Universidad Internacional de La Rioja
Escuela Superior de Ingeniería y Tecnología

Máster Universitario en Seguridad Informática

Desarrollo de un crawler para la búsqueda de información delictiva en la Dark Web

Trabajo fin de estudio presentado por:	Sergio Oteiza Echeverría
Tipo de trabajo:	Desarrollo software
Director/a:	Víctor Andrés Pimienta García
Fecha:	Julio 2022

Resumen

Las características de anonimidad y privacidad de la Dark Web la hacen atractiva para la comisión de delitos. Las autoridades necesitan herramientas para localizar patrones delictivos en esta red. Existen diferentes desarrollos de rastreadores o *crawlers* orientados a localizar contenidos en la Dark Web, generalmente desarrollados en inglés y enfocados al análisis del texto alojado en páginas web de la red TOR. Pese a que existen propuestas para implantar funcionalidades adicionales, como análisis de imágenes, rastreo focalizado en base a palabras clave o conexión a otras redes diferentes de la red TOR, se hace necesario el desarrollo de una herramienta modular, con arquitectura distribuida y multi idioma que aglutine en una única plataforma de manera gradual y flexible las diversas funcionalidades que habitualmente se implantan de manera aislada. Como resultado, se desarrolla mediante metodología de ciclo de vida de desarrollo software seguro (S-SDLC), un *crawler* con capacidad de conexión tanto a la red visible como a tres Dark Nets (TOR, I2P, Freenet) y que permite localizar páginas web en base no solo a criterios relacionados con su contenido de texto (como por ejemplo un *nickname* mostrado en una web), sino también según aspectos relativos a sus imágenes, mediante el análisis de hashes y metadatos. De este modo, el diseño modular realizado permite el planteamiento de nuevas líneas de desarrollo de la herramienta que puedan ser acometidas en un futuro e integradas sin dificultad.

Palabras clave: Rastreador, Crawler, TOR, I2P, Freenet, Dark Net, Dark Web.

Abstract

The anonymity and privacy characteristics of the Dark Web make it attractive for the commission of crimes. Authorities need tools to locate criminal patterns in this network. There are different developments of crawlers aimed at locating content on the Dark Web, generally developed in English and focused on the analysis of text hosted on web pages in the TOR network. Although there are proposals to implement additional functionalities, such as image analysis, focused crawling based on keywords or connection to other networks different from the TOR network, it is necessary to develop a modular tool, with distributed and multi-language architecture, that gradually and flexibly brings together in a single platform the various functionalities that are usually implemented in an isolated manner. As a result, a crawler is developed through a secure software development life cycle methodology (S-SDLC), with the ability to connect to both the visible network and three Dark Nets (TOR, I2P, Freenet) and to locate web pages based not only on criteria related to their text content (such as a nickname displayed on a website), but also according to aspects related to their images, through the analysis of hashes and metadata. In this way, its modular design allows for new lines of development of the tool that can be undertaken in the future and integrated without difficulty.

Keywords: Crawler, TOR, I2P, Freenet, Dark Net, Dark Web.

Índice de contenidos

1. Introducción 15
1.1. Motivación 17
1.2. Planteamiento del trabajo 17
1.3. Estructura de la memoria 18
2. Contexto y estado del arte 19
2.1. Estructura de Internet 19
2.1.1. Web visible 19
2.1.2. Web profunda (Deep Web) 20
2.1.3. Web oscura (Dark Web) 20
2.2. Principales Dark Nets 23
2.2.1. TOR 23
2.2.2. I2P 25
2.2.3. Freenet 26
2.3. Crawlers 27
2.3.1. Definición y modo de funcionamiento 27
2.3.2. Tipos de crawlers 30
2.3.3. Desafíos de los crawlers 32
2.3.4. Métricas asociadas a los crawlers 33
2.4. Información asociada a las imágenes 34
2.5. Trabajos relacionados 34
3. Objetivos y metodología 42
3.1. Objetivo principal 42
3.2. Objetivos secundarios 42
3.3. Metodología 43
3.3.1. Análisis 43
3.3.2. Diseño 44
3.3.3. Codificación 45

3.3.4. Pruebas 45
3.3.5. Producción 46
4. Desarrollo específico de la contribución 47
4.1. Análisis 50
4.2. Diseño 51
4.2.1. Configuración 57
4.2.2. DarkFinder 59
4.2.3. Crawler 60
4.2.4. Log 64
4.2.5. Gestor de base de datos 66
4.2.6. DarkSearch 70
4.2.7. Auxiliar 72
4.3. Codificación 73
4.3.1. Configuración 74
4.3.2. DarkFinder 75
4.3.3. Crawler 77
4.3.4. Log 79
4.3.5. Gestor de base de datos 80
4.3.6. DarkSearch 82
4.3.7. Auxiliar 84
4.3.8. Settings 85
4.3.9. Análisis estático de código fuente 85
4.3.10. Comprobación de la ausencia de vulnerabilidades en las librerías importadas 85
4.4. Pruebas 86
4.5. Producción 87
4.6. Resultados 88
5. Conclusiones y trabajo futuro 89

5.1. Conclusiones 89
5.2. Líneas de trabajo futuro 90
Referencias bibliográficas 92
Anexo A. Requisitos de la aplicación 103
Anexo B. Análisis de modelado de amenazas 109
Modelo de amenazas 109
Criterios de categorización de amenazas 109
Listado de amenazas 111
Resultado del modelado de amenazas 112
Informe de modelado de amenazas 115
Anexo C. Análisis de riesgos arquitectónico 119
Metodología 119
Determinación del contexto 120
Apreciación de riesgo 120
Tratamiento del riesgo 123
Determinación del contexto 123
Apreciación del riesgo 124
Listado de activos identificados y dependencias 125
Valoración del impacto 126
Listado de amenazas 127
Listado de riesgos residuales resultantes 130
Tratamiento del riesgo 141
Anexo D. Diseño de clases 142
Configuration 142
Page 145
Image 146
Anexo E. Plan de pruebas 148
Anexo F. Análisis estático de código fuente 162

Anexo G. Manual de instalación de la aplicación 163
Windows 163
Python 163
TOR 164
I2P 165
Freenet 165
MongoDB 166
Elasticsearch 169
Kali Linux 171
Python 171
TOR 172
I2P 173
Freenet 173
MongoDB 174
Elasticsearch 175
Anexo H. Resultados 176
Mostrar mensajes de ayuda y versión de la aplicación 176
Funcionamiento en español e inglés 176
Capacidad de rastreo en red visible, TOR, I2P y Freenet 178
Almacenamiento de información en base de datos no relacional MongoDB 181
Implementación de diferentes algoritmos de rastreo 182
Breadth-First Search 183
Depth-First Search 184
Best-First Search 185
Capacidad de búsqueda de páginas web 186
Funcionamiento en sistema operativo Kali Linux 188
Carga de datos de rastreo en aplicación Elasticsearch y visualización con Kibana 190
Anexo I. Código fuente 192

Índice de figuras

Figura 1. Estructura de Internet. 19
Figura 2. Indexación de buscadores en la Deep Web. 20
Figura 3. Clasificación por categorías de sitios web en la red TOR. 22
Figura 4. Número de usuarios conectados directamente a la red TOR. 24
Figura 5. Establecimiento de circuitos virtuales en TOR. 25
Figura 6. Establecimiento de túneles en red I2P. 26
Figura 7. Secuencia de petición típica en Freenet. 27
Figura 8. Arquitecturas centralizada y distribuida. 29
Figura 9. Estructura de crawler. 29
Figura 10. Interacción con formulario de crawler para la Deep Web. 39
Figura 11. Arquitectura de crawler distribuido de alto rendimiento propuesta. 41
Figura 12. Metodología de desarrollo propuesta. 43
Figura 13. Comparativa complejidad lenguajes de programación Python, C++, JavaScript y Java. 47
Figura 14. Ventajas e inconvenientes de Python. 48
Figura 15. Ventajas e inconvenientes de MongoDB. 49
Figura 16. Diagrama UML de casos de uso y abuso. 51
Figura 17. Diagrama UML de componentes de la aplicación. 52
Figura 18. Diagrama UML clases. 56
Figura 19. Fragmento de fichero de configuración *darkfinder.conf*. 57
Figura 20. Diagrama UML actividades módulo Configuración. 58
Figura 21. Diagrama UML actividades módulo DarkFinder. 59
Figura 22. Proxies de conexión a redes. 61
Figura 23. Diagrama de actividades UML módulo Crawler. 62
Figura 24. Diagrama UML actividad módulo Log – funciones '*debug*', '*warning*'. 66
Figura 25. Diagrama UML actividad módulo Log – funciones '*info'*, '*error', 'critical'*. 66
Figura 26. Diagrama Base de Datos No Relacional. 67

Figura 27. Ventajas e inconvenientes relación tipo *Embedded* en base de datos MongoDB. 68

Figura 28. Diagrama UML actividad Módulo Gestor base de datos – función 'save_web_page'. 70

Figura 29. Diagrama UML actividad Módulo Gestor base de datos – funciones 'url_crawled', 'search_web_page'. 70

Figura 30. Diagrama UML actividades módulo DarkSearch. 72

Figura 31. Ejemplo de código fuente. 73

Figura 32. Ejemplo de importación de funciones. 74

Figura 33. Ejecución del código en idiomas español e inglés. 75

Figura 34. Extracto de fichero *darkfinder_locales.po* en español. 75

Figura 35. Visualización de la ayuda del comando *darkfinder* en idiomas español e inglés. 76

Figura 36. Extracto de fichero *argparse_locales.po* en español. 76

Figura 37. Ejecución del proceso de rastreo. 76

Figura 38. Barra de progreso en el proceso de rastreo. 79

Figura 39. Extracto de fichero de log. 80

Figura 40. Captura de errores mediante bloques try-except. 80

Figura 41. Ejemplo de visualización de datos almacenados en base de datos MongoDB. 81

Figura 42. Visualización de la ayuda comando *darksearch* en idiomas español e inglés. 82

Figura 43. Visualización de resultados comando *darksearch.* 83

Figura 44. Visualización de resultados comando *darksearch.* Expresión regular para búsqueda de correos electrónicos. 83

Figura 45. Visualización de resultados comando *darksearch* con criterios de imágenes. 83

Figura 46. Visualización de resultados comando *darksearch* con criterios de *difference hash* relativos a imágenes. 84

Figura 47. Contenido del fichero requirements.txt. 86

Figura 48. Resultados comprobación de ausencia de vulnerabilidades en librerías importadas. 86

Figura 49. Resultados pruebas automatizadas. 87

Figura 50. Diagrama de modelado de amenazas. 113

Figura 51. Metodología de análisis de riesgos. 120

Figura 52. Matriz de cálculo de riesgos. 122

Figura 53. Distribución de riesgo arquitectónico residual. 125

Figura 54. Activos identificados en el análisis de riesgos y dependencias 126

Figura 55. Resumen del análisis estático de código fuente. 162

Figura 56. Resultados del análisis estático de código fuente en los scripts principales de la aplicación. 162

Figura 57. Resultados del análisis estático de código fuente en módulos de la aplicación. 162

Figura 58. Conexión a red TOR. 164

Figura 59. Configuración Privoxy (config.txt). Logfile. 164

Figura 60. Asistente de instalación de Freenet. Instalación de Java. 166

Figura 61. Ejecución de Freenet. Asistente primera conexión. 166

Figura 62. Fichero mongod.cfg. 167

Figura 63. Sección de base de datos MongoDB en el fichero de configuración *darkfinder.conf*. 169

Figura 64. Configuración automática de Elasticsearch y Kibana. 170

Figura 65. Exportación de colección MongoDB. 170

Figura 66. Importación de fichero CSV en Kibana. 170

Figura 67. Selección de base de datos y visualización de resultados en Kibana. 171

Figura 68. Conexión a red TOR. 172

Figura 69. Ejecución de Freenet. Asistente primera conexión. 173

Figura 70. Opción ayuda (-h) comandos *darkfinder* y *darksearch*. 176

Figura 71. Opción versión (-v) comandos *darkfinder* y *darksearch*. 176

Figura 72. Fichero de configuración. Clave 'locale'. 177

Figura 73. Funcionamiento comando *darkfinder* en inglés. 177

Figura 74. Funcionamiento comando *darksearch* en inglés. 177

Figura 75. Funcionamiento comando *darkfinder* en español. 177

Figura 76. Funcionamiento comando *darksearch* en español. 177

Figura 77. Rastreo de páginas web en red visible. 178

Figura 78. Rastreo de páginas web en red TOR. 178

Figura 79. Rastreo de páginas web en red I2P. 178

Figura 80. Rastreo de páginas web en red Freenet. 179

Figura 81. Rastreo de páginas web en múltiples redes. 179

Figura 82. Funcionamiento comando *darkfinder* en español modo silencioso (opción -q).. 179

Figura 83. Comienzo del proceso de rastreo de 1.000 páginas web. 180

Figura 84. Fin del proceso de rastreo de 1.000 páginas web. 180

Figura 85. Base de datos MongoDB en el proceso de rastreo de 1.000 páginas web. 180

Figura 86. Rastreo de 1.000 páginas web sin procesar imágenes. 181

Figura 87. Registro MongoDB página red visible. 181

Figura 88. Registro MongoDB página red TOR. 181

Figura 89. Registro MongoDB página red I2P. 182

Figura 90. Registro MongoDB página red Freenet. 182

Figura 91. Fichero de configuración rastreo tipo Breadth-First Search. 183

Figura 92. Proceso rastreo tipo Breadth-First Search. 184

Figura 93. Búsqueda *keywords* rastreo tipo Breadth-First Search. 184

Figura 94. Fichero de configuración rastreo tipo Depth-First Search. 184

Figura 95. Proceso rastreo tipo Depth-First Search. 185

Figura 96. Búsqueda *keywords* rastreo tipo Depth-First Search 185

Figura 97. Fichero de configuración rastreo tipo Best-First Search. 185

Figura 98. Proceso rastreo tipo Best-First Search. 186

Figura 99. Búsqueda *keywords* rastreo tipo Best-First Search. 186

Figura 100. Búsqueda de páginas web. Patrón de búsqueda URL. 187

Figura 101. Búsqueda de páginas web. Patrón de contenido mediante expresión regular en el cuerpo HTML. 187

Figura 102. Búsqueda de páginas web. Hash MD5 de imagen contenida en la página web. 187

Figura 103. Búsqueda de páginas web. Hash SHA1 de imagen contenida en la página web. 187

Figura 104. Búsqueda de páginas web. Distancia de Hamming a un *difference hash* de referencia de una imagen pasado como parámetro. 188

Figura 105. Búsqueda de páginas web. Patrón de búsqueda mediante expresión regular en los metadatos de una imagen. 188

Figura 106. Búsqueda de páginas web. Patrón de búsqueda mediante múltiples criterios. 188

Figura 107. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red visible. 189

Figura 108. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red TOR. 189

Figura 109. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red I2P… 189

Figura 110. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red Freenet. 190

Figura 111. Ejecución de comando *darksearch* en sistema operativo Kali Linux. 190

Figura 112. Análisis de información cargada y procesada por Elasticsearch mediante Kibana. 191

Figura 113. Análisis de redes rastreadas mediante Kibana. 191

Índice de tablas

Tabla 1. Análisis de servicios ocultos populares en la red TOR. 24
Tabla 2. Comparativa de diferentes tipos de web *crawler*. 32
Tabla 3. Distribución de tipos de enlaces de *hidden services* en la red TOR. 33
Tabla 4. *Crawlers* de la Dark Web con código disponible en Internet. 36
Tabla 5. Criterios de aceptación del análisis estático de código fuente. 45
Tabla 6. Comparativa lenguajes de programación C/C++, Java, Python y PERL. 47
Tabla 7. Módulos contemplados en el crawler. 53
Tabla 8. Principios de diseño de software seguro contemplados. 53
Tabla 9. Clases contempladas en el *crawler*. 57
Tabla 10. Interfaz módulo *Configuration*. 58
Tabla 11. Parámetros línea de comandos módulo DarkFinder. 59
Tabla 12. Puertos de conexión a proxy en función de la red. 60
Tabla 13. Interfaz módulo Crawler. 63
Tabla 14. Interfaces módulo *Log*. 65
Tabla 15. Entidad *Page*. 67
Tabla 16. Entidad *Image*. 67
Tabla 17. Interfaces módulo *Gestor de base de datos*. 68
Tabla 18. Parámetros línea de comandos módulo DarkSearch. 71
Tabla 19. Interfaces módulo Auxiliar. 73
Tabla 20. Funciones módulo Configuración. 74
Tabla 21. Funciones módulo DarkFinder. 76
Tabla 22. Funciones módulo Crawler. 79
Tabla 23. Funciones módulo Log. 80
Tabla 24. Funciones módulo Gestor de base de datos. 81
Tabla 25. Funciones módulo DarkSearch. 84
Tabla 26. Resultados análisis estático de código fuente. 85
Tabla 27. Librerías externas empleadas. 86

Tabla 28. Requisitos de la aplicación. 103

Tabla 29. STRIDE. 109

Tabla 30. Criterios de categorización de amenazas según el método DREAD. 110

Tabla 31. Criterios de categorización de la amenaza en función del riesgo. 111

Tabla 32. Listado de amenazas contempladas en el modelado de amenazas. 111

Tabla 33. Distribución de tipos de amenazas en el modelado. 113

Tabla 34. Medidas de seguridad para mitigación de amenazas. 114

Tabla 35. Vulnerabilidades CWE y patrones de ataque CAPEC asociados a amenazas. .. 114

Tabla 36. Criterios de categorización de activos de información y servicio en el análisis de riesgos. 121

Tabla 37. Criterios de evaluación de probabilidad en el análisis de riesgos. 122

Tabla 38. Criterios de aceptación del riesgo. 123

Tabla 39. DAFO con el análisis del contexto externo e interno. 124

Tabla 40. Impacto considerado en los activos esenciales por cada dimensión. 124

Tabla 41. Distribución de riesgos arquitectónicos residuales. 124

Tabla 42. Activos identificados en el análisis de riesgos por capa TOGAF. 125

Tabla 43. Valoración del impacto en los activos esenciales en el análisis de riesgos arquitectónico. 127

Tabla 44. Listado de amenazas contempladas en el análisis de riesgos arquitectónico. 128

Tabla 45. Medidas de seguridad para mitigación de riesgos arquitectónicos. 141

Tabla 46. Atributos Clase *Configuration*. 142

Tabla 47. Métodos Clase *Configuration*. 143

Tabla 48. Atributos Clase *Page* 145

Tabla 49. Métodos Clase *Page*. 146

Tabla 50. Atributos Clase *Image*. 146

Tabla 51. Métodos Clase *Image*. 147

Tabla 52. Precisión de los algoritmos de rastreo. 183

Tabla 53. Descripción de ficheros de código fuente. 192

1. INTRODUCCIÓN

Internet es un conjunto de redes de comunicaciones que no fue diseñado pensando en la privacidad y anonimidad de los usuarios, de modo que los intercambios de información pueden ser trazables. La Dark Web posibilita anonimidad y privacidad a los usuarios, pero esto a su vez abre la puerta a servicios TCP que se pueden proporcionar de manera anónima: son los llamados servicios ocultos o *hidden services* (Spitters et al., 2014). De este modo, esta red **sirve de refugio a diversos tipos de ciberdelitos**, como tráfico de drogas, humanos, revelación de información confidencial, contenidos relacionados con abusos de menores, fraudes, tráfico de armas, contratación de sicarios, visualización de torturas, difusión de contenidos íntimos no consentidos, y un largo etcétera (Kaur & Randhawa, 2020). De hecho, un 57% de las actividades criminales y contenidos ilegales están en la Dark Web (Nazah et al., 2020). Es tal el desarrollo de estos ciberdelitos, que en este entorno es incluso posible la contratación de Ciberataques como Servicio (CaaS-*Cyberattacks as a Service*) (Huang et al., 2019), habiéndose notado un incremento reciente en este fenómeno en la Dark Web (Europol, 2021b).

La pandemia derivada de la **COVID-19 no ha hecho sino aumentar este fenómeno delictivo**, produciéndose una subida de un 7% del porcentaje de material delictivo en la Dark Web, incrementándose las brechas de datos, pérdidas financieras y fraudes en esta red (Razaque et al., 2021).

En el ámbito de los abusos sexuales infantiles, el desarrollo de tecnologías online ha **facilitado la distribución y consumo de material de abuso sexual infantil** sobre Internet (Brown & Bricknell, 2018), alarmando sobre las estadísticas de este tipo de contenido. (Bissias et al., 2016) realizan un estudio de cinco redes Peer-to-Peer, estimando que en un mes aproximadamente 3 de cada 10.000 usuarios de Internet distribuían material de este tipo de contenido. Puede concluirse que la Dark Web sigue siendo una importante plataforma para el intercambio de este material (Europol, 2021b).

La Dark Web proporciona datos relacionados con terrorismo, como manuales, imágenes, entradas en foros, grabaciones de audio, etc. Los delincuentes son cada vez más desconfiados en la prestación de servicios desde la Dark Web, evitando ser descubiertos por representantes de la ley. Para ello, introducen múltiples filtros y pruebas para poder acceder libremente a sus contenidos (Franco, 2021).

Las investigaciones manuales, que son empleadas frecuentemente para la persecución de delitos, son costosas en tiempo y altamente ineficientes (Zhou et al., 2005) (Zulkarnine et al., 2016). Es por ello que los investigadores usan *crawlers* o rastreadores para detectar

actividades delictivas (Alayda et al., 2021). En el ámbito de la explotación sexual se necesitan soluciones técnicas para incrementar la identificación de las víctimas (Europol, 2021b).

El comercio ilícito en mercados de la Dark Web es un área dinámica sujeto a cambios rápidos, ya que dichos **mercados aparecen y desparecen**. Esto es debido a que en los últimos años se han desarrollado diversas acciones para la retirada de mercados en la Dark Web, así como que los promotores de estos *markets* cierran súbitamente para desaparecer con el dinero captado en ventas que no han sido servidas (Mathur et al., 2020). De este modo, la Dark Web no es estable y es tendente a cambiar rápidamente, existiendo diferentes sitios web que se crean y destruyen cada mes. (Owenson & Savage, 2015) realizan un estudio en el que sólo el 15% de los servicios ocultos analizados en la red TOR permanecían a largo plazo. Los portales de servicios delictivos en la Dark Web tienen ciclos de vida muy cortos, de sólo 200 días aproximadamente (Franco, 2021). Por otro lado, si bien los volúmenes de ventas de droga en la Dark Web es actualmente modesta, tiene potencial de crecimiento.

Existe una **necesidad de incrementar y desarrollar la capacidad de monitorización de la Dark Web** (Kaur & Randhawa, 2020), siendo necesaria la monitorización de actividades de mercados que empleen **otras lenguas diferentes del inglés** (Europol, 2021a). Las autoridades **requieren técnicas y herramientas automatizadas** para monitorizar los servicios ocultos (*hidden services*) en la Dark Web (Al-Nabki et al., 2019).

Por otro lado, el uso de información de la Dark Web es de gran interés hacia una ciberseguridad enfocada a la amenaza, siendo la continua evolución y automatización un proceso clave (Shakarian, 2018). La **investigación en la Dark Web es un aspecto esencial en la lucha contra el cibercrimen** (Basheer & Alkhatib, 2021). En relación a las **técnicas para localizar ciberdelitos en la Dark Web** (Nazah et al., 2020) destacan:

- Seguimiento por parte de los **organismos encargados de hacer cumplir la ley, si bien habitualmente no tienen los conocimientos técnicos** ni experiencia para luchar contra el delito.
- Búsqueda a través de **redes sociales**.
- **Herramientas específicas** de detección del delito como Memex.
- Seguimiento de **flujos monetarios de bitcoins**.
- **Técnicas y métodos de detección de crímenes** como análisis de valor hash, análisis de informante y cuentas de usuario títere o *sock puppets* (falsa identidad online o disponibilidad por una persona de múltiples identidades), metodologías de análisis de red, rastreo de *marketplaces* en la Dark Web, monitorización de la Dark web, despliegue de *honeypots*, implementación de métodos de detección de anomalías o intrusión.

(Décary-Hétu & Aldridge, 2015) describen tres **técnicas para recuperar información online** en tiempo real: (i) *Mirroring*, que toma una imagen estática de un recurso como sitios web o foros, (ii) Monitorización, que implica una observación continua de recursos como sitios web, foros, mercados y chats y (iii) Fugas, mediante la descarga de datos publicados online por delincuentes (por ejemplo mediante *doxing,* en la que un ciberdelincuente publica datos privados de un competidor, impactando en su reputación) o dejados por ellos de manera involuntaria. Este trabajo se centra en la primera de estas opciones.

1.1. Motivación

La motivación principal para el desarrollo de este estudio es **dotar a los Fuerzas y Cuerpos de Seguridad del Estado, así como a las autoridades judiciales y a los investigadores,** de herramientas que permitan realizar una **búsqueda automatizada y efectiva de patrones delictivos en Internet, tanto en la red visible como en la Dark Web**.

Existen múltiples estudios realizados hasta la fecha en relación al desarrollo de rastreadores o *crawlers* tanto en la web visible como en la Dark Web. El propósito de este estudio es el **desarrollo de una plataforma modular y multi idioma** que permita aglutinar e incorporar de manera gradual las diversas funcionalidades y especificaciones que ofrecen cada una de ellas de manera aislada, con el objetivo de detectar patrones delictivos tanto en la web superficial como en las diversas Dark Nets (TOR, I2P, Freenet), introducidas en el capítulo “2.2 Principales Dark Nets”.

1.2. Planteamiento del trabajo

El trabajo se plantea como el diseño y desarrollo de un *crawler* multiplaforma, ejecutable en sistemas Windows y Linux y que permita **recorrer sitios web tanto de la red visible como de diversas Dark Nets, como Tor, I2P y Freenet**. Su diseño permite alternar enlaces entre las diferentes redes, permitiendo seguir por ejemplo un enlace de una página de la web visible a la red TOR.

Se realiza un **diseño modular**, en el que se puedan ir incorporando funcionalidades de manera gradual sin afectar al resto del programa. Asimismo, se contempla que, con carácter general, **la mayor parte de aspectos relacionados con el funcionamiento sean configurables**.

Partiendo de **múltiples direcciones semilla**, el *crawler* irá recogiendo para cada web analizada toda la información recogida en la misma, **tanto la correspondiente al texto como a imágenes**.

La herramienta puede **trabajar de manera distribuida**, de modo que los resultados se van almacenando en tiempo real en una base de datos. Con el objeto de mejorar la eficiencia, el

programa comprueba si la página a rastrear ya ha sido registrada con anterioridad en la base de datos.

Una de las principales aportaciones del rastreador es que permite **el análisis de información de las imágenes** asociadas a sitios web, **registrando datos asociados a su hash y a los metadatos** que albergan. En este sentido, se debe tener en cuenta que, tal y como establece el artículo 189.5 de (Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal), se castiga con pena de prisión "*El que para su propio uso adquiera o posea pornografía infantil o en cuya elaboración se hubieran utilizado personas con discapacidad necesitadas de especial protección*". Por este motivo, el software debe evitar almacenar en disco duro o base de datos el conjunto de imágenes y fotografías analizadas por si pudieran ser constitutivas de delito, por ejemplo, por estar relacionadas con delitos de abusos sexuales infantiles.

Por último, el software permite realizar **búsqueda de patrones en la información almacenada**, de cara a poder obtener direcciones web en las que se almacenan, por ejemplo, información asociada a un determinado *nickname* de usuario o dirección de correo electrónico o imágenes con un hash determinado o metadatos que coinciden con alguno de los patrones de búsqueda especificados.

1.3. Estructura de la memoria

La memoria se estructura en cuadro apartados básicos.

En primer lugar, se realiza un estudio del contexto y estado del arte, analizando la **estructura de Internet** y diferenciando la web visible de las webs profunda (Deep Web) y oscura (Dark Web). En este punto se introducen las **diferentes Dark Nets** para las que funciona el rastreador (TOR, I2P, Freenet), así como **aspectos relevantes de los *crawlers*** o rastreadores, como su definición y modo de funcionamiento, tipologías, desafíos y métricas asociadas a los mismos. Tras resumir la principal **información que se puede asociar a las imágenes**, se recoge un listado de **trabajos relacionados** con el presente desarrollo.

En un segundo bloque se identifican los **objetivos y metodología** del trabajo, diferenciando el objetivo principal del proyecto, así como los secundarios derivados de él. Asimilismo, se introduce la metodología contemplada.

A continuación, se recoge el **desarrollo de la propuesta**, desarrollando el **diseño de la arquitectura realizada** y los detalles de la **implementación software** llevada a cabo.

