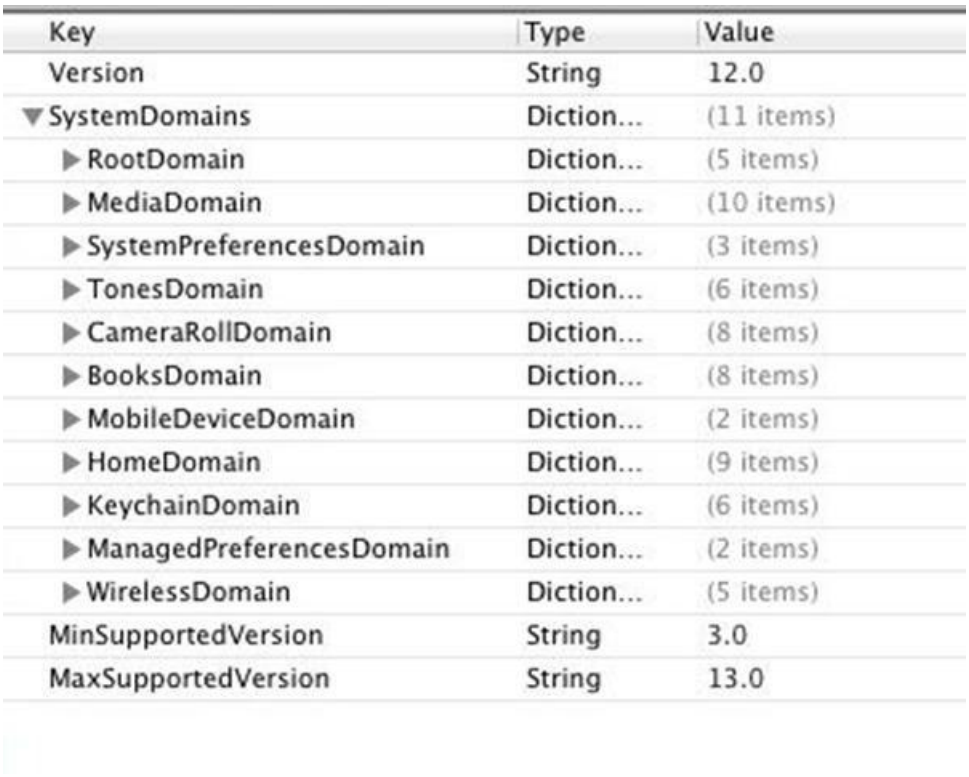


Figura 1 - Archivos de una copia de seguridad de iOS

Para poder identificar qué fichero corresponde con el fichero dentro del dispositivo entra en juego un fichero clave en los backup de iOS, el fichero Manifest.mbdb, que no es más que una base de datos que contiene información del resto de ficheros y sus propiedades.

La estructura de este fichero Manifest.mbdb es bien conocida[3], y se detalla en la Tabla 3. Así pues, este archivo actúa como una especie de índice, mostrando información acerca de que nombre de fichero codificado en SHA1 se corresponde con cada fichero del dispositivo. Esto permite entre otras cosas reconstruir el backup en un formato legible de una manera automatizada, como se verá en detalle en el Capítulo 5. Arquitectura y diseño de la herramienta, en la página 14.

Además del Manifest.mbdb, también existen otros ficheros, como el Info.plist, que contiene información diversa, como las aplicaciones instaladas en el dispositivo, el número IMEI, la fecha de la copia o el número de serie del terminal.



Key	Type	Value
Version	String	12.0
▼SystemDomains	Diction...	(11 items)
▶RootDomain	Diction...	(5 items)
▶MediaDomain	Diction...	(10 items)
▶SystemPreferencesDomain	Diction...	(3 items)
▶TonesDomain	Diction...	(6 items)
▶CameraRollDomain	Diction...	(8 items)
▶BooksDomain	Diction...	(8 items)
▶MobileDeviceDomain	Diction...	(2 items)
▶HomeDomain	Diction...	(9 items)
▶KeychainDomain	Diction...	(6 items)
▶ManagedPreferencesDomain	Diction...	(2 items)
▶WirelessDomain	Diction...	(5 items)
MinSupportedVersion	String	3.0
MaxSupportedVersion	String	13.0

Figura 2 - Contenido de Domains.plist

Como es de esperar, el primer paso para realizar un análisis de una copia de seguridad de iTunes de iOS, es comprender el fichero Manifest.mbdb, y poder leer su contenido, para conseguir un índice a partir del cual buscar los archivos que se consideren relevantes.

Una característica importante que se observa en la estructura del archivo Manifes.mbdb (ver Tabla 3) es el nivel de protección del archivo, que desde iOS4 se puede especificar haciendo uso de diferentes atributos, definidos en la Tabla 2

Tabla 2 - Atributos de seguridad de los archivos en iOS

ID	Protection Class	Descripción
1	**NSProtectionComplete**	El fichero es accesible únicamente cuando el móvil está desbloqueado
2	**NSFileProtectionCompleteUnlessOpen**	El fichero es accesible si el sistema está desbloqueado o el manejador se quedó abierto antes de bloquearse
3	**NSFileProtectionCompleteUntilFirst UserAuthentication**	El fichero es accesible desde el primer desbloqueo hasta que se reinicie
4	**NSProtectionNone**	El fichero es accesible incluso cuando el terminal está bloqueado
5	**NSFileProtectionRecovery**	No documentado

Otra característica que requiere especial mención es el formato Epoch de las fechas. El formato Epoch o Timestamp Unix, se define como la cantidad de segundos transcurridos desde la medianoche UTC (00:00:00) del 1 de Enero de 1970. Este formato es universalmente utilizado en numerosos sistemas computacionales. Sin embargo, a pesar de estar ampliamente estandarizado, Apple utiliza un formato propio en algunas de sus aplicaciones, como se verá más adelante, que contabilizan los segundos desde el 1 de Enero de 2001. Por tanto será necesario tener este aspecto en cuenta a la hora de procesar las fechas, que son un tipo de datos de suma importancia en un análisis forense.

Tabla 3 - Estructura del fichero Manifest.mbdb

Tipo	Dato	Descripción
String	Dominio	Nombre del dominio
String	Ruta	Ruta del fichero
String	Destino	Ruta absoluta para los enlaces simbólicos
String	Hash	Hash SHA1.
String	Clave de cifrado	0xFFFF para ficheros no cifrados
Unsigned Int 16	Modo	Identifica el tipo de fichero. 0xA000 para enlaces simbólicos, 0x4000 para directorios y 0x8000 para ficheros
Unsigned Int 64	Inodo	Número de inodo
Unsigned Int 32	User ID	UID del fichero, generalmente 501
Unsigned Int 32	Group ID	GUID del fichero, generalmente 501
Unsigned Int 32	Fecha Última Modificacion	Fecha de última modificación en formato Epoch
Unsigned Int 32	Fecha Último Acceso	Fecha del último acceso en formato Epoch
Unsigned Int 32	Fecha Creación	Fecha de creación en formato Epoch
Unsigned Int 64	Tamaño	Tamaño del fichero. 0 para enlaces simbólicos y directorios
Unsignet Int 8	Protection Class	Nivel de protección, desde 0x1 hasta 0xB
Unsignet Int 8	Número de propiedades	Número de propiedades
String	Nombre de propiedad	Nombre de la propiedad del fichero
String	Valor de propiedad	Valor de la propiedad

3. ESTADO DEL ARTE

Una de las herramientas más populares para el análisis de dispositivos móviles es Oxygen Forensic, que es una *suite* forense **comercial** que permite obtener información del dispositivo que se analiza.

Es una herramienta muy completa, con entorno gráfico, para sistemas Windows, que entre sus funciones tiene un extractor de backups de iTunes. Se trata de una herramienta profesional, orientado a grandes organizaciones, o empresas que puedan asumir su coste, que supera los 2.000 Euros[6], tal y como muestra la Figura 3

Figura 3 - Licencia de Oxygen Forensic

Sin embargo, Oxygen Forensic analiza el dispositivo conectándose directamente, a través de un cable, por lo que en la mayoría de las ocasiones, la amplitud del análisis es mayor. Pero hay que mencionar que analizar el dispositivo directamente nunca fue el objetivo de este trabajo.

Aunque el sector de aplicación de esta herramienta (Oxygen Forensic) es el mismo que el de la herramienta desarrollada en este trabajo (iBAFT), análisis forense en dispositivos iOS, el público objetivo de estos software son diferentes, ya que la primera busca ser una herramienta profesional de la que obtener beneficios, la segunda es una herramienta gratuita, Open Source que pretende automatizar los procesos de análisis forenses *manuales*, en los que no se cuenta con una gran *suite forense*.

Existe otra herramienta en el mercado para "*analizar*" backups de iTunes, iPhone Analyzer, que además es gratuita. Está desarrollada en Java, por lo que es multiplataforma, al igual que iBAFT, y posee un interfaz gráfico.

[6] http://www.shareit.com/product.html?productid=300393987

Se ha entrecomillado el término analizar, porque lo que hace realmente esa herramienta es extraer la información de la copia de seguridad de iTunes, leyendo los archivos Info.plist y Manifest.mbdb, tal y como se ha comentado en el Capítulo 2, y mostrar dicha información en un árbol de directorios a través de una interfaz gráfica. Se puede observar un ejemplo de uso de esta herramienta en la Figura 4.

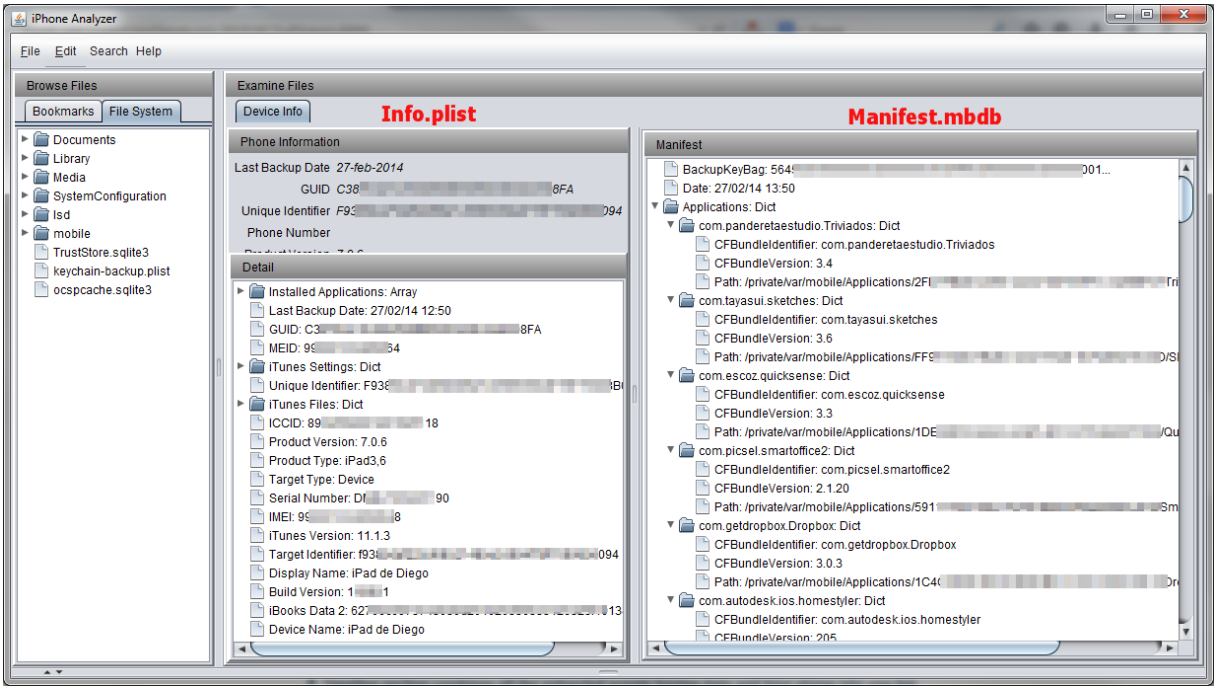


Figura 4 - GUI de iPhone Analyzer

El uso de la herramienta desarrollada permite poner en contexto la información obtenida a través de la copia de seguridad de iTunes, además de poder restaurar los archivos, tal y como hace iPhone Analyzer.

Por ese motivo, iBAFT aporta una nueva contribución a las herramientas forenses, ya que es la única que pone en contexto la información, en lugar de limitarse a extraerla o mostrarla para que sea el investigador quien la analice. Además, a través de la extensión mediante plugins permite centrarse en los aspectos más relevantes para cada investigación, aportando valor así al investigador, permitiéndole ahorrar tiempo y automatizar el proceso.

Es posible que en la versión inicial de la herramienta, no se disponga de todos los plugins que pueden tener otras suites forenses como la ya mencionada Oxygen Forensic, pero tampoco pretende ser un desarrollo monolítico que tenga como objetivo la producción de una herramienta preparada para todos los casos, sino que pretende ser una herramienta ligera y extensible que cada usuario pueda personalizar para sus investigaciones.

Además, conforme se desarrollen nuevos plugins que añadan más funcionalidades a la herramienta, ésta irá creciendo hasta poder cubrir toda la superficie de análisis del dispositivo, compitiendo así con grandes herramientas profesionales, y proporcionando una alternativa open source de calidad.

Lamentablemente, un Trabajo de Fín de Master no proporciona el tiempo suficiente para el desarrollo de una herramienta lista para comercializar y que aporte toda las funcionalidades que un usuario desearía, aunque ese no es el objetivo de este Trabajo.

El objetivo de este proyecto es el desarrollo de una herramienta que permita a la comunidad ampliar sus funcionalidades para hacerla crecer, y que sirva de utilidad en análisis manuales. No se pretende en ningún momento, al menos en las primeras versiones, sustituir ninguna herramienta de análisis forense, sino aportar un nuevo concepto de herramienta, que no sólo muestre la información de un dispositivo, sino que la muestre en un contexto que aporte valor al usuario.

Así pues, las contribuciones de esta herramienta son:

- Es una herramienta Gratuita.
- Es una herramienta Open Source, por lo que cualquier persona puede modificarla para adaptarla a sus necesidades, aunque la herramienta es lo suficientemente flexible como para no tener que modificar el núcleo, y ampliar funcionalidades únicamente con plugins.
- Es una herramienta Multiplataforma. Al estar desarrollada en Python, sólo es necesario tener instalado un intérprete de Python, independientemente del sistema operativo en el que se esté ejecutando.
- Es una herramienta Portable, ya que es muy ligera y no necesita instalación. Puede ser fácilmente transportable en cualquier USB.
- Es una herramienta Flexible, ya que permite el desarrollo e implantación de nuevos plugins sin tener que realizar modificaciones en la herramienta.
- Presenta diferentes formatos de informe (HTML y PDF). Uno está orientado a la visualización de los datos, para ayudar a obtener una conclusión, y otro para exponer los datos en formato impreso.
- En el informe HTML, se presenta un diseño novedoso para la visualización de los datos. Esto permite una revisión más rápida y eficaz por parte de los investigadores.

Comparando las diferentes herramientas analizadas con la que se ha desarrollado, en base a las características anteriores, se obtiene el resultado reflejado en la :

Tabla 4 - Comparación de las herramientas

Característica	iBAFT	Oxygen Forensic	Iphone Analyzer
Gratuita	✓	×	✓
Open Source	✓	×	×
Multiplataforma	✓	×	✓
Portable	✓	×	×
Flexible	✓	×	×
Informes en múltiples formatos	✓	✓	×
Diseño novedoso	✓	×	×

Así pues, se observa que aunque Oxygen FOrensic es una herramienta forense muy potente, no cumple con las primeras características, ya que no es gratuita ni Open Source, además está desarrollada para sistemas Windows y requiere instalación, por lo que no es portable. Aunque hace unos análisis muy profundos, no incluye la posibilidad de extensión de funcionalidades a través de plugins desarrollados por los investigadores. Y aunque presenta informes en diferentes formatos, el informe de timeline se limita a mostrar los datos en una tabla, por lo que no es muy amigable para la investigación, aunque sí para la presentación de resultados.

Sin embargo, la herramienta iPhone Analyzer es una herramienta gratuita y multiplataforma, ya que está desarrollada en java. Aunque se podría obtener el código fuente con un decompilador de .jar, no se facilita el código fuente directamente, por lo que no se considera Open Source. El gran problema de esta herramienta es la utilidad forense, ya que sólo parsea el archivo Manifest.mbdb del backup y muestra la información de los archivos que contiene en pantalla, pero no genera informes, ni permite filtrar resultados.

4. IDENTIFICACIÓN DE REQUISITOS

Para que la herramienta sea de ayuda para los investigadores forenses, debe cumplir una serie de requisitos mínimos, entre los que destacan los siguientes:

- Debe ser flexible mediante el uso de plugins
- Debe poder recopilar información acerca de las comunicaciones nativas del dispositivo, como es el caso de llamadas de voz o mensajes de texto. En el caso de la mensajería instantánea, los plugins juegan un papel fundamental, ya que cada cierto tiempo salen al mercado nuevas aplicaciones de mensajería, que pueden ser útiles en una investigación, pero que no se pueden cubrir por completo en un momento determinado.
- Debe poder recoger información acerca de la navegación web del dispositivo
- Debe poder obtener información relacionada con la lista de contactos
- Debe poder restaurar los ficheros con sus nombres originales
- Debe presentar un timeline en el que se visualice en una línea temporal las acciones realizadas en el dispositivo.

El primer requisito es obvio, ya que el uso de los dispositivos cada vez es mayor y cada día salen al mercado nuevas aplicaciones que almacenan sus propios datos, que podrían llegar a ser de utilidad en una investigación. Por ello, se ha desarrollado la herramienta pensando en la escalabilidad desde el principio. Todas las funcionalidades de extracción y transformación de los datos las realizan los plugins.

El segundo requisito también es de gran importancia, ya que sabiendo las comunicaciones establecidas por el dispositivo, generalmente comunicaciones con otras personas, se puede centrar la línea de investigación de determinados hechos.

La manera que utilizamos para comunicarnos cambia cada cierto tiempo. Primero, las comunicaciones de los dispositivos móviles se realizaban mediante llamadas de voz. Después se extendió el uso de los SMS y MMS, y ahora, con el creciente uso de las conexiones 3G, un gran porcentaje de las comunicaciones se realizan a través de conexiones de datos, utilizando distintas aplicaciones de mensajería instantánea. Es por eso importante resaltar que no se tiene la certeza de que todas las comunicaciones analizadas por la herramienta sean el total de las comunicaciones realizadas por el terminal, ya que puede existir una aplicación de mensajería instantánea o similar, de la que no se disponga un plugin para analizar sus datos.

Otro fuente importante de información la proporciona el navegador web. El historial de navegación muestra las búsquedas y visitas a diferentes sitios web que se han accedido desde el dispositivo móvil.

En muchas ocasiones se mostrará información, normalmente de comunicaciones, en la que aparezca un nombre, en lugar de un número de teléfono. Esto es debido a que ese número está guardado en la agenda de contactos del dispositivo, por lo que se necesita recuperarla también, sobre todo para poder obtener los números de teléfono que pertenecen a dichas comunicaciones.

Como ya se ha mencionado en el Capítulo 2 (Estructura de un Backup de iOS), un backup de iOS se conforma de múltiples archivos sin extensión. Debido a la incomodidad que produce trabajar directamente con estos nombres de archivos, desconociendo a primera vista el tipo de archivo de cada fichero, es un requisito fundamental que la herramienta pueda restaurar una copia legible del backup en un directorio destino. Esta copia restaurada puede servir al investigador para profundizar más en el análisis, buscando ficheros multimedia, como imágenes y vídeos, o analizando otros almacenes de datos, como bases de datos sqlite de otras aplicaciones que no analiza la herramienta (porque aún no se dispone de un plugin).

La herramienta pretende ser un punto de partida en un análisis forense de un dispositivo iOS, generando un informe de la información básica. Si la herramienta no dispone de los plugins de cada aplicación que se quiere analizar, e incluso alguno desarrollado a medida para el caso, será necesario una investigación posterior por parte del analista, en caso de no haber encontrado la evidencia que se buscaba.

En ningún caso la herramienta pretende sustituir la experiencia y el buen *ojo clínico* del perito, que será el que tenga que decidir en qué aspectos puede serle útil el software y en cuáles debe analizar los datos manualmente.

No hay que olvidar que las herramientas forenses ayudan a automatizar tareas repetitivas o complicadas y a mostrar la información de una manera más amigable, pero es siempre el investigador el que debe conocer el funcionamiento interno de estas herramientas y las posibles fuentes de información relevante para una investigación forense, que quizás la herramienta no contemple.

Las especificaciones descritas anteriormente son las especificaciones técnicas de la herramienta, pero además de éstas, debe cumplir también con los requisitos necesarios para que se cumplan las contribuciones que la herramienta debe aportar. Estas contribuciones son las siguientes:

- Debe ser una herramienta gratuita, por lo que no se licenciará como software propietario ni se cobrará por su uso.
- Debe ser una herramienta Open Source, por lo que no se limitará el acceso al código fuente, ni se ofuscará el mismo, y cualquier persona es libre de modificar dicho código con cualquier fin.
- Debe ser una herramienta Multiplataforma. Esta característica se consigue a través del lenguaje de programación, que en este caso es Python, el cual es un lenguaje multiplataforma, ya que no depende del sistema operativo anfitrión.
- Debe ser una herramienta Portable. Esto se consigue ya que es muy ligera y no necesita instalación. Puede ser fácilmente transportable en cualquier USB.
- Debe ser una herramienta Flexible, ya que debe permitir el desarrollo e implantación de nuevos plugins sin tener que realizar modificaciones en la herramienta. Simplemente basta con añadir los nuevos plugins a la carpeta correspondiente.
- Debe presentar dos informes (HTML y PDF). Uno está orientado a la visualización de los datos, para ayudar a obtener una conclusión, y otro para exponer los datos en formato impreso.
- En el informe HTML, se presentará un diseño novedoso para la visualización de los datos, que permitirá una revisión más rápida y eficaz por parte de los investigadores.

5. ARQUITECTURA Y DISEÑO DE LA HERRAMIENTA

La herramienta se divide en dos grandes partes. Una parte es la encargada de recoger toda la información pertinente a través de plugins y almacenarla en una base de datos sqlite. La otra parte se encarga de leer esa base de datos y generar los informes correspondientes.

Al separar estas dos partes se consigue que se pueda obtener la base de datos con la información necesaria, y que ésta sea guardada para generar distintos informes posteriormente.

5.1 Detalles de la Herramienta

Hasta ahora, se ha hecho referencia al software desarrollado como *la herramienta*, pero se ha decidido bautizar esta herramienta con el nombre de **iBAFT** (*iOS Backup Analysis Forensic Tool*, o *Herramienta Forense para el Análisis de Backups de iOS*).

iBAFT es una herramienta en línea de comandos desarrollada en Python, ya que se quiere que sea multiplataforma, y funcione en diferentes Sistemas Operativos. La razón de que sea una herramienta en línea de comandos, es porque suelen ser más ligeras y eficientes. Además, al investigador que haga uso de la herramienta se le supone un dominio de las diferentes consolas de los sistemas operativos.

Así pues, las dos partes mencionadas en el apartado anterior, reciben el nombre de *ibaft.py* y *ibaft_report.py*, respectivamente. Como su propio nombre indica, *ibaft.py* es el núcleo de la herramienta, que se encarga de obtener y procesar la información de la copia de seguridad, mientras que *ibaft_report.py* utiliza la información ya procesada, y guardada en su propia base de datos, para generar informes en base a los criterios del usuario.

La forma en la que se relacionan los dos componentes se puede ver en el siguiente diagrama de flujo de datos, en el que se detallan los dos componentes y su interacción con los elementos externos.

Se observa que en el punto de inicio de llama al script *ibaft.py* que genera la base de datos. Posteriormente, cuando se llama a *ibaft_report.py,* se lee la base de datos creada, y se crea una estructura de directorios en el destino indicado, que contiene los informes.

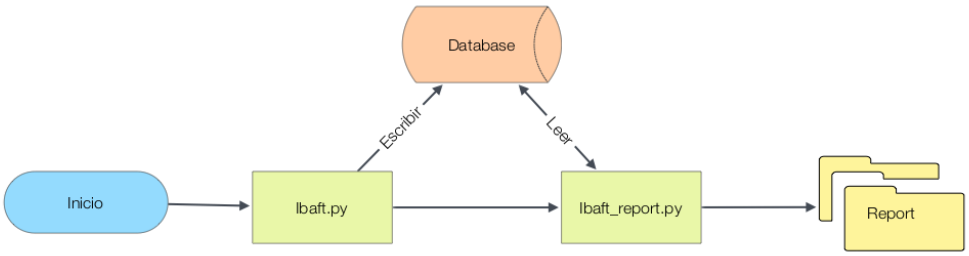


Figura 5 - Diagrama de flujo de datos de iBAFT

Los dos procedimientos, ibaft.py y ibaft_report.py, tienen a su vez sus diagramas de flujo asociados, que están representados en la Figura 6 y en la Figura 7, respectivamente.

5.2 Componente de recolección de evidencias

El componente *ibaft.py* es el encargado de la recolección de información del dispositivo para su posterior almacenamiento en la base de datos. Esta recolección de información se realiza a través de los plugins de la herramienta. Actualmente, dispone de plugins para la obtención de evidencias relacionadas con:

- Agenda de Contactos
- Calendario y Eventos
- Notas
- Llamadas de voz
- Mensajes de texto (SMS, MMS, iMessage)
- Historial de navegación (Safari)
- Conversaciones de mensajería instantánea (WhatsApp)

Esta parte de la herramienta necesita unos datos mínimos de entrada. Estos datos son el directorio donde se encuentra el backup de iOS, y un directorio destino, donde se creará la base de datos y el backup reconstruido, en caso de que se seleccione dicha opción (Las opciones de la herramienta se explican en el apartado 5.5).

