CREACIÓN DE UNA APLICACIÓN DE ANÁLISIS FORENSE EN JAVA

UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

PROYECTO FIN DE CARRERA

INGENIERÍA INFORMÁTICA SUPERIOR

Autor: *Armando Machuca Llorente*
Tutor: *Jorge Blasco Alis*
Año: *2009*

Agradecimientos

Me gustaría agradecer, en primer lugar a mi familia, mi padre, madre y hermano, por empujarme constantemente a realizar el proyecto e inculcarme lo necesario que es ampliar conocimientos para crecer como persona. También por haberme facilitado estudiar en las mejores condiciones posibles, sin reparar en gastos ni esfuerzos en mi educación. Si no fuera por su insistencia es posible que hubiera tardado muchos años o incluso nunca hubiera acabado la Carrera.

También quiero agradecer su interés y entusiasmo en el proyecto a Jorge Blasco Alis, tutor del mismo, que ha contestado rápidamente a todas mis peticiones y ha sido capaz de releer y revisar la memoria más veces de las que yo sería capaz.

Por otro lado, me gustaría agradecer a todos los profesores, que durante toda la carrera han contribuido a mi aprendizaje y han mostrado interés y apoyo. Tampoco puedo olvidar a todos los compañeros con los que he compartido apuntes y largas prácticas.

Por último me gustaría expresar mi agradecimiento especialmente a una persona, Ainhoa, por sus atenciones y apoyo incondicional.

Índice

1. INDICE DE TABLAS Y FIGURAS 6
   1.1. Indice de tablas 6
   1.2. Indice de Figuras 6
2. RESUMEN 8
3. INTRODUCCIÓN 9
   3.1. Objetivos del Presente Proyecto 9
   3.2. Organización del presente documento 10
4. GESTIÓN DEL PROYECTO 12
   4.1. Planificación 12
   4.1.1. Planificación inicial. Ciclo de vida 12
   4.1.2. Planificación final(ciclo de vida) 13
   4.2. Medios técnicos empleados para el proyecto 14
   4.2.1. Herramientas Hardware 15
   4.2.2. Herramientas Software 15
   4.2.2.1. Sistemas operativos 15
   4.2.2.2. Herramientas 15
   4.2.2.3. Librerías Java 15
   4.2.2.4. Suites informáticas 16
   4.3. Análisis económico del proyecto 17
   4.3.1. Metodología de estimación de costes 17
   4.3.2. Presupuesto inicial 18
   4.3.2.1. Costes asociados a recursos humanos 18
   4.3.2.2. Costes asociados a medios 19
   4.3.2.3. Costes asociados a licencias software 19
   4.3.2.4. Resumen de costes inicial 20
   4.3.3. Presupuesto final 20
   4.3.3.1. Costes asociados a recursos humanos 20
   4.3.3.2. Costes asociados a licencias software 21
   4.3.3.3. Costes asociados a recursos Hardware 21
   4.3.3.4. Resumen de costes final 22
   4.3.4. Comparativa de costes 22
   4.3.5. Análisis de la licencia usada: GNU GPL v3 22
5. ANÁLISIS 24
   5.1. Propósito del plan 24
   5.2. Ambito de la aplicación: Informática Forense 24
   5.2.1. Fases del análisis 24
   5.2.2. Software actual de informática forense 26
   5.2.3. Cuestiones legales. 28
   5.2.3.1. Confidencialidad 28
   5.2.3.2. Requerimientos legales de las evidencias 28
   5.2.3.3. Autoría de los resultados 29
   5.2.4. Área de aplicación del proyecto 30
   5.3. Análisis del efecto del software en los sistemas 30
   5.3.1. Instalaciones en el Sistema Operativo Windows 31
   5.3.1.1. Preparación 31
   5.3.1.2. Caracteristicas del Sistema operativo 31
   5.3.1.3. Procedimiento 34

5.3.1.4. PRUEBA 1:CAMOUFLAGE 34
5.3.1.5. Prueba 2: MP3Stego 35
5.3.1.6. PRUEBA 3: TOR 36
5.3.1.7. Conclusión 37
5.3.1.8. Enfoque 38
5.3.2. Análisis de la instalación de Programas en Linux 39
5.3.2.1. Propósito 39
5.3.2.2. Software utilizado 39
5.3.2.3. Información adicional 40
5.3.2.4. Prueba 1: StegHide 41
5.3.2.5. Prueba 2: GNUPG 43
5.3.2.6. Conclusión 45
5.3.2.7. Enfoque 45
5.3.3. Análisis de la instalación de Programas en MAC OS 46
5.3.3.1. Preparación 46
5.3.3.2. Sistema operativo 46
5.3.3.3. Información adicional 47
5.3.3.4. PRUEBA 1: GNUPG 49
5.3.3.5. PRUEBA 2: CRYPTIX 49
5.3.3.6. PRUEBA 3: XAES 50
5.3.3.7. Conclusión 50
5.3.3.8. Enfoque 51
5.4. Definición del Sistema 51
5.4.1. Determinación del Alcance del sistema 51
5.4.2. Especificación de Estándares y Normas 51
5.4.3. Restricciones Generales 52
5.4.4. Supuestos y Dependencias 53
5.4.5. Entorno Operacional 54
5.4.5.1. Identificación de los Usuarios y Participantes finales 54
5.5. Establecimiento de Requisitos Software 54
5.5.1. Especificación de Casos de Uso. 54
5.5.1.1. Diagrama de Casos de Uso. 55
5.5.1.2. Descripción textual de los Casos de Uso 57
5.5.1.3. Obtención de Requisitos 58
5.5.1.4. Requisitos Funcionales (f) 58
5.5.1.5. Requisitos de rendimiento (r) 63
5.5.1.6. Requisitos de interfaz 65
5.5.1.7. Requisitos operacionales (o) 67
5.5.1.8. Requisitos de recursos (RE) 72
5.5.1.9. Requisitos de restricción (rs) 74
5.5.1.10. Requisitos de manejo de errores (e) 75
5.5.1.11. Requisitos de documentación (d) 76
5.5.1.12. Requisitos de portabilidad (p) 77
5.5.1.13. Requisitos de calidad (c) 77
5.5.2. Diseño del plan de Pruebas de aceptación 80
5.5.3. Análisis de los Casos de Uso. 81
5.5.3.1. Análisis del sistema donde está ejecutando la aplicación 82
5.5.3.2. Análisis de una unidad extraíble 83
5.5.3.3. Búsqueda de un análisis anterior 84
6. DISEÑO 85
6.1. Propósito 85
6.2. Visión general del sistema 85
6.3. Contexto del sistema 85

6.3.1. Ficheros relacionados 86
6.3.1.1. Plugins 86
6.3.1.2. Informes 90
6.4. Diseño preliminar del sistema 91
6.4.1. Componentes del sistema 91
6.4.2. Componentes 92
6.4.3. Componente Vista 93
6.4.3.1. Pantalla inicial 94
6.4.3.2. Análisis de unidades concretas 94
6.4.3.3. Selección de detectores: 95
6.4.3.4. Proceso de análisis: 96
6.4.3.5. Informe del análisis: 96
6.4.3.6. Ayuda: 97
6.4.4. Diagrama de actividad 97
6.5. Diseño Detallado 98
6.5.1. Diseño de clases UML Completo 98
6.5.2. Componente Modelo 101
6.5.3. Componente Controlador 102
6.5.4. Componente Vista 104
7. IMPLEMENTACIÓN E IMPLANTACIÓN DEL SOFTWARE 106
7.1. Proceso de codificación 106
7.2. Problemas encontrados y soluciones adoptadas 110
7.3. Ejecución del plan de pruebas de la aplicación 112
8. CONCLUSIÓN Y LÍNEAS FUTURAS 113
8.1. Conclusiones del proyecto 113
8.1.1. Dificultades del proyecto 113
8.1.2. Resultados obtenidos 114
8.1.3. Desarrollo del proyecto 114
8.2. Ejemplos de implantación 115
8.2.1. Caso 1: Violaciones de la política empresarial 115
8.2.2. Caso 2: Delitos informáticos generados con determinadas aplicaciones 115
8.2.3. Caso 3: Comprobación de incompatibilidades entre determinado software 115
8.2.4. Caso 4: Análisis de software previo a una posible migración 116
8.3. Líneas Futuras 116
9. BIBLIOGRAFÍA 119
9.1. Recursos electrónicos 119
9.2. Libros de consulta 121
9.3. Apuntes 121
10. GLOSARIO DE TÉRMINOS 123
ANEXO 2: MANUAL DE INSTALACIÓN 126
ANEXO 3: MANUAL DE USO 130
ANEXO 4: MANUAL DE CREACIÓN DE PLUGINS 136
1. Introducción 136
2. Técnicas implementadas 136
2.1. Técnicas relacionadas con búsquedas en disco 136
2.2. Técnicas relacionadas con búsquedas en el registro 137
2.3. Técnicas relacionadas con búsquedas en el registro 137
3. Obtención de datos 137
4. Estructura del plugin 139
ANEXO 5: MANUAL DE CREACIÓN DE NUEVAS TÉCNICAS 142

1. Índice de tablas y figuras

1.1. Índice de tablas

TABLA 1 OBJETIVOS DEL PRESENTE PROYECTO 9
TABLA 2. COSTES INICIALES ASOCIADOS A RECURSOS HUMANOS 19
TABLA 3. COSTES ASOCIADOS A COMPONENTES HARDWARE 19
TABLA 4. COSTES ASOCIADOS A LICENCIAS SOFTWARE 19
TABLA 5. ESTIMACIÓN INICIAL DE COSTES 20
TABLA 6. COSTES FINALES ASOCIADOS A RECURSOS HUMANOS 20
TABLA 7. COSTES FINALES ASOCIADOS A LICENCIAS SOFTWARE 21
TABLA 8. COSTES FINALES ASOCIADOS A RECURSOS HARDWARE 21
TABLA 9. RESUMEN FINAL DE COSTES 22
TABLA 10. SECCIONES DEL REGISTRO DE WINDOWS 33
TABLA 11. DESCRIPCIÓN DE LAS SECCIONES DEL REGISTRO 34
TABLA 12. INFORMACIÓN DEL SOFTWARE CAMOUFLAGE 34
TABLA 13. INFORMACIÓN DEL SOFTWARE MP3STEGO 36
TABLA 14. INFORMACIÓN DEL SOFTWARE TOR 36
TABLA 15. INFORMACIÓN DEL SOFTWARE STEGHIDE 42
TABLA 16. INFORMACIÓN DEL SOFTWARE GNUPG 43
TABLA 17. INFORMACIÓN DEL SOFTWARE GNUPG 49
TABLA 18. INFORMACIÓN DEL SOFTWARE CRYPTIX 49
TABLA 19. INFORMACIÓN DEL SOFTWARE XAES 50
TABLA 20. RESTRICCIONES DE PRIORIDAD BAJA 52
TABLA 21. RESTRICCIONES DE PRIORIDAD MEDIA 52
TABLA 22. RESTRICCIONES DE PRIORIDAD ALTA 53
TABLA 23. PRUEBAS DE ACEPTACIÓN 81
TABLA 24. RESULTADO DE LAS PRUEBAS DE ACEPTACIÓN 112
TABLA 25. CONJUNTO DE PLUGINS IMPLEMENTADOS 135

1.2. Índice de Gráficos

GRÁFICO 1. PLANIFICACIÓN INICIAL (DIAGRAMA DE GANTT) 13
GRÁFICO 2. PLANIFICACIÓN FINAL (DIAGRAMA DE GANTT) 14
GRÁFICO 3. CAPTURA DE LA PÁGINA DEL PROYECTO EN GOOGLE CODE 17
GRÁFICO 4. DEPENDENCIAS DEL SISTEMA 53
GRÁFICO 5. DIAGRAMA DE CASO DE USO. FUNCIONALIDADES. 56
GRÁFICO 6. ANÁLISIS DEL SISTEMA DONDE SE EJECUTA LA APLICACIÓN 82
GRÁFICO 7. DIAGRAMA DEL ANÁLISIS DE UNA UNIDAD EXTRAÍBLE 83
GRÁFICO 8. DIAGRAMA DE BÚSQUEDA DE UN ANÁLISIS ANTERIOR 84
GRÁFICO 9. ESQUEMA DE UN PLUGIN 86
GRÁFICO 10. COMPONENTES DEL SISTEMA 92
GRÁFICO 11. PANTALLA INICIAL 94
GRÁFICO 12. ANÁLISIS DE UNIDADES CONCRETAS 94
GRÁFICO 13. SELECCIÓN DE DETECTORES 95
GRÁFICO 14. PROCESO DE ANÁLISIS 96
GRÁFICO 15. INFORME DEL ANÁLISIS 96
GRÁFICO 16. PANTALLA DE AYUDA 97

GRÁFICO 17. DIAGRAMA DE ACTIVIDAD 98
GRÁFICO 18. DIAGRAMA DE CLASES DE LA APLICACIÓN 100
GRÁFICO 19. DIAGRAMA UML DE COMPONENTE MODELO 101
GRÁFICO 20. DIAGRAMA DE CLASES DEL PAQUETE CONTROLADOR 103
GRÁFICO 21. DIAGRAMA DE CLASES DEL COMPONENTE VISTA 105
GRÁFICO 22. EJEMPLO DE RELACIÓN ENTRE PARÁMETROS Y TÉCNICAS/ACCIONES 107
GRÁFICO 23. ESTRUCTURA DE LOS PLUGINS 107
GRÁFICO 24. DESCARGA DE JAVA (JRE) 126
GRÁFICO 26. PROGRESO DE LA INSTALACIÓN 127
GRÁFICO 25. COMIENZO DE LA INSTALACIÓN 127
GRÁFICO 27. FINALIZACIÓN DE LA INSTALACIÓN DE JAVA 128
GRÁFICO 28. EJECUCIÓN DE LA APLICACIÓN 129
GRÁFICO 29. ARCHIVOS DE LA APLICACIÓN 129
GRÁFICO 30. EJECUCIÓN DE LA APLICACIÓN 130
GRÁFICO 31. SELECCIÓN DE UNIDADES 131
GRÁFICO 32. SELECCIÓN DE PLUGINS 131
GRÁFICO 33. EJECUCIÓN DEL ANÁLISIS 132
GRÁFICO 34. SELECCIÓN DE ACCIONES 133
GRÁFICO 35. SELECCIÓN DE FICHERO PDF DE SALIDA 133
GRÁFICO 36. PANTALLA FINAL 134
GRÁFICO 37. INFORME FINAL DE RESULTADOS 134

2. Resumen

El Proyecto de fin de Carrera (PFC) "Creación de una Aplicación de Análisis Forense" abarca el proceso de elaboración del software AFA, que permite reconocer si ciertas aplicaciones han sido instaladas en un sistema.

Este software multiplataforma permite ampliar la funcionalidad del mismo mediante plugins que definen los posibles restos que deben buscarse para cada programa, de manera que, de una forma rápida y sencilla es posible definir búsquedas para distintos tipos de software.

La aplicación genera informes en PDF que, opcionalmente podrán ser firmados digitalmente para garantizar la autoría del análisis.

También se pueden elegir distintos niveles de profundidad para el análisis, permitiendo generar análisis más profundos (en consecuencia más lentos) o menos rigurosos (más rápidos).

Las principales actividades, a grandes rasgos, de este proyecto, son:

- Analizar, diseñar e implementar una aplicación de Análisis Forense que permita identificar si una o varias aplicaciones han sido instaladas en algún momento en un disco.
- Implementar ejecución multiplataforma
- Estudiar las tecnologías existentes.
- Recopilar un conjunto de requisitos que definan la aplicación por completo.
- Realizar un diseño fácilmente ampliable y reutilizable.
- Definir los formatos que tendrán los archivos tanto de entrada como de salida que manejará la aplicación.
- Diseñar una interfaz intuitiva que permita a todo tipo de usuarios realizar un análisis de manera sencilla.
- Generar documentación suficiente y manuales que permitan ampliar el ciclo de vida del software.

3. Introducción

El Proyecto de Fin de Carrera (PFC) "Aplicación de Análisis Forense" abarca el proceso de elaboración del software AFA que permite encontrar restos de instalaciones de distintas aplicaciones. En primer lugar se describen los principales objetivos que se desea alcanzar con la realización de este proyecto y en segundo lugar se expone la organización de los contenidos de este documento.

3.1. Objetivos del Presente Proyecto

En la siguiente lista de objetivos se hace referencia a los objetivos globales a los que queremos llegar tras la realización del proyecto, que comprende la implementación del software así como el actual documento, en el que se plasman todos los detalles referentes al mismo. Los objetivos se exponen en la siguiente tabla.

Identificador del Objetivo	Descripción del objetivo
OBJ-1	Analizar, diseñar e implementar una aplicación de Análisis Forense que permita identificar si una o varias aplicaciones han sido instaladas en algún momento en un disco.
OBJ-2	Implementar distintos métodos de ejecución que permitan el funcionamiento bajo los principales sistemas operativos de ámbito general sin necesidad de realizar configuraciones previas.
OBJ-3	Estudiar las tecnologías existentes para implementar las distintas capacidades del software, los cambios que realizan los programas sobre los distintos sistemas operativos y las alternativas actuales que hay para obtener los mismos resultados que con la aplicación desarrollada.
OBJ-4	Realizar un diseño fácilmente ampliable y reutilizable.
OBJ-5	Definir los formatos que tendrán los archivos tanto de entrada como de salida que manejará la aplicación.
OBJ-6	Diseñar una interfaz intuitiva que permita a todo tipo de usuarios realizar un análisis de manera sencilla.
OBJ-7	Generar documentación suficiente y manuales que permitan ampliar el ciclo de vida del software con la colaboración de otros usuarios y permitan solucionar todo tipo de dudas sobre el uso del mismo.

Tabla 1 Objetivos del presente proyecto

3.2. Organización del presente documento

Las distintas etapas de elaboración del proyecto se encuentran reflejadas dentro del presente documento en distintos capítulos. A continuación se detalla el nombre y descripción de los mismos.

- **Introducción**: En este capítulo se describe el alcance del documento resaltando el propósito y los objetivos de alto nivel que se pretenden alcanzar con la creación del software.

- **Gestión del proyecto:** En este capítulo se realiza un estudio sobre su planificación y se comparara con los tiempos de desarrollo reales. También se realiza un pequeño análisis de costes y se analizan los aspectos legales de la licencia que usará el software.

- **Análisis:** en este capítulo se abordan los estudios necesarios que permiten alcanzar un conocimiento suficiente del problema. En primer lugar se realiza un análisis de las tecnologías existentes para la elaboración de un proyecto de estas características. Por otro lado, se realizará un estudio del resultado de la instalación y desinstalación de herramientas en los distintos sistemas operativos.
  Por último se estudia el patrón arquitectónico que servirá de base para diseñar los distintos elementos software. Concluyendo este capítulo, se describirá cual debe ser el comportamiento del sistema y se expondrán los requisitos del software.

- **Diseño detallado:** en este capítulo se identifican y diseñan los diversos elementos software del sistema, describiendo detalladamente sus partes. También se describe el modelo lógico de datos, de acuerdo a las plataformas tecnológicas elegidas en la etapa de Análisis.

- **Implementación e implantación del software:** en este capítulo se exponen los detalles de implementación más relevantes llevados a cabo durante la etapa de Implantación, así como las dificultades encontradas en la elaboración de la aplicación.

- **Conclusiones y líneas futuras:** en este capítulo se exponen las conclusiones del autor con respecto a la elaboración del proyecto. También se proponen líneas futuras de trabajo y posibles mejoras e integraciones del software implementado para proporcionar mayor funcionalidad y calidad al mismo.

- **Bibliografías y referencias:** se exponen todos los recursos bibliográficos tanto electrónicos como físicos que han sido utilizados durante la elaboración del proyecto.

- **Anexos:** dentro de los anexos se incluyen ciertos aspectos de relevancia que se caracterizan por no poder ser clasificados en ninguno de los anteriores capítulos.
  Dentro de los anexos se encuentra el glosario de términos, el manual de instalación, de uso, de ampliación y de generación de plugins.

4. Gestión del proyecto

En este capítulo se expondrán los elementos relacionados con el proyecto a alto nivel. Entre estos aspectos cabe resaltar la planificación y el seguimiento de la realización del mismo. Se realizará una comparación entre el desarrollo real del proyecto y la planificación inicial.

También se comentarán aspectos relacionados con el ciclo de vida del software y las decisiones que se han tomado sobre cómo debería ser dicho ciclo. Por otro lado, se realiza una clasificación de los medios y recursos que han sido necesarios en el transcurso del proyecto, relacionándolos con un presupuesto simulado teniendo en cuenta los costes directos e indirectos del mismo.

4.1. Planificación

La elaboración de la planificación del proyecto se ha realizado mediante diagramas de Gantt. Se detallará la planificación inicial con estimaciones temporales y por otro lado se mostrará el desarrollo real del proyecto comparándolos e indicando las razones por las que existe cierta divergencia entre ellos, normalmente basados en dificultades en la etapa de desarrollo e indisposición de los recursos durante periodos largos de tiempo.

4.1.1. Planificación inicial. Ciclo de vida

El ciclo de vida seleccionado para usar en este proyecto ha sido el clásico ciclo de vida en cascada, pasando por las fases de:

- **Análisis inicial de tecnologías:** Comprende el estudio de las tecnologías de análisis forense, así como el conjunto de pruebas desarrolladas para comprobar el efecto que tienen la instalación de aplicaciones sobre los distintos sistemas operativos.

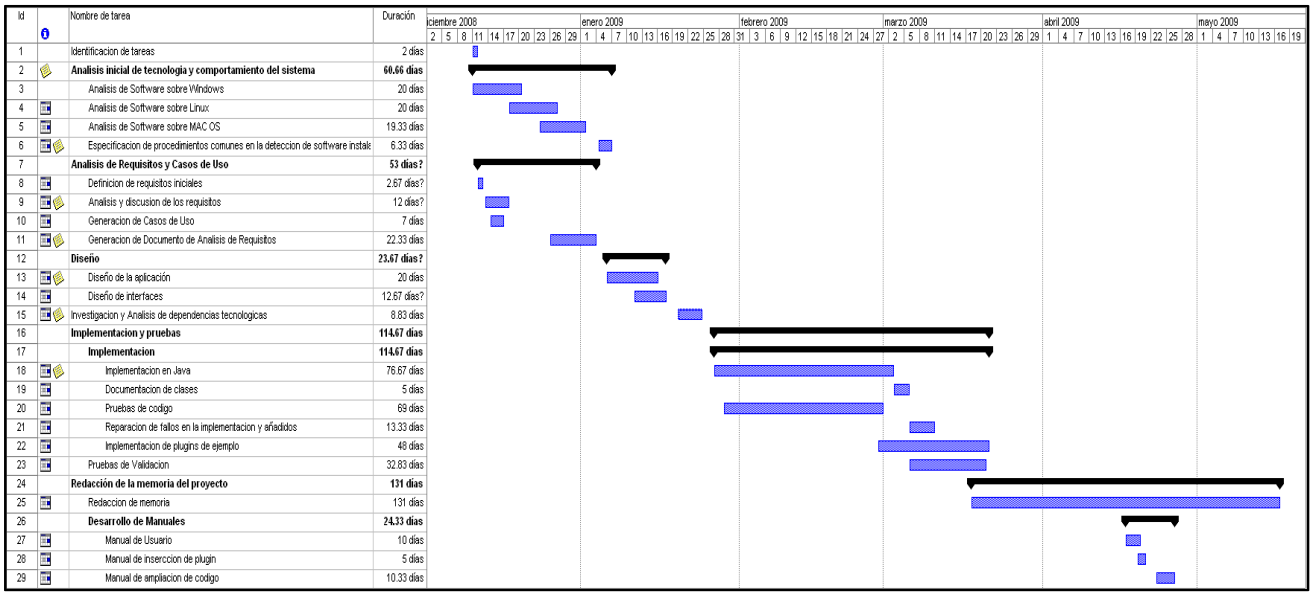
- **Análisis de requisitos**: Fase en la que se ha recopilado los requisitos en función del enunciado propuesto en el PFC y distintas conversaciones con el Tutor del mismo.

- **Diseño del sistema:** Elaborado a partir de los requisitos, desde el principio se estableció como prioritario utilizar una arquitectura que independizara los distintos componentes, mostrando una clara predilección por el patrón MVC(Modelo-Vista-Controlador).

- **Implementación + Pruebas:** Proceso de implementación de las clases que componen la aplicación implementando a su vez un conjunto de pruebas.

- **Implantación:** Proceso de adecuación de la plataforma y creación de un conjunto de Plugins inicial que permita un uso inmediato de la plataforma. También en este punto se incluye la redacción de la memoria del proyecto estableciendo las bases para poder realizar un mantenimiento adecuado de la plataforma mediante manuales y procedimientos proporcionando conocimiento adquirible al usuario.

En principio las fases de implementación y pruebas deberían ser diferentes, pero en nuestro caso se han solapado porque han sido necesarias durante el transcurso del proyecto para verificar el buen funcionamiento del mismo. La fase de mantenimiento no se ha planificado ya que se puede considerar externa al proyecto.



Id	Nombre de tarea	Duración
1	Identificacion de tareas	2 días
2	**Analisis inicial de tecnologia y comportamiento del sistema**	**60.66 días**
3	Analisis de Software sobre Windows	20 días
4	Analisis de Software sobre Linux	20 días
5	Analisis de Software sobre MAC OS	19.33 días
6	Especificacion de procedimientos comunes en la deteccion de software instal	6.33 días
7	**Analisis de Requisitos y Casos de Uso**	**53 días?**
8	Definicion de requisitos iniciales	2.67 días?
9	Analisis y discusion de los requisitos	12 días?
10	Generacion de Casos de Uso	7 días
11	Generacion de Documento de Analisis de Requisitos	22.33 días
12	**Diseño**	**23.67 días?**
13	Diseño de la aplicación	20 días
14	Diseño de interfaces	12.67 días?
15	Investigacion y Analisis de dependencias tecnologicas	8.83 días
16	**Implementacion y pruebas**	**114.67 días**
17	**Implementacion**	**114.67 días**
18	Implementacion en Java	76.67 días
19	Documentacion de clases	5 días
20	Pruebas de codigo	69 días
21	Reparacion de fallos en la implementacion y añadidos	13.33 días
22	Implementacion de plugins de ejemplo	48 días
23	Pruebas de Validacion	32.83 días
24	**Redacción de la memoria del proyecto**	**131 días**
25	Redaccion de memoria	131 días
26	**Desarrollo de Manuales**	**24,33 días**
27	Manual de Usuario	10 días
28	Manual de inserccion de plugin	5 días
29	Manual de ampliacion de codigo	10.33 días

Gráfico 1. Planificación inicial (Diagrama de Gantt)

Como se puede observar en la imagen, el proyecto se inicio a mediados de diciembre de 2008 y su desarrollo, según la planificación inicial, se prolongaba hasta mediados de Mayo del año 2009.

4.1.2. Planificación final. Ciclo de vida

En la planificación inicial del apartado anterior se contaba con la disponibilidad completa de los recursos entre las fechas indicadas durante el horario de tarde incluyendo también jornada completa en los sábados y domingos. Sin embargo, la mudanza del desarrollador del proyecto a principios de Marzo y un ligero incremento de su jornada laboral en su puesto durante parte de este periodo han retrasado varios meses esta planificación inicial.

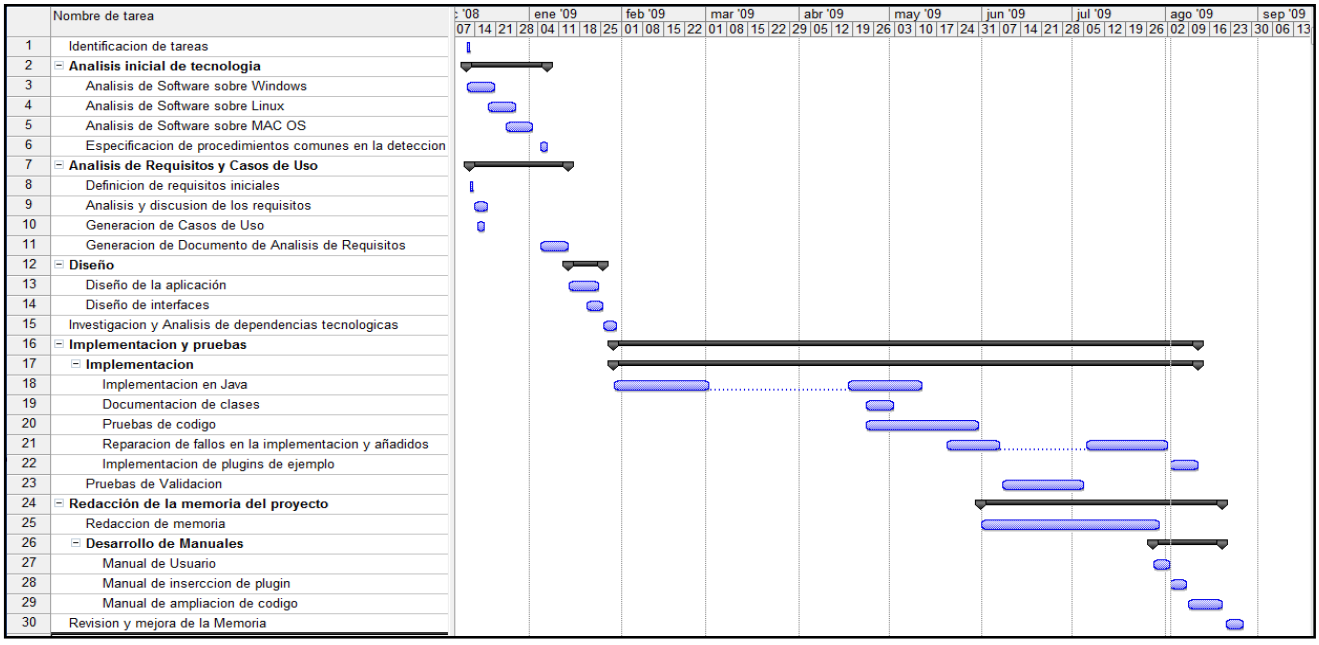


Gráfico 2. Planificación final (Diagrama de Gantt)

Tras observar ambos gráficos, la gran diferencia entre ambos se observa rápidamente en un periodo durante el que se suspendió el desarrollo del proyecto. Tal como se indica en el párrafo anterior, este periodo entre Marzo y finales de Abril corresponde al tiempo dedicado a la mudanza del diseñador y desarrollador del proyecto. Al disponer de un único recurso humano en esta fase del proyecto y ante la imposibilidad de este de continuar con el desarrollo durante este periodo, se vio suspendido durante más de mes y medio, retrasando la fecha de finalización del proyecto al mes de Septiembre.

También se observan diferencias notables en algunas fases del proyecto, que han sido realizadas en un orden y duración ligeramente diferentes a los mostrados en la planificación original. Por ejemplo, la fase de "Reparación de fallos en la implementación y añadidos" se desarrollo en dos periodos: uno tras las pruebas básicas y otro tras las pruebas con los plugins reales implementados.

Por último, se ha añadido una fase de revisión y mejora de la Memoria Final del PFC, producto de las revisiones realizadas por el Tutor y la mejora de la misma por parte del Autor.

4.2. Medios técnicos empleados para el proyecto

En la realización del proyecto se ha hecho uso de herramientas software y hardware especificas. A continuación se enumeran los detalles respecto a estos medios técnicos:

4.2.1. Herramientas Hardware

Equipo sobremesa:

- Microprocesador Intel Pentium IV
- 1 Giga de RAM
- 250 Gigas de Disco Duro

Equipo portátil:

- Portátil HP 6730s
- Microprocesador Intel Centrino Core 2 Duo
- 2 Giga de RAM
- 160 Gigas de Disco Duro

Periféricos:

- Memoria USB de 8 Gigas.

4.2.2. Herramientas Software

En este proyecto ha sido necesario realizar varias pruebas con distintos sistemas operativos. En este apartado se ven reflejados todos ellos, aunque la mayor parte del proyecto se codificó en Windows™ XP con la ayuda del IDE Eclipse.

4.2.2.1. Sistemas operativos

- Sistema Operativo Microsoft® Windows™ XP™ SP3
- Sistema Operativo Microsoft® Windows™ Vista™ SP1
- Sistema Operativo Ubuntu

4.2.2.2. Herramientas

- Java Development Kit 6.0
- Java Development Kit 5.0
- Java Runtime Enviroment (JRE). ***Es necesaria la versión 6 para firmar informes.***
- Eclipse Ganymede
- Netbeans 5.0
- Systracer http://www.blueproject.ro/systracer
- Sun ® VirtualBox

4.2.2.3. Librerías Java

- **Bouncy Castle** es un proyecto de software libre que pretende desarrollar una serie de librerías criptográficas libres y, entre otros, ofrece métodos

intermedios para las herramientas de seguridad de Java facilitando así su uso. En este caso es usada para firmar documentos PDF. http://www.bouncycastle.org/.
Esta librería nos permite firmar el documento con un certificado firmado con el estándar **PKCS12**, usado de forma bastante generalizada. Por ejemplo, en España, los certificados expedidos por la **FNMT (Fábrica Nacional de Moneda y Timbre)** son de este tipo.

- **iText** es una biblioteca Open Source para crear y manipular archivos PDF, RTF, y HTML en Java. http://www.lowagie.com/iText/
- **JDOM** es una biblioteca de código libre para manipulaciones de datos XML optimizados para Java.
  http://www.jdom.org
- **JWizard** es una biblioteca que permite realizar de una forma sencilla una interfaz gráfica a modo de asistente o **Wizard**. Normalmente utilizado para instaladores de software, en este caso nos proporciona una interfaz basada en pasos muy intuitiva para todo tipo de usuarios. Se ha modificado el código para permitirnos cambiar algunas características a fin de integrarlo mejor en nuestra aplicación.
  http://flib.sourceforge.net/JWizard/doc/index.html
- **JNIregistry** nos permite navegar por el registro de Windows™ haciendo uso de librerías nativas del sistema operativo.
  http://www.trustice.com/java/jnireg/
- **Swing-Worker** mejora el comportamiento de las interfaces gráficas Java mientras se ejecutan por debajo tareas pesadas. Sin ella, estas tareas bloquearían la interfaz gráfica sin obtener respuesta ante ninguna de las acciones del usuario. Forma parte de JAVA 6.0, sin embargo se ha añadido la librería para garantizar mayor compatibilidad con sistemas antiguos. https://swingworker.dev.java.net/
- **StegSecret** consiste en un conjunto de herramientas libres que permiten la detección de información esteganografiada en diferentes medios de información. Se han reutilizado los métodos que analizan los archivos buscando indicios de contenido oculto.
  http://stegsecret.sourceforge.net/.
  Código facilitado por Alfonso Muñoz, desarrollador principal de la herramienta.