Finalmente, se resumen las **principales conclusiones** del presente Trabajo Fin de Master (en adelante TFM) y se sugieren **nuevas líneas de trabajo** para desarrollar aquellos aspectos que se hayan quedado sin cubrir en el mismo.

2. CONTEXTO Y ESTADO DEL ARTE

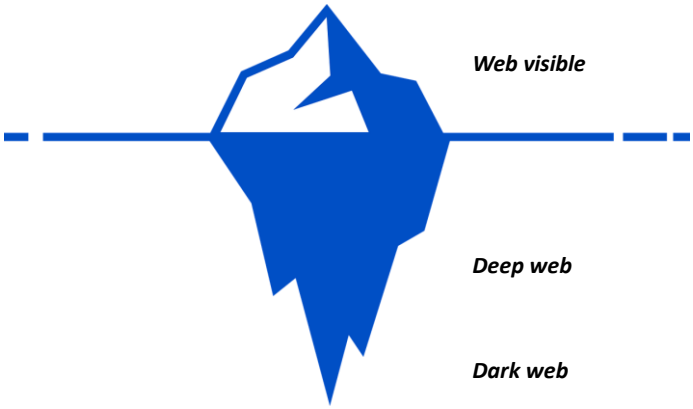
Se introduce en este capítulo información relevante para el desarrollo del *crawler* objeto del presente TFM por medio del estudio de:

- Estructura de Internet.
- Principales Dark Nets.
- *Crawlers*.
- Información asociada a las imágenes.
- Trabajos relacionados.

2.1. Estructura de Internet

Internet está compuesta por tres partes (Kaur & Randhawa, 2020), tal y como se recoge en la Figura 1:

Figura 1. Estructura de Internet.



Fuente: (Kaur & Randhawa, 2020).

- Web visible.
- Web profunda (Deep Web).
- Web oscura (Dark Web).

2.1.1. Web visible

Está compuesta por todo el **contenido que es accesible a través de motores de búsqueda** web convencionales (como Google, Bing o Yahoo). Se estima que sólo el 0,03% de los resultados se obtienen a través de estos buscadores (Kaur & Randhawa, 2020). En este entorno se definió el concepto de motores de búsqueda, diseñados para realizar exploraciones basadas en palabras específicas incluidas en sitios web (Franco, 2021). Google introdujo un cambio de paradigma en este concepto de buscador, al presentar los resultados

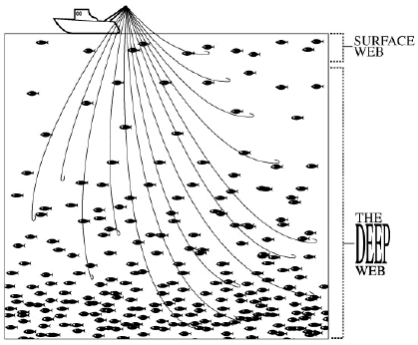
en base a un ranking de las páginas encontradas basado en análisis y clasificación de consultas con metodologías de inteligencia artificial.

2.1.2. Web profunda (Deep Web)

La red profunda (Deep Web, Hidden Web o Invisible Web) es la parte de la red compuesta por el **contenido que no está indexado por los motores de búsqueda** (Sherman & Price, 2001). Entre otros, incluye contenidos que requieren autenticación para el acceso, correo electrónico, páginas dinámicas, bases de datos, etc. El término fue acuñado en 1994 por Jill Ellsworth (Bergman, 2001).

Los mecanismos de búsqueda tradicionales en la web visible crean sus índices en las páginas superficiales, lo que hace que la indexación que realizan es meramente superficial, según muestra la Figura 2. Para explicarlo, (Bergman, 2001) establece una comparativa con la pesca que puede realizar un barco en el océano, que sólo puede recoger peces que estén en los niveles más superficiales con las técnicas tradicionales, debiendo emplear otras para zonas más profundas.

Figura 2. Indexación de buscadores en la Deep Web.



Fuente: (Bergman, 2001).

Un uso legal e interesante de la Deep Web es la búsqueda de información en diversas bases de datos en Internet (Hawkins, 2016).

Existen diversas casuísticas que pueden causar problemas a un *crawler* para obtener información en la Deep Web (Sherman & Price, 2001), como: (i) Páginas que contienen formularios para realizar búsquedas, (ii) Páginas generadas dinámicamente, (iii) Páginas sin contenido HTML, (iv) Sitios que ofrecen información en tiempo real, (v) Ficheros PDF o Postscript, (v) Páginas que proporcionan acceso a una base de datos, siendo una combinación de los casos (i) y (ii).

2.1.3. Web oscura (Dark Web)

Se trata de la **parte de web que se destina a ser anónima** (Winkler & Gomes, 2016), representando un grupo de páginas web a las que **sólo se puede acceder usando**

navegadores específicos. Incluye contenidos que no son accesibles a través de navegadores convencionales, incorporando características de anonimidad y privacidad. Está compuesta de sub redes anónimas llamadas Dark Nets. En esta parte de Internet es donde se alojan principalmente **contenidos ilícitos**.

En las primeras estimaciones, el tamaño de la Deep Web se estimaba en unas 500 veces superior al correspondiente a la web visible, siendo más de un 95% de su contenido accesible públicamente sin necesidad de suscripción (Bergman, 2001). Más recientemente, se estima que el 96% del contenido está formado por la Deep y Dark Web (Kaur & Randhawa, 2020), siendo en torno al 6% el contenido de la Dark Web (Vignoli & Monteiro, 2020), estando metafóricamente en la parte inferior de la Deep Web.

Una de las primeras actividades ilegales y con más notoriedad en la Dark Web fue el sitio *Silk Road,* que comenzó operando en 2011 como un mercado ilegal (*dark market*) de drogas ilícitas (Van Hout & Bingham, 2013) y que fue desmantelado por el FBI en octubre de 2013. Como ejemplos de uso de la Dark Web se pueden diferenciar los relativos al tráfico de drogas, armas, contenidos relacionados con abusos de menores, venta de documentos de identidad/pasaportes o licencias de conducir falsas, venta de tarjetas de crédito robadas, tráfico de personas, contratación de sicarios, terrorismo, herramientas para el cibercrimen, comercialización de datos robados, intercambios económicos usando Bitcoin o Monero (notándose un incremento en el uso de esta criptomoneda), falsificación de moneda, tráfico de órganos humanos, etcétera (Nazah et al., 2020) (Kaur & Randhawa, 2020) (Europol, 2021b) (Ilić & Spalević, 2017).

Existen multitud de *markets* o portales en la Dark Web, como *Agora*, que posibilita la obtención ilegal de elementos como drogas, identificadores falsos y armas y que fue analizado en (Baravalle et al., 2016) o "*Red Room*", donde se muestran torturas y asesinatos de personas mediante retransmisión en vivo (Ilić & Spalević, 2017).

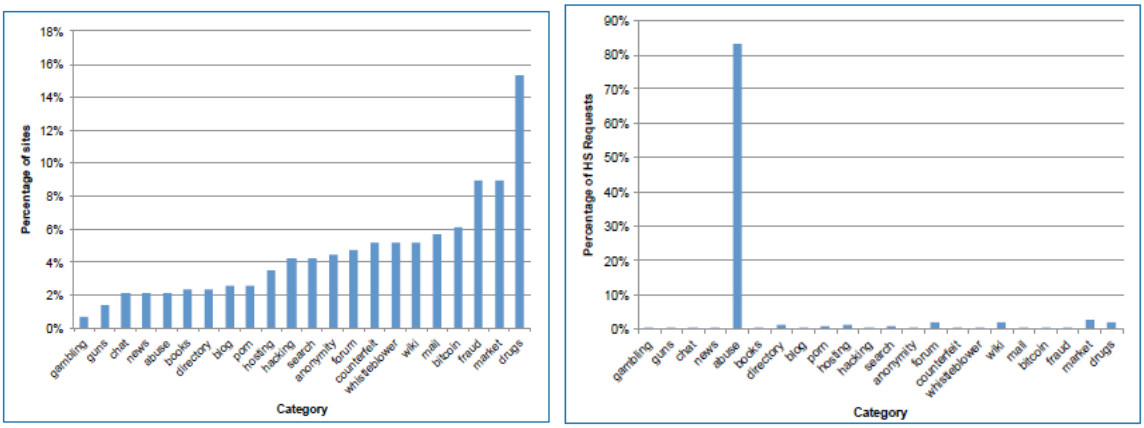
Es habitual que las transacciones monetarias asociadas a estas actividades se realicen mediante criptomonedas, como por ejemplo Bitcoin. Si bien la mayor parte de estos *markets* en la Dark Web están en inglés, han proliferado sitios con otros idiomas, como ruso, chino, sueco, finlandés, etc.

No obstante, debe tenerse en cuenta que **en modo alguno todo el contenido de esta parte de la web es constitutivo de delito**, ya que es empleado como un mecanismo efectivo para proteger la identidad de los usuarios. (Jardine et al., 2020) indican que menos del 7% de los usuarios globales emplean la red TOR para usos maliciosos, si bien el porcentaje varía en función del país y del contexto. (Al-Nabki et al., 2019) cuantifican en un 20% las direcciones TOR consideradas sospechosas. La Dark Web no aumenta de forma natural las actividades

delictivas, simplemente puede considerarse como herramienta que utilizan los ciberdelincuentes para llevar a cabo actividades ilícitas. Por otro lado, también incluye otros usos, como expresar creencias sociales o políticas con libertad en países en los que estas actuaciones están reprimidas, siendo utilizadas por periodistas, activistas y denunciantes (Mirea et al., 2019).

La Figura 3 muestra en su parte izquierda una clasificación por categorías de sitios en la Dark Net TOR, siendo los contenidos relacionados con drogas los más frecuentes. Sin embargo, según se comprueba en la parte derecha de la citada figura, los sitios con contenido relacionado con abuso infantil son los que recibieron mayor cantidad de peticiones.

Figura 3. Clasificación por categorías de sitios web en la red TOR.



Fuente: (Owenson & Savage, 2015).

Existen centros de información que proveen información sobre *markets* en la Dark Web y servicios ocultos, proporcionando direcciones .onion que pueden ser utilizadas para identificar *markets* activos.

(Biryukov et al., 2014) realizan un estudio de servicios ocultos en la red TOR, aprovechando fallos en el protocolo y la implementación de la red. Concluyen que las direcciones .onion más populares corresponden a centros de comando y control de *botnets* y recursos proporcionando contenidos para adultos.

Debe tenerse en cuenta que durante el escaneo de sitios que no sean confiables existe un riesgo alto de procesar información **cuyo almacenamiento no es recomendable o incluso es ilegal** (Razaque et al., 2021).

(Kaur & Randhawa, 2020) describen diferentes **mecanismos de ataque** y defensa en la Dark Web, incluyendo (i) Ataques de correlación, (ii) Ataques de congestión, (iii) Ataques de correlación de tiempo y tráfico, (iv) Ataques de análisis de tráfico, (v) Ataques de denegación de servicio distribuido, (vi) Ataques a servicios ocultos, (vii) Phishing, (viii) Ataques de captura de sesión (*session hijacking*) y hombre en el medio (*Man In The Middle*), (ix) *Cross-Site*

Scripting, (x) *SQL Injection* y (xi) Reutilización de credenciales. (Catakoglu et al., 2017) despliegan un *honeypot* de alta interacción en la red TOR para explorar el modus operandi de los atacantes en la Dark Web. Mediante ataques de análisis y correlación de tráfico es posible conocer si un usuario accede a un determinado sitio web sin necesidad de acceder a su contenido encriptado ni controlar ningún nodo de la red TOR (Owenson & Savage, 2015).

Existen **diversos buscadores que permiten acceder a sitios web alojados en la Dark Web**, como por ejemplo Ahmia, Torch, Haystak u OnionLand Search. La volatilidad de los sitios web de esta red no permite proporcionar una infraestructura de búsqueda fiable, siendo las *Hidden Wiki* o listas de servicios ocultos la forma más destacada de obtener puntos de acceso a la Dark Web (Iliou et al., 2017).

2.2. Principales Dark Nets

Una red que utiliza técnicas para mantener la privacidad y anonimato de los usuarios se denomina Dark Net. Con la combinación de todas estas redes oscuras diferentes, nació la Dark Web. Se introducen a continuación las principales Dark Nets, como son TOR, I2P y Freenet.

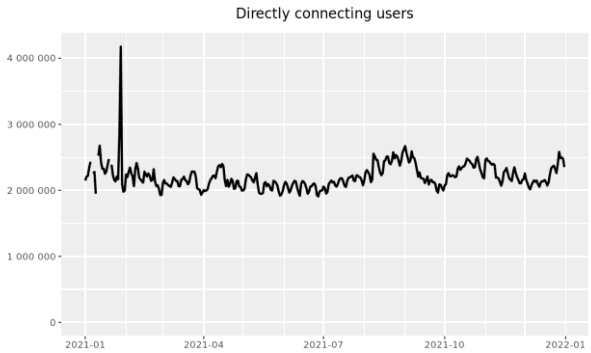
2.2.1. TOR

David Goldschlag, Mike Reed, and Paul Syverson, dentro del Laboratorio de Investigación Naval de los Estados Unidos (en adelante NRL, *Naval Research Laboratory*) desarrollaron en 1995 los primeros diseños y prototipos del *onion routiung* que dieron lugar en 2002 a la primera versión de *The Onion Routing* (en adelante TOR), proporcionada bajo licencia de software abierto y libre y que funcionaba como **una red descentralizada de nodos que mantenían la privacidad de los intervinientes en la comunicación** (The Tor Project, Inc., s. f.-c).

De este modo, se trata de una **herramienta que proporciona anonimato y privacidad**. **Impide que los sitios web o un observador externo identifiquen al usuario que está realizando la visita**. Otra característica relevante de esta web son los servicios ocultos (*hidden services*), entendidos como la capacidad para alojar un sitio web o servicio de Internet de forma anónima, de modo que tanto el visitante como el sitio son anónimos. Con esta característica, también se permite la posibilidad de alojar material de orientación criminal con un grado de impunidad (Owenson & Savage, 2015).

La Figura 4 muestra el número de usuarios conectados diariamente a la red TOR a lo largo del año 2021, comprobando que **supera con carácter general los 2 millones**.

Figura 4. Número de usuarios conectados directamente a la red TOR.



Fuente: (The Tor Project, Inc., s. f.-e).

La Tabla 1 recoge un análisis no secuencial de servicios ocultos populares en la red TOR, siendo un sitio de abuso infantil el que más visitas recibió.

Tabla 1. Análisis de servicios ocultos populares en la red TOR.

Dirección servicio	Peticiones / día	Días observado	Descripción
censored	168.152	12	Abuso infantil
silkroad6ownowfk	8.067	11	Silk Road (market)
agorabasakxmewww	3.035	8	Agora (market)
k5zq47j6wd3wdvjq	2.589	5	Evolution
xmh57jrzrnw6insl	1.341	7	Torch
3g2upl4pq6kufc4m	1.223	4	DuckDuckGo
wikitjerrta4ggz4	555	12	HiddenWiki
mail2tor2zyjdctd	266	8	Correo electrónico

Fuente: (Owenson & Savage, 2015).

Esta Dark Net utiliza el método del *onion routing*, en el que la información se encripta y transfiere, **existiendo siempre tres (3) nodos** que participan en la comunicación:

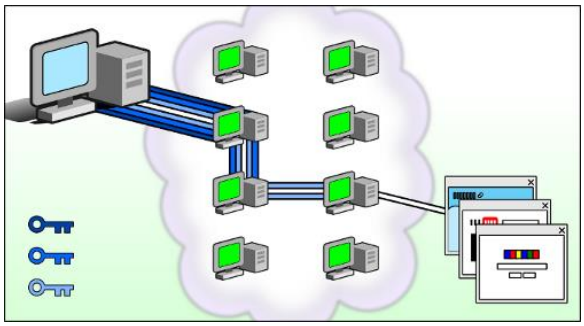
- **Nodo de entrada (*guard node*)**, que recibe el tráfico del usuario, encaminándolo al nodo intermedio. El navegador TOR dispone de un listado de las autoridades de directorio que incluirán el nodo de entrada que se seleccione para establecer la comunicación. Este nodo es el único que conoce la dirección IP real del emisor.
- **Nodo intermedio (*middle node*)**, que encamina la información de un nodo al siguiente.
- **Nodo de salida (*exit node*)**, que entrega el tráfico al destino. Es el único punto de la red TOR que interactúa con él y conoce su dirección IP.

Como punto de partida para la búsqueda de información en la red TOR existen sitios como *The Hidden Wiki*, que con una estructura análoga a Wikipedia, contiene links a diferentes

contenidos de esta red. Asimismo, existen motores de búsqueda que proporcionan enlaces a otros contenidos (tanto lícitos como ilegales o maliciosos) de la Dark Web (Nazah et al., 2020), tal y como se recoge en el capítulo "2.1.3 Web oscura (Dark Web)".

Para la comunicación se establece un circuito virtual que dura unos 10 minutos haciendo uso de tres nodos elegidos aleatoriamente. **La petición se encripta en múltiples capas que van siendo desencriptadas por cada nodo** según el mensaje va avanzando en la red. En el **nodo de salida se desencripta el último mensaje y se entrega al destino sin cifrar**. La respuesta viaja por el mismo camino establecido para la petición.

Figura 5. Establecimiento de circuitos virtuales en TOR.



Fuente: (The Tor Project, Inc., s. f.-a).

En la actualidad, para el acceso a servicios *onion* en la red TOR se establecen rutas virtuales que se componen de seis nodos de tránsito a través de un punto de encuentro (*rendezvous point*) (The Tor Project, Inc., s. f.-d), gestionando de este modo un canal más robusto desde el punto de vista de seguridad, pero más lento al no ser una comunicación directa y con nodos intermedios. El mecanismo de *onion routing* asegura que cada nodo sólo conoce la dirección IP de los nodos adyacentes, desconociendo las correspondientes al resto de actores de la comunicación.

Para acceder a la red se debe instalar un navegador TOR disponible en (The Tor Project, Inc., s. f.-b). También es posible el acceso mediante un navegador convencional a través del servicio Tor2Web que permite navegar de manera anónima sin necesidad de instalar el navegador TOR. Asimismo, es posible acceder a través de móvil mediante la aplicación Orbot, disponible en Google Play.

2.2.2. I2P

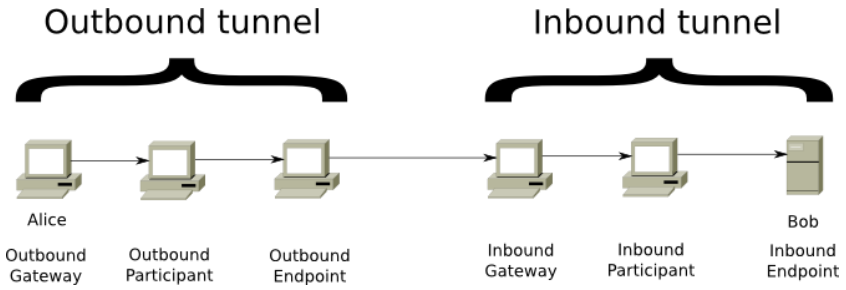
I2P (*Invisible Internet Project*) es un proyecto similar a TOR, en el que la red cuenta con centenares de *routers* y **las comunicaciones entre ellos están cifradas** (Gulyás, 2020).

Para el establecimiento de las comunicaciones se hace uso de túneles de salida (*Outbound tunnel*) o de entrada (*Inbound tunnel*). El emisor establece un *outbound tunnel*, mientras que el receptor hace lo propio con un *inbound tunnel*. El emisor envía información cifrada al

extremo del *outbound tunnel* para que pueda redireccionar al siguiente nodo perteneciente al *inbound tunnel*.

A diferencia de la red TOR, la **petición y la respuesta viajan por caminos distintos**.

Figura 6. Establecimiento de túneles en red I2P.



Fuente: (*I2P: A scalable framework for anonymous communication - I2P*, s. f.).

En esta red se hace uso de mensajería tipo *garlic*, que permite el envío conjunto de más de un mensaje en un mismo paquete, **incrementando la tasa de transferencia de datos y haciendo el análisis de tráfico más complejo**.

2.2.3. Freenet

Se trata de software que permite el **intercambio de ficheros anónimos, navegación y publicación de los llamados "*freesites*", accesibles únicamente a través de esta red**. La comunicación es cifrada y enrutada. Al usar Freenet, los usuarios contribuyen proporcionando ancho de banda y una porción de su disco duro para un almacenamiento distribuido de la información. Los *freesites* son páginas HTML, no soportándose Javascript o scripting del lado de servidor.

Freenet tiene sus orígenes en la tesis realizada en la Universidad de Edimburgo por Ian Clarke donde creó un sistema de almacenamiento y recuperación de información distribuido y descentralizado (Beckett, 2009).

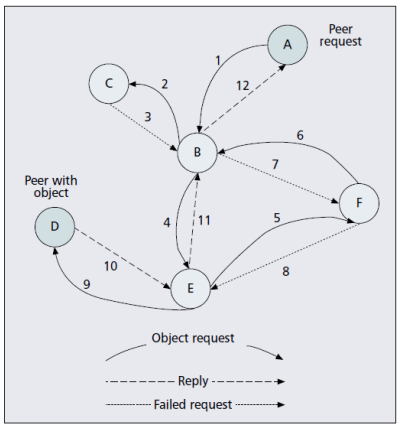
La red dispone de una funcionalidad conocida como "*darknet"* o modo *friend-to-friend* en la que los usuarios sólo se conectan con otros a los que conocen y, a través de estos contactos, se conectan con otros usuarios en los que confían. Este modo es muy difícil de detectar. Por el contrario, existe un modo abierto (*opennet*) en el que los usuarios se pueden conectar automáticamente a cualquier otro destinatario, siendo un sistema mucho menos seguro (The Freenet Project Inc., s. f.).

Las peticiones de los usuarios se pasan de nodo a nodo a través de una cadena de peticiones proxy. En cada nodo se decide cuál es el siguiente al que se envía la petición. Cada nodo sólo tiene conocimiento de los vecinos anterior y posterior en la cadena de proxy, para mantener

la privacidad. Ningún nodo tiene privilegios o diferencia sobre los demás, con lo que no existe jerarquía o punto central de fallo.

Para acceder a los ficheros en Freenet se requiere una clave.

Figura 7. Secuencia de petición típica en Freenet.



Fuente: (Lua et al., 2005).

2.3. Crawlers

Se define y describe a continuación el concepto y funcionamiento de los *crawler* o rastreadores, identificando diversos trabajos y desarrollos anteriores relativos a estos programas. Asimismo, con el objeto de obtener una mejor comprensión y facilitar la especificación de los requisitos de la aplicación software, se analizan los diversos tipos de crawlers y desafíos que presentan según la diversa literatura existente, así como métricas para poder determinar su eficacia de funcionamiento.

2.3.1. Definición y modo de funcionamiento

Un ***crawler* web,** también conocido como rastreador, *spider* (araña) o robot consiste en **un proceso automatizado de recolección de páginas web** que tiene diversas aplicaciones como entre otros, motores de búsqueda, minería de datos, análisis de medios sociales, detección de spam, sitios fraudulentos o detección de actividades ilegales (Shestakov, 2013).

Los *crawlers* funcionan **identificando los links diversos de una página web y accediendo a los mismos de manera recursiva**, pudiendo ser utilizados asimismo para extraer contenido. Parten de una URL conocida como semilla y recorren las diversas páginas web, las cuales almacenan, para que posteriormente puedan ser recuperadas y analizadas. Con el objeto de **realizar una búsqueda más eficiente, se controla las páginas que ya han sido visitadas**, finalizando el proceso cuando ya no quedan más páginas que visitar, bien porque se ha llegado a un número determinado de páginas o alcanzado una profundidad máxima de rastreo.

Tras el proceso de rastreo (*crawling*) se hace necesario un **proceso de raspado (*scraping*) para extraer datos del fichero HTML** (Ball et al., 2021).

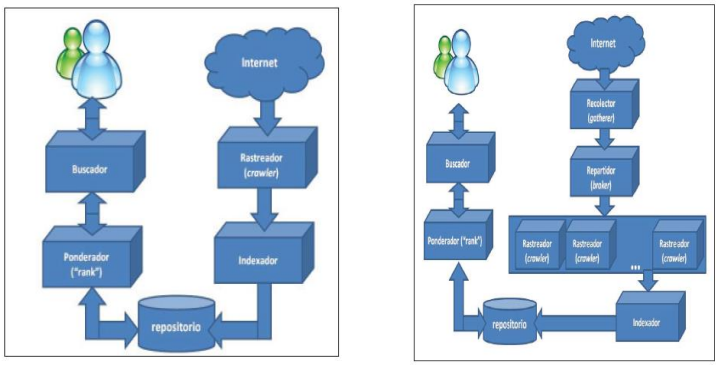
(Brin & Page, 1998) describen el *crawler* que da lugar a Google, el buscador más popular. Estructuran un buscador web en diversos componentes, como un web *crawler* (descarga la información disponible en la web), un indexador (genera un índice de términos e información relevante), un ponderador (ordena las entradas del indexador), un motor de búsqueda (procesa la consulta de usuario) y un repositorio de páginas (aloja una versión de las páginas rastreadas). Se introduce el concepto de *PageRank* como medio para calcular una clasificación de la calidad de las webs. (Cho et al., 1998) definen métricas que regularon el fundamento del algoritmo de Google durante los primeros años, incluyendo métricas como la similaridad de página (medida para descartar páginas iguales en la web), conteo regresivo (para aumentar la relevancia de una página según esté más referenciada por otras), *PageRank* (para ponderar mejor las páginas referenciadas por portales importantes), completitud de la información y ubicación (según el lugar del código en el que se encuentre un resultado).

El administrador de una web puede evitar que el *crawler* recorra determinadas partes del sitio, como puede ser mediante la restricción por usuario-contraseña o la creación de un fichero *robots.txt*, que es un fichero opcional, creado en el directorio raíz, donde se indican los ficheros que se solicitan no sean analizados por los rastreadores. No siendo de obligado cumplimiento, la mayor parte de los motores de búsqueda lo respetan (Sherman & Price, 2001).

Con carácter general, los sistemas de búsqueda de información web se basan en las siguientes arquitecturas (Camargo Sarmiento & Ordóñez Salinas, 2013) (Brin & Page, 1998), recogidas en la Figura 8:

- **Centralizada**: compuesta por *crawler*, indexador, ponderador, motor de búsqueda y repositorio de páginas. Se utilizan en entornos con capacidades limitadas.
- **Distribuida**: versión mejorada de la anterior, que puede ser alojada en diferentes servidores. Cuenta con dos elementos adicionales: (i) *Gatherers*, que extraen la información a recopilar periódicamente y (ii) *Brokers*, encargados de indexar la información recopilada por los *gatherers* y otros *brokers*, permitiendo compartir el trabajo y limitar la transmisión de la información.

Figura 8. Arquitecturas centralizada y distribuida.



Arquitectura centralizada Arquitectura distribuida

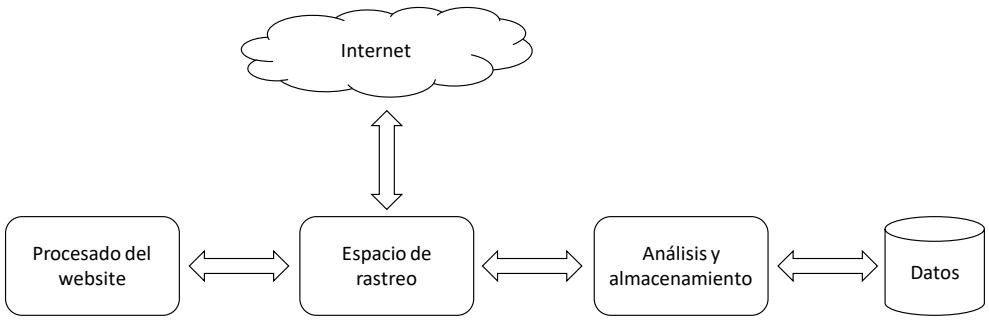
Fuente: (Camargo Sarmiento & Ordóñez Salinas, 2013).

Finalmente, para poder hacer uso eficaz de la información recopilada es **necesario clasificarla y adaptarla**, siendo estas tareas propias del Procesamiento de Lenguaje Natural (PLN). Debe determinarse qué contenido es relevante en base a uno o varios criterios (Camargo Sarmiento & Ordóñez Salinas, 2013).

(Alkhatib & Basheer, 2019a) introducen los componentes necesarios de un crawler:

- **Espacio de rastreo**: dominio de información a ser recuperada.
- **Procesado del website**: registro, validación, cookies de sesión, etc.
- **Análisis y almacenamiento**: procesado y almacenamiento de las páginas HTML.

Figura 9. Estructura de crawler.



Fuente: Elaboración propia a partir de (Alkhatib & Basheer, 2019a).