Para comprender mejor el funcionamiento de este componente, se puede ver su diagrama de flujo de datos en la Figura 6, donde se observa que antes de cargar los plugins, que son el núcleo de la adquisición de evidencias, se leen dos archivos importantes, Info.plist y Manifest.mbdb, de los que ya se ha hablado en el capítulo 2.

Estos archivos deben existir en el directorio que se indica como origen del backup de iOS. El script *ibaft.py* busca en primer el archivo Info.plist, y después el archivo Manifest.mbdb. Si no se encuentran, la ejecución terminará, produciendo un código de retorno 4 o 5 (Los códigos de retorno se detallan más adelante, más concretamente en la sección 5.4).

Si los archivos se encuentran en el directorio, la herramienta parseará primero en archivo Info.plist, extrayendo su información, de la que obtendrá la opción denominada **Device Name** cuyo valor será el nombre de la base de datos a crear, con extensión sqlite. Si se produjese algún error leyendo este dato, o su contenido fuese nulo, el nombre de la base de datos será la palabra ios_ concatenada con una marca temporal en formato YYYYmmddHMMSS.

Una vez está la base de datos creada, se guardará en ella toda la información obtenida del archivo Info.plist en la tabla options en parejas clave/valor. Si alguna opción no se pudiese representar de manera nativa como un String, como por ejemplo un diccionario o un array, se codificará en formato JSON.

Después de almacenar la información del dispositivo en la tabla options, se lee el fichero Manifest.mbdb y se guarda su contenido descifrado en la tabla manifest (La estructura de las tablas se muestra con más detalle en el Capítulo 6).

Una vez se dispone de la información descifrada del archivo Manifest.mbdb en la base de datos, se puede proceder a obtener la información sin tener que restaurar todo el backup, simplemente haciendo una consulta a la tabla para obtener el nombre cifrado del archivo deseado.

Claro está que si el usuario quiere restaurar el contenido del backup puede hacerlo, indicando la opción -b, como se observa en el primer nodo condicional del diagrama de la Figura 6.

Una vez se ha cargado la información de Manifest.mbdb y se ha restaurado el backup, si así se ha decidido, se procede a cargar los plugins. Para ello, se inicializan todos los archivos con extensión *.py* del directorio plugins y se llama a su método analyze(), que es el que se encarga de procesar y almacenar la información correspondiente (Se dispone de más información sobre el sistema de plugins en el Capítulo 7).

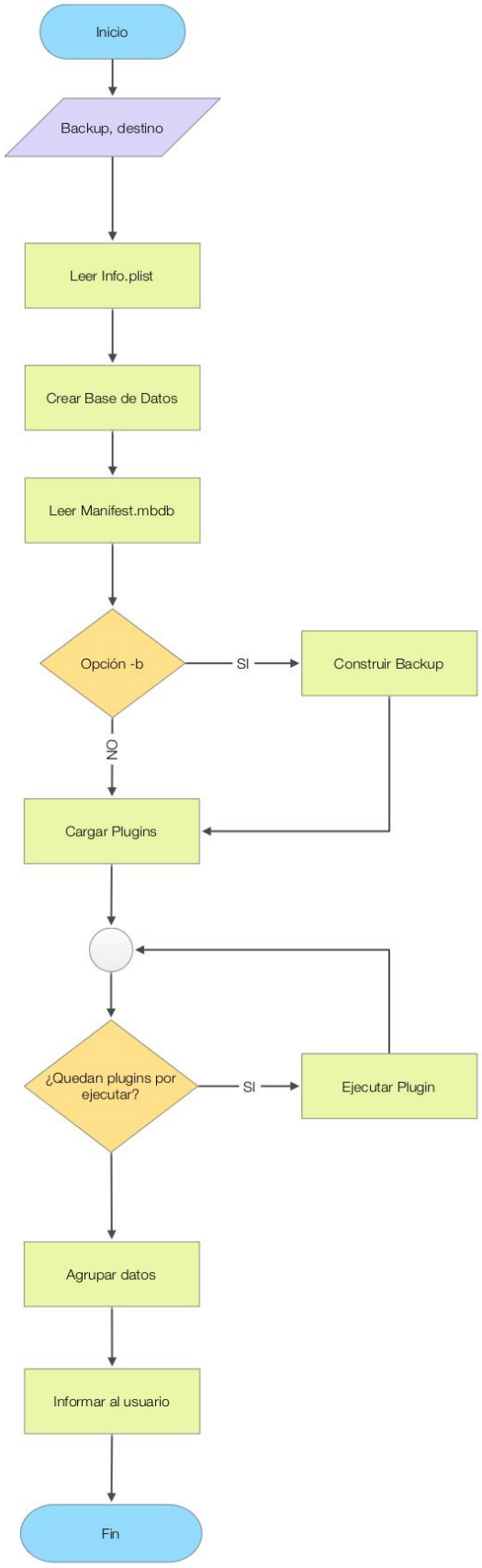


Figura 6 - Diagrama de Flujo de Datos de ibaft.py

Cuando los plugins han terminado sus funciones, se agrupan los datos en una tabla output con unos campos comunes a la mayoría de datos obtenidos, con el fin de estandarizar el acceso a la información y hacerla más rápida y eficiente. Esto puede incrementar el tiempo de ejecución de este componente de la herramienta, pero reducirá el tiempo de ejecución de cada generación de informe, por lo que parece una buena alternativa, ya que este script sólo se debería lanzar una vez por cada caso, y el script *ibaft_report.py* podrá lanzarse varias veces, según los criterios de búsqueda.

Una vez ha terminado de recopilar la información en la tabla output, se informa al usuario de la correcta finalización de este componente, indicando también la ruta donde se ha creado la base de datos y el siguiente paso a realizar, si se quiere obtener un informe.

5.3 Componente de informe de evidencias

El script *ibaft_report.py* es el encargado de generar los informes en base a los criterios indicados. Para ello lee la base de datos generada por el script *ibaft.py*, más concretamente su tabla output, que contiene toda la información recogida en un formato común.

Este componente puede recibir varios filtros, entre ellos:

- No filtrar: Obtiene todos los datos analizados. No se aplican filtros
- Filtrar con una fecha de inicio: Se le puede indicar una fecha a partir de la cual generar las evidencias
- Filtrar por un fecha de fin: Se le puede indicar una fecha límite hasta la cual generar las evidencias
- Filtrar por categorías: Se puede filtrar, indicando el tipo de información que se quiere obtener. Se disponen de varios tipos, ampliables también a través de plugins, de los cuales, los más básicos son:
    - Llamada
    - SMS
    - Nota
    - Contacto
    - WhatsApp
    - Evento
    - Historial
- Filtrar por palabras clave: También se puede indicar una o varias palabras clave que se quieren buscar, ya que esto es muy habitual en las investigaciones forenses.

Si se decide aplicar un filtro, todos los filtros mencionados anteriormente se pueden combinar libremente. Si no se decide aplicar un filtro, se debe indicar explícitamente que se quiere obtener todos los datos, ya que la herramienta necesita un criterio de búsqueda, aunque éste sea un indicador para no aplicar filtros.

El diagrama de flujo de datos de este componente se puede ver en la Figura 7. Como se puede apreciar en el diagrama, realmente esta parte de la herramienta sólo lee la base de datos generada anteriormente, carga el módulo de reporting, y establece unos datos en función de los criterios indicados. Cuando ya se han comprobado todos los criterios y se tiene un conjunto de datos definitivos sobre los que generar el informe, se crea primero la estructura de carpetas que contendrá el informe HTML, para después generar en la raiz del directorio que contiene el informe HTML, un archivo llamado *reportPDF.pdf,* que se corresponde con la versión en PDF del informe generado.

A pesar de que los detalles del informe se detallan en capítulos posteriores, es importante mencionar que la calidad de los diferentes informes es significativamente distinta. Obviamente, el informe HTML está mucho más orientado a la presentación, y a la interacción con el usuario, que puede incluso filtrar resultados sobre los datos del informe. Sin embargo, el informe PDF está más orientado a poder imprimir los datos de una manera más compacta. Es por eso, que el informe en PDF no dispone de una maquetación muy trabajada, ni de detalles visuales. Simplemente representa un conjunto de tablas que modelan diferentes tipos de datos, y que muestran en cada una de sus filas un registro con los campos considerados relevantes.

Durante la investigación forense, a menudo será habitual generar varios informes para poder esclarecer los hechos investigados. Por ejemplo, puede ser posible que se empiece a investigar las conversaciones de WhatsApp en busca de algún indicio en concreto, pero que una vez se encuentre, o no, se tenga que optar por otras líneas de investigación, que aborden otro tipo de datos, o incluso otro tipo de palabras clave.

Una vez se conocen las evidencias que demuestran los hechos investigados, se podrá generar un informe más completo, ya sea en HTML o en PDF, que incluya todas las evidencias relevantes para el caso. Además, si el investigador considera que el formato del informe de evidencias generadas no se ajusta a sus necesidades, siempre podrá trabajar directamente sobre la base de datos sqlite, para tratar con los datos *en crudo* y poder construir el formato que considere apropiado.

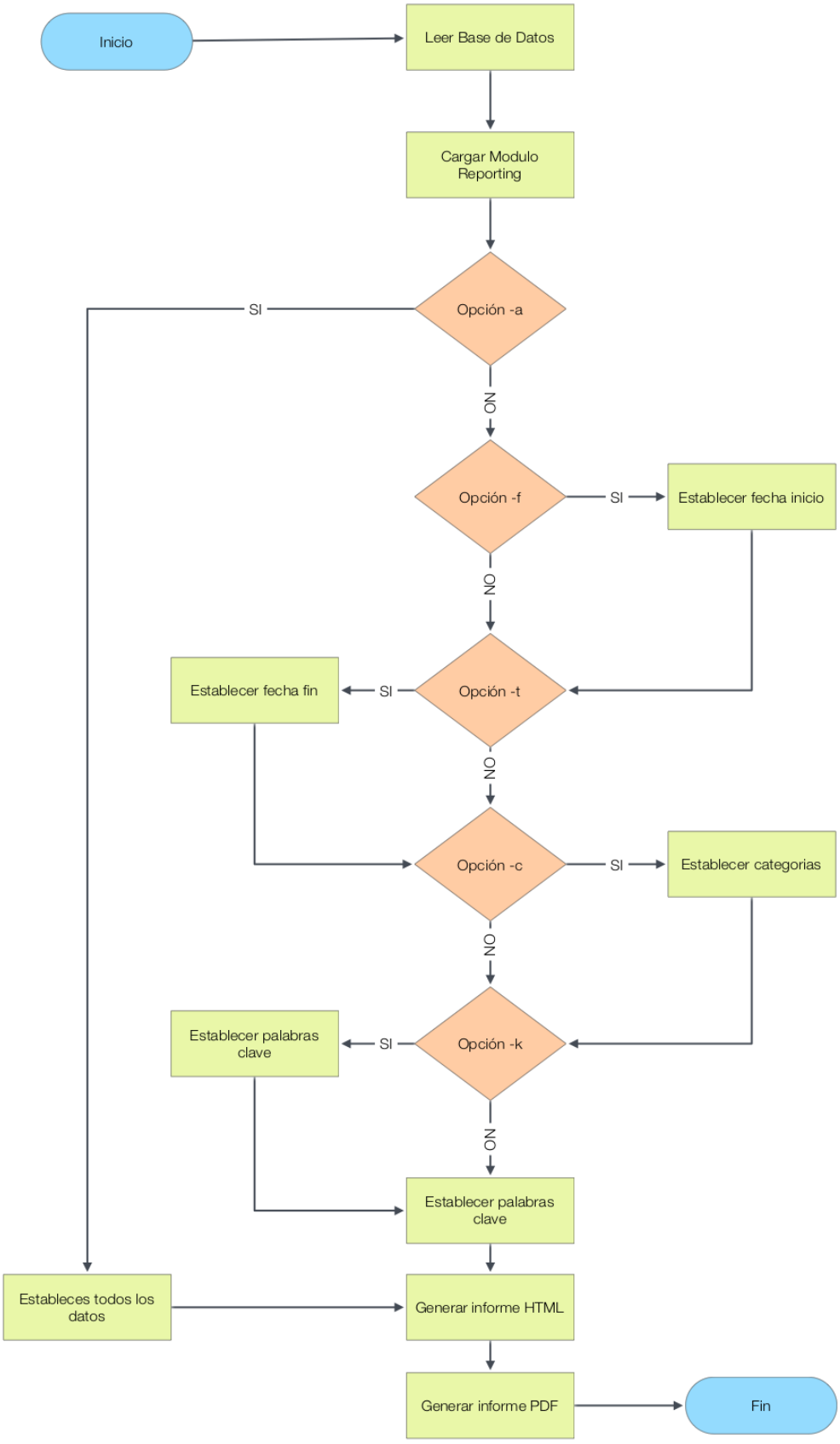


Figura 7 - Diagrama de Flujo de Datos de ibaft_report.py

5.4 Códigos de retorno de la aplicación

Cuando se ejecuta un comando en una shell (o consola), siempre se devuelve un código de salida, que suele ser 0 si la ejecución ha sido correcta, o cualquier otro número que indica un error determinado.

Es una buena práctica incluir diferentes códigos de salida en las aplicaciones para poder determinar si se ha producido un error en la ejecución, y poder depurar dicho error. Aunque muchas herramientas, incluido esta, ya informan al usuario del error que se ha producido, es conveniente que se indique un código de salida, para que lo pueda interpretar correctamente el sistema operativo, o incluso otras herramientas o scripts que utilicen por debajo la aplicación.

También es una buena práctica el comprobar el código de retorno de cada comando ejecutado, sobre todo los críticos, para cerciorarse de que se ha ejecutado con éxito. Por ejemplo, en Linux, se puede comprobar el código de retorno del último comando ejecutado a través de la variable **$?** que contendrá dicho código. De esto modo se pueden hacer comprobaciones automatizadas para validar si un comando se ha ejecutado correctamente o no.

La herramienta iBAFT también produce distintos códigos de retornos, pero pertenecientes a cada componente. De modo que el script *ibaft.py* producirá una series de códigos de retorno, y el script *ibaft_report.py* producirá otros distintos. Dichos códigos de retorno se especifican en la Tabla 5 y en la Tabla 6, respectivamente.

Tabla 5 - Códigos de retorno de ibaft.py

Código de retorno	Descripción
1	No se han recibido argumentos
2	Algún argumento u opción son incorrectos
3	No se ha indicado el directorio del backup o el directorio de destino
4	No se ha encontrado el fichero Info.plist en el directorio de backup
5	No se ha encontrado el fichero Manifest.mbdb en el directorio de backup
6	No se ha podido recuperar el nombre del dispositivo del fichero Info.plist

Como se puede observar, la mayoría de los códigos de error se corresponden con errores en el paso de argumentos a la herramienta. Lo mismo ocurre con los códigos de retorno de *ibaft_report.py*.

Tabla 6 - Códigos de retorno de ibaft_report.py

Código de retorno	Descripción
1	No se han recibido argumentos
2	Algún argumento u opción son incorrectos
3	No se ha indicado la base de datos que se quiere analizar
4	No se ha indicado ningún filtro, ni la opción de no filtrar
5	No se ha indicado el directorio donde guardar el informe

Gracias a estos códigos de retorno, se podrá incorporar la herramienta al flujo de ejecución de otro software, y facilitará un método para comprobar el éxito o fallo de la ejecución del programa.

5.5 Opciones y argumentos de la herramienta

A lo largo de este capítulo se han descrito las diferentes funcionalidades que tiene cada parte de la herramienta, pero aún no se han descrito el nombre de las opciones que ejecutan esas funcionalidades. Por ello, en este apartado se especifican las diferentes opciones que soporta cada parte de la aplicación.

Aunque estas opciones están bien documentadas en los mensajes de ayuda de cada componente, la descripción de cada opción se describe en la Tabla 7 y en la Tabla 8.

Tabla 7 - Opciones de ibaft.py

Opción	Opción larga	Descripción
-s	--source-dir	Indica el directorio en el que se encuentra el backup a analizar.
-d	--destination-dir	Indica el directorio donde se guardará la base de datos maestra, y los archivos recuperados, en caso de haber indicado la opción -b.
-b	--build-backup	Si se indica, se copiaran los ficheros recuperados al directorio destino.
-h	--help	Muestra el texto de ayuda

Como se aprecia en la tabla, la herramienta soporta dos tipos de opciones, al estilo Unix, precedidas por un guión (quizás las más cómodas y habituales), y al estilo GNU, precedidas por un doble guión, que suelen ser más largas, pero también más explicativas. Lo mismo ocurre con las opciones de *ibaft_report.py*.

Tabla 8 - Opciones de ibaft_report.py

Opción	Opción larga	Descripción
-d	--database	Indica la base de datos que utiliza para leer los datos. Tiene que ser un archivo sqlite generado con la herramienta ibaft.py
-f	--from	Indica la fecha desde la que se buscarán los datos. Debe estar en formato [YYYY-mm-dd HH:MM:SS]
-t	--to	Indica la fecha hasta la que se buscarán los datos. Debe estar en formato [YYYY-mm-dd HH:MM:SS]
-k	--keywords	Indica las palabras clave que buscar
-c	--categories	Indica la lista de categorias en las que buscar, separadas por comas.
-a	--all	Indica que se quiere obtener toda la información
-o	--output-dir	Indica el directorio donde se guardará el informe
-h	--help	Muestra el texto de ayuda

Aunque se hayan detallado las opciones, también hay que explicar la sintaxis de cada componente, aunque realmente su uso es sencillo e intuitivo. De hecho, si se ejecuta el script sin opciones ni argumentos, además de obtener el error asociado al código de retorno 1, se mostrará un mensaje con el uso básico de la herramienta, en la que se incluye su sintaxis. Los corchetes indican que son argumentos opcionales.

La sintaxis de *ibaft.py* es:

```
ibaft.py -s backup_dir -d destination_dir [-h] [-b]
```

La sintaxis de *ibaft_report.py* es:

```
ibaft_report.py -d database -o output_dir (-a | [-f date][-t date][-k keywords][-c categories])
```

La sintaxis del script ibaft_report.py puede parecer un poco más compleja debido a los filtros, pero realmente sólo es necesario indicar la base de datos, el directorio destino y un filtro, que puede ser o bien el filtro ALL, con la opción -a, o cualquiera de los filtros preestablecidos.

6. ESQUEMA DE LA BASE DE DATOS

La base de datos sqlite que utiliza la herramienta se construye con un esquema predefinido en un script SQL dentro de la propia herramienta. Este script se puede encontrar en la carpeta *db* de la herramienta, bajo el nombre de *scheme.sql* cuya estructura de base de datos es la que se representa en la siguiente imagen:

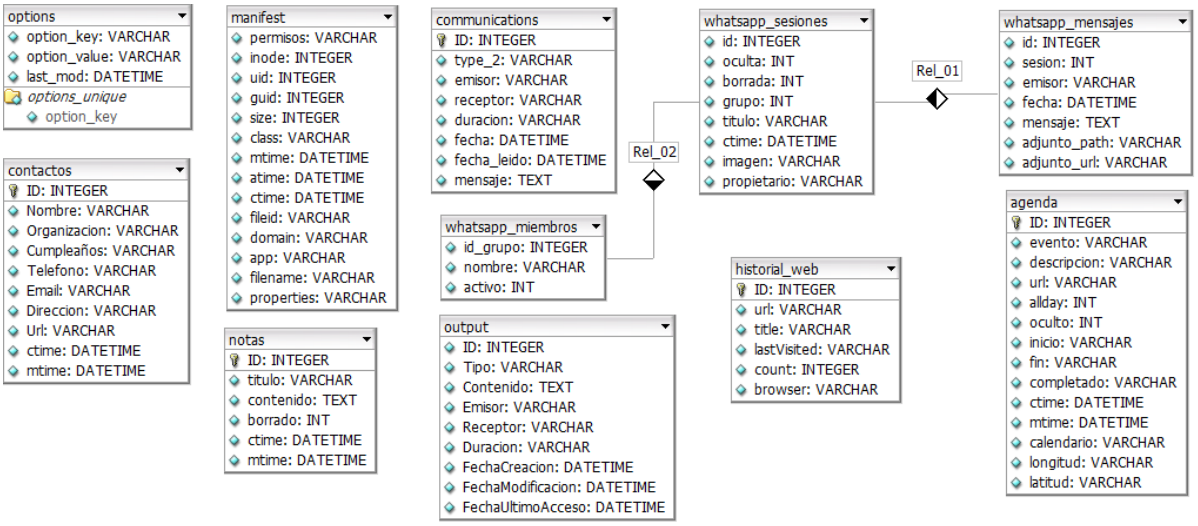


Figura 8 - Esquema de la base de datos de la herramienta

Como se puede observar en la Figura 8, apenas existen relaciones entre las tablas, con la excepción de las tablas de conversaciones de WhatsApp, que por su naturaleza requieren varias tablas para modelar la información. El resto de tablas actúan como almacén de datos aislado, para después volcar toda la información de cada tabla en el formato de la tabla output, que será a partir de la cual generar el informe.

El detalle de cada tabla, y el cómo se recoge la información que debe almacenar en ella, se trata en el próximo capítulo, mientras se habla de la labor que realiza cada plugin.

7. SISTEMA DE PLUGINS

Uno de los requisitos de la herramienta es la implementación de un sistema de plugins que permite la flexibilidad de la herramienta. Estos plugins permiten al usuario desarrollar sus propias funcionalidades para la herramienta, de modo que puede ir creciendo la cantidad de información que se analiza de manera automática.

Los plugins se encuentran en el directorio plugins de la herramienta y son archivos fuente de python, que se cargarán en el momento de la ejecución.

En un aspecto más técnico, los plugins son definiciones de la clase Plugin, que hereda de la clase PluginBase, definida en el modulo *plugin.py.* La clase padre tiene como propiedades una serie de campos que aportan información sobre el plugin, el directorio actual de trabajo y conexión a la base de datos sqlite, de forma que se crea una capa de abstracción, que evita que desarrollador del plugin tenga que conocer los detalles de conexión a la base de datos. El desarrollador sólo debe conocer que existe un atributo heredado que hace referencia a la conexión de la base de datos.

Esta clase padre (PluginBase) además de diferentes métodos para interactuar con la base de datos, crear una estructura de plugin vacía y otras utilidades, dispone de dos métodos imprescindibles para el correcto funcionamiento del sistema de plugins. El primero de ellos es el método *load_plugins()* que, como se puede deducir, se encarga de cargar los plugins, buscando cada archivo python del directorio plugins y agregándolo a una lista de plugins cargados. El segundo método es el método *analyze()*, que recorre la lista de plugins cargados, y ejecuta su método *analyze()*, de esta manera, en el flujo de ejecución del programa, sólo se llamará al método analyze de la clase padre, y este será el que dinámicamente ejecute cada plugin.

Cada archivo de plugin puede llamarse de una forma completamente arbitraria, aunque se recomienda asignarles un nombre descriptivo, pero en su interior debe haber una definición de la clase Plugin, que sobrescriba al menos el método analyze. El constructor de cada plugin recibe como parámetros, una instancia de la conexión a la base de datos, y el directorio del backup, aunque estos parámetros se le pasarán mediante el método analyze() de la clase padre de una forma completamente transparente para el plugin.

La mayoría de plugins que se vayan a desarrollar, lo que harán es buscar un archivo de base de datos sqlite en el dispositivo, y analizar la información que éste contiene, para introducirla en la base de datos maestra. Para facilitar la tarea, la estructura por defecto de un plugin (existe un plantilla para crear un plugin vacía) incluye como propiedades de la

clase una llamada *path_db*, que contiene la ruta a la base de datos que se quiere buscar dentro del dispositivo. Esta propiedad hay que establecerla a mano en el constructor, pero sólo con indicar la ruta, ya se dispondrá en el resto de la clase dos instancias de base de datos, la maestra donde se debe guardar la información y la del componente que analiza el plugin.

Normalmente el método analyze() del plugin se encargará de obtener la información de la base de datos que se ha abierto, realizar modificaciones si fuese necesario (normalmente en formatos de fecha) y almacenarla en la base de datos maestra.

Para no complicar un método con asignaciones de dos cursores a bases de datos y diferentes bucles anidados, se recomienda añadir métodos al plugin para facilitar la separación y reutilización de código. Al final, sólo se ejecutará el método analyze, per éste puede llamar a tantos métodos como se desee, por lo que no es aconsejable tener un único método monolítico que realice todas las acciones, sino la separación del código mediante funciones, lo que permite además un mejor mantenimiento del código.

Una vez explicado el sistema de plugins, sólo queda explicar el funcionamiento de cada uno de los plugins que trae la aplicación por defecto.

7.1 Plugin para el análisis de las llamadas

Uno de los plugins más básicos es el de llamadas de voz, debido a los pocos campos que se recogen y la sencillez de la base de datos en la que se encuentra la información.