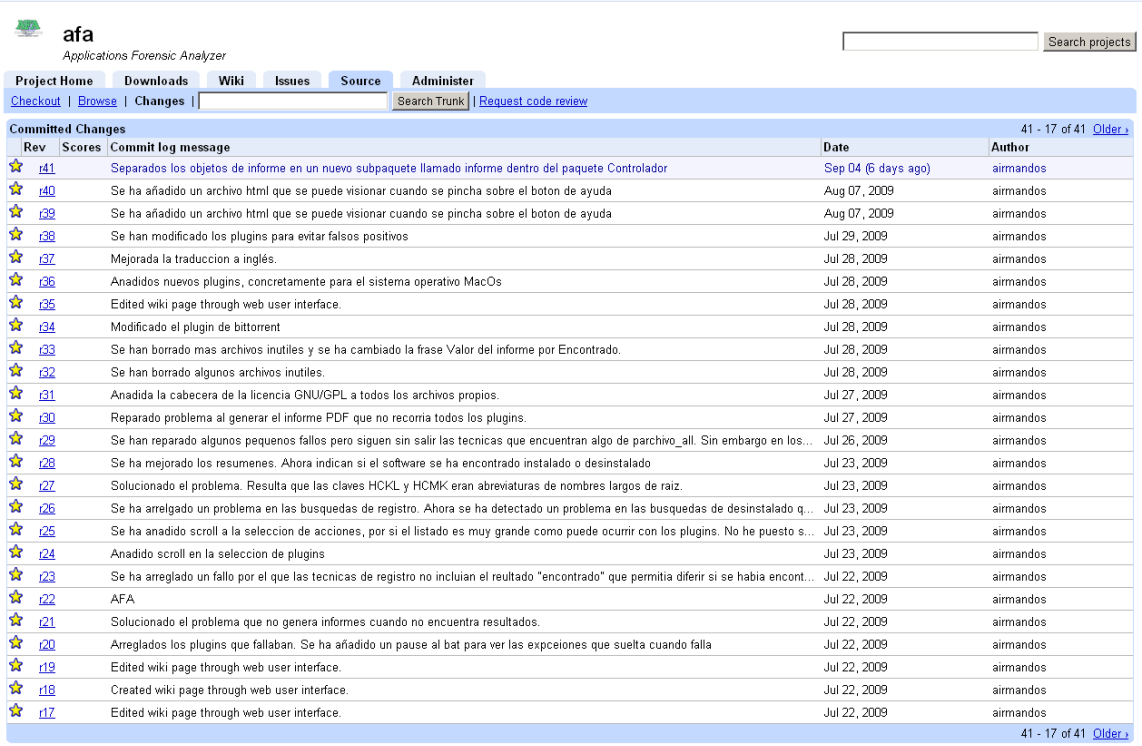
4.2.2.4. Suites informáticas

- **Microsoft® Office™ Project 2007 Professional**
- **Microsoft® Office™ 2007 Professional**

4.2.2.5. Herramientas Web

- **Google Code.** Se ha abierto un proyecto en la url http://code.google.com/p/afa/ de la herramienta de gestión de proyectos software que facilita Google. Se utiliza por proporcionar servicios SVN (Subversión) para el control de versiones y permite espacio de

almacenamiento para publicación de productos de software libre. Con esto se pretende poner la aplicación desarrollada a disposición del público para su uso, modificación o ampliación del mismo.



afa
Applications Forensic Analyzer

Search projects

Project Home | Downloads | Wiki | Issues | Source | Administer

Checkout | Browse | Changes | Search Trunk | Request code review

Committed Changes 41 - 17 of 41 Older ›

Rev	Scores	Commit log message	Date	Author
r41		Separados los objetos de informe en un nuevo subpaquete llamado informe dentro del paquete Controlador	Sep 04 (6 days ago)	airmandos
r40		Se ha añadido un archivo html que se puede visionar cuando se pincha sobre el boton de ayuda	Aug 07, 2009	airmandos
r39		Se ha añadido un archivo html que se puede visionar cuando se pincha sobre el boton de ayuda	Aug 07, 2009	airmandos
r38		Se han modificado los plugins para evitar falsos positivos	Jul 29, 2009	airmandos
r37		Mejorada la traduccion a inglés.	Jul 28, 2009	airmandos
r36		Anadidos nuevos plugins, concretamente para el sistema operativo MacOs	Jul 28, 2009	airmandos
r35		Edited wiki page through web user interface.	Jul 28, 2009	airmandos
r34		Modificado el plugin de bittorrent	Jul 28, 2009	airmandos
r33		Se han borrado mas archivos inutiles y se ha cambiado la frase Valor del informe por Encontrado.	Jul 28, 2009	airmandos
r32		Se han borrado algunos archivos inutiles.	Jul 28, 2009	airmandos
r31		Anadida la cabecera de la licencia GNU/GPL a todos los archivos propios.	Jul 27, 2009	airmandos
r30		Reparado problema al generar el informe PDF que no recorria todos los plugins.	Jul 27, 2009	airmandos
r29		Se han reparado algunos pequenos fallos pero siguen sin salir las tecnicas que encuentran algo de parchivo_all. Sin embargo en los...	Jul 26, 2009	airmandos
r28		Se ha mejorado los resumenes. Ahora indican si el software se ha encontrado instalado o desinstalado	Jul 23, 2009	airmandos
r27		Solucionado el problema. Resulta que las claves HCKL y HCMK eran abreviaturas de nombres largos de raiz.	Jul 23, 2009	airmandos
r26		Se ha arrelgado un problema en las busquedas de registro. Ahora se ha detectado un problema en las busquedas de desinstalado q...	Jul 23, 2009	airmandos
r25		Se ha anadido scroll a la seleccion de acciones, por si el listado es muy grande como puede ocurrir con los plugins. No he puesto s...	Jul 23, 2009	airmandos
r24		Anadido scroll en la seleccion de plugins	Jul 23, 2009	airmandos
r23		Se ha arreglado un fallo por el que las tecnicas de registro no incluian el reultado "encontrado" que permitia diferir si se habia encont...	Jul 22, 2009	airmandos
r22		AFA	Jul 22, 2009	airmandos
r21		Solucionado el problema que no genera informes cuando no encuentra resultados.	Jul 22, 2009	airmandos
r20		Arreglados los plugins que fallaban. Se ha añadido un pause al bat para ver las expceiones que suelta cuando falla	Jul 22, 2009	airmandos
r19		Edited wiki page through web user interface.	Jul 22, 2009	airmandos
r18		Created wiki page through web user interface.	Jul 22, 2009	airmandos
r17		Edited wiki page through web user interface.	Jul 22, 2009	airmandos

41 - 17 of 41 Older ›

Gráfico 3. Captura de la página del proyecto en google code

4.3. Análisis económico del proyecto

En este apartado se realizará un análisis económico de los costes asociados al proyecto. Nuestro proyecto disponía de un presupuesto inicial. Tras la finalización del proyecto se puede presentar un presupuesto definitivo, permitiéndonos realizar un análisis de la desviación comparando ambas cifras y analizar los motivos.

4.3.1. Metodología de estimación de costes

Los gastos que conforman el presupuesto se dividen en varios tipos de recursos que serán clasificados en los siguientes grupos:

- **Recursos humanos**: Corresponden a la multiplicación de recursos humanos X el coste asociado por cada uno por hora trabajada X el tiempo trabajado en horas. Para eso dispondremos de los datos de la etapa de planificación, pudiendo obtener una estimación aproximada de las horas empleadas por cada recurso humano.

- **Herramientas software:** Corresponden al conjunto del coste asociado a licencias software del proyecto.

- **Herramientas hardware:** Para las herramientas hardware que se han utilizado se ha de contabilizar el precio de adquisición o alquiler, aplicando si procede el correspondiente prorrateo.

El proyecto no ha sufrido costes indirectos derivados de costes de transporte. En todo caso podría considerarse como coste indirecto la conexión a internet utilizada para descargar las herramientas necesarias y consultar la documentación necesaria para el desarrollo del mismo. Sin embargo esta conexión se trata de un ADSL estándar con tarifa plana, de manera que no podemos medir la proporción de la factura telefónica que corresponde a costes indirectos de nuestro proyecto.

4.3.2. Presupuesto inicial

En este apartado se indica el coste asociado a los recursos usados en el transcurso del proyecto.

4.3.2.1. Costes asociados a recursos humanos

Tal como se ha indicado anteriormente, el coste asociado al personal se calcula multiplicando el número de horas empleadas por el coste unitario de los empleados.

En este proyecto han participado dos recursos. El primero es el alumno que se ha encargado de la planificación, diseño, desarrollo y documentación del mismo. Por otro lado está el tutor, que ha participado en la adquisición de requisitos de usuario y en tareas de dirección, aparte de las tareas de revisión pertinentes tras las distintas fases.

Para establecer los salarios de los componentes humanos del proyecto se recogerán los datos estimados de la página web de la ***Asociación de Doctores, Licenciados e Ingenieros en Informática.***

Según estos valores, un director de proyecto tiene un salario anual aproximado de 37000 €/año, que representan aproximadamente 19 €/hora en jornada completa de 40 horas semanales. Por otro lado, un analista programador tiene un salario anual de 26000 €/año, que representan aproximadamente 13 €/hora. En base a estos valores y la planificación inicial, teniendo en cuenta 4 horas de trabajo por cada jornada, obtendremos las siguientes tablas.

Fase	Días	Horas	Coste/Hora	Coste total
Identificación de tareas(Analista)	2	5.7	13.00 €	74.29 €
Identificación de tareas(Tutor)	3	8.6	19.00 €	162.86 €
Análisis de Requisitos y Casos de Uso	60	171.4	13.00 €	2,228.57 €
Diseño	23	65.7	13.00 €	854.29 €

Implementación y pruebas	114	325.7	13.00 €	4,234.29 €
Pruebas (Tutor)	2	5.7	19.00 €	108.57 €
Redacción de la memoria del proyecto	131	374.3	13.00 €	4,865.71 €
Revisión	5	14.3	19.00 €	271.43 €
Coste total del proyecto				**12,800.00 €**

Tabla 2. Costes Iniciales asociados a Recursos Humanos

4.3.2.2. Costes asociados a medios

En la siguiente tabla se muestran el precio de los componentes hardware utilizados en el desarrollo del proyecto.

Recurso	Precio Compra/Leasing	Meses Uso	Meses Vida útil	%Asignado al proyecto
Portátil 6730s	427 €	6.0	50	25.62 €
PC sobremesa	350 €	6.0	50	21.00 €
Memoria USB 8 Gb	16 €	6.0	50	0.96 €
Coste total del proyecto				**47.58 €**

Tabla 3. Costes asociados a componentes Hardware

4.3.2.3. Costes asociados a licencias software

En la siguiente tabla se muestran el precio de los componentes software utilizados en el desarrollo del proyecto.

Software	Coste de Licencia	Validez en meses	Uso en meses	Coste total
Microsoft® Office™ Project 2007 Professional	Licencia universitaria	Indefinido	6.00	- €
Microsoft® Office™ 2007 Professional	Licencia universitaria	Indefinido	6.00	- €
Sistema Operativo Microsoft® Windows™ XP™ SP3	Incluido en PC sobremesa	Indefinido	6.00	- €
Sistema Operativo Microsoft® Windows™ Vista™ SP1	Incluido en PC portátil	Indefinido	6.00	- €
Sistema Operativo Ubuntu Desktop	Gratuito	Indefinido	6.00	- €
Java Development Kit 6.0	Gratuito	Indefinido	6.00	- €
Java Development Kit 5.0	Gratuito	Indefinido	6.00	- €
Eclipse Ganymede	Gratuito	Indefinido	6.00	- €
Netbeans 5.0	Gratuito	Indefinido	6.00	- €
Systracer	Versión de prueba	1	1.00	- €
Coste total				**- €**

Tabla 4. Costes asociados a licencias Software

4.3.2.4. Resumen de costes inicial

A continuación se presenta el total de la estimación presupuestaria que se realiza al inicio del proyecto.

Recurso	Total
Humanos	12,800.00 €
Hardware	47.58 €
Software	- €
Total	12,847.58 €

Tabla 5. Estimación inicial de costes

4.3.3. Presupuesto final

Como se ha indicado en la planificación, el proyecto fue sometido a un retraso considerable de mes y medio de duración por varios motivos. En este presupuesto final no se contarán esos meses en lo referente a uso de recursos humanos, sin embargo si se tendrá en cuenta en los costes de hardware y licencias de software.

A continuación, al igual que se ha hecho en el apartado anterior, se desglosan los costes en función de su naturaleza.

4.3.3.1. Costes asociados a recursos humanos

Teniendo en cuenta la duración final de las distintas etapas de desarrollo del proyecto se ha elaborado la siguiente tabla:

Fase	Días	Horas	Coste/Hora	Coste total
Identificación de tareas(Analista)	3	8.6	13.00 €	111.43 €
Identificación de tareas(Tutor)	5	14.3	19.00 €	271.43 €
Análisis de Requisitos y Casos de Uso	30	85.7	13.00 €	1,114.29 €
Diseño	30	85.7	13.00 €	1,114.29 €
Implementación y pruebas	110	314.3	13.00 €	4,085.71 €
Pruebas (Tutor)	2	5.7	19.00 €	108.57 €
Redacción de la memoria del proyecto	87	248.6	13.00 €	3,231.43 €
Revisión	5	14.3	19.00 €	271.43 €
Coste total del proyecto				**10,308.57 €**

Tabla 6. Costes Finales asociados a recursos humanos

4.3.3.2. Costes asociados a licencias software

A pesar de la diferencia temporal que hay entre ambas planificaciones (inicial y final), el coste por licencias software no ha sufrido desviación debido al uso íntegramente de licencias universitarias, gratuitas y de prueba para el desarrollo del proyecto.

Software	Coste de Licencia	Validez en meses	Uso en meses	Coste total
Microsoft® Office™ Project 2007 Professional	Licencia universitaria	Indefinido	9	- €
Microsoft® Office™ 2007 Professional	Licencia universitaria	Indefinido	9	- €
Sistema Operativo Microsoft® Windows™ XP™ SP3	Incluido en PC sobremesa	Indefinido	9	- €
Sistema Operativo Microsoft® Windows™ Vista™ SP1	Incluido en PC portátil	Indefinido	9	- €
Sistema Operativo Ubuntu Desktop	Gratuito	Indefinido	9	- €
Java Development Kit 6.0	Gratuito	Indefinido	9	- €
Java Development Kit 5.0	Gratuito	Indefinido	9	- €
Eclipse Ganymede	Gratuito	Indefinido	9	- €
Netbeans 5.0	Gratuito	Indefinido	9	- €
Systracer	Versión de prueba	1	1	- €
Coste total				- €

Tabla 7. Costes finales asociados a licencias Software

4.3.3.3. Costes asociados a recursos Hardware

En este aspecto, el retraso en el desarrollo del proyecto sí ha modificado el coste final total que han supuesto los recursos hardware:

Recurso	Precio Compra/Leasing	Meses Uso	Meses Vida útil	%Asignado al proyecto
Portátil 6730s	427 €	9.0	50	38.43 €
Ordenador sobremesa	350 €	9.0	50	31.50 €
Memoria USB 8 Gb	16 €	9.0	50	1.44 €
Coste total del proyecto				71.37 €

Tabla 8. Costes finales asociados a recursos Hardware

4.3.3.4. Resumen de costes final

Finalmente, se muestra un resumen del total de gastos asociados al proyecto, divididos en función de su naturaleza.

Recursos	Total
Humanos	10,308.57 €
Hardware	71.37 €
Software	- €
Total	10,379.94 €

Tabla 9. Resumen final de Costes

4.3.4. Comparativa de costes

Entre el presupuesto inicial y los costes reales se pueden observar una diferencia de 2467.64 Euros. Esta desviación se justifica por los periodos de pausa producidos en el desarrollo (horas excluidas del total) unidos a la variación en la duración de las fases. Algunas etapas suponen un coste por hora superior a otras y, aunque la duración total del proyecto no difiera demasiado, un gran cambio en la duración de etapas con costes muy dispares puede variar considerablemente el coste total del proyecto, tal y como ha pasado en este caso.

En un proyecto real, esta diferencia habría aumentado los beneficios obtenidos por el desarrollo de la aplicación al haber gastado en el mismo una cifra inferior a la esperada. En este caso el presupuesto de alquiler de hardware ha sido superior, pero la variación en las fases ha producido un ahorro considerable al haber requerido más tiempo en tareas de coste/hora menores.

4.3.5. Análisis de la licencia usada: GNU GPL v3

Este proyecto se distribuirá bajo la licencia GNU GPL. Los motivos de dicha elección son varios, pero principalmente, se pretende que cualquier usuario pueda contribuir en la aplicación y mejora de la herramienta y esta es una de las premisas de la mencionada licencia.

Por otro lado está la utilización de librerías y parte de código de StegSecret, una aplicación liberada bajo licencia GPL. En esta se especifica que el código puede ser alterado y reutilizado siempre que sea bajo la misma licencia que el original.