Los *crawlers* deben presentar algunas características como (Shkapenyuk & Suel, 2002):

- **Flexibilidad**, para poder ser utilizados en diversos escenarios.
- **Bajo coste y alto rendimiento**, siendo escalables y haciendo un uso eficiente de los accesos a disco.

- **Robustez,** debiendo ser capaces de tratar con HTML mal formados, así como cualquier otro aspecto relacionado con la mala configuración de los servidores. Asimismo, como un rastreo puede llevar semanas o meses, el sistema debe ser capaz de tolerar caídas e interrupciones de la red sin perder demasiada información.
- **Etiqueta y control de velocidad,** para no afectar negativamente a los servidores rastreados.
- **Gestión y configuración,** proporcionando un interfaz para poder monitorizar el rendimiento del *crawler*, así como configurar diversos aspectos de su funcionamiento.

Los *crawlers* destinados al análisis de *markets* operan en tres pasos: realizan un rastreo por página (*page crawl*), seguido de un rastreo por productos (*product crawl*) y por último un rastreo por vendedor (*vendor crawl*). En estos casos, la implementación del *crawler* puede estandarizarse, ya que los *markets* son relativamente consistentes (Ball et al., 2021).

(Camargo Sarmiento & Ordóñez Salinas, 2013) introducen la importancia en la evolución de los *crawlers* actuales del Procesamiento del Lenguaje Natural (PLN), disciplina utilizada para analizar el lenguaje hablado o escrito. Debe tenerse en cuenta que existen diferentes modelos para determinar qué contenido es relevante dado uno o varios criterios, por lo que debe determinarse un modelo para el cálculo de la relevancia de los documentos recopilados en relación a la consulta.

2.3.2. Tipos de crawlers

(Chaitra et al., 2019) introducen una clasificación de tipos de *crawler* web:

- **Enfocado/focalizado (*focused*)**: buscan de manera selectiva una web en función de un campo de búsqueda o tema concreto. Al estar focalizados en un tema particular, este tipo de *crawlers* suelen incorporar técnicas de clasificación basados en técnicas de *machine learning* supervisado. Como ejemplos se pueden citar el bayesiano, kNN (*k-Nearest Neighbors*), modelos de asociación semántica, ontologías, siendo los más utilizados por precisión y exactitud los basados en Máquinas de Vector de Soporte (SVM-*Support Vector Machines*) (Camargo Sarmiento & Ordóñez Salinas, 2013).
- **Incremental**: visitan los sitios web y acceden a las páginas actualizadas.
- **Distribuido (*distributed*)**: el rastreo se realiza entre diversos crawlers, existiendo un servidor central que sincroniza los nodos. Puede utilizar un algoritmo como *Page Rank* para incrementar su eficiencia y calidad de búsqueda.
- **Paralelo (*parallel*)**: existen diferentes procesos de *crawling* que descargan las páginas y las almacenan localmente.

- **Escondido (*hidden*)**: orientado a la obtención de información de la Deep Web.

Desde otro punto de vista, los crawlers pueden ser clasificados en función de las estrategias seguidas para el rastreo (Gupta & Bhatia, 2014):

- **Orientados a la extensión (*breadth oriented*)**: enfocados a analizar un conjunto lo más extenso posible de fuentes más que a profundizar en una de ellas en concreto. La estrategia *Breadth-First Search* (BFS) no tiene en cuenta la relevancia de las páginas a la hora de hacer el rastreo, por lo que a esta técnica también se le conoce como algoritmo de búsqueda ciega (*blind search*). Su adopción se basa en el hecho de suponer que páginas de un mismo nivel tienen una alta probabilidad de ser relevantes (Yu et al., 2018).
- **Orientados a la profundidad (*depth oriented*)**: su objetivo es obtener el máximo de información posible de una fuente de datos determinada. Existe importante literatura estudiando este campo. El algoritmo *Depth-First Search* (DFS) puede derivar en un bucle prácticamente infinito, llevando a que sólo se analice un conjunto limitado del total del árbol de páginas web a explorar. Para solventar este problema se puede hacer uso de límite de profundidad.

Adicionalmente, (Yu et al., 2018) introducen otras estrategias como ***Best-First Search (BFS)***, en la que se analiza la similitud entre la URL candidata y la búsqueda a realizar, escogiendo aquellas que proporcionen una mejor puntuación. Requiere identificar un algoritmo de medición de dicha similitud, normalmente un ranking o puntuación, como puede ser el algoritmo *Page Rank*, que proporciona importancia a las páginas en función de las citas o links que otras páginas tienen hacia ella (Pavalam et al., 2011). En muchas ocasiones, se aplica un algoritmo de clasificación como *Naive Bayes*. Se trata de un método heurístico que no garantiza la mejor solución, pero que proporciona información útil para la resolución del problema.

Adicionalmente, para *crawlers* focalizados se pueden considerar otros algoritmos como *Fish Search Algorithm* o *Shark Search Algorithm*, versión mejorada del anterior (Manikandan & Kavitha, 2021).

En relación al almacenamiento de la información pueden considerarse enfoques de **bases de datos no estructuradas** (contienen documentos de texto sin formato que proporcionan una interfaz de búsqueda simple basada en palabras clave tipo texto) o **estructuradas** (ofrecen la posibilidad de búsqueda mediante múltiples atributos relacionados con diferentes aspectos del contenido) (Gupta & Bhatia, 2014). Debe tenerse en cuenta que el uso de sistemas de almacenamiento no estructurados (como bases de datos no SQL) permite una mayor capacidad de procesamiento o capacidad que el empleo de bases de datos estructuradas (SQL) (Ochando, 2014).

La Tabla 2 recoge una comparativa entre los tipos de web *crawler* identificados:

Tabla 2. Comparativa de diferentes tipos de web *crawler*.

Parámetro	*Hidden*	*Distributed*	*Incremental*	*Parallel*	*Focused*
Técnica de búsqueda	DFS	BFS	BFS	BFS	BFS
Reducción capacidad red	-	No	No	Sí	-
Escalabilidad	Sí	No	No	Sí	Sí
Extensibilidad	-	No	Sí	No	-
Solape	-	No	No	Sí	-
Selección páginas	Analizador formularios	De URL semilla	De cola de prioridad	De URL semilla	Según tema específico

Fuente: (Chaitra et al., 2019).

2.3.3. Desafíos de los crawlers

Para el desarrollo de crawlers deben afrontarse diversos retos y dificultades (Alkhatib & Basheer, 2019a) (Ball et al., 2021):

- **Escalabilidad**, dado el gran número de sitios web existentes. Un web *crawler* distribuido puede ser un enfoque para resolver este problema.
- **Obligaciones sociales**, no afectando al servicio de los sitios web que se rastrea.
- **Volatilidad de los sitios web y *markets* en la Dark Web**, estando no operativos gran cantidad de tiempo.
- **Sortear defensas de ataques contra Denegación de Servicio (DoS).** Muchos sitios web implementan servicios de protección anti-DoS, que bloquean al usuario solicitante en caso de que se realicen un número relevante de peticiones en un corto periodo de tiempo. Una de las posibles estrategias de solución sería introducir un periodo aleatorio entre dos peticiones consecutivas.
- **Ausencia de indexado** o catálogo de sitios webs.
- **Tiempo necesario para rastrear un sitio web,** debido a las limitaciones de velocidad de la red TOR y su inestabilidad.
- **Accesibilidad**, requiriendo en muchos casos registro de usuario o resolver retos **CAPTCHA** destinados a evitar accesos para rastreadores automatizados.
- **Búsqueda de palabras clave**: los *crawlers* almacenan palabras clave de las páginas indexadas. La selección adecuada de qué términos se almacenan es fundamental para

poder proporcionar de manera adecuada resultados de búsqueda. Google, por ejemplo, lo consigue por medio de programación avanzada.

Por otro lado, existen **páginas aisladas**, que no son referenciadas por ningún otro sitio web, constituyendo la parte más básica de la web no visible (Sherman & Price, 2001). La Tabla 3 recoge los resultados de un estudio sobre las tipologías de enlaces que tienen diferentes servicios ocultos de la red TOR, diferenciando de si son al propio sitio, a la web visible o a la Dark Web.

Tabla 3. Distribución de tipos de enlaces de *hidden services* en la red TOR.

Tipo de enlace	Porcentaje
Sitios que sólo tiene enlaces al propio sitio web	59%
Sitios que sólo tienen enlaces a sitios de la web visible	23%
Sitios que sólo tienen enlaces a sitios de la Dark Web	7%
Sitios que tienen enlaces tanto a la Dark Web como a la web visible	11%

Fuente: (Owenson & Savage, 2015).

Adicionalmente, los ***crawlers* de la Deep Web tienen algunos retos específicos**, como son (i) La necesidad de contar con un sistema automático, escalable y eficiente que permita la indexación de millones de formularios HTML, (ii) La necesidad de generar valores válidos para dichos formularios, (iii) Gestionar las múltiples combinaciones de los parámetros de entrada de los formularios (ya que constituyen un producto cartesiano), (iv) El rastreo de páginas subsiguientes, (v) La resolución de incidencias relacionadas con Javascript, (vi) La resolución de falta de atributos, tanto a nivel de registro como de página, (vii) La validación de datos y limpieza, (viii) Normalización e (ix) Integración de datos (Madhavan et al., 2008) (Mundluru & Xia, 2008).

Además, deben considerarse medidas de protección, ya que las técnicas de rastreo pueden ser descubiertas, siendo en tal caso vulnerable a ciberataques por parte de los ciberdelincuentes que se analizan en la Dark Web (Basheer & Alkhatib, 2021). Por este motivo debe asegurarse que el objetivo del experimento no perciba que se está realizando, ya que puede modificar su comportamiento (Christin, 2012). La renovación periódica de los circuitos establecidos a través de la red TOR dificulta que se pueda detectar este hecho mediante monitorización de los circuitos y conexiones a través de esta red.

2.3.4. Métricas asociadas a los crawlers

Para medir la fiabilidad de los resultados proporcionados por crawlers focalizados se pueden emplear diferentes métricas (Fu et al., 2010):

- **Exhaustividad (*recall*)**, calculado como la proporción de documentos relevantes recuperados satisfactoriamente para la búsqueda:

$$recall = \frac{\langle documentos\ relevantes \rangle \cap \langle documentos\ recuperados \rangle}{\langle documentos\ relevantes \rangle}$$

- **Precisión (*precision*)**, calculado como la proporción de documentos recuperados que son relevantes para la búsqueda:

$$precision = \frac{\langle documentos\ relevantes \rangle \cap \langle documentos\ recuperados \rangle}{\langle documentos\ recuperados \rangle}$$

- **F-Measure**, calculada como la media armónica de exhaustividad y precisión:

$$F - measure = \frac{2 * recall * precision}{\langle recall + precision \rangle}$$

Existe un balance en un motor de búsqueda entre la velocidad, la precisión y la exhaustividad (Kobayashi & Takeda, 2000).

2.4. Información asociada a las imágenes

La localización asociada a una imagen puede almacenarse explícitamente como información adicional o como metadatos, siendo su formato más común EXIF (*Exchangeable Image File Format*). Las especificaciones EXIF tienen múltiples anomalías, que pueden causar problemas en la extracción de metadatos.

Finalmente, debe tenerse en cuenta que según el estudio realizado por (Fazal et al., 2019), menos del 1% de las imágenes analizadas tenían información geográfica asociada a la imagen.

2.5. Trabajos relacionados

(Camargo Sarmiento & Ordóñez Salinas, 2013) recogen una evolución histórica de los web *crawlers*, incluyendo UbiCrawler, Viuva Negra, Google, Heritrix, Nutch, Combine, WIRE, Wanderer, JumpStation, RBSE, WebCrawler, WWWWorm, MOMspider etc.

Existe literatura diversa en relación a desarrollos de ***crawlers* para la Dark web**. (Zhou et al., 2005) proponen un sistema de gestión de conocimiento para localizar, recopilar, acceder, analizar y gestionar datos de la Dark Web, incluyendo un portal de conocimiento web con capacidad de análisis y soporte multi idioma.

(Fu et al., 2010) desarrollan un *crawler* focalizado para recoger información relativa a foros en la Dark Web que utiliza un enfoque de accesibilidad asistida por humanos para obtener el acceso.

(Paganini, 2013) desarrolla diferentes rastreadores para analizar los tipos de sitios en la red TOR en el contexto del proyecto Artemis, orientado a analizar de forma masiva un conjunto de URLs y servicios ocultos de la red TOR, así como explorar la posibilidad de monitorizar usuarios en dicha red.

(Spitters et al., 2014) utilizan un *crawler* con el objetivo de obtener información relativa a servicios ocultos de la red TOR y poder organizarlos temáticamente. El estudio recoge que en el 83,3% de los servicios ocultos el idioma empleado es el inglés y que su disponibilidad y tiempo de vida es muy impredecible.

(Kalpakis et al., 2016) desarrollan un motor para la búsqueda de recursos web con contenido relacionado con recetas para sintetizar explosivos caseros. El motor está compuesto por cuatro módulos: *crawler* web, componente de consultas, clasificador post-proceso e interfaz de usuario.

(Iliou et al., 2017) proponen un *crawler* focalizado para descubrir recursos de cualquier tema que residan en la web visible o en la Dark Web, incluyendo TOR, I2P y Freenet. Adapta el comportamiento de rastreo, seleccionando hiperenlaces en función de la red de destino.

(Williams et al., 2018) desarrollan un rastreador orientado a la información relacionada con ciberinteligencia que incluye numerosas contramedidas anti-rastreo, incorporando un enfoque *Deep Learning* para clasificar automáticamente los *exploits* en categorías predefinidas, y desarrollar visualizaciones interactivas que permitan a los profesionales e investigadores explorar los *exploits* recopilados.

(Colmenares Malaver et al., 2019) desarrollan una herramienta denominada *Deep Dark Web & Social Crawler* (DDW&SC) para apoyar la gestión de ciberinteligencia que permite realizar búsquedas parametrizadas en Deep Web, Dark Web y redes sociales. Los resultados de la búsqueda se almacenan en una base de datos no relacional MongoDB para su almacenamiento y en Elasticsearch para su posterior procesamiento.

(Scrivens et al., 2019) describen un *crawler* web orientado a recopilar e interpretar información a gran escala relacionada con contenidos extremistas y terrorismo. Para ello realiza búsqueda en base a palabras clave y otros parámetros, analizando la información recuperada de cada página para clasificarla en función de si contiene o no contenido extremista.

(Alkhatib & Basheer, 2019a) proponen una implementación de *crawler* para la Dark Web denominado *Darky* que resuelve los CAPTCHA y automatiza el paso de login, descargando únicamente los datos relevantes y no toda la página.

(Al-Nabki et al., 2019) desarrollan un rastreador web que busca direcciones en la red TOR y proponen el algoritmo *ToRank* para clasificación de *hidden services*, estableciendo una comparativa con otros algoritmos como PageRank, HITS y Katz.

(Kawaguchi & Ozawa, 2019) proponen un sistema que rastrea la Dark Web y recoge URLs maliciosas en función del resultado proporcionado por los motores VirusTotal y Gred.

(Shinde et al., 2020) desarrollan un *crawler* focalizado para obtener material de abusos de mujeres y niños tanto en la web visible como en la Dark Web.

(Narayanan et al., 2020) desarrollan una herramienta de inteligencia *open source* basada en técnicas OSINT para la Dark Web y en un *crawler* de la red TOR.

(Xu et al., 2021) desarrollan un *crawler* para la red TOR basado en Python que rastrea la Dark web para obtener un conjunto de direcciones, almacenando la información en una base de datos Mongo DB.

(Dalvi et al., 2021) introducen *SpyDark*, *crawler* que recopila información tanto de la web visible como de la Dark Web a partir de una consulta realizada por el usuario. Los hipervínculos se almacenan en una base de datos y luego son procesados en base a un modelo PLN (Procesamiento del Lenguaje Natural) previamente entrenado. De este modo, la página se categoriza como relevante o irrelevante.

(Alharbi et al., 2021) emplean un *crawler* desarrollado en Python para generar gráficos sobre la estructura de la red TOR. El estudio concluye que **gran parte de los servicios ocultos no son referenciados en otras páginas**, de modo que muchos servicios ocultos sólo serían accesibles conociendo la dirección .onion o mediante un rastreo de fuerza bruta. No obstante, es común que **referencien páginas de la web visible** como redes sociales, gestión de contenidos web, contenidos de noticias y adultos, presentando en ocasiones **enlaces a la red I2P**. (Iliou et al., 2017) también recogen esta realidad, por lo que es necesario que el proceso de *crawling* pueda ser capaz de **propagarse entre la web visible y las diferentes Dark Nets**.

Existen diferentes rastreadores web cuyo código fuente está disponible en Internet. La Tabla 4 recoge una comparativa de diversas alternativas identificadas, identificando las redes a las que permite acceso y lenguajes o librerías utilizadas:

Tabla 4. *Crawlers* de la Dark Web con código disponible en Internet.

Referencia	Nombre *crawler*	Redes	Lenguaje / librerías utilizadas
(Miller, 2019)	TorCrawler	TOR	Python BeautifulSoup

Referencia	Nombre *crawler*	Redes	Lenguaje / librerías utilizadas
(Foo-Manroot, 2018)	ToR_crawler	TOR	Python PySocks, Stem, Beautifulsoup, Pycurl, Html5lib
(Webfp, 2021a)	TOR-browser-crawler	TOR	Python Stem, Scapy, Psutil, Pyvirtualdisplay, Selenium, Tbselenium
(Webfp, 2021b)	Tor-browser-selenium	TOR	Python Stem, Pytest, Pytest-cov, Pytest-rerunfailures, Psutil
(Magán-Carrión et al., 2021)	c4darknet	I2P Freenet	Python
(Santos & Pham, 2022)	ACHE	Web TOR	Java
(Narayanan et al., 2022)	TorBot	TOR	Python Beautifulsoup4, Pyinstaller, PySocks Termcolor, Requests, Requests_mock, Yattag, Numpy
(Micard & Ashurst, 2019)	Trandoshan	TOR	Go

Fuente: Elaboración propia.

Asimismo, existe literatura relativa al desarrollo de *crawlers* para obtención de información relativa a **mercados en la Dark Web**. En estos casos, se realizan diversos pasos para la obtención de información, como (i) Recorrer todo el portal del sitio web, (ii) Extraer toda la información y datos relevantes en relación a la temática buscada en el mercado, (iii) Clasificar los listados en categorías en función del producto específico que se esté buscando. (Mathur et al., 2020) contemplan un *crawler* automatizado para el análisis de *markets* en la Dark Web, centrado en la identificación de productos ilegales específicos (drogas). (Christin, 2012) emplea un método simple de recoger información del *market Silk Road* a partir de una adaptación de la herramienta HTTrack y haciendo uso de las cookies de autenticación para evitar medidas de protección basadas en CAPTCHA. (Décary-Hétu & Aldridge, 2015) introducen el *crawler* '*Datacrypto',* software que recorre sitios webs y genera bases de datos de listados de producto, información del vendedor y *feedback* de los compradores.

Un aspecto relevante en el análisis de *markets* en la Dark Web es la clasificación de datos obtenidos y obtención de patrones, para lo cual se han utilizado técnicas de *data mining* (análisis automatizado de una gran cantidad de datos para extraer nueva información relacional y estadística) y *machine learning*. (Alkhatib & Basheer, 2019b) utilizan técnicas de

minería de datos para establecer patrones que puedan ser útiles para clasificar los contenidos de estos mercados.

Los trabajos anteriores se han centrado en obtener de manera automática contenidos de texto de sitios web, existiendo **pocos estudios orientados a la obtención de información relativas a las imágenes** (Uzun et al., 2020). A este respecto**,** (Zulkarnine et al., 2016) desarrollan un *crawler* que realiza búsquedas en Internet basadas en palabras clave predefinidas y/o hashes de imágenes por medio de algoritmos MD5, SHA-1 o *PhotoDNA* (tecnología de identificación de imagen empleada para detectar pornografía infantil y otras actividades ilegales). Está basado en un *crawler* web denominado *Child Explotation Network Extractor* (CENE), focalizado en la obtención de material de explotación infantil, permitiendo obtener información tanto en la web visible como en la Dark Web.

(Fidalgo et al., 2017) realizan un estudio de clasificación de imágenes de la Dark Web en diversas categorías de actividades ilegales.

(Fazal et al., 2019) desarrollan un rastreador denominado *Mopsi Image Crawler* (MIC), destinado a descargar fotos con información geográfica. Como criterio para seleccionar imágenes a analizar consideran aquellas que tienen una **anchura o altura superior a 400 px**.

En relación a la relevancia de las imágenes de una página web, (Gali et al., 2015) establecen determinadas características que deben tener las imágenes de una web para ser relevantes, así como aquellos atributos que pueden responder a otras no importantes como iconos, logos, banners, etc. Por otro lado, (Vyas & Frasincar, 2020) proponen un algoritmo mejorado basado en el uso de *Support Vector Machines* (SVM) para identificar la imagen más representativa de una página web.

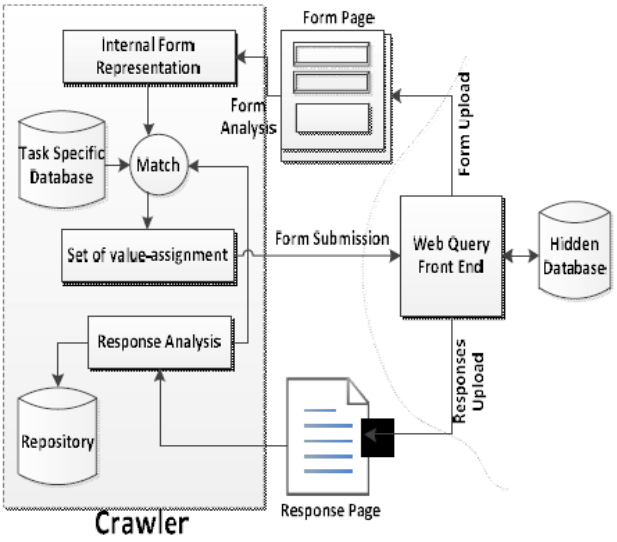
Dadas las características de las técnicas criptográficas de hashing como MD5 o SHA1, debe tenerse en cuenta que **pequeños cambios en las imágenes** (como por ejemplo, un ligero ajuste de tamaño o el cambio de formato JPG a PNG) **cambian totalmente el hash resultante**. En este sentido, para poder realizar **comparaciones relativas a la similitud de las imágenes** se pueden emplear técnicas avanzadas de **hashing de imágenes (*image hashing)***. Entre ellas, destaca el algoritmo ***difference hashing,*** en el que el cálculo se realiza en función del aumento o disminución de la intensidad píxel a píxel. De este modo, se pueden obtener resultados relativos a la similitud de dos imágenes, lo cual no es posible con los algoritmos tradicionales como MD5 o SHA1.

(Fitas et al., 2021) realizan un estudio y comparativa de diferentes algoritmos de hashing de imágenes en un caso de uso de tapones de corcho, obteniendo los mejores resultados mediante la técnica de *difference hash* (DH).

(D. Wang & Liang, 2019) desarrollan un *crawler* para analizar el problema de comparación de imágenes por medio del algoritmo *difference hash* combinado con otros algoritmos como *Web Text Cosine Correlation* y *Link PageRank.*

En relación a ***crawlers* para la Deep Web**, existe diversa literatura. Con carácter general, este tipo de rastreadores se basan en interactuar con formularios para la realización de consultas en base a uno o múltiples campos y/o sitios web que requieren autenticación o registro previo, del modo mostrado en la Figura 10. Su modo de operar se basa en las actuaciones que haría un usuario, interactuando con un formulario para la obtención de información.

Figura 10. Interacción con formulario de crawler para la Deep Web.



Fuente: (Gupta & Bhatia, 2014).

Existen diversos estudios que proporcionan una visión general sobre este tipo de rastreadores. (Gupta & Bhatia, 2014) realizan un estudio comparativo de este tipo de *crawlers*. (Shelke & Sagar, 2017) realizan una recopilación de diferentes estudios relacionados con técnicas de URL y *content matching* para descubrir páginas en la Deep Web. Por otro lado, (Hernández et al., 2019) analizan las principales características de las técnicas existentes relacionadas con el rastreo de la web profunda, proporcionando una imagen general sobre el rastreo de la Deep Web.

Hasta la fecha se han desarrollado múltiples estudios relacionados con *crawlers* para la web profunda. (Raghavan & Garcia-Molina, 2001) desarrollan un rastreador para la Deep Web denominado *Hidden Web Exposer (HiWE)*.

(Mundluru & Xia, 2008) realizan un estudio sobre el rastreo y extracción de datos de búsqueda de productos y servicios geográficamente locales, desarrollando un *crawler* denominado *LocalDeepBot*.

(Madhavan et al., 2008) describen un sistema para rastrear la Deep Web incorporado en el motor de búsqueda de Google.

(Soulemane et al., 2012) describen un algoritmo para un *crawler* enfocado a obtener páginas indexadas en la Deep Web en base a consultas de formularios de un único campo de texto y enviadas mediante método GET.

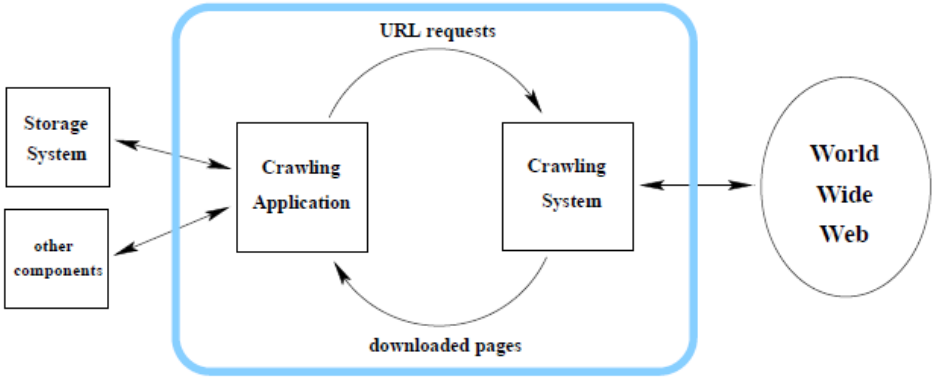
(Rashmi et al., 2016) proponen un web *crawler* para la Deep Web de dos niveles. En el primer nivel, el rastreador explora los formularios partiendo de un sitio semilla determinado, clasificando los sitios para priorizar los relevantes en base a un clasificador bayesiano ingenuo (*Naive Bayes Classifier*) y extrae los enlaces para encontrar los formularios. En el segundo nivel, busca en los formularios según la preferencia y el resultado se mejora al volver a clasificar, según las preferencias de los usuarios. Hace uso de funciones de los motores de búsqueda existentes, como la función "*link*" de Google.

(Rahayuda & Santiari, 2017) emplean un *crawler* que almacena datos de la Hidden Web en una base de datos y realiza un proceso de clasificación de datos basado en el método *Fuzzy-K Nearest Neighbor (Fuzzy-KNN).*

(Kaur et al., 2021) desarrollan un rastreador denominado *SmartCrawler*, que desarrolla un algoritmo de clasificación mejorado para priorizar sitios de la Deep Web y aumentar la cobertura de los rastreadores de la web oculta.

Existen diversas iniciativas relacionadas con **crawler distribuidos**. (Thelwall, 2001) introduce un diseño de *crawler* web distribuido para análisis de datos. (Shkapenyuk & Suel, 2002) introducen un *crawler* de alto rendimiento, orientado a funcionar de manera adecuada en entornos distribuidos de red. Proponen una estructura basada en dos módulos, según se muestra en la Figura 11. La aplicación de rastreo (*crawling application*) decide qué página solicitar a continuación en base a las páginas rastreadas anteriormente, dando la orden al sistema de rastreo (*crawling system*) quien descarga las páginas solicitadas y las proporciona a la aplicación de rastreo para su análisis y almacenamiento. El sistema es modular, constando de varios componentes que pueden ser replicados para mayor rendimiento.

Figura 11. Arquitectura de crawler distribuido de alto rendimiento propuesta.



Fuente: (Shkapenyuk & Suel, 2002).

(Tomala et al., 2013) desarrollan un *crawler* multi agente para obtener datos de la red social Twitter a partir de búsquedas basadas en palabras clave.

(Raj et al., 2021) introducen un *crawler* distribuido eficiente usando un algoritmo de ancho de banda mejorado, orientado a proporcionar un mejor rendimiento en sistemas distribuidos.