La base de datos que contiene la información de las llamadas de voz se encuentra en el directorio Library/CallHistory, en una base de datos sqlite llamada call_history.db, que tiene la estructura que se aprecia en la siguiente imagen:

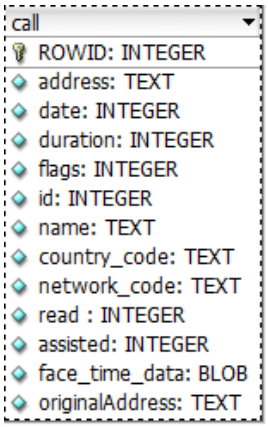


Figura 9 - Estructura de call_history.db

Se observa que la base de datos sólo tiene una tabla, llamada call, con unos cuantos campos, de los cuáles, los más importantes (y los que obtiene el plugin) son:

- address: Emisor de la llamada
- duration: Duración de la llamada
- date: Fecha en la que se produce la llamada

El plugin obtiene esta información y la inserta en la tabla communications con las siguientes peculiaridades:

- El campo type se establece como **Llamada**
- La fecha se convierte a un formato legible: YYYY-mm-dd HH:MM:SS

7.2 Plugin para el análisis de los mensajes de texto

El análisis de los mensajes de texto también se basa en una base de datos sqlite que contiene esta información. La base de datos en cuestión se encuentra en el directorio Library/SMS, bajo el nombre de sms.db, y que tiene la siguiente estructura:

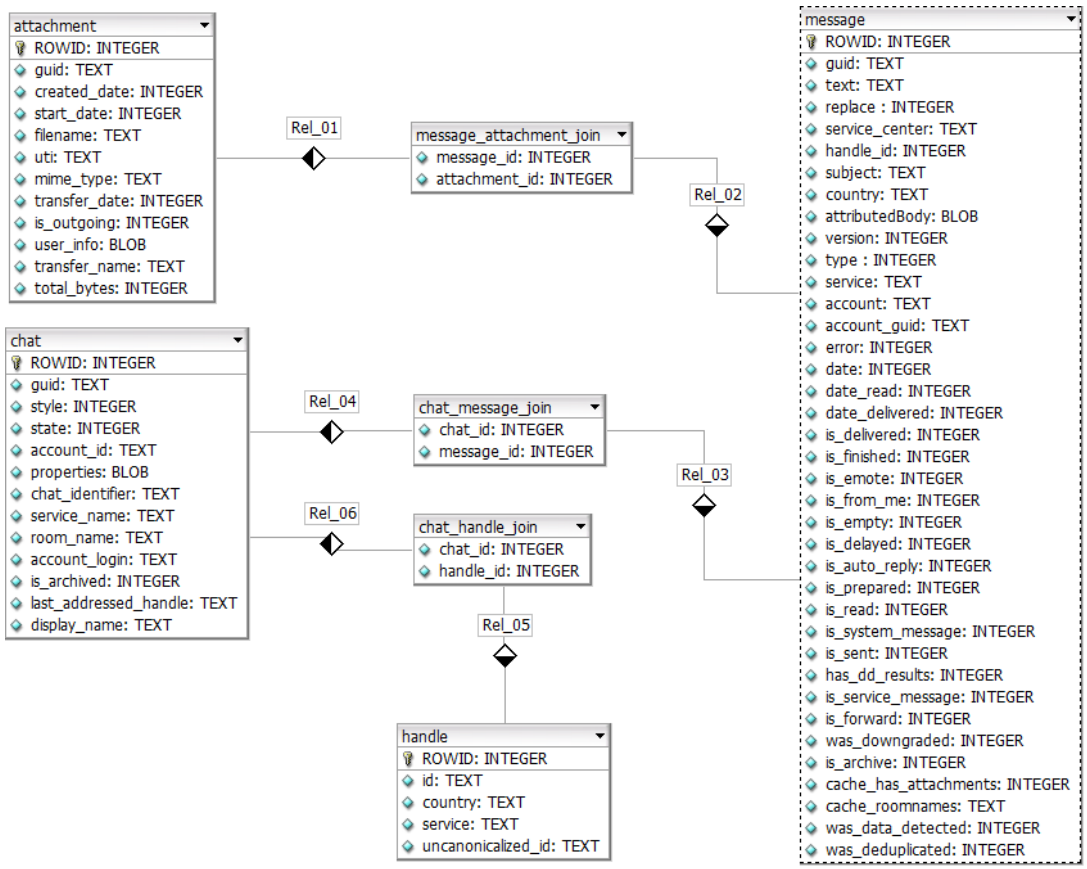


Figura 10 - Estructura de sms.db

Esta base de datos tiene una estructura más compleja que la de llamadas, pero realmente, el plugin no maneja todos estos datos, sino que sólo obtiene los siguientes datos:

- handle.id: Nombre o numero de teléfono del emisor
- message.text: Contenido del mensaje
- message.service: Distingue entre SMS e iMessage
- message.date: Fecha de emisión
- message.date_read: Fecha de lectura

Estos datos se insertan en la tabla communications con las siguientes peculiaridades:

- El campo type se establece como el valor del campo Service (SMS o iMessage)
- Las fechas se convierten a un formato legible: YYYY-mm-dd HH:MM:SS
- Se establece el valor de duración a 'N/A', ya que no existe un valor de duración para mensajes de texto.

7.3 Plugin para el análisis de los contactos

La estructura de la base de datos que contiene la información de los contactos es bastante más compleja que las que se han ido viendo en apartados anteriores, pero mucha información es redundante, por lo que sólo se trabajará sobre un pequeño conjunto de datos. La base de datos en cuestión se encuentra en el directorio Library/AddressBook y se llama AddressBook.sqlitedb. Su estructura se detalla en la Figura 11, y aunque puede parecer demasiado compleja, la información necesaria para un investigador reside en un pequeño conjunto de tablas. El resto de tablas se utilizan sobre todo para funcionalidades de la agenda de contactos en el dispositivo.

En concreto, y debido a que la tabla ABPersonFullTextSearch_content contiene información desnormalizada de la tabla ABPerson, es más fácil obtener dicha información de la primera, en lugar de la segunda. Se obtienen los siguientes campos relativos a un contacto:

- First: Nombre
- Middle: Segundo Nombre
- Last: Apellido(s)
- Organization: Organizacion del contacto
- Birthday: Fecha de nacimiento/cumpleaños
- Nickname: Apodo
- Phone: Teléfono del contacto
- Email: Correo electrónico del contacto

- Address: Dirección del contacto
- URL: Sitio web del contacto
- CreationDate: Fecha de creación del contacto
- ModificationDate: Fecha de modificación del contacto

En muchas ocasiones, la gran mayoría de estos campos serán nulos o estarán vacíos, ya que es habitual guardar un contacto en un dispositivo, que sólo tenga un nombre (aunque incluya apellidos) y un número de teléfono, dejando vacíos los campos de cumpleaños, organización, dirección y demás.

Generalmente lo que suele interesar es identificar un número de teléfono y un nombre para poder traducirlos en las comunicaciones que se encuentren en otros componentes. Así pues, si se encuentra por ejemplo una llamada de un contacto identificado por un nombre, se podrá buscar cuál es el número de teléfono que le corresponde en la agenda de contactos.

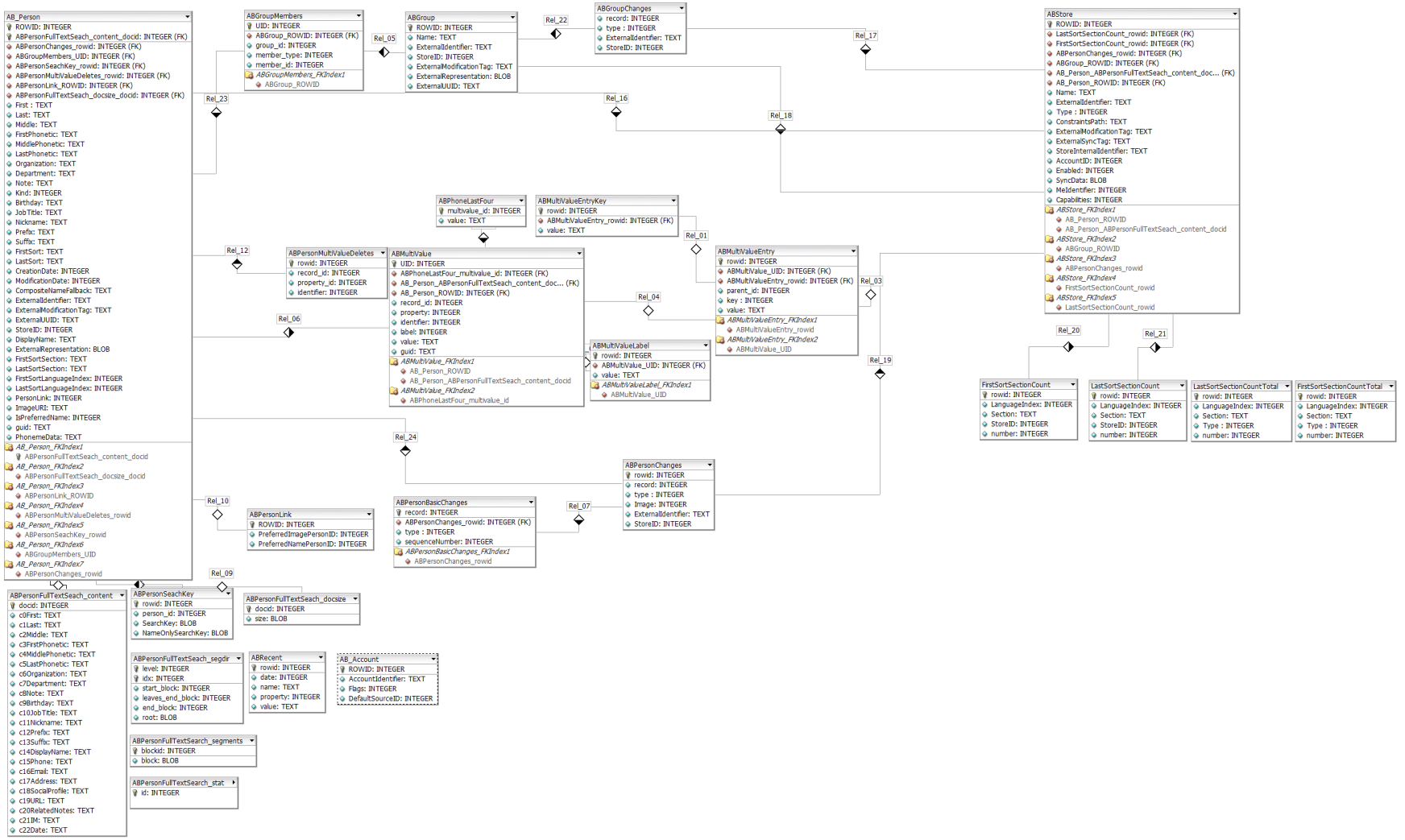


Figura 11 - Estructura de AddressBook.sqlitedb

7.4 Plugin para el análisis del historial web

Este plugin puede resultar un poco más complicado, pues la fuente de información no es una base de datos sqlite, sino un archivo binario plist. Realmente un archivo plist tiene una estructura XML, pero no puede ser leído con cualquier editor. Por ello, es necesario utilizar una librería externa para poder parsear los datos correctamente desde la aplicación.

El archivo en cuestión se encuentra en el directorio Library/Safari/ bajo el nombre de History.plist, y su estructura XML sigue el siguiente DTD:

```
<!ENTITY % plistObject "(array | data | date | dict | real | integer |
string | true | false )" >
<!ELEMENT plist %plistObject;>
<!ATTLIST plist version CDATA "1.0" >

<!-- Collections -->
<!ELEMENT array (%plistObject;)*>
<!ELEMENT dict (key, %plistObject;)*>
<!ELEMENT key (#PCDATA)>

<!--- Primitive types -->
<!ELEMENT string (#PCDATA)>
<!ELEMENT data (#PCDATA)> <!-- Contents interpreted as Base-64 encoded -->
<!ELEMENT date (#PCDATA)> <!-- Contents should conform to a subset of ISO
8601 (in particular, YYYY '-' MM '-' DD 'T' HH ':' MM ':' SS 'Z'.  Smaller
units may be omitted with a loss of precision) -->

<!-- Numerical primitives -->
<!ELEMENT true EMPTY>  <!-- Boolean constant true -->
<!ELEMENT false EMPTY> <!-- Boolean constant false -->
<!ELEMENT real (#PCDATA)> <!-- Contents should represent a floating point
number matching ("+" | "-")? d+ ("."d*)? ("E" ("+" | "-") d+)? where d is a
digit 0-9.  -->
<!ELEMENT integer (#PCDATA)> <!-- Contents should represent a (possibly
signed) integer number in base 10 -->
```

Este DTD simplemente define los tipos de datos y la estructura que utilizará el XML, que contiene un nodo dict, con propiedades clave valor, cuya propiedad WebHistoryDates contiene un array de nodos dict (diccionarios) que modelan a su vez cada visita a una URL. Cada uno de estos diccionarios tienen un aspecto similar al que se muestra en la Figura 12.

Debido a que no se trata de una base de datos sqlite, el constructor del plugin requiere más modificaciones de las habituales, entre ellas, hacer uso de la librería plutil para Python, que permite almacenar el contenido del archivo plist en una variable del plugin llamada data_plist, que será sobre la que trabaje el método analyze, para obtener la información pertinente.

```
<dict>
    <key></key>
    <string>http://www.unir.net</string>
    <key>D</key>
    <array>
        <integer>2</integer>
    </array>
    <key>lastVisitedDate</key>
    <string>413972176.358618</string>
    <key>title</key>
    <string>UNIR - Universidad Internacional de la Rioja</string>
    <key>visitCount</key>
    <integer>2</integer>
</dict>
```

Figura 12 - Formato de cada visita en el archivo History.plist

Las propiedades que se muestran en la imagen anterior son autoexplicativas, y aunque hay nodos que puedan tener más propiedades, las que aportan más valor al investigador son la URL visitada y la fecha de última visita.

7.5 Plugin para el análisis del Calendario y los Eventos

En este plugin ocurre algo similar al plugin de contactos. La gran cantidad de tablas y campos de éstas que contiene la base de datos puede parecer abrumadora, pero una vez más, la información relevante reside en un pequeño conjunto de tablas y campos, siendo el resto propias del sistema para garantizar un correcto funcionamiento. La base de datos se encuentra en el directorio Library/Calendar con el nombre de Calendar.sqlitedb y contiene la estructura que se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

En este caso, la información relevante se encuentra en las tablas Calendar y CalendarItem, de las cuales el plugin obtiene los siguientes campos:

- CalendarItem.summary: Resumen del evento
- CalendarItem.description: Descripción del evento
- CalendarItem.start_date: Fecha de inicio del evento
- CalendarItem.end_date: Fecha de fin del evento
- CalendarItem.all_day: Indica si el evento es durante todo el día
- Calendar.title: Nombre del calendario
- CalendarItem.url: URL guardada del evento
- CalendarItem.last_modified: Fecha de última modificación del evento
- CalendarItem.hidden: Indica si es un evento oculto
- CalendarItem.completion_date: Fecha de finalización del evento
- CalendarItem.creation_date: Fecha de creación del evento

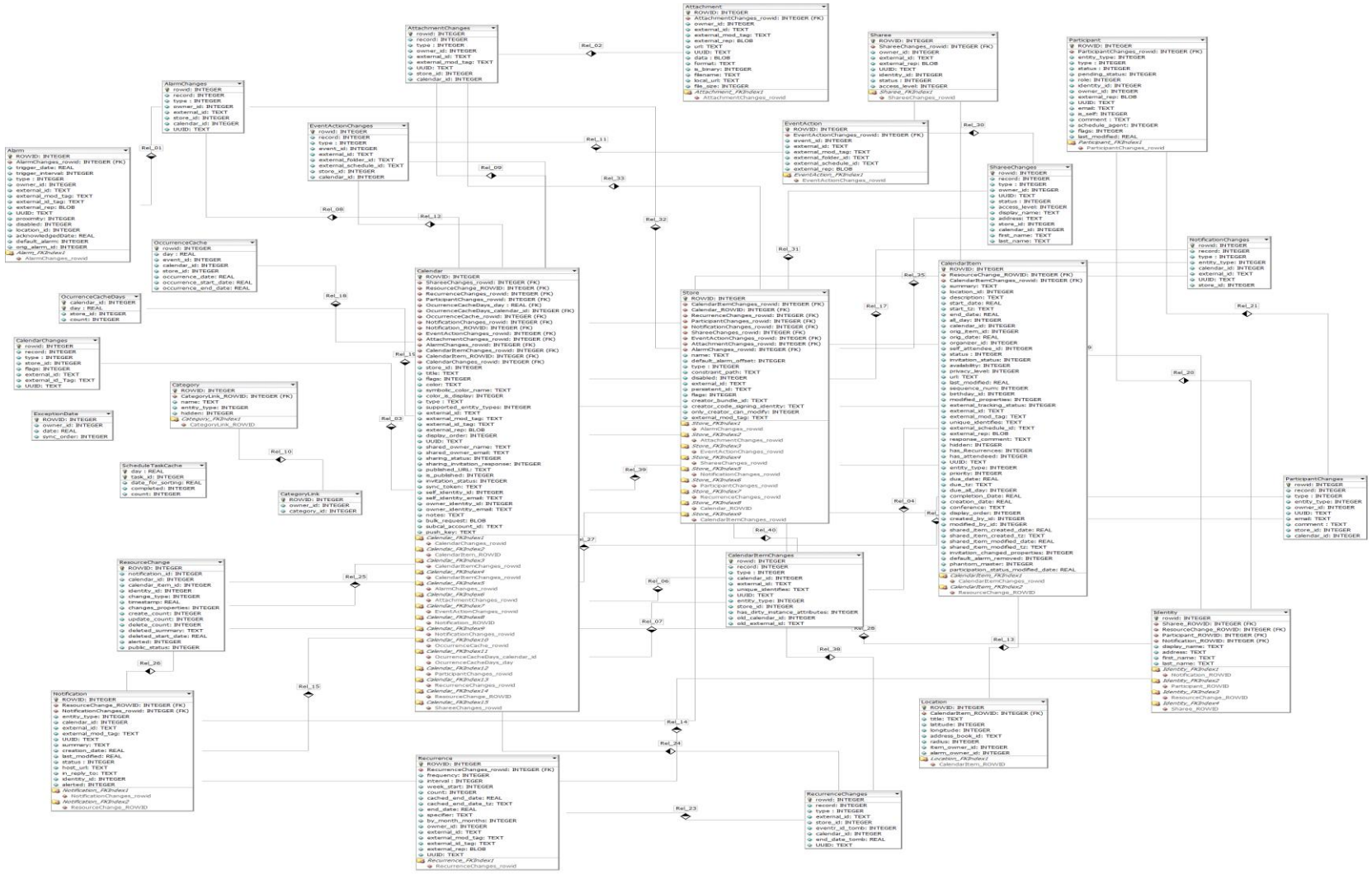


Figura 13 - Estructura de Calendar.sqlitedb

Los campos obtenidos se insertan en la tabla agenda con ciertas modificaciones:

- Las fechas se convierten a un formato legible: YYYY-mm-dd HH:MM:SS
- Se añaden campos sobre la longitud y latitud del evento, si están informados en la tabla Location

7.6 Plugin para el análisis de las Notas

Las notas se almacenan en Library/Notes, en un archivo llamado notes.sqlitedb. Esta base de datos tiene una estructura bastante simple, representada en la Figura 14, de la cual el plugin obtiene los siguientes campos:

- znote.ztitle: Titulo de la nota
- znotebody.zcontent: Contenido de la nota
- znote.zdeletedflag: Indica si la nota ha sido borrada
- znote.zcreationdate: Fecha de creación de la nota
- znote.zmodificationdate: Fecha de modificación de la nota

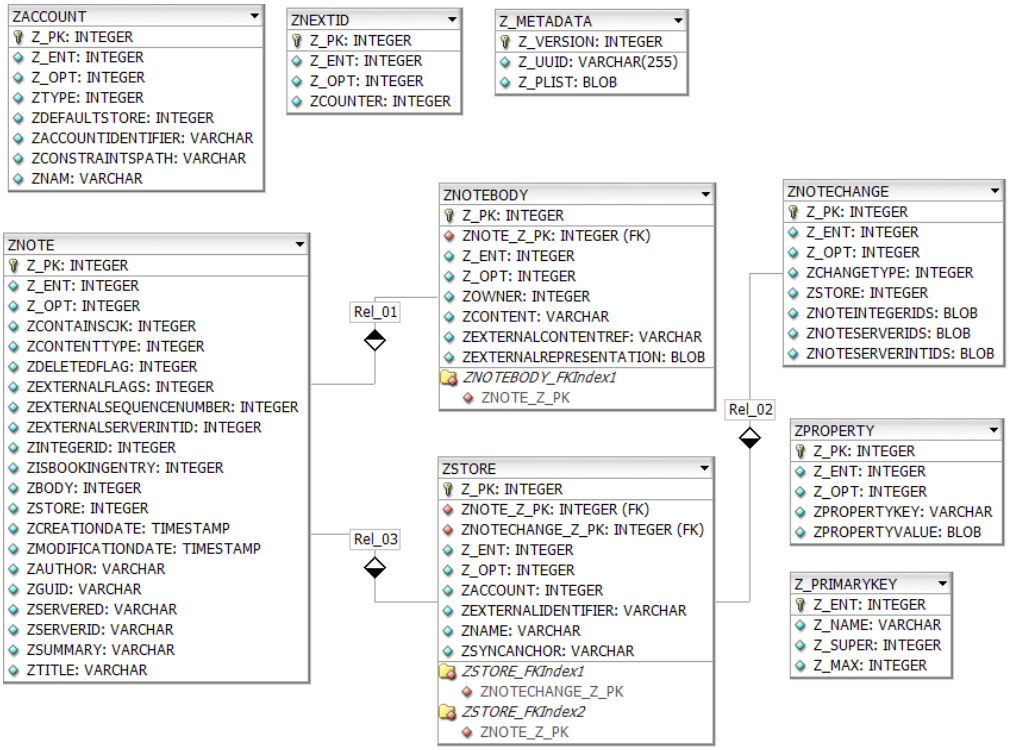


Figura 14 - Estructura de notes.sqlitedb

La información recogida por el plugin se inserta tal cual en la tabla notas, únicamente cambiando el formato de fecha a un formato más legible, de manera similar a los otros plugins.

7.7 Plugin para el análisis de las conversaciones de WhatsApp

Este plugin es sin duda uno de los más complejos de desarrollar, debido a la naturaleza de las conversaciones de mensajería instantánea, en la que pueden intervenir varios interlocutores, formar grupos, enviar adjuntos, etc. Es por ello que se necesitan tres tablas para modelar la información adquirida por el plugin. Sin embargo, la base de datos de WhatsApp, situada en Documents/ChatStorage.sqlite contiene más tablas, pero como en casos anteriores, sólo se necesita una pequeña parte de la información.

La estructura de la base de datos de WhatsApp no es demasiado compleja. De hecho, es bastante intuitiva, tal y como se puede ver en la Figura 15.

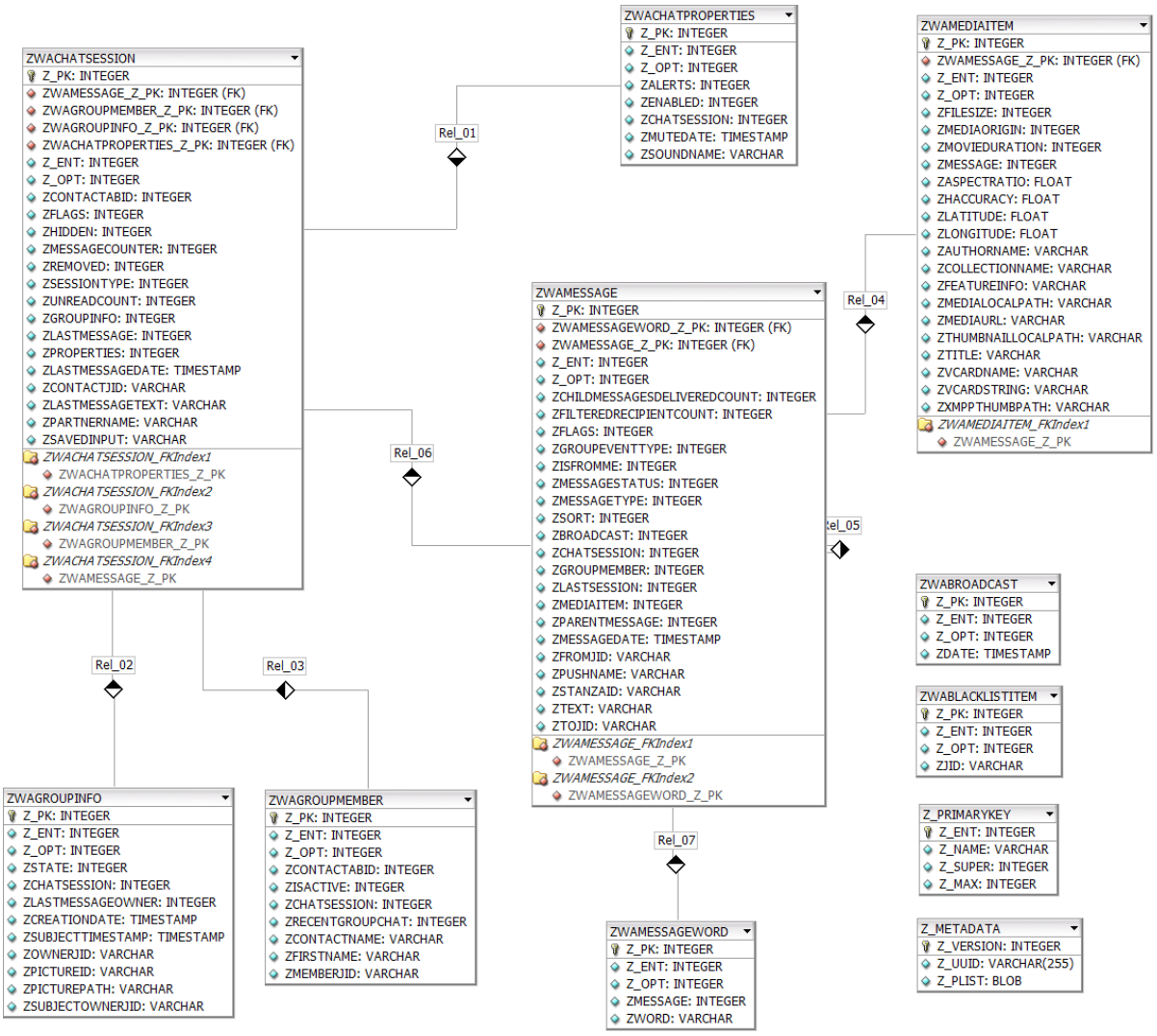


Figura 15 - Estructura de ChatStorage.sqlite (WhatsApp)

De todas las tablas representadas en la imagen anterior, los campos que obtiene el plugin son los siguientes:

- zwamessage.zchatsession: Sesión de chat
- zwamessage.zpushname: Nombre del emisor de un mensaje
- zwamessage.zmessagedate: Fecha de emisión del mensaje
- zwamediaitem.zmedialocalpath: Ruta local del archivo multimedia enviado
- zwamediaitem.zmediaurl: URL del archivo multimedia enviado
- zwamessage.ztext: Mensaje
- zwamessage.zisfromme: Indica si es un mensaje enviado por el propio dispositivo
- zwagroupmember.zcontactname: Nombre del contacto perteneciente a un grupo
- zwagroupmember.zisactive: Indica si un miembro de un grupo esta activo

- zwagroupinfo.zcreationdate: Fecha de creación del grupo
- zwagroupinfo.zpicturepath: Ruta local de la imagen de grupo
- zwachatsession.zhidden: Indica si una sesión está oculta
- zwachatsession.zremoved: Indica si una sesión de ha eliminado
- zwachatsession.zpartnername: Titulo de la sesión

Es necesaria cierta lógica para darle sentido a la información en bruto obtenida, por ejemplo para determinar el nombre de cada emisor del mensaje, o cuantos miembros tiene una conversación. Por eso, la información recogida pasa a tres tablas en la base de datos de la herramienta:

- whatsapp_miembros: Contiene información sobre los miembros de cada grupo
- whatsapp_sesiones: Contiene información acerca de las conversaciones
- whatsapp_mensajes: Contiene los mensajes

8. ESTUDIO DE DIFERENTES FORMATOS DE INFORME

Durante el desarrollo de la herramienta se tuvo en mente siempre generar dos tipos de informe, uno en HTML y otro en PDF, pero el modelo que se utilizaría en cada uno de ellos no se tenía tan claro. Por ese motivo, el modelo del informe HTML ha sufrido variaciones durante todo el desarrollo.