Para hacer uso de esta licencia es necesario incluir un comentario en la cabecera de todos los ficheros de código del proyecto con el siguiente enunciado:

```
This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Para más detalles acerca de esta licencia, se puede leer de forma íntegra en el Anexo 6: Licencia GNU GPL.

5. Análisis

5.1. Propósito del plan

El objetivo del presente capitulo de análisis es recoger una especificación detallada del sistema que se va a construir, de manera que constituya la base para el posterior diseño del mismo.

También se analizará el estado actual del análisis forense, área al que pertenece la aplicación, revisando algunas de las soluciones existentes para cubrir la funcionalidad que aporta este software.

A partir de los requisitos indicados por el usuario, en la etapa de análisis estos se detallan y desarrollan, constituyendo los denominados requisitos software, que permiten especificar el problema a resolver.

En definitiva, el presente capítulo dedicado al análisis no es sino una especificación de qué debe hacer el sistema y cómo lo debe llevar a cabo mediante la descripción de requisitos software.

También, en el transcurso de este capítulo se diseñará un conjunto de pruebas previas realizadas para comprobar el efecto que produce la instalación de varias aplicaciones en los distintos sistemas operativos a fin de conocer qué tipo de funcionalidad se debe implementar para detectar los residuos dejados por estos programas.

5.2. Ámbito de la aplicación: Informática Forense

Se considera informática forense a la aplicación de un conjunto de métodos científicos y analíticos especializada que permiten descubrir y presentar evidencias que sean válidas dentro de un proceso legal.

En algunos casos, un analista forense puede llegar a recuperar información que haya sido borrada, siempre cuando se cumplan determinadas condiciones que lo permitan.

La Informática Forense es una ciencia relativamente nueva y no existen estándares aceptados, sin embargo existen unas características comunes en la mayor parte de manuales y procedimientos que hacen referencia a esta ciencia.

5.2.1. Fases del análisis

El análisis forense se divide comúnmente en los siguientes pasos, que han de ser ejecutados en un estricto orden:

- **Identificación**

Consiste en la recopilación de información acerca del entorno del bien informático que se va a analizar. Es importante definir en esta fase que se está buscando, donde se va a buscar y que métodos se utilizarán. Incluye muchas veces la identificación de los elementos informáticos que se analizan y el inicio de la cadena de custodia.

- **Preservación**

Esta fase garantiza que los medios originales no son alterados en el proceso de análisis. Comúnmente en este paso se generan imágenes forenses de las evidencias, siendo estas imágenes sobre las que se realizará el análisis. Las imágenes se realizan copiando la información física bit a bit del origen aplicando distintos métodos para garantizar que la información original no sea modificada ni lo más mínimo. En muchos casos se utilizan elementos hardware específicos para realizar esta tarea.

En este paso es necesario continuar la cadena de custodia típicamente iniciada en la fase anterior para garantizar la integridad de los datos con intenciones de mostrarlos en un proceso legal.

- **Análisis**

Durante este paso se aplican las técnicas analíticas de las que disponen los expertos forenses para encontrar pruebas de ciertos comportamientos. Se analiza todo tipo de información procedente del medio analizado, especialmente la información de registro del sistema operativo y de navegación a través de internet(correos, páginas visitadas,etc), que es la que más pistas puede ofrecer para determinar las acciones realizadas sobre el dispositivo.

- **Presentación**

Esta fase consiste en recopilar toda la información que se obtuvo a partir del análisis para realizar el reporte y la presentación a los abogados, la generación (si es el caso) de una pericial y de su correcta interpretación sin hacer uso de tecnicismos.

Esta fase consiste en agrupar todas las evidencias encontradas en uno o varios informes ofreciendo en algunos casos resúmenes y explicaciones comprensibles por el entorno jurídico.

Aplicando estos conceptos a nuestro software, podemos analizar la división lógica de su funcionamiento en base a estas fases:

1. **Identificación**: El usuario podrá elegir sobre que dispositivos se realizarán las búsquedas así como el software que se pretende buscar, es decir, identifica los bienes informáticos que son base del análisis y especifica el que se busca sobre ellos.

2. **Análisis**: La aplicación buscará en todo el disco determinados elementos que concuerden con los rastros dejados por la instalación de una aplicación concreta. Los plugins que corresponden con cada aplicación están definidos por comportamientos comprobados previamente.

3. **Presentación**: El resultado del análisis quedará plasmado en un informe que, opcionalmente incluye la posibilidad de ser firmado por un archivo con un certificado PKCS12 garantizando, de esta manera, la autoría del análisis.

Se puede observar que la fase de preservación no ha sido incluida porque no forma parte de las capacidades del software. Se ha planteado como una alternativa pero no se ha llegado a implementar por varios motivos que se enumeraran en el capítulo ***7.2 Líneas Futuras***.

La finalidad de este software, como se verá durante su desarrollo, es generar una plataforma sencilla de utilizar y ampliable que permita, en unos pocos pasos, cubrir unas necesidades muy concretas de análisis forense, sin necesidad de recurrir a herramientas complicadas. Sin embargo, por este mismo motivo, es limitada y para conseguir las mismas funcionalidades requiere trabajo previo con otras utilidades. Por ejemplo, es compatible con la fase de preservación si la generación de la imagen bit a bit se genera primero y luego se ejecuta el software sobre la misma montando la imagen.

Por lo tanto, nuestro objetivo es el de hacer una aplicación complementaria que busque restos de manera automatizada pero no un sustituto para el software actual.

5.2.2. Software actual de informática forense

La informática forense como ciencia engloba un conjunto de actividades mucho mayor que nuestra aplicación, que, como se ha explicado previamente, tiene una finalidad muy concreta. Por ejemplo, no se mencionará en este apartado software que permite romper contraseñas, desencriptar discos, analizar procesos, etc. En nuestro caso vamos a mencionar herramientas que se utilizan en la informática forense para descubrir pistas en el disco de la actuación de un software.

Las utilidades que se utilizan para estos fines generalmente son kits herramientas que en conjunto ofrecen las herramientas necesarias para completar las fases previamente descritas en un análisis forense de bienes informáticos.

En el marco del software libre, los más importantes se enumeran a continuación:

- **The Coroner's Toolkit(TCT)**: Es un suite de programas creados por dos de los pioneros en la I.F. Dan Farmer and Wietse Venema presentado en Agosto de 1999 en un curso de Análisis Forense. Están pensadas para realizar el análisis de un sistema basado en UNIX después de una intrusión. Permite capturar información, recuperar contraseñas, recuperar ficheros borrados y establecer una referencia temporal respecto a los archivos basados en su fecha de modificación y/o creación.

- **The Sleuth Kit(TSK):** Se trata de un conjunto de aplicaciones Unix basadas en línea de comandos que permite el análisis de distintos tipos de particiones de

discos. La aplicación permite realizar un análisis del sistema de fichero sin generar cambios en el mismo. Permite recuperar archivos, generar imágenes físicas de discos así como cargarlas para analizarlas.

Se trata de una suite basada en TCT intentando ampliar las funcionalidades que este ofrecía. Existe una aplicación complementaria llamada **Autopsy Forensic Browser** que ofrece una interfaz web para facilitar el uso de la suite. También existe otra interfaz alternativa llamada **PTK**.

- **Foundstone Forensic Utilities:** Conjunto de utilidades de análisis forense que permite encontrar evidencias de distintos tipos del sistema operativo Windows™, tal como residuos de navegación de Internet Explorer, cookies, papelera de reciclaje y análisis de la actividad con el sistema operativo Windows™ sobre disco.

- **Forensic and Log Analysis GUI (FLAG)**: Diseñado para simplificar el proceso de análisis de ficheros de log en investigaciones forenses. Establece un sistema de correlación de manera que facilita la obtención de resultados sobre gran cantidad información. Esta aplicación establece algunas semejanzas con algunas de las funcionalidades de la aplicación desarrollada, concretamente con las opciones que analizan los logs del sistema.

Estas y otras aplicaciones están incluidas en varias distribuciones de Linux que tienen como fin realizar análisis forense de equipos. En las referencias del apartado **"Bibliografías y referencias"** se puede encontrar más software relacionado y amplia información sobre los kits y programas expuestos.

Respecto a aplicaciones de pago, al tratarse de software muy especializado, suelen suponer un coste de licencia muy alto. Un ejemplo muy claro de este tipo de soluciones es la herramienta Encase:

- **Encase ®:** Proporciona medios para cubrir todas las fases previamente mencionadas sobre el análisis forense. Genera imágenes en formatos válidos a efectos legales y utiliza varios medios para verificar la autenticidad de las mismas. También garantiza la compatibilidad con numerosos sistemas de ficheros distintos, así como hardware de uso profesional (raid, servidores,etc).
  Proporciona un lenguaje de programación que permite ampliar la funcionalidad de la aplicación.
  También, por otro lado, permite grandes posibilidades de configuración de los reportes generados. Se puede ver todas las posibilidades de la herramienta de forma resumida en el siguiente enlace: http://www.guidancesoftware.com/WorkArea/DownloadAsset.aspx?id=673

5.2.3. Cuestiones legales.

En este capítulo se analizarán los elementos que rodean al proyecto y que pueden tener algún tipo de relación con elementos jurídicos.

5.2.3.1. Confidencialidad

La aplicación realizará un análisis de las unidades y sistemas seleccionados por el usuario generando unos resultados que ponen en evidencia el contenido del disco. Estos datos serán almacenados en forma de logs e informes en el equipo informático del analizador en determinadas ubicaciones que se especificarán en este documento. En ningún momento estos datos podrán ser enviados ni compartidos con terceros si no es por acción propia del analizador. No se considera necesario realizar una declaración de privacidad al respecto dado que estos datos no se almacenan en ningún servidor centralizado y son responsabilidad única del usuario de la aplicación.

5.2.3.2. Requerimientos legales de las evidencias

"La evidencia digital es cualquier información obtenida a partir de un dispositivo electrónico o medio digital que sirve para adquirir convencimiento de la certeza de un hecho".

Extraído de http://www.borrmart.es/articulo_redseguridad.php?id=1376

Las fuentes de una evidencia digital corresponden a tres tipos:

- Sistemas de computación abiertos (computadores personales y sus periféricos, computadoras portátiles y servidores). Las evidencias que se obtendrán mediante nuestra aplicación pertenecerán a este grupo concreto al tratarse de análisis forense local.
- Sistemas de comunicación: redes de telecomunicaciones, comunicación inalámbrica e Internet.
- Sistemas convergentes de computación: teléfonos celulares, PDAs, etc.

Las evidencias digitales no están amparadas por una normativa legal en la que sustentarse. Al tratarse de un tipo de evidencia jurídica muy actual aún no están reguladas por el derecho procesal. Este problema deriva del hecho de la falta de conocimiento técnico informático por parte de los jueces para validar jurídicamente tales evidencias digitales.

Esta falta de conocimiento hace que no exista certeza sobre la admisibilidad de las evidencias digitales en un proceso judicial por parte de un juez porque no está capacitado para valorar la validez de estas pruebas.

A pesar de todo, en medios digitales existen iniciativas internacionales como las de IOCE (International Organization of Computer Evidence), la convención de Cybercrimen presentada por la comunidad europea, el ***Digital Forensic Reseach Workshop***, entre otros donde se establecen lineamiento de acción y parámetros que cobijan el tratamiento de la

evidencia en medios electrónicos, los cuales deben ser revisados y analizados en cada uno de los contextos nacionales para su posible incorporación.

En definitiva, hay una serie de aspectos que hay que tener en cuenta para considerar una evidencia legal como válida ante un tribunal:

- Las evidencias deben obtenerse de manera que se asegure la autenticidad y validez y no debe haber alteración alguna.
- Los procedimientos de búsqueda de computador no deben dañar, destruir o comprometer las evidencias.
- Se debe etiquetar, controlar y transmitir adecuadamente las copias de los datos, impresiones y resultado de la investigación.
- Se debe establecer y mantener una continua cadena de custodia. Debe garantizarse que las evidencias no han sido modificadas por terceros entre el momento en el que se extrajeron y el que se presenta la prueba ante el tribunal.
- No se debe divulgar y se debe respetar cualquier información del cliente (desde el punto de vista ético y legal) que inadvertidamente se pueda haber adquirido durante una exploración forense.
- Las pruebas deben ser obtenidas mediante orden judicial salvo en caso de no violar la expectativa razonable de privacidad o contar con el consentimiento de la persona afectada.

También existen varios aspectos externos que se deben tener en cuenta y que pueden determinar la admisibilidad de las evidencias recogidas:

- Las evidencias digitales pueden contener pérdidas o errores que generan cierta incertidumbre respecto a su veracidad y que pueden poner en duda la admisibilidad de la misma en un proceso legal. Estos errores pueden ser de diversa índole: mal estado de soportes físicos, problemas de configuración en el equipo donde se han recogido las evidencias, etc. Es importante reducir esta incertidumbre al máximo.
- La formación constante de los analistas forenses es fundamental para los procesos judiciales en los que intervienen, por lo que, también pueden existir errores humanos en la aplicación de las técnicas forenses que vean comprometida la admisibilidad de una evidencia digital.

5.2.3.3. Autoría de los resultados

Como parte de las solicitudes realizadas por el tutor, el resultado de los análisis ofrecerá la posibilidad de garantizar la autoría del analista que ejecuta el programa y por lo tanto se proporcionarán medios para ello. Los detalles sobre la metodología serán descritos en los siguientes apartados.

5.2.4. Área de aplicación del proyecto

Teniendo en cuenta los elementos investigados respecto al análisis forense en los apartados precedentes, terminaremos concluyendo los aspectos y el campo de acción sobre el que actuará nuestro software.

- **Ventajas de la aplicación sobre las soluciones existentes:**
    - Búsqueda de residuos en el disco duro siguiendo patrones de comportamiento definibles por el usuario de forma sencilla mediante Plugins.
    - Generación de Informes opcionalmente firmados con un certificado digital de forma automática.
    - Interfaz sencilla sin necesidad de profundos conocimientos en informática.
    - Permite la ejecución sin instalación en distintos sistemas operativos.
    - Se ofrece de forma gratuita.
    - Se ofrecen grandes facilidades para ampliar la funcionalidad.
    - Permite realizar otras tareas aparte de la intención original del software. Por ejemplo, se puede utilizar para realizar búsquedas complejas de forma automática.

- **Desventajas de la aplicación sobre las soluciones existentes:**
    - No realiza la generación de copias exactas de la información en forma de imágenes de forma nativa
    - No permite la recuperación de información borrada mediante técnicas de análisis físico del disco de forma nativa.

Por lo tanto, podemos definir la aplicación como una solución que permite a gran parte del público no especializado hacer una búsqueda de restos de actividad en el disco duro de forma automatizada. Específicamente se podrán encontrar residuos dejados por aplicaciones al ser instaladas/ejecutadas, aunque también puede ser utilizado para otros fines forenses definiendo otros tipos de Plugins. Se profundizará más sobre el tema en el capítulo **Conclusiones y líneas futuras** de esta misma memoria.

5.3. Análisis del efecto del software en los sistemas

En este apartado se mostrará un estudio para comprobar cuales son los restos que habitualmente dejan las aplicaciones cuando se instalan en los distintos sistemas operativos. Se separaran los estudios de estos sistemas operativos en distintos subapartados a fin de organizar mejor esta información y sacar conclusiones globales.

5.3.1. Instalaciones en el Sistema Operativo Windows

5.3.1.1. Preparación

- **Propósito**

El objetivo de este análisis es averiguar los cambios que se producen en el sistema operativo Windows en tras realizar la instalación de un nuevo software. Hay que tener en cuenta que estás pruebas solo son válidas para en versiones del sistema operativo a partir de Windows XP.

- **Software utilizado**

Para realizar este análisis se va a utilizar el siguiente software:

- **Sun ® VirtualBox** – Aplicación que permite simular una máquina virtual dentro de un sistema anfitrión.
- **Systracer** – Aplicación que permite realizar "snapshots" del estado de un sistema en un determinado momento y comparar las diferencias que se han producido.

5.3.1.2. Características del Sistema operativo

Es conveniente, antes de comenzar el análisis, detenerse a estudiar algunos aspectos del sistema operativo Windows.

- **Directorios de Windows**

Existen una serie de rutas genéricas en Windows donde se almacena información acerca del uso e instalación de programas. El nombre de estas carpetas en Windows varía en función del idioma, sin embargo se pueden averiguar a partir de variables de entorno del propio sistema operativo, como por ejemplo %windir% (el directorio donde está instalado Windows).

- **\Archivos de Programa:** Directorio que cuelga de la raíz del sistema de ficheros Windows y ruta por defecto para almacenar los archivos del programa en los instaladores típicos de Windows.
- **\Documents and Settings\Usario\Configuración local\Temp**: Directorio por defecto donde se almacenan los archivos temporales. Pueden almacenar residuos de instalaciones previas o del uso de programas.
- **\WINDOWS\Prefetch:** El prefetching es un servicio que lleva Windows XP para acelerar la carga del sistema operativo y sobre todo las aplicaciones al utilizar este catálogo por el que se habrán de iniciar de forma más rápida. En este directorio se almacenan datos para optimizar la ejecución de dichos programas. Es importante observar si

estos ficheros no son borrados tras una desinstalación, pudiéndose convertir en un gran aliado para nuestros objetivos.

- **\WINDOWS\system32:** Directorio que almacena las librerías dll y ocx del sistema. Si un programa necesita agregar una nueva librería, la instalará en este directorio.
- ***\WINDOWS\system32\config:*** Directorio que almacena los archivos del registro. Cualquier modificación que se realice sobre el registro modificará alguno de los archivos ubicados en esta carpeta.
- **\Documents and Settings\All Users\Menú Inicio\Programas\:** Directorio que almacena accesos directos (enlaces simbólicos) a los ejecutables y documentación de las distintas aplicaciones instaladas en el sistema.
- **\Documents and Settings\User\Ntuser.dat:** Archivo que contiene los datos relevantes al registro de la configuración del usuario.
- **\Documents and Settings\User\Datos de Programa:** Almacena archivos referentes a la configuración del programa de manera específica para cada usuario.

- ***Registro de Windows***

El Registro contiene información que Windows utiliza como referencia continuamente, por ejemplo los perfiles de los usuarios, las aplicaciones instaladas en el equipo y los tipos de documentos que cada aplicación puede crear, las configuraciones de las hojas de propiedades para carpetas y los iconos de aplicaciones, los elementos de hardware que hay en el sistema y los puertos que se están utilizando.

Una sección del Registro es un grupo de claves, subclaves y valores del Registro que cuentan con un conjunto de archivos auxiliares que contienen copias de seguridad de sus datos. Los archivos auxiliares de todas las secciones excepto HKEY_CURRENT_USER están en la carpeta %SystemRoot%\System32\Config en Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003 y Windows Vista. Los archivos auxiliares para HKEY_CURRENT_USER están en la carpeta %SystemRoot%\Profiles\nombreDeUsuario. Las extensiones de los archivos de estas carpetas indican el tipo de datos que contienen. A veces, la falta de extensión también puede indicar el tipo de datos que contienen.

Las distintas secciones del registro se encuentran almacenadas en los siguientes archivos del sistema tal como se indica en la siguiente tabla:

Sección del Registro	Archivos auxiliares
HKEY_LOCAL_MACHINE\SAM	Sam, Sam.log, Sam.sav
HKEY_LOCAL_MACHINE\Security	Security, Security.log, Security.sav
HKEY_LOCAL_MACHINE\Software	Software, Software.log, Software.sav
HKEY_LOCAL_MACHINE\System	System, System.alt, System.log, System.sav
HKEY_CURRENT_CONFIG	System, System.alt, System.log, System.sav, Ntuser.dat,

HKEY_USERS\DEFAULT	Ntuser.dat.log Default, Default.log, Default.sav

Tabla 10. Secciones del registro de Windows

Cada sección almacena un tipo de información. La descripción de las diferentes secciones se puede ver a continuación:

Carpeta	Descripción
HKEY_CURRENT_USER	Contiene la raíz de la información de configuración del usuario que ha iniciado sesión. Las carpetas del usuario, los colores de la pantalla y la configuración del Panel de control se almacenan aquí. Esta información está asociada al perfil del usuario. Esta clave a veces aparece abreviada como "HKCU".
HKEY_USERS	Contiene todos los perfiles de usuario cargados activamente en el equipo. HKEY_CURRENT_USER es una subclave de HKEY_USERS. HKEY_USERS puede aparecer abreviada como "HKU".
HKEY_LOCAL_MACHINE	Contiene información de configuración específica del equipo (para cualquier usuario). Esta clave a veces aparece abreviada como "HKLM".
HKEY_CLASSES_ROOT	Es una subclave de HKEY_LOCAL_MACHINE\Software. La información que se almacena aquí garantiza que cuando abra un archivo con el Explorador de Windows se abrirá el programa correcto. Esta clave a veces aparece abreviada como "HKCR". En el caso de Windows 2000, esta información se almacena en dos claves: HKEY_LOCAL_MACHINE y HKEY_CURRENT_USER. La clave HKEY_LOCAL_MACHINE\Software\Classes contiene la configuración predeterminada que se puede aplicar a todos los usuarios del equipo local. La clave HKEY_CURRENT_USER\Software\Classes contiene la configuración que invalida la configuración predeterminada y que se aplica únicamente al usuario interactivo. La clave HKEY_CLASSES_ROOT proporciona una vista del Registro que combina la información de estos dos orígenes. HKEY_CLASSES_ROOT también proporciona una vista combinada para los programas diseñados para versiones anteriores de Windows. Para cambiar la configuración del usuario interactivo, se deben realizar los cambios en HKEY_CURRENT_USER\Software\Classes en lugar de en HKEY_CLASSES_ROOT. Para cambiar la configuración predeterminada, se deben realizar los cambios en HKEY_LOCAL_MACHINE\Software\Classes. Si escribe valores en una clave de HKEY_CLASSES_ROOT, el sistema almacena la información en HKEY_LOCAL_MACHINE\Software\Classes. Si escribe valores para una clave en HKEY_CLASSES_ROOT y la clave ya existe en HKEY_CURRENT_USER\Software\Classes, el sistema almacenará la información ahí, en lugar de en HKEY_LOCAL_MACHINE\Software\Classes.

HKEY_CURRENT_CONFIG	Contiene información acerca del perfil de hardware que utiliza el equipo local cuando se inicia el sistema.

Tabla 11. Descripción de las secciones del registro

Es importante conocer el tipo de información que se almacena en el registro para poder determinar cuáles de las entradas del registro que han cambiado corresponden a restos dejados por un software a analizar.

Extraído de Microsoft: http://support.microsoft.com/kb/256986/es

5.3.1.3. Procedimiento

A continuación se definen los pasos que se llevaran a cabo para realizar el análisis. Se trata de unos pasos genéricos para analizar los cambios que genera cualquier tipo de programa al ser instalado en Windows.

1. Instalación de la aplicación Virtual Box.
2. Instalación del sistema operativo Microsoft® Windows en una máquina Virtual.
3. Creación de un "snapshot" con el software Systracer.
4. Instalación del software a analizar.
5. Creación de un segundo "snapshot" con el software Systracer.
6. Realizar comparativa de ambas instantáneas.
7. Desinstalación del software a analizar.
8. Creación de un segundo "snapshot" con el software Systracer.
9. Realizar comparativa de la instantánea vacía y la del programa desinstalado.

En los distintos análisis que se realizarán se podrán ver modificaciones en el registro producidas por el simple uso del sistema operativo, principalmente en los registros relativos al usuario.

5.3.1.4. PRUEBA 1: CAMOUFLAGE

Camouflage es un programa que permite ocultar ficheros de datos dentro de imágenes que, después, se puede recuperar con la misma herramienta. Tras la instalación modifica muchos elementos de Windows, creando, por ejemplo, nuevas opciones en los menús contextuales del navegador de archivos "explorer" de Windows.

Se podrá observar como añade y modifica una gran cantidad de campos en el Registro de Windows (solo se mostrarán unos pocos para este estudio) y algunos ficheros temporales.

Software	Camouflage
Versión	1.2.1
Información y descargas	http://camouflage.unfiction.com/

Tabla 12. Información del software Camouflage

- ***Instalación:***

Cambios en el sistema entre un sistema sin el software y otro con el software instalado.

- ***Registros identificados***

Podemos observar como el instalador del software Camouflage ha añadido:

- Archivos temporales que se han conservado tras la instalación.
- Varios enlaces simbólicos en el menú de inicio de Windows.
- La librería MSCOMCTL.OCX. Tras investigar su naturaleza en internet, se llega a la conclusión de que se trata de una librería ActiveX que contiene controles usados para la representación de interfaces graficas típicas de Windows (botones, combo list, etc.). Previamente también hemos encontrado en el registro nuevas entradas referentes a estos tipos de elementos visuales. Se trata de una librería bastante genérica por lo que parece que no puede ayudarnos a averiguar si lo ha añadido este programa en concreto porque es usado por otros muchos programas.
- Añade nuevos archivos en el directorio \Windows\prefetch. Como se ha comentado anteriormente, estos archivos son catálogos que permiten acelerar la ejecución de aplicaciones en Windows. Vemos archivos referentes al instalador, desinstalador y al propio ejecutable del programa Camouflage.

- ***Desinstalación:***

Cambios en el sistema entre un sistema sin el software y otro con el software desinstalado.

- ***Registros identificados***

Vemos que tras la desinstalación aun permanecen gran cantidad de registros que hacen referencia al software.

- ***Ficheros modificados/añadidos***

Se observa, que tras la desinstalación solo quedan los archivos temporales que utilizó el instalador y los archivos de Prefetch, así como algunas modificaciones en los ficheros que contienen el registro de Windows.

5.3.1.5. Prueba 2: MP3Stego

MP3Stego es un software que permite ocultar un archivo dentro de un archivo de sonido con el formato mp3. Es conceptualmente parecido a Camouflage pero utilizando un medio distinto de ocultar los datos (los archivos de sonido).

Este programa no instala archivos, se ejecuta directamente y se utiliza por línea de comandos. Lo usamos un par de veces antes de hacer el análisis para poder evaluar si su uso deja algún rastro en el sistema.

Software	mp3stego
Versión	1.1.18

Información y descargas	http://www.petitcolas.net/fabien/steganography/mp3stego/

Tabla 13. Información del software Mp3stego

- ***Instalación:***

Cambios en el sistema entre un sistema sin el software y otro con el software usado.

- ***Registros identificados***

Al no instalarse el programa, solo copiarlo al disco duro, deja pocos restos en el registro. En este caso no se ha observado ninguna diferencia en el registro relacionada con el software.

- ***Ficheros modificados/añadidos***

Podemos ver que los únicos archivos nuevos pertenecen a la carpeta y subcarpetas donde se ha copiado el contenido el programa. También podemos ver un nuevo archivo en *\Windows\prefetch* que hace referencia al archivo "encode.exe", nombre el ejecutable que codifica los archivos mp3.

- ***Desinstalación:***

Cambios en el sistema entre un sistema sin el software y otro con el software desinstalado.

Se observan los mismos cambios que tras la ejecución, salvo que los archivos borrados ya no se encuentran en el disco duro.

En el directorio *\Windows\prefetch* sigue estando el archivo que hace referencia al ejecutable "encode.exe".

5.3.1.6. PRUEBA 3: TOR

Tor es un proyecto software que transmite comunicaciones a través de una red distribuida de repetidores llevados por voluntarios de todo el mundo, permitiendo realizar conexiones a través de internet anónimas y difícilmente monitorizables. También permite saltar restricciones establecidas en entornos de producción para ciertas aplicaciones.

Software	Tor
Versión	**0.2.0.32**
Información y descargas	http://www.torproject.org/download.html.es

Tabla 14. Información del software TOR

- ***Instalación:***

Cambios en el sistema entre un sistema sin el software y otro con el software instalado.

- ***Registros identificados***

En el registro se puede observar cómo se han instalado 3 aplicaciones:

- Tor – La propia aplicación.
- Privoxy – Programa que funciona como proxy web.

- Vidalia – Aplicación que monitoriza el estado de Tor.

- ***Ficheros modificados/añadidos***

Podemos ver cómo, además de los archivos propios de las aplicaciones, se añaden directorios y archivos en la ruta ***\Documents and Settings\Administrador\Datos de programa\.*** También se puede ver que ha añadido 4 archivos al directorio **\Windows\Prefetch**.

El programa también añade al inicio del sistema un ejecutable de una aplicación que incluye en la instalación, de manera que se ejecuta al iniciar la sesión.

- ***Desinstalación:***

Cambios en el sistema entre un sistema sin el software y otro con el software desinstalado.

- ***Registros identificados***

Vemos que hay multitud de registros que aún hacen referencia a las aplicaciones Vidalia, esto se debe a que el desinstalador de Tor no desinstala la aplicación Vidalia. Sin embargo, la aplicación Vidalia solo tiene sentido si se tiene instalado un cliente de Tor, por lo tanto, la existencia de este software nos puede indicar que ha sido instalado Tor en algún momento.

- ***Ficheros modificados/añadidos***

Tras la desinstalación no se observan restos de archivos de Tor. Tan solo los mismos archivos en ***\WINDOWS\Prefetch*** que había en la instalación y todos los archivos referentes al software Vidalia, que tal como se indicaba anteriormente, continúa instalado.

5.3.1.7. Conclusión

Tras analizar los efectos que se han producido en el sistema tras usar y borrar estas tres aplicaciones podemos llegar a una serie de conclusiones:

- Existen dos tipos de programas:
  - Aquellos que se instalan mediante paquetes msi(Microsoft® installer) con aplicaciones tipo Wise o InstallShield.
  - Aquellos que no se instalan y se utilizan en modo "portable". Estos últimos son más difíciles de detectar y no modifican el registro de Windows a no ser que el programa en cuestión realice modificaciones mediante su uso.
- La instalación de un software con un paquete msi, en general genera cambios palpables en el registro de Windows. Estos cambios pueden ser debidos al añadido de nuevos registros así como a la modificación de registros ya existentes.

- Existen una serie de directorios donde se pueden encontrar pruebas de la existencia del software instalado en Windows. Ya se ha hecho referencia a estos directorios anteriormente.
- Tras la desinstalación de un software hemos comprobado cómo algunos programas mantienen algunos registros sin borrar, de manera que se puede averiguar si el software ha estado instalado en algún momento.
- En general, cuando se ejecuta una aplicación, se crea un nuevo archivo en el directorio ***\WINDOWS\Prefetch ,*** independientemente de la naturaleza de la aplicación.

Prefetch es una carpeta del sistema operativo que introduce a Windows XP en el uso de una especie de caché de programas. Eso quiere decir que la información contenida en esta carpeta refiere a los programas, librerías y todo tipo de archivos que se utilizan habitualmente, ya sea por nuestro trabajo o por el trabajo del sistema operativo.
De esta manera, los accesos futuros serán más rápido puesto que el sistema operativo tiene una referencia directa en Prefetch. Esta nueva característica es beneficiosa si se ejecutan pocas aplicaciones.

En principio, los archivos de esta carpeta no son borrados ni sustituidos a no ser que se haga de forma manual o mediante alguna aplicación de limpieza de archivos temporales y registro de Windows.

Esta función de Windows nos puede ayudar de gran manera en nuestro objetivo de construir una aplicación de análisis forense, puesto que nos proporciona una manera rápida de ver si un determinado ejecutable ha sido ejecutado en algún momento, permitiendo descubrir si una aplicación, aunque no modifique el registro, ha sido usada en un equipo, incluyendo las llamadas “aplicaciones portables”.

5.3.1.8. Enfoque

Tras analizar las conclusiones, a priori se plantean una serie de acciones que debería realizar el software de análisis forense para detectar estas aplicaciones.

1. Realizar búsquedas concretas en el Registro de Windows – Búsqueda en la que se índica que registro debería existir en que ruta concreta del registro para concluir que el software ha sido instalado.
2. Realizar búsquedas generales en una sección del Registro de Windows – Búsqueda de una cadena concreta en una sección del registro.
3. Realizar búsquedas generales en todo el Registro de Windows – Búsqueda de una cadena concreta en todo el registro de Windows.
4. Realizar búsquedas de archivos en Rutas determinadas – Búsqueda de un archivo en una ruta concreta.
5. Realizar búsquedas de archivos en todo el disco – Búsqueda de un archivo en todo el disco duro seleccionado.

6. Realizar una búsqueda en la carpeta ***\WINDOWS\Prefetch (%WINDIR%\prefecht***) – Buscar un archivo en dicha carpeta en cuyo nombre contenga una cadena determinada. Comúnmente se buscara el nombre del ejecutable. Por ejemplo "camouflaje.exe".
7. Realizar una búsqueda de un archivo determinado **X** en una carpeta determinada **Y** – Buscar un archivo concreto en la carpeta seleccionada. Se deberían poder utilizar variables de entorno como **%APPDATA%** .

Estas acciones servirán para determinar los dos estados (software residente o desinstalado) aunque no necesariamente tendrían que ser iguales. Por ejemplo, se puede producir una búsqueda en el registro X para ver si el programa "Napster" está **instalado** pero para ver si el programa ha sido **desinstalado** se ejecutaría una búsqueda del registro Y.

5.3.2. Análisis de la instalación de Programas en Linux

5.3.2.1. Propósito

El objetivo de este análisis es averiguar los cambios que se producen en el sistema operativo Linux tras realizar la instalación de un nuevo software. Para ello se instalarán varios programas en distintas distribuciones utilizando los distintos gestores de paquetes que incluyen.

5.3.2.2. Software utilizado

En este apartado se detalla el software utilizado para realizar estas pruebas.

- **Sistemas operativos**

Ubuntu - Distribución basada en Debian centrada en la facilidad de uso. Probablemente la distribución más extendida y con mucho soporte en la comunidad. El entorno de escritorio por defecto es GNOME. Utiliza el gestor de paquetes "apt" con paquetes ".deb".

Debian – Se utilizará una instalación mínima de Debian para comprobar el efecto de la compilación de los distintos programas en el sistema sin usar ninguna herramienta de gestión de paquetes. Se utiliza Debian para poder servirse de APT para instalar las dependencias necesarias previas a la compilación.

Fedora - Esta es una distribución patrocinada por RedHat y soportada por la comunidad. Es fácil de instalar y de buena calidad. Existen distintas versiones con distintos escritorios. Utiliza el gestor de paquetes RPM.

Para realizar este análisis se va a utilizar el siguiente software:

- **Sun VirtualBox**: Aplicación que permite simular una maquina virtual dentro de un sistema anfitrión.
- **Strace:** Comando de Linux que muestra una traza con todas las llamadas del sistema que se realizan tras la ejecución del sistema, incluyendo las llamadas "open" a ficheros. Muy útil para ver los ficheros que se crean al lanzar una instalación con un gestor de paquetes.

5.3.2.3. Información adicional

Es conveniente, antes de comenzar el análisis, detenerse a estudiar algunos aspectos del sistema operativo Linux.

- **Gestión de Paquetes**

Las distribuciones de Linux actuales, en su mayoría utilizan herramientas de gestión de paquetes para instalar nuevas aplicaciones. Los gestores de paquetes son aplicaciones propias de cada distribución que permiten la instalación directa (sin tener que compilar) de una aplicación preparada para dicho sistema operativo.

La mayoría de estos gestores permiten descargar dichos paquetes y sus dependencias, de manera que la instalación de una nueva aplicación sea lo más sencilla posible para el usuario.

Sin embargo, también se pueden instalar las aplicaciones a través del método tradicional, compilándolas. En general las aplicaciones vienen preparadas en un archivo comprimido "tar.tgz" e incluyen un fichero "MAKEINSTALL" que ejecuta las acciones necesarias para realizar la instalación.

En principio, ambos sistemas de instalación de nuevo software deberían dejar los mismos archivos en el sistema, exceptuando el propio paquete o archivo "tar.tgz" que se haya descargado/copiado antes de la instalación.

- **Directorios de Linux**

Existen una serie de rutas genéricas en Linux donde se almacena información acerca del uso e instalación de programas.

- **/bin/:** Comandos/programas binarios esenciales (cp, mv, ls, rm, etc.),
- **/etc/:** Ficheros de configuración utilizados en todo el sistema y que son específicos del ordenador
- **/etc/opt/:** Ficheros de configuración utilizados por programas alojados dentro de /opt/
- **/home/:** Directorios de inicios de los usuarios (Opcional)
- **/lib/:** Bibliotecas compartidas esenciales para los binarios de /bin/,/sbin/ el núcleo del sistema.
- **/mnt/:** Sistemas de ficheros montados temporalmente.
- **/media/:** Puntos de montaje para dispositivos de medios como unidades lectoras de discos compactos o tarjetas de memoria.
- **/opt/:** Paquetes de aplicaciones estáticas.

- **/proc/:** Sistema de ficheros virtual que documenta sucesos y estados del núcleo. Contiene principalmente ficheros de texto.
- **/root/:** Directorio de inicio del usuario root (super-usuario) (Opcional)
- **/sbin/:** Comandos/programas binarios de administración de sistema.
- **/tmp/:** Ficheros temporales
- **/usr/:** Jerarquía secundaria para datos compartidos de solo lectura.
- **/usr/bin/:** Comandos/programas binarios.
- **/usr/include/:** Ficheros de inclusión estándar (cabeceras de cabecera utilizados para desarrollo).
- **/usr/lib/:** Bibliotecas compartidas.
- **/usr/share/:** Datos compartidos independientes de la arquitectura de sistema. Imágenes, ficheros de texto, etc.
- **/usr/src/:** Códigos fuente (Opcional)
- **/usr/X11R6/:** Sistema X Window, versión 11, lanzamiento 6 (Opcional)
- **/usr/local/:** Jerarquía terciaria para datos compartidos de solo lectura específicos del ordenador que los comparte.
- **/var/:** Ficheros variables, como son logs, bases de datos, directorio raíz de servidores HTTP y FTP, colas de correo, ficheros temporales, logs de aplicaciones.
- **/var/cache/:** Cache da datos de aplicaciones. Almacena los paquetes obtenidos con apt en debían y derivados.
- **/var/lib/:** Información de estado variable. Algunos servidores como MySQL y PostgreSQL almacenan sus bases de datos en directorios subordinados de éste.
- **/var/log/:** Ficheros y directorios de registro del sistema (logs).
- **/var/spool/:** Colas de datos de aplicaciones.
- **/var/tmp/:** Ficheros temporales preservados entre reinicios.

- **Procedimiento**

1. Instalación de la aplicación Virtual Box.
2. Instalación de la distribución Linux en una maquina Virtual.
3. Instalación del software a analizar mediante un gestor de paquetes. Usar el comando strace para comprobar que archivos crea/modifica.
4. Desinstalación del software a analizar. Usar el comando strace para comprobar que archivos borra/modifica.

5.3.2.4. Prueba 1: StegHide

Steghide es un programa de estenografía que permite ocultar datos en varios tipos de imagen- y archivos de audio.

Sus características incluyen el compactado y el encriptado de los datos adjuntos, y revisión automática de integridad usando un checksum. Se reconocen los formatos de archivo JPEG, BMP, WAV y AU para usar como archivos de encubrimiento. No existen restricciones en el formato de los datos ocultos.

Software	StegHide
Versión	0.5.1
Información y descargas	http://steghide.sourceforge.net/

Tabla 15. Información del software StegHide

- **Fedora**

  - ***Instalación:***

Instalamos software con la utilidad de gestión de paquetes ***GPK Application*** y observamos que se han creado varios archivos.

También instala dos librerías que necesita para su funcionamiento. Pero la simple existencia de las mismas no nos valdría para verificar que ha sido instalado.

  - ***Desinstalación:***

La desinstalación parece haber sido limpia y haber borrado todos los ficheros que inicialmente fueron instalados.

Tan solo se han encontrado referencias al software en el log ubicado en /var/log/yum.log.

También en el contenido del comando ***history***, se puede observar si ha sido ejecutado desde línea de comandos recientemente.

- ***Ubuntu***

  - **Instalación**

Instalamos software con la utilidad de gestión de paquetes ***Synaptic*** y observamos que se han creado nuevos archivos.

Al igual que en Fedora, se instalas dos librerías que necesita para su funcionamiento. Pero la simple existencia de las mismas no nos valdría para verificar que ha sido instalado.

  - ***Desinstalación***

El gestor ***Synaptic*** permite dos opciones a la hora de eliminar un programa. ***Eliminar*** y ***Eliminar completamente.*** El primero solo elimina los archivos propios de la aplicación (ejecutables, documentación, etc). El segundo elimina además los archivos de configuración asociados al usuario.

Vamos a borrar la aplicación usando la segunda opción, cuyo equivalente en modo texto es la opción "—purge".

En primer lugar, observamos que tras desinstalar el software, el paquete steghide_0.5.1-8_i386.deb continua en el directorio /var/cache/apt/archives.

También en el "log" /var/log/dpkg.log se encuentran referencias a la instalación y desinstalación del software.

- **Debian (compilación)**

  - **Instalación**

Tras la instalación de todas las dependencias requeridas con el comando "apt-get" se procede a compilar el archivo "***tar.gz***" descargado de la propia página de Steghide en SourceForge. El archivo debe descomprimirse, dejando el contenido en un directorio temporal.

Observamos tras la instalación que se han instalado varios archivos en el directorio **/usr/share/locale** y en **/usr/bin/steghide**.

  - ***Desinstalación***

Se puede realizar un "***make uninstall***" para desinstalar la aplicación. Esto dejaría los archivos temporales que fueron creados al compilar en el directorio donde se descomprimió el archivo tar.gz.

Si se ejecuta el comando "***make clean***", los archivos temporales de compilación se borrarán, sin embargo el archivo descargado y los archivos originales que contenía se mantendrán en el disco duro.

Los logs del compilador se almacenan en el archivo "***config.log***" del propio directorio usado para la compilación. Este archivo que no es borrado con ninguno de los dos comandos anteriores.

5.3.2.5. Prueba 2: GNUPG

GnuPG es un programa, al igual que PGP, para cifrar y firmar nuestros ficheros o correos electrónicos. La principal diferencia con PGP es que es completamente libre y la licencia que lo acompaña es totalmente gratuita incluso para usos comerciales.

Software	GNUPG
Versión	2.2
Información y descargas	http://steghide.sourceforge.net/

Tabla 16. Información del software GNUPG

- ***Fedora***

  - **Instalación**

Instalamos software con la utilidad de gestión de paquetes ***GPK Application*** y observamos que se han creado numerosos archivos.

Se puede observar que ha dejado gran cantidad de archivos en distintos directorios del sistema, incluyendo documentación y páginas de manual accesibles mediante el comando "***man***".

- ***Desinstalación***

Tras desinstalar con la misma aplicación de gestión de paquetes rpm se han encontrado las siguientes referencias al software en distintos archivos de log del directorio /var/log:

También se puede observar que el paquete se ha quedado almacenado en la ruta **"/var/cache/yum/fedora/packages**".

- **Debian (compilación)**

  - **Instalación**

Tras la instalación de todas las dependencias requeridas con el comando "apt-get" se procede a compilar el archivo "***tar.gz***" descargado de la propia página de GNUPG. El archivo debe descomprimirse, dejando el contenido en un directorio temporal. También observamos tras la instalación que se han creado nuevos archivos.

  - ***Desinstalación***

Analizando los logs de desinstalación nos encontramos con el mismo caso que antes. Se han borrado los mismos ficheros que se instalaron, sin embargo se mantienen los archivos temporales.

También, al igual que antes, se conserva en el histórico de comandos, los comandos de compilación que hacen referencia a dicho programa.

- ***Ubuntu (instalación)***

  - **Instalación**

Instalamos software con la utilidad de gestión de paquetes ***Synaptic*** y observamos que se han creado la siguiente lista de archivos:

  - ***Desinstalación***

El gestor ***Synaptic*** permite dos opciones a la hora de eliminar un programa. ***Eliminar*** y ***Eliminar completamente.*** El primero solo elimina los archivos propios de la aplicación (ejecutables, documentación, etc). El segundo elimina además los archivos de configuración asociados al usuario.

Vamos a borrar la aplicación usando la segunda opción, cuyo equivalente en modo texto es la opción "—purge".

En primer lugar, observamos que tras desinstalar el software, el paquete ***.deb*** continua en el directorio /var/cache/apt/archives.

También en el "log" ***/var/log/dpkg.log se*** y el log ***/var/log/scrollkepper-update*** encuentran referencias a la instalación y desinstalación del software. Estos últimos hacen referencia al paquete de la interfaz **gui** para gnupg que se instaló junto a la aplicación.

5.3.2.6. Conclusión

Tras analizar los efectos que se han producido en el sistema tras usar y borrar estas dos aplicaciones podemos llegar a una serie de conclusiones:

- Prácticamente todos los restos que deja un software instalado en Linux se encuentran en forma de archivos o contenido en texto plano dentro de los mismos.

- **Existen dos tipos de instalación en Linux:**
  - Gestores de Paquetes
  - Compilación de código fuente

- **Mediante un gestor de paquetes**:

Las distribuciones más extendidas entre la mayor parte de usuarios incluyen aplicaciones que proporcionan una forma sencilla de instalar y actualizar el software. Mediante pocos comandos o una sencilla interfaz gráfica descargan un paquete preparado para la distribución que estemos usando. También, en casi todas estas aplicaciones se puede hacer la instalación en local, obteniendo el paquete previamente.

Estas aplicaciones, como sucede habitualmente en los sistemas Unix, dejan restos de las acciones que ejecutan en los logs del sistema, ubicados en archivos de texto plano en el directorio "***/var/log***". También hemos observado como el paquete que la aplicación se descarga para instalar se queda almacenado en un directorio junto al resto de paquetes descargados.

- **Mediante compilación del código fuente**.

Normalmente el código fuente de las aplicaciones incluye archivos MakeFile y Config que simplifican en gran medida la tediosa tarea de compilar un programa. Estos ficheros también guardan logs del proceso de compilado.

Los ficheros fuente han de ser borrados a mano porque los procesos de desinstalación no los eliminan.

5.3.2.7. Enfoque

Tras analizar las conclusiones, a priori se plantean una serie de acciones que debería realizar el software de análisis forense para detectar aplicaciones instaladas y/o desinstaladas en sistemas Linux.

- **Realizar búsquedas de archivos en Rutas determinadas:** Búsqueda de un archivo en una ruta concreta.
- **Realizar búsquedas de archivos en todo el disco:** Búsqueda de un archivo en todo el disco duro seleccionado.

- **Realizar búsquedas de archivos** cuyo nombre contenga una cadena en rutas de terminadas.
- **Realizar búsquedas de archivos** cuyo nombre contenga una cadena en todo el disco.
- **Comprobar si el contenido de un archivo log** contiene una cadena determinada en texto plano.
- **Comprobar si algún archivo en un directorio determinado** contiene una cadena determinada en texto plano.
- **Comprobar si algún archivo en todo el disco duro** contiene una cadena determinada.

Estas acciones servirán para determinar los dos estados (software residente o desinstalado) aunque no necesariamente tendrían que ser iguales. El proceso de buscar el programa desinstalado puede realizar búsquedas distintas a buscar el programa instalado, iguales o de ambos tipos.

5.3.3. Análisis de la instalación de Programas en MAC OS

5.3.3.1. Preparación

- ***Propósito***

El objetivo de este análisis es averiguar los cambios que se producen en el sistema operativo MAC OS tras realizar la instalación de un nuevo software. Para ello se comprobaran los cambios producidos en la versión actual del sistema operativo de Mac "***Leopard***" tras la instalación de varias aplicaciones.

5.3.3.2. Sistema operativo

- ***Leopard***

Mac OS X v10.5 «Leopard» es la sexta revisión del sistema operativo de Apple Mac OS X para equipos Macintosh. Leopard se encuentra disponible en 2 formas: una versión de escritorio para uso personal y una versión para servidores conocida como Mac OS X Server.

Nosotros utilizaremos la versión de escritorio para evaluar los cambios producidos en el sistema. MAC usa un **kernell** llamado ***Darwin BSD*** basado en ***FreeBSD***, inspirado a su vez en el sistema ***UNIX***. Esto significa que nos vamos a encontrar bastantes semejanzas con el sistema operativo LINUX.

En general MAC está desarrollado como un sistema operativo altamente visual muy centrado en la interfaz gráfica, sin embargo a través de una consola semejante al xterm de los sistemas Unix se puede tener un control más exhaustivo del sistema.

Para realizar este análisis se va a utilizar el siguiente software:

- ***Time Machine***

Se trata de una aplicación que permite realizar backups completos de la partición que incluye el sistema operativo en una segunda partición.

Este software permite mediante una interfaz muy atractiva recuperar archivos, carpetas o el contenido total del disco recuperando alguno de los backups periódicos que realiza sobre el sistema.

En primer lugar, tras realizar un Backup del sistema navegaremos a la partición secundaria donde se almacenan las Copias de seguridad del sistema.

Observamos que los backups se almacenan en distintos directorios identificados con la fecha y hora a la que se realizo la copia, pudiendo acceder a los cambios que se han realizado entre un backup y el siguiente.

- ***Diff***

Considerando los backups creados por la aplicación Time Machine como estados del sistema de ficheros en un momento dado, mediante el comando diff podemos realizar comparaciones entre ellos.

El comando Diff compara ficheros línea a línea y te muestra las diferencias. Utilizaremos los parámetros:

- -r Recorre recursivamente los directorios
- -q Muestra tan solo los elementos que son diferentes en distintas líneas.

5.3.3.3. Información adicional

Es conveniente, antes de comenzar el análisis, detenerse a estudiar algunos aspectos del sistema operativo Mac Os.

- **Gestión de Paquetes**

Las aplicaciones de MAC OS suelen obtenerse en un paquete instalable. Estos paquetes son comúnmente archivos de extensión ***.dmg***.

Cuando se descarga uno de estos paquetes, automáticamente el sistema lo "monta" como si de una unidad externa se tratara y, en la mayoría de los casos aparece una pantalla de presentación del software indicando las instrucciones para instalar.

Estos paquetes .dmg se quedan almacenados en el disco, en la ubicación en la que se haya descargado/copiado inicialmente.

La desinstalación de software se realiza borrando manualmente los archivos. Esto es así porque en principio las aplicaciones de MAC suelen estar en un único archivo. Sin

embargo algunas aplicaciones requieren librerías, pudiendo dejar restos una vez borrado el programa principal.

- **Directorios de MAC OS**

Existen una serie de rutas genéricas en MAC OS donde se almacena información acerca del uso e instalación de programas.

- ***/Applications****:* Directorio donde se encuentran las aplicaciones.
- ***/Developer****:* Espacio utilizado por las herramientas de desarrollo de Apple (Apple's Developer Tools).
- ***/Library****:* Librerías compartidas, archivos necesarios para que el sistema y las aplicaciones funcionen correctamente, también incluye preferencias, configuraciones y otros archivos de personalización.
- **/Network***: Archivos* de red, conexiones con servidores, librerías de conexión, etc.
- **/System**: Archivos de sistema, librerías preferencias y configuración críticas para el arranque y el funcionamiento correcto de Mac OS X.
- ***/Users****:* Las cuentas de usuario del sistema y todas las configuraciones únicas, bastante parecido al directorio /home de Linux.
- **/Volumes:** Dispositivos y volúmenes que se han montado en el sistema ya sean virtuales o reales como discos físicos, CDs, DVDs, discos de imagen DMG, etc.
- */* - la raíz, presente en todos los sistemas derivados de UNIX. es el directorio padre de todo el árbol de directorios.
- ***/bin****:* Directorio de binarios comunes, mantiene archivos relacionados con el arranque y las operaciones básicas del sistema, es necesario para que éste funcione correctamente.
- ***/etc****: Configuración* local del sistema, mantiene datos de administración y configuración de los archivos del sistema.
- ***/dev****: Ficheros* de dispositivo, representan periféricos como teclado, ratón, etc.
- **/usr***:* El segundo en nivel de importancia incluye subdirectorios que contiene información sobre la configuración del sistema.
- ***/sbin****:* Otro directorio que contiene binarios esenciales para la administración del sistema.
- ***/tmp****:* Archivos temporales, cache, etc.
- ***/var****:* Directorio de datos variables, contiene archivos cuyo contenido varía mientras el sistema se ejecuta.

- ***Procedimiento***

1. Configuración de la herramienta ***Time Machine.***
2. Realizar un Backup con la herramienta ***Time Machine.***
3. Instalar Software seleccionado
4. Realizar un segundo Backup con la herramienta ***Time Machine.***
5. Utilizar el comando ***diff*** para evaluar los cambios.

5.3.3.4. PRUEBA 1: GNUPG

GnuPG es un programa, al igual que PGP, para cifrar y firmar nuestros ficheros o correos electrónicos. La principal diferencia con PGP es que es completamente libre y la licencia que lo acompaña es totalmente gratuita incluso para usos comerciales.

El software de GNUPG para Mac se llama MACPG.

Software	GNUPG
Versión	1.4.8
Información y descargas	http://macgpg.sourceforge.net/

Tabla 17. Información del software GNUPG

- ***Instalación:***

Descargamos el paquete instalable de la página de GNUPG y seguimos las instrucciones para instalarlo. Tras la instalación observamos bastantes ficheros modificados.

Observamos, además de la creación de estos nuevos archivos, que en el archivo ***/private/var/log/install.log*** se hace referencia al software instalado.

También observamos que se ha modificado los archivos:

/var/db/.TimeMachine.Results.plist
/Users/air/Library/Preferences/com.apple.finder.plist

/Applications/.DS_Store

Tras investigar por internet, se averigua que los dos primeros son ficheros de preferencias de las aplicaciones **Time Machine** y **Finder**. El tercero almacena información relativa a nuestra configuración en Mac OS X para el directorio que lo contiene.

- ***Desinstalación:***

Tal como indicamos antes, la desinstalación se realiza a mano. A priori consistirá en borrar el archivo de la carpeta ***/Applications.*** Dejando el resto de archivos y logs indicados anteriormente intactos.

5.3.3.5. PRUEBA 2: CRYPTIX

Colección de utilidades de criptografía está disponible para Mac, una herramienta muy completa con la que podremos generar contraseñas aleatorias con encriptaciones de 64 o 128 bits, usar OpenSSL para cifrar con AES, BF; CAST, DES o RC, codificar y descodificar textos y ficheros con una amplia gama de algoritmos de cifrado, incluyendo a funciones hash con algoritmos como el MD5 o el SHA.

Software	CRYPTIX
Versión	0.64b
Información y descargas	http://www.rbcafe.com/Welcome

Tabla 18. Información del software Cryptix

- ***Instalación***

Instalamos el software y observamos, de nuevo, varios cambios. Al igual que en la ocasión anterior, se han creado nuevos archivos en los directorios /applications y se han modificado varios archivos de preferencias.

- ***Desinstalación***

Tal como indicamos antes, la desinstalación se realiza a mano. A priori consistirá en borrar el archivo de la carpeta ***/Applications.*** Dejando el resto de archivos y logs indicados anteriormente intactos.

5.3.3.6. PRUEBA 3: XAES

Xaes es un port hacia Mac de un proyecto GNU GPL que permite la encriptación de textos usando el estándar de cifrado avanzado AES, bien usando la interfaz de Xaes bien desde el menú de servicios del sistema.

Software	XAES
Versión y Información descargas	1.0 http://homepage.mac.com/stephaneleon/PhotoAlbum13.html

Tabla 19. Información del software XAES

- ***Instalación***

Al igual que en la ocasión anterior, se han creado nuevos archivos en los directorios /applications y /Library y se han modificado varios archivos de preferencias.

- ***Desinstalación***

Tal como indicamos antes, la desinstalación se realiza a mano. A priori consistirá en borrar el archivo de la carpeta ***/Applications.*** Dejando el resto de archivos y logs indicados anteriormente intactos.

5.3.3.7. Conclusión

Tras analizar los efectos que se han producido en el sistema tras usar y borrar estas dos aplicaciones podemos llegar a una serie de conclusiones:

- Prácticamente todos los restos que deja un software instalado en MAC e encuentran en forma de archivos o contenido en texto plano dentro de los mismos.
- Las aplicaciones MAC habitualmente están compuestas de pocos archivos.
- Los ejecutables de aplicaciones MAC suelen encontrarse en el directorio ***/Applications***
- Los librerías usadas por aplicaciones MAC suelen encontrarse en el directorio ***/Library***

5.3.3.8. Enfoque

Tras analizar las conclusiones, a priori se plantean una serie de acciones que debería realizar el software de análisis forense para detectar aplicaciones instaladas y/o desinstaladas en sistemas MAC OS. Se tratan de las mismas acciones que se determinaron en los sistemas Linux.

1. Realizar búsquedas de archivos en Rutas determinadas – Búsqueda de un archivo en una ruta concreta.
2. Realizar búsquedas de archivos en todo el disco – Búsqueda de un archivo en todo el disco duro seleccionado.
3. Realizar búsquedas de archivos cuyo nombre contenga una cadena en rutas de terminadas.
4. Realizar búsquedas de archivos cuyo nombre contenga una cadena en todo el disco.
5. Comprobar si el contenido de un archivo log contiene una cadena determinada en texto plano.
6. Comprobar si algún archivo en un directorio determinado contiene una cadena determinada en texto plano.
7. Comprobar si algún archivo en todo el disco duro contiene, en el nombre, una cadena determinada.

Estas acciones servirán para determinar los dos estados (software residente o desinstalado) aunque no necesariamente tendrían que ser iguales. El proceso de buscar el programa desinstalado puede realizar búsquedas distintas a buscar el programa instalado.

5.4. Definición del Sistema

Durante este capítulo se va a realizar una descripción del sistema más completa y detallada que en capítulos anteriores del documento, en los que simplemente se han dibujado pequeños retazos del conjunto del mismo.

5.4.1. Determinación del alcance del sistema

Se va a crear una aplicación de análisis forense de aplicaciones instaladas en un equipo. La aplicación deberá ser capaz de informar al usuario de los elementos encontrados en el equipo que evidencien que una aplicación esta o ha estado instalado en el equipo. Esto se realizará desde una interfaz visual sencilla que en pocos pasos permita iniciar un análisis.

La aplicación se realizará en en lenguaje de programación **Java** para proporcionar versatilidad en la ejecución en distintos sistemas operativos. Estando inicialmente preparada para su correcto funcionamiento en los principales sistemas operativos más extendidos: Microsoft® Windows, Linux y Mac OS X.

Esta aplicación deberá identificar distintas aplicaciones instaladas y facilitar al usuario añadir nuevos detectores para aumentar el número de aplicaciones detectables.

5.4.2. Especificación de Estándares y Normas

El desarrollo del proceso de análisis está adaptado a partir de las siguientes normas:

- Métrica 3: Define una metodología de seguimiento y desarrollo de cualquier proyecto.
- Métrica UML [2]: Define un lenguaje de modelado para hacer diseños en metodologías orientadas a objetos.
- Para este proceso de análisis también mantenemos un estilo estándar para la recolección de requisitos, es el siguiente:

Identificador: **SR-xxx**		Tipo:
Prioridad: ☐Alta ☐ Media ☐ Baja	Fuente: ☐ Usuario ☐ Desarrollador	
Necesidad: ☐ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☐ Alta ☐ Media ☐ Baja	Verificabilidad: ☐ Alta ☐ Media ☐ Baja	
Estabilidad:		
Descripción:		

5.4.3. Restricciones Generales

A continuación se identificarán las restricciones que afectarán a la aplicación.

Interfaz Gráfica	**Prioridad**	**Baja**
• Interfaz amigable y sencilla de usar.		
• Lenguaje y simbología utilizada no ambigua.		
• Se debe orientar la aplicación a usuarios con un nivel básico de conocimientos en informática, permitiendo a usuarios más experimentados realizar configuraciones más extensas.		

Tabla 20. Restricciones de prioridad baja

Implantación	**Prioridad**	**Media**
• El sistema se utilizará en una computadora con la maquina Virtual de Java instalada para poder ejecutar aplicaciones Java.		
• Cliente (características mínimas): o Máquina Virtual Java. o Monitor SVGA.		
• Características del equipo: o Hardware suficiente para ejecutar la máquina virtual Java.		
• Cumplirá con los requisitos establecidos, así como los que se puedan definir en un futuro		

Tabla 21. Restricciones de prioridad media

Portabilidad y codificación	Prioridad	Alta
• Sistema que correrá bajo distintas plataformas gracias a las características de Java.		
• A priori estará preparado para detectar las aplicaciones en los sistemas operativos: o Windows o Linux o MAC OS		
• Cumplirá con los requisitos de usuario y SW establecidos así como los que se establezcan en un futuro.		

Tabla 22. Restricciones de prioridad alta

Se ha descompuesto las restricciones en 3 grupos para dotarles de un parámetro de riesgo. Todas las restricciones han de cumplirse sin excepción alguna, pero esto permite dotarles de una mayor prioridad a las mismas con el objetivo de aplicarlas antes y así tener cerrado la mayor probabilidad de riesgo en el menor tiempo posible.

5.4.4. Supuestos y Dependencias

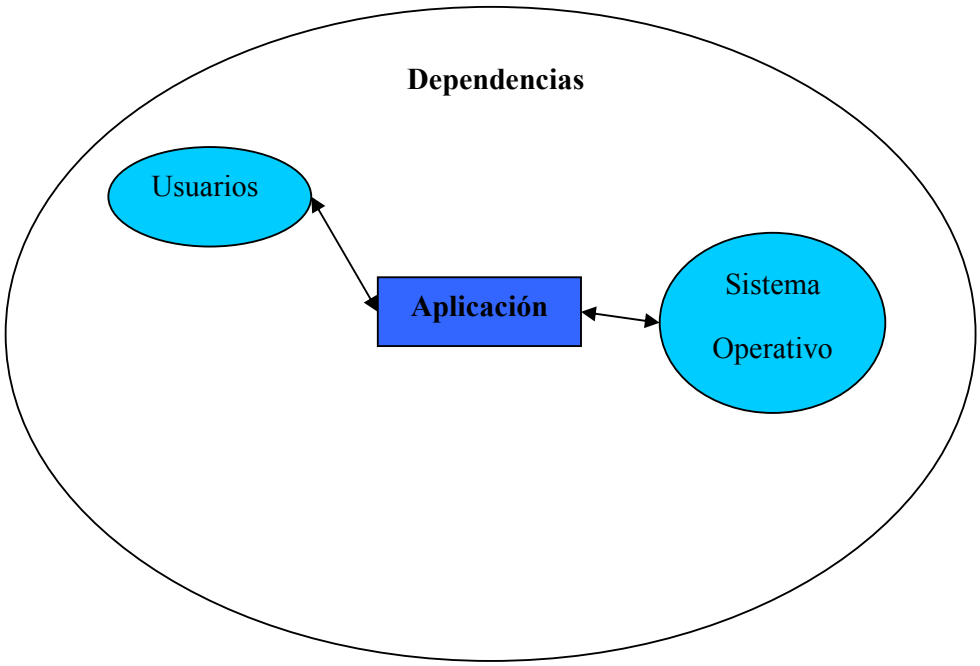


Gráfico 4. Dependencias del sistema

La aplicación dependerá del sistema operativo bajo el que ejecute. La aplicación no será vulnerable a los fallos propios del sistema operativo por lo que no se puede garantizar un

funcionamiento correcto bajo malas condiciones del mismo en situaciones, por ejemplo, de falta de memoria RAM o conflictos de dispositivos.

Nuestra aplicación utilizará el sistema de ficheros del sistema operativo, limitando la velocidad de análisis a la velocidad de acceso del propio sistema operativo. El rendimiento global de la aplicación dependerá del hardware utilizado. La velocidad de los dispositivos E/S serán determinantes para mejorar este aspecto.

La configuración del software será mínima. En principio solo habrá que indicar que programas se quieren buscar y en que unidad de disco.

5.4.5. Entorno Operacional

A continuación nos disponemos a analizar los medios elegidos para solucionar el problema con el objetivo de detallar posibles condicionantes y las restricciones que se generan en consecuencia.

5.4.5.1. Identificación de los Usuarios y Participantes finales

- **Tutor de proyecto:** El tutor de proyecto será el encargado de marcar las directrices del desarrollo y establecer los requisitos iníciales. Contribuirá a definir los requisitos finales y será el encargado de validarlos. También realizará revisiones periódicas del desarrollo del proyecto.
- **Analista, diseñador y desarrollador**: Es el encargado de realizar el proyecto en sí, incluyendo la especificación de requisitos, análisis, diseño, documentación y pruebas.
- **Usuario final:** Van a existir dos tipos de usuarios.
    - *Usuario inexperto*: Es el usuario que simplemente ejecutará la aplicación para averiguar el estado de su sistema. No necesitara realizar configuraciones avanzadas, modificar código ni profundizar en el desarrollo de plugins.
    - *Usuario avanzado*: Es el usuario que desea usar la herramienta para ampliar su funcionalidad, añadiendo formas de detectar software instalado o plugins para detectar un mayor número de aplicaciones.

5.5. Establecimiento de Requisitos Software

En este capítulo se redactarán, mediante diversas herramientas, el conjunto de requisitos software que definirán de manera detallada los aspectos y funcionalidades de la aplicación.

5.5.1. Especificación de Casos de Uso.

Como primer paso en la recopilación de requisitos software, se generarán un conjunto de diagramas de caso de uso que ayudarán a establecer las bases de los requisitos y la forma de interactuar del software con el usuario.

5.5.1.1. Diagrama de Casos de Uso.

En este apartado se presentará el diagrama de Caso de Uso que representará la funcionalidad del sistema. Posteriormente se describirá textualmente cada Caso de Uso.

En una primera apreciación podemos observar los tres tipos de roles que puede desempeñar un actor que interactúe con el sistema:

- Usuario Básico
- Usuario Avanzado

En los que cada uno permite realizar una serie de interacciones con el sistema como vemos en el diagrama:

(En la siguiente página se muestra el diagrama)

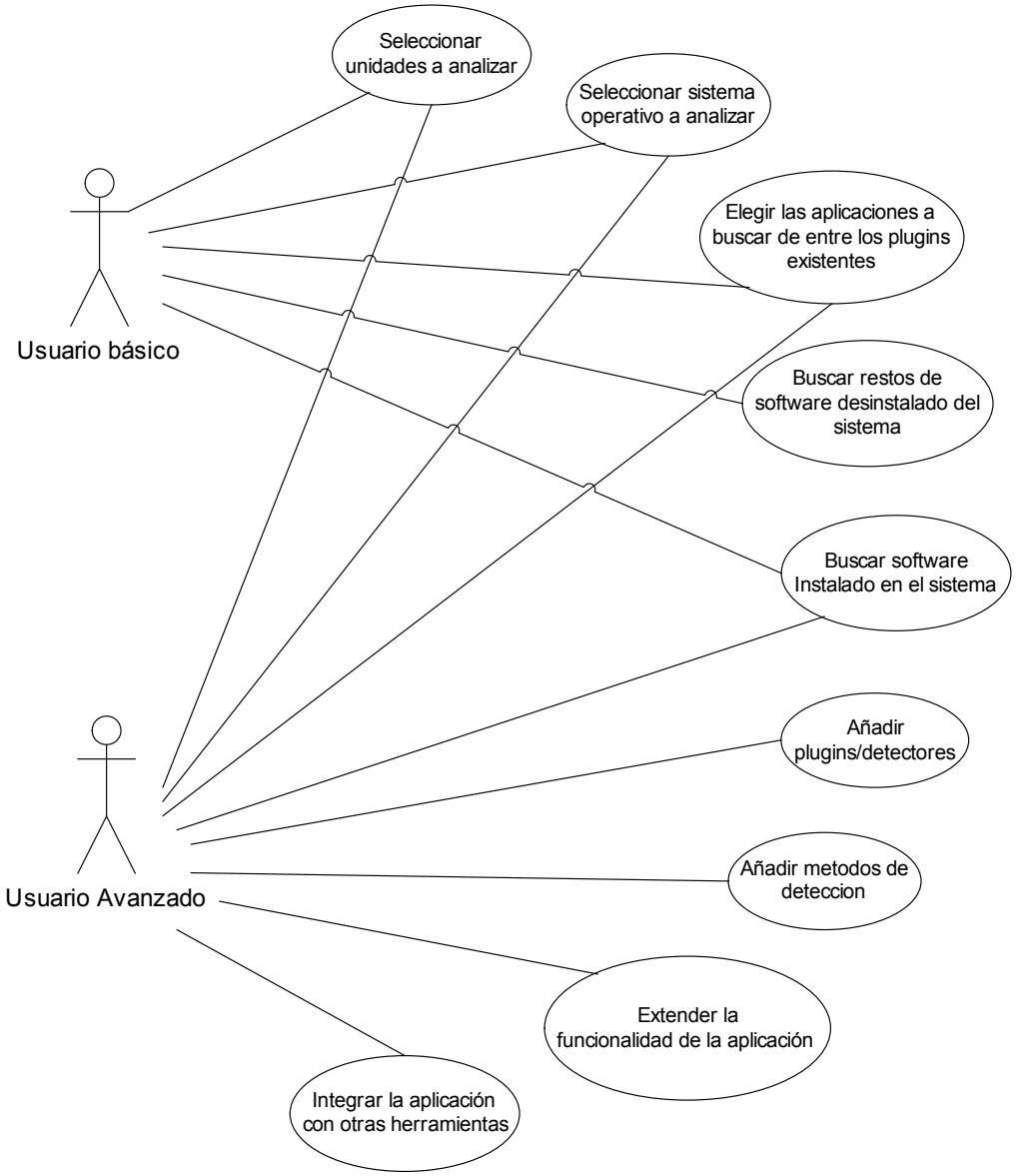


Gráfico 5. Diagrama de caso de uso. Funcionalidades.

5.5.1.2. Descripción textual de los Casos de Uso

Código	CU01
Nombre	**Análisis del sistema donde está ejecutando la aplicación**
Actores	Usuario básico.
Objetivo	Analizar el sistema en busca de programas instalados/desinstalados en el sistema.
Precondiciones	Ejecutar la aplicación en un sistema operativo con plugins implementados. Ejecutar con privilegios de administrador para poder acceder a todos los archivos del sistema.
Postcondiciones	
Escenario básico	1. El usuario elige una unidad de disco del equipo. 2. El usuario selecciona el conjunto de aplicaciones que querría detectar. 3. El usuario obtiene los resultados.

Código	CU03
Nombre	**Análisis de una unidad extraíble**
Actores	Usuario básico.
Objetivo	Analizar el sistema en busca de programas instalados/desinstalados en una unidad extraíble del sistema.
Precondiciones	La unidad extraíble debe haber sido usada en un sistema operativo para el que haya plugins implementados. Ejecutar con privilegios de administrador para poder acceder a todos los archivos del sistema.
Postcondiciones	
Escenario básico	El usuario elige una unidad de disco extraíble. El usuario selecciona el conjunto de aplicaciones que querría detectar El usuario obtiene los resultados.

Código	CU04
Nombre	**Búsqueda de un análisis anterior**
Actores	Usuario básico.
Objetivo	Obtener el resultado de un análisis anterior del sistema donde se ejecuta la aplicación.
Precondiciones	Se debe haber realizado un análisis previo con la misma herramienta sin haber borrado o modificado los archivos de la misma.
Postcondiciones	
Escenario básico	El usuario accede al directorio en el que se almacenan los informes en formato xml. El usuario obtiene la información relativa a anteriores escaneos.

5.5.2. Obtención de Requisitos

El propósito de la definición de requisitos es producir un conjunto de requisitos software tan completo y constante como sea posible a partir de la información facilitada por el usuario.

En este caso, la definición de los requisitos es responsabilidad del desarrollador. Los participantes en esta fase serán desarrollador y tutor del proyecto. Entre ellos tienen un concepto diferente del producto final, y estos conceptos se deben analizar, y sintetizar en una declaración completa y constante de requisitos para que todos tengan la misma visión global del sistema.

Los requisitos definen 'qué' debe hacer el producto, son una referencia para verificar el diseño y el producto. Los aspectos relativos al 'cómo' no deben incluirse, excepto aquellos que restringen el software.

A continuación se expone el catálogo de requisitos obtenido:

5.5.2.1. Requisitos Funcionales (f)

Identificador: **SR-F01**	Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador

Identificador: **SR-F01**		Tipo: Funcional
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta podrá ser ejecutada tanto en **Windows, Linux y Mac OS X.**	

Identificador: **SR-F02**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta deberá detectar si ciertos programas están instalados en el dispositivo a analizar.	

Identificador: **SR-F03**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta deberá detectar si ciertos programas estuvieron instalados en el dispositivo a analizar.	

Identificador: **SR-F04**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta debe ser capaz de realizar búsquedas en dispositivos externos (pen-drives de alta capacidad, discos duros, discos de máquinas virtuales, etc.) independientemente del sistema de ficheros o sistema operativo siempre que estén montados en el momento del análisis.	

Identificador: **SR-F05**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta debe ser capaz de realizar búsquedas en el ordenador donde se está ejecutando.	

Identificador: **SR-F06**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		

Identificador: **SR-F06**		Tipo: Funcional
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta debería permitir de forma sencilla la adición de nuevas funcionalidades.	

Identificador: **SR-F07**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta deberá detectar si ciertos programas estuvieron instalados en el dispositivo a analizar.	

Identificador: **SR-F08**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	

Identificador: **SR-F08**	Tipo: Funcional
Descripción:	Por cada aplicación encontrada en un análisis, la herramienta podría permitir la ejecución de un conjunto de acciones predefinidas para esa herramienta. Estas acciones se deberán poder añadir de forma sencilla.

Identificador: **SR-F09**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación deberá guardar registro sobre el resultado de los análisis efectuados.	

Identificador: **SR-F10**		Tipo: Funcional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación deberá permitir la consulta de informes anteriores.	

Identificador: **SR-F11**		Tipo: Funcional
Prioridad: ☐Alta ☐ Media ☑ Baja	Fuente: ☑ Tutor ☐ Desarrollador	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional		
Claridad: ☐ Alta ☑ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta deberá intentar extraer la información oculta tras determinados archivos escaneados y generar un informe con los resultados.	

5.5.2.2. Requisitos de rendimiento (r)

Identificador: **SR-R1**		Tipo: Rendimiento
Prioridad: ☐ Alta ☑ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Se implementará el código de manera que realice en primer lugar los procesos detectores que menos carga computacional tienen, para en caso de encontrar evidencias de los programas buscados no tenga que realizar todos los procesos.	

Identificador: **SR-R2**		Tipo: Rendimiento
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	

Identificador: **SR-R2**		Tipo: Rendimiento
Estabilidad:	Durante toda la vida del sistema	
Descripción:	El tiempo de análisis será dependiente del hardware utilizado para ejecutar el sistema, principalmente los aspectos referentes al acceso al disco.	

5.5.2.3. Requisitos de interfaz

Identificador: **SR-I1**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La herramienta tendrá que estar en español.	

Identificador: **SR-I2**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Alternativamente se facilitará la posibilidad de implementar facilidades para que la interfaz sea multi-idoma mediante archivos con una estructura estándar.	

Identificador: **SR-I3**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La interfaz debe ser intuitiva y sencilla de utilizar.	

Identificador: **SR-I4**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La interfaz mostrará las unidades de disco disponibles para realizar el análisis (Incluyendo unidades extraíbles).	

Identificador: **SR-I5**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La interfaz mostrará los plugins/detectores disponibles.	

Identificador: **SR-I6**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La interfaz permitirá marcar/desmarcar que plugins/detectores se utilizarán sobre un análisis.	

Identificador: **SR-I7**		Tipo: Interfaz
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Se realizará una interfaz alternativa en modo texto para ejecutar desde un terminal.	

Identificador: **SR-I8**		Tipo: Interfaz
Prioridad: ☐ Alta ☐ Media ☑ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Una vez encontrada una aplicación de esteganografía se ofrecerá la posibilidad de implementar un analizador en busca de archivos con información oculta.	

5.5.2.4. Requisitos operacionales (o)

Identificador: **SR-O1**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de reconocer las unidades actualmente conectadas a la maquina ejecutable.	

Identificador: **SR-O2**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de acceder en modo lectura a archivos de texto plano y buscar una determinada cadena en su interior.	

Identificador: **SR-O3**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de recorrer el sistema de ficheros en busca de un archivo concreto.	

Identificador: **SR-O4**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de buscar un archivo en los directorios predefinidos que se indiquen en los plugins/detectores.	

Identificador: **SR-O5**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de comprobar si existe una entrada concreto en una ruta concreta del registro de Windows.	

Identificador: **SR-O6**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de recorrer el registro de Windows en busca de una entrada concreta.	

Identificador: **SR-O7**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de buscar una cadena concreta en una ruta del registro de Windows.	

Identificador: **SR-O9**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de recorrer todo el registro de Windows en busca de una cadena concreta.	

Identificador: **SR-10**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de recorrer todo el sistema de ficheros en busca de un fichero que contenga una cadena concreta en el nombre.	

Identificador: **SR-11**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de buscar un fichero que contenga una cadena concreta en el nombre en una ruta determinada.	

Identificador: **SR-12**		Tipo: Operacional
Prioridad: ☐ Alta ☐ Media ☑ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de comprobar si un archivo concreto del sistema de ficheros alberga información oculta.	

Identificador: **SR-13**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de recorrer el sistema de ficheros en busca de aquellos que alberguen información oculta.	

Identificador: **SR-14**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☐ Esencial ☐ Deseable ☑ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe ser capaz de añadir los detectores a través de ficheros *legibles* en texto plano indicando que acciones son necesarias para encontrar cada aplicación.	

Identificador: **SR-15**		Tipo: Operacional
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☐ Esencial ☑ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación generará unos informes en texto plano en forma de logs que permitirán tanto la visualización del usuario inicial como la posibilidad de integrar los análisis con herramientas externas que parseen dichos logs.	

5.5.2.5. Requisitos de recursos (RE)

Identificador: **RE-R1**		Tipo: Recursos
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación no debería ocupar más de 5 Mb	

Identificador: **RE-R2**		Tipo: Recursos
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	

Identificador: **RE-R2**		Tipo: Recursos
Descripción:	Los plugins serán archivos de texto plano siempre inferiores a 1Mb.	

Identificador: **RE-R3**		Tipo: Recursos
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Los archivos de log no ocuparan en ningún caso más de 2 Mb	

Identificador: **RE–R4**		Tipo: Recursos
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación no deberá realizar modificaciones en el sistema operativo. Es decir, permitirá la ejecución portable.	

5.5.2.6. Requisitos de restricción (rs)

Identificador: **SR-RS1**		Tipo: Restricción
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Tecnología de desarrollo: La tecnología utilizada para el desarrollo de la aplicación será el lenguaje de alto nivel JAVA.	

Identificador: **SR-RS2**		Tipo: Restricción
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación tendrá que ser ejecutada en modo administrador, para poder analizar la totalidad del disco elegido.	

Identificador: **SR-RS3**		Tipo: Restricción
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	

Identificador: **SR-RS3**	Tipo: Restricción
Descripción:	No se podrá analizar una partición o unidad de disco con un formato que no esté soportada por el sistema anfitrión que ejecuta la aplicación.

Identificador: **SR-RS4**		Tipo: Restricción
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	No se podrá buscar un software sin incluir el plugin/detector del mismo.	

5.5.2.7. Requisitos de manejo de errores (e)

Identificador: **SR-ER1**		Tipo: Errores
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	**Mensajes de errores:** los mensajes de error han de ayudar al usuario a solucionar estos de forma clara y sencilla, a fin de que el usuario los entienda.	

5.5.2.8. Requisitos de documentación (d)

Identificador: **SR-D1**		Tipo: Documentación
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	**Toda la fase de desarrollo** deberá ser correctamente **documentada.**	

Identificador: **SR-D02**		Tipo: Documentación
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	**Todo el código fuente** debe ser **documentado.** Será obligatorio utilizar comentarios Javadoc para el código Java.	

5.5.2.9. Requisitos de portabilidad (p)

Identificador: **SR-P1**		Tipo: Portabilidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	La aplicación debe funcionar correctamente en cualquier sistema con maquina virtual JAVA. Se probará con Windows, Linux y Mac OS.	

5.5.2.10. Requisitos de calidad (c)

Identificador: **SR-C1**		Tipo: Calidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	**Presentación de una interfaz homogénea.** De forma que a lo largo de los diferentes menús de la aplicación se mantenga una misma estructura.	

Identificador: **SR-C2**		Tipo: Calidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	

Identificador: **SR-C2**		Tipo: Calidad
Descripción:	**Se distinguirán visualmente las opciones activas de las que no lo están**. El usuario sabrá en todo momento que puede y que no puede hacer.	

Identificador: **SR-C3**		Tipo: Calidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Utilización de un **fondo liso** para aumentar la **legibilidad**.	

Identificador: **SR-C4**		Tipo: Calidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	**Orden de las opciones de los menús**: Las diferentes secciones se han de ordenar **por grado** de utilización, según criterio de los diseñadores. Las más usadas tendrán un acceso más rápido y más "visible".	

Identificador: **SR-C5**		Tipo: Calidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	

Identificador: **SR-C5**		Tipo: Calidad
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Uso rápido de la aplicación. a la hora de realizar un análisis es conveniente no tener que atravesar excesivos menús. Atravesar muchos menús sería muy pesado y no facilitaría el recuerdo de los pasos realizados.	

Identificador: **SR-C6**		Tipo: Calidad
Prioridad: ☑ Alta ☐ Media ☐ Baja	Fuente: ☐ Tutor ☑ Desarrollador	
Necesidad: ☑ Esencial ☐ Deseable ☐ Opcional		
Claridad: ☑ Alta ☐ Media ☐ Baja	Verificabilidad: ☑ Alta ☐ Media ☐ Baja	
Estabilidad:	Durante toda la vida del sistema	
Descripción:	Se utilizarán mensajes de confirmación para operaciones sensibles. Estos mensajes serán escuetos y explicativos.	

5.5.3. Diseño del plan de Pruebas de aceptación

Como parte del proyecto se impondrán un conjunto de pruebas que determinarán cuando el software se considera apto para la implantación.

Es importante resaltar que por la envergadura del proyecto no es necesario realizar un documento específico para determinar el conjunto de pruebas a realizar. A continuación se muestra la tabla que especifica el plan de pruebas de aceptación.

Id	Descripción	Datos de entrada	Salida esperada
PA-01	Cargar un conjunto de plugins y aplicarlos a un análisis.	Ficheros con datos de plugins.	Informe de resultados con las técnicas ejecutadas.
PA-02	Realizar una búsqueda de un fichero identificado mediante una expresión regular en un directorio concreto sabiendo previamente que existe.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados con el archivo que se había comprobado previamente que existía y se encontraba en la carpeta definida.
PA-03	Realizar una búsqueda de un fichero identificado mediante una expresión regular en todo el disco. Se ubica el fichero en determinadas carpetas distintas para comprobar que lo encuentra en todas las ocasiones.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados con el archivo que se había ubicado en las distintas ubicaciones, señalando estas en los resultados.
PA-04	Realizar una búsqueda en el interior de un fichero concreto en busca de una cadena.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando el fichero y la cadena localizada.
PA-05	Realizar una búsqueda en el interior de un directorio concreto recorriendo todos los archivos de su interior en busca de una cadena. Ubicando previamente la cadena en varios archivos dentro del directorio y fuera.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando todos los ficheros en los que se ha localizado la cadena dentro del directorio. No deben aparecer los archivos externos a la carpeta.
PA-06	Realizar una búsqueda recorriendo todo el disco buscando en todos los ficheros una cadena concreta.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando todos los ficheros en los que se ha localizado la cadena en el disco.
PA-07	Realizar una búsqueda de una cadena en una ubicación concreta del registro de Windows. Asegurarse que esta cadena existe.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando la cadena localizada.
PA-08	Realizar una búsqueda de una cadena en todo el registro de Windows. Asegurarse que esta cadena existe y añadirla en varias ubicaciones del registro.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando las distintas ubicaciones donde se localizó la cadena.
PA-09	Realizar una búsqueda de un valor en una ubicación concreta del registro de Windows. Asegurarse que esta	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando la cadena localizada.

	cadena existe.		
PA-10	Realizar una búsqueda de un valor en todo el registro de Windows. Asegurarse que esta cadena existe y añadirla en varias ubicaciones del registro.	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados indicando las distintas ubicaciones donde se localizo la cadena.
PA-11	Ejecutar la aplicación en distintos sistemas operativos y realizar el mismo análisis simple de la prueba **PA-02.**	Fichero con la técnica de búsqueda descrita definida en su sintaxis.	Informe de resultados con el resultado de la búsqueda.
PA-12	Realizar un análisis y seleccionar firmar el fichero con un archivo P12 previamente generado.	Fichero con la técnica de búsqueda descrita definida en su sintaxis. Fichero con certificado. Clave privada para firmar con el fichero adjunto.	Informe de resultados indicando las distintas ubicaciones donde se localizó la cadena incluyendo una la firma del certificado utilizado.

Tabla 23. Pruebas de aceptación

5.5.4. Análisis de los Casos de Uso.

En este apartado se analizarán varios casos de uso mediante diagramas de secuencia a fin de definir de una forma lo más detallada posible cuales serán los pasos que realizará la aplicación para ofrecer las distintas funcionalidades.

5.5.4.1. Análisis del sistema donde está ejecutando la aplicación

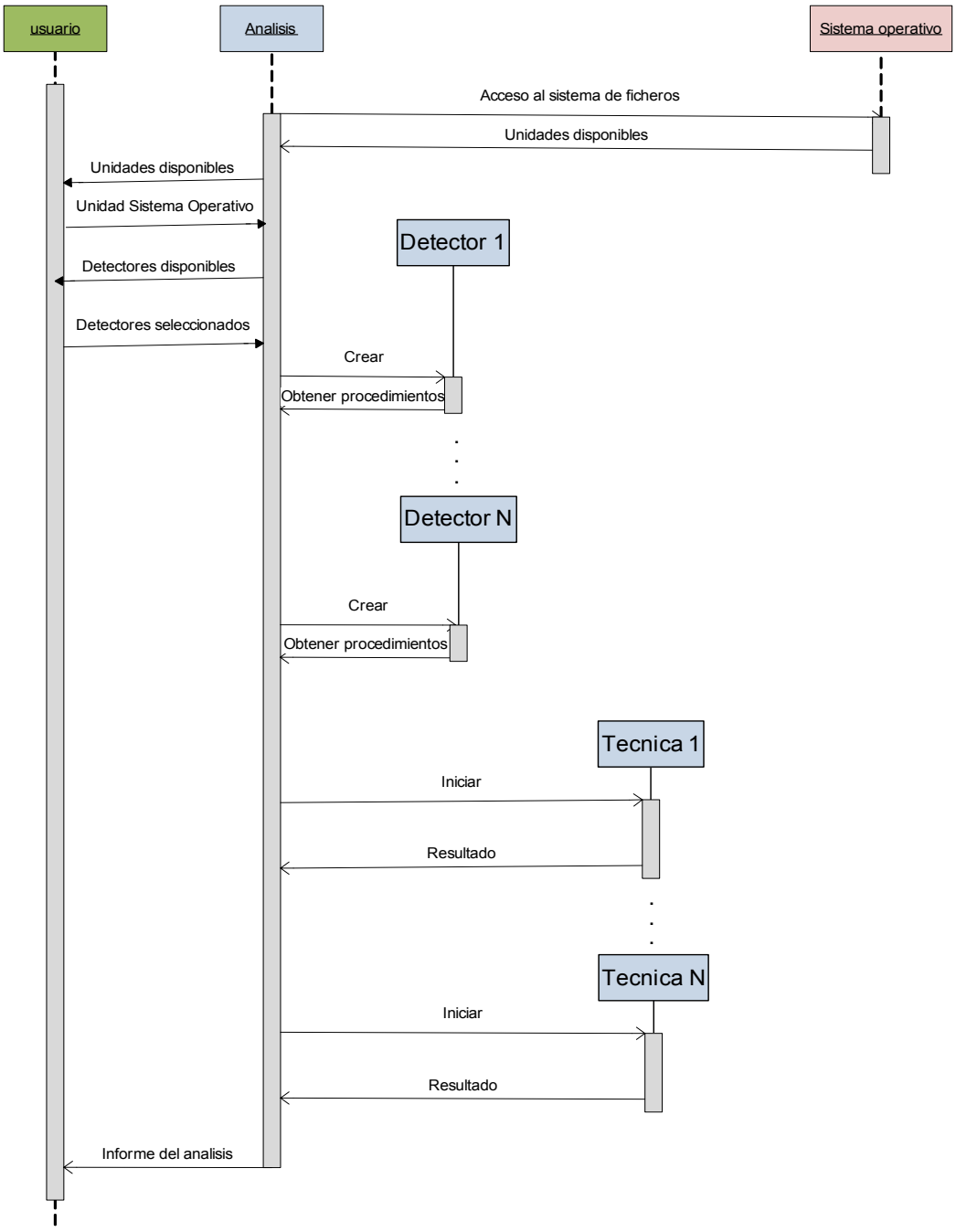


Gráfico 6. Análisis del sistema donde se ejecuta la aplicación

El sistema ofrece al usuario el conjunto de unidades disponibles en el sistema y el usuario elige el disco duro que contiene el sistema operativo actual. También selecciona el conjunto de detectores que desea utilizar en el escaneo. El sistema recoge las acciones a llevar a cabo por cada uno de los detectores seleccionados y llama secuencialmente a las técnicas

necesarias para ejecutar todas estas acciones. Finalmente la aplicación genera un informe para el usuario con el resultado del análisis.

5.5.4.2. Análisis de una unidad extraíble

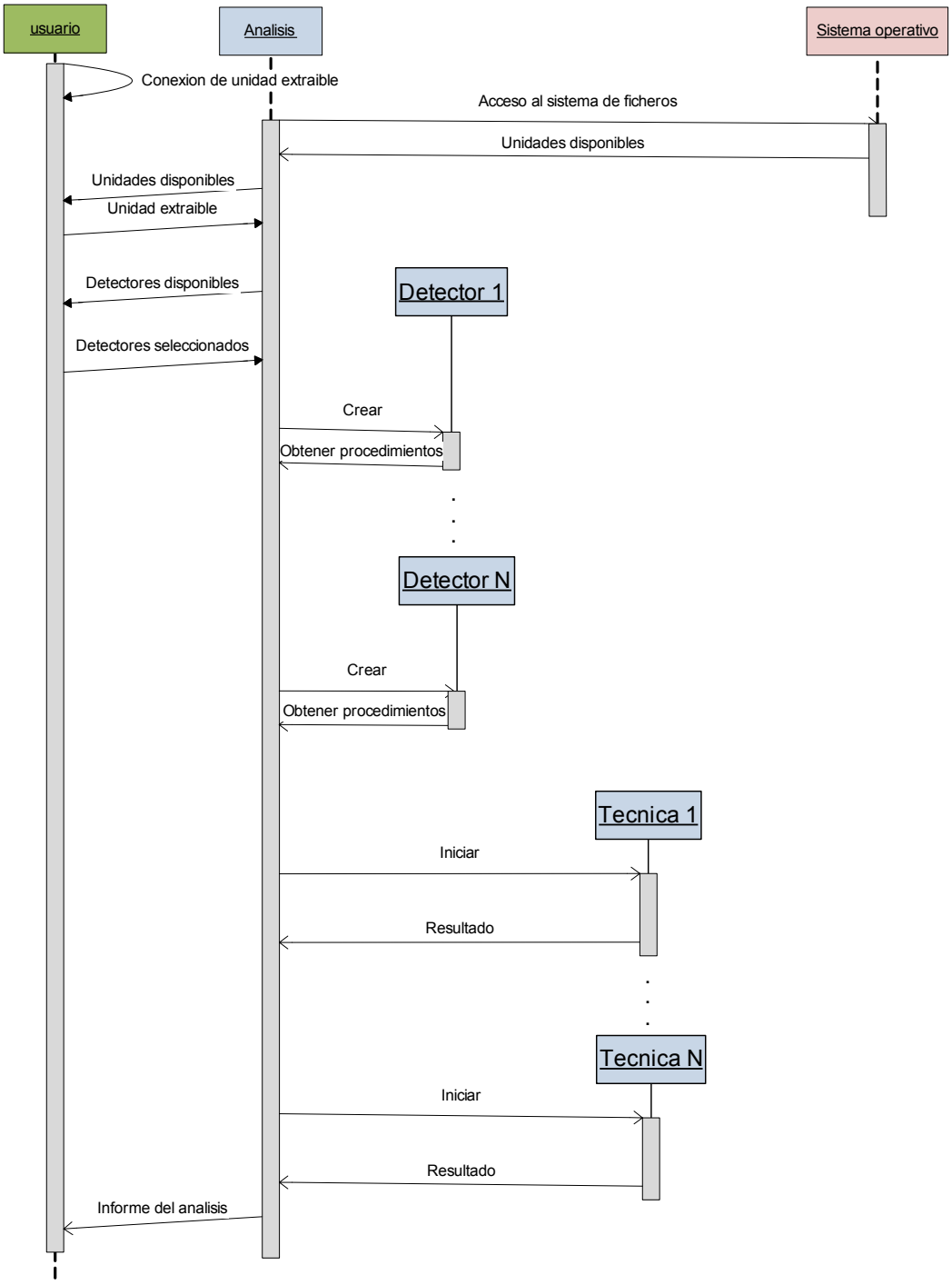


Gráfico 7. Diagrama del análisis de una unidad extraíble

El sistema ofrece al usuario el conjunto de unidades disponibles en el sistema y el usuario elige la unidad extraíble. También selecciona el conjunto de detectores que desea utilizar en el escaneo. El sistema recoge las acciones a llevar a cabo por cada uno de los detectores seleccionados y llama secuencialmente a las técnicas necesarias para ejecutar todas estas acciones. Finalmente la aplicación genera un informe para el usuario con el resultado del análisis.

5.5.4.3. Búsqueda de un análisis anterior

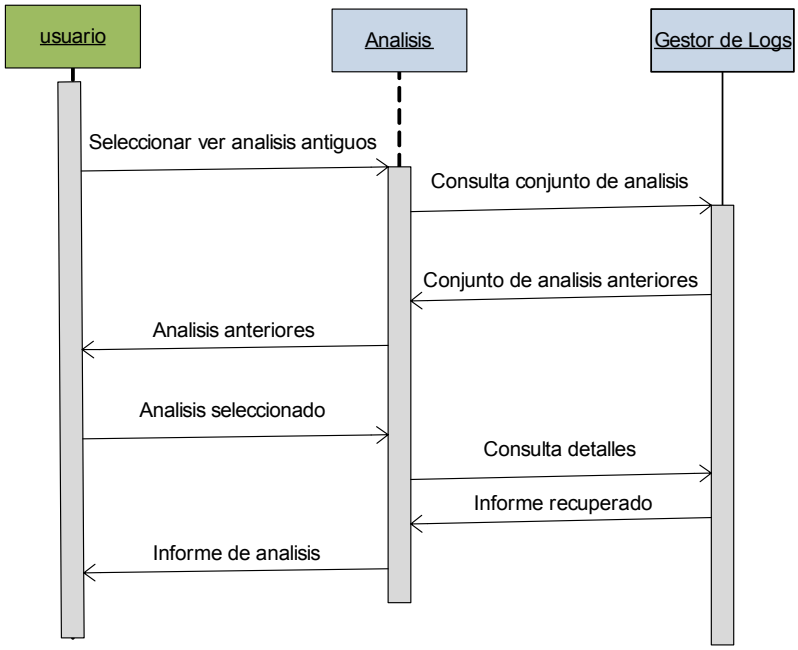


Gráfico 8. Diagrama de búsqueda de un análisis anterior

El usuario elige ver análisis antiguos realizados. Se accede a los logs almacenados de análisis anteriores identificados con la fecha y hora del análisis. El usuario selecciona uno y el sistema le proporciona el informe almacenado referente al análisis.

6. Diseño

6.1. Propósito

En este apartado se especifica el diseño que se seguirá en el desarrollo de la aplicación. Como ya se especificó anteriormente, el diseño estará basado en la arquitectura MVC. El principal objetivo de este apartado es establecer las bases sobre las que se va a construir el sistema.

6.2. Visión general del sistema

El sistema tiene como objetivo realizar análisis forense de aplicaciones instaladas/desinstaladas mediante unos sencillos pasos.

El diseño de la aplicación está marcado por la arquitectura MVC (Modelo-Vista-Controlador) con el objetivo de separar estos tres componentes, proporcionando mayor independencia y facilitando tanto el mantenimiento como los posibles futuros cambios en la aplicación.

En general no será necesario almacenar grandes cantidades de datos en disco, únicamente el resultado de los distintos análisis que se realicen durante la vida del software, para lo cual se optará por utilizar archivos de log en texto plano.

Los principales conceptos que utilizaremos para realizar las búsquedas y sus acciones correspondientes serán:

- **Técnicas** - Métodos de búsqueda implementados y que pueden ser utilizados para encontrar determinados restos. Por ejemplo, una técnica concreta busca un archivo en todo el disco en función de su nombre.
- **Acciones** - Una vez encontrado un programa concreto se pueden definir una serie de acciones pre-implementadas a partir de unos parámetros definidos.
- **Plugins** - Conjunto de ***Técnicas*** y ***Acciones*** que determinan como encontrar un programa y que acciones relacionadas se pueden ejecutar automáticamente.

6.3. Contexto del sistema

La aplicación se ejecutará en una máquina con máquina virtual de Java sin requerir grandes capacidades especiales de hardware. En general debería funcionar bajo cualquier sistema operativo compatible con java, sin embargo solo se probará con los 3 principales sistemas operativos en el mercado de consumo.

6.3.1. Ficheros relacionados

Tal como se ha indicado en anteriores apartados la aplicación admitirá datos desde ficheros y exportará información en forma de informes y logs. Un paso importante es tener bien definido el formato de esos datos de entrada y salida para el correcto funcionamiento de la aplicación. Durante este apartado se indicarán cuales son estos datos y se detallará su estructura.

6.3.1.1. Plugins

Un plugin es un conjunto de definiciones de comprobaciones que se deben realizar para encontrar los residuos de un determinado programa. Bajo esta premisa se ha desarrollado un formato de fichero que permita una gran flexibilidad en la representación de los datos y a la par permita designar una estructura por lo que se ha decidido utilizar ficheros XML en colaboración con archivos de Definición de datos DTD.

La organización de la información dentro de los plugins se define tal como sigue el siguiente esquema:

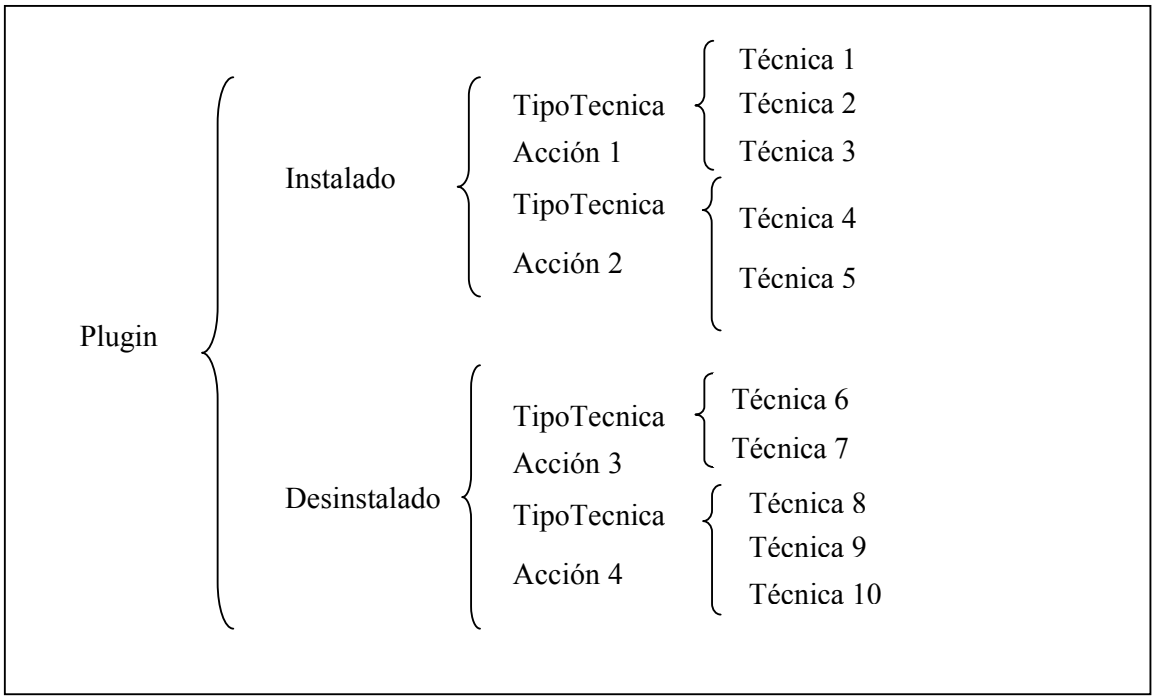


Gráfico 9. Esquema de un Plugin

Como se puede ver, se divide la información del plugin en dos grandes grupos. Por un lado el conjunto de técnicas y acciones que se utilizan/lanzan para comprobar si un software está instalado y por otro lado si un software estuvo instalado y ha sido desinstalado del sistema.

Dentro de estos grandes grupos se encuentran las ***Técnicas*** y ***Acciones*** correspondientes. Existen distintos tipos de técnica dependiendo de su naturaleza. En principio

se han definido 3 tipos de ***Técnica***: 1) aquellas que buscan ficheros en el disco 2) aquellas que buscan cadenas dentro de ficheros 3) aquellas que buscan en el registro de Windows.

Para la definición de los distintos plugins hay disponibles una serie de técnicas implementadas que referencian distintos tipos de búsquedas en el sistema. A continuación se explicaran las características de cada una:

- **Técnicas relacionadas con búsquedas en disco**
  - **barchivo:** Busca un determinado archivo en un determinado directorio. Para ello hay que definir los parámetros **nombre** y **directorio.** Tiene un parámetro opcional que permite comprobar si el archivo encontrado además concuerda con el hash del fichero original. Este hash puede estar generado en MD5 o en SHA indistíntamente.
  - **barchivo_pname**: Busca un determinado archivo en todo el disco. Para ello hay que definir el parámetro **nombre** que hace referencia al nombre del archivo buscado. Tiene un parámetro opcional que permite comprobar si el archivo encontrado además concuerda con el hash del fichero original. Este hash puede estar generado en MD5 o en SHA-1 indistíntamente.

- **Técnicas relacionadas con búsquedas en el registro**
  - **breg_ub:** Técnica que permite buscar un registro concreto en una ubicación. El nombre y valor dentro de la clave del registro es opcional, de manera que si no se especifican estos valores solo comprobará si existe la clave definida.
  - **breg_all:** Técnica que busca un valor en todo el registro. Recorre todos los atributos de todas las claves para comprobar si existen.
  - **breg_all_sec:** Técnica que busca un registro concreto en una determinada sección del registro de forma recursiva en sus subsecciones. Al igual que la técnica **breg_ub**, el atributo y su valor es opcional, por lo que si no se especifican solo comprueba que exista la clave.

- **Técnicas relacionadas con búsquedas dentro de logs**
  - **blog:** Técnica que busca una cadena dentro de un fichero concreto. Es necesario indicar los parámetros cadena y fichero, con la cadena a buscar y la ruta del fichero donde se buscará respectivamente.
  - **blog_pd:** Técnica que busca una cadena en los archivos de todo un directorio de forma recursiva. Se especifica la cadena y el directorio y recorre todos los archivos del mismo para determinar en cuales aparece.
  - **blog_all:** Técnica que busca una cadena en los archivos de todo el disco. Se especifica la cadena y recorre todos los archivos del disco para determinar en cuales aparece.

La estructura de estos ficheros viene definida por un fichero DTD que se expone a continuación.