Por último, en relación a los **rastreadores web focalizados (*focused crawler*)**, se han desarrollado diversas soluciones para realizar búsquedas específicas. (Kumar et al., 2018) desarrollan un *crawler* focalizado en el que se emplea un conjunto de palabras clave o *keywords* relevantes a la hora de lanzar consultas en el interfaz de búsqueda, lo que permite obtener los enlaces más relevantes sin tener que recorrer todas las páginas. (J. Wang et al., 2019) diseñan un *crawler* focalizado multi idioma basado en palabras clave, soportando chino, inglés, japonés, ruso y árabe. Se configura a partir de una *keyword* en chino, que se traduce el resto de idiomas. Finalmente, (Suzanti et al., 2021) proponen un rastreador para búsquedas relativas a ingredientes basados en carne en el ámbito de Indonesia.

3. OBJETIVOS Y METODOLOGÍA

En este capítulo se introducen los objetivos principal y secundarios contemplados para el presente TFM, así como la metodología empleada para el desarrollo del *crawler* objeto del mismo.

3.1. Objetivo principal

El objetivo principal es proporcionar una herramienta software que permita realizar una **búsqueda automatizada y efectiva de patrones delictivos en Internet, tanto en la web visible como en la Dark Web**.

3.2. Objetivos secundarios

La principal aportación de la herramienta es poder incorporar, en **un único software, funcionalidades que han sido incorporadas con anterioridad de manera aislada**, en base a los objetivos secundarios enumerados a continuación y las especificaciones establecidas en el "Anexo A. Requisitos de la aplicación".

Como objetivos secundarios del trabajo se plantean los siguientes:

- **Crawler multiplaforma**: ejecutable en sistemas Windows y Linux.
- **Multi idioma:** pudiendo ser utilizada por autoridades de diversos países para la búsqueda de patrones delictivos[1].
- **Multi red**: analiza contenido tanto de la web visible como de diversas Dark Nets (TOR, I2P y Freenet). Su diseño permite alternar enlaces entre las diferentes redes, posibilitando seguir, por ejemplo, un enlace de una página de la web visible a la red TOR.
- **Diseño modular**: en el que se puedan ir incorporando funcionalidades de manera gradual sin afectar al resto del programa.
- **Funcionamiento configurable:** se contempla que múltiples aspectos relacionados con el funcionamiento sean personalizables por el usuario mediante un fichero de configuración.
- **Imágenes:** obtención de información relativa a imágenes, como hashes o metadatos. No se almacenan imágenes descargadas que pudieran contener contenidos delictivos. Teniendo en cuenta que ligeras modificaciones en una imagen pueden alterar por

[1] En esta fase soporta los idiomas español e inglés.

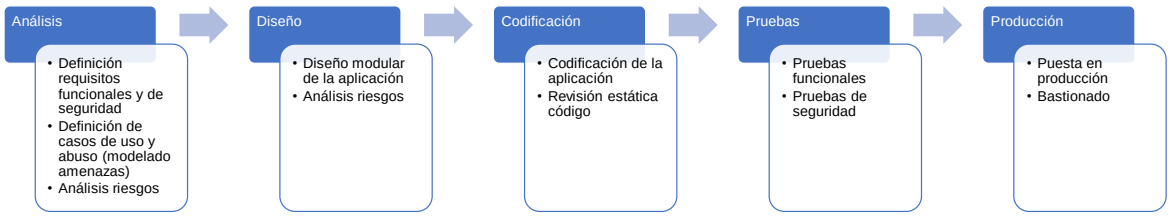
completo los hashes correspondientes, se contempla el uso de técnicas de hashing de imágenes que permitan relacionar aquellas que tengan un cierto parecido.

- **Herramienta distribuida:** los resultados se almacenan en tiempo real en una base de datos centralizada, permitiendo el funcionamiento simultáneo de diversos *crawlers* ejecutados en diferentes máquinas.
- **Búsqueda de patrones delictivos**: el software permite realizar búsqueda de patrones en la información almacenada, de cara a poder obtener direcciones web en las que se almacene información asociada a un determinado *nickname* o imágenes con un determinado hash o con metadatos que coincidan con un determinado patrón.
- **Ciclo de vida de desarrollo software seguro (S-SDLC)**: la aplicación se desarrolla según esta metodología para reducir el riesgo de que sufra vulnerabilidades que puedan afectar a su operación.

3.3. Metodología

Para el desarrollo del software se sigue una metodología de **ciclo de vida de desarrollo software seguro** (S-SDLC *Secure Software Development Life Cycle*), basada en (McGraw, 2006) y mostrada en la Figura 12.

Figura 12. Metodología de desarrollo propuesta.



Fuente: Elaboración propia derivada de (McGraw, 2006).

Se describe a continuación los pasos contemplados en cada una de las fases identificadas.

3.3.1. Análisis

En esta fase se realiza en primer lugar la **definición de requisitos**. En este punto se contemplan tanto los requisitos funcionales que marcan el funcionamiento de la aplicación como los no funcionales, incluyendo:

- Requisitos de servicios de seguridad, entendidos como especificaciones que implementan funcionalidades de seguridad.

- Requisitos de software seguro, que garanticen que el sistema sea seguro, aunque se materialicen las amenazas contempladas en el modelado de amenazas.

En paralelo se realiza la **definición de los casos de uso** que determinan el funcionamiento esperado del sistema y los **casos de abuso.** Para la determinación de los casos de abuso se realiza un **modelado de amenazas** mediante la herramienta **Microsoft Threat Modelling Tool** versión 7.3.2012.2. Las amenazas se identifican siguiendo el modelo STRIDE (*Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privileges*), en base a las propuestas realizadas por la propia herramienta. La valoración de las amenazas se realiza por medio del método DREAD (*Damage, Reproducibility, Explotability, Affected Users, Discoverability*). Los criterios contemplados para la realización del análisis se recogen en el "Anexo B. Análisis de modelado de amenazas".

Para aquellas amenazas cuya valoración global de riesgo sea de nivel 'Medio' ('*Medium*') o 'Alto' ('*High*') se realiza una identificación de las vulnerabilidades asociadas a la amenaza mediante el listado MITRE CWE (*Common Weakness Enumeration*), donde se recoge de manera exhaustiva vulnerabilidades presentes en el desarrollo software, permitiendo una mejor comprensión y mitigación de las mismas (The MITRE Corporation, 2022b). Asimismo, se identifican los modelos de ataque recogidos en el listado MITRE CAPEC (*Common Attack Pattern Enumeration and Classification*) (The MITRE Corporation, 2022a). Como consecuencia, todas estas amenazas deben presentar un requisito de seguridad que mitigue el riesgo asociado.

Finalmente, en esta fase se realiza un análisis de riesgo arquitectónico mediante la herramienta online SandaS GRC (GOVERTIS Advisory Services, S.L., 2015), que se completa en la fase de diseño.

3.3.2. Diseño

En esta etapa se lleva a cabo el **análisis de riesgos arquitectónico**, con las fases y criterios especificados en el "Anexo C. Análisis de riesgos arquitectónico".

Asimismo, en esta fase se realiza el diseño de la aplicación. Teniendo en cuenta los objetivos determinados para la misma, se debe contemplar un **diseño modular**. Para ello, en esta fase se definen y diseñan los diferentes módulos contemplados. Para la interacción entre los diversos módulos se contempla el uso de:

- Clases: para agrupar un conjunto de información relacionada.
- Interfaces: considerados como funciones que serán invocadas desde otros módulos.

Para facilitar la comprensión del diseño, los módulos y sus interacciones se contempla el desarrollo de diversos ***diagramas UML*** (*Unified Modeling Language*) (Object Management Group, s. f.):

- **Diagrama de casos de uso y abuso**, para ilustrar los usos previstos por el programa y los principales ataques que puede sufrir.
- **Diagrama de clases**, al objeto de facilitar la comprensión de las clases necesarias para el desarrollo.
- **Diagrama de componentes**, para describir los diferentes módulos de la aplicación y sus relaciones.
- **Diagrama de actividades**, para describir las acciones a desarrollar dentro de los diferentes módulos.

3.3.3. Codificación

En fase de codificación se realiza la **codificación de la aplicación para la implantación de los requisitos** establecidos en la etapa de análisis y en base al diseño arquitectónico desarrollado en la fase de diseño. Asimismo, se realizan las verificaciones indicadas a continuación:

- **Análisis estático de código fuente**: por medio de la aplicación **Sonarqube** versión 9.3.0.51899 (SonarSource S.A., s. f.). Para la comprobación de la cobertura de pruebas automatizadas se emplea el software Coverage versión 6.3.2 (Batchelder, s. f.). La comprobación verifica el cumplimiento de las reglas contenidas en la Tabla 5:

Tabla 5. Criterios de aceptación del análisis estático de código fuente.

Comprobación	Criterio
Cobertura de pruebas automatizadas	≥ 90%
Líneas duplicadas	≤ 3 %
Bugs detectados y no resueltos	0
Vulnerabilidades de seguridad detectadas y no resueltas	0

Fuente: Elaboración propia.

- **Comprobación de la ausencia de vulnerabilidades** en las librerías importadas en el código mediante la aplicación **Dependency-check** versión 7.0.4 (OWASP Foundation, Inc., s. f.).

3.3.4. Pruebas

En esta fase se contempla la **automatización de pruebas** que deben contemplar:

- Pruebas funcionales, orientadas a comprobar el correcto funcionamiento del programa en base a las especificaciones realizadas en el “Anexo A. Requisitos de la aplicación”.
- Pruebas de seguridad, orientadas a probar los casos de abuso detectados en la fase de análisis, según lo descrito en “3.3.1 Análisis”[2].

Las pruebas se automatizan por medio de la ejecución de scripts.

Nota: si bien se representa esta fase de manera secuencial posterior a la de codificación, realmente se ejecutan de manera simultánea en paralelo.

3.3.5. Producción

En esta fase se lleva a cabo la **puesta en producción de la aplicación** junto con un **proceso de bastionado del software en base a guías de configuración segura** a nivel de:

- Sistema operativo.
- Virtualización.
- Sistema gestor de base de datos.

[2] La defensa frente a los casos de abuso ha sido contemplada mediante requisitos de seguridad definidos en el “Anexo A. Requisitos de la aplicación”.

4. DESARROLLO ESPECÍFICO DE LA CONTRIBUCIÓN

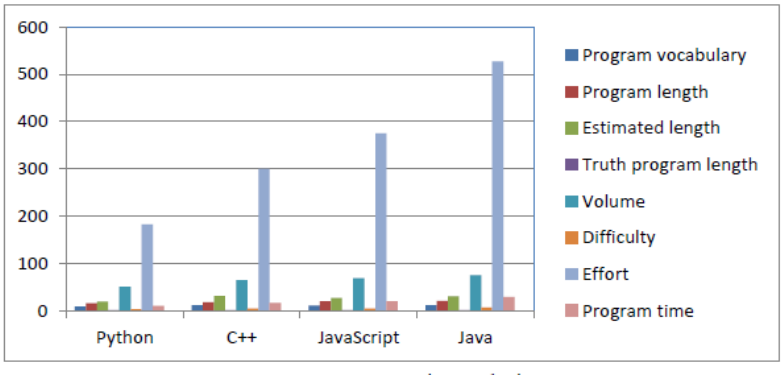
En este capítulo se desarrolla la propuesta realizada en base a la metodología especificada en el apartado “3.3 Metodología”.

Uno de los aspectos fundamentales a considerar a la hora de plantear la propuesta es el lenguaje de programación utilizado. En el desarrollo se emplea **Python** versión 3.10.2, ya que aporta las siguientes ventajas:

- **Sencillez**, con una curva de aprendizaje menor que otras alternativas.
- **Gran cantidad de librerías y desarrollos.** Python proporciona una gran cantidad de librerías dedicadas al rastreo web, el análisis de HTML, *Data Mining*, *Deep Learning* y otras funcionalidades importantes necesarias para el desarrollo de *crawlers* y motores de búsqueda (Williams et al., 2018).
- **Portabilidad**, para poder emplearlo en diferentes sistemas operativos (Windows, Linux, MacOS).
- **Licencia de código abierto,** lo que evita costes y facilita futuros desarrollos.

La Figura 13 y la Tabla 6 muestran una comparativa entre diversos lenguajes de programación.

Figura 13. Comparativa complejidad lenguajes de programación Python, C++, JavaScript y Java.



Fuente: (Abdulkareem & Abboud, 2021).

Tabla 6. Comparativa lenguajes de programación C/C++, Java, Python y PERL.

Aspecto	C / C++	Java	Python	PERL
Tiempo ejecución	Bajo	Medio	Medio	Medio
Consumo memoria	Bajo	Medio	Medio	Medio
Longitud programa	Alta	Media	Media	Media

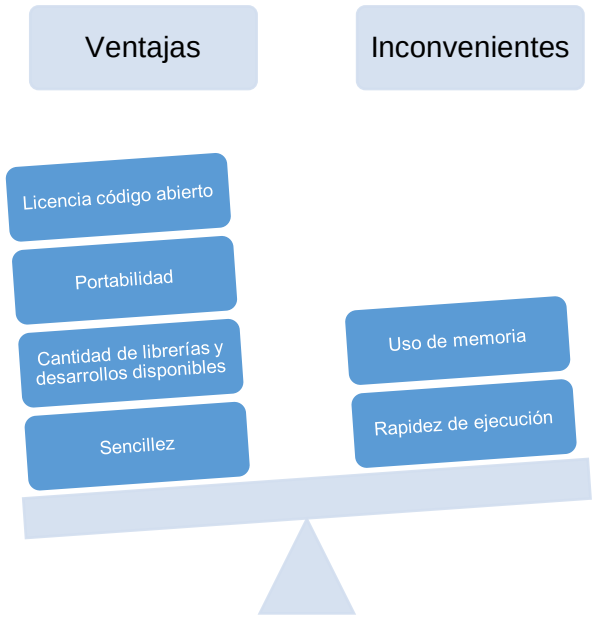
Aspecto	C / C++	Java	Python	PERL
Complejidad	Media	Alta	Baja	Baja

Fuente: Elaboración propia a partir de (Prechelt, 2000) (Fourment & Gillings, 2008) y (Abdulkareem & Abboud, 2021).

Teniendo en cuenta que la herramienta se propone como un software modular que pueda ser desarrollada en el futuro para incorporar nuevas funcionalidades, el uso de Python facilita que nuevos desarrolladores sigan impulsándola por la sencillez del lenguaje, al tiempo que reduce el tiempo de desarrollo por el gran número de librerías disponibles. De este modo, abre la puerta en el futuro a realizar desarrollos avanzados en el campo del *machine learning* o *big data* para el análisis de la información obtenida o mejora de los algoritmos de *crawling*.

Entre los principales inconvenientes de Python se encuentran la velocidad y uso de memoria, existiendo otras alternativas más eficientes como C/C++. No obstante, se considera que las ventajas mencionadas suponen un mayor peso a la hora de seleccionar esta alternativa.

Figura 14. Ventajas e inconvenientes de Python.



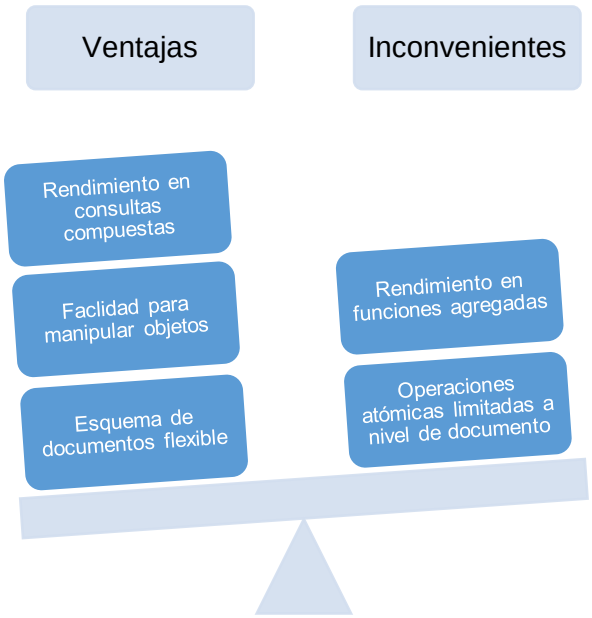
Fuente: Elaboración propia.

Para el almacenamiento de la información rastreada se propone el uso de **MongoDB Community Server** versión 5.0.7, base de datos no relacional que almacena datos en un formato BSON (*Binary JSON*, basado en JSON -*JavaScript Object Notation*-) con pares clave – valor, a diferencia de una base de datos SQL relacional en la que la información se almacena en tablas. Una base de datos MongoDB no relacional proporciona las siguientes ventajas sobre una base de datos SQL convencional (MongoDB, Inc., s. f.-a):

- **Esquema de documentos flexible**: el formato de datos de MongoDB, inspirado en JSON, permite tener objetos en una colección con diferentes conjuntos de campos (por ejemplo, información específica que sólo se aplica a algunos registros).
- **Facilidad para manipular los objetos**: se puede manipular los datos mediante estructuras de datos nativas en el lenguaje de programación (diccionarios en el caso de Python).
- **Rendimiento en consultas compuestas:** la información se puede insertar en un solo documento en lugar de depender de operaciones UNION como en sistemas gestores de bases de datos relacionales tradicionales. Esto hace que las consultas sean mucho más rápidas, devolviendo toda la información necesaria en una sola llamada a la base de datos.

Como contrapartida, MongoDB ofrece algunas desventajas, como limitaciones para proporcionar operaciones atómicas (haciéndolo únicamente a nivel de un documento) o peor rendimiento para funciones agregadas y consultas basadas en valores no clave (Parker et al., 2013).

Figura 15. Ventajas e inconvenientes de MongoDB.



Fuente: Elaboración propia.

Se recoge a continuación el desarrollo de la propuesta en base a las fases recogidas en el capítulo “3.3 Metodología”: Análisis, Diseño, Codificación, Pruebas y Producción. Finalmente se recogen los resultados obtenidos como consecuencia del trabajo realizado.

4.1. Análisis

La fase de análisis comienza con la definición de los requisitos, tanto funcionales como no funciones, según lo especificado en el capítulo “3.3.1 Análisis”.

Los requerimientos del software se recogen en el “Anexo A. Requisitos de la aplicación”. Para el desarrollo de la aplicación se ha considerado un enfoque tipo ***agile***, de modo que, en vez de contemplar un conjunto de requisitos único a ser desarrollados en un único entregable, estos se han dividido en tres fases iterativas:

- **Fase 1: Requisitos imprescindibles** para el producto mínimo viable, que deberán ser implantados obligatoriamente en el entregable del presente TFM.
- **Fase 2: Requisitos deseables** para el producto mínimo viable, que pueden ser implantados opcionalmente con posterioridad a finalizar la fase 1.
- **Fase 3: Requisitos futuros** que no serán incluidos en el entregable del presente TFM y que podrían considerarse opcionalmente en futuros desarrollos de la herramienta.

De este modo, se puede contemplar el **desarrollo de un producto mínimo viable** con el que empezar a obtener resultados de búsqueda de patrones delictivos en la Dark Web.

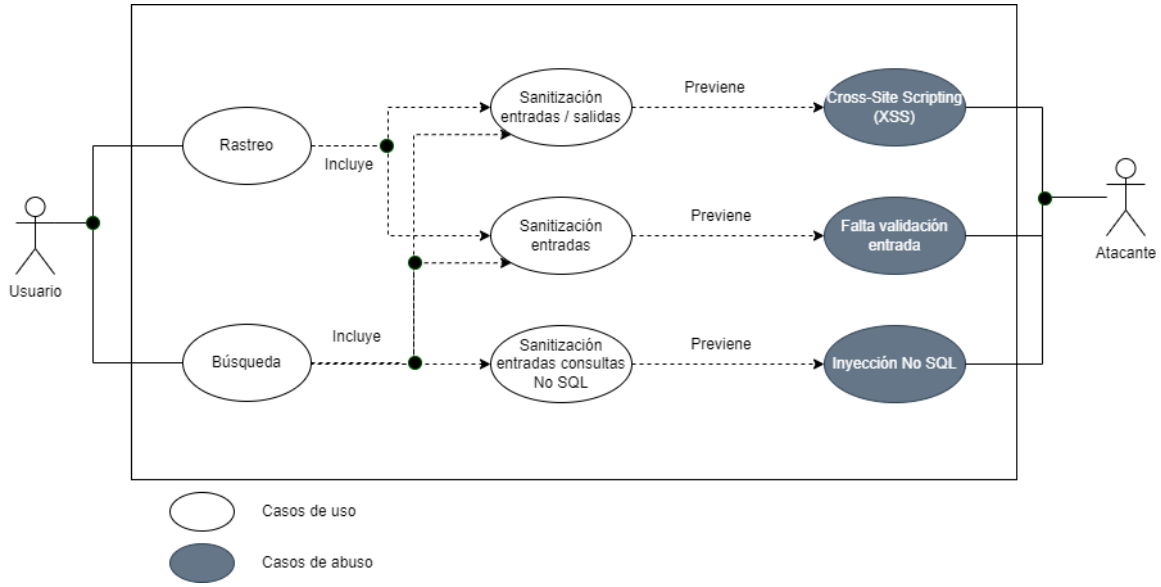
Conjuntamente con la recopilación de requisitos, se ha realizado un diagrama de casos de uso, recogido en la Figura 16, en el que se identifican dos casos:

- **Rastreo,** que lanza el proceso de *crawling* a partir de los datos de un fichero de configuración y almacena la información rastreada en una base de datos.
- **Búsqueda**, que permite identificar las páginas tanto de la web visible como de la Dark Web que alojan el contenido que cumple los criterios pasados como parámetro (como por ejemplo un patrón de nombre de usuario).

En paralelo a los casos de uso se ha realizado un análisis de **modelado de amenazas** para determinar las principales amenazas sobre la aplicación, cuya metodología y resultados se detallan en el “Anexo B. Análisis de modelado de amenazas”.

Tras la identificación de los casos de uso identificados anteriormente y las amenazas recogidas en la Tabla 35 (cuyo valor de riesgo es ‘Medio’ o ‘Alto’) se desarrolla un diagrama UML (*Unified Modeling Language*) de casos de uso y abuso en la Figura 16.

Figura 16. Diagrama UML de casos de uso y abuso.



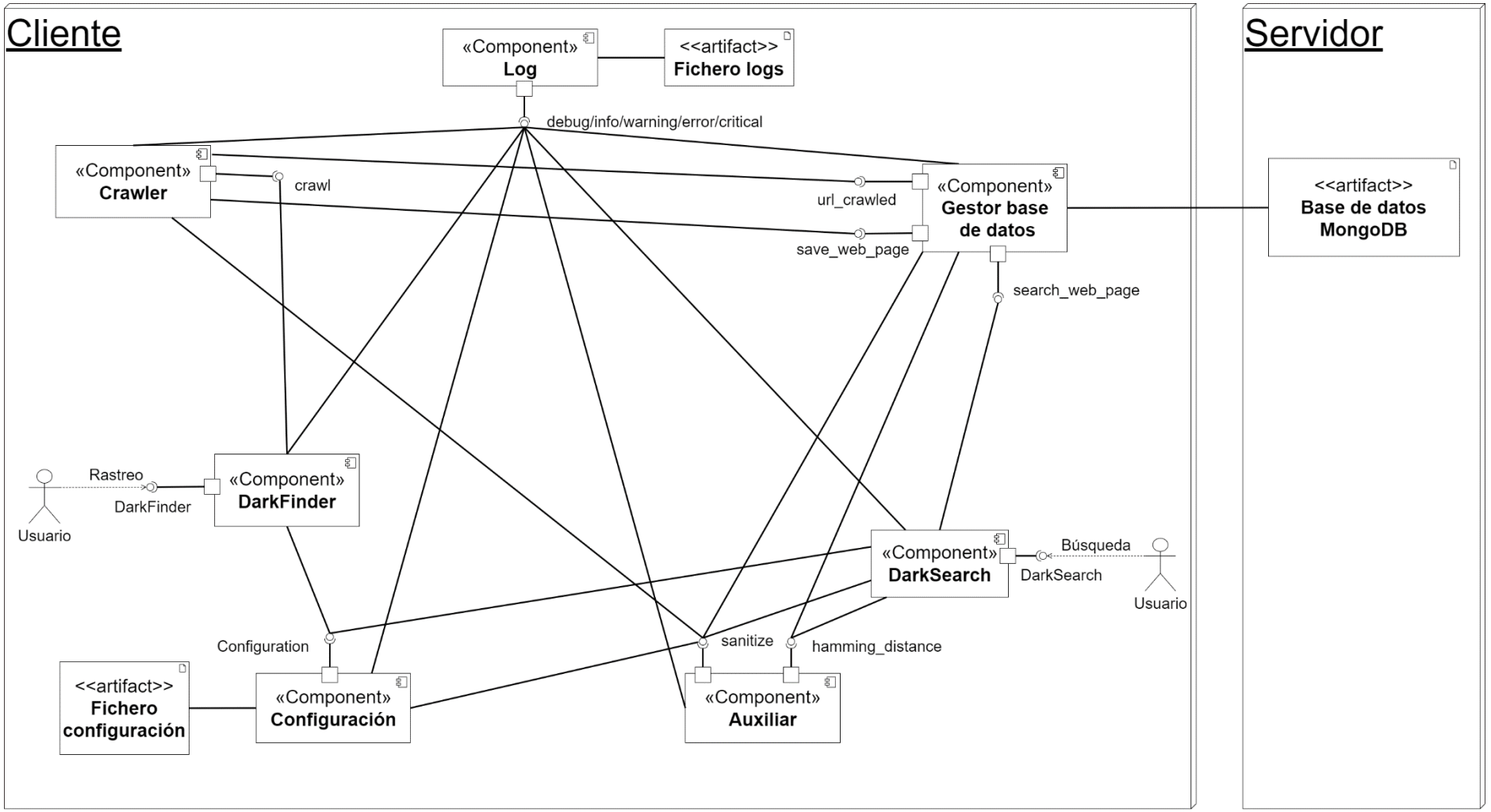
Fuente: Elaboración propia.

4.2. Diseño

Tras finalizar la fase de análisis se realiza un **análisis de riesgos arquitectónico**, siguiendo la metodología y obteniendo los resultados identificados en el "Anexo C. Análisis de riesgos arquitectónico".

Uno de los aspectos principales de esta fase es la realización del **diseño de la aplicación**. La Figura 17 representa la arquitectura propuesta para la herramienta de *crawling*, denominada *DarkFinder,* mediante un diagrama UML de componentes. En ella se pueden apreciar los diferentes módulos contemplados para su desarrollo, cuya funcionalidad se recoge en la Tabla 7:

Figura 17. Diagrama UML de componentes de la aplicación.



Fuente: Elaboración propia.

Tabla 7. Módulos contemplados en el crawler.

Módulo	Descripción
Configuración	Módulo encargado de leer los parámetros de configuración determinados en el fichero *darkfinder.conf* (ubicado en el mismo path que la aplicación) y cargarlos en la clase *Configuration.*
DarkFinder	Módulo principal del caso de uso de rastreo que lanza el proceso de *crawling*.
Crawler	Módulo encargado de realizar el rastreo en la web visible y Dark Web a partir de las URL semilla especificadas en el fichero de configuración. Para cada web rastreada carga la información en clases de tipo *Page* (información relativa a páginas web) e *Image* (información relativa a imágenes contenidas en la página web).
Log	Módulo encargado de escribir en un fichero log (*darkfinder.log*) y mostrar por pantalla un mensaje (*message*) generado en la aplicación.
Gestor de base de datos	Módulo encargado de realizar de interfaz con la base de datos para: • Almacenar la información de rastreo (interacción con el módulo DarkFinder). • Indicar al módulo C*rawler* si una determinada URL ha sido rastreada. • Realizar consultas relativas a la información rastreada (interacción con el módulo DarkSearch).
DarkSearch	Módulo principal del caso de uso de búsqueda de información. Invoca al módulo gestor de base de datos para obtener las páginas cuyo contenido coincide con los argumentos de búsqueda especificados por el usuario.
Auxiliar	Módulo auxiliar que contiene funciones invocadas desde el resto de módulos para sanitización de entradas.

Fuente: Elaboración propia.

Con carácter general, en el diseño y codificación de la herramienta se siguen los **principios de diseño de software seguro** contemplados en la Tabla 8, desarrollados a partir de (Howard & LeBlanc, 2003) y (Goertzel & Winograd, 2008).

Tabla 8. Principios de diseño de software seguro contemplados.

Principio de diseño	Objetivo	Implementación
Defensa en profundidad	Introducir múltiples capas de seguridad para reducir la capacidad de compromiso del sistema.	• Medidas de seguridad perimetral[3]. • Desarrollo de software seguro.

[3] No especificadas en el presente TFM por no ser específicas del mismo y dependientes de la propia organización en la que se implanten.