8.1 Formato de informe HTML

Al principio se pensó en generar un informe HTML interactivo que incluyese un cuadro de búsqueda avanzada, para poder filtrar los resultados, tal y como muestra la Figura 16

Figura 16 - Cuadro de búsqueda avanzada de un modelo de informe

De esta forma, el usuario podría filtrar por categorías, fechas o palabras clave dinámicamente, y obtener un subconjunto coincidente de los datos, sobre el conjunto global de datos que se manejaría.

Figura 17 - Busqueda por fecha en el informe HTML

Esta opción puede resultar muy atractiva, pero conforme se probaba la herramienta con datos reales, la usabilidad del informe descendía en picado, generalmente por el gran

número de registros que tenía que cargar (registros de la tabla manifest y whatsapp normalmente), lo que hacía que el informe tardarse demasiado tiempo en cargar.

El problema es obvio, el gran número de registros. Por lo tanto, había que intentar reducir cargar tantos datos si no iban a ser utilizados. El siguiente modelo se consideró quitar los registros de la tabla manifest, ya que no aportan mucho valor final, sino que más bien se utilizan durante la investigación para encontrar otro tipo de evidencias. Esta decisión se mantuvo hasta el final, pero el problema persistía.

Para intentar reducir el tiempo de carga se probó a generar informes independientes por categoría, lo que en general dio buen resultado, pero el problema seguía cuando se trataba de las conversaciones de WhatsApp. Y es que esta aplicación almacena en su base de datos un gran número de datos, sobre todo si es usada diariamente, como lo hace gran parte de sus usuarios.

Viendo que no se podría conseguir filtrar mediante el cuadro de búsqueda de la Figura 16 en el informe HTML, se decidió generar el informe HTML desde la herramienta en Python y filtrar según los parámetros indicados. Esta decisión es la razón por la que la herramienta se divide en dos scripts.

Sin embargo, y debido a que el problema del tiempo de carga, sucedía sólo con datos de conversaciones de WhatsApp, se pensó en poder filtrar el subconjunto de los demás tipos de datos, como los contactos, agenta, llamadas, etc. Aun así, hay una diferencia significativa con lo que se hacía antes.

Antes se obtenía la totalidad de la información y se filtraba en el *cliente* (informe HTML). Ahora se obteine un pequeño conjunto de información desde el *backend* (herramienta Python), y se ofrece la posibilidad de hacer a su vez búsquedas en la parte *cliente.*Esta funcionalidad está disponible sólo para los datos que se muestran en tablas, y es implementada a través del plugin para jQuery, DataTables[7].

DataTables es un plugin que permite realizar filtros, ordenar, paginar datos en formas de tablas. Además permite cargar los datos dinámicamente a través de AJAX desde diferentes tipos de archivos. Por ese motivo, se optó por generar distintos archivos JSON con los datos de contactos, calendario, historial, llamadas y mensajes. De esta forma, el informe podía gestionar información sin que el archivo HTML soportara el peso de los datos, por lo que era ligero en cargar, y se obtenían los datos sólo cuando el usuario lo solicitaba.

[7] https://datatables.net/

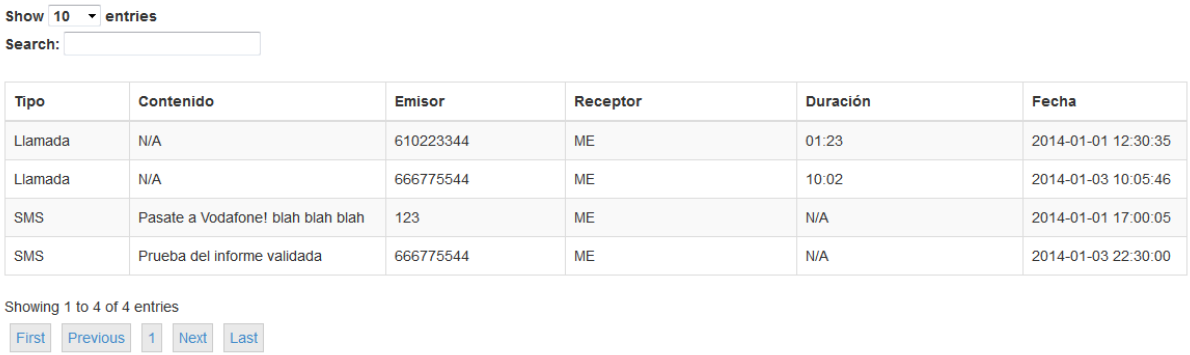


Figura 18 - Ejemplo de tabla con DataTables

La Figura 18 muestra un ejemplo de la tabla de comunicaciones utilizando DataTables. En el ejemplo sólo se tienen cuatro registros, pero debido a que DataTables soporta paginación, podría tener miles de registros que sólo cargaría la primera página, hasta que se solicitasen datos nuevos.

Además, se informa también del número total de registro encontrados, tanto al principio como filtrando por palabras. Permite también aumentar el número de registros que mostrará cada página de resultados, pudiendo examinar así más datos en menos tiempo.

Debido a que esta opción se consideró suficientemente elegante, se optó por mantenerla así, es decir, los datos separados en archivo JSON que se cargan desde AJAX de manera dinámica, directamente en el informe.

Sin embargo, aun no se ha hablado mucho de dos puntos fuertes del informe HTML, que son el **timeline** y las conversaciones de WhatsApp.

Aunque las conversaciones de Whatsapp también se podrían haber representado en forma de tablas con el plugin DataTables, esto habría sido muy pesado, sobre todo cuando se trata de conversaciones de más de dos personas. Por ese motivo se optó por aplicar un estilo CSS personalizado, que dote a esta parte del informe de una apariencia similar al que tendrían las conversaciones en un dispositivo real.

Por ello, los mesnajes de WhatsApp tienen du propia sección dentro del informe (separada de la sección de comunicaciones) y se visualizan dentro de un *globo* de conversación, que puede ser verde si el mensaje es enviado por el propio dispositivo, o gris, y con el nombre del emisor, si no es el propio dispositivo quien lo envía.

Figura 19 - Ejemplo de una conversación de WhatsApp en el informe HTML

En la Figura 19 se muestra un ejemplo de una conversación de grupo de la famosa aplicación de mensajería instantánea. Se observa que todos los mensajes recibidos tienen un nombre de emisor, para poder identificar quién es el que envía el mensaje. Este nombre puede coincidir con el nombre de alguno de los contactos de la agenda, o puede ser el número de teléfono del emisor.

También se observa que la conversación tiene un título, que se corresponde con el nombre del grupo, en caso de que se trate de un grupo, o el nombre, o número de teléfono, del otro interlocutor en caso de tratarse de una conversación entre dos personas.

Realmente la sección correspondiente a las conversaciones de WhatsApp en el informe, no es más que una lista de conversaciones maquetadas como se ve en la Figura 19, enmarcadas con un borde, para poder diferenciar unas de otras.

Es importante resaltar que debido a que en una investigación, cuando se está revisando conversaciones, es de gran importancia el contexto de la misma. Por ese motivo, al filtrar por fechas o palabras clave, se obtendrán los mensajes coincidentes y los correspondientes a esas conversaciones un día antes y después del mensaje procesado. De esta forma se le puede dotar de contexto a la conversación para que el investigador la interprete de una manera adecuada.

Si tras interpretar la conversación del informe, el investigador necesitase revisar fechas anteriores o posteriores, deberá realizar otra búsqueda en la herramienta, indicando las fechas que considere oportunas.

NOTA: La herramienta permite acceder a las conversaciones de una manera íntegra, pero un buen investigador forense debe ser consciente de que no debe analizar la información simplemente visualizando las conversaciones, sino que se deben buscar mensajes en concreto en base a indicios, para no violar el derecho a la intimidad, ni el de secreto de comunicaciones. Pero como ya se ha comentado, eso es responsabilidad del investigador, y no de la herramienta.

Otro de los puntos importantes del informe HTML, y que además se corresponde con su página de inicio, es la generación de un timeline, o línea temporal, en la que se situen los hechos realizados para poner en un contexto temporal al investigador.

La línea temporal es algo común en todo tipo de investigaciones, por eso se ha considerado que no podía faltar en esta herramienta. Además, es una funcionalidad que la distingue claramente de otras posibles herramientas, que lo único que hacen es recuperar los archivos del backup, sin informar al usuario sobre las acciones realizadas en el dispositivo.

Al igual que ocurre con las conversaciones de WhatsApp, esta información se puede mostrar en tablas, pero debido a que es probable que el número de registros sea elevado como para ver con claridad una tabla, y que en muchas ocasiones la línea temporal es el punto de partida de la investigación, se ha optado por darle una apariencia propia, que permita situar un hecho en una fecha con un simple vistazo. Por ese motivo se ha cargado un estilo CSS en el informe que permite modificar la apariencia de la estructura HTML del informe, quedando algo parecido a lo que se puede ver en la Figura 20.

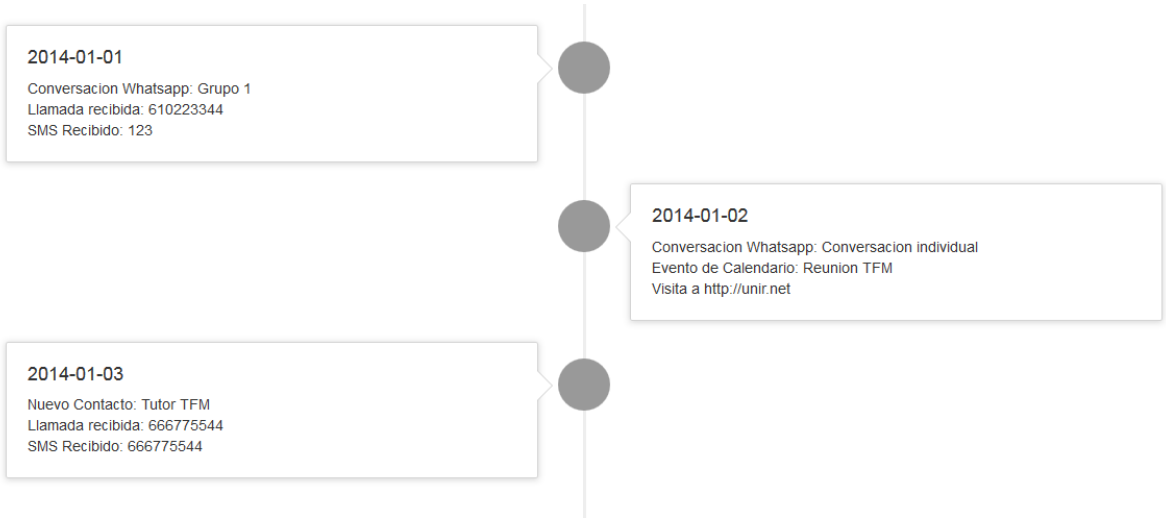


Figura 20 - Ejemplo de Timeline en el informe HTML

8.2 Modelos de informe PDF

El informe en formato PDF está más orientado a poder exportar los datos en papel, por lo que su diseño es más sobrio, permitiendo almacenar un mayor número de datos, ahorrando papel.

Debido también a que el informe en este formato se genera a través de elementos flotantes, indicando coordenadas y medidas, se ha optado por un diseño lo más simple posible, obteniendo como resultado un documento que tiene una tabla por cada tipo de datos.

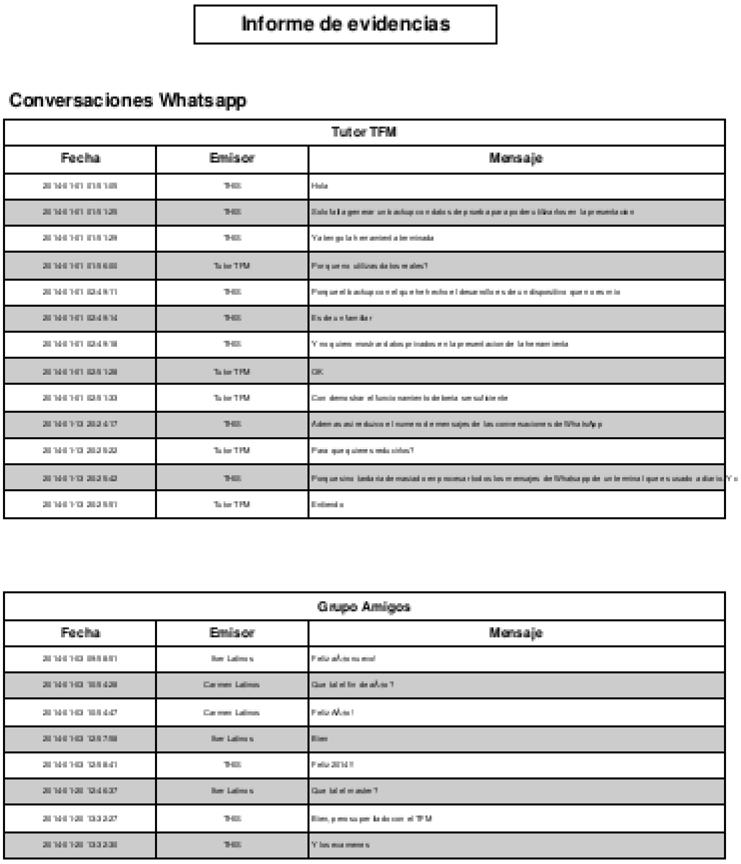


Figura 21 - Ejemplo de informe PDF

En la Figura 21 se puede ver un ejemplo de una parte del informe PDF generado. Se observa que es muy simple, pero realmente, una vez se han conseguido demostrar los hechos, no se necesita más. Una vez que se han identificado las evidencias fundamentales, se puede generar un informe PDF con ellas, para poder tener un documento que demuestre tales hechos.

NOTA: Al igual que ocurría con el visionado de las conversaciones, es responsabilidad del investigador garantizar la autenticidad del documento, sobre todo en un proceso judicial. La herramienta no proporciona funcionalidades de firma digital, por lo que el investigador debería adquirir las evidencias (backup de iOS) ante un tercero y verificar la integridad de la copia a través de algún algoritmo hash, dejando una copia intacta para poder garantizar la cadena de custodia.

9. MODULO DE REPORTING

De la misma manera que se generan dos informes con formatos distintos, también los módulos encargados de cada uno de ellos son distintos. Este capítulo pretende analizar los módulos que permiten la generación de los informes, tanto en formato HTML, como en formato PDF.

Recordemos que el flujo del componente de la herramienta encargado de generar los informes tenía como objetivo guardar en una variable los datos sobre los que se generará el informe. Así pues, el flujo de ejecución converge en un punto, que se puede ver en la Figura 22, en el que se tendrán todos los datos necesarios, y a partir del cual se debe generar los informes sobre los mismos.

Este punto, aunque se podía intuir, no estaba marcado en la Figura 7, en la que se detallaba el flujo de datos de *ibaft_report.py,* por lo que se incluye una modificación del diagrama en la que se pueda ver claramente el punto al que se está haciendo referencia.

El objetivo de este capítulo es detallar el funcionamiento de los procesos "Generar informe HTML" y "Generar informe PDF" que vienen a continuación del punto mencionado, y que como se observa, son los últimos pasos antes de la finalización de la ejecución de la herramienta.

Aunque sus nombres sean autodescriptivos, tienen su propio flujo de ejecución, un orden de operaciones y ciertas peculiaridades que merecen la pena ser analizadas.

Este capítulo no pretende ser demasiado técnico, ya que la estructura de lasdiferentes clases, y la documentación de cada método se detallan en capítulos posteriores. Por el contrario, se pretende mostrar el flujo de cada proceso de generación de informe, y mostrar también la estructura HTML generada, y que debe mantenerse para que se puedan aplicar correctamente los estilos CSS.

Más adelante, en la sección 9.3 de este mismo capítulo, se especifican los detalles de las estructuras HTML que deben tener los ficheros plantilla para que se puedan generar los informes de manera correcta.

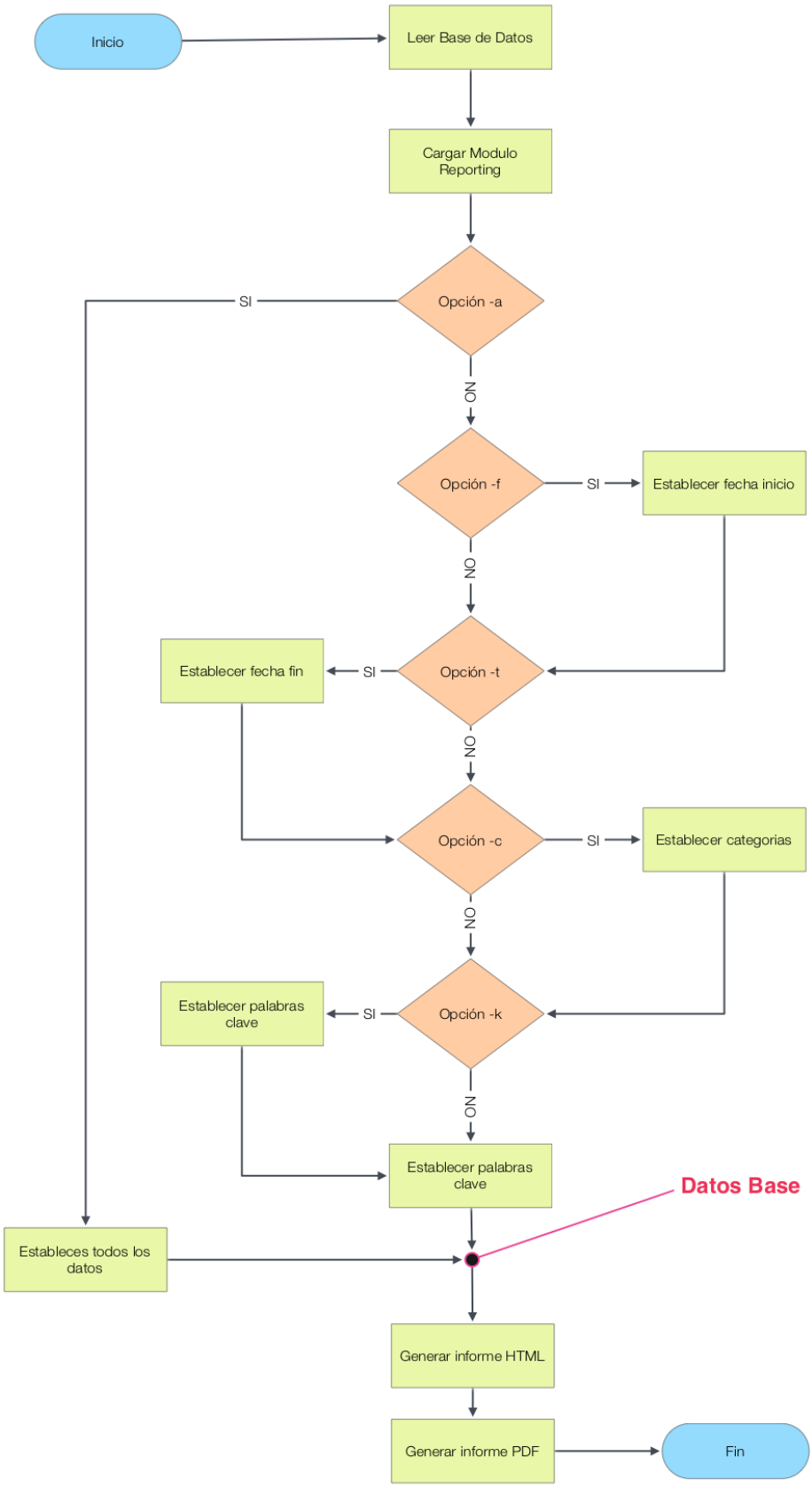


Figura 22 - Flujo de Datos del componente de reporting

9.1 Generar informe HTML

Lo primero que hace este proceso es copiar la estructura de archivos de la carpeta report, incluida en la herramienta al directorio destino indicado por línea de comandos. Por ello, al pirncipio se tendrá un informe completamente vacío en el directorio final.

A continuación se genera el archivo JSON con los datos de las comunicaciones, y lo guarda en el directorio data del informe copiado en la carpeta destino. En este punto, el informe ya está en condiciones de mostrar datos de las comunicaciones, ya que los obtendrá vía AJAX y no es necesaria modificaciones en los archivos HTML.

El proceso descrito anteriormente para las comunicaciones, se repite para los contactos, el calendario y el historial web, generando así tres archivos JSON adicionales, cada uno con la estructura que se considera más apropiada.

Debido a que la carga de los datos de comunicaciones, contactos, agenda e historial se realiza a través de archivos externos, basta con generar estos archivos. Pero para poder generar el timeline y las conversaciones de WhatsApp es necesario sustituir una marca (se detalla en el apartado 9.3) por la estructura HTML generada, guardando el resultado en el archivo correspondiente del informe.

En la Figura 23 se puede ver el flujo de ejecución que permite generar el informe en HTML. El diagrama de flujo de datos es sencillo y secuencial, pero permite hacerse con una visión del procedimiento con un sólo vistazo.

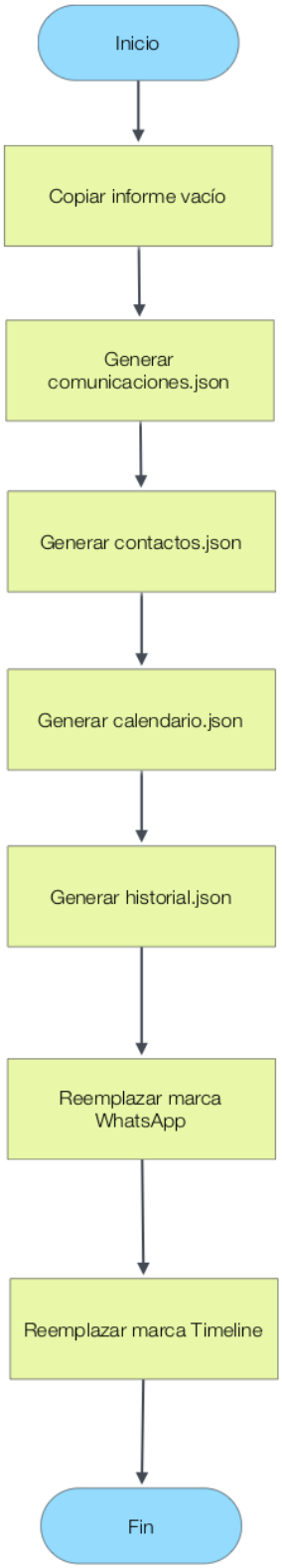


Figura 23 - Flujo de ejecución para generar el informe HTML

9.2 Generar informe PDF

Para generar el informe en PDF se utiliza un fork de la libreria FPDF, usada en PHP, a la que se han realizado cambios para adaptarla al lenguaje de programación Python. Existe un módulo, llamado *reporting_pdf.py* que define la lógica que permite generar el informe. Este módulo contiene una clase que hereda de FPDF y define sus propiedades y métodos, como el método Header() o Footer(), pero también tiene métodos propios, que permiten añadir el contenido al informe, y que son básicamente los que utiliza la herramienta.

Estos métodos reciben como parámetro un conjunto de datos que tienen que mostrar, o bien una ruta de fichero (en el caso de los ficheros JSON) que se encargarán de abrir, leer y volcar la información al informe PDF.

Primero se genera el timeline, después las conversaciones de WhatsApp, y por último, se mostrarán los datos relativos a las comunicaciones, a la agenda, a los contactos y al historial web, cada uno en su tabla y página destinada para ello.