```
<!ELEMENT plugin (instalado?,desinstalado?)>
<!ATTLIST plugin
      name CDATA #REQUIRED
      version CDATA #IMPLIED
      so CDATA #IMPLIED>
      <!ELEMENT instalado (tecnicas,acciones?)>
            <!ELEMENT tecnicas (tecdisco?,tecreg?,teclog?)>
                  <!ELEMENT tecdisco (barchivo*,barchivo_pname*)>
                        <!ELEMENT barchivo EMPTY><!-- Buscar un archivo en un directorio -->
                        <!ATTLIST barchivo
                              nombre CDATA #REQUIRED
                              directorio CDATA #REQUIRED
                              hash CDATA #IMPLIED>
                        <!ELEMENT barchivo_pname EMPTY><!-- Buscar un archivo en todo el disco -->
                        <!ATTLIST barchivo_pname
                              nombre CDATA #REQUIRED
                              hash CDATA #IMPLIED>
                  <!ELEMENT tecreg (breg_ub*,breg_all*,breg_all_sec*)>
                        <!ELEMENT breg_ub EMPTY><!-- Buscar un registro en una ubicacion-->
                              <!ATTLIST breg_ub
                                    registro CDATA #REQUIRED
                                    valor CDATA #IMPLIED
                                    atributo CDATA #IMPLIED>
                        <!ELEMENT breg_all EMPTY> <!-- Buscar un valor en todas partes.Se recomienda no usar-->
                              <!ATTLIST breg_all
                                    valor CDATA #REQUIRED>
                        <!ELEMENT breg_all_sec EMPTY> <!-- Buscar un valor en una seccion del registro-->
                              <!ATTLIST breg_all_sec
                                    valor CDATA #REQUIRED
                                    seccion CDATA #REQUIRED>