Principio de diseño	Objetivo	Implementación
Mínimo privilegio	Minimizar el número de actores con altos niveles de privilegios, así como el tiempo en el que disponen de los mismos.	• Ejecución de herramienta en sistema operativo con gestión de usuarios. • Ejecución de herramienta por usuarios sin privilegios de administración.
Separación de dominios	Limitar acceso a ubicaciones de memoria u objetos de datos de sistema.	• Instalación de la herramienta en entorno virtualizado.
Separación de código, ejecutables y datos de configuración y programa	Aislar los datos de programa y configuración del ejecutable del programa.	• Datos alojados en base de datos en servidor aislado de la ejecución de código en cliente.
Entorno de datos no confiable	Asumir que todas las entradas externas pueden ser manipuladas y convertir el programa en no confiable.	• Sanitización de las entradas procedentes de línea de comandos, fichero de configuración, acceso web y base de datos.
Usar interfaces seguros a recursos del entorno	Limitar las llamadas a interfaces del sistema operativo.	• Empleo de librerías estándar Python, evitando el uso directo de llamadas a funciones del sistema operativo.
Simplicidad en el diseño	Reducir la complejidad del diseño.	• Diseñar los módulos con el número mínimo posible de funcionalidades. • Cada método o función sirve a una única finalidad. • Minimizar interdependencia entre módulos. • Análisis de la complejidad cognitiva[4] del conjunto de funciones y métodos de la aplicación por medio de herramienta de análisis estático de código.
Registro de eventos de seguridad	Generar logs de seguridad para garantizar que se observan y registran las acciones de los usuarios y del software.	• Registro en logs de errores y actuaciones del usuario.

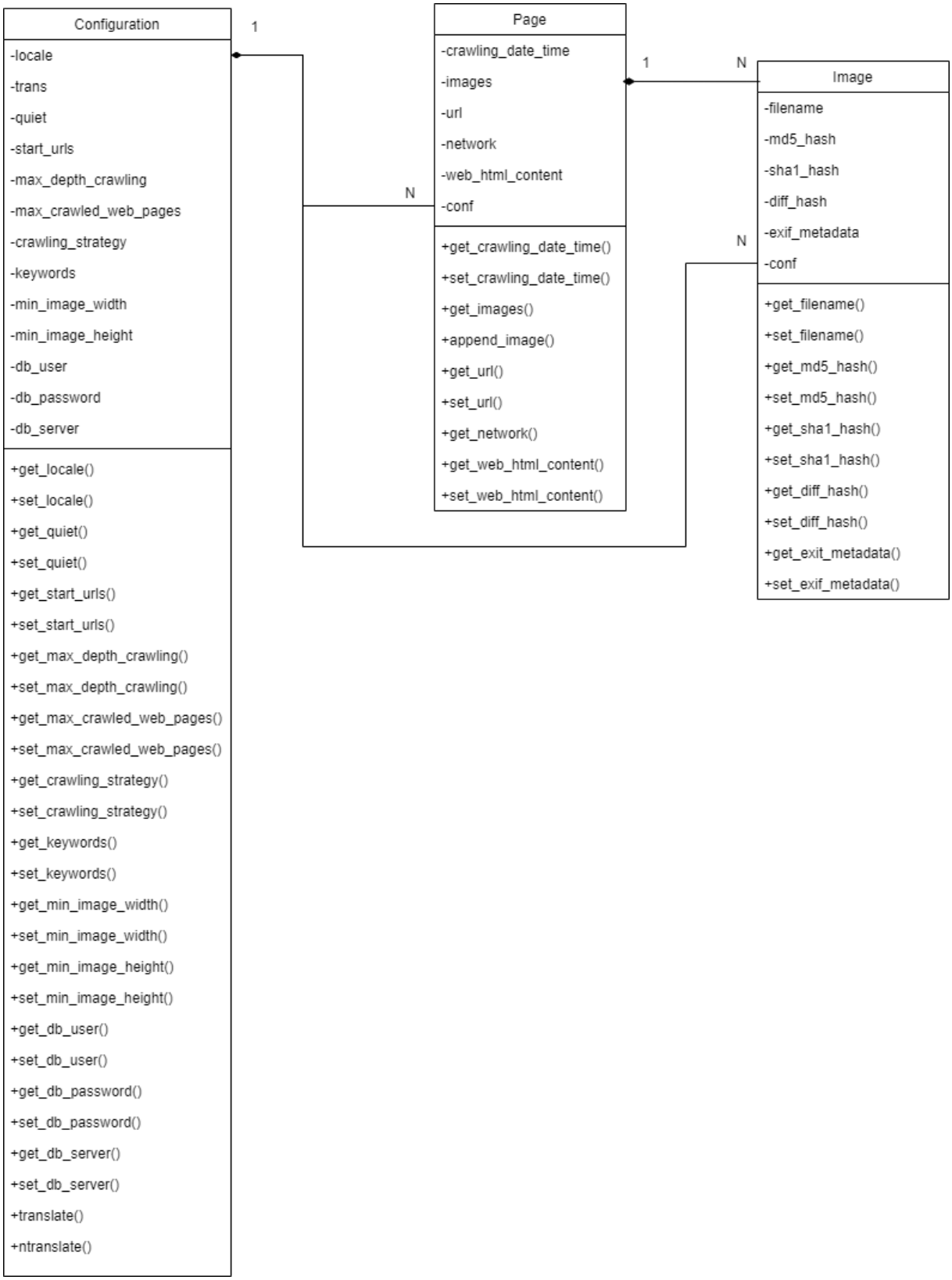
[4] Medida de la complejidad del flujo de control de un método o función, calculada según la metodología indicada en (Campbell, 2021).

Principio de diseño	Objetivo	Implementación
Fallar de forma segura	Evitar que un fallo en el software deje la aplicación en un estado inseguro.	• Captura de errores de programa mediante bloques *try-except*.
Seguridad por defecto	Reducir la superficie de ataque.	• Bastionado del sistema operativo, virtualización y sistema gestor de base de datos. • Importar únicamente aquellas funciones que vayan a ser empleadas en la aplicación mediante la secuencia [*from module import function*], evitando realizar importaciones globales del tipo *import.** • Evitar la inserción de contraseñas en el código fuente (*hardcoded password*). • Eliminar usuarios por defecto en sistema operativo y sistema gestor de base de datos.

Fuente: Elaboración propia a partir de (Howard & LeBlanc, 2003) y (Goertzel & Winograd, 2008).

La herramienta se diseña haciendo uso de clases, contemplando una **programación orientada a objetos**. La Figura 18 muestra las clases contempladas en el diseño, descritas en la Tabla 9 y detalladas en el “Anexo D. Diseño de clases”:

Figura 18. Diagrama UML clases.



Fuente: Elaboración propia.

Tabla 9. Clases contempladas en el *crawler*.

Clase	Descripción
Configuration	Clase que almacena los parámetros de configuración especificados en el fichero *darkfinder.conf.*
Page	Clase que almacena datos de las páginas web rastreadas. Contiene un array de objetos tipo *Image* con las imágenes de cada web[5].
Image	Clase que almacena datos específicos de las imágenes detectadas en las webs rastreadas.

Fuente: Elaboración propia.

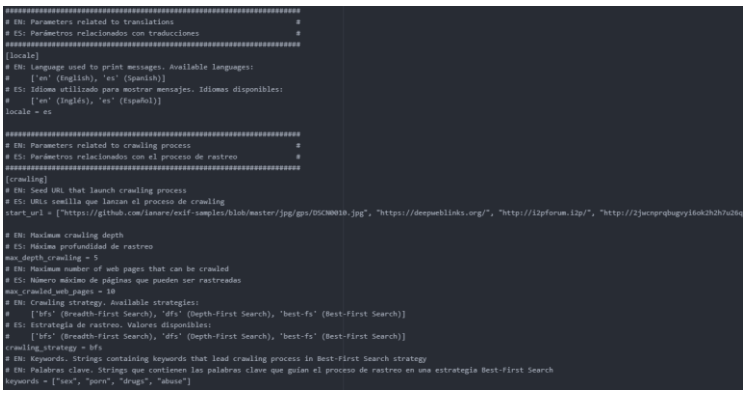
Los atributos de las clases tienen el carácter de privado (*Private*), por lo que sólo pueden ser accedidos desde métodos de la propia clase. Por el contrario, todos los métodos tienen carácter público (*Public*), pudiendo ser accedidos desde cualquier parte del código. De este modo, asociado a cada atributo se desarrollan dos métodos:

- *Get*: devuelve el valor correspondiente al atributo.
- *Set*: establece el valor del atributo, una vez realizada la comprobación de que el valor que se quiere proporcionar a dicho atributo es correcto.

4.2.1. Configuración

Este módulo es el encargado de leer la configuración del fichero de texto *darkfinder.conf,* mostrado en la Figura 19[6], donde el usuario puede particularizar las variables de configuración que contempla el programa. Toda la información de configuración se almacena en la clase *Configuration* (cuyos atributos y métodos se recogen en la Tabla 46 y la Tabla 47 del "Anexo D. Diseño de clases"), a la cual acceden el resto de módulos del programa.

Figura 19. Fragmento de fichero de configuración *darkfinder.conf*.



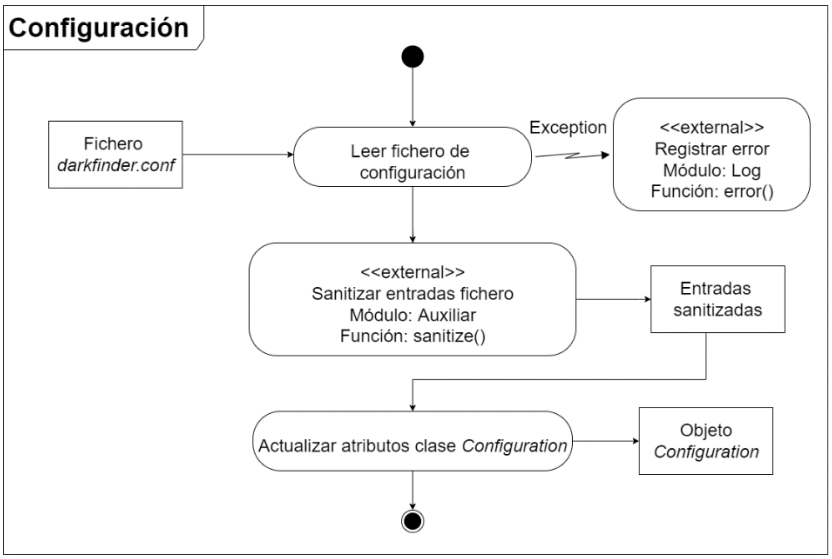
Fuente: Elaboración propia.

[5] Sólo se registran imágenes cuyo tamaño supera el mínimo especificado en el fichero de configuración.
[6] Se recoge de manera completa en el "Anexo I. Código fuente".

La Figura 20 muestra el diagrama UML de actividades del módulo Configuración. El módulo lee los parámetros configurados por el usuario en el fichero de texto *darkfinder.conf* y los carga en los atributos un objeto de tipo *Configuration* por medio de los métodos *set* indicados en el "Anexo D. Diseño de clases".

Con el objeto de poder realizar la **traducción de los textos a diferentes idiomas**, la clase *Configuration* contiene un atributo denominado *trans* que contiene un objeto de tipo *translation* con el contenido necesario para poder realizar adecuadamente dichas traducciones. Por ello, se realiza la configuración de esta clase en función del idioma configurado por el usuario por medio del atributo *locale* en el fichero *darkfinder.conf*.

Figura 20. Diagrama UML actividades módulo Configuración.



Fuente: Elaboración propia.

Tal y como se recoge en la Figura 17, este módulo proporciona un interfaz denominado *Configuration*, compuesto por el constructor de la clase del mismo nombre y cuyo detalle se proporciona en la Tabla 10.

Tabla 10. Interfaz módulo *Configuration*.

Interfaz	Parámetro(s) entrada	Descripción
Configuration	String (*conf_file*): nombre del fichero de configuración. Por defecto: *darkfinder.conf*.	Constructor que devuelve un objeto de tipo *Configuration* con la información cargada en el fichero de configuración *darkfinder.conf*.

Fuente: Elaboración propia.

En caso de que se produzca un error se registra el correspondiente log mediante las funciones del módulo Log, especificado en el capítulo "4.2.4 Log".

4.2.2. DarkFinder

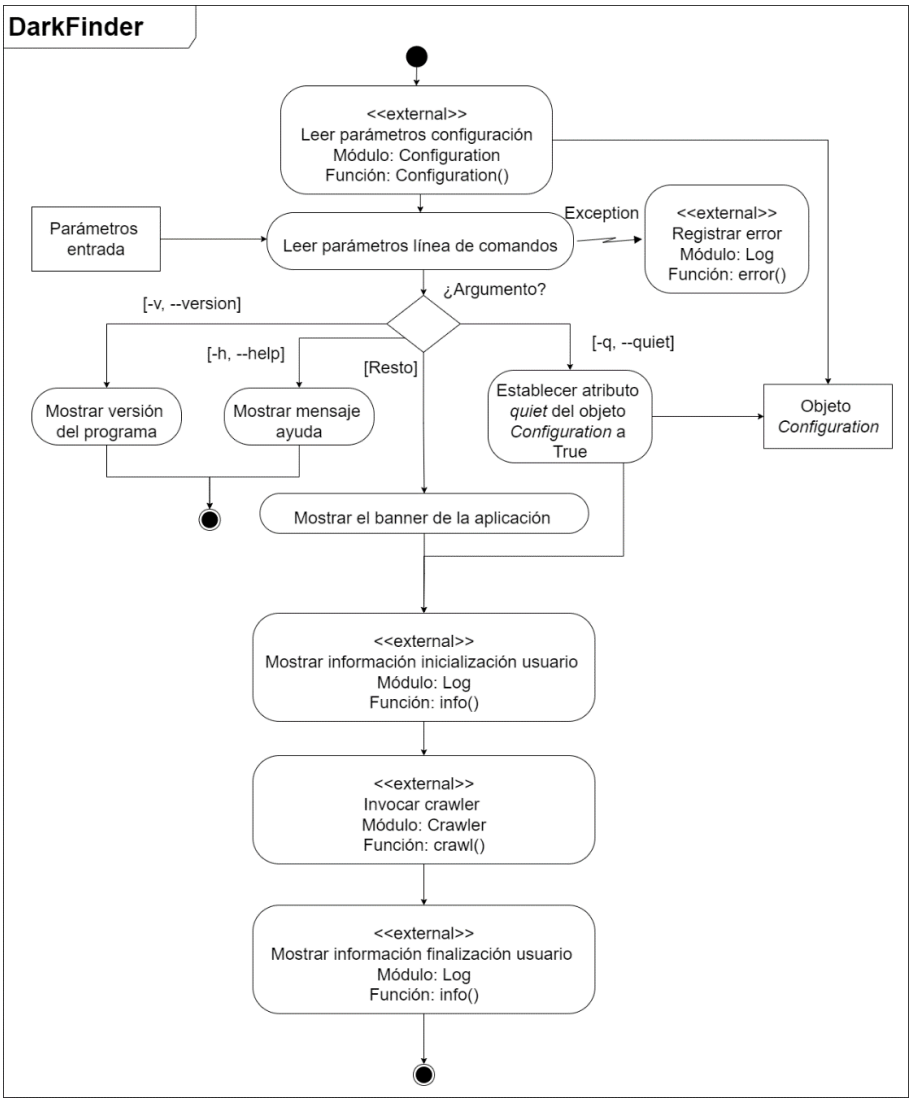
Módulo principal del caso de uso para el rastreo de páginas web. El usuario puede especificar los parámetros de entrada por línea de comandos, según lo especificado en la Tabla 11. En función de los parámetros indicados por el usuario se realizarán las acciones correspondientes, según lo indicado en la Figura 21.

Tabla 11. Parámetros línea de comandos módulo DarkFinder.

Parámetro	Descripción
-h, --help	Muestra mensaje de ayuda.
-v, --version	Muestra mensaje con la versión de la aplicación.
-q, --quiet	No muestra mensajes de tipo '*debug*' y '*warning*' por pantalla.

Fuente: Elaboración propia.

Figura 21. Diagrama UML actividades módulo DarkFinder.



Fuente: Elaboración propia.

Para mostrar mensajes a usuario se hace uso de las funciones del módulo Log, según lo establecido en el capítulo “4.2.4 Log”.

4.2.3. Crawler

Se trata del módulo que realiza el rastreo de páginas web por medio de un proceso recursivo que, partiendo de las URL semilla, recoge de manera sistemática los hiperenlaces a otras webs, obteniendo información relativa a la web (URL, red en la que se encuentra, fecha de rastreo, texto HTML) y a las imágenes contenidas en dicha página (nombre del fichero, hashes -MD5, SHA1, *difference hash*- y metadatos EXIF). Posteriormente, visita las webs referenciadas por las URL semilla, descargando toda la información y volviendo a realizar el proceso de manera recursiva hasta que se llegue a una profundidad máxima de rastreo o bien se hayan recorrido un número máximo de páginas, parámetros configurables por el usuario y obtenidos mediante el módulo de Configuración.

El módulo permite conexión tanto a la web visible como a las Dark Nets TOR, I2P y Freenet. La conexión a las diferentes redes se realiza de manera **transparente por medio de proxies**, según lo indicado a continuación y recogido en la Tabla 12 y la Figura 22:

- Red visible: la conexión se realiza directamente, sin requerir proxy.
- Red TOR: la conexión se realiza a través del servicio de conexión a la red TOR proporcionado por el browser de conexión a esta red usando un proxy SOCKS5 (**Privoxy** versión 3.0.33) que pasa las peticiones encaminadas al puerto 8118 (puerto por defecto de Privoxy) al puerto 9150. Esto se realiza así porque el puerto por defecto de TOR espera conexiones proxy SOCKS5 (y no HTTP/HTTPS) en el puerto 9150.
- Red I2P: conexión realizada a través del servicio de conexión a la red I2P proporcionado por el router I2P a través del puerto 4444 para HTTP y 4445 para HTTPS.
- Red Freenet: conexión realizada a través del servicio de conexión a la red Freenet a través del puerto 8888.

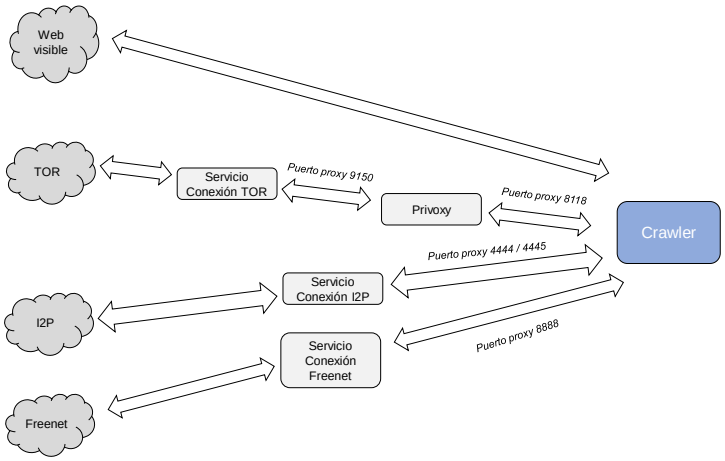
Tabla 12. Puertos de conexión a proxy en función de la red.

Red	Proxy a red
Web visible	-
TOR	localhost:8118
I2P	localhost:4444 / localhost:4445
Freenet	localhost:8888

Fuente: Elaboración propia.

Desde este modo, para la conexión a las Dark Nets contempladas en el presente TFM, se ejecutan en el cliente los servicios de conexión a redes indicados, los cuales proporcionan un proxy en los puertos especificados en la Figura 22.

Figura 22. Proxies de conexión a redes.

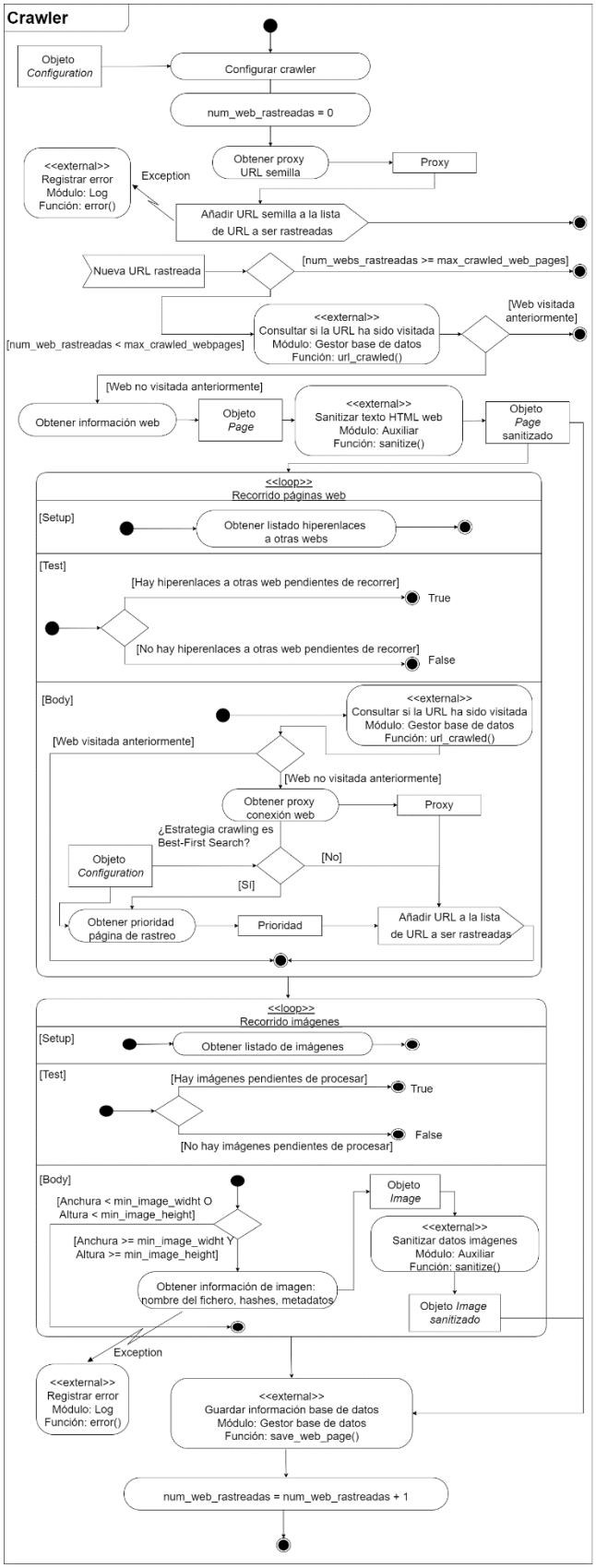


Fuente: Elaboración propia.

La Figura 23 muestra el diagrama UML de actividades del módulo Crawler, el cual parte de las semillas URL y recorre recursivamente las webs encontradas hasta que se llegue a una profundidad máxima de rastreo o se llegue a un número máximo de URLs visitadas (tanto las URL semillas, como la profundidad y número máximo de páginas web rastreables son parámetros de configuración especificados en fichero *darkfinder.conf* y almacenados en un objeto de tipo *Configuration* generado en el módulo de Configuración).

Teniendo en cuenta que la aplicación tiene un carácter distribuido y que diferentes *crawlers* pueden estar operando en diferentes máquinas al mismo tiempo, por cada URL se realiza una consulta a base de datos para verificar si ha sido analizada con anterioridad, en cuyo caso se omite.

Figura 23. Diagrama de actividades UML módulo Crawler.



Fuente: Elaboración propia.

Por cada página web se recoge información relativa a su URL, red en la que se aloja, fecha/hora de rastreo y texto del cuerpo HTML. Asimismo, se obtiene información de las imágenes contenidas, incluyendo nombre del fichero, hashes de la imagen y metadatos EXIF. Según lo indicado en el capítulo "2.5 Trabajos relacionados", un cambio ligero en una imagen produce una alteración completa de su hash, por lo que se contempla el almacenamiento de hash de imagen mediante el algoritmo ***difference hash***, de modo que permita realizar búsquedas de imágenes con ligeras modificaciones o similares a un determinado patrón. Para ello se utiliza la **distancia de Hamming**[7], de modo que se buscan aquellas imágenes cuyo *difference hash* esté a una distancia de Hamming determinada respecto a un patrón de referencia.

Teniendo en cuenta el carácter penal que pudiera tener material de abuso infantil, no se almacenan imágenes en el disco duro.

Para almacenar e intercambiar la información relativa a webs e imágenes, el módulo genera instancias de objetos de las clases *Page* (relativa a la página web, estando sus atributos y métodos definidos en la Tabla 48 y la Tabla 49 del "Anexo D Diseño de clases") e *Image* (relativa a las imágenes de una web, estando sus atributos y métodos definidos en la Tabla 50 y la Tabla 51 del "Anexo D Diseño de clases") que serán utilizadas por el módulo gestor de base de datos para almacenar la información.

Este módulo proporciona un interfaz denominado *crawl*, según se recoge en la Figura 17, encargado de configurar y lanzar el rastreador, con el detalle proporcionado en la Tabla 13.

En caso de que se produzca un error se registra el correspondiente log mediante las funciones del módulo Log, especificados en el capítulo "4.2.4 Log".

Tabla 13. Interfaz módulo Crawler.

Interfaz	Parámetro(s) entrada	Descripción
crawl	Configuration (*conf*): objeto de tipo *Configuration* que contiene los parámetros de configuración del programa incluidos en el fichero *darkfinder.conf*. Boolean (*test*): usado para la realización de pruebas automáticas. Parámetro opcional.	Función que configura el *crawler* y lanza el rastreo por el conjunto de páginas web a analizar, almacenando la información en base de datos.

Fuente: Elaboración propia.

[7] Métrica para comparar dos cadenas de datos binarios. Al comparar dos cadenas binarias de igual longitud, la distancia de Hamming es el número de posiciones de bits en las que los dos bits son diferentes.

4.2.4. Log

Este módulo se encarga de almacenar en el fichero de log *darkfinder.log* y mostrar por pantalla los mensajes de error y aviso generados a lo largo de la aplicación.

El log almacenado en fichero tiene la siguiente estructura:

[*DD/MM/YYYY HH:MM:SS*] – [*nivel*] - Usuario:[*usuario*] – [*mensaje*] - Módulo:[*módulo*] - Función:[*función*]

Siendo:

[*DD/MM/YYYY HH:MM:SS*]: fecha y hora.

[*nivel*]: nivel de mensaje ('*Debug*', '*Info*', '*Warning*', '*Error*', '*Critical*').

[*usuario*]: nombre del usuario que ejecuta la aplicación.

[*mensaje*]: mensaje a ser mostrado (generado a partir de los parámetros *message, message-start y mesage_end*).

[*módulo*]: nombre del módulo en el que se genera el log (parámetro *module*).

[*función*]: nombre de la función en la que se genera el log (parámetro *function*).

Los mensajes de log tienen en cuenta las preferencias de idioma seleccionadas por el usuario e introducidas en el capítulo "4.2.1 Configuración". La Tabla 14 recoge los interfaces generados en el módulo *Log*. Con el objeto de evitar que los textos se traduzcan en cada módulo de manera particularizada a lo largo de todo el programa antes de invocar al módulo *Log*, este proceso de traducción se centraliza en este módulo. Teniendo en cuenta que la traducción se realiza a partir de literales (en pares inglés-español), y para proporcionar mayor flexibilidad, se divide el mensaje a incluir en el log en tres componentes (*message_start, message* y *message_end*). De este modo, si por ejemplo se tiene que traducir el mensaje ('*Error in module: Auxiliary*') se puede dividir en los bloques '*Error in module:*' (traducido por '*Error en el módulo:*') y '*Auxiliary*'. Se establece por tanto una correspondencia entre los literales '*Error in module:*' en inglés y '*Error en el módulo:*' en español. Si posteriormente se debe mostrar el mensaje '*Error in module: Crawler*' no sería necesario volver a traducir '*Error in module:*' al haberse realizado con anterioridad.

Tabla 14. Interfaces módulo *Log*.