Una vez que se ha volcado todos los datos la fichero, éste se guarda en el directorio final del informe, bajo el nombre de reportPDF.pdf

En la Figura 24 se muestra el flujo de ejecución que permite generar el informe PDF. El diagrama de flujo de datos es muy similar al del informe HTML, con la excepción de que en este, al trabajar sobre un único fichero, es necesario añadir saltos de página, con el fin de separar los diferentes tipos de información. Esto se hacía en el informe HTML a través de diferentes archivos html enlazados por medio de enlaces.

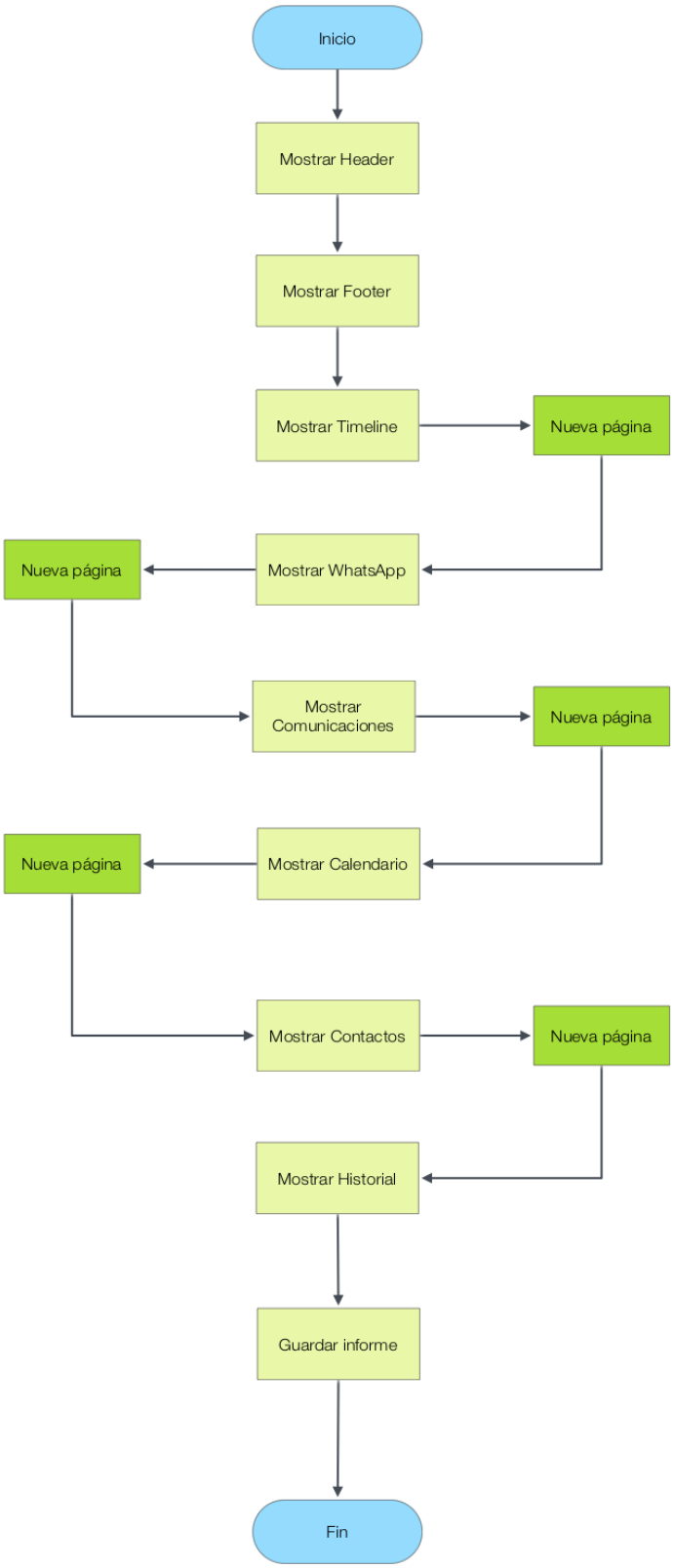


Figura 24 - Flujo de ejecución para generar el informe PDF

9.3 Estructuras HTML

En el caso del timeline y las conversaciones de WhatsApp, la información no se obtiene de archivos externos, por lo que es necesario plasmarla dentro del archivo HTML. Para ello se utilizan la siguiente marca de texto, que está comentada:

```
<!-- [%%_HTML_CONTENT_%%] -->
```

De esta forma, el módulo de reporting abrirá el archivo que ha copiado desde el directorio *esqueleto* de la herramienta y leerá su contenido. Sobre ese contenido, reemplazará la marca anterior, por el código HTML adecuado en cada caso.

Ese código HTML es diferente en cada caso. Así pues, en el caso del timeline deben existir estos elementos obligatorios:

```
<div class="panel-body">
	<ul class="timeline">
	<!-- Aqui se itera cada elemento, correspondiente a un día -->
	<!-- Se debe alternar entre clases timeline-inverted para colocar el nodo a
	cada lado del eje -->
		<li class="[timeline-inverted]">
			<div class="timeline-badge"><i class="fa fa-check"></i></div>
			<div class="timeline-panel">
				<div class="timeline-heading">
					<h4><!-- Fecha --></h4>
				</div>
				<div class="timeline-body">
					<!-- Contenido -->
				</div>
			</div>
		</li>
	</ul>
</div>
```

En caso de tratarse de una conversación de WhatsApp, la estructura HTML mínima debería ser la siguiente:

```
<div class="conversacion">
	<h3 class="title"><!-- Titulo Conversacion --></h3>
	<!-- Aqui se itera cada mensaje -->
	<div class="chat-bubble (emisor|receptor)" id="<!-- ID del mensaje -->">
		<span><b><!-- Emisor --></b></span><br/>
		<p><!-- Mensaje --></p>
		<small class="date"><! -- Fecha --></small>
		<div class="clearfix"></div>
		<div class="borde-(flecha-derecha|flecha-izquierda)"></div>
		<div class="(flecha-derecha|flecha-izquierda)"></div>
	</div>
	<div class="clearfix"></div>
</div>
```

10. DISEÑO TÉCNICO DE LA HERRAMIENTA

Este capítulo pretende mostrar el diseño técnico de la herramienta. Para ello, se utilizar diagramas UML de clases, primero para tener una visión global de los componentes de la aplicación, y después para conocer en detalle cada una de las clases.

10.1 Diagramas UML de clases

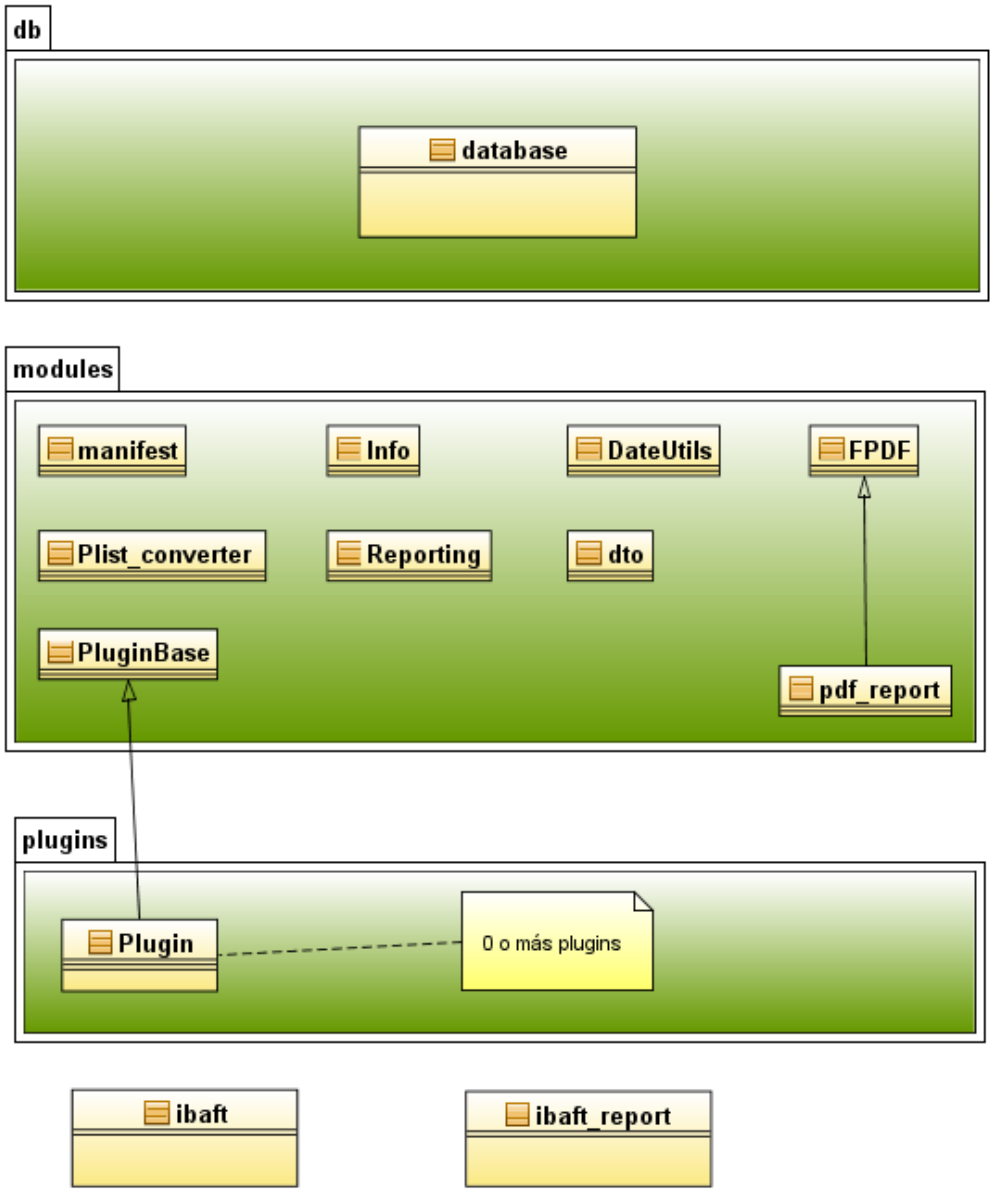


Figura 25 - Clases de la herramienta

En la Figura 25 se puede observar el diagrama de clases, en el que se muestra la estructura de la herramienta. Se divide en tres directorios:

- **db**: Contiene la clase encargada de manejar la base de datos, así como scripts SQL (no se muestran en el diagrama)
- **modules**: Contiene las clases que realizan ciertas funcionalidades obligatorias de la herramienta
- **plugins**: Contiene los plugins de la herramienta. Todos deben ser de la clase Plugin y seguir los detalles de implementación definidos en el Capítulo 7

En el directorio raíz se encuentran dos elementos, que aunque no son clases, se han querido añadir al diagrama. En realidad son scripts que ejecutan diferentes partes de la herramienta.

A continuación se va a ir analizando clase por clase, para conocer sus métodos y atributos. Se observará que en las clases python, cada método recibe como primer parámetro una instancia de sí mismo. Esto es un requisito del lenguaje, pero en realidad no se le está pasando nada en la llamada.

10.1.1 Paquete db

10.1.1.1 Diagrama UML de la clase database

Esta clase es la encargada de gestionar la conexión con la base de datos de la herramienta. El sistema gestor de base de datos es sqlite3, así que el constructor de esta clase recibirá como parámetro la ruta del fichero sqlite que contiene la base de datos.

En la Figura 26 se observa que dispone de métodos para abrir una conexión a la base de datos, cerrar esa conexión, ejecutar consultas y crear la estructura de la base de datos principal. También existen métodos que realizan funciones más específicas, como leer los ficheros o directorios de la información de la tabla manifest, o permitir añadir una opción en la tabla options, lo que generalmente se utiliza para guardar la información del archivo Info.plist.

La descripción de cada método se puede observar con más detalle en la Tabla 9, en donde se especifica cada atributo y cada método, así como sus parámetros, en caso de tenerlos, y la función que realiza cada elemento.

Figura 26 - Diagrama UML de la clase database

Esta clase es una las primeras en ejecutarse realmente, ya que permite crear la base de datos que almacenara todos los datos futuros. Para ello, se le pasa una ruta al constructor, y al ejecutar el método connect(), se creará la base de datos, estableciéndose así una conexión con la misma. A partir de ese momento se podrá leer y escribir información en el archivo de base de datos.

Una de las funcionalidades de esta clase es la posibilidad de reconstruir el backup con sus nombres de archivo originales. Para ello utiliza la información de la tabla manifest, que previamente se ha obtenido del archivo Manifest.mbdb, para poder mapear el nombre cifrado y el nombre real de cada fichero o directorio. Una vez que se consigue esa información, sólo es necesario ir copiando y renombrando cada fichero en el directorio adecuado.

Tabla 9 - Documentación de la clase database

Elemento	Tipo	Parámetros	Descripción
db	Atributo		Ruta del fichero de base de datos
con	Atributo		Conexión a la base de datos correspondiente
setup_script	Atributo		Ruta del script de creación de la base de datos
reporting_query_script	Atributo		Ruta del script que genera la agrupación de datos en la tabla output
database	Método	string filename	Constructor de la clase que recibe la ruta a la base de datos
connect	Método		Conecta a la base de datos definida en el constructor. Si la base de datos no existe, se creará.
close	Método		Cierra la conexion de la base de datos
create_database	Método		Ejecuta el script indicado en el atributo setup_script de la clase, que contiene la ruta a un script SQL con la estructura de la base de datos
get_manifest_directories	Método		Obtiene un listado de todos los directorios especificados en la tabla manifest. No se inclyen los ficheros
get_manifest_files	Método		Obtiene un listado de todos los ficheros especificados en la tabla manifest.
build_backup	Método	string source, string destination	Reconstruye el contenido del backup de iOS. Se le indica en el parámtero source el directorio que contiene el backup, y en el parámetro destination, el directorio donde se quiere construir el backup.
add_option	Método	string key, Object value	Añade un registro clave/valor a la tabla de opciones de la base de datos
get_option	Método	string key	Obtiene una opción de la base de datos por su option_key
saveInfoplist	Método	Object plist	Recorre el diccionario que contiene la información del archivo Info.plist y añade la pareja clave/valor en la base de datos, como opciones
saveManifest	Método	Object manifest	Guarda el contenido del objeto que contiene la información del archivoManifest.mbdb en la base de datos
get_fileid	Método	string path	Obtiene el nombre cifrado del archivo correspondiente con la ruta indicada en el parámetro
query	Método	string query, array parametros	Ejecuta una consulta SQL genérica. Recibe la query en el parametro query formateada con un símbolo ? en caso de recibir un parámetro. Los parámetros se establecen en el array parámetros en el mismo órden en el que se definieron en la query
get_output	Método		Ejecuta el script indicado en el atributo reporting_query_script para obtener los datos que debe insertar en la tabla output
insertOutput	Método		Inserta el resultado del método get_output() en la tabla output

10.1.2 Paquete modules

10.1.2.1 Diagrama UML de la clase manifest

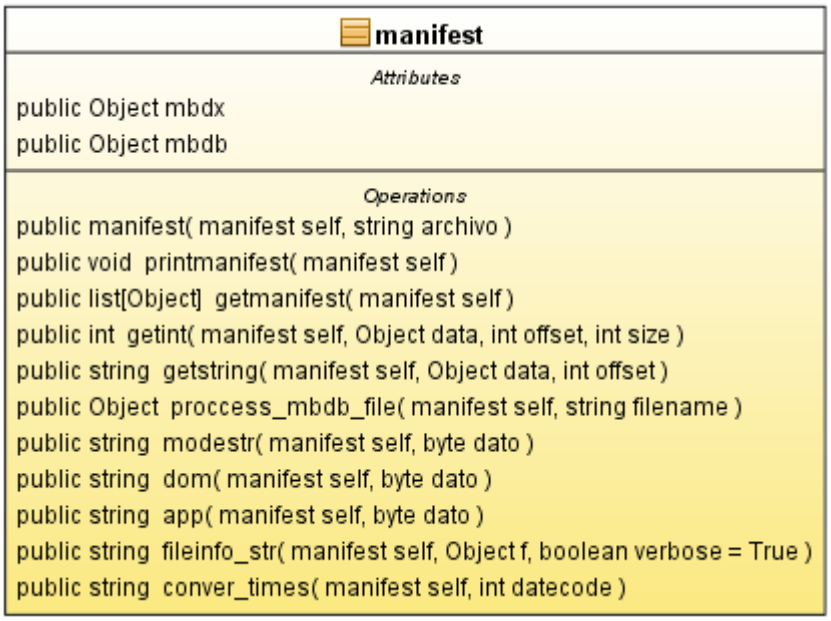


Figura 27 - Diagrama UML de la clase manifest

Esta clase permite parsear los datos del archivo Manifest.mbdb para poder manejarlos como un objeto, o lista de objetos, y poder insertarlos en la base de datos (esto lo hace la clase database).

La estructura del archivo Manifest.mbdb se ha comentado ya en el Capítulo 3, por lo que no se entrará en detalle aquí. Sin embargo, la Tabla 10 muestra una realción de los métodos de la clase y su descripción.

En realidad, todos los métodos de la clase son de uso interno, y sólo es necesario llamar al método process_mbdb_file() que devolverá una lista con los datos que contiene el archivo Manifest.mbdb en un formato adecuado.

Tabla 10 - Documentación de la clase manifest

Elemento	Tipo	Parámetros	Descripción
mbdx	Atributo		Objeto auxiliar con datos del archivo manifest
mbdb	Atributo		Objeto con al información de manifest
manifest	Método	string archivo	Constructor de la clase.Recibe el archivo Manifest.mbdb como parámetro
printmanifest	Método		Muestra por pantalla el contenido del fichero Manifest.mbdb, cargado previamente a través del constructor de esta misma clase, en formato CSV
getmanifest	Método		Obtiene una lista con todas las entradas del archivo Manifest.mbdb, previamente cargado a través del constructor de esta misma clase.
getint	Método	Object data, int offset, int size	Obtiene un entero (big-endian) y el nuevo offset del offset actual
getstring	Método	Object data, int offset	Obtiene un string y el nuevo offset del offset actual
process_mbdb_file	Método	string filename	Procesa el archivo Manifest.mbdb, previamente cargado a través del constructor de esta clase, y devuelve una lista con los datos.
modestr	Método	byte dato	Parsea el valor de los caracteres, para expresar los permisos en formato UNIX
dom	Método	byte dato	Parsea el valor del dato correspondiente al dominio
app	Método	byte dato	Parsea el valor del dato correspondiente a application
fileinfo_str	Método	Object f, boolean verbose	Construye y devuelve un string con los datos del Manifest.mbdb
convert_times	Método	int datecode	Convierte los datos de fechas a una cadena comprensible

10.1.2.2 Diagrama UML de la clase Info

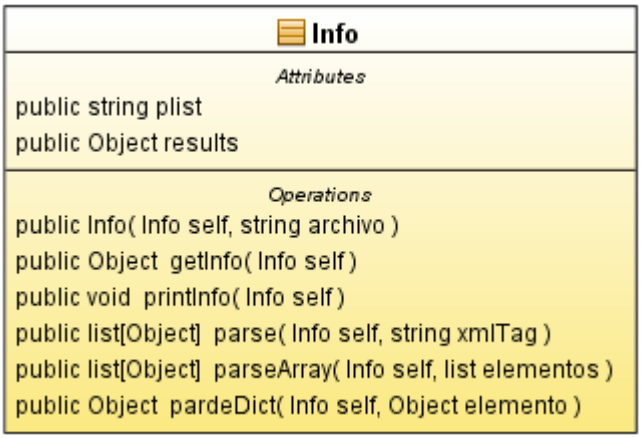


Figura 28 - Diagrama UML de la clase Info

Esta clase es la encargada de parsear el archivo Info.plist para obtener los valores de las propiedades que éste contiene. Para parsear el archivo, hace uso de una librería llamada minidom.

Es la primera clase que se ejecuta, ya que es necesario conocer el nombre del dispositivo que contiene este archivo para nombrar de ese modo al archivo de base de datos. Una vez la base de datos se haya creado, el contenido de este archivo se insertará en la tabla options, a través de registros clave/valor, serializados en JSON, en caso de tratarse de objetos.

Tabla 11 - Documentación de la clase Info

Elemento	Tipo	Parámetros	Descripción
plist	Atributo		Ruta al archivo Info.plist
results	Atributo		Contendrá el contenido del archivo representaod como un objeto
Info	Método	string archivo	Constructor de la clase. Recibe la ruta al archivo Info.plist
getInfo	Método		Devuelve el atrbuto results
printInfo	Método		Muestra la información del archivo plist por pantalla
parse	Método	string xmlTag	Parsea el contenido del archivo en base a un tag de XML
parseArray	Método	list[Object] elementos	Parsea un tag XML array, devolviendo una lista con el contenido
parseDict	Método	Object elemento	Parsea un tag XML dict, devolviendo un diccionario con parejas clave/valor

10.1.2.3 Diagrama UML de la clase DateUtils

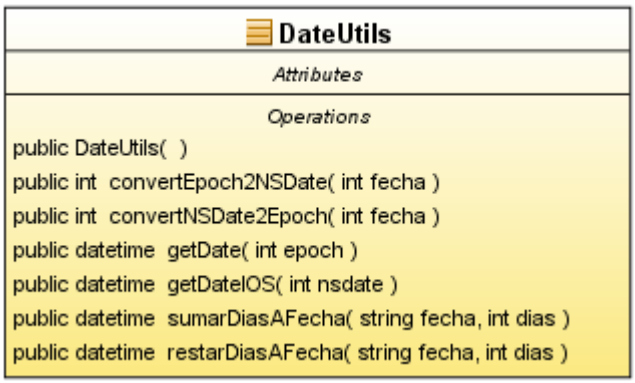


Figura 29 - Diagrama UML de la clase DateUtils

La clase DateUtils es una clase que se usa para facilitar la tarea, a la hora de trabajar con fechas. La documentación de cada método se puede ver en la Tabla 12.

Es la única clase que no tiene un constructor, y en la que todos sus métodos son estáticos

Tabla 12 - Documentación de la clase DateUtils

Elemento	Tipo	Parámetros	Descripción
convertEpoch2NSDate	Método	int fecha	Método estático que recibe un timestamp en formato EPOCH y lo devuelve en formato NSDate
convertNSDate2Epoch	Método	int fecha	Método estático recibe un timestamp en formato NSDate y lo devuelve en formato EPOCH
getDate	Método	int epoch	Método estático recibe un timestamp en formato EPOCH y devuelve un string con la fecha
getDateIOS	Método	int nsdate	Método estático recibe un timestamp en formato NSDate y devuelve un string con la fecha
sumarDiasAFecha	Método	string fecha, int dias	Método estático que recibe una representación de fecha y el numero de días que se quieren sumar. Devuelve un string con la fecha resultado
restarDiasAFecha	Método	string fecha, int dias	Método estático que recibe una representación de fecha y el numero de días que se quieren restar. Devuelve un string con la fecha resultado

10.1.2.4 Diagrama UML de la clase Plist_converter

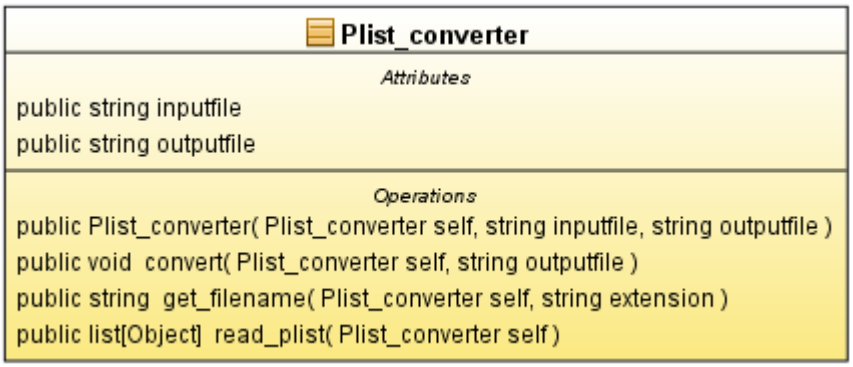


Figura 30 - Diagrama UML de la clase Plist_converter

Esta clase es utilizada para facilitar la tarea de leer archivos plist, que de otra manera no se podrían leer sin más. Es el caso, por ejemplo, del historial de Safari, que está almacenado en un archivo plist, del que no se puede extraer la información directamente.

Para poder parsear el archivo plist, se hace uso de un pequeño script en Perl, llamado plutil.pl. La documentación de esta clase se muestra en la Tabla 13.