                        <!ELEMENT teclog (blog*,blog_pd*,blog_all*)>
                              <!ELEMENT blog EMPTY><!-- Buscar una cadena en un fichero concreto-->
                              <!ATTLIST blog
                              cadena CDATA #REQUIRED
                              fichero CDATA #REQUIRED>
                              <!ELEMENT blog_pd EMPTY><!-- Buscar una cadena en los logs de un directorio-->
                              <!ATTLIST blog_pd
                              cadena CDATA #REQUIRED
                              directorio CDATA #REQUIRED>
                              <!ELEMENT blog_all EMPTY><!-- Buscar una cadena en los archivos de todo el disco-->
                              <!ATTLIST blog_all
                              cadena CDATA #REQUIRED>

            <!ELEMENT acciones (ejprogext*,bimg_encrp*)>
                  <!ELEMENT ejprogext EMPTY><!-- Ejecuta un programa externo con parametros -->
                  <!ATTLIST ejprogext
                  comando CDATA #REQUIRED
                  parametros CDATA #REQUIRED>
            <!ELEMENT bimg_encrp EMPTY> <!-- Busca imagenes encriptadas de un tipo o de todos -->
                  <!ATTLIST bimg_encrp
                  tipo CDATA #REQUIRED>
<!ELEMENT desinstalado (tecnicas?,acciones?)>
```

Código 1. DTD de los plugins en XML

Y a continuación se muestra un ejemplo de un plugin implementado para detectar el software Camouflagge:

```
<?xml version="1.0" encoding="UTF-8" ?>
  <!DOCTYPE plugin (View Source for full doctype...)>
<plugin name="Camouflagge" so="Windows" version="1.2.1">
<instalado>
<tecnicas>
  <tecdisco>
<barchivo directorio="\Archivos de programa\Camouflage" nombre="Camouflage.exe" />
<barchivo directorio="\Archivos de programa\Camouflage" nombre="CamShell.dll" />
<barchivo_pname nombre="Camouflage Readme.lnk" />
<barchivo_pexp directorio="\WINDOWS\Prefetch" exp="*CAMOUFLAGE.EXE*" />
  </tecdisco>
<tecreg>
<breg_ub registro="{29557489-990B-11D4-9413-004095490AD4}"
  ubicacion="HKEY_CLASSES_ROOT\*\shellex\ContextMenuHandlers\Camouflage" />
<breg_all registro="CamouflageShell.ShellExt" />
  </tecreg>
<teclog>
<blog cadena="c:\WINDOWS\system32\config\software.LOG" fichero="camouflage" />
<blog_all cadena="camouflage.exe" />
  </teclog>
  </tecnicas>
<acciones>
<ejprogext comando="Uninstall Camouflage.exe" parametros="-t *" />
<bimg_encrp tipo="camouflage" />
  </acciones>
  </instalado>
<desinstalado>
<tecnicas>
<tecdisco>
<barchivo directorio="\Documents and Settings\Administrador\Configuración
  local\Temp" nombre="SETUP.INI" hash="c259c3462d02d904beb67177583fe42b" />
<barchivo_pname nombre="camouflage.EXE" hash="c62b050117c2cba3518e5a734fedef1f" />
<barchivo_pexp directorio="\WINDOWS\Prefetch" exp="*CAMOUFLAJE.EXE*" />
  </tecdisco>
<tecreg>
<breg_ub registro="{29557489-990B-11D4-9413-004095490AD4}"
  ubicacion="HKEY_CURRENT_USER\Software\Camouflage" />
<breg_all registro="{29557489-990B-11D4-9413-004095490AD4}" />
  </tecreg>
  </tecnicas>
<acciones>
<bimg_encrp tipo="camouflage" />
  </acciones>
  </desinstalado>
  </plugin>
```

6.3.1.2. Informes

Los informes se definen como el conjunto de resultados obtenidos de los análisis generados con la herramienta. Existirán dos tipos de informe:

- **Informe PDF**

El informe PDF tendrá una estructura legible por el usuario con ciertos elementos de formato de texto como encabezados, títulos, formato de texto (tamaño, negrita, etc) y algunos separadores.

- **Informe XML**

Los informes XML se generarán automáticamente en una carpeta predefinida como seguimiento de los análisis realizados en el equipo. Los informes en XML también vienen definidos por medio de un archivo de definición de datos XML llamado "informe.dtd". Dicho fichero se detalla a continuación:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT informe (plugin*)>
<!ATTLIST informe
dia CDATA #REQUIRED
mes CDATA #REQUIRED
año CDATA #REQUIRED
hora CDATA #REQUIRED
Unidad CDATA #IMPLIED>
<!ELEMENT plugin (tecnica*)>
<!ATTLIST plugin
fichero CDATA #REQUIRED>
<!ELEMENT tecnica (detectado*)>
<!ATTLIST tecnica
tipo CDATA #REQUIRED
tecnica CDATA #REQUIRED>
<!ELEMENT detectado (acciones*)>
<!ATTLIST detectado
tipo CDATA #REQUIRED
plugin CDATA #REQUIRED
elemento CDATA #REQUIRED>
<!ELEMENT acciones (detectado*))>
<!ATTLIST acciones
tipo CDATA #REQUIRED>
```

Código 2. DTD de los informes XML

- **Ficheros de log**

El fichero de log es un fichero que almacena en distintas líneas los resultados positivos de los escaneos. Cuando se realiza un escaneo en busca de restos de una instalación de un programa X, si lo encuentra, aparecerá una línea como la siguiente en el archivo de log:

Fecha Hora Located results with plugin X

Ejemplo:

Thu Jul 09 13:47:33 CEST 2009 Located results with plugin Camouflagge

De esta manera se genera un histórico de programas encontrados con su momento justo de ejecución de análisis. Este log puede ser integrado con aplicaciones externas que recojan estos resultados para correlarlos.

6.4. Diseño preliminar del sistema

La arquitectura de la aplicación se divide en tres componentes fundamentales de la aplicación: control de datos, procesamiento y vista.

El componente vista, representa el medio por el cual el usuario interactúa y recibe información de la aplicación. La vista envía las acciones del usuario al controlador. Para la vista se utilizará la librería JFC Swing.

El controlador define la lógica del sistema. Controla las peticiones del usuario, las interpreta y las convierte para ser interpretadas por el modelo.

El modelo representa como estarán formados los datos. En nuestro caso ofrece las estructuras de información que manejara nuestra aplicación, tales como ***Plugins***, ***Técnicas***, ***Acciones***, ***Informes***, ***etc***.

Para el almacenamiento físico de los datos se utilizarán ficheros en texto plano accesibles mediante el paquete *.io de java.

6.4.1. Componentes del sistema

El diseño estará constituido por 3 componentes principales, tal como lo establece el patrón de arquitectura MVC.

El diagrama de componentes será el siguiente:

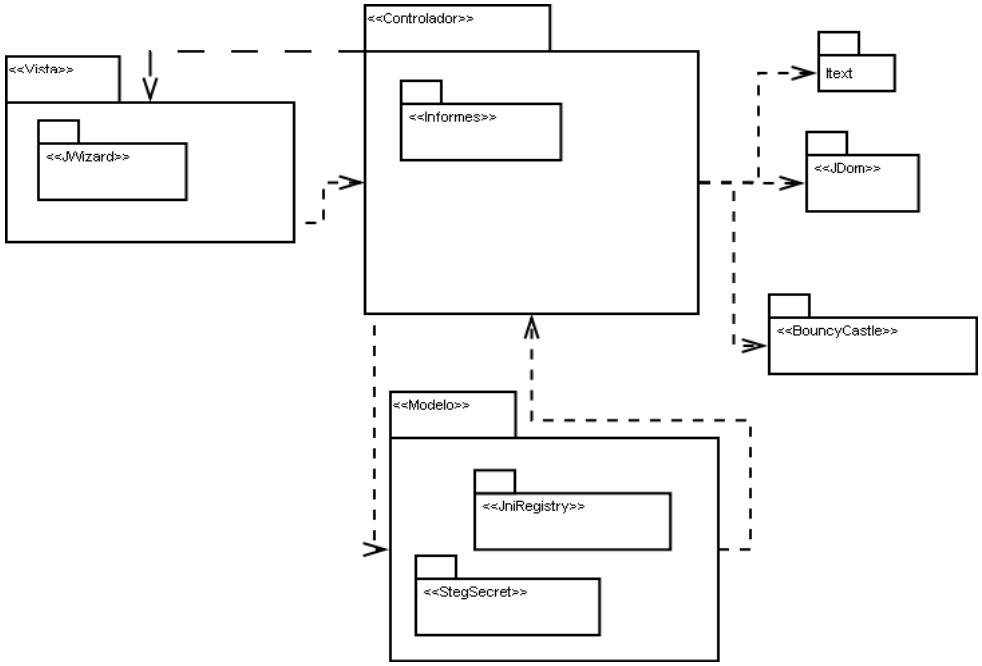


Gráfico 10. Componentes del sistema

6.4.2. Componentes

Nombre	Vista
Tipo	Subsistema.
Propósito	El componente vista, representa el contenido de un modelo. Específica cual es la representación de los datos y los actualiza cuando se realizan cambios. La vista envía las acciones del usuario al controlador.
Dependencias	Es utilizado por los subsistemas Controlador y Modelo.
Interfaces	En este componente proporciona todo lo relacionado con la visualización de la información.

Nombre	Controlador
Tipo	Subsistema.
Propósito	Recibe las peticiones del cliente llevando después a cabo las acciones correspondientes.
Dependencias	Este componente controla las peticiones de la vista y controla el flujo del modelo de manera oportuna.
Interfaces	Determinar en todo momento el flujo de la aplicación.

Nombre	Modelo
Tipo	Subsistema.
Propósito	El modelo representa los datos y la lógica de negocio u operaciones que permiten el acceso y modificación de los datos. Cuando son modificados el modelo se lo notifica a las vistas. También proporciona al controlador la capacidad para acceder a la funcionalidad de la aplicación.
Dependencias	Notifica a la vista para que se actualice cuando se modifica el modelo.
Interfaces	Ofrece todas las interfaces de intercambio de datos.

6.4.3. Componente Vista

Este componente se encarga de toda la representación visual de la aplicación. A continuación se muestra un boceto de las pantallas que conformarán la interfaz gráfica. Este diseño no es definitivo, simplemente ofrece una idea del funcionamiento de la misma mediante prototipos de las distintas pantallas.

6.4.3.1. Pantalla inicial

Gráfico 11. Pantalla inicial

En la pantalla inicial se muestran las opciones disponibles: analizar todo el sistema, analizar una unidad concreta y recuperar los informes de los anteriores análisis.

6.4.3.2. Análisis de unidades concretas

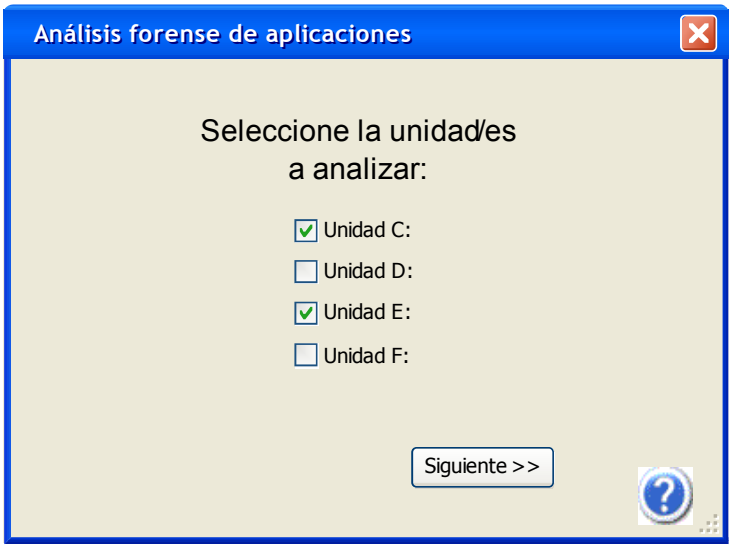


Gráfico 12. Análisis de unidades concretas

En la pantalla se muestran las unidades disponibles y se puede seleccionar una o varias de ellas para realizar el análisis.

6.4.3.3. Selección de detectores:

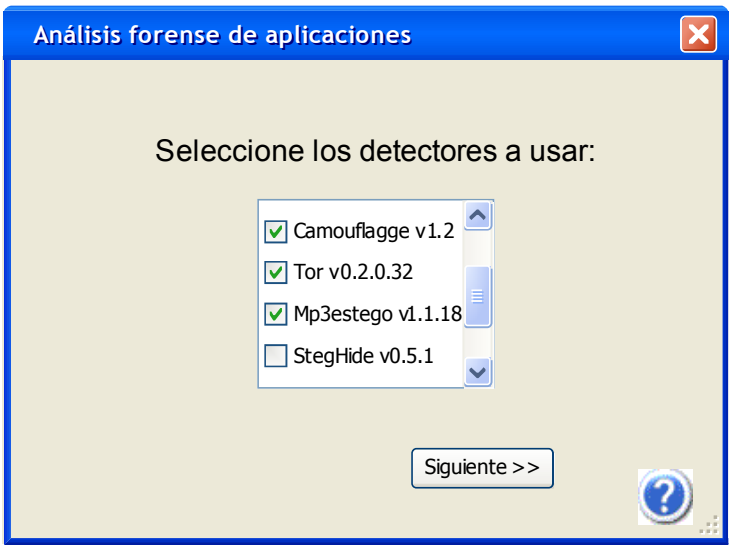


Gráfico 13. Selección de detectores

Se muestran los detectores disponibles con posibilidad de marcarlos.

6.4.3.4. Proceso de análisis:

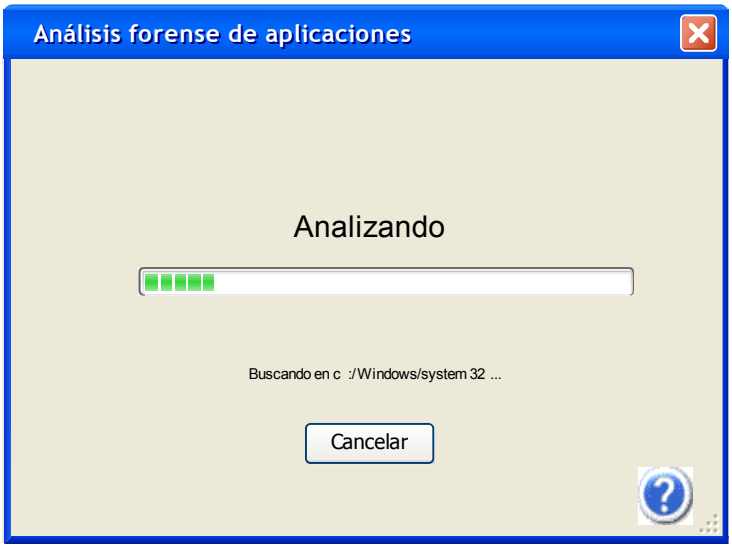


Gráfico 14. Proceso de análisis

Se muestra una barra de progreso que indica, aproximadamente el progreso del análisis. También se actualiza un campo de texto indicando las acciones que está llevando a cabo.

6.4.3.5. Informe del análisis:

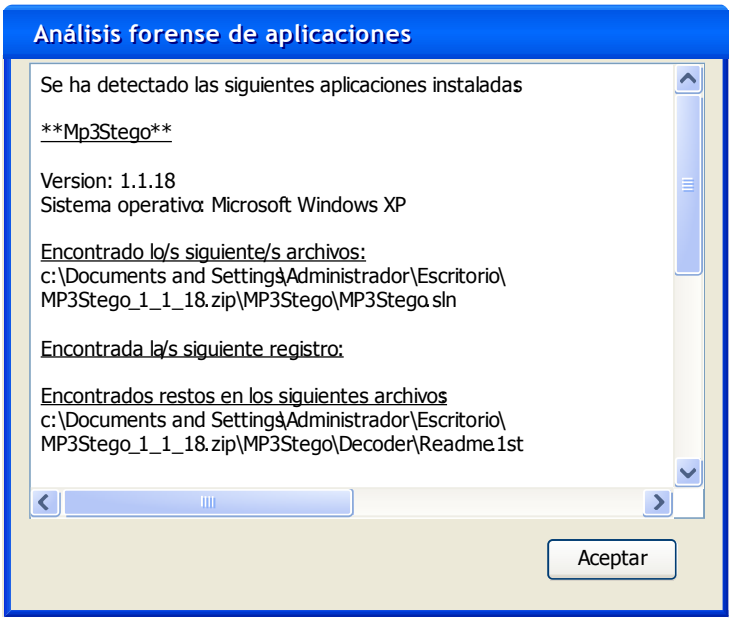


Gráfico 15. Informe del análisis

Se muestran los resultados asociados al análisis realizado.

6.4.3.6. Ayuda:

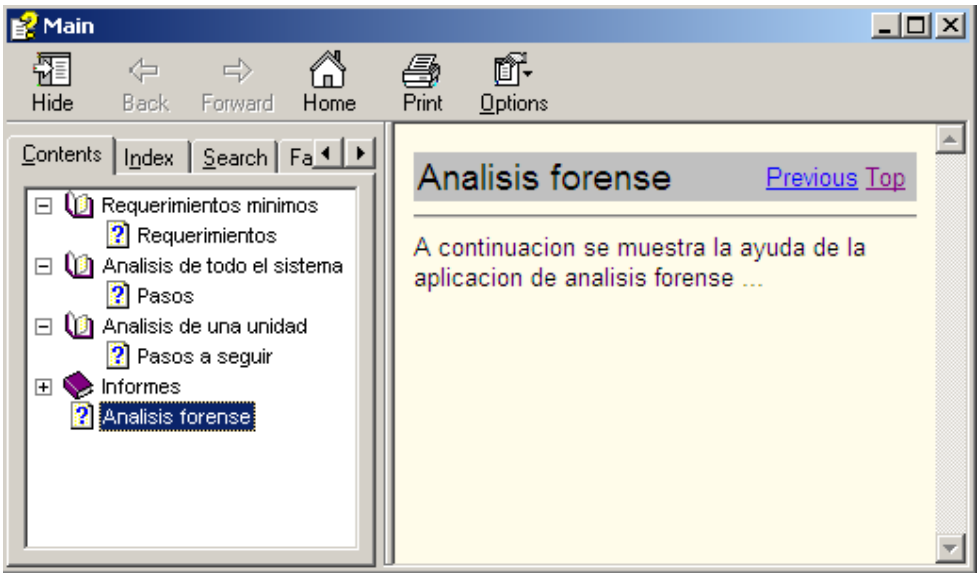


Gráfico 16. Pantalla de ayuda

Se mostrará la ayuda en un archivo en formato hlp. Se estudia usar mejor html por su compatibilidad con más sistemas operativos.

6.4.4. Diagrama de actividad

El diagrama de actividad muestra las distintas acciones que puede ejecutar un usuario a partir de la pantalla de inicio y por los estados que va pasando durante la ejecución de la aplicación.

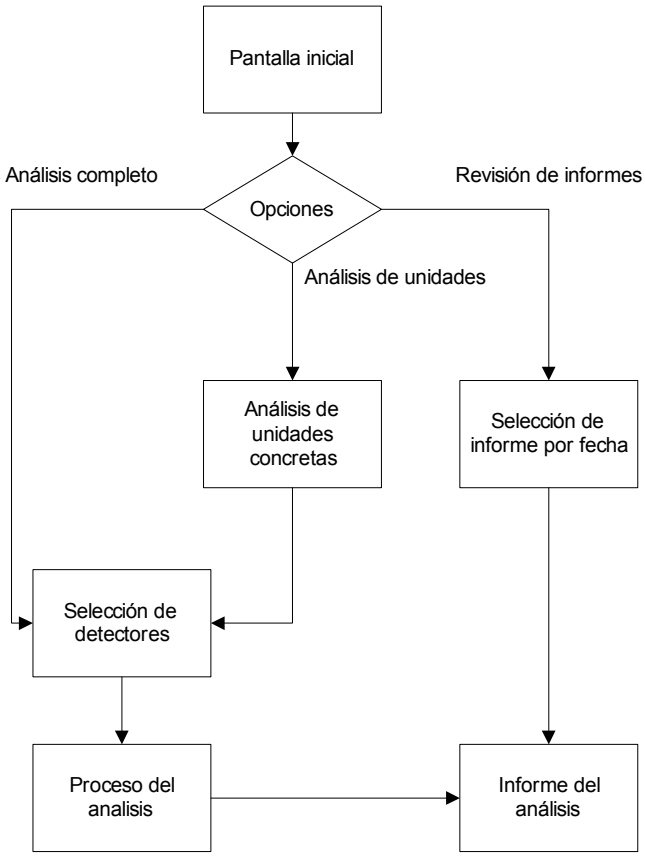


Gráfico 17. Diagrama de actividad

6.5. Diseño Detallado

Tras sentar las bases de nuestro diseño, se mostrará de una forma más detallada el diseño, ya teniendo en cuenta que la aplicación se implementará sobre JAVA y especificando en mayor medida sus componentes.

Se ofrecerán también, diagramas UML más detallados que podrán dar una idea en profundidad de la estructura de la aplicación.

Desde el inicio del proyecto se ha indicado que el diseño sería realizado bajo el patrón MVC (Modelo-Vista-Controlador). La principal ventaja de este modelo radica en la independencia de sus componentes. Esto nos permite diseccionar el diseño dividiéndolo de forma muy sencilla en los 3 componentes básicos.

6.5.1. Diseño de clases UML Completo

En el siguiente diagrama se presenta la estructura completa de nuestra aplicación dividida en los tres citados componentes con las relaciones que se establecen entre ellos.

En los siguientes apartados se definirá la estructura específica de cada componente, pero mediante este diagrama general se puede ofrecer una idea general del funcionamiento de la estructura y funcionamiento de la aplicación.

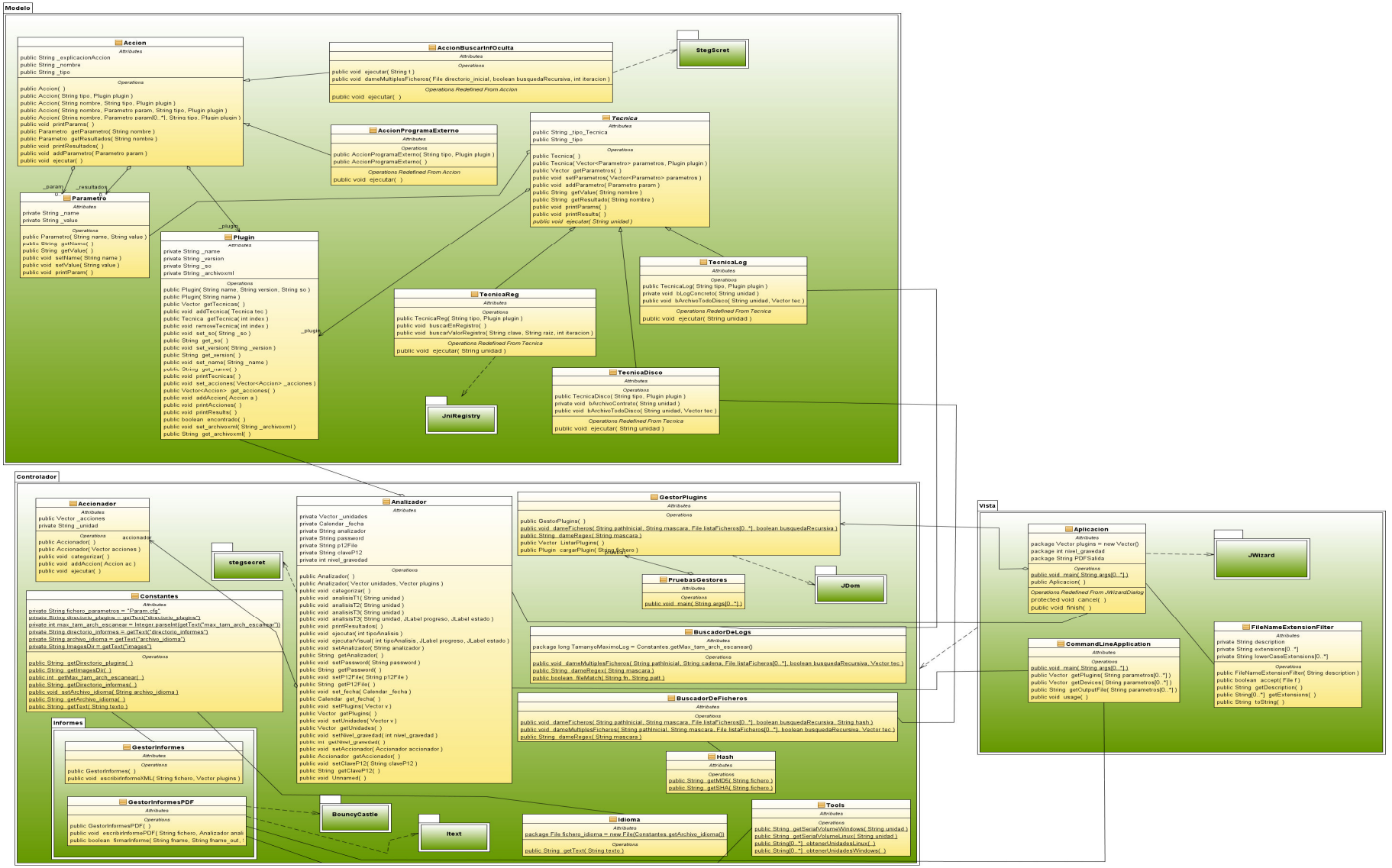


Gráfico 18. Diagrama de clases de la aplicación

6.5.2. Componente Modelo

El componente Modelo es la representación específica de la información con la que el sistema opera. Existen distintas clases para definir los datos de los distintos elementos de la aplicación, tales como técnicas de análisis, plugins, acciones, etc.