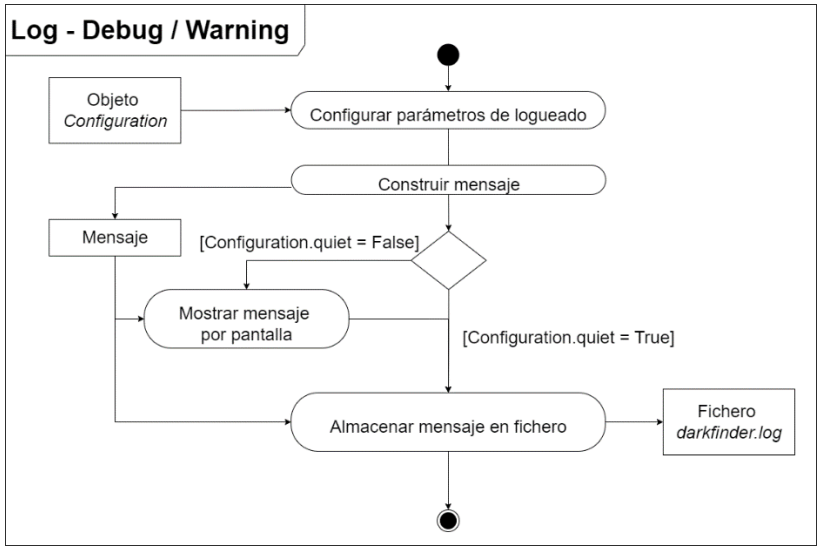
Interfaz	Parámetro(s) entrada	Descripción
debug	Configuration (*conf*): objeto de tipo *Configuration* que contiene los parámetros de configuración del programa incluidos en el fichero *darkfinder.conf*. String (*message*): parte central del mensaje a ser mostrado (inglés). El texto de este parámetro se introduce en inglés y se muestra en pantalla/fichero traducido en función de la variable *locale* incluida en el objeto *Configuration*. String (*module*): nombre del módulo en el que se produce el aviso / error. Parámetro opcional. String (*function*): nombre de la función incluida en el módulo anterior en el que se produce el aviso / error. Parámetro opcional. String (*start_message*): inicio del mensaje a ser mostrado (inglés). Se traduce siguiendo las consideraciones del parámetro *message*. Parámetro opcional. String (*end_message*): fin del mensaje a ser mostrado (inglés). Se traduce siguiendo las consideraciones del parámetro *message*. Parámetro opcional.	Función que muestra por pantalla y almacena en fichero log un mensaje de texto con criticidad '*debug*'.
warning		Función que muestra por pantalla y almacena en fichero log un mensaje de texto con criticidad '*warning*'.
info		Función que muestra por pantalla y almacena en fichero log un mensaje de texto con criticidad '*info*'.
error		Función que muestra por pantalla y almacena en fichero log un mensaje de texto con criticidad '*error*'.
critical		Función que muestra por pantalla y almacena en fichero log un mensaje de texto con criticidad '*critical*'.

Fuente: Elaboración propia.

Nota: Respecto a la traducción de los parámetros message, start_message y end_message debe tenerse en cuenta que, si se introduce un literal que no tiene correspondencia en español, el valor mostrado será directamente el propio mensaje.

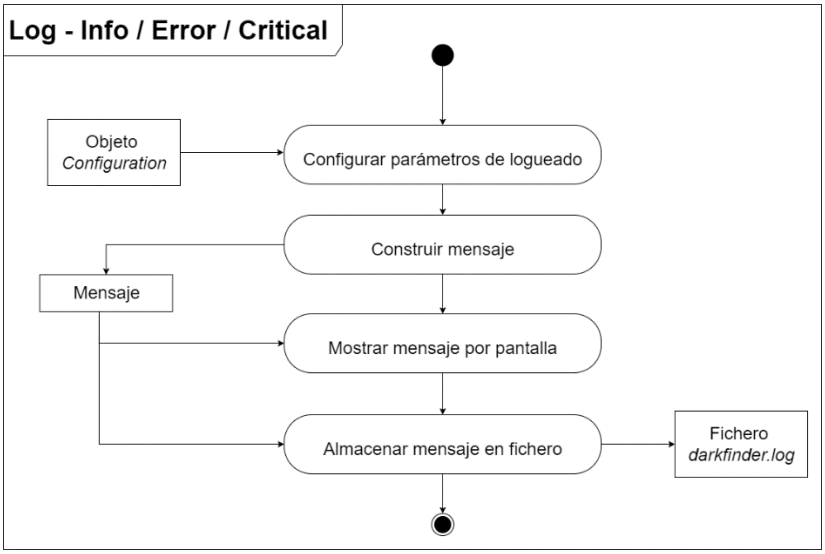
La Figura 24 y la Figura 25 muestran el diagrama UML de actividad para el módulo *Log*. Nótese que el mensaje en pantalla puede omitirse en el caso de notificaciones de tipo '*Debug*' y '*Warning*' si el usuario especifica la opción [-q, --quiet] por línea de comandos.

Figura 24. Diagrama UML actividad módulo Log – funciones 'd*ebug*', '*warning*'.



Fuente: Elaboración propia.

Figura 25. Diagrama UML actividad módulo Log – funciones '*info'*, '*error', 'critical'*.



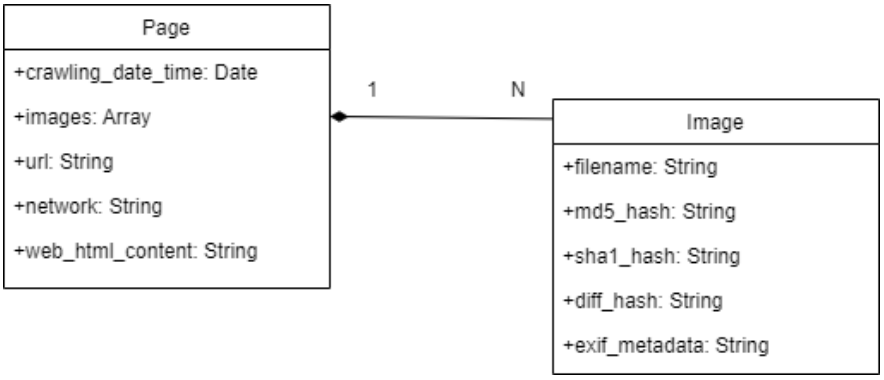
Fuente: Elaboración propia.

4.2.5. Gestor de base de datos

El módulo gestor de base de datos es el encargado de interactuar con la base de datos no relacional, aislando al resto del programa. La Figura 26, la Tabla 15 y la Tabla 16 muestran el diagrama de la base de datos, incluyendo la definición de tipos de datos MongoDB especificada en (MongoDB, Inc., s. f.-b). Se contemplan dos entidades:

- *Page*: almacena la información relativa a la página web.
- *Image*: almacena información relativa a las imágenes contenidas en la página web.

Figura 26. Diagrama Base de Datos No Relacional.



Fuente: Elaboración propia.

Tabla 15. Entidad *Page*.

Atributo	Tipo	Descripción
crawling_date_time	Date	Fecha/hora en la que se realizó el rastreo.
images	Array	Array de documentos (con el contenido de la entidad *Image* especificado en la Tabla 16) conteniendo información relativa a las imágenes de la página web cuya información debe ser almacenada.
url	String	URL de la página web rastreada.
network	String	Tipo de red correspondiente al atributo *url*: 'tor', 'i2p', 'freenet', 'visible'.
web_html_content	String	Contenido HTML de la página web rastreada.

Fuente: Elaboración propia.

Tabla 16. Entidad *Image*.

Atributo	Tipo	Descripción
filename	String	Nombre del fichero correspondiente a la imagen.
md5_hash	String	Hash MD5 de la imagen.
sha1_hash	String	Hash SHA1 de la imagen.
diff_hash	String	*Difference hash* de la imagen.
exif_metadata	String	Metadatos de la imagen en formato 'clave': 'valor'.

Fuente: Elaboración propia.

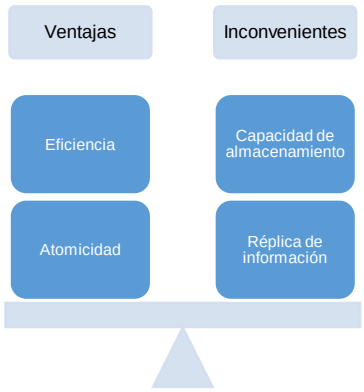
Para el almacenamiento de la información se contempla la **creación de una única colección** (*Collection*, el equivalente a una tabla en una base de datos relacional). Con ello, la relación entre las entidades *Page* e *Image* es de tipo *Embedded*, de modo que la información de las imágenes se almacena conjuntamente con la relativa a la página web a la que pertenecen. De este modo, asociado a la información de la web se genera un array de objetos compuesto por tantos elementos como imágenes existan y con el contenido de la entidad *Image*. Esta

propuesta presenta diversas ventajas con respecto a la creación de colecciones *Page* e *Image* separadas y referenciadas (relación tipo *Referenced,* que sería un esquema equivalente al empleado en una base de datos relacional):

- La grabación de la información de un documento en la base de datos es **atómica** (que constituye un requisito de la aplicación según lo especificado en el "Anexo A. Requisitos de la aplicación").
- **Eficiencia en la búsqueda y almacenamiento** de datos de entidades relacionadas.

Como contrapartida, se replica la información de una misma imagen almacenada en diversas páginas web, al tiempo que se limita la capacidad de almacenamiento de información para el texto HTML de la página, ya que el total de capacidad de un documento es de 16 MB.

Figura 27. Ventajas e inconvenientes relación tipo *Embedded* en base de datos MongoDB.



Fuente: Elaboración propia.

Este módulo proporciona diversos interfaces al resto de módulos para el acceso a la base de datos, según lo especificado en la Tabla 17.

Tabla 17. Interfaces módulo *Gestor de base de datos*.

Interfaz	Parámetro(s) entrada	Descripción
url_crawled	Configuration (*conf*): objeto de tipo *Configuration* que contiene los parámetros de configuración del programa incluidos en el fichero *darkfinder.conf*. String (*url*): dirección URL de la página web para la que se quiere saber si ha sido rastreada anteriormente.	Función que realiza consulta a la base de datos para saber si una determinada URL de una web está almacenada, con lo que ya habrá sido anteriormente rastreada.

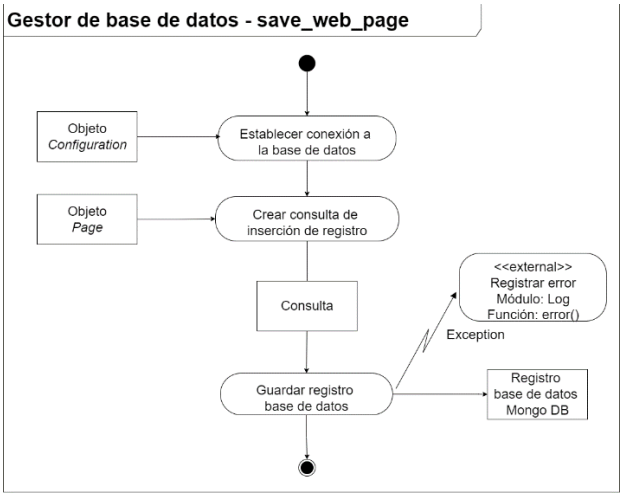
Interfaz	Parámetro(s) entrada	Descripción
save_web_page	Configuration (*conf*): objeto de tipo *Configuration* que contiene los parámetros de configuración del programa incluidos en el fichero *darkfinder.conf*. Page (*page*): objeto instancia de una clase de tipo *Page* conteniendo información de la web a almacenar y las imágenes contenidas.	Función que inserta un documento en la base de datos correspondiente a la página web cuya información es pasada como parámetro por medio de un objeto tipo *Page*.
search_web_page	Configuration (*conf*): objeto de tipo *Configuration* que contiene los parámetros de configuración del programa incluidos en el fichero *darkfinder.conf*. String (*url*): texto con expresión regular que se busca en el campo *url* de la página web. Parámetro opcional. String (*text)*: texto con expresión regular que se busca en campo *web_html_content* de la página web. Parámetro opcional. String (*md5_sha1_hash*): hash que se busca en los campos *md5_hash*, *sha1_hash* de la imagen. Parámetro opcional. Array (d*hash*): array que contiene dos elementos. El primero es un string con el *difference hash* de la imagen tomado como referencia para hacer una búsqueda en base al campo *diff_hash.* El segundo es un entero especificando la distancia Hamming considerada para la búsqueda de imágenes en base al campo *diff_hash*. Parámetro opcional. String (*metadata*): texto con expresión regular que se busca en los metadatos: campo *exif_metadata* de la imagen. Parámetro opcional. Boolean (*test*): usado para la realización de pruebas automáticas. Parámetro opcional.	Función que realiza una consulta a base de datos con campos asociados a página web o imagen. Devuelve uno o varios documentos con los resultados de la consulta. Si se especifican varios parámetros, deberán cumplirse todos ellos para proporcionar respuesta (consulta tipo 'AND').

Fuente: Elaboración propia.

En caso de que se produzca un error en la gestión de la base de datos, se invocan las funciones del módulo Log para su registro, según lo indicado en el capítulo "4.2.4 Log".

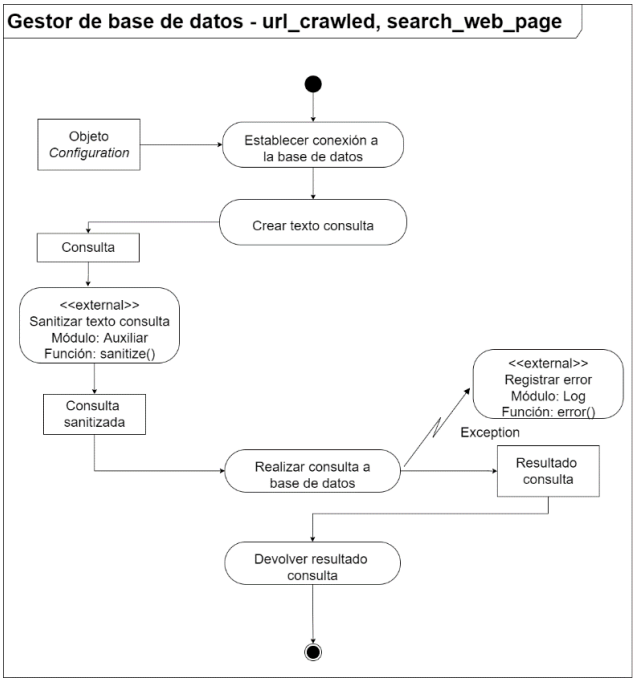
La Figura 28 y la Figura 29 muestran el diagrama UML de las funciones indicadas en la Tabla 17.

Figura 28. Diagrama UML actividad Módulo Gestor base de datos – función 'save_web_page'.



Fuente: Elaboración propia.

Figura 29. Diagrama UML actividad Módulo Gestor base de datos – funciones 'url_crawled', 'search_web_page'.



Fuente: Elaboración propia.

4.2.6. DarkSearch

Módulo principal del caso de uso para la búsqueda de páginas web en base a su contenido o el de las imágenes que contienen. El usuario puede especificar los parámetros de entrada por línea de comando, según lo especificado en la Tabla 18.

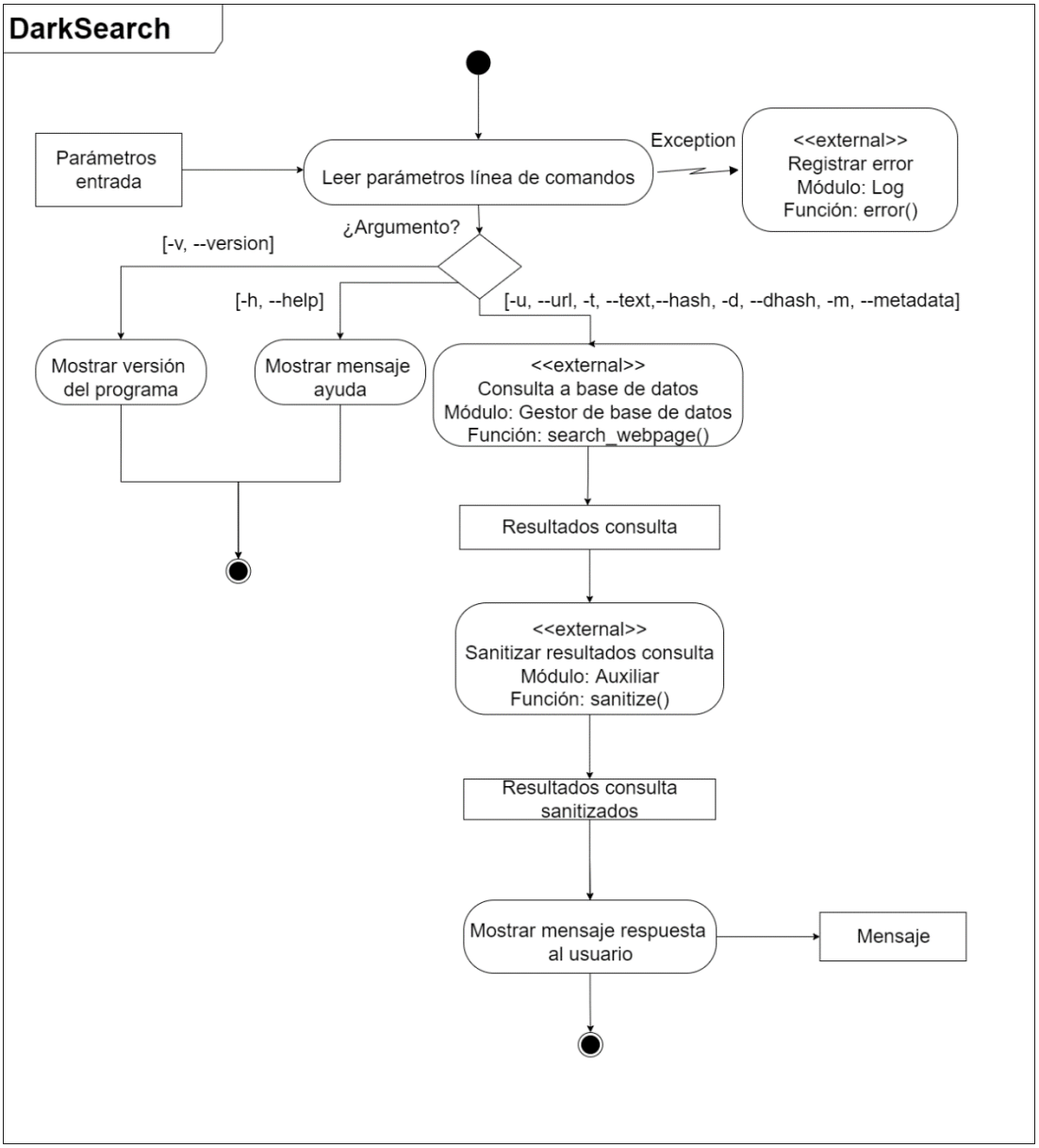
Tabla 18. Parámetros línea de comandos módulo DarkSearch.

Parámetro	Descripción
-h, --help	Muestra mensaje de ayuda.
-v, --version	Muestra mensaje con la versión de la aplicación.
-u, --url [URL]	Devuelve páginas web cuya URL (campo *url*) contenga el texto o expresión regular dada por el parámetro *[URL].*
-t, --text [TEXT]	Devuelve páginas web cuyo texto HMTL (campo *web_html_content*) contiene el texto o expresión regular dada por el parámetro *[TEXT].*
--hash [HASH]	Devuelve páginas web que contengan imágenes cuyo hash MD5/SHA1 coincida con el parámetro *[HASH]*.
-d, --dhash [REF_HASH] [DISTANCE]	Devuelve páginas web que contengan imágenes cuyo *difference hash* tenga una distancia de Hamming con respecto al parámetro *[REF_HASH]* inferior a [*DISTANCE*].
-m, --metadata [METADATA]	Devuelve páginas web que contengan imágenes cuyos metadatos contengan texto o expresión regular que coincida con *[METADATA]*.

Fuente: Elaboración propia.

En función de los parámetros indicados por el usuario se realizarán las acciones correspondientes, según lo indicado en la Figura 30.

Figura 30. Diagrama UML actividades módulo DarkSearch.



Fuente: Elaboración propia.

En caso de que se produzca error se registra el log mediante las funciones del módulo Log, según se recoge en el capítulo “4.2.4 Log”.

4.2.7. Auxiliar

Este módulo contiene funciones auxiliares que son invocadas desde varios módulos del resto del programa. Tal y como se recoge en la Figura 17, proporciona los interfaces cuyo detalle se proporciona en la Tabla 19.

Tabla 19. Interfaces módulo Auxiliar.

Interfaz	Parámetro(s) entrada	Descripción
sanitize	Configuration (*conf*): objeto de tipo *Configuration* que contiene los parámetros de configuración del programa incluidos en el fichero *darkfinder.conf*. String (*input_text)*: texto a sanitizar.	Función que sanitiza un texto de entrada para evitar las amenazas identificadas en el análisis de amenazas contemplado en el capítulo “4.1 Análisis”.
hamming_distance	String (*string1*): primera cadena. String (*string2*): segunda cadena.	Función que calcula la distancia de Hamming entre 2 strings, correspondientes a *difference hash* de imágenes.

Fuente: Elaboración propia.

El módulo hace uso de las funciones del módulo Log para registrar cualquier tipo de error que se produzca, según lo especificado en el capítulo “4.2.4 Log”.

4.3. Codificación

En esta fase se realiza tanto la codificación de los diversos módulos que componen la aplicación en base a los requisitos y diseño establecidos en las fases anteriores como las verificaciones de análisis estático de código fuente y comprobación de ausencia de vulnerabilidades en librerías importadas.

La codificación se realiza mediante entorno **Spyder** versión 5.1.5. Según se indica en la Figura 31, todo el código está **comentado en español e inglés** para facilitar su desarrollo posterior. Asimismo, se verifica el **cumplimiento de las pautas de codificación de código en Python de la guía PEP-8**, ya que la propia aplicación muestra un mensaje de aviso en caso contrario.

El detalle del código fuente está disponible en el “Anexo I. Código fuente”.

Figura 31. Ejemplo de código fuente.

Fuente: Elaboración propia.

A lo largo del capítulo se hace referencia a librerías que son importadas para diversas funcionalidades. Según lo indicado en la Tabla 8 del capítulo "4.2 Diseño", **se importan únicamente aquellas funciones que vayan a ser empleadas en la aplicación** mediante la secuencia [*from module import function*], evitando realizar importaciones globales del tipo *import.**.

Figura 32. Ejemplo de importación de funciones.

```
# -*- coding: utf-8 -*-
# EN: Standard Modules | ES: Módulos estándar
from gettext import translation
from configparser import ConfigParser, Error as configparser_error
from json import loads as json_loads
# EN: DarkFinder Modules | ES: Módulos DarkFinder
from .logger import warning as warning, error as error
from .settings import LOCALE, AVAILABLE_LOCALES, STARTING_URLS
from .settings import MAX_DEPTH_CRAWLING, MAX_CRAWLED_WEB_PAGES
from .settings import MINIMUM_IMAGE_WIDTH, MINIMUM_IMAGE_HEIGHT
from .settings import CRAWLING_STRATEGY, AVAILABLE_CRAWLING_STRATEGIES
from .settings import KEYWORDS, DB_SERVER, DB_USER
from .auxiliary import sanitize
```

Fuente: Elaboración propia.

4.3.1. Configuración

En este módulo se realiza la lectura del fichero de configuración *darkfinder.conf*, para lo cual se hace uso de la **librería *configparser*** nativa de Python. Mediante la función *read* se hace una lectura del fichero de configuración y se obtiene cada uno de los parámetros contenidos en dicho archivo mediante los métodos *get/getint* (en función de si el parámetro es de tipo string o entero). En el módulo se definen los atributos y métodos de la clase *Configuration* (definidos en el "Anexo D. Diseño de clases") que aloja los parámetros de configuración para que sean tenidos en cuenta en toda la aplicación. Adicionalmente a los métodos especificados en esta clase, la Tabla 20 contiene las funciones de este módulo.

Tabla 20. Funciones módulo Configuración.

Función	Descripción
set_database_user_password	Lee la ruta del archivo que contiene usuario y contraseña para conectarse a la base de datos y las almacena en los atributos *db_user* y *db_password* de un objeto tipo *Configuration*.

Fuente: Elaboración propia.

Con el objeto de poder incluir la traducción de los textos en castellano e inglés (y a futuro en otros idiomas), la clase *Configuration* incluye un objeto de tipo *translation* de la librería *gettext* nativa de Python. Como parámetro del constructor de esta clase se incluye el *locale*, variable correspondiente al idioma almacenada en el fichero de configuración. Para realizar las traducciones, la propia clase *Configuration* incluye los métodos *translate* y *ntranslate* referidos en el capítulo "4.2.1 Configuración". La configuración se basa en la definición de un fichero

'*darkfinder_locales.po'* con los literales en los idiomas según la definición dada en (Free Software Foundation, Inc, s. f.) y mostrado en la Figura 34. A partir de este fichero se obtiene un archivo compilado, usado por la función *gettext*, por medio de la herramienta **Poedit** versión 3.0.1.

Figura 33. Ejecución del código en idiomas español e inglés.

Fuente: Elaboración propia.

Figura 34. Extracto de fichero *darkfinder_locales.po* en español.

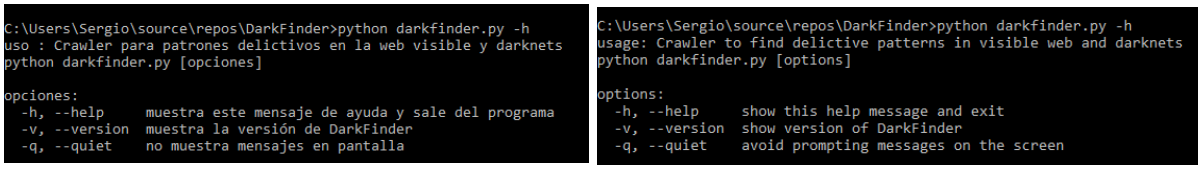
```
#: Module: DarkFinder
#, python-brace-format
msgid "{0} second"
msgid_plural "{0} seconds"
msgstr[0] "{0} segundo"
msgstr[1] "{0} segundos"

#: Module: DarkFinder
msgid "show version of DarkFinder"
msgstr "muestra la versión de DarkFinder"

#: Module: DarkFinder
msgid "avoid prompting messages on the screen"
msgstr "no muestra mensajes en pantalla"

#: Module: DarkFinder
msgid "Number of crawled web pages:"
msgstr "Número de páginas web rastreadas:"

#: Module: DarkFinder
msgid "Error reading arguments"
msgstr "Error leyendo argumentos"
```

Fuente: Elaboración propia.

4.3.2. DarkFinder

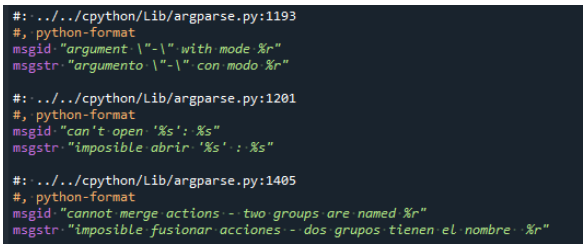
Este es el módulo que recoge los parámetros introducidos por el usuario y lanza el proceso de rastreo invocando al módulo Crawler. La función principal lee el fichero de configuración (módulo Configuración) y los argumentos de entrada establecidos en el capítulo "4.2.2 DarkFinder", para lo cual hace uso de la **librería nativa de Python *argparse***. Para poder **traducir los mensajes de ayuda de la aplicación** (opción -h) mostrados en la Figura 35, a la clase *argparse* se le asigna un objeto *translation* del modo indicado en el capítulo "4.3.1 Configuración". En este caso se ha traducido al español el conjunto de literales de *argparse* en un fichero denominado *argparse_locales.po,* del modo mostrado en la Figura 36.

Figura 35. Visualización de la ayuda del comando *darkfinder* en idiomas español e inglés.



Fuente: Elaboración propia.

Figura 36. Extracto de fichero *argparse_locales.po* en español.



Fuente: Elaboración propia.

Tras mostrar por pantalla un **banner identificativo** de la aplicación **lanza el proceso de rastreo** invocando a la función *crawl* del módulo de Crawler. Se calcula el **tiempo transcurrido en la ejecución del proceso**, formateando el mensaje por medio de la función *format_elapsed_time*. El mensaje visualizado en pantalla se muestra en la Figura 37.

Figura 37. Ejecución del proceso de rastreo.

Fuente: Elaboración propia.

Nota: el número de páginas web rastreadas es 10, siendo únicamente ejemplo representativo, no pretendiendo realizar un rastreo real.

Este módulo contiene las funciones especificadas en la Tabla 21.

Tabla 21. Funciones módulo DarkFinder.

Función	Descripción
banner	Devuelve el banner de la aplicación que será mostrado en pantalla.
get_args	Parsea los argumentos pasados a la aplicación por línea de comandos empleando la librería *argparse*.

Función	Descripción
format_elapsed_time	Función que formatea un tiempo transcurrido (segundos) a números enteros correspondientes a días, horas, minutos y segundos.
darkfinder	Función principal de la aplicación *DarkFinder*.

Fuente: Elaboración propia.

4.3.3. Crawler

En este módulo se realiza el rastreo de las páginas web, para lo cual se hace uso de la **librería externa scrapy** versión 2.6.1. Se emplea esta librería por las siguientes ventajas:

- Completo, potente y extensible framework de rastreo *open source*.
- Permite gestión de conexiones asíncronas.
- Proporciona funcionalidades de parseo de los ficheros HTML obtenidos como respuesta.
- Soporta múltiples opciones de configuración.