Tabla 13 - Documentación de la clase Plist_converter

Elemento	Tipo	Parámetros	Descripción
inputfile	Atributo		Ruta al archivo plist que se quiere convertir
outputfile	Atributo		Ruta del archivo que se generará en formato XML
Plist_converter	Método	string inputfile, string outputfile	Constructor que recibe la ruta al archivo plist a convertir y la ruta destino del archivo legible
convert	Método	string outputfile	Convierte el archivo pasado en el constructor a un formato legible. Para ello utiliza el script plutil.pl, que recibe el archivo y genera otro de salida
get_filename	Método	string extension	Devuelve el nombre que genera el script perl plutil.pl por defecto al pasarle el archivo almacenado en inputfile
read_plist	Método		Obtiene una lista con los elementos del archivo plist

10.1.2.5 Diagrama UML de la clase dto

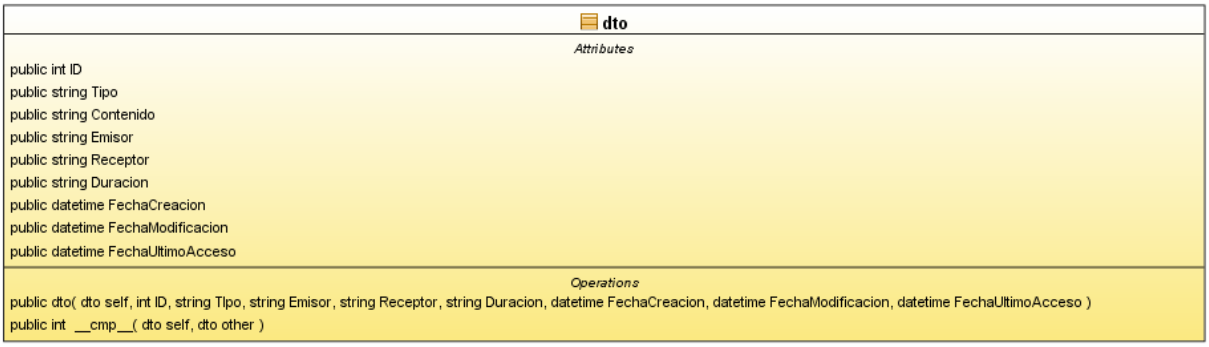


Figura 31 - Diagrama UML de la clase dto

El desarrollo de esta clase surge ante la necesidad de volver a reordenar (en Python) una lista de datos con los registros de la tabla output. En un primer momento, la información de la tabla output se manejaba como diccionarios, pero al necesitar ordenarlos, surgió la necesidad de modelar esa información como objeto, para así poder implementar el método __cmp__ que indica el criterio de ordenación, en este caso el campo de FechaCreación.

Tabla 14 - Documentación de la clase dto

Elemento	Tipo	Parámetros	Descripción
ID	Atributo		Identificador del registro
Tipo	Atributo		Tipo del registro. Normalmente el tipo se define en el plugin, y pueden ser, por ejemplo, Llamada, SMS, Nota, WhatsApp, Historial…etc
Contenido	Atributo		Contenido del registro. Normalmente usado para mensajes, conversaciones..etc
Emisor	Atributo		Emisor de la comunicación, si aplica
Receptor	Atributo		Receptor de la comunicación, si aplica
Duracion	Atributo		Duración de la comunicación, si aplica
FechaCreacion	Atributo		Fecha en la que se produjo el registro
FechaModificación	Atributo		Fecha en la que se modificó el registro
FechaUltimoAcceso	Atributo		Fecha de la última vez que se accedió al registro
dto	Método	int ID, string Tipo, string Contenido, string Emisor, string Receptor, string Duracion, datetime FechaCreacion, datetime FechaModificación, datetime FechaUltimoAcceso	Contstructor de la clase que recibe todos los atributos en el orden especificado para instanciar el objeto con el valor de todos ellos.
__cmp__	Método	dto other	Método interno que usa python para establecer la lógica que se debe aplicar al comparar dos objetos dto. En este caso se le indica a python que debe comparar por el campo FechaCreación.

10.1.2.6 Diagrama UML de la clase Reporting

La clase Reporting es el núcleo del script *ibaft_report.py*, ya que es la que filtrará según los criterios indicados por el usuario, y construirá los informes, tanto el informe HTML como el informe PDF, apoyándose en la clase pdf_report.

La estructura de la clase se puede ver en la Figura 32, mientras que la documentación técnica, junto con la descripción de cada método aparece en la Tabla 15.

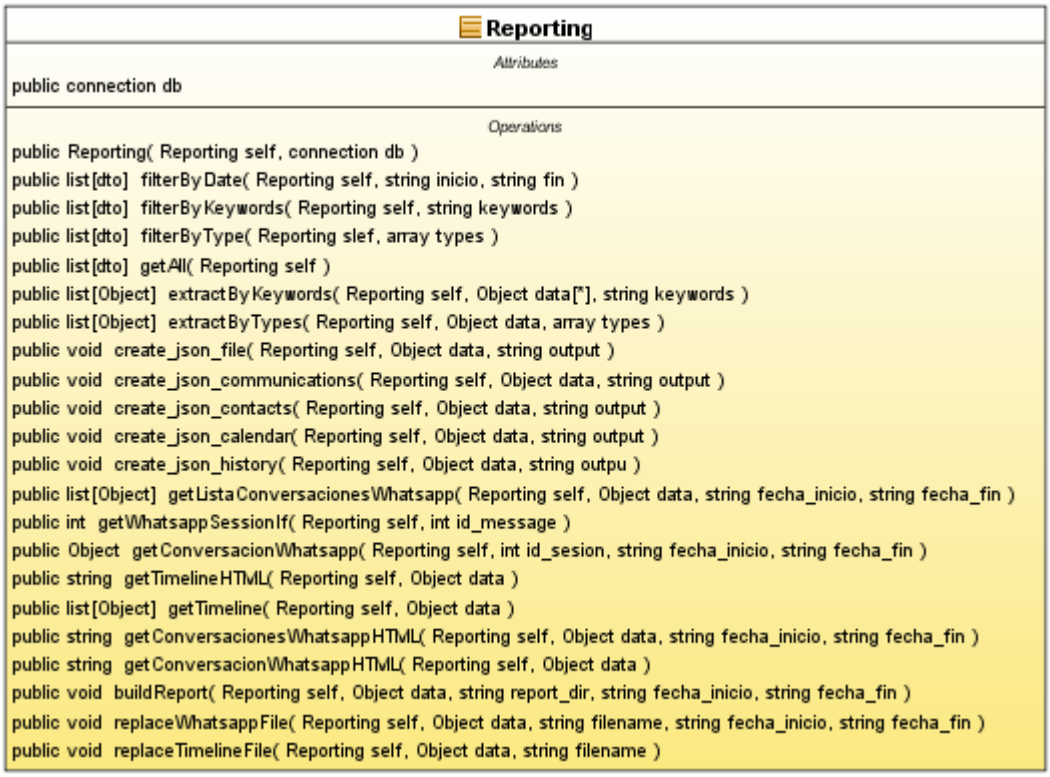


Figura 32 - Diagrama UML de la clase Reporting

Tabla 15 - Documentación de la clase Reporting

Elemento	Tipo	Parámetros	Descripción
db	Atributo		Instancia de la clase database
Reporting	Método	connection db	Constructor de la clase que rebice una instancia de la clase database
filterByDate	Método	string inicio, string fin	Filtra por fechas los datos. Si se indican las dos fechas se buscaran los datos que se encuentren entre ambas. Si sólo se indica la fecha de inicio, se buscan los datos con una fecha posterior. Si solo de indica una fecha de fin, se buscan los datos con una fecha anterior. Por el contrario, si no se indica ninguna fecha, se buscarán todos los datos.
filterByKeywords	Método	string keywords	Filtra por palabras clave. Recibe una o varias palabras clave y devuelve los datos que contengan alguna de ellas
filterByType	Método	array types	Filtra por tipo. Recibe un array con los tipos que se quieren buscar, y devulve los que coinciden con alguno de ellos.
getAll	Método		Obtiene todos los registros de la tabla output.

extractByKeywords	Método	Object data, string keywords	Extrae datos por palabras clave. Recibe una lista de objetos, y una listas de palabras clave. Devuelve aquellos objetos de la lista que contengan alguna de las palabras clave.
extractByTypes	Método	Object data, array types	Extrae por tipo. Recibe una lista de objetos y una lista de tipos. Devuelve aquellos objetos que coinciden en tipo con algunos de ellos. La lista de objetos que recibe debe ser el resultado de la ejecución extractByKeywords.
create_json_file	Método	Object data, string output	Crea un fichero JSON con el volcado de los datos pasados como parámetro data, en el archivo destino indicado como parametro output
create_json_communications	Método	Object data, string output	Crea el fichero JSON con el volcado de las comunicaciones (Llamadas y SMS) a partir de los datos pasados como parámetro data, en el archivo destino indicado como parametro output
create_json_contacts	Método	Object data, string output	Crea el fichero JSON de los contactos a partir de los datos pasados como parámetro data, en el archivo destino indicado como parametro output
create_json_calendar	Método	Object data, string output	Crea el fichero JSON del calendario a partir de los datos pasados como parámetro data, en el archivo destino indicado como parametro output
create_json_history	Método	Object data, string output	Crea el fichero JSON del historial a partir de los datos pasados como parámetro data, en el archivo destino indicado como parametro output
getListaConversacionesWhatsapp	Método	Object data, string fecha_inicio, string fecha_fin	Obtiene, a partir de una lista con los datos, una lista de conversaciones de whatsapp, que se encuentren en el parámetro data, desde la fecha fstart hasta la fecha fend, en caso de indicarse. Si no se indican fechas obtendrá toda la conversación.
getWhatsappSessionId	Método	int id_message	Obtiene el id de sesión de un mensaje de whatsapp
getConversacionWhatsapp	Método	int id_Sesion, string fecha_inicio, string fecha_fin	Obtiene un objeto que representa una conversación de Whatsapp. La conversación se obtiene desde la fecha finicio hasta la fecha ffin, si se han indicado. Si no se indican, se obtiene toda la conversación
getTimelineHTML	Método	Object data	Obtiene el codigo HTML para mostrar el Timeline de los datos pasados como parámetro data.
getTimeline	Método	Object data	Obtiene un objeto que modela la información del timeline
getConversacionesWhatsappHTML	Método	Object data, string fecha_inicio, string fecha_fin	Obtiene el código HTML para mostrar todas las conversaciones de whatsapp contenidas en los datos pasados como parámetro data

getConversacionWhatsappHTML	Método	Object data	Obtiene el cófigo HTML de una conversación de whatsapp
buildReport	Método	Object data, string report_dir, string fecha_inicio, string fecha_fin	Construye el directorio que contiene el informe, y crea los archivos de datos necesarios.
replaceWhatsappFile	Método	Object data, string filename, string fecha_inicio, string fecha_fin	Abre el fichero del informe correspondiente a las conversaciones de Whatsapp, y reemplaza la marca de contenido por el código HTML para visualizar las conversaciones.
replaceTimelineFile	Método	Object data, string filename	Abre el fichero del informe correspondiente al timeline, y reemplaza la marca de contenido por el código HTML para visualizar dicho timeline.

10.1.2.7 Diagrama UML de la clase pdf_report

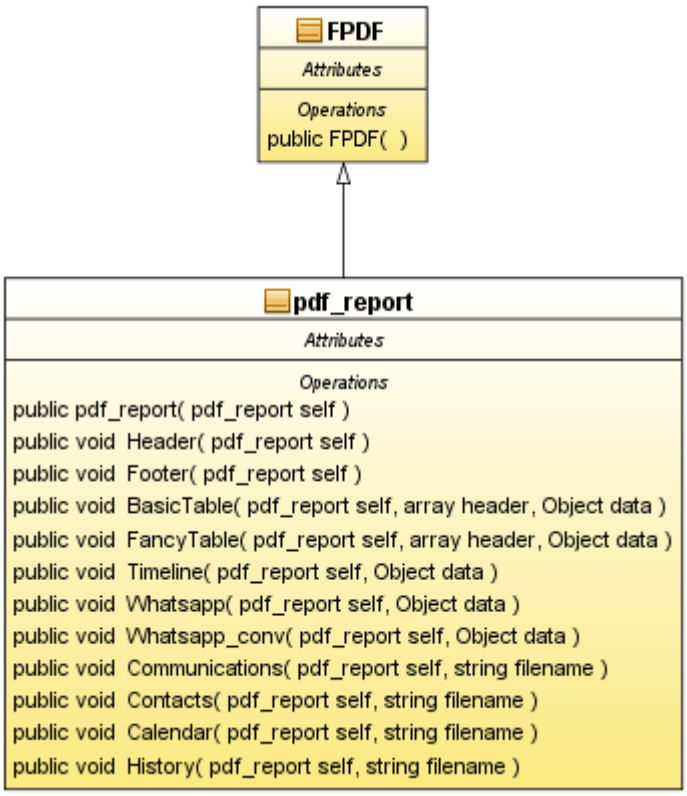


Figura 33 - Diagrama UML de la clase pdf_report

Esta clase es la encargada de generar el informe PDF. Es una clase muy sencilla con los métodos justos para generar el informe con los datos que corresponden. Tiene un método para mostrar cada tipo de dato, y de los que se puede ver su descripción en la Esta clase hereda de la clase FPDF, de la cual no se incluye diagrama de clase por su gran número de

métodos (y por no ser un desarrollo propio), que es una modificación de la librería FPDF de PHP para adaptarla al lenguaje Python.

Tabla 16.

Esta clase hereda de la clase FPDF, de la cual no se incluye diagrama de clase por su gran número de métodos (y por no ser un desarrollo propio), que es una modificación de la librería FPDF de PHP para adaptarla al lenguaje Python.

Tabla 16 - Documentación de la clase pdf_report

Elemento	Tipo	Parámetros	Descripción
pdf_report	Método		Constructor de la clase, que a su vez llama al constructor de la clase padre FPDF
Header	Método		Imprime el contenido del encabezado en el archivo PDF
Footer	Método		Imprime el contenido del pie de página en el archivo PDF
BasicTable	Método	array header, Object data	Imprime una tabla con un formato sencillo, indicandole las cabeceras de las tablas (Columnas), y los datos que debe contener
FancyTable	Método	array header, Object data	Imprime una tabla con un formato más elegante, indicandole las cabeceras de las tablas (Columnas), y los datos que debe contener
Timeline	Método	Object data	Imprime la tabla que contiene la información del Timeline
Whatsapp	Método	Object data	Imprime la información de las conversaciones de Whatsapp.
Whatsapp_conv	Método	Object data	Imprime la tabla que contiene la información de una conversación de Whatsapp
Communications	Método	string filename	Imprime la tabla que contiene la información de las comunicaciones
Contacts	Método	string filename	Imprime la tabla que contiene la información de los contactos
Calendar	Método	string filename	Imprime la tabla que contiene la información de los eventos de la agenda y el calendario
History	Método	string filename	Imprime la tabla que contiene la información del historial web.

10.1.2.8 Diagrama UML de la clase PluginBase

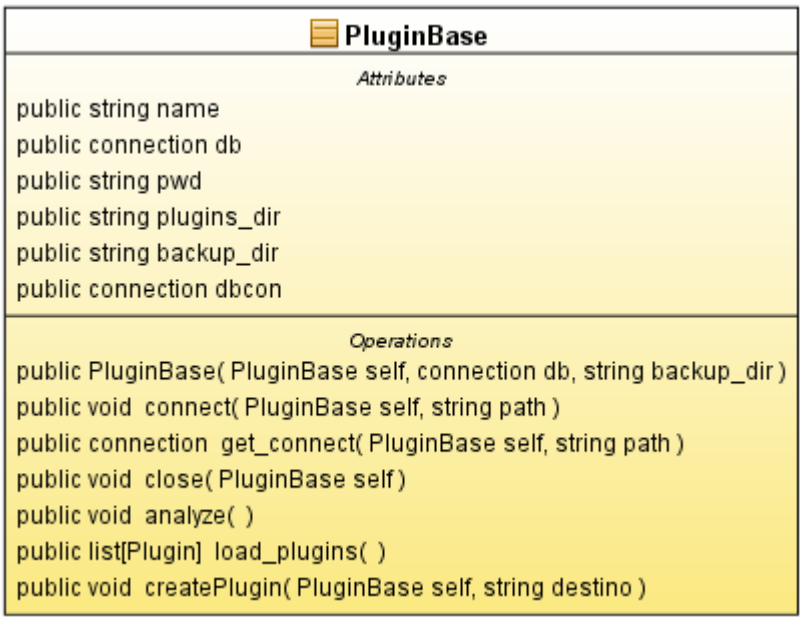


Figura 34 - Diagrama UML de la clase PluginBase

Esta es la clase de la que heredarán las clases de cada futuro plugin (clase Plugin), y la encargada tanto de cargar dichos plugins, como de ejecutarlos. Para ambas tareas dispone de diferentes métodos, que se detallan en la Tabla 17.

También dispone de diferentes atributos que pueden ser de utilidad a la hora de desarrollar un plugin para la herramienta, tales como el directorio de trabajo actual, el directorio de los plugins o el directorio que contiene el backup.

Además, se define en esta clase dos conexiones a bases de datos, una para la base de datos de la herramienta, y una conexión sin inicializar, destinada a abrir la base de datos de la que extraer la información, y métodos para gestionarlas, por lo que supone una capa de abstracción para el desarrollador de plugins, que no tiene porque conocer los detalles de conexión de ambas bases de datos, sino que simplemente debe indicar la ruta de la base de datos de la que quiere obtener la información.

Tabla 17 - Documentación de la clase PluginBase

Elemento	Tipo	Parámetros	Descripción
name	Atributo		Nombre del plugin
db	Atributo		Instancia de la clase database
pwd	Atributo		Directorio de trabajo actual
plugins_dir	Atributo		Directorio donde se encuentran los plugins
backup_dir	Atributo		Directorio donde se encuentra el backup
dbcon	Atributo		Conexión a la base de datos del componente. Se trata de una nueva conexión de base de datos que supondrá la forma de conectar con el origen de los datos que se quieran obtener.
PluginBase	Método	connection db, string backup_dir	Constructor de la clase que recibe la instancia de la calse database y el directori del backup
connect	Método	string path	Conecta con la base de datos indicada en el parámetro path y guarda la conexión en el atributo dbcon
get_connect	Método	string path	Obtiene la conexión de la base de datos indicada en el parámetro path
close	Método		Cierra la conexión de la base de datos indicada en el atributo dbcon
analyze	Método		Método que ejecuta el método analyze de cada plugin cargado.
load_plugins	Método		Carga todos los plugins del directorio plugins, que tengan extensión .py y los añade en una lista de instancias que representan los plugins cargados
createPlugin	Método	string destino	Copia la plantilla de un plugin al destino indicado. Si es un directorio, se llamará plugin.py, si es un fichero que no existe, se creará. Si el fichero existe, no se creará el plugin, y se informará de ello.

10.1.3 Paquete plugins

10.1.3.1 Diagrama UML de la clase Plugin

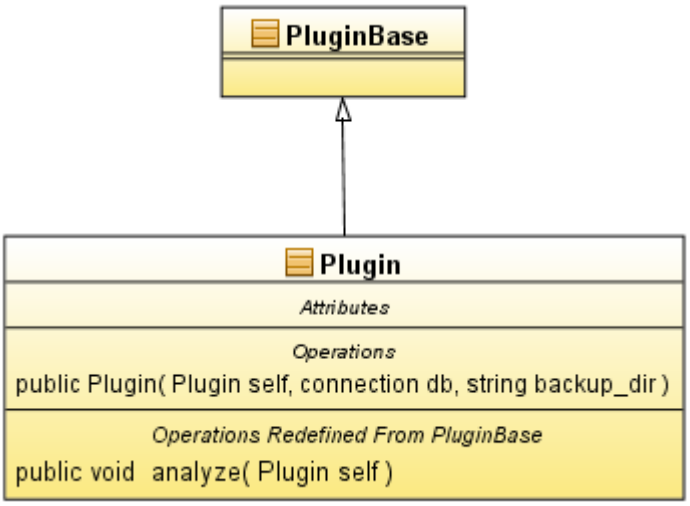


Figura 35 - Diagrama UML de la clase Plugin

Tal y como se ha comentado en el Capítulo 7, los plugins pueden tener cualquier nombre de archivo, pero deben contener un clase python llamada Plugin, que herede de PluginBase.

El único requisito es que debe contener un método analyze(), que será además el único método que se ejecute del plugin. Esto no impide al desarrollador crear nuevos métodos que son llamados desde el método analyze.

Debido a que el método analyze, y los métodos adicionales que el desarrollador considere oportunos, hacen una funcionalidad diferente en cada plugin, no se incluye una tabla con documentación técnica de esta clase.

11. MANUAL DE USUARIO

En este capítulo se muestra un ejemplo de uso de la herramienta, aportando capturas de pantalla y explicando paso a paso las posibilidades que tiene la herramienta.La ejecución se realiza en un sistema Debian.

Para poder llevar a cabo, se ha copiado en dos directorios distintos, tanto el backup de iOS, como la herramienta. Se encuentran en /usr/local/iosBackup y en /usr/local/ibaft, respectivamente.

```
root@debian:~# ls -l /usr/local
total 640
drwxrwsr-x 2 root staff   4096 jun  7 20:39 bin
drwxrwsr-x 2 root staff   4096 jun  7 20:39 etc
drwxrwsr-x 2 root staff   4096 jun  7 20:39 games
drwx------ 7 root staff   4096 may 28 19:36 ibaft
drwxrwsr-x 2 root staff   4096 jun  7 20:39 include
drwx------ 2 root staff 618496 jun  4 17:30 iosBackup
drwxrwsr-x 4 root staff   4096 jun  7 21:25 lib
lrwxrwxrwx 1 root staff      9 jun  7 20:39 man -> share/man
drwxrwsr-x 2 root staff   4096 jun  7 20:39 sbin
drwxrwsr-x 7 root staff   4096 jun  7 21:35 share
drwxrwsr-x 2 root staff   4096 jun  7 20:39 src
```

Figura 36 - Directorios de ibaft y el backup de iOS en Debian

El backup de iOS ha sido modificado, al menos las bases de datos y componentes que serán analizados por la herramienta, para evitar mostrar información privada, pero en un entorno real, se mostraría toda la información.

Para comenzar, si intentamos ejecutar el componente *ibaft.py* sin argumentos, nos mostrará un mensaje con la sintaxis, tal y como se puede ver en la Figura 37.

```
root@debian:/usr/local/ibaft# python ibaft.py

        ibaft.py -s backup_dir -d destination_dir [-h] [-b]
```

Figura 37 - Sintaxis de ibaft.py

Se observa que tiene una opción -h, que se corresponde con el menú de ayuda, así que lo ejecutamos para que muestre el texto de ayuda. Este texto se muestra en la Figura 38, y contiene tanto el texto de la sintaxis como el texto de ayuda propiamente dicho.

```
root@debian:/usr/local/ibaft# python ibaft.py -h

        ibaft.py -s backup_dir -d destination_dir [-h] [-b]

        ========================================================================

        -s, --source-dir:       Indica el directorio en el que se encuentra el backup a analizar.
        -d, --destination-dir:  Indica el directorio donde se guardará la base de datos maestra,
                                y los archivos recuperados, en caso de haber indicado la opción -b.
        -b, --build-backup:     Si se indica, se copiaran los ficheros recuperados al directorio destino.
        -h, --help:             Muestra este texto de ayuda
```

Figura 38 - Texto de ayuda de ibaft.py

Vemos que los argumentos requeridos son el origen del backup y el destino del backup, y que podemos optar por reconstruirlo. En la ejecución, vamos a reconstruirlo en el directorio /usr/local/backup, previamente creado:

```
root@debian:/usr/local/ibaft# python ibaft.py -s /usr/local/iosBackup/ -d /usr/local/backup/ -b
Cargando el plugin sms.....
Cargando el plugin whatsapp.....
Cargando el plugin llamadas.....
Cargando el plugin safari.....
Cargando el plugin contactos.....
Cargando el plugin notas.....
Cargando el plugin calendario.....
Analizando registros....

        ========================================================================
           SE HA COMPLETADO EL PROCESO CORRECTAMENTE
        ========================================================================

        Base de datos creada: /usr/local/backup/TFM.sqlite

        Ahora debe ejecutar el script ibaft_report.py indicandole en el parámetro -d la ruta a la base
        de datos que acaba de crear. Deberá indicar además las opciones de búsqueda de evidencias y un direcotrio
        para la creación del informe. Si tiene dudas, puede ejecutar ibaft_report.py -h.
```

Figura 39 - Reconstrucción del backup

Como se observa en la Figura 39, se ha indicado la opción -b, que hace que la estructura del backup se recupere en el directorio indicado, en el que se ha creado un subdirectorio llamado backup, en el que muestra todo el contenido del backup.

```
root@debian:/usr/local/ibaft# ls -l /usr/local/backup/
total 7116
drwxr-sr-x 10 root staff    4096 jun  8 10:58 backup
-rw-r--r--  1 root staff 7275520 jun  8 10:59 TFM.sqlite
```

Figura 40 - Directorio destino

Si no se hubiese indicado la opción -b, en este directorio sólo existiría el archivo TFM.sqlite, que es la base de datos que contiene toda la información analizada.

Ejecutando un tree en el directorio, vemos que contiene los archivos con su nombre correcto, tal y como muestran las siguientes imágenes.