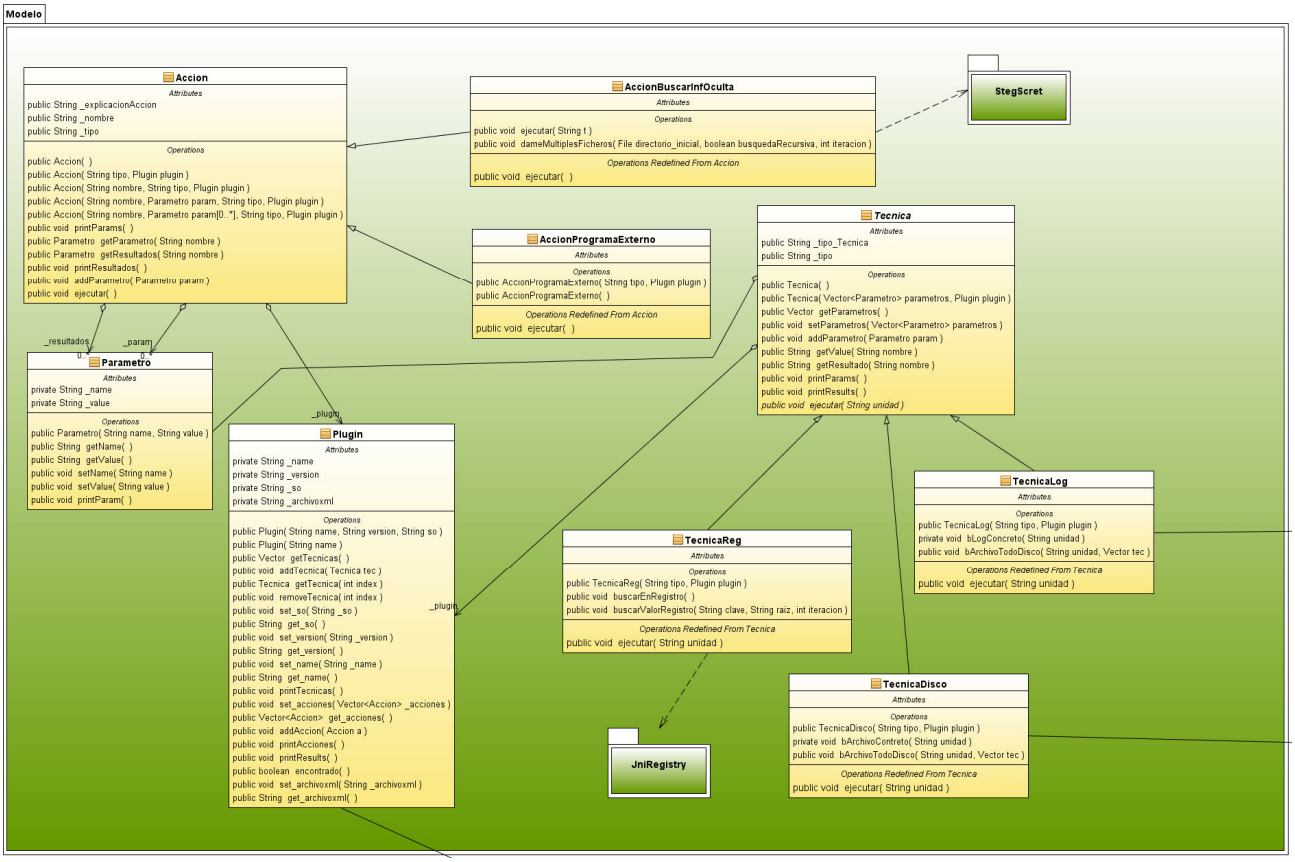


Gráfico 19. Diagrama UML de componente Modelo

La información de carácter genérico de los análisis, es decir, los detalles sobre las búsquedas que deben realizar los análisis y los resultados obtenidos se almacenan en objetos de una clase llamada **Parametro**. Cada parámetro tiene un nombre y un valor asociado que determinan el contenido del mismo. Por otro lado, los análisis que se van a realizar definen sus características (técnicas de búsqueda, acciones y resultados) en vectores de tamaño indeterminado, permitiendo gran flexibilidad a la hora de generar análisis distintos.

Los elementos más importantes del componente **Modelo** son:

- **Tecnica**: Una técnica representa una forma de búsqueda. Existen técnicas concretas que buscan residuos en el disco de determinadas maneras genéricas. Por ejemplo, un tipo de técnica busca en determinados directorios, otro busca cadenas dentro de archivos, etc. Las peculiaridades de cada técnica se incluyen mediante instancias de la clase **Parametro**. Siguiendo el ejemplo anterior, si se

quiere buscar archivos de nombre "prueba.txt" en el directorio "C:\pruebadir", existirán dos parámetros, uno con nombre "archivo" y valor "prueba.txt" y otro con nombre "directorio" y valor "C:\pruebadir". Todas las técnicas heredan de la clase genérica **Tecnica** que incluye un método sobrecargado llamado **ejecutar** que se utiliza para comenzar la búsqueda definida en la técnica concreta.
Las técnicas, almacenan en forma de instancias de la clase **Parametro** el conjunto de resultados derivados de la ejecución, de manera que se tiene modelado tanto la naturaleza de la búsqueda como los resultados obtenidos en un solo objeto.

- **Plugin:** La clase Plugin almacena el conjunto de técnicas que son definidas para realizar la búsqueda de residuos de un programa concreto. La idea consiste en que para cada programa se implemente un plugin indicando donde se deben buscar los residuos que deja dicho programa tras su instalación.
  Los plugins están escritos en xml siguiendo una estructura que se definirá más adelante en este mismo capítulo de la memoria y son convertidos a instancias de la clase Plugin mediante una clase del componente controlador Llamada GestorPlugins.

- **Accion:** Existen un conjunto de acciones definibles para ejecutar después de que la aplicación determine que se ha encontrado un Software especifico. Por ejemplo, si se ha encontrado un programa de esteganografía se puede definir una acción que ejecute un análisis en todo el disco en busca de archivos con información ocultada mediante este mismo programa.
  El funcionamiento del objeto **Accion** es idéntico al del objeto **Tecnica**, incluyendo el modelado de datos y el modo de ejecución.

Por último, cabe destacar, el uso de dos librerías externas: StegSecret y JniRegistry, utilizadas para la búsqueda de archivos con información oculta y para el análisis del registro de Windows respectivamente.

6.5.3. Componente Controlador

El Controlador es el componente que añade la funcionalidad a la aplicación. El grueso de clases y algoritmos que proporcionan los métodos para realizar las búsquedas y acciones necesarias se encuentran en este componente.

En el siguiente diagrama se puede observar la estructura del componente:

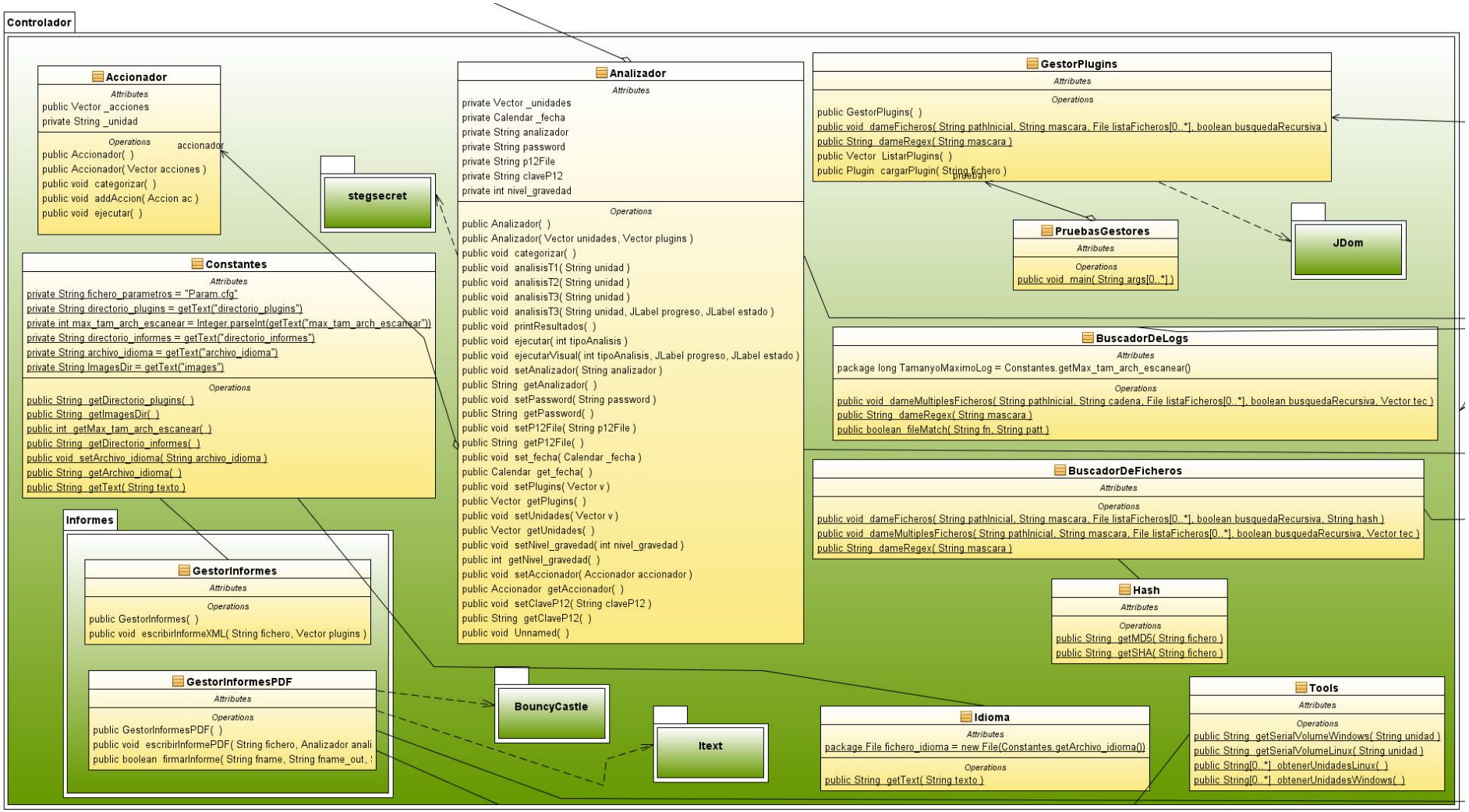


Gráfico 20. Diagrama de clases del paquete Controlador

- La clase principal en este controlador es **Analizador**, que se encarga de recopilar los datos del análisis que se va a realizar para así ejecutar las técnicas y acciones de forma secuencial. Esta clase contiene el conjunto de instancias del objeto **Plugin**, contiene los dispositivos a analizar y especifica otros parámetros externos como el nombre del analista que lanza la aplicación, su certificado, etc.

- La clase **Accionador** es muy parecida a **Analizador.** Gestiona todo lo relativo a las acciones predefinidas que se pueden lanzar tras encontrar residuos de software en el equipo.

- La clase **Constantes** contiene un conjunto de valores que determinan distintas características de uso en la aplicación como el idioma, la ubicación de los plugins, etc. Estos parámetros los coge de un archivo definido también en la misma clase como una constante.

- **GestorInformes** y **GestorInformesPDF** proporcionan los métodos principales para generar un informe a partir de una instancia del objeto **Analizador** del paquete **Controlador.** El primero genera informes en xml y el segundo en PDF, permitiendo a su vez firmar dicho documento con un certificado PKCS12.
  Para la generación y firmado de los documentos PDF se utiliza dos paquetes de librerías externos: **Itext** y **BouncyCastle**. Ambas vienen descritas en el apartado **5.5.2.1 Librerías Java** de esta misma memoria.

- La clase **GestorPlugins** permite parsear los plugins XML para convertirlos en instancias del Objeto **Plugin**. Para realizar este parseo se sirve de la librería externa **JDom**.

6.5.4. Componente Vista

El componente **Vista** corresponde a las distintas interfaces de usuario que permitirán acceder a la funcionalidad de la aplicación. Se trata de un componente completamente independiente del Modelo y Controlador.

Las dos interfaces implementadas se encuentran en las clases **Aplicacion** y **CommandLineApplication**. La primera ofrece una interfaz visual semejante a la mostrada en el capitulo **5.4.3 Componente Vista**, mientras que la segunda permite lanzar la aplicación adjuntando los parámetros que definen el análisis mediante línea de comandos.

En el siguiente diagrama se puede observar la estructura del componente:

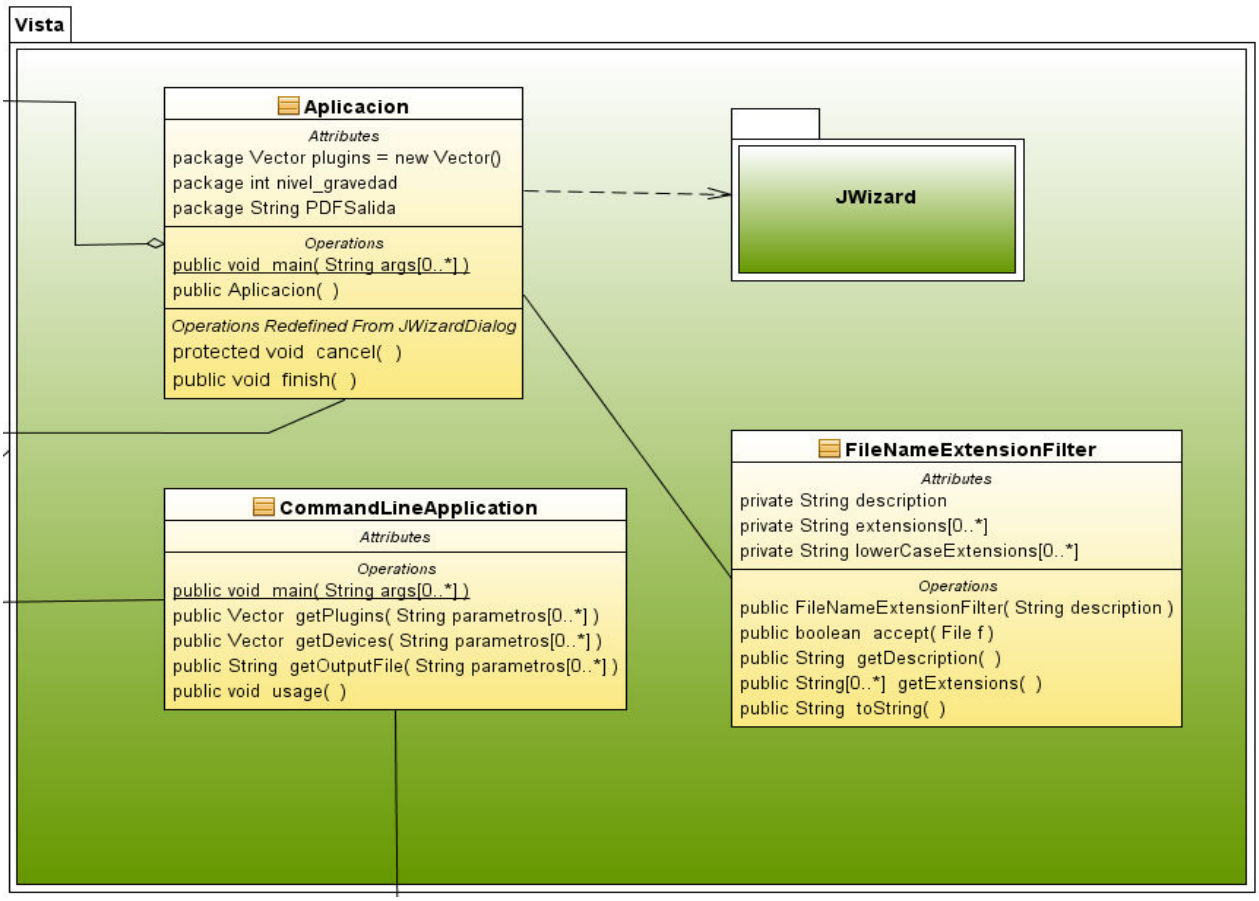


Gráfico 21. Diagrama de clases del componente Vista

7. Implementación e implantación del software

El objetivo de este capítulo es plasmar el diseño realizado en código abarcando todas las funcionalidades identificadas en anteriores capítulos, generando el programa ejecutable.

En anteriores etapas se ha establecido el diseño enfocado a un lenguaje de programación concreto, de manera que se ha sentado las bases que definirán la implementación de la aplicación.

Se realiza hincapié en obtener una implementación de calidad, que obtenga resultados fiables y que funcione eficientemente facilitando el mantenimiento futuro.

En esta etapa se realizan las pruebas de software, tanto unitarias como de integración, de modo que se pueda garantizar que cada pieza o programa funciona correctamente tanto a nivel individual como en el contexto del sistema.

En el segundo y último apartado de este capítulo se exponen los resultados de las pruebas de aceptación propuestas durante la fase de análisis. Es importante indicar que dichas pruebas han sido realizadas al mismo tiempo que se desarrollaba el código.

7.1. Proceso de codificación

Tras lo expuesto en la etapa de diseño se puede obtener una imagen global del funcionamiento de la aplicación. A continuación se explica el flujo de funcionamiento de la aplicación paso por paso:

1. La aplicación interpreta el plugin y lo convierte en una instancia del objeto **Plugin**.
2. Se añaden esas instancias a una instancia del objeto **Analizador**.
3. Se determina los dispositivos/unidades que serán analizados por la aplicación y se añaden también a la instancia del objeto **Analizador**.
4. Con esta información se inicia el análisis. Cada plugin contiene un conjunto de instancias de objetos **Tecnica**. Se ejecutan secuencialmente estas técnicas.
5. Dentro de cada objeto **Tecnica** hay un conjunto de resultados y los resultados del escaneo de almacenan dentro de estos objetos.
6. Cuando ha finalizado el escaneo se revisa el conjunto de los resultados de las técnicas y si se han encontrado evidencias mediante las técnicas de ese plugin, muestra las acciones asociadas a dicho Plugin.
7. Se ejecutan secuencialmente las acciones solicitadas.
8. Se escribe el informe.

Para obtener una aplicación flexible que pudiera procesar los distintos elementos de una forma genérica fue necesario generar una clase llamada **Parametro**. La clase **Parametro** es una clase sencilla compuesta por un nombre y un valor. De esta manera se puede utilizar

esta clase como unidad genérica de información permitiendo definir qué tipo de información corresponde al parámetro y su valor. Se puede observar este concepto con el siguiente diagrama, que muestra un ejemplo de Técnica y otro de Acción y su relación con los elementos **Parametro**.

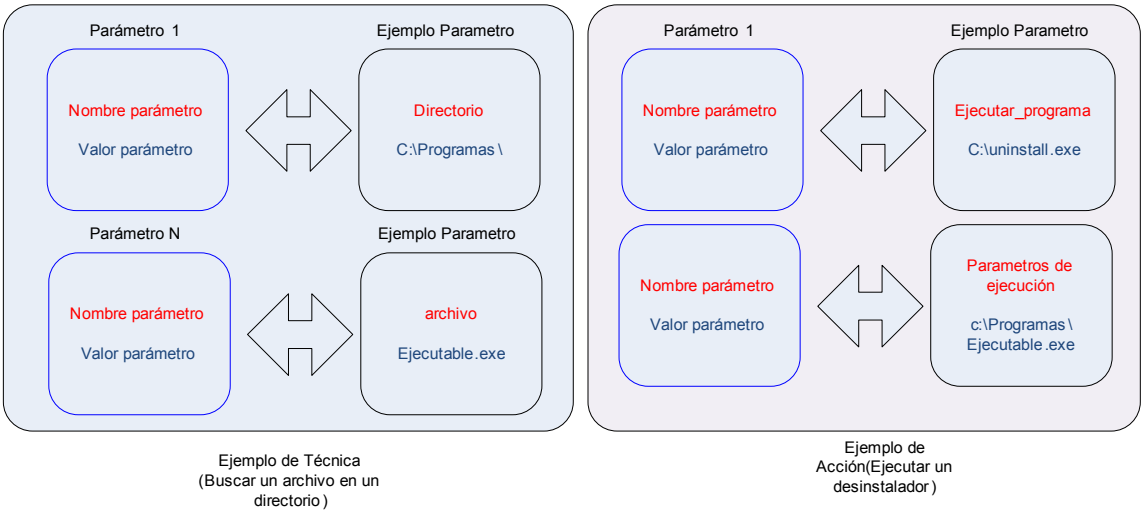


Gráfico 22. Ejemplo de relación entre Parámetros y Técnicas/Acciones

Los plugins están compuestos de **Tecnicas** y **Acciones**. Las técnicas y acciones solo difieren entre ellas en los parámetros que las contienen, de manera que, mediante este método se establece una forma genérica de definir los elementos variables de nuestra aplicación.

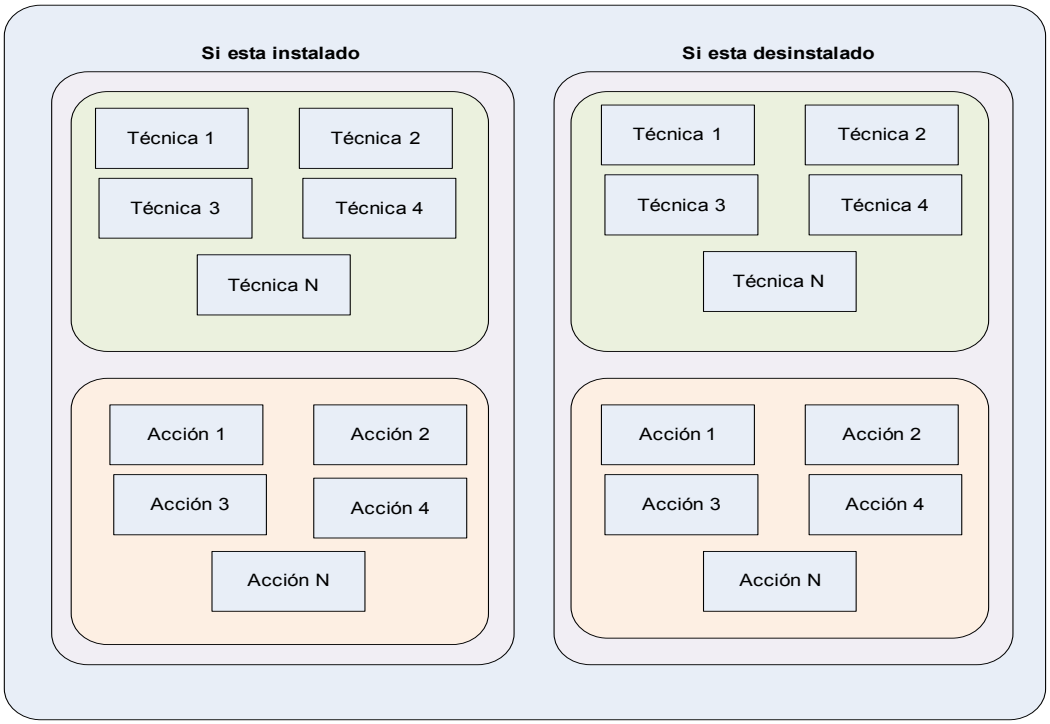


Gráfico 23. Estructura de los plugins

Para la interfaz gráfica se ha ideado un sistema mediante el cual se pueden seleccionar los plugins de una lista. Para ello, existe una función que coge todos los plugins presentes en una carpeta predefinida. Esta carpeta viene definida en el archivo de parámetros al que accede la clase Constantes.

Para evitar tiempos de espera muy grandes se han definido 3 tipos de búsquedas dependiendo de la profundidad del análisis. Estos tipos están basados en una división de las técnicas en función de su naturaleza. Por ejemplo, las técnicas que requieran el recorrido de todo el disco duro formarán parte del grupo de mayor profundidad, mientras que otras que requieran menos tiempo de procesamiento se asignarán a grupos de profundidades menores.

Estas técnicas se ejecutan de forma secuencial, comenzando por las que menores tiempos de procesamiento requieren hasta las que tienen mayor duración, salvo dos excepciones que se detallan en el subcapítulo ***6.2.Problemas encontrados y soluciones adoptadas***.

Como ejemplo de acción, se ha implementado un buscador de imágenes con información oculta. Esta acción utiliza los algoritmos de la aplicación **StegSecret**, que permite analizar si un fichero contiene información oculta mediante distintas técnicas detalladas en la propia aplicación. Cuando esta acción es lanzada, se ejecuta una búsqueda en todas las unidades de disco. No se han añadido la posibilidad de elegir más opciones alrededor de esta búsqueda porque se buscaba realizar una acción independiente a la aplicación y, añadir nuevas funciones para personalizar esta búsqueda concreta contradicen ésta premisa.

También se ha implementado una acción que permite lanzar un comando externo con unos parámetros determinados. De esta manera, cuando se encuentre un programa instalado, por ejemplo se podría lanzar un programa desinstalador.

En cualquier acción, tal como ocurre con las técnicas, la ejecución será personalizable dentro de los **Plugins** mediante elementos **Parametro** que permitirán definir formas específicas de ejecución dependiendo del software al que hace referencia el **Plugin**.

Para realizar unos resultados genéricos, compatibles con cualquier añadido de la aplicación, también se han basado en la clase **Parametro**. Un resultado es un **Parametro** con un nombre y un valor. De manera que si lo que se buscaba era un archivo, el resultado será un **Parametro** de nombre “archivo” y valor “c:\archivo….”, por ejemplo.

Se ha incorporado un sistema de registro para la aplicación que se aplica una vez terminado el análisis. Este sistema de registro introduce en el archivo predeterminado una línea indicando un resumen de los resultados obtenidos.

También, de manera automática, se genera un informe en XML en un directorio predefinido, de manera que se deja constancia de todos los análisis ejecutados. Los nombres de los archivos correspondientes a estos informes en XML se forman en función de la fecha y la hora, de manera que estén ubicados temporalmente de forma sencilla.

Algunas técnicas incluyen parámetros opcionales. Por ejemplo, en las búsquedas de archivos se puede incluir el hash del archivo, de manera que en las búsquedas además de comprobar el nombre del archivo, también se comprueba que el resultado de la función Hash MD5 o SHA-1 coincida con el introducido.

Las técnicas relacionadas con ficheros permiten, en el campo del fichero a buscar, incluir expresiones regulares, de manera que, cuando se busque *cadena* se encuentre todo lo que tenga la cadena aunque este rodeado de otros caracteres. Por ejemplo, la búsqueda con un parámetro "*plugin.xml" encontraría el archivo "pruebadeplugin.xml".

Los informes en formato PDF se escriben mediante la librería IText. Se escribe de manera automática recorriendo todos los plugins utilizados en el análisis y recorriéndolos imprimiendo los resultados al documento PDF.

Se permite la posibilidad de firmar los documentos con certificados PKCS12 exportados a ficheros de extensión P12. Esta firma se realizará mediante un método aislado que hará uso del paquete BouncyCastle. De esta manera, se ofrece la posibilidad de garantizar la autoría de los resultados obtenidos mediante análisis.

Todo lo relativo a los informes en documentos PDF se encuentra en una clase llamada GestorInformesPDF, al igual que los informes en formato XML se generan mediante los métodos de la clase GestorInformes.

Existen varios parámetros que alteran el funcionamiento de la aplicación y que pueden variar en función de las necesidades del usuario tales como el idioma, directorio de plugins, directorio de informes, etc. Estos parámetros son configurables a partir de un archivo de texto plano que indica los valores de los parámetros para un valor concreto. Estos valores son accesibles a las distintas clases de la aplicación a través de una clase estática llamada Constantes, que es la que se encarga de recuperar estos parámetros. Su nombre deriva de que inicialmente la clase constituía un conjunto de parámetros constantes que podían ser modificados, pero siempre editando el código. Después se decidió implementar un sistema que permite cambiar estos valores sin tener que acceder al código de la aplicación.

Se ha implementado un sistema de configuración de idioma que permite cambiar el idioma de la interfaz de la aplicación a partir de un fichero de texto plano que incluye las traducciones de los textos incrustados en la aplicación. Mediante este sistema se pueden generar distintos ficheros de idioma y traducir la aplicación de manera sencilla sin necesidad de modificar el código. El fichero de Lenguaje que va a ser utilizado tiene que ser definido en el archivo de parámetros.

La clase **Tools** del componente **Controlador** está constituida de un conjunto de métodos auxiliares que son requeridos en algunos de los procesos internos de la aplicación. Estos métodos son estáticos y por lo tanto no requieren la construcción de una instancia de la clase para su utilización.

7.2. Problemas encontrados y soluciones adoptadas

En determinadas fases de la implementación de la aplicación nos hemos encontrado con algunos inconvenientes que no habían sido contemplados en las fases previas de diseño.

- **Búsqueda en todo el disco, registro, etc.**

Una vez definidos los distintos análisis y tras definir el diseño se encontró un problema de rendimiento en la ejecución de ciertas técnicas. Específicamente con las técnicas que recorrían todo el disco duro o todo el registro de Windows.

En el planteamiento inicial, para favorecer la independencia entre las técnicas, se había ideado la ejecución secuencial. Sin embargo, para este tipo de técnicas, siguiendo ese paradigma se habrían realizado varios recorridos de todo el disco dos veces para buscar un determinado archivo, o todo el registro de Windows tres veces para buscar tres cadenas en su interior. Evidentemente, en aspectos de rendimiento esto es intolerable, por lo que se decidió hacer una excepción para ese tipo de técnicas.

En el método en la que las técnicas se agrupan en distintas categorías dependiendo de la duración en la ejecución de las mismas, se genera una técnica auxiliar que engloba las búsquedas a realizar de este tipo en una única técnica. Durante la ejecución de esta técnica auxiliar, en el mismo recorrido del disco se comprobaran los elementos a buscar de todas estas técnicas agrupadas, mejorando considerablemente el comportamiento de la aplicación en estos casos.

Para la acción implementada de búsqueda de información oculta mediante las librerías de StegSecret no se ha implementado esta mejora de rendimiento, principalmente por los argumentos que se han ofrecido para no ampliar las posibilidades de configuración previa a la acción: independizar completamente las acciones del conjunto de la aplicación y tratarlos como elementos genéricos. Para un análisis más óptimo y específico se podría lanzar la ejecución la misma aplicación original StegSecret, donde se podrá elegir varias opciones.

- **Obtener el volumen de los dispositivos**

En los informes de resultados, un elemento muy importante es definir donde se ha realizado el análisis. Una forma univoca de hacer referencia a un dispositivo montado es mostrar el número de serie el volumen del disco.

Para conseguir obtener este valor del disco se investigo en las herramientas que proporciona JDK para los dispositivos de entrada /salida pero no se encontró la manera de obtener el número de serie mediante este camino. Finalmente se tuvo que optar por ejecutar scripts externos llamando a comandos del sistema operativo nativo para conseguir dicho valor.

Existen dos scripts, que se lanzan dependiendo la plataforma sobre la que se ejecuta la aplicación y que permiten obtener este resultado.

- **Análisis físico del disco**

Como se ha visto en otros ejemplos de software de Análisis forense, una parte importante de este campo de la informática es la recuperación de información borrada pero que aún persiste en el disco. Los archivos que se borran en el disco duro no se borran realmente de la superficie del disco, sino que simplemente se borran los enlaces que definen el fichero y la información asociada al mismo.

Si los sectores del disco que contienen el fichero no han sido sobrescritos, la información sigue presente y puede ser recuperada. En nuestro caso, sería interesante poder recorrer los sectores del disco buscando conjuntos de sectores que correspondan con la información de un fichero concreto.

El inconveniente es que para realizar esta tarea es necesario acceso a disco a bajo nivel y el sistema de máquina Virtual de Java impone una capa entre la aplicación y el sistema operativo que impide realizar las llamadas necesarias al sistema y solo permite acceder a los discos mediante los mecanismos que la implementación de la máquina Virtual proporciona.

Es problema podría solucionarse utilizando una segunda aplicación generada en un lenguaje que permita estos accesos al disco a bajo nivel como C.

Una de las premisas originales de la aplicación consistía en garantizar el funcionamiento de la aplicación bajo varios sistemas operativos y la inclusión de código escrito en un lenguaje compilado eliminaría esa capacidad.

Se podría haber generado una aplicación adicional para cada sistema operativo programada en C. Sin embargo el tiempo de desarrollo se habría incrementado considerablemente y finalmente ha sido preferible crear una herramienta robusta sin esta funcionalidad adicional ante la construcción de esta funcionalidad sacrificando otros detalles.

- **Obtener los dispositivos montados**

La interfaz gráfica de la aplicación permite la selección de los dispositivos de entre todas las unidades montadas en el sistema. Para ello, Java incluye en la Clase “File” un método llamado “listRoots” que devuelve todas las unidades montadas. Sin embargo se ha observado que, mientras que en el sistema operativo Windows funciona correctamente, en los sistemas Unix devuelve solo del directorio raíz “/”.

Para subsanar este problema se ha recurrido, al igual que en otro caso, la obtención del conjunto de unidades mediante un script que realiza llamadas a los comandos nativos del sistema operativo.

- **Tamaño máximo de disco a analizar**

Una de las funcionalidades más importantes de la aplicación consiste en buscar una cadena de texto en el contenido de un fichero. Esto es muy interesante para analizar los ficheros de log que generan las distintas aplicaciones y que puede prestarnos pistas definitivas para valorar si un programa ha sido instalado en un equipo.

Un aspecto a tener en cuenta en esta búsqueda es que puede retrasar en gran medida el transcurso de la búsqueda si se busca en ficheros de tamaño muy grande. Por otro lado, un fichero de texto plano tiene un tamaño comúnmente pequeño y estar recorriendo el contenido de ficheros grandes, como Videos o imágenes de DVDs no tendría mucho sentido.

Este problema se ha solucionado añadiendo un parámetro configurable en el que se establece el tamaño máximo de los archivos que van a ser analizados. En principio este valor es de 2000000 bytes, pero puede ser modificado en el fichero de parámetros, por defecto "Param.cfg".

7.3. Ejecución del plan de pruebas de la aplicación

En este capítulo se muestran los resultados de las pruebas de aceptación establecidas en el apartado ***4.4.2. Diseño del plan de pruebas de aceptación.***

El resultado de la ejecución de dichas pruebas se puede ver en la siguiente tabla.

Id	Estado
PA-01	Superado
PA-02	Superado
PA-03	Superado
PA-04	Superado
PA-05	Superado
PA-06	Superado
PA-07	Superado
PA-08	Superado
PA-09	Superado
PA-10	Superado
PA-11	Superado
PA-12	Superado

Tabla 24. Resultado de las pruebas de aceptación

8. Conclusión y líneas futuras

Tras todos los apartados anteriores y la finalización del desarrollo de la aplicación de análisis forense "AFA" durante el tiempo expuesto en la planificación, se procederá a exponer una serie de conclusiones sobre el proyecto realizado. Para terminar, se mostraran una serie de conceptos e ideas acerca de una posible evolución del software en el subapartado **Líneas Futuras**.

8.1. Conclusiones del proyecto

Este apartado abarca tres cuestiones principales. En primer lugar se plantea la dificultad que conlleva la realización de un proyecto de estas características. En segundo lugar se analiza el resultado obtenido y el cumplimiento de los objetivos inicialmente planteados. En tercer y último lugar se comenta brevemente algunos detalles acerca de la etapa de Desarrollo del proyecto.

8.1.1. Dificultades del proyecto

El desarrollo de este software se ha generado con un conjunto inicial muy reducido de requisitos. Se especificaban unas premisas principales:

- Ejecución multiplataforma.
- Análisis forense en busca de residuos de aplicaciones instaladas.
- Diseño flexible que facilite la ampliación del software.

Bajo estas ideas preliminares se han ido desarrollando los conceptos para generar una aplicación con todos los elementos mencionados en los anteriores apartados.

Esto nos lleva a la primera dificultad con la que nos hemos encontrado en el proyecto: definir un conjunto de requisitos que conformen una aplicación bajo estas premisas y un diseño adecuado para la misma.

Se contemplaron varias posibilidades, pero era importante ofrecer la capacidad de aumentar el número de aplicaciones que serían detectadas mediante métodos externos que no requieran la inserción de nuevo código en la máquina. Para ello se decidió generar un sistema de plugins que permita insertar estos nuevos añadidos de la forma más sencilla y organizada posible. Esto añadió un gran esfuerzo de diseño y planificación previa a la implementación de la aplicación.

La segunda gran dificultad encontrada ha radicado en la propia naturaleza de la informática forense, una ciencia muy reciente y emergente que cuenta con mucha menos documentación que otros campos. Este hecho genera varios inconvenientes. Por ejemplo, la existencia de pocas herramientas y la dificultad que entraña la dominación de las mismas. Por otro lado, el lenguaje de programación utilizado no ofrece mecanismos orientados a este campo.

Otro de los problemas encontrados ha sido establecer un compromiso entre facilidad de uso y funcionalidad. Como se ha comentado anteriormente, existen aplicaciones más completas pero con fines más abstractos y manejo complicado que limitan su uso a un conjunto reducido de usuarios expertos. En nuestra aplicación se ha buscado diseñar e implementar una serie de funcionalidades con un objetivo definido y orientado a un grupo de usuarios muy amplio.

Una dificultad añadida ha sido garantizar la ejecución en varios sistemas operativos distintos. Java garantiza el funcionamiento bajo la máquina virtual, pero existen determinadas diferencias como la arquitectura de los sistemas de ficheros, comandos internos del sistema que son llamados desde la aplicación.

Se añade además la dificultad de multiplicar la investigación y el conjunto de pruebas para garantizar su funcionamiento en varios sistemas operativos.