Para lanzar el proceso de rastreo se crea un objeto *CrawlerProcess* a partir del constructor proporcionado por la librería, pasando como parámetro un diccionario con los parámetros de configuración necesarios.

El **establecimiento de la estrategia de rastreo** se realiza de la siguiente manera en función del valor que se configure en el fichero de configuración.

- **Depth-First Search** (DFS): es el comportamiento por defecto de la librería *scrapy*.
- **Breadth-First Search** (BFS): modificación de los parámetros de configuración del objeto *CrawlerProcess* de la librería *scrapy*.
- **Best-First Search**: se calcula una prioridad para la página objetivo de rastreo en la función *webpage_link_priority* en función del número de veces que aparecen en la página web 'padre' un conjunto de palabras clave o *keywords* introducidas en el fichero de configuración. A este valor se le suma el total de apariciones de las palabras clave en el propio enlace que lleva a la web destino. Ambos valores se multiplican por un valor constante definido en el fichero settings.py.

*prioridad = WEBPAGE_PRIORITY_MULT_FACTOR_KEYWORD_IN_BODY * body_count +*

*WEBPAGE_PRIORITY_MULT_FACTOR_KEYWORD_IN_LINK * link_count*

Siendo:

- *WEBPAGE_PRIORITY_MULT_FACTOR_KEYWORD_IN_BODY*: factor multiplicador para el número de veces que aparece la palabra clave en el cuerpo HTML de la web 'padre'.

- *body_count*: número de veces que aparece la palabra clave en el cuerpo HTML de la web 'padre'.
- *WEBPAGE_PRIORITY_MULT_FACTOR_KEYWORD_IN_LINK:* factor multiplicador para el número de veces que aparece la palabra clave en el enlace HTML a la web destino.
- *link_count:* número de veces que aparece la palabra clave en el enlace HTML a la web destino.

Para establecer la lógica de rastreo es necesario sobrescribir los métodos *start_requests* y *parse* de la clase *DarkFinderSpider*. Esta es derivada de la clase *Spider* de la librería *scrapy* y se pasa como parámetro al método *crawl* del objeto *CrawlerProcess*. En el método *start_request* se configuran las URL semilla configuradas en el fichero de configuración. Para poder conectarse a las diferentes redes ('visible', 'tor', 'i2p, 'freenet') se determina por medio de la función *connection_proxy_url* el proxy al que se debe conectar según lo establecido en el capítulo "4.2.3 Crawler". Para determinar la red a la que corresponde cada URL se hace uso de patrones de expresiones regulares, establecidas en el fichero settings.py.

El **crawler funciona de manera asíncrona**, de modo que cuando se obtenga respuesta se invoca al método *parse* de la clase *DarkFinderSpider*. En el mismo se procesa el documento HTML devuelto correspondiente a la página web rastreada por medio de las funciones proporcionadas por la librería *scrapy*, implantando el algoritmo especificado en la Figura 23 en el capítulo "4.2.3 Crawler":

- Obteniendo todos los enlaces a otras páginas web. Por cada uno de ellos se identifica la red a la que pertenece por medio de la función *connection_proxy_url*, del modo indicado anteriormente.
- Extrayendo todas las imágenes contenidas en la página web. Sólo se procesan aquellas imágenes cuyo tamaño supere un determinado valor especificado en el fichero de configuración. Se descargan las imágenes sin guardarlas en disco por medio de la función *get* de la **librería nativa de Python *requests***. La obtención de atributos de las imágenes (anchura, altura, metadatos, etc.) se realiza haciendo uso de la **librería externa *pillow*** versión 9.1.0, que proporciona importantes funcionalidades para este fin. Para el cálculo de los hashes MD5 y SHA1 de las imágenes se hace uso de la **librería *hashlib*** nativa de Python. Para la obtención del *difference hash* se emplea la **librería externa *imagehash*** versión 4.2.1, que permite realizar este cálculo particular.

La información de imágenes obtenida en el proceso de rastreo se almacena en un objeto tipo *Image* por medio de las funciones *set* indicadas en el capítulo "4.2.3 Crawler". Para cada página web rastreada, se almacena junto con un array correspondiente a las diferentes

imágenes contenidas en la misma, la información propia de la página en un objeto tipo *Page*, según lo establecido en el citado capítulo. Las funciones *set* que establecen el valor a almacenar en la base de datos realizan comprobaciones de tipo y valor específicas y llevan a cabo una sanitización de los parámetros de entrada por medio de la función *sanitize* del módulo Auxiliar.

Para mostrar el **progreso del proceso de rastreo,** la aplicación hace uso de la librería externa ***progressbar2*** versión 4.0.0 (Van Hattem, 2022), que permite particularizar la información visualizada para informar del proceso. De este modo, el rastreador identifica el número de páginas procesadas, grado de avance, tiempo transcurrido y tiempo pendiente estimado para la finalización del rastreo, del modo indicado en la Figura 38. El mensaje se muestra traducido al español o inglés en función del valor especificado en el fichero de configuración.

Figura 38. Barra de progreso en el proceso de rastreo.

Fuente: Elaboración propia.

La Tabla 22 recoge las funciones definidas en el módulo Crawler adicionales a las interfaces con otros módulos ya recogidas en el capítulo “4.2.3 Crawler”. Adicionalmente, en este módulo se modifican e incluyen diversos métodos de la clase *DarkFinderSpider* para adecuar el proceso el rastreo a los requisitos de la aplicación.

Tabla 22. Funciones módulo Crawler.

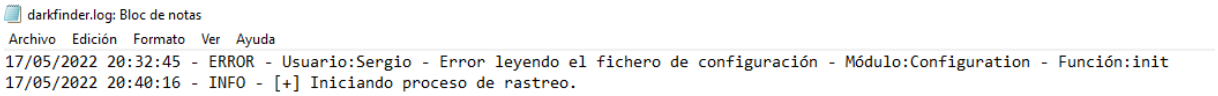
Función	Descripción
connection_proxy_url	Devuelve el tipo de red que corresponde a una URL: (i) Visible, (ii) TOR, (iii) I2P, (iv) Freenet También proporciona el proxy que se proporcionará al rastreador scrapy para conectarse a la red.
is_url	Verifica si una URL tiene un patrón válido correspondiente a (i) web visible, (ii) web TOR, (iii) web I2P, (iv) web Freenet.
webpage_link_priority	Devuelve la prioridad de un enlace a una página web mediante un algoritmo de tipo Best First Search. Utiliza las palabras clave proporcionadas en el archivo de configuración.

Fuente: Elaboración propia.

4.3.4. Log

Este módulo proporciona funciones cuyo objetivo es almacenar en un archivo log y/o mostrar por pantalla los mensajes que se pasen como parámetro, según lo establecido en el capítulo “4.2.4 Log”. Para ello se hace uso de la **librería *logging*** nativa de Python. Los logs se almacenan en español / inglés en función de la variable *locale* almacenada en el fichero de configuración.

Figura 39. Extracto de fichero de log.



Fuente: Elaboración propia.

Con carácter general, se capturan los posibles errores mediante sentencias ***try-except***, mostrando por pantalla y registrando en log un mensaje de error en tal caso.

Figura 40. Captura de errores mediante bloques try-except.

```
try:
    records = collection.find(query_filter)
except Exception:
    error(conf, message="Error making query to database",
          module='database_manager', function='search_webpage')
    return False
```

Fuente: Elaboración propia.

Este módulo contiene, además de los interfaces especificados en el capítulo “4.2.4 Log”, las funciones indicadas en la Tabla 23.

Tabla 23. Funciones módulo Log.

Función	Descripción
config_logger	Configuración del *logger*. Establece el formato del log en función de la variable '*locale*' especificada en el fichero de configuración.
complete_message	Establece el mensaje a ser registrado en el fichero de logs o mostrado por pantalla.

Fuente: Elaboración propia.

4.3.5. Gestor de base de datos

En este módulo se realizan las consultas a la base de datos no relacional MongoDB, para lo cual se hace uso de la librería externa ***pymongo*** versión 4.1.1. El contenido principal de este módulo es proporcionar los interfaces especificados en el capítulo “4.2.5 Gestor de base de datos”. Para la realización de la búsqueda de páginas webs almacenadas en la base de datos que responden a los criterios de búsqueda introducidos como argumentos en el comando *darksearch*, se hace uso de la función *find* de la librería *pymongo* (MongoDB, Inc., s. f.-c). Para ello se genera un filtro que corresponde a un diccionario cuya primera clave es ‘*$and*’ (como se indica en el capítulo “4.2.5 Gestor de base de datos”, si se especifican varios parámetros, deberán cumplirse todos ellos para proporcionar respuesta). El valor correspondiente a esta primera clave es un array en el que por cada uno de los argumentos de búsqueda empleados se añade una nueva entrada al filtro. En aquellos casos en los que se debe hacer uso de **expresiones regulares**, el valor correspondiente al campo incluye un

nuevo diccionario con la palabra clave '*$regex*', especificando la opción *-i* para que la búsqueda **no distinga entre mayúsculas y minúsculas** (*case insensitive*). Para la búsqueda de campos relacionados con imágenes, al estar almacenadas como un array dentro de la información correspondiente a la página, se hace uso de la palabra clave '*$elemMatch*'.

Para la grabación de datos se hace una **conversión de los objetos** de tipo *Page* e *Image* obtenidos en el módulo Crawler **a tipo diccionario**, de modo que se pueda grabar directamente en la base de datos MongoDB. De manera previa, se eliminan aquellos campos auxiliares que no sea necesario almacenar (objeto tipo *Configuration*). El resultado se muestra en la Figura 41:

Figura 41. Ejemplo de visualización de datos almacenados en base de datos MongoDB.

Fuente: Elaboración propia.

En caso de que se deba realizar búsqueda basada en el campo *difference hash* de una imagen, se realiza un recorrido por todos los valores de este campo en la función *get_difference_hashes_from_reference_distance*, construyendo un array con todos los valores cuya distancia de Hamming (calculada mediante la función *hamming_distance* del módulo Auxiliar) al valor pasado como referencia ([*REF_HASH*]) sea menor al segundo parámetro introducido por el usuario ([*DISTANCE*]). Este conjunto de hashes se introduce como filtro de búsqueda por medio de palabra clave '*$in*'.

En todos los casos, los nombres de los campos se especifican en el propio código en base al valor indicado en el módulo settings.py, evitando inyecciones de tipo NoSQL. Asimismo, los datos han sido sanitizados haciendo uso de la función *sanitize* del módulo Auxiliar antes de ser almacenados en la base de datos.

Este módulo, además de los interfaces especificados en el capítulo "4.2.5 Gestor de base de datos", contiene las funciones especificadas en la Tabla 24.

Tabla 24. Funciones módulo Gestor de base de datos.

Función	Descripción
get_db_connection	Obtiene una conexión a la base de datos MongoDB.

Función	Descripción
get_difference_hashes_ from_reference_distance	Devuelve un subconjunto de los *difference hash* almacenados en base de datos que tienen una distancia de Hamming determinada al *difference hash* de referencia pasado como parámetro. Permite buscar imágenes 'similares', ya que hashes MD5/SHA1 de ficheros similares son completamente diferentes.
add_image_field_filter	Completa un filtro para construir una consulta No-SQL, correspondiente a un campo relativo a un atributo de imagen. Reduce complejidad cognitiva en la función *'search_webpage*'.
add_page_field_filter	Completa un filtro para construir una consulta No-SQL, relativa a un campo asociado a un atributo de página web. Reduce la complejidad cognitiva en la función '*search_webpage*'.

Fuente: Elaboración propia.

4.3.6. DarkSearch

En este módulo se lanza la búsqueda de aquellos registros almacenados en base de datos que cumplen las condiciones de búsqueda introducidas por el usuario por línea de comandos, mostrando el resultado por pantalla. Al igual que lo especificado en el capítulo “4.3.2 DarkFinder”, la obtención de argumentos pasados por línea de comandos se realiza por medio de la librería nativa Python *argparse*. Del mismo modo, se soporta que la ayuda (opción -h) se muestre en inglés y español, según lo mostrado en la Figura 42.

Figura 42. Visualización de la ayuda comando *darksearch* en idiomas español e inglés.

Fuente: Elaboración propia.

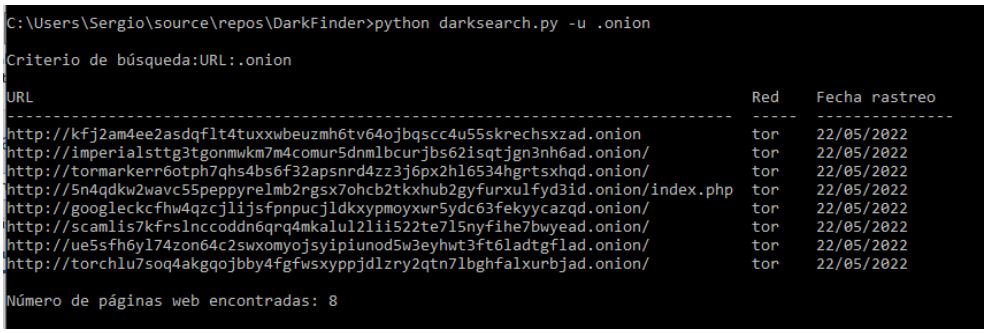
El módulo realiza una búsqueda de los registros que cumplen las condiciones introducidas por el usuario por línea de comandos por medio de la función *search_web_page* del módulo Gestor de base de datos y obtiene el contenido de los resultados a mostrar por pantalla por medio de la función *print_results*.

La visualización muestra en **modo texto comprensible por el usuario los criterios de búsqueda** introducidos por línea de comandos por medio de la función *get_search_criteria*. Los **resultados se muestran por pantalla conveniente tabulados**, del modo mostrado en la Figura 43. Para ello se hace uso de la **librería externa *tabulate*** versión 0.8.9 (Python Software Foundation, 2021). De modo general se muestran los campos correspondientes a la URL, red (‘visible, ‘tor’, ‘i2p‘, ‘freenet’) y fecha de rastreo. El módulo permite realizar **búsquedas mediante patrones de expresiones regulares**, como se muestra en la Figura 44 para el caso de un patrón correspondiente a **correo electrónico**.

Adicionalmente, en el caso de que la búsqueda contenga algún **criterio relativo a imágenes, se muestra el nombre de fichero correspondiente,** según lo mostrado en la Figura 45. Teniendo en cuenta que las imágenes se almacenan como array asociado a la información de cada página, se realiza un filtrado, **mostrando sólo aquellas imágenes de la página que cumplen la condición**. En caso de que la condición de búsqueda haga referencia al *difference hash*, también se **muestra la distancia de Hamming entre cada imagen encontrada y el valor dado de referencia**, del modo mostrado en la Figura 46. Esto permite conocer la similitud entre las imágenes rastreadas y el valor proporcionado de referencia.

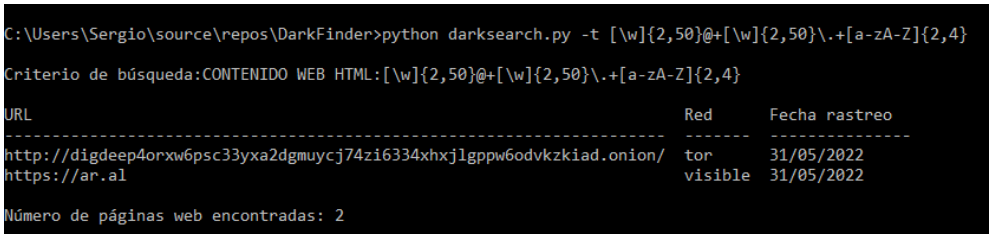
En ambos casos se visualiza el **número de páginas y, en su caso imágenes, que cumplen la(s) condición(es) especificada(s)**.

Figura 43. Visualización de resultados comando *darksearch.*



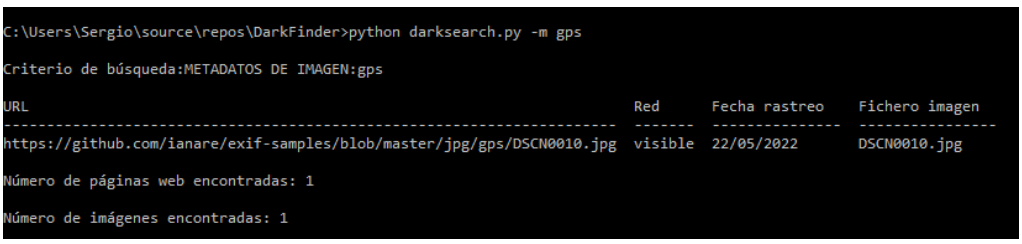
Fuente: Elaboración propia.

Figura 44. Visualización de resultados comando *darksearch.* Expresión regular para búsqueda de correos electrónicos.



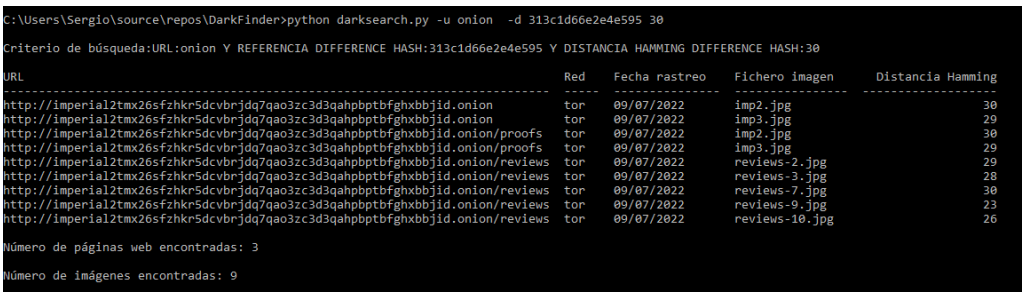
Fuente: Elaboración propia.

Figura 45. Visualización de resultados comando *darksearch* con criterios de imágenes.



Fuente: Elaboración propia.

Figura 46. Visualización de resultados comando *darksearch* con criterios de *difference hash* relativos a imágenes.



Fuente: Elaboración propia.

La Tabla 25 muestra las funciones definidas en el módulo DarkSearch.

Tabla 25. Funciones módulo DarkSearch.

Función	Descripción
get_args	Parsea los argumentos pasados a la aplicación por línea de comandos empleando la librería *argparse*.
darksearch	Función principal del módulo DarkSearch. Hace una búsqueda en base de datos según los parámetros especificados por el usuario por línea de comandos.
get_search_criteria	Proporciona un string con las condiciones de búsqueda introducidas por el usuario por línea de comandos.
get_print_results	Devuelve el mensaje a imprimir en pantalla con los resultados de una búsqueda convenientemente tabulados.
add_item_to_search_criteria	Añade un elemento a una cadena de búsqueda con un texto que contiene el criterio especificado por el usuario mediante el comando '*darksearch*'. Usada para reducir la complejidad cognitiva en '*get_search_criteria*'
set_result_images	Añade a un array '*data*' los datos de la imagen en proceso si se ajustan a los criterios de búsqueda para que sea mostrada por pantalla. Usada para reducir la complejidad cognitiva en '*get_print_results*'.

Fuente: Elaboración propia.

4.3.7. Auxiliar

En este módulo se define la función *sanitize*, que filtra un valor de entrada pasado como parámetro haciendo uso de la función *escape* de la librería interna de Python ***html***. De este modo, se eliminan metacaracteres (como '<', '>') que pudieran conducir a vulnerabilidades por una interpretación inadecuada de código.

Asimismo, este módulo incluye la función *hamming_distance*, encargada de calcular la distancia de Hamming entre 2 strings.

4.3.8. Settings

El fichero settings.py recoge todo un conjunto de valores almacenados como constantes y que son utilizados a lo largo de todo el programa, incluyendo:

- Versión del programa.
- Nombre del fichero de configuración.
- Valores por defecto del fichero de configuración (usados en caso de error en su lectura) y valores admisibles en algunos de los parámetros de configuración (listas blancas).
- Parámetros asociados al proceso de logging.
- Patrones de expresión regular para determinar el tipo de red en función de la URL: (i) Visible, (ii) TOR, (iii) I2P HTTP, (iv) I2P HTTPS, (v) Freenet.
- Parámetros relacionados con la estrategia de rastreo.
- Nombres de los campos en la base de datos MongoDB.

4.3.9. Análisis estático de código fuente

La Tabla 26 muestra los resultados del análisis estático de código fuente, cuyo detalle se recoge en el “Anexo F. Análisis estático de código fuente”.

Tabla 26. Resultados análisis estático de código fuente.

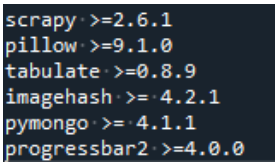
Comprobación	Criterio	Resultado
Cobertura de pruebas automatizadas	≥ 90%	98,1%
Líneas duplicadas	≤ 3 %	0,0%
Bugs detectados y no resueltos	0	0
Vulnerabilidades de seguridad detectadas y no resueltas	0	0

Fuente: Elaboración propia.

4.3.10. Comprobación de la ausencia de vulnerabilidades en las librerías importadas

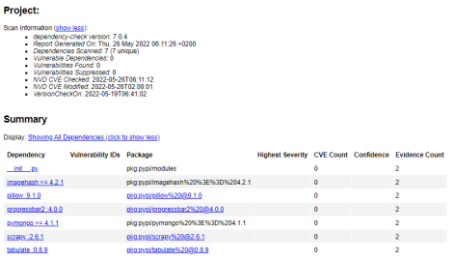
El conjunto de librerías externas se recoge en un fichero *requirements.txt* (mostrado en la Figura 47) que facilita su instalación por medio del comando *pip install -r requirements.txt*. Como resultado del análisis realizado con la aplicación Dependency-check no se ha detectado ninguna vulnerabilidad en las librerías importadas, según se recoge en la Figura 48.

Figura 47. Contenido del fichero requirements.txt.



Fuente: Elaboración propia.

Figura 48. Resultados comprobación de ausencia de vulnerabilidades en librerías importadas.



Fuente: Elaboración propia.

La Tabla 27 muestra el conjunto y versión de las librerías externas empleadas.

Tabla 27. Librerías externas empleadas.

Librería	Versión
scrapy	2.6.1
pillow	9.1.0
tabulate	0.8.9
imagehash	4.2.1
pymongo	4.1.1
progressbar2	4.0.0

Fuente: Elaboración propia.

4.4. Pruebas

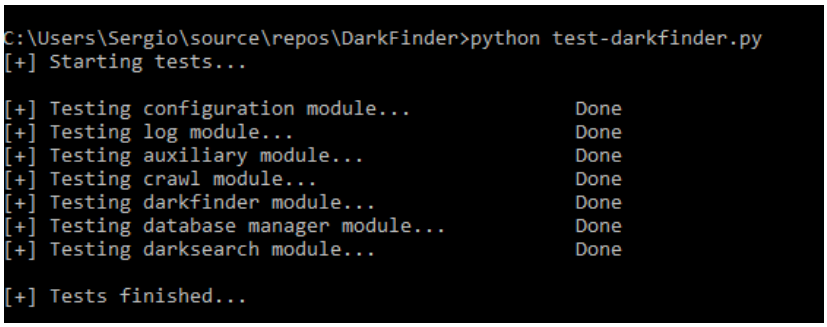
En paralelo al proceso de codificación se han realizado pruebas tanto manuales como automáticas, siguiendo el plan de pruebas definido en el “Anexo E. Plan de pruebas”.

El plan de pruebas contiene un **completo conjunto de verificaciones**, incluyendo:

- Test manuales.
- Test automatizados.
- Test con valores esperados.
- Test con valores erróneos / maliciosos.

Para la creación de las pruebas automatizadas se hace uso de la función *assert,* que muestra un mensaje de error en caso de que el valor devuelto por el método/función testado no coincida con el valor esperado. Los resultados de cobertura de pruebas automatizadas se recogen en la Tabla 26 mostrada anteriormente. De este modo, se han testado todos los requisitos que han sido implantados, según la especificación dada en el "Anexo A. Requisitos de la aplicación".

Figura 49. Resultados pruebas automatizadas.



Fuente: Elaboración propia.

4.5. Producción

En esta fase se contempla la realización del bastionado de sistemas, en base a las guías indicadas a continuación:

- Sistema operativo Windows Server según guía CCN-STIC-573 Implementación de Seguridad en Servidor de Ficheros Sobre Microsoft Windows Server 2016 / CIS Benchmark Microsoft Windows Server.
- Sistema operativo Windows 10 según guía CCN-STIC-599 Configuración segura de Windows 10 / CIS Benchmark Microsoft Intune for Windows 10.
- Sistema operativo Linux según guía CIS Benchmark Ubuntu Linux.
- Sistema virtualización Hyper-V según guía CCN-STIC-578 Implementación de Seguridad en Microsoft Hyper-V sobre Windows Server 2016.
- Base de datos MongoDB según guía CIS Benchmark MongoDB.

Asimismo, en esta fase y dentro del presente TFM, se realiza la configuración del conjunto de herramientas ELK (Elasticsearch, Logstash y Kibana). Se utiliza Elasticsearch para recopilar y procesar la información captada en la base de datos MongoDB, usando Kibana como medio de representación gráfica. A futuro se plantea el uso de Logstash para poder cargar automáticamente la información de las páginas web rastreadas de manera dinámica. El conjunto ELK proporciona una potente herramienta de búsqueda que podría utilizarse para realizar análisis de la información obtenida.

La instalación del conjunto de herramientas necesarias para la puesta en marcha de la aplicación DarkFinder se describe en el "Anexo G. Manual de instalación de la aplicación".

4.6. Resultados

Como resultado del presente TFM se ha obtenido un ***crawler* completamente funcional**, con capacidad de **rastrear páginas web en las redes visible y Dark Web (TOR, I2P, Freenet)**. En el "Anexo H. Resultados" se completan las **evidencias de funcionamiento de la herramienta** proporcionadas en este capítulo, comprobando entre otros aspectos:

- Mensajes de **ayuda y versión** de la aplicación.
- Funcionamiento en **español e inglés**. Los mensajes son mostrados en los dos idiomas, con fechas en formato MM/DD/YYYY en inglés y DD/MM/YYYY en español.
- **Capacidad de rastreo en red visible, TOR, I2P y Freenet**. Capacidad de **saltar de un tipo de red a otra** (p.ej. pasar de red visible a red TOR) en un mismo rastreo.
- Almacenamiento de información en **base de datos no relacional MongoDB**.
- Implementación de **diferentes algoritmos de rastreo**, verificando la aparición de determinadas palabras clave.
- Capacidad de búsqueda de páginas web por:
    - Patrón de búsqueda de **URL** (expresión regular).
    - Patrón de contenido en el **cuerpo HTML** (expresión regular).
    - **Hash MD5/SHA1** de imagen contenida en la página web.
    - Distancia de Hamming a un ***difference hash* de referencia de una imagen** pasado como parámetro.
    - Patrón de búsqueda en los **metadatos de una imagen** (expresión regular).
    - Búsqueda **multicriterio**.
- Funcionamiento en sistema operativo Kali Linux.
- Carga de datos de rastreo en aplicación Elasticsearch y visualización con Kibana.

5. CONCLUSIONES Y TRABAJO FUTURO

Finalmente se presentan las conclusiones del presente TFM y se proponen líneas de desarrollo de trabajo futuro.

5.1. Conclusiones

Como consecuencia de todo el desarrollo realizado, se concluye que **se ha cumplido el objetivo principal del TFM**, habiendo desarrollado un *crawler* capaz de realizar una **búsqueda automatizada y efectiva de patrones delictivos en Internet, tanto en la web visible como en la Dark Web**. Asimismo, aglutina importantes funcionalidades, dando asimismo **cumplimiento a todos los objetivos secundarios** planteados.

Se trata de un **crawler multiplaforma**, ejecutable en sistemas Windows y Linux y desarrollado en lenguaje **Python,** cuya rápida curva de aprendizaje facilita su comprensión por otros desarrolladores, de modo que pueda ser fácilmente ampliado en un futuro. Se trata de una herramienta con **carácter multi idioma** que puede ser empleada tanto en inglés como en español, pudiendo **ser en el futuro traducida a nuevos idiomas sin necesidad de modificar el código fuente**, sino únicamente los ficheros con los literales de traducción. Esto proporciona a la herramienta un **carácter internacional**, pudiendo ser empleada por autoridades de diversos países para la búsqueda de patrones delictivos.

Según se muestra en el "Anexo H. Resultados", la herramienta es **multi red, capaz de rastrear indistintamente tanto la web visible como las Dark Nets TOR, I2P y Freenet**. Se ha diseñado con total flexibilidad, permitiendo seguir enlaces entre distintas redes, pasando por ejemplo de la red visible a la red TOR en un mismo rastreo.