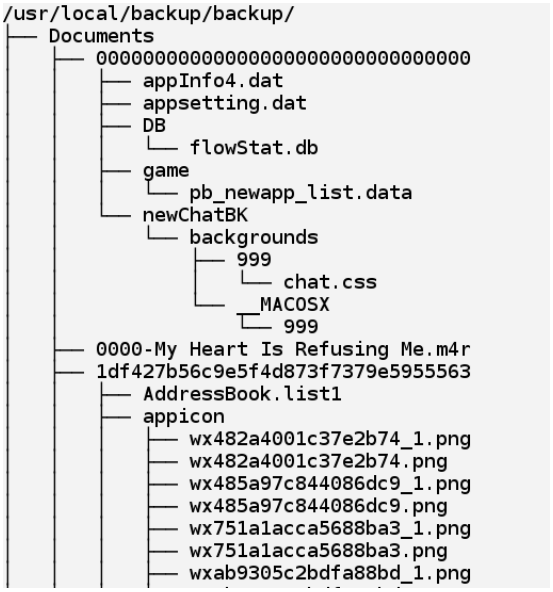


Figura 41 - tree de /usr/local/backup/backup (I)

Figura 42 - tree de /usr/local/backup/backup (II)

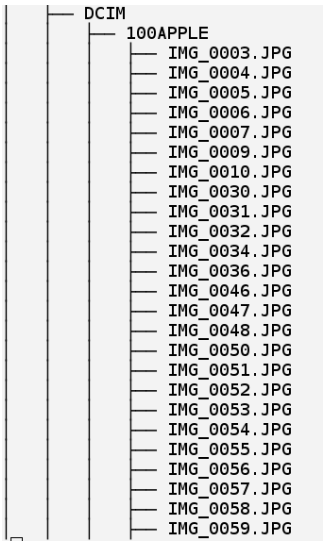


Figura 43 - tree de /usr/local/backup/backup (III)

Una vez tenemos la base de datos creada, se puede ejecutar ibaft_report.py para la generación de informes. Si se intenta ejecutar sin argumentos, se muestra el texto de sintaxis.

```
root@debian:/usr/local/ibaft# python ibaft_report.py

        ibaft_report.py -d database -o output_dir ([-f date][-t date][-k keywords][-c categories] | [-a]) [-h]
```

Figura 44 - Sintaxis de ibaft_report.py

Si se indica la opción -h o --help, mostrará el texto de ayuda:

```
root@debian:/usr/local/ibaft# python ibaft_report.py -h

        ibaft_report.py -d database -o output_dir ([-f date][-t date][-k keywords][-c categories] | [-a]) [-h]

        ================================================================================

        -d, --databse:          Indica la base de datos que utiliza para leer los datos. Tiene que ser
                                un archivo sqlite generado con la herramienta ibaft.py
        -f, --from:             Indica la fecha desde la que se buscarán los datos. Debe estar en formato
                                [YYYY-mm-dd HH:MM:SS]
        -t, --to:               Indica la fecha hasta la que se buscarán los datos. Debe estar en formato
                                [YYYY-mm-dd HH:MM:SS]
        -k, --keywords:         Indica las palabras clave que buscar
        -c, --categories:       Indica la lista de categorias en las que buscar, separadas por comas.
                                p.e: 'sms,whatsapp,nota,llamada,historial'
        -a, --all:              Indica que se quiere obtener toda la información
        -o,--output-dir         Indica el directorio donde se guardará el informe
        -h, --help:             Muestra este texto de ayuda
```

Figura 45 - Texto de ayuda de ibaft_report.py

Conocemos la ruta de la base de datos (/usr/local/backup/TFM.sqlite), por lo que podemos generar los informes. Vamos a generarlos en la carpeta temporal /tmp. Además, se van a ejecutar varios filtros.

11.1 Filtrando por palabras clave

Se puede generar informes filtrando por palabras clave, con la opción -k. En este caso vamos a filtrar por la palabra "herramienta", de la que se es consciente que existe información.

```
root@debian:/usr/local/ibaft# python ibaft_report.py -d /usr/local/backup/TFM.sqlite -o /tmp/ -k herramienta
Creando directorio de informe....
Generando informe PDF....
Informe HTML generado en /tmp//report
Informe PDF guardado en /tmp//report/reportPDF.pdf
```

Figura 46 - Ejecución de ibaft_report.py filtrando por palabras clave

Si accedemos a /tmp/report/ encontraremos el informe, tanto HTML como PDF. Al abrir el informe HTML, vemos que el timeline solo ha encontrado esa palabra en conversaciones de WhatsApp (que era lo que se esperaba).

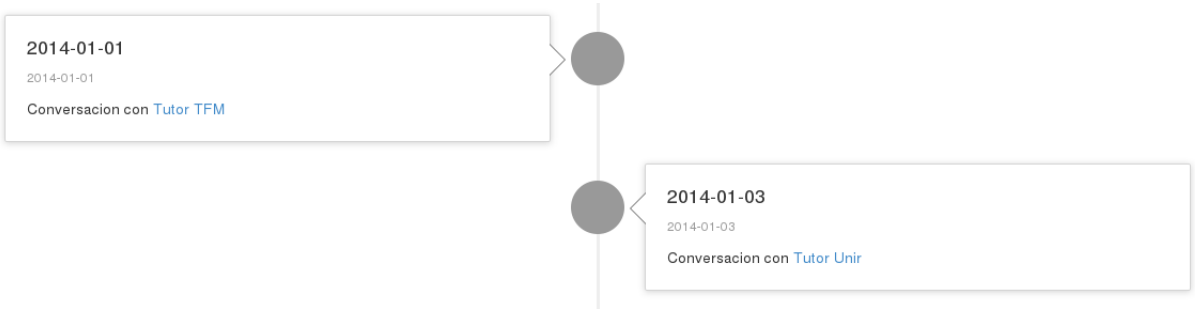


Figura 47 - Timeline filtrado por palabras clave

Si pinchamos en alguno de los links, nos llevará a la conversación mantenida, que se puede ver en la Figura 48. Obviamente, y debido a que el backup ha sido alterado para no revelar información privada, esta conversación es totalmente falsa y no se ha llegado a producir realmente.

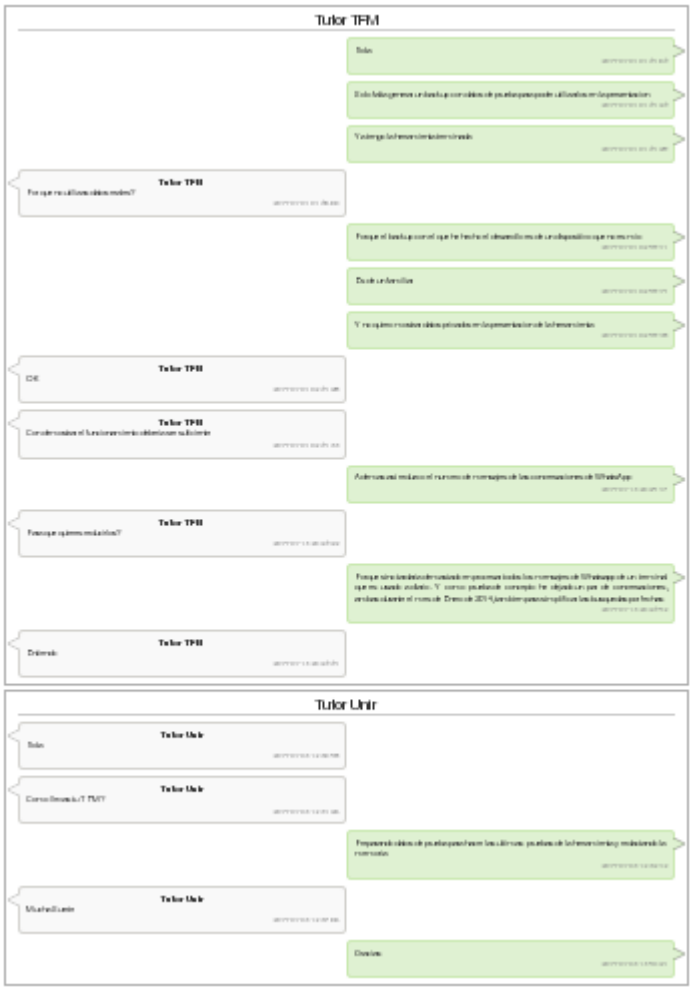


Figura 48 - Conversaciones Whatsapp filtradas por palabra clave

Sin embargo, si se observa otras secciones del informe, como la de Comunicaciones, Contactos, Agenda o Historial, se observa que las tablas no tienen datos que mostrar, como se ve en las siguientes imágenes.

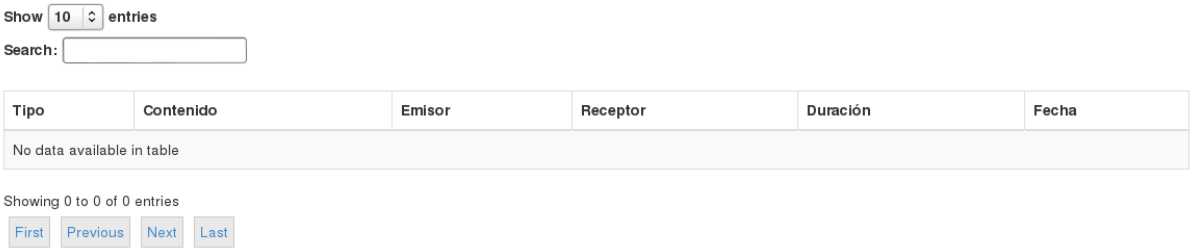


Figura 49 - Comunicaciones sin datos

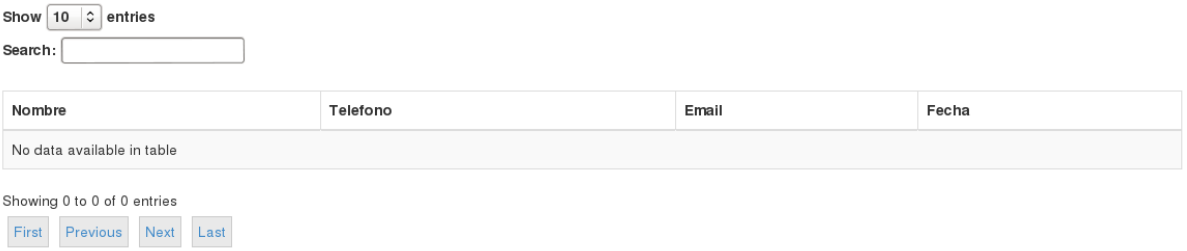


Figura 50 - Contactos sin datos

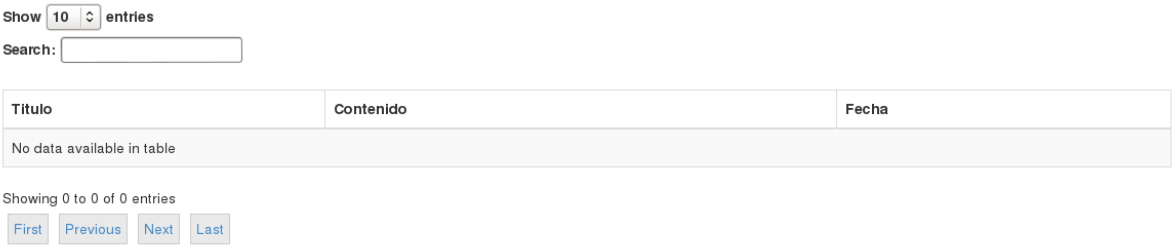


Figura 51 - Agenda sin datos

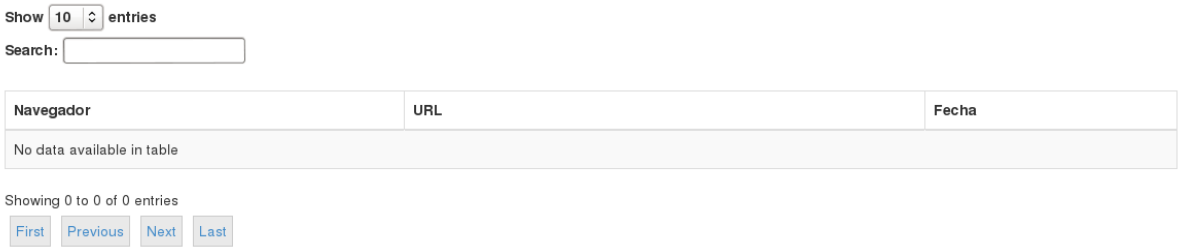


Figura 52 - Historial web sin datos

El informe PDF, al igual que el informe HTML, solo dispone de las dos entradas de timeline y las conversaciones de WhatsApp, mientras que el resto de datos son tablas vacías.

Timeline

Fecha	Evento
2014-01-01	Conversacion con Tutor TFM
2014-01-03	Conversacion con Tutor Unir

Conversaciones Whatsapp

Tutor TFM		
Fecha	**Emisor**	**Mensaje**
2014-01-01 01:51:05	THIS	Hola
2014-01-01 01:51:25	THIS	Solo falta generar un backup con datos de prueba para poder utilizarlos en la presentacion
2014-01-01 01:51:29	THIS	Ya tengo la herramienta terminada
2014-01-01 01:56:00	Tutor TFM	Porque no utilizas datos reales?
2014-01-01 02:49:11	THIS	Porque el backup con el que he hecho el desarrollo es de un dispositivo que no es mio
2014-01-01 02:49:14	THIS	Es de un familiar
2014-01-01 02:49:18	THIS	Y no quiero mostrar datos privados en la presentacion de la herramienta
2014-01-01 02:51:28	Tutor TFM	OK
2014-01-01 02:51:33	Tutor TFM	Con demostrar el funcionamiento deberia ser suficiente
2014-01-13 20:24:17	THIS	Ademas asi reduzco el numero de mensajes de las conversaciones de WhatsApp
2014-01-13 20:25:22	Tutor TFM	Para que quieres reducirlos?
2014-01-13 20:25:42	THIS	Porque sino tardaria demasiado en procesar todos los mensajes de Whatsapp de un terminal que es usado a diario. Y o
2014-01-13 20:25:51	Tutor TFM	Entiendo

Tutor Unir		
Fecha	**Emisor**	**Mensaje**
2014-01-03 12:30:48	Tutor Unir	Hola
2014-01-03 12:31:26	Tutor Unir	Como llevas tu TFM?
2014-01-03 12:32:12	THIS	Preparando datos de prueba para hacer las ultimas pruebas de la herramienta y redactando la memoria
2014-01-03 12:37:06	Tutor Unir	Mucha Suerte
2014-01-03 13:40:21	THIS	Gracias

Figura 53 - Informe PDF filtrado por palabras clave

11.2 Filtrando por fechas

Se le puede indicar a la herramienta que sólo se desea obtener datos en un intervalo de fechas concreto, a través de las opciones -f y -t (from y to). Las fechas indicadas deben ser en formato YYYY-mm-dd HH:MM:SS. Por ejemplo, si queremos ver los datos ocurridos entre el día 1 de Enero de 2014 y el día 4 de Enero de 2014, ejecutamos el comando de la siguiente imagen:

```
root@debian:/usr/local/ibaft# python ibaft_report.py -d /usr/local/backup/TFM.sqlite -o /tmp/ -f "2014-01-01 00:00:00" -t "2014-01-04 00:00:
00"
Creando directorio de informe....
Generando informe PDF....
Informe HTML generado en /tmp//report
Informe PDF guardado en /tmp//report/reportPDF.pdf
```

Figura 54 - Ejecución de ibaft_report.py filtrando por fechas

Ahora en /tmp/report se encuentra el directorio del informe (perdiendose el anterior) y en el que vemos el siguiente timeline:

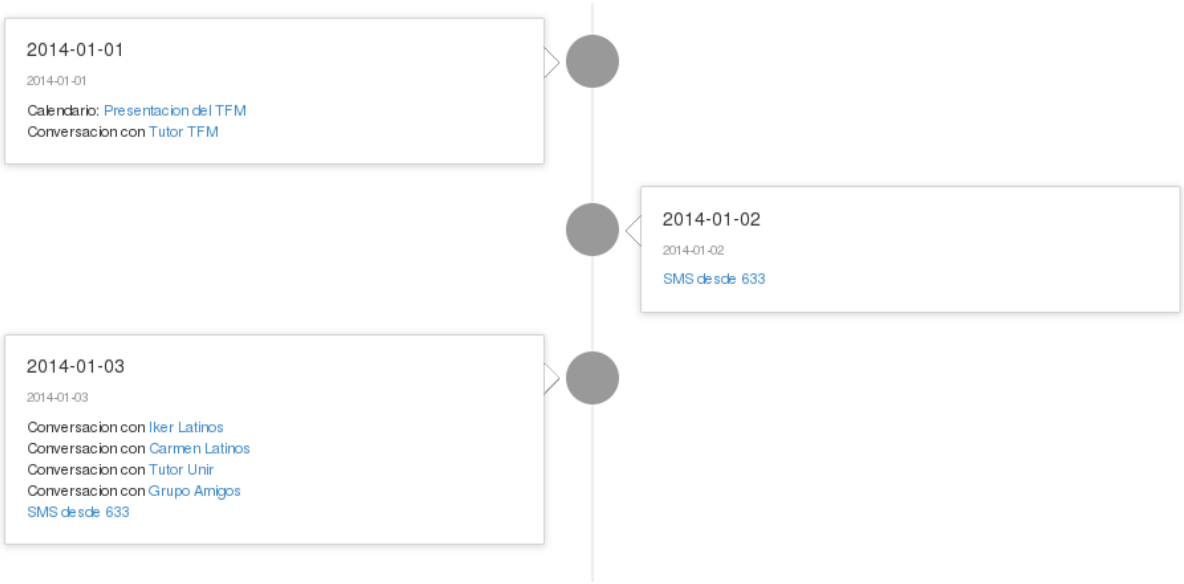
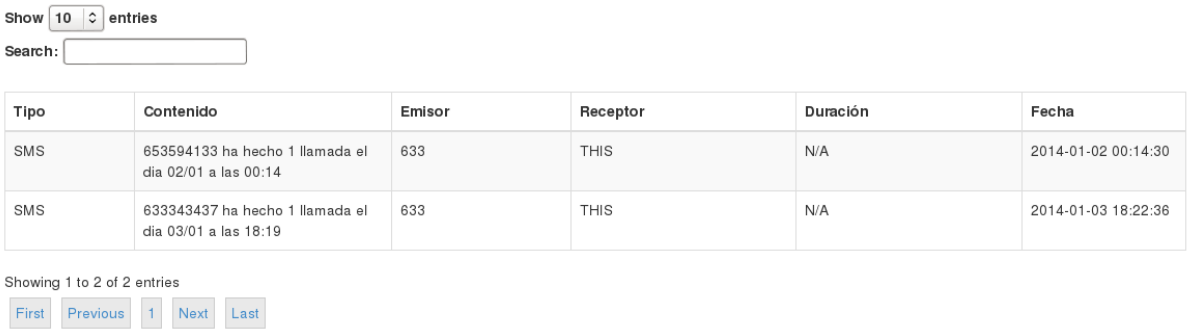


Figura 55 - Timeline filtrado por fechas

Se ven las siguientes conversaciones de WhatsApp:

Figura 56 - Conversaciones de WhatsApp filtradas por fecha

En la tabla de Comunicaciones vemos dos registros:

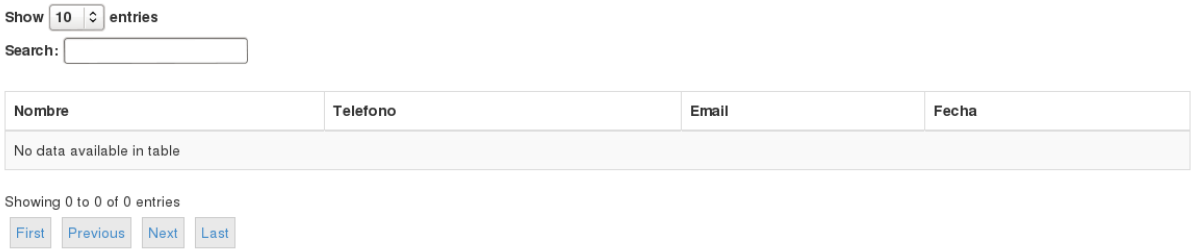

Show 10 entries
Search:

Tipo	Contenido	Emisor	Receptor	Duración	Fecha
SMS	653594133 ha hecho 1 llamada el dia 02/01 a las 00:14	633	THIS	N/A	2014-01-02 00:14:30
SMS	633343437 ha hecho 1 llamada el dia 03/01 a las 18:19	633	THIS	N/A	2014-01-03 18:22:36

Showing 1 to 2 of 2 entries
First Previous 1 Next Last

Figura 57 - Comunicaciones filtradas por fecha

No se han podido recuperar los contactos:


Show 10 entries
Search:

Nombre	Telefono	Email	Fecha
No data available in table			

Showing 0 to 0 of 0 entries
First Previous Next Last

Figura 58 - Contactos filtrados por fechas

En la sección de Agenda se observa un registro:

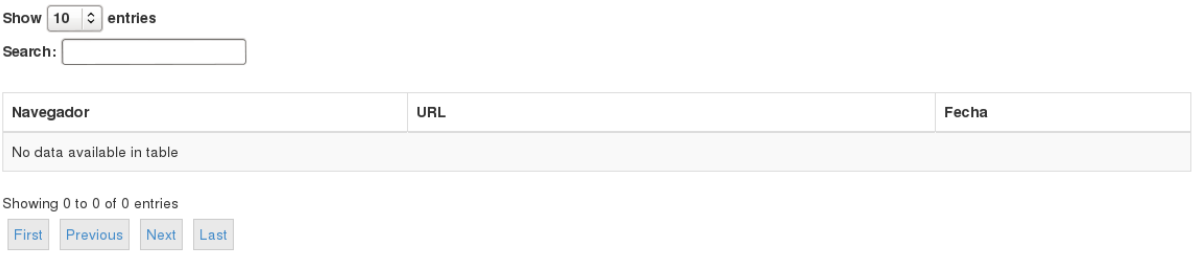

Show 10 entries
Search:

Titulo	Contenido	Fecha
Presentacion del TFM	Exposicion presencial del TFM	2014-01-01 01:05:48

Showing 1 to 1 of 1 entries
First Previous 1 Next Last

Figura 59 - Agenda filtrada por fechas

Sin embargo, la sección de historial está vacía:


Show 10 entries
Search:

Navegador	URL	Fecha
No data available in table		

Showing 0 to 0 of 0 entries
First Previous Next Last

Figura 60 - Historial web filtrado por fechas

El informe PDF muestra el siguiente contenido:

Informe de evidencias

Timeline

Fecha	Evento
2014-01-01	Calendario: Presentacion del TFM
2014-01-01	Conversacion con Tutor TFM
2014-01-02	SMS desde 633
2014-01-03	Conversacion con Iker Latinos
2014-01-03	Conversacion con Carmen Latinos
2014-01-03	Conversacion con Tutor Unir
2014-01-03	Conversacion con Grupo Amigos
2014-01-03	SMS desde 633

Figura 61 - Informe PDF filtrado por fechas (Timeline)

Conversaciones Whatsapp

Tutor TFM		
Fecha	**Emisor**	**Mensaje**
2014-01-01 01:01:05	THIS	Hola
2014-01-01 01:01:26	THIS	Solo tal y genero un backup con datos de prueba para poder utilizarlos en la presentacion
2014-01-01 01:01:29	THIS	Ya tengo la herramienta terminada
2014-01-01 01:06:00	Tutor TFM	Porque no utilizas datos reales?
2014-01-01 02:09:11	THIS	Porque el backup con el que he hecho el desarrollo es de un dispositivo que no es mio
2014-01-01 02:09:14	THIS	Es de un familiar
2014-01-01 02:09:18	THIS	Y no quiero mostrar datos privados en la presentacion de la herramienta
2014-01-01 02:01:28	Tutor TFM	OK
2014-01-01 02:01:33	Tutor TFM	Considero que el funcionamiento debería ser suficiente

Grupo Amigos		
Fecha	**Emisor**	**Mensaje**
2014-01-03 09:58:01	Iker Latinos	Feliz Año nuevo!
2014-01-03 10:04:26	Carmen Latinos	Que tal el fin de Año?
2014-01-03 10:04:47	Carmen Latinos	Feliz Año!
2014-01-03 12:57:58	Iker Latinos	Bien
2014-01-03 12:58:41	THIS	Feliz 2014!

Tutor Unir		
Fecha	**Emisor**	**Mensaje**
2014-01-03 13:30:48	Tutor Unir	Hola
2014-01-03 13:31:26	Tutor Unir	Como llevas tu TFM?
2014-01-03 13:32:13	THIS	Preparando datos de prueba para hacer las ultimas pruebas de la herramienta y redactando la memoria
2014-01-03 13:37:06	Tutor Unir	Muchas Suerte
2014-01-03 13:40:21	THIS	Gracias

Figura 62 - Informe PDF filtrado por fechas (WhatsApp)

Comunicaciones					
Fecha	**Tipo**	**Emisor**	**Receptor**	**Duracion**	**Contenido**
2014-01-02 00:14:30	SMS	633	THIS	N/A	653594133 ha hecho 1 llamada el dia 02/01 a las 00:14
2014-01-03 18:22:36	SMS	633	THIS	N/A	633343437 ha hecho 1 llamada el dia 03/01 a las 18:19

Figura 63 - Informe PDF filtrado por fechas (Comunicaciones)

Calendario		
Fecha	**Titulo**	**Contenido**
2014-01-01 01:05:48	Presentacion del TFM	Exposicion presencial del TFM

Figura 64 - Informe PDF filtrado por fechas (Agenda)

Todos los otros datos son tablas vacías, ya que no se han encontrado registros coincidentes

11.3 Filtrando por tipos

También se puede filtrar por tipos, con la opción -c. Por ejemplo, en este caso se van a obtener todos los contactos, y todas las comunicaciones (Llamada y SMS). Para ello se ejecuta el comando de la siguiente imagen:

```
root@debian:/usr/local/ibaft# python ibaft_report.py -d /usr/local/backup/TFM.sqlite -o /tmp/ -c Contacto,Llamada,SMS
Creando directorio de informe....
Generando informe PDF....
Informe HTML generado en /tmp//report
Informe PDF guardado en /tmp//report/reportPDF.pdf
```

Figura 65 - Ejecución de ibaft_report.py

Si ahora vamos a /tmp/report vemos el informe generado, en el que el timeline muestra lo que se ve en la Figura 66. Se puede observar que se dispone de muchos más registros, por lo que es un informe más completo.

Además en este informe está muy claro que las secciones correspondientes a WhatsApp, Historial y Agenda no contendrán datos, ya que no se han indicado en los tipos. Así pues, sólo se han recuperado los contactos, que se pueden ver en la Figura 67, y las comunicaciones, que están representadas en la Figura 68.