Por último, se han encontrado dificultades a la hora de generar certificados desde la plataforma Java, requiriendo finalmente el uso de una librería para simplificar un proceso para el que aún no se proporcionan mecanismos potentes y usables de forma nativa en el JDK.

8.1.2. Resultados obtenidos

El desarrollo del presente proyecto ha sido motivador para las distintas partes implicadas por tratarse de una aplicación innovadora dentro de su campo (ampliando el perfil de usuario analista forense) y basarse en la premisa de ofrecerla tras su entrega a la comunidad de software libre para que la use y amplié.

El proceso de creación de este software, como en todos los casos, se basa en unos patrones de trabajo que garanticen un producto final con una documentación y calidad aceptables. La ingeniería del software es la encargada de proporcionar estos métodos y herramientas que han sido adaptados para el desarrollo del proyecto paso por paso.

Tras un proceso de implementación superior al planificado ante los problemas encontrados durante este periodo, se puede decir que se han cumplido las expectativas propuestas logrando obtener la funcionalidad inicialmente descrita y añadiendo nuevos

Por último, es destacable mencionar las facilidades proporcionadas por los entornos de desarrollo IDE utilizados (Netbeans y Eclipse), que han facilitado en gran manera la implementación de un proyecto disgregado en tantos elementos. Proporcionando incluso el primero de ellos herramientas para realización de algunos de los diagramas UML requeridos.

8.1.3. Desarrollo del proyecto

Para elaborar un proyecto de estas características es necesario un estudio exhaustivo de las distintas tecnologías que se aplican en su desarrollo además de una investigación profunda del área asociado al software.

También ha sido necesario realizar pruebas sobre los distintos sistemas operativos, (en algunas ocasiones profundas como en el análisis del registro de Windows), e investigar acerca

de aspectos desconocidos para el desarrollador como el funcionamiento interno del sistema Mac OS. Por otro lado ha sido necesario incorporar scripts individuales para los distintos Sistemas, teniendo que aprender nociones del lenguaje de scripting correspondiente.

Por lo tanto, el desarrollo del proyecto no solo ha servido para generar una aplicación sino que ha permitido incorporar nuevos conocimientos de diversa índole a los adquiridos por el desarrollador a lo largo de la carrera.

Por último, cabe destacar el conjunto de ampliaciones definidas en el apartado **Líneas Futuras**, que índica que la aplicación desarrollada no es un software estático y que ofrece muchas posibilidades de cara al futuro.

8.2. Ejemplos de implantación

En este apartado se indicarán posibles casos en los que la aplicación desarrollada podría resultar especialmente útil.

8.2.1. Caso 1: Violaciones de la política empresarial

Es habitual en empresas grandes establecer unos límites a los trabajadores en las aplicaciones que se puedan instalar en los equipos corporativos. Nuestra aplicación podría ser utilizada para averiguar periódicamente el uso que han hecho estos usuarios de sus equipos y averiguar si han instalado aplicaciones prohibidas como Emule u otras aplicaciones que pudieran producir fugas de información o facilitar puertas traseras a la red corporativa.

Si se conserva correctamente los elementos que definen las evidencias digitales, tales como cadena de custodia y autoría, se podría presentar este mal uso de los recursos corporativos en un litigio por despido improcedente.

8.2.2. Caso 2: Delitos informáticos generados con determinadas aplicaciones

Para determinados delitos informáticos se suelen utilizar determinados tipos de aplicaciones tales como gestores remotos de equipos infectados, software de acceso anónimo a través de internet, herramientas para el desarrollo de exploits, etc.

Estas herramientas podrían ser detectadas por la aplicación AFA y podrían ser usadas como evidencias en un proceso legal, siempre que se garantizara la autoría y la cadena de custodia.

8.2.3. Caso 3: Comprobación de incompatibilidades entre determinado software

Si los fabricantes de un software venden una aplicación indicando que mantiene ciertas incompatibilidades con otras aplicaciones, es posible, que antes de su adquisición, se quiera garantizar que estas aplicaciones nunca han estado presentes en los sistemas donde se implantara el primer software.

Por otro lado, si se ha adquirido ese software y muestra incompatibilidades con los sistemas en los que se ha instalado, la empresa desarrolladora puede garantizar que es por culpa de esas aplicaciones anteriormente instaladas, y para ello puede utilizar la aplicación AFA y presentar sus informes certificados como prueba, siempre que se respeten el resto de restricciones de las evidencias digitales.

8.2.4. Caso 4: Análisis de software previo a una posible migración

Podría ser interesante para un usuario o empresa, realizar un inventario previo del software instalado en sus sistemas para garantizar que estos van a mantener su usabilidad tras una migración de hardware o Sistema operativo. Con un número suficiente de plugins, esta tarea resultaría mucho menos tediosa utilizando la aplicación AFA que realizando el inventario de manera manual.

8.3. Líneas Futuras

En este subcapítulo se expondrán las posibles mejoras que se han considerado interesantes sobre el sistema desarrollado.

El tiempo de realización de un proyecto y los escasos recursos imponen unos límites que impiden la ampliación del sistema en determinados casos. Estos casos concretos de mejora y otras distintas aplicaciones del software desarrollado se describen con más detalle en las siguientes líneas.

- **Integración con ids y gestores de logs**: Sería interesante para una empresa configurar la aplicación para que lance un escaneo diario en cada equipo y luego recolectar estos logs para averiguar si se ha instalado algún software prohibido, tipo Emule o Bittorrent, en el algún equipo. En el ámbito del open source, sería muy sencillo integrarlo en una plataforma que utilice OSSIM (Open Source Security Information Management) para correlar los datos proporcionados por el log de la aplicación y controlar los elementos instalados en los equipos de la red.

- **Añadir la fase de preservación de la información**: Tal como se indicaba en el apartado ***2.3.1. Fases del computo forense***, una de las fases típicas del análisis forense consiste en la replicación del disco bit a bit antes de iniciar el análisis. También se indicó que esa fase no pertenece al conjunto de funcionalidades actual de la aplicación. Los motivos son varios, sin embargo los principales son la contradicción de esta funcionalidad con la premisa de plantear un análisis sencillo. La automatización de esta tarea sería complicada, pues se requeriría el uso de una aplicación externa implementada en un lenguaje que admita llamadas a bajo nivel, para poder acceder a la información bit a bit.
  En primer lugar, la generación de una imagen de todo el disco requiere un tamaño libre en otro disco igual que el disco origen y un tiempo de

procesamiento bastante elevado, algo que la mayoría de los usuarios de nuestra aplicación no tolerarían.
Se había planteado la posibilidad de generar imágenes AFF, un formato de imagen de discos orientado al análisis forense. Sin embargo se trata de una tecnología aún poco extendida y las facilidades para implementar esta opción bajo Java son nulas.
Por otro lado, también se estudio la posibilidad de cargar imágenes de este tipo. Mediante el código del proyecto Fuse (http://fuse.sourceforge.net/) se ofrecía un camino para una implementación de esta posibilidad. Sin embargo, la versión para Java no era multiplataforma y llevaba dos años abandonada, por lo que se descartó la idea.

- **Añadir fase de recuperación de la información:** Otro de los elementos de la informática forense más importante consiste en la recuperación de información borrada pero presente físicamente en el disco. Mediante mecanismos que acceden directamente a nivel de bloque, se reconstruyen ficheros o parte de ellos que podrían contener los residuos del software que la aplicación busca.
  Como se ha indicado en capítulos previos de la memoria, Java no permite mecanismos para acceso al disco a nivel de bloque, de manera que de forma nativa no se podía implementar una solución para este problema. La solución habría pasado por integrar un programa externo programado en C para este fin, que, de nuevo, se habría enfrentado con la premisa inicial del funcionamiento multiplataforma de la aplicación.

- **Nuevas técnicas integrando con sistemas externos de análisis forense:** Sería interesante incluir nuevas técnicas que usaran las funcionalidades de aplicaciones ya existentes de análisis forense para favorecernos posibilidades que no están soportadas por nuestra aplicación.

- **Mejora de los informes realizados:** Los informes automáticos que genera la aplicación incluyen la lista de resultados de una forma estructurada. Sin embargo se podrían implementar informes más completos que incluyan datos estadísticos de aplicaciones encontradas, técnicas que han obtenido resultados, probabilidad de la existencia del software en base a los resultados obtenidos (actualmente si detecta una técnica de un plugin ya determina que ha sido instalado), etc.

- **Aplicación adjunta para crear plugins mediante entorno gráfico:** Los plugins se han estructurado bajo código XML con un archivo de definición de datos para definir de la forma más explícita el formato que deben tener. Sin embargo para el gran público todavía es demasiado complicado definir nuevos plugins.
  Sería interesante incluir una aplicación que genere estos plugins mediante herramientas gráficas, de manera que no sea necesario editar ficheros XML para incluir una nueva búsqueda.

- **Aplicación adjunta para navegar por los distintos informes obtenidos**: En la aplicación se ha implementado un sistema automático que almacena los

resultados en un informe XML en una carpeta definida en el archivo de parámetros. Sin embargo, para ver estos informes es necesario abrir un editor de texto e interpretar el código XML. Para la mayor parte de los usuarios eso aún es complicado y por lo tanto sería interesante incluir un visor de informes XML para llevar el control histórico de análisis realizados.

- **Permitir realizar análisis cancelados a medias:** En la actual implementación un análisis representa un recorrido completo por los plugins definidos y no acaba hasta que ha ejecutado todas las técnicas que se han solicitado. Si en medio de un análisis se cancela, los resultado obtenidos hasta ese momento son ignorados y la aplicación finaliza. Sería interesante incluir una solución que permita seguir trabajando con los resultados obtenidos hasta ese momento.

- **Incluir técnicas de análisis de procesos:** Sería interesante la implementación de técnicas que analizaran los procesos activos del sistema en busca de actividad de los programas buscados. Actualmente esta es una técnica habitual en los antivirus para detectar rápidamente elementos sospechosos.

- **Uso de la aplicación como antivirus configurable**: Un posible uso de la aplicación podría ser la de búsqueda de virus. Evidentemente, con las técnicas implementadas no podría competir con un antivirus de verdad pero si existen un gran conjunto de virus que podrían ser detectados. Por ejemplo, al definir el hash de un archivo podemos encontrar gran cantidad de ejecutables que corresponden a virus. También la búsqueda de cadenas en el registro de Windows es uno de los métodos más comunes de identificación de malware.

- **Añadir acciones de descifrado de archivos:** Al igual que se ha implementado, a forma de ejemplo, un sistema que busca archivos con información oculta en caso de encontrar aplicaciones de esteganografía, si se encuentra con una aplicación de cifrado de datos, sería interesante buscar archivos cifrados en el disco e implementar herramientas de descifrado.

- **Mejorar el sistema de firma de informes:** Actualmente se utiliza el paquete Itext en conjunción con el BouncyCastle para realizar el firmado de documentos PDF. Itext no soporta marcas de tiempo de fuentes fiables de fecha y hora para realizar el firmado y, por lo tanto, la firma no será válida al 100%.
  Sería interesante en un futuro mejorar este aspecto para garantizar la validez de los informes firmados.

9. Bibliografía

En este capítulo es indicarán elementos de referencia usados para el desarrollo del proyecto, incluyendo tanto los recursos electrónicos como los libros de consulta.

9.1. Recursos electrónicos

- La información relacionada con el análisis forense necesaria para realizar ***4.2. Estudio del estado actual de la informática forens*e** ha sido obtenida de los siguientes recursos electrónicos.
  - **Arcert (Coordinación de emergencias en redes teleinformáticas de Argentina)**. Presentación en PDF introductoria a la informática forense.
    Enlace:
    http://www.arcert.gov.ar/ncursos/material/aspectos_legales_delitos_y_forense.pdf
  - ***Página Web de Rafael Ausejo Prieto***. Una recopilación de información acerca de la informática forense y su campo de aplicación.
    Enlace: http://www.ausejo.net/seguridad/forense.htm
  - ***Wikipedia***. Resumen general de la informática forense y sus aspectos generales.
    Enlace: http://es.wikipedia.org/wiki/Informática_forense
  - ***Blog "bad-robot"***. Donde se indican algunas de las aplicaciones de análisis forense y seguridad de código libre más utilizadas.
    Enlace: http://bad-robot.blogspot.com/2009/03/digital-forensic-tools-imaging.html
  - ***Documentación del kit de aplicaciones "The Sleuth Kit"***. Donde se indican las características principales del kit y su modo de empleo entre otras cosas:
    Enlace: http://wiki.sleuthkit.org/index.php?title=The_Sleuth_Kit
  - **Internet Security Auditors**. Presentación realizada por Daniel Fernández Bleda para el encuentro sobre Hacking de 2004 Hackmeeting (Sevilla).
    Enlace: http://www.isecauditors.com/es/index.html
  - ***Bormart S.A.*** Artículo escrito por David Echarri en referencia a las evidencias electrónicas. En el último párrafo se hace referencia al uso legal de las mismas indicando las diferencias en los procesos con las evidencias convencionales.
    Enlace: http://www.borrmart.es/articulo_redseguridad.php?id=1376
  - ***El país (Diario)***. Artículo de Tomás Delclos del 11/12/2008 donde se indica la controversia que generan las evidencias electrónicas en procesos legales por culpa de una legislación que no contempla artículos específicos para ellas.
    Enlace:
    http://www.elpais.com/articulo/portada/marana/criterios/rodea/obtencion/pruebas/electronicas/elpepisupcib/20081211elpcibpor_1/Tes

- IDG.es. Artículo en el que se indican los pasos a seguir en un análisis forense que implica evidencias digitales con objeto de presentarlas en un proceso judicial.
  Enlace: http://www.idg.es/pcworldtech/Analisis-forense.-Como-investigar-un-incidente-de-/art194718-Seguridad.htm
- Conectronica.com. Información general acerca de la informática forense y las pruebas digitales.
  Enlace: http://www.conectronica.com/Seguridad/Seguridad-forense-técnicas-antiforenses-respuesta-a-incidentes-y-gestión-de-evidencias-digitales.html

- Las estimaciones de costes se han realizado en base a los datos obtenidos en la página web de la Asociación de Doctores, Licenciados e Ingenieros en Informática (A.L.I).
  Enlace: http://www.ali.es/modules/miprofesion/item.php?itemid=36

- La información sobre los distintos sistemas operativos, la estructura de ficheros y los posibles elementos principales que se ven modificados tras la instalación de software sobre los mismos ha sido obtenida de los siguientes recursos electrónicos:
  - ***Linux.org***. Los principales ficheros y directorios de configuración del sistema operativo Linux han sido recogidos de la página web
    Enlace: http://www.linux-es.org/node/112
  - ***Wikipedia***. La información general referente al sistema operativo Mac Os 10 ha sido obtenida al siguiente artículo de la Wikipedia.
    Enlace: http://es.wikipedia.org/wiki/Mac_OS_X_v10.5.
  - ***Blog BrightMac***. Información acerca del sistema de ficheros y directorios de configuración del sistema operativo Max Os X.
    Enlace:http://brightmac.wordpress.com/2007/12/18/estructura-de-directorios-en-mac-os-x/
  - **Microsoft®.** La información acerca del registro de Windows, su estructura y sus funciones se ha obtenido de la página web de soporte de Microsoft®.
    Enlace: http://support.Microsoft®.com/kb/256986/es

- **CSI**. Para la redacción de los documentos se ha utilizado una adaptación libre de la Métrica en su versión 3:
  Enlace: http://www.csi.map.es/csi/metrica3/index.html

- **LinuxManPages**. Para la implementación de los scripts de Linux y Mac Os se han utilizado las páginas de manual de Linux.
  Enlace: http://linuxmanpages.com/

- **Programacion.com.** Para desarrollar la interfaz gráfica de la aplicación se ha utilizado un manual de la citada página web.
  Enlace: http://www.programacion.com/java/tutorial/swing/

- **Monografías.com**. Se ha utilizado la información de esta página web para realizar la estimación de costes.
  Enlace:http://www.monografias.com/trabajos15/estimacionhipermedia/estimacion-hipermedia.shtml

- **GNU GPL v3.** Licencia utilizada para la liberación del software.
  Enlace: http://www.gnu.org/copyleft/gpl.html

- **OMG**. Especificación de UML para realizar el diseño y modelado de la aplicación.
  Enlace: http://www.omg.org/cgi-bin/doc?formal/09-02-03.pdf

- **W3C**. Especificación del Lenguaje de Etiquetado Extensible (XML), usado en la implementación de los Plugins.
  Enlace: http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML

9.2. Libros de consulta

- Para la implementación de la aplicación se han utilizado los siguientes libros de consulta:
  - **El lenguaje de Programación Java**. Fco. Javier Ceballos. Editorial RA-MA.
  - **Thinking in Java (4th Edition)**.Bruce Eckel**.**
  - **Learning UML 2.0.** Russ Miles; Kim Hamilton. Editorial O'Reilly.

- En la búsqueda de información sobre los sistemas operativos, sus características propias y parte de su funcionamiento se han consultado los siguientes libros:
  - **Unix. Sistema V. Version 4.** Kenneth H.Rosen, Richard T. Rosinski, James M.Farber, Doublas A.Host. Editorial Mc Graw Hill.
  - **Sistemas operativos.** Willian Stallings. Editorial Prentice Hall.

9.3. Apuntes

En el desarrollo del proyecto también se ha consultado apuntes de clase y transparencias adquiridas durante los años de estudio en la carrera de **Ingeniería Informática.**

- **Análisis y diseño de algoritmos**. Asignatura de 1º curso de carrera. Conceptos generales de programación.
- **Programación**. Asignatura de 2º curso de carrera. Conceptos generales de programación y algoritmos complejos.
- **Estructura de datos.** Asignatura de 2º curso de carrera. Conceptos generales de programación, almacenamiento de datos y algoritmos complejos.
- **Interfaces de usuario**. Asignatura de 3º curso de carrera. Conceptos generales sobre el diseño de interfaces de usuario en aplicaciones informáticas.

- **Ingeniería del software I**. Asignatura de 4º curso de carrera. Introducción a la ingeniería del software.
- **Ingeniería del software II**. Asignatura de 4º curso de carrera. Métodos y aplicación de los estándares de desarrollo. Documentación.
- **Ingeniería del software III.** Asignatura de 5º curso de carrera Gestión de proyectos.
- **Sistemas informáticos.** Asignatura de 5º curso de carrera. Desarrollo general de una aplicación combinada con la ingeniería del software.
- **Sistemas operativos.** Asignatura de 2º curso de carrera. Conocimiento general del funcionamiento de los sistemas operativos.

10. Glosario de términos

A continuación se muestra, en orden alfabético, un conjunto de definiciones de términos que forman parte del documento que podrían ser desconocidos para el lector.

- ***A.L.I.*** - Asociación de Doctores, Licenciados e Ingenieros en Informática.
- ***ADSL*** - Tipo de conexión ofrecida por las operadoras de manera más común.
- ***AFF*** – Avanced Forensic Format. Formato de imagen que alberga una copia exacta del contenido de un dispositivo de almacenamiento y que proporciona ciertas ventajas relacionadas con el análisis forense.
- ***Autoría*** – Propiedad que se atribuye al creador o generador de un objeto u acción.
- ***Bittorrent*** - Software P2P utilizado comúnmente para obtener archivos recientes disponibles por gran cantidad de gente. La diferencia respecto a Emule es que es más veloz pero dispone de un contenido más limitado.
- ***Cadena de custodia*** - Proceso que verifica la integridad y manejo adecuado de la evidencia.
- ***Certificado digital*** - Un Certificado Digital es un documento digital mediante el cual un tercero confiable (una autoridad de certificación) garantiza la vinculación entre la identidad de un sujeto o entidad y su clave pública.
- ***Codificación*** – Proceso de implementación de los distintos ficheros que componen el contenido en un lenguaje de programación de una aplicación.
- ***Correlación*** – Análisis que ofrece resultados asumibles para el procesamiento humano a partir de un número elevado de variables.
- ***Diff*** – Comando unix que muestra la diferencia entre dos ficheros de texto.
- ***Documentos Parseables*** - Ficheros en texto plano que siguen una determinada estructura idéntica de manera que se puede obtener la información de los mismos a través de un proceso automatizado.
- ***Eclipse*** - Entorno de desarrollo creado por IBM especialmente diseñado para la implementación de código Java.
- ***Ejecución portable*** - Software que no requiere ningún proceso previo a la ejecución tales como modificaciones sobre el sistema de ficheros o registros del sistema.
- ***Emule*** - Software P2P que representa una de las plataformas de intercambio de archivos entre usuarios más importantes.
- ***Esteganografía*** - La esteganografía es la disciplina en la que se estudian y aplican técnicas que permiten el ocultamiento de mensajes u objetos, dentro de otros, llamados portadores, de modo que no se perciba su existencia.
- ***Exploit*** - Software o fragmento de código malicioso que se aprovecha de una vulnerabilidad de seguridad en un sistema.
- ***Fedora*** – Versión libre de la distribución de Linux orientada a empresas "Red Hat".
- ***FreeBSD*** - FreeBSD es un sistema operativo derivado de BSD, la versión de UNIX® desarrollada en la Universidad de California, Berkeley.
- ***Fuse*** - Filesystem in Userspace (FUSE, Sistema de archivos en Espacio de usuario) es un módulo cargable de núcleo para sistemas operativos de

computador tipo Unix, que permite a usuarios no privilegiados crear sus propios sistemas de archivos sin necesidad de editar el código del núcleo.

- ***Gestor de logs*** **–** Herramienta que recolecta logs de distintas fuentes y los centraliza ofreciendo resultados estadísticos.
- ***IDE*** - Entorno de desarrollo de código.
- ***Ids*** **–** Sistema de detección de intrusos. Aplicación o conjunto de aplicaciones que tiene como objetivo averiguar si un sistema o red está siendo objeto de un ataque que compromete la seguridad del mismo.
- ***Implantación*** **-** Proceso de instalar, ajustar y mantener una solución en un entorno de explotación.
- ***Implementación*** **-** Poner en funcionamiento, aplicar los métodos y medidas necesarios para llevar algo a cabo: En nuestro contexto la implementación es el proceso de desarrollar los algoritmos que componen la aplicación.
- ***Informática forense*** **-** Se considera informática forense a la aplicación de un conjunto de métodos científicos y analíticos especializada que permiten descubrir y presentar evidencias que sean válidas dentro de un proceso legal.
- interfaz
- ***Java*** – Lenguaje de programación que permite la ejecución del mismo código sobre distintos sistemas operativos gracias a que incorpora una capa intermedia que independiza el código del sistema sobre el que corre.
- ***Linux*** - LINUX es un sistema operativo, compatible Unix. Dos características muy peculiares lo diferencian del resto de los sistemas que podemos encontrar en el mercado, la primera, es que es libre, esto significa que no tenemos que pagar ningún tipo de licencia a ninguna casa desarrolladora de software por el uso del mismo, la segunda, es que el sistema viene acompañado del código fuente. El sistema lo forman el núcleo del sistema (kernell) más un gran número de programas / librerías que hacen posible su utilización.
- ***Litigio*** - es una controversia jurídica que surge entre dos o más personas. El término se utiliza habitualmente como sinónimo de juicio, pero su significado es algo más amplio. Su uso está más extendido en controversias jurídicas de carácter civil, mercantil o administrativo, y no tanto en juicios de carácter penal.
- ***MAC Os*** **-** (Macintosh Operative System). Mac OS es el nombre del sistema operativo creado por Apple para las computadoras Macintosh.
- ***Microsoft® Windows*** **-** Familia de sistemas operativos para pcs desarrollada por la empresa Microsoft®.
- ***Multiplataforma*** **-** Aplicación que funciona bajo distintos sistemas operativos.
- ***MVC*** **–** Siglas del modelo de arquitectura "Modelo Vista Controlador" que define un mecanismo para independizar las aplicaciones de la interfaz de presentación.
- ***Netbeans*** - Entorno de desarrollo creado por Sun Microsystems especialmente diseñado para la implementación de código Java.
- ***Ossim*** **-** Software de código libre que integra las más importantes herramientas de seguridad en comunicaciones open source y proporciona métodos de correlación para convertir todo este flujo de datos en unos datos manejables por un equipo humano.
- ***Páginas de manual de Linux*** **–** Conjunto de ficheros de ayuda de Linux accesibles desde línea de comando.

- ***PDF*** – Formato de visualización de documentos de gran difusión que está especialmente ideado para documentos susceptibles de ser impresos, ya que especifica toda la información necesaria para la presentación final del documento.
- ***PKCS12 -*** PKCS se refiere a un grupo de estándares de criptografía de clave pública concebidos y publicados por los laboratorios de RSA en California.
- ***Plugin-*** Añadido a una aplicación que amplía la funcionalidad de la misma sin necesidad de recompilar su código fuente.
- ***Registro del Sistema Windows*** – Mecanismo del sistema operativo Windows donde se almacena el conjunto de variables que define la configuración y estado del sistema.
- ***Requisitos software*** – Descripción completa del comportamiento del sistema que se va a desarrollar.
- ***Snapshot -*** Imagen de un momento concreto de un sistema informático donde se conserva el estado global y los valores de los registros del mismo en ese instante preciso.
- ***StegSecret*** – Software de código libre que permite escanear los archivos de un equipo en busca de ficheros que albergan información oculta mediante varios algoritmos. Esta programado en Java y permite la ejecución multiplataforma.
- ***Swing*** - Conjunto de herramientas ofrecidas en el paquete de desarrollo JDK de Java para facilitar la implementación de interfaces gráficas bajo este lenguaje.
- ***Ubuntu*** – Distribución de Linux de gran difusión actualmente, principalmente debido al impulso económico prestado por el multimillonario africano Mark Shuttleworth.
- ***UML*** – El "Lenguaje Unificado de Modelado" es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
- ***UNIX*** - Unix (registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T.
- ***Url*** – (Uniform Resource Locator) Es una secuencia de caracteres, de acuerdo a un formato estándar, que se usa para nombrar recursos, como documentos e imágenes en Internet, por su localización.
- ***Usabilidad*** - Característica de un sistema que define la capacidad de ser utilizado para las tareas que para las cuales el sistema se ha hecho.

Anexo 2: Manual de Instalación

En este anexo se explicaran los procedimientos que deben realizarse para poder ejecutar la aplicación en un equipo.

Como paso previo a la ejecución solo requeriremos la instalación de la máquina virtual de JAVA:

1. Descargar el archivo instalador para el sistema operativo sobre el que se ejecutará la aplicación de la url http://java.com/es/download/.

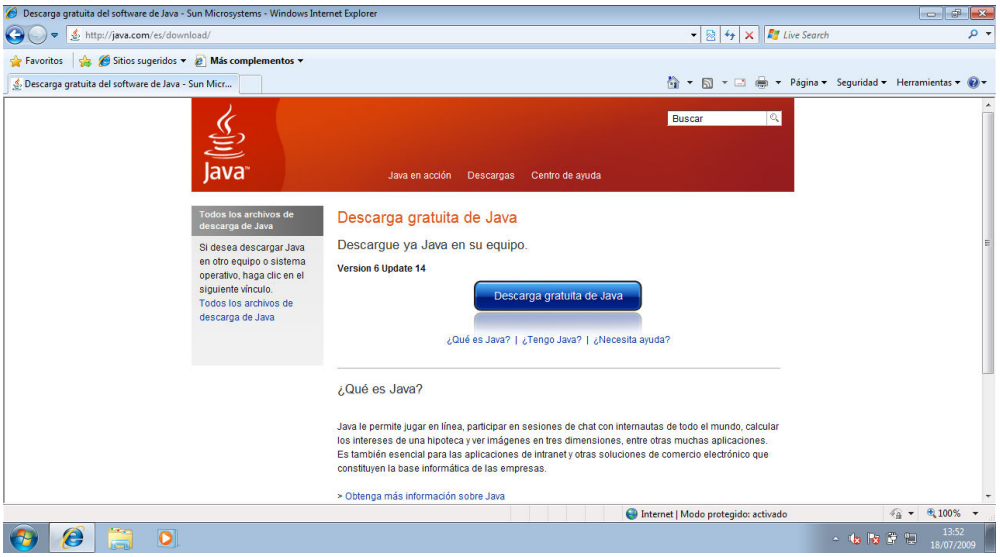


Gráfico 24. Descarga de Java (JRE)

2. A continuación se ejecuta el instalador descargado y aparece la siguiente pantalla de presentación:

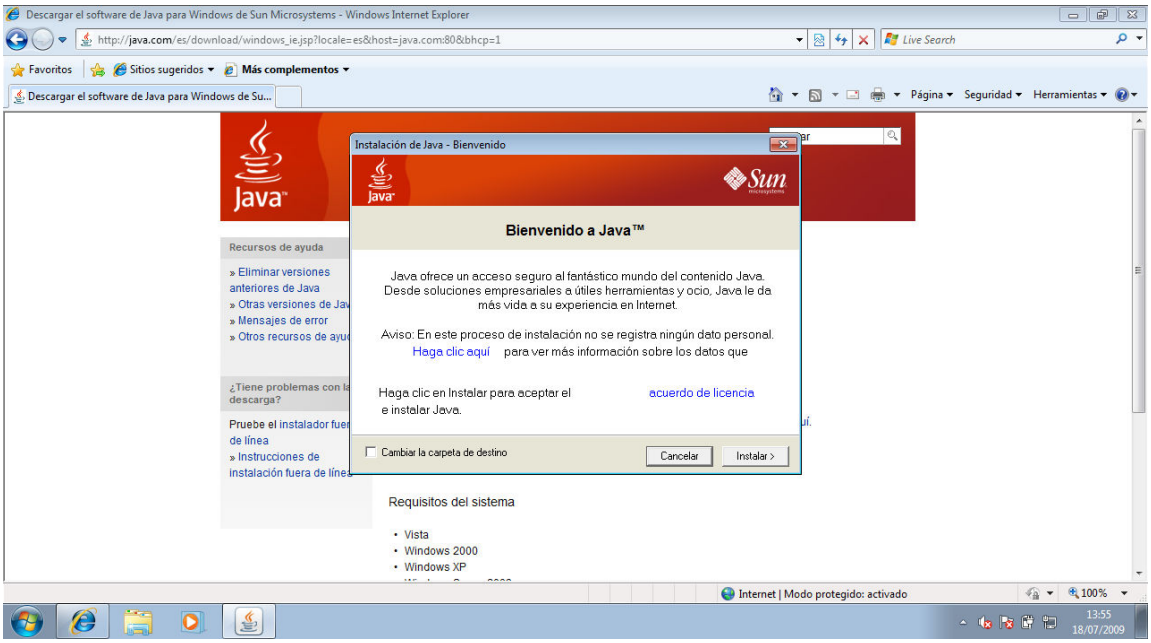


Gráfico 25. Comienzo de la instalación

3. A continuación pincha en el botón instalar y comienza la instalación de la máquina virtual JAVA:

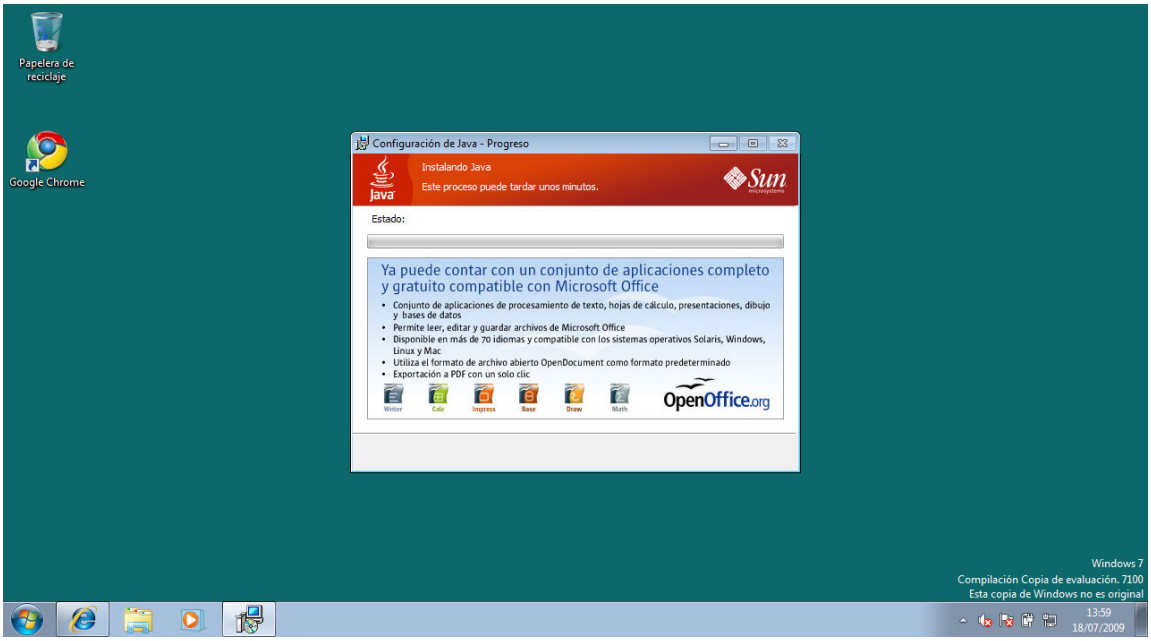


Gráfico 26. Progreso de la instalación

4. Por último el instalador nos informa que ya se ha instalado JAVA y que se realizan actualizaciones de forma periódica. Por último solo queda cerrar.

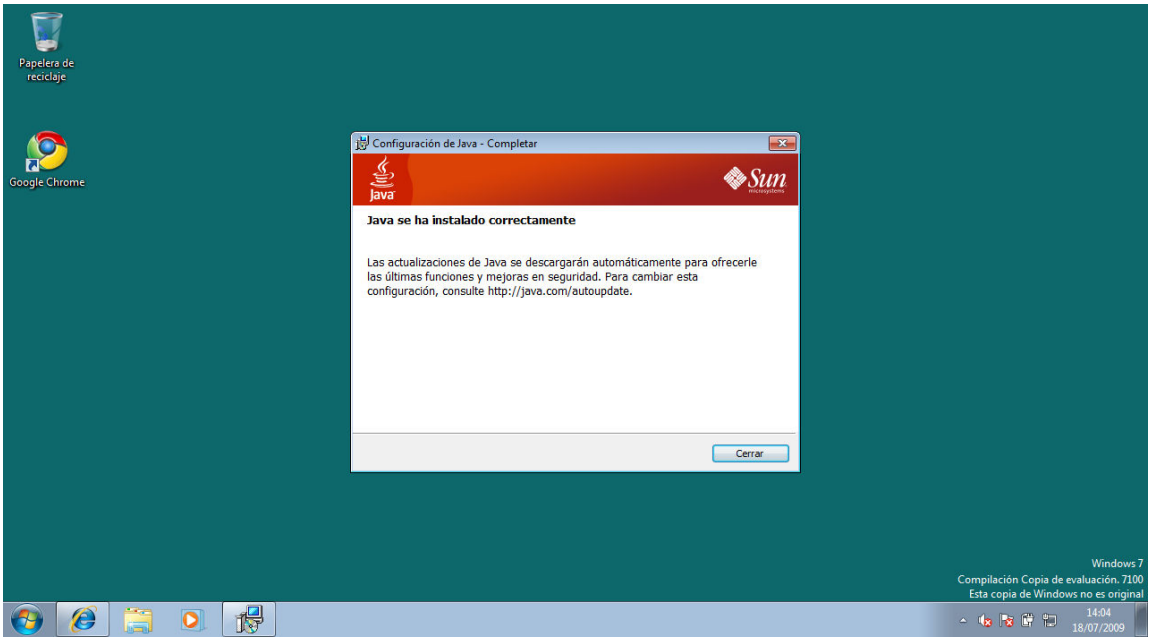


Gráfico 27. Finalización de la instalación de JAVA

5. Por último solo queda ejecutar la aplicación. La aplicación, típicamente vendrá empaquetada en un archivo comprimido .zip o .rar.
   Este archivo debe descomprimirse en una carpeta y ejecutar, dependiendo de la plataforma:

   - Para sistemas Windows se debe ejecutar el archivo AFAWIN.exe
   - Para sistemas Unix (Linux, MAC, etc) se debe ejecutar el archivo AFAlinux.sh.