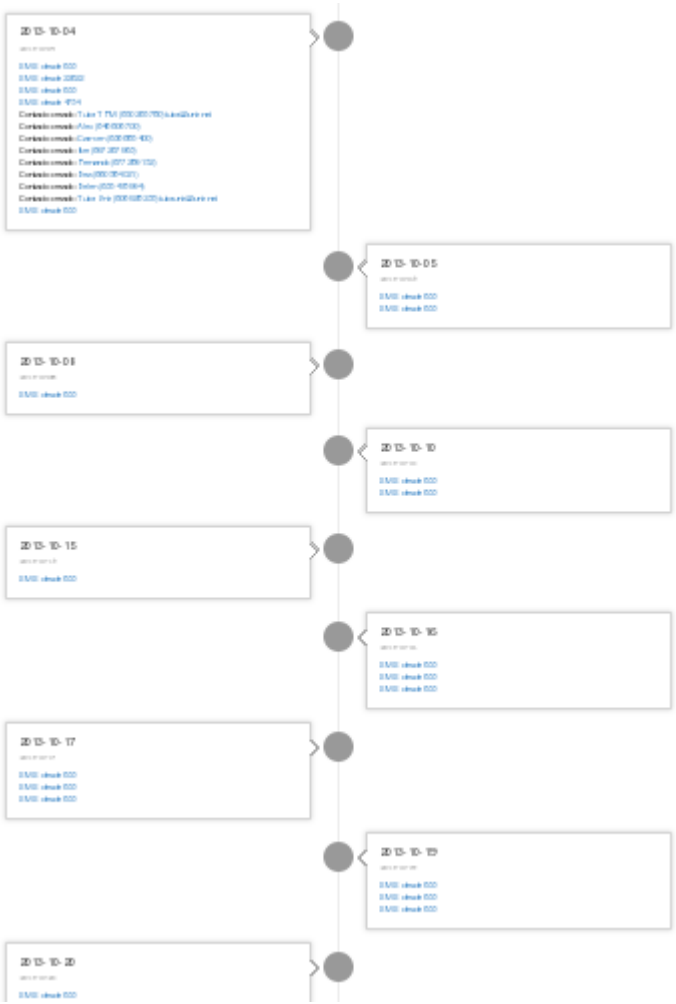


Figura 66 - Timeline filtrado por tipos

Show 10 entries

Search:

Nombre	Telefono	Email	Fecha
Alex	646 606 700		2013-10-04 13:06:02
Bea	660 584 021		2013-10-04 13:08:16
Belen	635 418 994		2013-10-04 13:09:03
Carmen	636 685 400		2013-10-04 13:06:32
Fernando	677 289 152		2013-10-04 13:07:41
Iker	697 267 963		2013-10-04 13:06:59
Tutor TFM	650 265 780	tutor@unir.net	2013-10-04 13:05:34
Tutor Unir	606 928 235	tutorunir@unir.net	2013-10-04 13:09:29

Showing 1 to 8 of 8 entries

First Previous 1 Next Last

Figura 67 - Contactos filtrados por tipo

Tipo	Contenido	Emisor	Receptor	Duración	Fecha
Llamada		THIS	655893468	N/A	2014-01-22 13:22:08
Llamada		THIS	653594133	N/A	2014-01-23 10:40:32
Llamada		THIS	646606700	N/A	2014-01-23 10:46:49
Llamada		674384404	THIS	N/A	2014-01-23 11:51:55
Llamada		650265780	THIS	N/A	2014-01-23 12:20:18
Llamada		697267963	THIS	N/A	2014-01-23 13:13:36
Llamada		THIS	615370137	N/A	2014-01-23 15:27:26
Llamada		THIS	976212084	N/A	2014-01-23 17:50:52
Llamada		THIS	976212084	N/A	2014-01-23 17:51:32
Llamada		THIS	650265780	N/A	2014-01-24 00:18:45

Showing 1 to 10 of 129 entries

First Previous 1 2 3 4 5 Next Last

Figura 68 - Comunicaciones filtradas por tipo

Se puede observar que se han obtenido 129 registros de comunicaciones pero que actualmente sólo se están mostrando 10. Se puede aumentar este número seleccionando en el desplegable de la tabla. En la Figura 69, por ejemplo, se ven 25 registros por página.

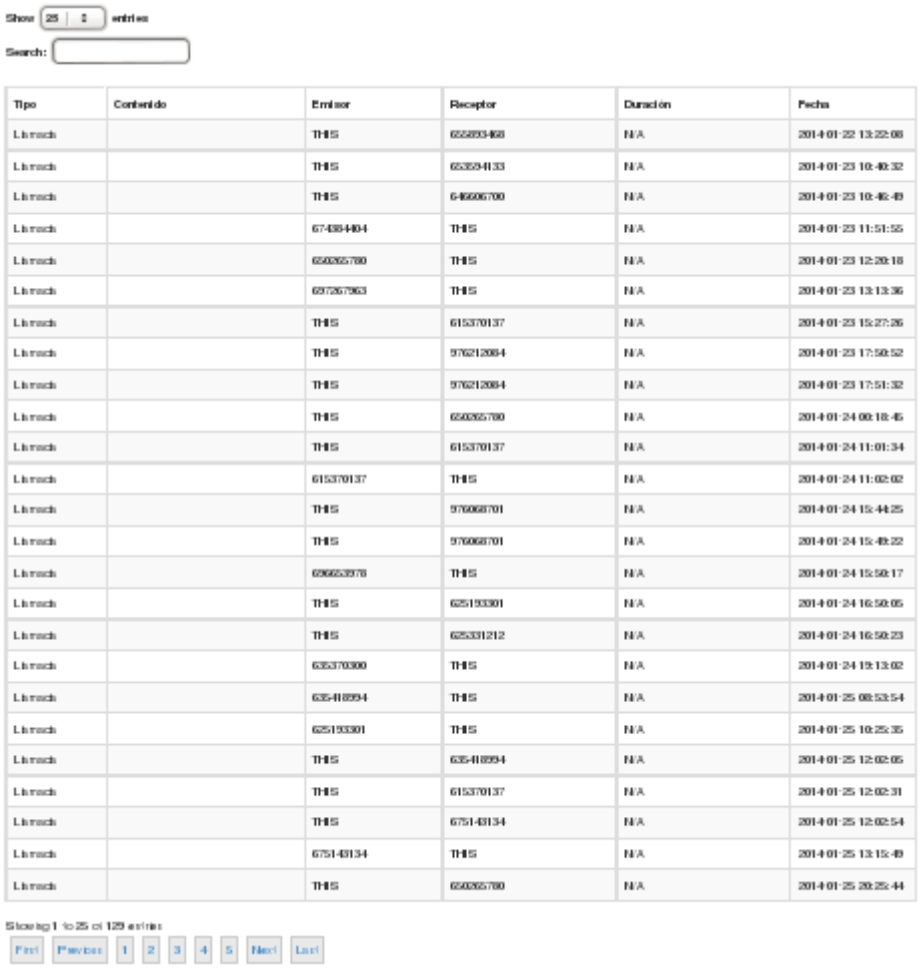


Figura 69 - Comunicaciones filtradas por tipo (25 por página)

Además, también se pueden realizar búsquedas dentro del informe. Para ello, las tablas disponen de un campo de búsqueda. Así que por ejemplo, si queremos buscar la palabra SMS, para filtrar solo los mensajes de texto, y que además contenga la palabra "Apple", obtendremos un resultado como el que se ve en la Figura 70

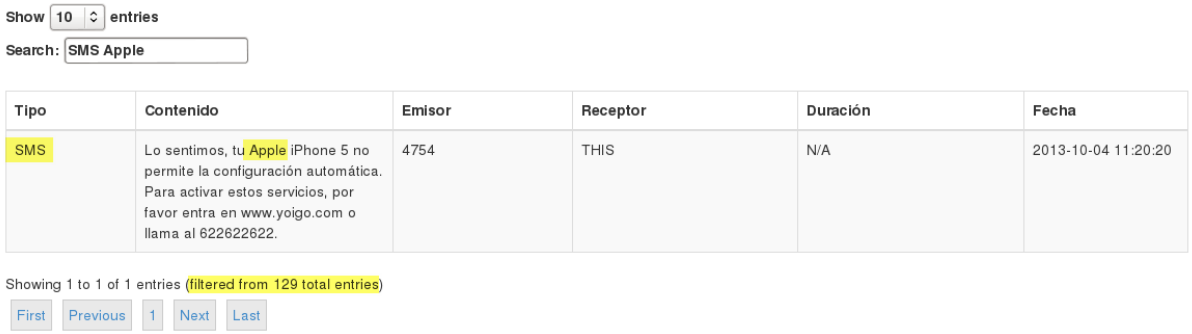


Figura 70 - Búsqueda dentro de la tabla de Comunicaciones

En el informe PDF se han generado un total de 12 páginas, mostrando sólo las comunicaciones, timeline y contactos. Esto permite hacerse una idea de la cantidad de información que se generaría si no se indicasen filtros, como se verá en el siguiente apartado.

11.4 Obteniendo toda la información

Si no se desea filtrar por nada en concreto, se puede obtener toda la información usando la opción -a, pero hay que tener en cuenta que se podría llegar a generar una cantidad ingente de información, en casos reales.

Para obtener toda la información basta con utilizar la opción -a. Si se indica esta opción, el resto de criterios de filtrado serán ignorados, y se recuperará toda la información. Para usarlo se ejecuta el comando de la siguiente imagen:

```
root@debian:/usr/local/ibaft# python ibaft_report.py -d /usr/local/backup/TFM.sqlite -o /tmp/ -a
Creando directorio de informe....
Generando informe PDF....
Informe HTML generado en /tmp//report
Informe PDF guardado en /tmp//report/reportPDF.pdf
```

Figura 71 - Ejecución de ibaft_report.py sin filtros

Ahora en /tmp/report se encuentra el informe, que muestra el siguiente timeline:

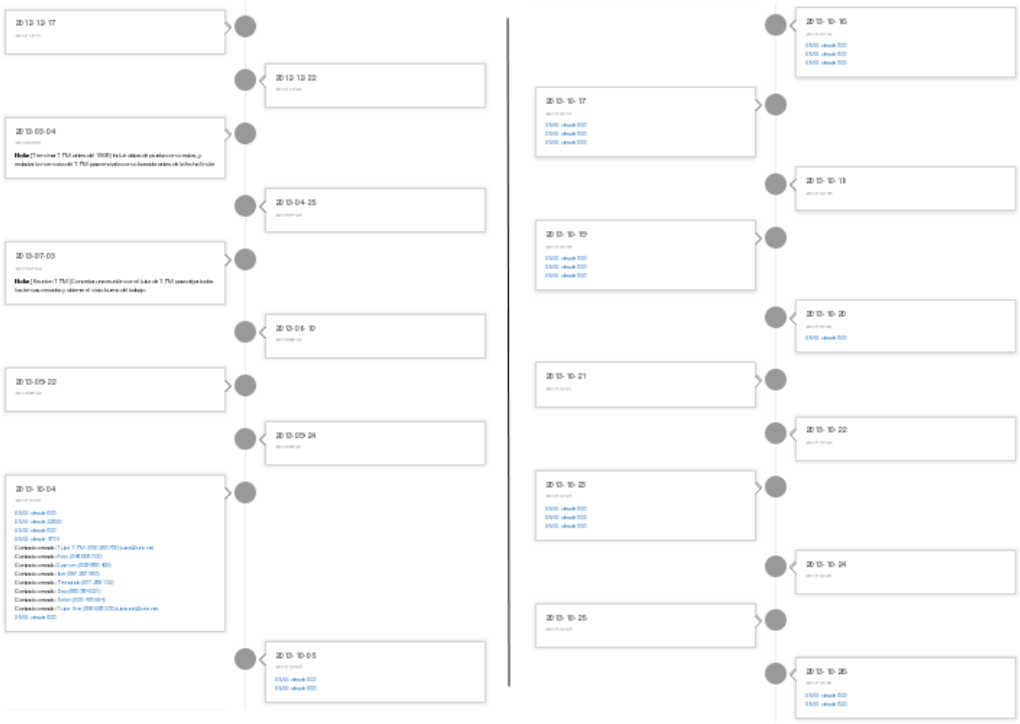


Figura 72 - Parte del timeline sin filtrar

El timeline es tan largo que no es viable mostrarlo completo en este documento. En lugar de ello, se ha capturado dos partes del mismo y se han mostrado en una misma imagen, la Figura 72.

NOTA: El timeline resultante es demasiado largo, y hay que tener en cuenta que el backup contiene datos solo de 5 meses, por lo que una vez más se demuestra que la opción -a no es la mejor decisión en una investigación real.

Lo mismo ocurre con las conversaciones de WhatsApp, aunque al estar modificadas sólo hay tres conversaciones, que se muestran en la Figura 73.

Los contactos y las comunicaciones son los mismos que los mostrados en la Figura 67 y la Figura 68 respectivamente, ya que cuando se realizó el filtrado por tipo no se indico ningún otro criterio para esos tipos.

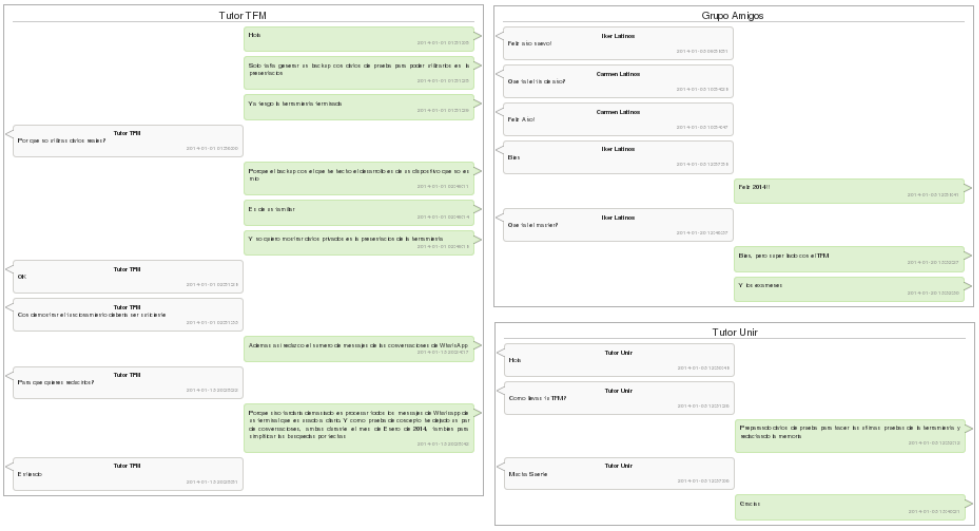


Figura 73 - Conversaciones Whatsapp

La Agenda sólo contiene un registro, por lo que su contenido es el que se muestra en la siguiente imagen:

Show 10 entries

Search:

Titulo	Contenido	Fecha
Presentacion del TFM	Exposicion presencial del TFM	2014-01-01 01:05:48

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

Figura 74 - Agenda

Se han recuperado 12 entradas del historial web, que se pueden ver en la Figura 75.

Show 25 entries

Search:

Navegador	URL	Fecha
Safari	http://m.loteriasyapuestas.es/	2014-01-24 12:14:28
Safari	http://www.idealista.com/inmueble/27253851/foto?multimediaNumber=1	2014-01-24 12:14:29
Safari	http://www.idealista.com/	2014-01-24 12:15:10
Safari	http://www.telepizza.es/productos/promociones	2014-01-29 19:37:08
Safari	https://www.ono.es/sitemovil/clientes/cambiar-password/	2014-02-09 13:05:40
Safari	https://www.ono.es/sitemovil/error/	2014-02-09 13:06:12
Safari	https://www.ono.es/sitemovil/clientes/cambiar-password/	2014-02-09 13:06:28
Safari	https://www.ono.es/sitemovil/clientes/cambiar-password/#	2014-02-09 13:12:11
Safari	https://www.ono.es/sitemovil/clientes/cambiar-password/	2014-02-10 18:07:17
Safari	https://www.google.es/	2014-02-11 01:50:11
Safari	http://www.crea.es/inicio.htm	2014-02-13 09:16:16
Safari	https://iforgot.apple.com/password/reset	2014-02-13 17:28:21

Showing 1 to 12 of 12 entries

First Previous 1 Next Last

Figura 75 - Historial web

En el informe PDF se han generado un total de 14 páginas que contiene todos los datos mostrados anteriormente.

Así pues, el investigador puede obtener la información del dispositivo en baso a los criterios que considere oportunos, para poder visualizar los posibles indicios de evidencias, y cuando encuentre algo que le parezca interesante, poder seguir la investigación profundizando en ello.

12. CONCLUSIONES

Desde el principio, el desarrollo de esta herramienta ha tenido como objetivo personal, hacerme profundizar en la arquitectura de los sistemas iOS, que son los que en menor medida conocía.

Además de adquirir conocimientos sobre estos sistemas, quería que la herramienta resultante fuese útil de alguna manera. Quizás esta herramienta no esté a la altura de suites forenses como EnCase, Oxygen Forensic o similares, pero está desarrollada para ser adaptable y flexible, y es de código abierto.

Debido al escaso tiempo para el desarrollo de un Trabajo de Fin de Máster, la herramienta no cuenta, en su primera versión, con un gran número de plugins, pero permite su extensión mediante el desarrollo de nuevos plugins, lo que la hace muy flexible a la rápida evolución de aplicaciones móviles.

Gracias a su estructura modular y su sistema de plugins, la herramienta puede ir creciendo para cubrir las necesidades de cada usuario, y puede servir como alternativa a otro tipo de herramientas comerciales.

El objetivo principal del desarrollo era que fuese flexible y que aportase valor al investigador, permitiéndole ahorrar tiempo en la tarea de obtención y análisis de evidencias. Este objetivo se ha cumplido gracias al uso de plugins para aumentar funcionalidades, y a la automatización de la obtención de evidencias y generación de informes atractivos que permiten una investigación más eficiente.

Revisando las características, se puede observar que se han cumplido, ya que es una herramienta Gratuita y Open Source, que además es multiplataforma y portable, por lo que se puede transportar fácilmente en un USB y ejecutarla en cualquier máquina, independientemente del sistema operativo. Además, es una herramienta flexible, que permite la extensión de funcionalidades a través de plugins, que pueden desarrollar los investigadores para adaptar la herramienta a sus necesidades. También genera diferentes formatos de informe (en pdf y html), que están orientado a dos funciones distintas. El informe html se centra en la visualización de la información, con un diseño atractivo, que permite procesar la información más fácilmente, por lo que reduce el tiempo necesario. Sin embargo, el informe pdf está diseñado para presentar los datos en formato impreso.

El hecho de que se cumplan todas las características que se marcaron como objetivo produce la principal contribución: el desarrollo de una herramienta forense que aporta valor a las investigaciones de manera gratuita y libre, y que permite una flexibilidad para adaptarse a la constante evolución de la tecnología.

Comparando la herramienta desarrollada con las analizadas en el Capítulo 3, se podría decir que se ha construido una herramienta capaz de realizar análisis tan exhaustivos como los de Oxygen Forensic, pero con las características de gratuidad y portabilidad de iPhone Analyzer. Además, dispone de la suficiente flexibilidad como para adaptarse a la constante evolución de los dispositivos móviles, por lo que un investigador no tiene que esperar a que se lance una nueva versión de la herramienta para poder incluir el análisis de una determinada aplicación en su investigación, sino que él mismo puede desarrollar un plugin para incluir ese análisis.

Algunas de las características anteriores se basan en una simple decisión, como el hecho de que sea gratuita y de código abierto, pero otras requieren un desarrollo técnico, como la flexibilidad, que es la más importante, ya que permite a los usuarios adaptar la herramienta para cada caso particular.

Personalmente pienso que una herramienta de este tipo, con los plugins necesarios, puede ser de gran utilidad en un análisis forense, sobre todo si se realiza a partir de un backup (siempre será más efectivo acceder directamente al teléfono, o a una copia forense de éste), y se me ocurren muchos posibles futuros proyectos.

13. TRABAJO FUTURO

La principal línea de desarrollo es crear nuevos plugins para las aplicaciones populares. Plugins muy interesantes son, por ejemplo, los dedicados a obtener la información de las redes sociales y mensajería instantánea.

Como aplicaciones de mensajería instantánea, Line y Telegram están ganado cuota de mercado a WhatsApp, por lo que es interesante el desarrollo de un plugin que permita la extracción y análisis de las evidencias de dichas aplicaciones.

Las redes sociales también están en auge y, a pesar de que existen un gran número de ellas, las más populares son Twitter y Facebook, por lo que el desarrollo de estos plugins cobran especial importancia.

En el caso de que el desarrollo de los plugins anteriormente mencionados difiriera dependiendo de la versión de la aplicación, se debería desarrollar plugins para distintas versiones, garantizando así que funcionará correctamente independientemente de la versión de la aplicación que tenga instalada el dispositivo.

Como nueva funcionalidad, no tan prioritaria, se podría mencionar el desarrollo de una interfaz gráfica para hacer más atractiva la herramienta.

También se puede continuar el desarrollo a partir de *forks* de la herramienta, que generasen nuevas herramientas que cubran otras necesidades, como por ejemplo, la creación de un sitio web que permita subir el backup de un dispositivo, y se analice la información en la nube, mostrando un informe de evidencias.

Otro proyecto interesante podría ser el desarrollo de una aplicación nativa de iOS que realizara esto mismo, y subiera los datos al servidor, para analizarlos y mandar un informe por correo electrónico.

Cuando la herramienta esté suficientemente madura, como para poder analizar toda la superficie de un dispositivo móvil y presentar los datos en un proceso judicial, se le podría añadir funcionalidades de firma digital para poder verificar la autenticidad de las evidencias.

Como se puede observar, en este campo existen muchos desarrollos posibles, y de gran utilidad, pero que lamentablemente he tenido que dejar fuera de este Trabajo, por no disponer del tiempo suficiente.

14. BIBLIOGRAFÍA

Actualidad Iphone. (2010). AddressBook.sqlitedb. Recuperado de http://www.actualidadiphone.com/foro/dudas-y-problemas/14565-addressbook-sqlitedb.html

Apple. (2014a). Mac OS X: Límites de los archivos y volúmenes del formato de Mac OS Plus (HFS Plus). Recuperado de http://support.apple.com/kb/HT2422?viewlocale=es_ES&locale=es_ES

DBSnippets. (2012). Cifrado simétrico bajo IOS (parte 1 : Teórica). Recuperado de http://www.dbsnippets.com/2012/09/30/cifrado-simetrico-bajo-ios-parte-1-teorica/

Eguía, A. (2013). Análisis forense en dispositivos Android y Apple (iOS). Recuperado de http://www.spamloco.net/2013/06/analisis-forense-en-dispositivos-android-y-apple.html

NorfiPC. (2014). Rutas y ubicación de los archivos, datos e información en el iPhone. Recuperado de http://norfipc.com/celulares/rutas-ubicacion-archivos-datos-informacion-iphone.html

Oxygen Forensics. (2000). Oxygen Forensic® Suite - Features. Recuperado de http://www.oxygen-forensic.com/en/features

Porras, E. (2012). Ingenieria de sistemas: Sistemas operativos móviles: iOS. Recuperado de http://eve-ingsistemas-u.blogspot.com.es/2012/04/sistemas-operativos-moviles-ios.html

Proffitt, T. (2012).Forensic Analysis on iOS. Recuperado de. http://www.sans.org/reading-room/whitepapers/forensics/forensic-analysis-ios-devices-34092

Ramos, A. (2012a). Recuperar la contraseña de un backup de iTunes - Security By Default. Recuperado de http://www.securitybydefault.com/2012/04/recuperar-la-contrasena-de-un-backup-de.html

Ramos, A. (2014). Descifrando msgstore.db.crypt5, la nueva base de datos de WhatsApp - Security By Default. Recuperado de http://www.securitybydefault.com/2014/03/descifrando-msgstoredbcrypt5-la-nueva.html

Rebollo, C. (2013a). Análisis forense iPhone – Parte I – Análisis del Backup (1a parte). Recuperado de http://highsec.es/2013/07/analisis-forense-iphone-analisis-del-backup-1a-parte/

Rebollo, C. (2013b). Análisis forense iPhone – Parte II – Análisis del Backup (2a parte). Recuperado de. http://highsec.es/2013/08/analisis-forense-iphone-analisis-del-backup-2a-parte/

Restrepo, J.A. (2011). Taller: Análisis forense de dispositivos iOS. Recuperado de http://www.slideshare.net/dragonjar/taller-analisis-forense-de-dispositivos-ios

Rodriguez, J. (2011). Estructura de directorios y su acceso en iOS. Recuperado de http://programitis.blogspot.com.es/2011/01/estructura-de-directorios-y-su-acceso.html

Seguridad Apple. (2010). Seguridad Apple: Oxygen Forensic Suite: Análisis Forense de iPhone. Recuperado de http://www.seguridadapple.com/2010/11/oxygen-forensic-suite-analisis-forenses.html

Seguridad Apple. (2013). Seguridad Apple: Obtener base de datos de mensajes WhatsApp en iPhone. Recuperado de http://www.seguridadapple.com/2013/02/obtener-base-de-datos-de-mensajes.html

Seguridad Apple. (2014). Seguridad Apple: Almacenamiento de mensajes SMS e iMessage en iOS. Recuperado de http://www.seguridadapple.com/2014/05/almacenamiento-de-mensajes-sms-e.html

15. REFERENCIAS

[1]. Apple. (2014b). iTunes: Acerca de las copias de seguridad de iOS. Recuperado de http://support.apple.com/kb/HT4946?viewlocale=es_ES

[2]. Cisco. (2014). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018. Recuperado de http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf

[3]. Ramos, A. (2012b). Seguridad en IOS: Análisis del fichero binario Manifest.mbdb - Security By Default. Recuperado de http://www.securitybydefault.com/2012/12/seguridad-en-ios-analisis-del-fichero.html

[4]. Satish B. (2012). iPhone Forensics – Analysis of iOS 5 backups : Part 1 - InfoSec Institute. Recuperado de http://resources.infosecinstitute.com/ios-5-backups-part-1/