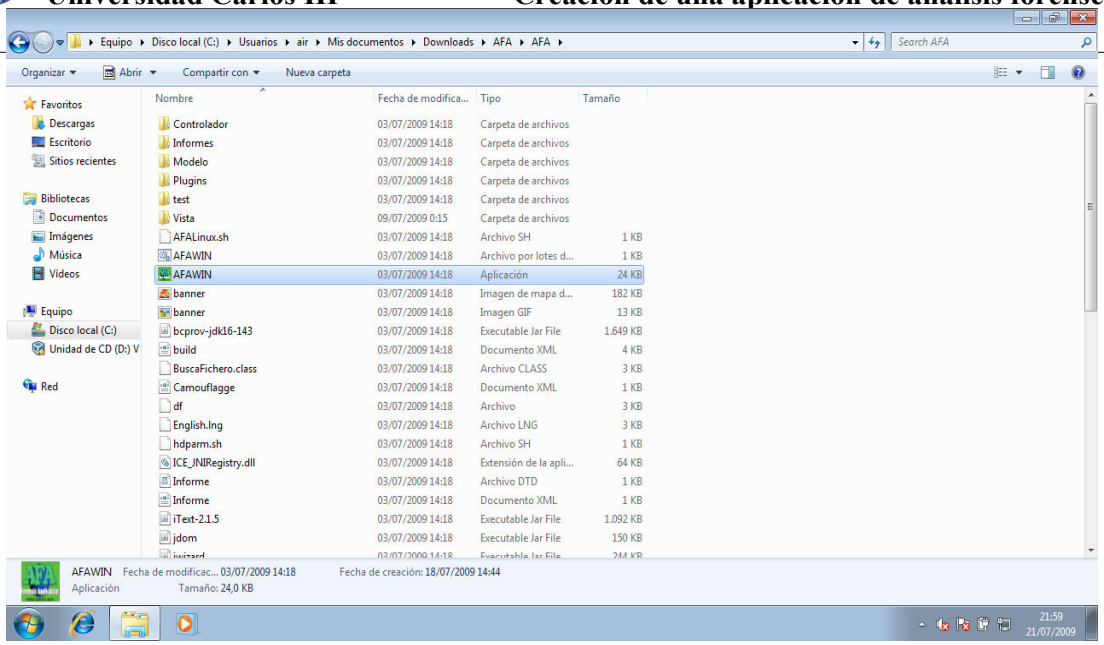


Gráfico 29. Archivos de la aplicación

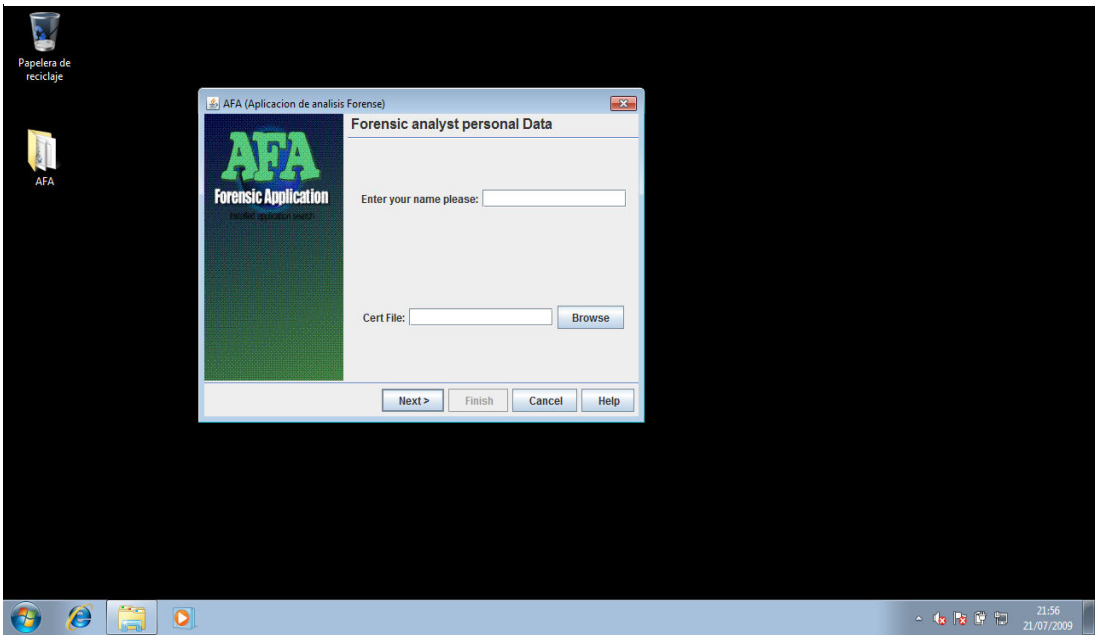


Gráfico 28. Ejecución de la aplicación

Anexo 3: Manual de Uso

En este apartado se mostrará una ejecución paso por paso explicando detalladamente que acciones se realizan en cada apartado.

1. **Pantalla inicial**. Se solicita al usuario su nombre y un fichero de certificado PSCK12.
   Introducir el nombre es obligatorio porque es necesario para especificarlo en el informe final en PDF. El fichero de certificado es opcional, si se incluye se pedirá también la clave privada necesaria para firmar el informe final con el certificado aportado.

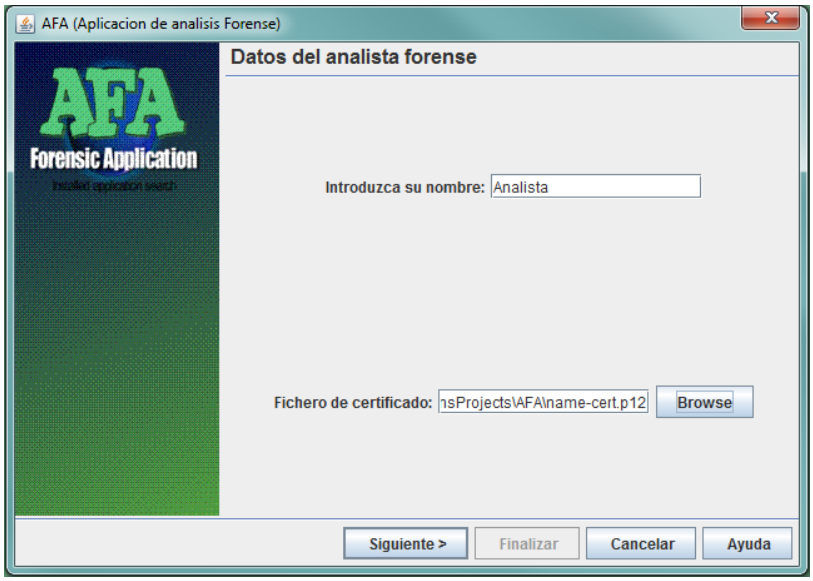


Gráfico 30. Ejecución de la aplicación

2. **Selección de dispositivos.** Se permite al usuario indicar las unidades en las que se desea realizar las pertinentes búsquedas. Se pueden seleccionar varios dispositivos de almacenamiento.
   Solo se mostraran los dispositivos de almacenamiento que estén montados en el sistema anfitrión en el momento de la ejecución de la aplicación.
   Hay que tener en cuenta que las técnicas del registro en Windows no buscan en los dispositivos sino en el registro diréctamente, de manera que estas técnicas no se verán afectadas por la selección.

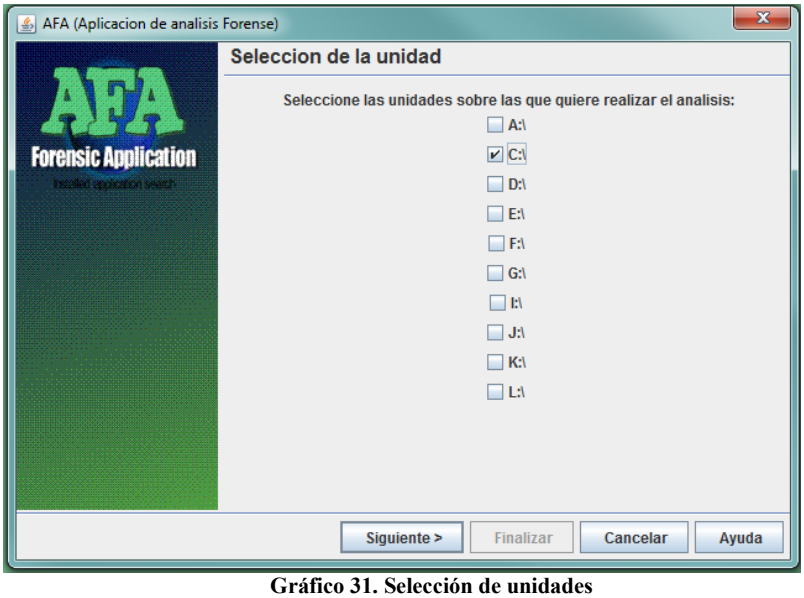


Gráfico 31. Selección de unidades

3. **Selección de Plugins.** Se permite al usuario indicar que plugins se quieren usar en el análisis. Cada Plugin define las búsquedas que se realizarán para encontrar una aplicación concreta en un sistema operativo concreto.
   Se pueden seleccionar todas mediante el botón de la parte derecha de la pantalla, o bien elegir manualmente los que se quieren incluir en el análisis.

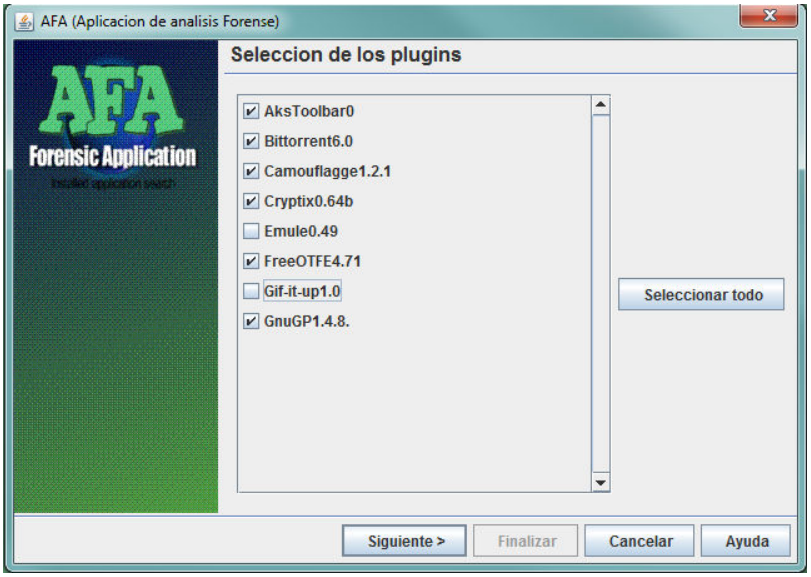


Gráfico 32. Selección de Plugins

4. **Selección de nivel de profundidad.** Se puede elegir entre 3 tipos de profundidad a la hora de realizar el análisis.

- **Baja** - Ejecuta solo las técnicas que menos tiempo de procesamiento requieren. Búsquedas concretas de archivos, cadenas en archivos concretos y claves de registro concretas.
- **Media** - Ejecuta técnicas que requieren más tiempo de procesamiento. Búsquedas de cadenas en todos los archivos de un directorio, búsquedas de registros en toda una sección del registro, etc.
- **Alta** - Ejecuta las técnicas que más tiempo requieren. Búsquedas que recorren en todo el disco y todo el registro.

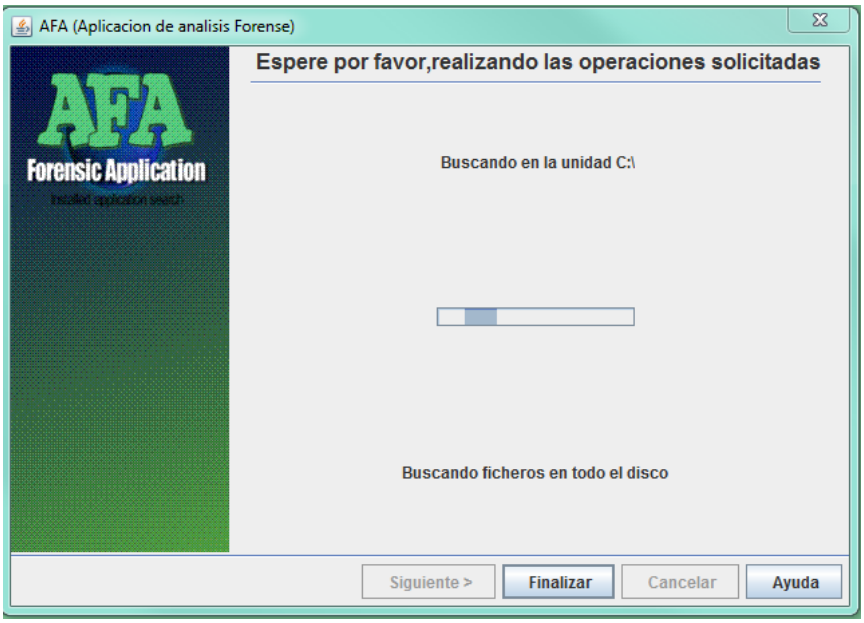


Gráfico 33. Ejecución del análisis

5. **Ejecución de acciones.** En esta fase es mostrará el conjunto de aplicación de las cuales se ha encontrado restos. Si existen acciones definidas para estos programas, se puede optar por ejecutar estas acciones. Para ello se seleccionan aquellas que interesan ejecutarse y se continúa pinchando en "siguiente", tras lo cual se ejecutarán secuencialmente y de forma automática.

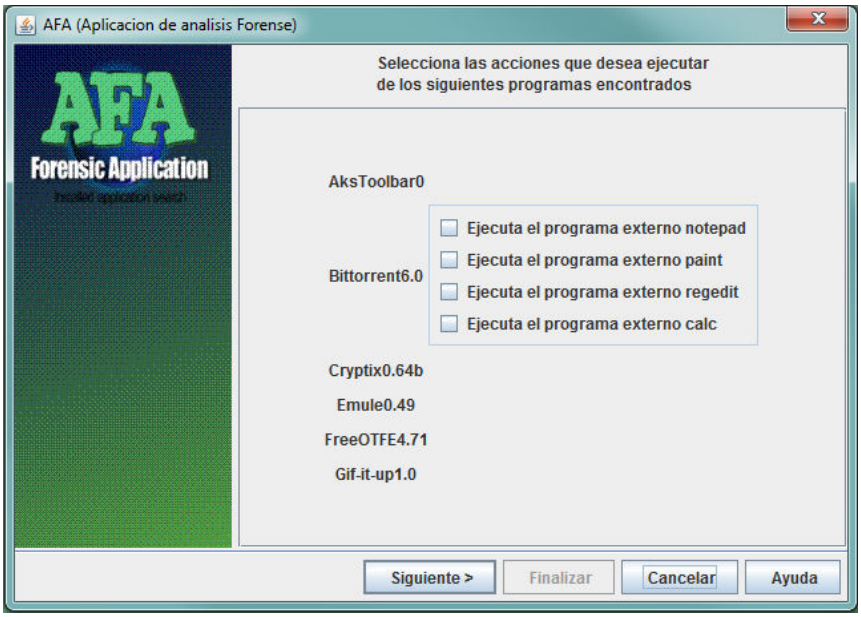


Gráfico 34. Selección de acciones

6. **Selección de fichero de salida.** Por último se incluirá la ruta en la que se quiere almacenar el fichero PDF con el informe del análisis. Se debe especificar la extensión del mismo (P.Ej. "informe.pdf").
   Es importante mencionar que este paso, en caso de haber elegido un certificado PKCS12, falla con versiones de Java anteriores a la 6.0.

Gráfico 35. Selección de fichero PDF de salida

7. **Pantalla final.** Se muestra la ruta del informe final indicando que ya listo para su visionado.

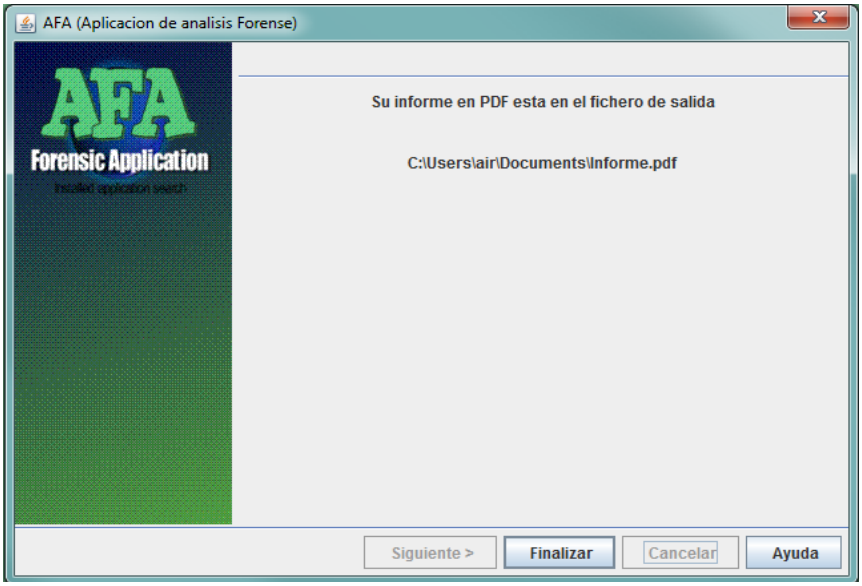
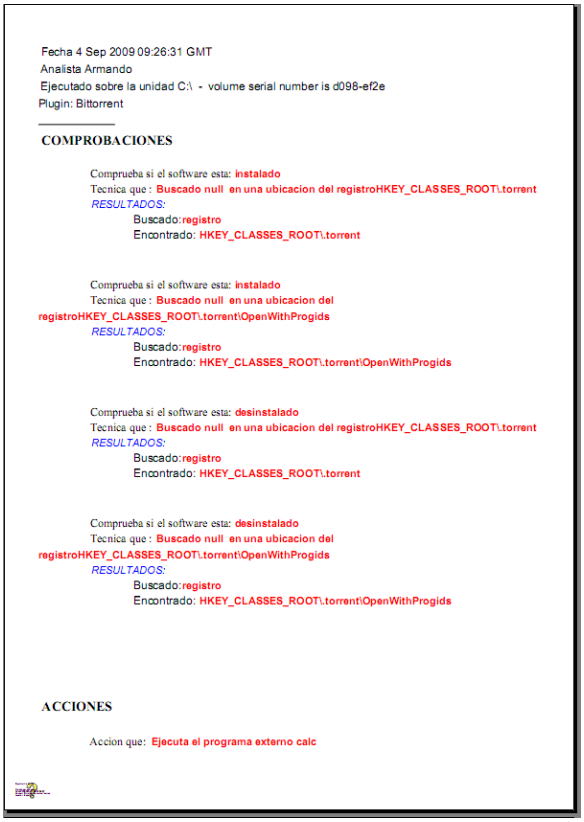


Gráfico 36. Pantalla final

8. **Por último, se abre el archivo indicado, obteniendo un informe como el siguiente:**



Fecha 4 Sep 2009 09:26:31 GMT
Analista Armando
Ejecutado sobre la unidad C:\ - volume serial number is d098-ef2e
Plugin: Bittorrent

COMPROBACIONES

Comprueba si el software esta: instalado
Tecnica que : Buscado null en una ubicacion del registroHKEY_CLASSES_ROOT\.torrent
RESULTADOS:
Buscado:registro
Encontrado: HKEY_CLASSES_ROOT\.torrent

Comprueba si el software esta: instalado
Tecnica que : Buscado null en una ubicacion del registroHKEY_CLASSES_ROOT\.torrent\OpenWithProgids
RESULTADOS:
Buscado:registro
Encontrado: HKEY_CLASSES_ROOT\.torrent\OpenWithProgids

Comprueba si el software esta: desinstalado
Tecnica que : Buscado null en una ubicacion del registroHKEY_CLASSES_ROOT\.torrent
RESULTADOS:
Buscado:registro
Encontrado: HKEY_CLASSES_ROOT\.torrent

Comprueba si el software esta: desinstalado
Tecnica que : Buscado null en una ubicacion del registroHKEY_CLASSES_ROOT\.torrent\OpenWithProgids
RESULTADOS:
Buscado:registro
Encontrado: HKEY_CLASSES_ROOT\.torrent\OpenWithProgids

ACCIONES

Accion que: Ejecuta el programa externo calc

Gráfico 37. Informe final de resultados

Hasta el momento, y a modo de ejemplo, se han creado varios Plugins que permitirán detectar varios programas que se detallan a continuación:

Plugin	Descripción
AskToolbar	Toolbar para el navegador Internet Explorer que se instala automáticamente junto a numerosas aplicaciones
Bittorrent	Programa P2P de intercambio de archivos
Camouflagge	Software de esteganografía que permite ocultar cualquier tipo de archivo en un archivo de imagen
Cryptix	Colección de utilidades de criptografía para el sistema operativo MAC Os
Emule	Software P2P de intercambio de archivos
FreeOTFE	Programa que permite realizar el cifrado de discos de forma transparente a usuario. Funciona bajo el sistema operativo Windows
Gif-it-up	Programa de esteganografía que permite ocultar archivos de texto, txt o html, dentro de imágenes Gif.
GNUPG-MAC	Software para cifrar y firmar ficheros o correos.

Tabla 25. Conjunto de plugins implementados

Anexo 4: Manual de creación de plugins

1. Introducción

Los Plugins son definiciones de acciones y técnicas asociadas a la búsqueda de restos de un programa concreto. Estos Plugins se escriben en código XML basándose en un fichero de definición de datos DTD.

A continuación se enumeran los pasos necesarios para escribir un Plugin. Vamos a escribir un Plugin de ejemplo para describir estos pasos de forma coherente. En este caso el Plugin va a detectar el programa FreeOTFE http://www.freeotfe.org/.

Es conveniente informarse, de manera previa a la implementación del Plugin tener cierto conocimientos sobre el formato XML. Este tema no se tratará de forma extendida en este apartado pero se proporcionan documentos de referencia en la bibliografía de esta memoria, aunque posiblemente la información más fiable y actualizada se podrá encontrar en la página oficial de w3c (http://www.w3c.es/divulgacion/guiasbreves/tecnologiasXML).

2. Técnicas implementadas

Para la definición de los distintos plugins hay disponibles una serie de técnicas implementadas que referencian distintos tipos de búsquedas en el sistema. A continuación se explicaran las características de cada una:

2.1. Técnicas relacionadas con búsquedas en disco

- **barchivo:** Busca un determinado archivo en un determinado directorio. Para ello hay que definir los parámetros **nombre** y **directorio.** Tiene un parámetro opcional que permite comprobar si el archivo encontrado además concuerda con el hash del fichero original. Este hash puede estar generado en MD5 o en SHA indistintamente.
- **barchivo_pname**: Busca un determinado archivo en todo el disco. Para ello hay que definir el parámetro **nombre** que hace referencia al nombre del archivo buscado. Tiene un parámetro opcional que permite comprobar si el archivo encontrado además concuerda con el hash del fichero original. Este hash puede estar generado en MD5 o en SHA indistintamente.

2.2. Técnicas relacionadas con búsquedas en el registro

- **breg_ub:** Técnica que permite buscar un registro concreto en una ubicación. El nombre y valor dentro de la clave del registro es opcional, de manera que si no se especifican estos valores solo comprobará si existe la clave definida.
- **breg_all:** Técnica que busca un valor en todo el registro. Recorre todos los atributos de todas las claves para comprobar si existen.
- **breg_all_sec:** Técnica que busca un registro concreto en una determinada sección del registro de forma recursiva en sus subsecciones. Al igual que la técnica **breg_ub**, el atributo y su valor es opcional, por lo que si no se especifican solo comprueba que exista la clave.

2.3. Técnicas relacionadas con búsquedas en el registro

- **blog:** Técnica que busca una cadena dentro de un fichero concreto. Es necesario indicar los parámetros cadena y fichero, con la cadena a buscar y la ruta del fichero donde se buscara respectivamente.
- **blog_pd:** Técnica que busca una cadena en los archivos de todo un directorio de forma recursiva. Se especifica la cadena y el directorio y recorre todos los archivos del mismo para determinar en cuales aparece.
- **blog_all:** Técnica que busca una cadena en los archivos de todo el disco. Se especifica la cadena y recorre todos los archivos del disco para determinar en cuales aparece.

3. Obtención de datos

Existen varios métodos para obtener el listado de las modificaciones que realiza sobre el sistema la instalación de un software. En el apartado **5.3 Análisis del efecto del software en los sistemas** se especifican los métodos utilizados por el Analista del proyecto.

La lista de elementos modificados por la instalación del software forman parte de los resultados extraídos de un estudio plasmado en el artículo ***FAUST: Forensic artifacts of uninstalled steganography tools*** de la revista electrónica de investigación www.sciencedirect.com. En este artículo se analizan los cambios que producen tras la instalación de distintos programas de esteganografía en el sistema operativo Windows. Estos resultado se detallan a continuación:

Ubicación en el registro:

HKLM\SYSTEM\CurrentControlSet\Enum\Root

Claves creadas:

LEGACY_FREEOTFECYPHERAES_LTC
LEGACY_FREEOTFECYPHERBLOWFISH

LEGACY_FREEOTFECYPHERCASTS
LEGACY_FREEOTFECYPHERCAST6_GLADMAN
LEGACY_FREEOTFECYPHERDES
LEGACY_FREEOTFECYPHERMARS_GLADMAN
LEGACY_FREEOTFECYPHERRC6_GLADMAN
LEGACY_FREEOTFECYPHERRC6_LTC
LEGACY_FREEOTFECYPHERSERPENT_GLADMAN
LEGACY_FREEOTFECYPHERTWOFISH_LTC
LEGACY_FREEOTFEHASHMD
LEGACY_FREEOTFEHASHRIPEMD
LEGACY_FREEOTFEHASHSHA
LEGACY_FREEOTFEHASHTIGER
LEGACY_FREEOTFEHASHWHIRLPOOL

Nueva Clave en el registro:

HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\FileExts\.txt\OpenWithList
ValueName: *
ValueData: FreeOTFE.exe

Nueva clave en el registro:

HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\FileExts\.vol\OpenWithList
ValueName: *
ValueData: FreeOTFE.exe

Nueva clave en el registro:

HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\MenuOrder\Start Menu2\Programs\FreeOTFE

Nuevos valoren en el registro:

Clave: HKCU\Software\Microsoft®\Windows\ShellNoRoam\MUICache
ValueName: C:\Documents and Settings\{username}\Local Settings\Temp\wnsu.tmp\ Au_.exe,
ValueData: FreeOTFE
ValueName: C:\Documents and Settings\{username}\My Documents\FreeOTFE_4_00.exe,
ValueData: FreeOTFE
ValueName: C:\Program Files\FreeOTFE\FreeOTFE.exe,
ValueData: FreeOTFE

Directorio:
C:\Program Files\FreeOTFE

Subdirectorio: alternate_drivers

FreeOTFECypherAES_Gladman
FreeOTFECypherTwoFish_Gladman
FreeOTFECypherTwoFish_HifnCS

Subdirectorio: weak_drivers

FreeOTFECypherNull
FreeOTFECypherXOR
FreeOTFEHashNull

Subdirectorios: amd64,x86

FreeOTFE
FreeOTFECypherBlowfish
FreeOTFECypherCAST6_Gladman
FreeOTFECypherMARS_Gladman
FreeOTFECypherRC6_ltc
FreeOTFECypherTwofish_ltc
FreeOTFEHashRIPEMD
FreeOTFEHashTiger
FreeOTFECypherAES_ltc
FreeOTFECypherCAST5
FreeOTFECypherDES
FreeOTFECypherRC6_Gladman
FreeOTFECypherSerpent_Gladman
FreeOTFEHashMD
FreeOTFEHashSHA
FreeOTFEHashWhirlpool

Directorio: My Documents

FreeOTFE_4_00

Directorio: C:\WINDOWS\Prefetch

FREEOTFE.EXE-1BCEAFCB.pf
FREEOTFE.EXE-16362389.pf
FREEOTFE_4_00.EXE-3610AC25.pf

4. Estructura del plugin

En primer lugar tenemos que estructurar el plugin.

Como se comento previamente, estos plugins deben respetar la estructura definida en el archivo “plugin.dtd” expuesto en el apartado 5.3.1.1. Definición de plugins:

En primer lugar escribimos las cabeceras que referirán a la codificación y el fichero DTD sobre el que se generarán las restricciones. Si el fichero va a estar en el mismo directorio que el archivo DTD (Comúnmente el directorio Plugins de la aplicación) no es necesario indicar la ruta

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM 'Plugin.dtd'>

El siguiente paso es definir los datos principales del Plugin.

<plugin name='FreeOTFE' so="Windows" version="4.71">
A continuación hay que definir las técnicas asociadas al Plugin. Estas se engloban en dos etiquetas, "instalado" y "desinstalado" haciendo referencia al estado del software en el sistema. Cada técnica sirve para un tipo de búsqueda.

<instalado>
<tecnicas>

Comenzaremos con las técnicas relacionadas con búsquedas en el disco.

```
<tecdisco>
        <barchivo directorio='%ProgramFiles%/FreeOTFE/alternate_drivers' nombre='FreeOTFECypherAES_Gladman' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/alternate_drivers' nombre='FreeOTFECypherTwoFish_Gladman' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/alternate_drivers' nombre='FreeOTFECypherTwoFish_HifnCS' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/weak_drivers' nombre='FreeOTFECypherNull' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/weak_drivers' nombre='FreeOTFECypherXOR' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/weak_drivers' nombre='FreeOTFEHashNull' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/amd64' nombre='FreeOTF*' />
        <barchivo directorio='%ProgramFiles%/FreeOTFE/x86' nombre='FreeOTF*' />
        <barchivo directorio='/WINDOWS/Prefetch' nombre='FreeOTF*' />
        <barchivo_pname nombre='FreeOTF*'/>
</tecdisco>
```

La técnica **barchivo** busca un archivo concreto en un directorio. Mientras que la ultima técnica definida **barchivo_pname** busca un archivo recorriendo todo el disco. Haciendo así el Plugin tenemos la opción de realizar una búsqueda de profundidad baja en la que busque todas las técnicas **barchivo** y si no encuentra nada, realizar un análisis de profundidad alta (y por lo tanto alta duración) que busque archivos en todo el disco.

A continuación se definen las técnicas relacionadas con búsquedas en el registro:

<**tecreg**>

```
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERAES_LTC'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERBLOWFISH'/>
        <breg_ub registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERCASTS'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERCAST6_GLADMAN'/>
        <breg_ub registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERDES'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERMARS_GLADMAN'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERRC6_GLADMAN'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERSERPENT_GLADMAN'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFECYPHERTWOFISH_LTC'/>
        <breg_ub registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFEHASHMD'/>
        <breg_ub registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFEHASHRIPEMD'/>
        <breg_ub registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFEHASHSHA'/>
        <breg_ub registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFEHASHTIGER'/>
        <breg_ub
registro='HKLM\SYSTEM\CurrentControlSet\Enum\Root\LEGACY_FREEOTFEHASHWHIRLPOOL'/>
        <breg_ub    registro='HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\FileExts\.txt\OpenWithList'
atributo='*' valor='FreeOTFE.exe' />
        <breg_ub    registro='HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\FileExts\.vol\OpenWithList'
atributo='*' valor='FreeOTFE.exe' />
        <breg_ub    registro='HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\FileExts\.txt\OpenWithList'
atributo='*' valor='FreeOTFE.exe' />
        <breg_ub
registro='HKCU\Software\Microsoft®\Windows\CurrentVersion\Explorer\MenuOrder\Start\Menu2\Programs\FreeOTFE' />
        <breg_ub registro='HKCU\Software\Microsoft®\Windows\ShellNoRoam\MUICache' atributo='C:\Documents and
Settings\{username}\Local Settings\Temp\wnsu.tmp\
Au_.exe,' valor='FreeOTFE' />
        <breg_ub registro='HKCU\Software\Microsoft®\Windows\ShellNoRoam\MUICache' atributo='C:\Documents and
Settings\{username}\My Documents\FreeOTFE_4_00.exe,' valor='FreeOTFE' />
        <breg_ub    registro='HKCU\Software\Microsoft®\Windows\ShellNoRoam\MUICache'    atributo='C:\Program
Files\FreeOTFE\FreeOTFE.exe,' valor='FreeOTFE' />
 </tecreg>
```

La técnica ***breg_ub*** busca un valor en una clave del registro concreta. Los parámetros atributo y valor son opcionales, de manera que si no existen no realiza comprobaciones sobre ellos y simplemente nos indica si la clave indicada existe en el registro.

Como no hay archivos de log que hayan sido modificados en la instalación no se definirán técnicas para los mismos. Lo siguiente sería comprobar los cambios en el sistema que se han generado tras la instalación e introducir estas técnicas en la etiqueta **<desinstalado>** de la misma manera que se ha hecho en la etiqueta **<instalado>**.

Por último solo queda cerrar las etiquetas xml abiertas y ubicar el archivo xml en la carpeta predefinida para los plugins en el archivo de parámetros de la aplicación. Por defecto este directorio es “Plugins”, un subdirectorio de la ubicación de la aplicación.

Anexo 5: Manual de creación de nuevas técnicas

Una de las prioridades en el diseño de la aplicación ha consistido en facilitar al máximo la ampliación de la herramienta, enfocando estas facilidades principalmente a la posibilidad de añadir más técnicas al conjunto de las disponibles.

Las técnicas representan las distintas formas de buscar restos de software instalado. Las técnicas son los elementos que definen un determinado plugin, tal como se ha visto en el ANEXO 4.

En primer lugar se tiene que implementar la técnica en una clase que herede de la clase **Tecnica** del paquete **Modelo**. Esta clase tendrá un método llamado **ejecutar()** que necesariamente tendrá que ser sobrecargado con las instrucciones pertinentes para ejecutar la técnica implementada.

Por otro lado hay unas estructuras heredadas donde se debe almacenar la información relevante a la técnica. Los datos de ejecución (tipo de técnica, parámetros de entrada, etc.) se recogerán del vector **_param.** Se pueden obtener estos datos con el método getValue() indicando el nombre del atributo que se quiere obtener. Por ejemplo, si una técnica busca en un directorio, un parámetro será el elemento que se busca y otro será el directorio y tendrán nombres de atributos “nombre” y “directorio” u semejantes.

Estos atributos estarán introducidos en la fase de parseo XML, como se muestra en apartados siguientes.

Una vez implementada la técnica es necesario modificar el archivo DTD “plugin.dtd” para ampliar la estructura general de los plugins y que incluyan estos nuevos tipos de técnica. Para ello, hay 3 grupos diferentes dependiendo de la naturaleza de la técnica: técnicas de registro, de disco y de búsqueda en logs. Se deben añadir en la categoría más apropiada o crear una nueva categoría. No se explicará en este apartado el funcionamiento de los archivos DTD pero a partir del archivo original y con un poco de documentación es fácil realizar estas modificaciones.

Por último hay que modificar el método **cargarPlugin(String fichero)** de la clase **GestorPlugins** del paquete **Controlador**.

En este método se parsean los archivos xml. Es necesario incluir código para que parsee el elemento implementado de forma coherente con las modificaciones realizadas sobre el archivo **plugin.dtd.**

Cuando se lee una técnica se crea una instancia del objeto **Tecnica** con los parámetros recibidos. En este caso, la instancia será de la técnica recientemente implementada. Finalmente solo hay que añadirla al vector de técnicas de la instancia de la clase Plugin que se crea durante el parseo utilizando el método **p.addTecnica(tec_implementada).** También es

necesario incluir el valor de la variable **_tipo_Tecnica** donde se informa de modo textual de la acción que realiza la técnica. Esta información se mostrara en el informe final**.**

Se muestra un ejemplo a continuación:

```
TecnicaDisco t=new TecnicaDisco("instalado",p);     //Se crea una nueva técnica que contendrá la información.Se define si está comprobando si el software está instalado o desinstalado.
                                                    t._tipo_Tecnica=Idioma.getText("Buscado el archivo") + " " + e.getAttributeValue("nombre") + " " + Idioma.getText("en el directorio")+ " " + e.getAttributeValue("directorio");//Se muestra la explicación de que hace la técnica. Utiliza las funciones de idioma para que la explicación sea traducida dependiendo del idioma utilizado en la ejecución.
                                                    Parametro param0=new Parametro("tipo","barchivo"));//Se crean parámetros con la información recogida del archivo xml.
                                                    Parametro param1=new Parametro("directorio",e.getAttributeValue("directorio"));//Se crean parámetros con la información recogida del archivo xml.
                                                    Parametro param2=new Parametro("nombre",e.getAttributeValue("nombre"));
"));//Se crean parámetros con la información recogida del archivo xml.
                                                    t.addParametro(param0); //Se introducen los parámetros en la técnica
                                                    t.addParametro(param1); //Se introducen los parámetros en la técnica
                                                    t.addParametro(param2); //Se introducen los parámetros en la técnica
                                                        if(e.getAttributeValue("hash")!=null){ //Se introducen los parámetros opcionales en la técnica
                                                        Parametro param3=new Parametro("hash",e.getAttributeValue("hash"));
                                                        t.addParametro(param3);
                                                    }
                                                    p.addTecnica(t); //Se añade la técnica al Plugin
```

Código 3. Ejemplo de plugin utilizado en el Manual de creación de Plugins

Se debe añadir este código dos veces, en la primera si la técnica se utiliza para comprobar si el software está instalado y en la segunda si se utiliza para comprobar que la técnica esta desinstalada.