La aplicación se ha diseñado de manera **modular**, compuesta por un total de siete (7) módulos. De esta manera se pueden realizar modificaciones en alguno de ellos sin que el funcionamiento afecte al resto de la aplicación. Uno de los módulos es el encargado de leer el fichero de **configuración**, permitiendo total flexibilidad para que el funcionamiento de la herramienta pueda ser parametrizado por el usuario en base a un fichero de texto.

La herramienta almacena **información relevante de la propia página web**, como la **URL, red a la que pertenece, fecha/hora de rastreo y el texto HTML** que contiene. Los trabajos desarrollados hasta la fecha se centran principalmente en el análisis de texto, existiendo poco desarrollo de rastreadores que obtengan **información de las imágenes** contenidas en la página web. Por ello, el *crawler* es capaz de obtener información relativa al conjunto de imágenes de las webs rastreadas. Se centra **únicamente en aquellas cuyo tamaño supere un umbral mínimo para una mayor eficiencia,** evitando procesar innecesariamente logos, banners, etc. Se permite de este modo realizar búsquedas de imágenes por **hashes**

MD5/SHA1 o los metadatos que contienen. Teniendo en cuenta que pequeñas variaciones en una imagen producen hashes totalmente diferentes, de manera novedosa se obtiene el ***difference hash*** **de las imágenes, permitiendo la búsqueda de imágenes con semejanzas relevantes** con respecto a un patrón calculando la distancia de Hamming.

Teniendo en cuenta el carácter delictivo que pueden tener las imágenes de la Dark Web y las posibles consecuencias penales por su almacenamiento, la aplicación **no almacena en ningún caso las imágenes rastreadas**.

El crawler tiene implantados **diversos algoritmos de rastreo** que pueden ser seleccionados en el fichero de configuración (*Breadth-First Search*, *Depth-First Search*, *Best First Search*). Esto permite realizar comparativas y análisis sobre aquellos algoritmos más adecuados para buscar patrones delictivos en la Dark Web.

Toda la información se almacena en una **base de datos no relacional MongoDB**, por su eficiencia en la realización de consultas de entidades relacionadas. Esta característica permite un **funcionamiento paralelo y distribuido de diferentes ejecuciones de la herramienta** en diversas máquinas cliente, permitiendo reducir los tiempos necesarios para rastrear una gran cantidad de webs.

Para realizar la **búsqueda de patrones delictivos** se proporciona un importante conjunto de opciones. Es posible encontrar páginas web de manera conjunta tanto de la web visible como de las Dark Nets rastreadas por medio de **expresiones regulares** asociadas a la URL, texto HTML, metadatos de las imágenes, etc. Esto proporciona una **gran flexibilidad para buscar determinado tipo de información** (*nicknames*, teléfonos, etc.). Asimismo, proporciona información de aquellas webs que contienen imágenes cuyo hash MD5/SHA1 coincide con un valor proporcionado como parámetro. Con el desarrollo realizado, no sólo se permite la búsqueda de imágenes exactas, sino que mediante la técnica de *difference hashing* se permiten realizar búsquedas de imágenes 'parecidas' que presentan una determinada semejanza a una imagen referencia tomada como patrón.

Por último, en todo el proceso se han seguido los principios y buenas prácticas de desarrollo seguro, contemplando una metodología de **ciclo de vida de desarrollo software seguro (S-SDLC)** para reducir el riesgo de que el *crawler* sufra vulnerabilidades que puedan afectar a su operación.

5.2. Líneas de trabajo futuro

Son diversas las posibles líneas de trabajo futuro que se plantean en el trabajo, gran parte de las cuales se han recogido como requisitos a futuro en el "Anexo A. Requisitos de la aplicación".

- **Desarrollar diversos algoritmos de rastreo** para determinar aquellos que sean más efectivos para buscar patrones e información delictiva en la Dark Web. En este sentido, pueden resultar de interés las líneas de investigación basadas en *machine learning* introducidas en el capítulo "2. Contexto y estado del arte".
- Incorporar **técnicas que dificulten el bloqueo del crawler por servidores de la Dark Web**, como realizar una rotación de la dirección IP al realizar la conexión a la red TOR.
- Permitir el **rastreo de formularios de la Deep Web** mediante el relleno automático de campos de texto en formularios, así como completar de manera automática los campos de usuario-contraseña o patrones CAPTCHA para poder acceder a zonas restringidas.
- Incorporar **nuevas redes de la Dark Web** para el rastreo de patrones delictivos.
- **Integrar automáticamente** mediante Logstash (o mecanismo equivalente) los nuevos **registros almacenados** en la base de datos MongoDB **en la herramienta Elasticsearch**,
- Incorporar **nuevos sistemas gestores de bases de datos (p.ej. MySQL)** para almacenar información.
- Permitir **búsquedas relativas a las coordenadas geográficas** reflejadas en los metadatos de las imágenes obtenidas.
- **Mejorar el algoritmo de búsqueda de imágenes usando como criterio el *difference hash***. Para ello se propone utilizar estructuras de datos que permitan determinar de manera eficaz qué páginas web contienen imágenes cuyo *difference hash* esté a una distancia de Hamming no superior a un determinado valor, como por ejemplo *KD-Trees*, *VP-Trees* o *Ball trees*. A modo de referencia puede considerarse la implementación dada en (Scikit-learn developers, s. f.).

Referencias bibliográficas

Abdulkareem, S. A., & Abboud, A. J. (2021). Evaluating Python, C++, JavaScript and Java Programming Languages Based on Software Complexity Calculator (Halstead Metrics). *IOP Conference Series: Materials Science and Engineering*, *1076*(1), 012046. https://doi.org/10.1088/1757-899X/1076/1/012046

Alayda, S., Almowaysher, N. A., Alserhani, F., & Humayun, M. (2021). Terrorism on Dark Web. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(10), 3000-3005. https://doi.org/10.17762/turcomat.v12i10.4950

Alharbi, A., Faizan, M., Alosaimi, W., Alyami, H., Agrawal, A., Kumar, R., & Khan, R. A. (2021). Exploring the Topological Properties of the Tor Dark Web. *IEEE Access*, *9*, 21746-21758. https://doi.org/10.1109/ACCESS.2021.3055532

Alkhatib, B., & Basheer, R. (2019a). Crawling the Dark Web: A Conceptual Perspective, Challenges and Implementation. *Journal of Digital Information Management*, *17*(2), 51. https://doi.org/10.6025/jdim/2019/17/2/51-60

Alkhatib, B., & Basheer, R. (2019b). Mining the Dark Web: A Novel Approach for Placing a Dark Website under Investigation. *International Journal of Modern Education and Computer Science*, *11*(10), 1-13. https://doi.org/10.5815/ijmecs.2019.10.01

Al-Nabki, M. W., Fidalgo, E., Alegre, E., & Fernández-Robles, L. (2019). ToRank: Identifying the most influential suspicious domains in the Tor network. *Expert Systems with Applications*, *123*, 212-226. https://doi.org/10.1016/j.eswa.2019.01.029

Ball, M., Broadhurst, R., Niven, A., & Trivedi, H. (2021). Data Capture and Analysis of Darknet Markets. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3344936

Baravalle, A., Lopez, M. S., & Lee, S. W. (2016). Mining the Dark Web: Drugs and Fake Ids. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 350-356. https://doi.org/10.1109/ICDMW.2016.0056

Basheer, R., & Alkhatib, B. (2021). Threats from the Dark: A Review over Dark Web Investigation Research for Cyber Threat Intelligence. *Journal of Computer Networks and Communications*, *2021*. https://doi.org/10.1155/2021/1302999

Batchelder, N. (s. f.). *Coverage.py 6.4.1 documentation*. Coverage.py. Recuperado 14 de junio de 2022, de https://coverage.readthedocs.io/en/6.4.1/

Beckett, A. (2009, noviembre 26). The dark side of the internet. *The Guardian*. https://www.theguardian.com/technology/2009/nov/26/dark-side-internet-freenet

Bergman, M. K. (2001). White Paper: The Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, *7*(1), 147. https://doi.org/10.3998/3336451.0007.104

Biryukov, A., Pustogarov, I., Thill, F., & Weinmann, R. P. (2014). Content and Popularity Analysis of Tor Hidden Services. *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW)*, 188-193. https://doi.org/10.1109/ICDCSW.2014.20

Bissias, G., Levine, B., Liberatore, M., Lynn, B., Moore, J., Wallach, H., & Wolak, J. (2016). Characterization of contact offenders and child exploitation material trafficking on five peer-to-peer networks. *Child Abuse & Neglect*, *52*, 185-199. https://doi.org/10.1016/j.chiabu.2015.10.022

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, *30*(1-7), 107-117. https://doi.org/10.1016/S0169-7552(98)00110-X

Brown, R., & Bricknell, S. (2018). What is the profile of child exploitation material offenders? *Trends & issues in crime and criminal justice*, *564*, 1-14. https://www.aic.gov.au/publications/tandi/tandi564

Camargo Sarmiento, F. I., & Ordóñez Salinas, S. (2013). Evolución y tendencias actuales de los Web crawlers. *Ingeniería*, *18*(2). https://doi.org/10.14483/udistrital.jour.reving.2013.2.a02

Campbell, G. A. (2021). *Cognitive Complexity-A new way of measuring understandability*. Sonarsource. https://www.sonarsource.com/docs/CognitiveComplexity.pdf

Catakoglu, O., Balduzzi, M., & Balzarotti, D. (2017). Attacks landscape in the dark side of the web. *Proceedings of the Symposium on Applied Computing*, 1739-1746. https://doi.org/10.1145/3019612.3019796

Centro Criptológico Nacional. (2020, mayo). *Guía de Seguridad de las TIC CCN-STIC 803. ENS. Valoración de los sistemas*. https://www.ccn-cert.cni.es/series-ccn-stic/800-guia-esquema-nacional-de-seguridad/682-ccn-stic-803-valoracion-de-sistemas-en-el-ens-1/file.html

Chaitra, P. G., Deepthi, V., Vidyashree, K. P., & Rajini, S. (2019). A Study on Different Types of Web Crawlers. *Intelligent Communication, Control and Devices*, 781-789. https://doi.org/10.1007/978-981-13-8618-3_80

Cho, J., Garcia-Molina, H., & Page, L. (1998). Efficient crawling through URL ordering. *Computer Networks and ISDN Systems*, *30*(1-7), 161-172. https://doi.org/10.1016/S0169-7552(98)00108-1

Christin, N. (2012). Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. *Proceedings of the 22nd international conference on World Wide Web*, 213-224. https://doi.org/10.1145/2488388.2488408

Colmenares Malaver, G. D., Méndez González, N., & Virgüez Castro, O. D. (2019). *Deep Dark Web & Social Crawler (DDW&SC): Aplicativo para apoyar la gestión de Ciberinteligencia* [Pontificia Universidad Javeriana]. https://repository.javeriana.edu.co/handle/10554/47278

Dalvi, A., Paranjpe, S., Amale, R., Kurumkar, S., Kazi, F., & Bhirud, S. G. (2021). SpyDark: Surface and Dark Web Crawler. *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, 45-49. https://doi.org/10.1109/ICSCCC51823.2021.9478098

Décary-Hétu, D., & Aldridge, J. (2015). Sifting through the Net: Monitoring of Online Offenders by Researchers. *The European Review of Organised Crime*, *2*(2), 122-141. https://standinggroups.ecpr.eu/sgoc/wp-content/uploads/sites/51/2020/01/decaryhetualdridge.pdf

Europol. (2021a). *Drugs and the darknet: Perspectives for enforcement, research and policy*. https://www.europol.europa.eu/publications-events/publications/drugs-and-darknet-perspectives-for-enforcement-research-and-policy

Europol. (2021b). *Internet Organised Crime Threat Assessment (IOCTA) 2021*. https://www.europol.europa.eu/publications-events/main-reports/internet-organised-crime-threat-assessment-iocta-2021

Fazal, N., Nguyen, K. Q., & Fränti, P. (2019). Efficiency of Web Crawling for Geotagged Image Retrieval. *Webology*, *16*(1), 16-39. https://doi.org/10.14704/WEB/V16I1/a177

Fidalgo, E., Alegre, E., González-Castro, V., & Fernández-Robles, L. (2017). Illegal Activity Categorisation in DarkNet Based on Image Classification Using CREIC Method. *International Joint Conference SOCO'17-CISIS'17-ICEUTE'17 León, Spain*. https://doi.org/10.1007/978-3-319-67180-2_58

Fitas, R., Rocha, B., Costa, V., & Sousa, A. (2021). Design and Comparison of Image Hashing Methods: A Case Study on Cork Stopper Unique Identification. *Journal of Imaging*, *7*(3), 48. https://doi.org/10.3390/jimaging7030048

Foo-Manroot. (2018, febrero 4). *Foo-Manroot/ToR_crawler: Little crawler using Tor*. GitHub. https://github.com/Foo-Manroot/ToR_crawler

Fourment, M., & Gillings, M. R. (2008). A comparison of common programming languages used in bioinformatics. *BMC Bioinformatics*, *9*(1), 82. https://doi.org/10.1186/1471-2105-9-82

Franco, J. A. R. (2021). Desmitificando a la Deep Web a través de un fugaz viaje por la Dark Web. *Revista Ingeniería, Matemáticas y Ciencias de la Información*, *15*(8), 13-32. https://doi.org/10.21017/rimci.2021.v8.n15.a89

Free Software Foundation, Inc. (s. f.). *GNU gettext utilities*. GNU. Recuperado 17 de mayo de 2022, de https://www.gnu.org/software/gettext/manual/gettext.html

Fu, T., Abbasi, A., & Chen, H. (2010). A Focused Crawler for Dark Web Forums. *Journal of the American Society for Information Science and Technology*, *61*(6), 1213-1231. https://doi.org/10.1002/asi.21323

Gali, N., Tabarcea, A., & Fränti, P. (2015). Extracting Representative Image from Web Page. *Proceedings of the 11th International Conference on Web Information Systems and Technologies (WEBIST-2015)*, 411-419. https://doi.org/10.5220/0005438704110419

Goertzel, K., & Winograd, T. (2008). *Enhancing the Development Life Cycle to Produce Secure Software*. https://www.researchgate.net/publication/233575692_Enhancing_the_Development_Life_Cycle_to_Produce_Secure_Software

GOVERTIS Advisory Services, S.L. (2015, septiembre 4). *SANDAS G.R.C. - GOVERTIS*. https://www.govertis.com/sandas-grc

Gulyás, A. (2020). DarkWeb. *Strategic Impact*, *77*(4), 152-177. https://www.researchgate.net/profile/Attila-Gulyas-8/publication/354845286_DarkWeb/links/614f98f6f8c9c51a8af336e4/DarkWeb.pdf

Gupta, S., & Bhatia, K. K. (2014). *A Comparative Study of Hidden Web Crawlers*. http://arxiv.org/abs/1407.5732

Hawkins, B. (2016). Under The Ocean of the Internet—The Deep Web. *SANS Institute*, 1-19. https://www.sans.org/white-papers/37012/

Hernández, I., Rivero, C. R., & Ruiz, D. (2019). Deep Web crawling: A survey. *World Wide Web*, *22*(4), 1577-1610. http://dx.doi.org/10.1007/s11280-018-0602-1

Howard, M., & LeBlanc, D. (2003). *Writing secure code* (2nd ed). Microsoft Press.

Huang, K., Siegel, M., Pearlson, K., & Madnick, S. (2019). Casting the Dark Web in a New Light. *MIT Sloan Management Review*, *60*(4), 1-9. https://cams.mit.edu/wp-content/uploads/2019-07-15-SMR-FA19-From-SMR-web-site.pdf

I2P: A scalable framework for anonymous communication—I2P. (s. f.). Recuperado 8 de marzo de 2022, de https://geti2p.net/en/docs/how/tech-intro

Ilić, M., & Spalević, Ž. (2017). The Use of Dark Web for the Purpose of Illegal Activity Spreading. *Ekonomika, Journal for Economic Theory and Practice and Social Issues, 63*(1), 73-82. https://www.ceeol.com/search/article-detail?id=528854

Iliou, C., Kalpakis, G., Tsikrika, T., Vrochidis, S., & Kompatsiaris, I. (2017). Hybrid focused crawling on the Surface and the Dark Web. *EURASIP Journal on Information Security*, *2017*(1), 1-13. https://doi.org/10.1186/s13635-017-0064-5

Jardine, E., Lindner, A. M., & Owenson, G. (2020). The potential harms of the Tor anonymity network cluster disproportionately in free countries. *Proceedings of the National Academy of Sciences*, *117*(50), 31716-31721. https://doi.org/10.1073/pnas.2011893117

Kalpakis, G., Tsikrika, T., Iliou, C., Mironidis, T., Vrochidis, S., Middleton, J., Williamson, U., & Kompatsiaris, I. (2016). Interactive Discovery and Retrieval of Web Resources Containing Home Made Explosive Recipes. *International Conference on Human Aspects of Information Security, Privacy, and Trust*, 221-233. https://doi.org/10.1007/978-3-319-39381-0_20

Kaur, S., & Randhawa, S. (2020). Dark Web: A Web of Crimes. *Wireless Personal Communications*, *112*(4), 2131-2158. https://doi.org/10.1007/s11277-020-07143-2

Kaur, S., Singh, A., Geetha, G., Masud, M., & Alzain, M. A. (2021). SmartCrawler: A Three-Stage Ranking Based Web Crawler for Harvesting Hidden Web Sources. *Computers, Materials and Continua*, *69*(3), 2933-2948. https://doi.org/10.32604/cmc.2021.019030

Kawaguchi Y., & Ozawa S. (2019). Exploring Malicious URL in Dark Web Using Tor Crawler. *IEICE Technical Report; IEICE Tech. Rep.*, *118*(478), 7-12. https://www.ieice.org/ken/paper/20190307l1k6/eng/

Kobayashi, M., & Takeda, K. (2000). Information retrieval on the web. *ACM Computing Surveys*, *32*(2), 144-173. https://doi.org/10.1145/358923.358934

Kumar, M., Bindal, A., Gautam, R., & Bhatia, R. (2018). Keyword query based focused Web crawler. *Procedia Computer Science*, *125*, 584-590. https://doi.org/10.1016/j.procs.2017.12.075

Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal. *Boletín Oficial del Estado, núm 281, de 24 de noviembre de 1995, 33987-34058*. https://www.boe.es/buscar/act.php?id=BOE-A-1995-25444

Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., & Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, *7*(2), 72-93. https://doi.org/10.1109/COMST.2005.1610546

Madhavan, J., Ko, D., Kot, Ł., Ganapathy, V., Rasmussen, A., & Halevy, A. (2008). Google's Deep Web crawl. *Proceedings of the VLDB Endowment*, *1*(2), 1241-1252. https://doi.org/10.14778/1454159.1454163

Magán-Carrión, R., Abellán-Galera, A., Maciá-Fernández, G., & Figueras, E. (2021, diciembre 28). *Nesg-ugr/c4darknet*. GitHub. https://github.com/nesg-ugr/c4darknet

Manikandan, N. K., & Kavitha, M. (2021). Focused Web Crawler For Retrieving Relevant Contents. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(10), 2025-2028. https://doi.org/10.17762/turcomat.v12i10.4708

Mathur, A., Bruno, R., Man, N., Barratt, M. J., Roxburgh, A., Van Buskirk, J., & Peacock, A. (2020). *Methods for the analysis of trends in the availability and type of drugs sold on the internet via cryptomarkets.* National Drug and Alcohol Research Centre. https://ndarc.med.unsw.edu.au/resource/methods-trends-cryptomarket-drug-listings

McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley Professional.

Micard, A., & Ashurst, M. (2019, diciembre 19). *trandoshan-io/crawler: Go process used to crawl websites*. GitHub. https://github.com/trandoshan-io/crawler

Microsoft. (2022, marzo 1). *Threats—Microsoft Threat Modeling Tool—Azure | Microsoft Docs*. https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats

Miller, A. (2019, diciembre 16). *alex-miller-0/Tor_Crawler: Web crawling with IP rotation via Tor*. GitHub. https://github.com/alex-miller-0/Tor_Crawler

Ministerio de Hacienda y Administraciones Públicas. (2012a, octubre). *MAGERIT - versión 3.0 | Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información | Libro I - Método*. https://www.ccn-cert.cni.es/documentos-publicos/1789-magerit-libro-i-metodo/file.html

Ministerio de Hacienda y Administraciones Públicas. (2012b, octubre). *MAGERIT - versión 3.0 | Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información | Libro II - Catálogo de Elementos*. https://www.ccn-cert.cni.es/documentos-publicos/1791-magerit-libro-ii-catalogo/file.html

Mirea, M., Wang, V., & Jung, J. (2019). The not so dark side of the darknet: A qualitative study. *Security Journal*, *32*(2), 102-118. https://doi.org/10.1057/s41284-018-0150-5

MongoDB, Inc. (s. f.-a). *Advantages Of MongoDB*. Recuperado 19 de abril de 2022, de https://www.mongodb.com/advantages-of-mongodb

MongoDB, Inc. (s. f.-b). *BSON Types—MongoDB Manual*. Recuperado 20 de abril de 2022, de https://www.mongodb.com/docs/manual/reference/bson-types/

MongoDB, Inc. (s. f.-c). *PyMongo 4.1.1 Documentation*. Recuperado 18 de mayo de 2022, de https://pymongo.readthedocs.io/en/stable/

Mundluru, D., & Xia, X. (2008). Experiences in crawling deep web in the context of local search. *Proceedings of the 5th Workshop on Geographic Information Retrieval*, 35-42. https://doi.org/10.1145/1460007.1460016

Narayanan, P. S., Ani, R., & King, A. T. L. (2020). TorBot: Open Source Intelligence Tool for Dark Web. *Inventive Communication and Computational Technologies*, *89*, 187-195. https://doi.org/10.1007/978-981-15-0146-3_19

Narayanan, P. S., Ani, R., & King, A. T. L. (2022, abril 15). *DedSecInside/TorBot: Dark Web OSINT Tool*. GitHub. https://github.com/DedSecInside/TorBot

Nazah, S., Huda, S., Abawajy, J., & Hassan, M. M. (2020). Evolution of Dark Web Threat Analysis and Detection: A Systematic Approach. *IEEE Access*, *8*, 171796-171819. https://doi.org/10.1109/ACCESS.2020.3024198

Object Management Group. (s. f.). *About the Unified Modeling Language Specification Version 2.5.1*. Recuperado 23 de abril de 2022, de https://www.omg.org/spec/UML/

Ochando, M. B. (2014). Nuevos retos de la tecnología web crawler para la recuperación de información. *Métodos de información*, *4*(7), 115-128. https://doi.org/10.5557/IIMEI4-N7-115128

OWASP Foundation, Inc. (s. f.). *OWASP Dependency-Check*. Recuperado 14 de abril de 2022, de https://owasp.org/www-project-dependency-check/

Owenson, G. H., & Savage, N. J. (2015). *The Tor Dark Net*. Global Commission on Internet Governance. https://www.cigionline.org/publications/tor-dark-net

Paganini, P. (2013, julio 1). *Project Artemis – OSINT activities on Deep Web*. Infosec Institute. https://resources.infosecinstitute.com/topic/project-artemis-osint-activities-on-deep-web/

Parker, Z., Poe, S., & Vrbsky, S. V. (2013). Comparing NoSQL MongoDB to an SQL DB. *Proceedings of the 51st ACM Southeast Conference*, 1-6. https://doi.org/10.1145/2498328.2500047

Pavalam, S. M., Raja, S. K., Akorli, F. K., & Jawahar, M. (2011). A Survey of Web Crawler Algorithms. *International Journal of Computer Science Issues (IJCSI)*, *8*(6), 309. https://www.ijcsi.org/papers/IJCSI-8-6-1-309-313.pdf

Prechelt, L. (2000). An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program. *IEEE Computer*, *33*(10), 23-29. https://www.nsl.com/papers/phone/jccpprtTR.pdf

Python Software Foundation. (2021, febrero 22). *Tabulate · PyPI*. https://pypi.org/project/tabulate/

Raghavan, S., & Garcia-Molina, H. (2001). *Crawling the Hidden Web*. 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italia. http://ilpubs.stanford.edu:8090/725/

Rahayuda, I. G. S., & Santiari, N. P. L. (2017). Crawling and cluster hidden web using crawler framework and fuzzy-KNN. *2017 5th International Conference on Cyber and IT Service Management (CITSM)*, 1-7. https://doi.org/10.1109/CITSM.2017.8089225

Raj, S. S., Ahamed, S. A., Rajmohan, R., & Guruprakash, K. S. (2021). Design and Implementation of Distributed Web Crawler for Drug Website Search using Hefty based Enhanced Bandwidth Algorithms. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, *12*(9), 75-81. https://turcomat.org/index.php/turkbilmat/article/view/2837

Rashmi, K. B., Vijaya Kumar, T., & Guruprasad, H. S. (2016). Deep Web Crawler: Exploring and Re-ranking of Web Forms. *International Journal of Computer Applications*, *150*(1), 32-35. https://doi.org/10.5120/ijca2016911448

Razaque, A., Valiyev, B., Alotaibi, B., Alotaibi, M., Amanzholova, S., & Alotaibi, A. (2021). Influence of COVID-19 Epidemic on Dark Web Contents. *Electronics*, *10*(22), 2744. https://doi.org/10.3390/electronics10222744

Santos, A., & Pham, K. (2022, febrero 7). *VIDA-NYU/ache: ACHE is a web crawler for domain-specific search.* GitHub. https://github.com/VIDA-NYU/ache

Scikit-learn developers. (s. f.). *sklearn.neighbors.BallTree—Scikit-learn 1.1.1 documentation*. Recuperado 20 de mayo de 2022, de https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.BallTree.html

Scrivens, R., Gaudette, T., Davies, G., & Frank, R. (2019). Searching for Extremist Content Online Using the Dark Crawler and Sentiment Analysis. *Methods of Criminology and Criminal Justice Research*, *24*, 179-194. https://doi.org/10.1108/S1521-613620190000024016

Shakarian, P. (2018). Dark-Web Cyber Threat Intelligence: From Data to Intelligence to Prediction. *Information*, *9*(12), 305. https://doi.org/10.3390/info9120305

Shelke, S., & Sagar, P. (2017). A Survey on Uniform Resource Locator and Content Matching to Discover Deep- Web Pages. *Indian Journal of Science and Technology*, *10*(17), 1-5. https://doi.org/10.17485/ijst/2017/v10i17/110304

Sherman, C. B., & Price, G. (2001). *The Invisible Web: Uncovering Information Sources Search Engines Can't See* (Vol. 1). Information Today, Inc.

Shestakov, D. (2013). Current Challenges in Web Crawling. *International Conference on Web Engineering*, *7977*, 518-521. https://doi.org/10.1007/978-3-642-39200-9_49

Shinde, V., Dhotre, S., Gavde, V., Dalvi, A., Kazi, F., & Bhirud, S. G. (2020). CrawlBot: A Domain-Specific Pseudonymous Crawler. *International Conference on Cybersecurity in Emerging Digital Era*, 89-101. https://doi.org/10.1007/978-3-030-84842-2_7

Shkapenyuk, V., & Suel, T. (2002). Design and Implementation of a High-Performance Distributed Web Crawler. *Proceedings 18th International Conference on Data Engineering*, 357-368. https://doi.org/10.1109/ICDE.2002.994750

SonarSource S.A. (s. f.). *Code Quality and Code Security | SonarQube*. Recuperado 14 de abril de 2022, de https://www.sonarqube.org/

Soulemane, M., Rafiuzzaman, M., & Mahmud, H. (2012). Crawling the Hidden Web: An Approach to Dynamic Web Indexing. *International Journal of Computer Applications*, *55*(1), 7-15. https://doi.org/10.5120/8717-7290

Spitters, M., Verbruggen, S., & Van Staalduinen, M. (2014). Towards a Comprehensive Insight into the Thematic Organization of the Tor Hidden Services. *2014 IEEE Joint Intelligence and Security Informatics Conference*, 220-223. https://doi.org/10.1109/JISIC.2014.40

Suzanti, I. O., Razi, F., Husni, Rochman, E. M. S., & Fitriani, N. (2021). Focused Web Crawlers on Domain-Specific Retrieval Systems. *IOP Conference Series: Materials Science and Engineering*, *1125*(1), 012045. https://doi.org/10.1088/1757-899X/1125/1/012045

The Freenet Project Inc. (s. f.). *Help*. Recuperado 9 de marzo de 2022, de https://freenetproject.org/pages/help.html

The MITRE Corporation. (2022a, mayo 20). *CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC™)*. https://capec.mitre.org/index.html

The MITRE Corporation. (2022b, mayo 20). *CWE - Common Weakness Enumeration*. https://cwe.mitre.org/

The Open Group. (2018). *The TOGAF® Standard, Version 9.2*. https://publications.opengroup.org/c182

The Tor Project, Inc. (s. f.-a). *ABOUT TOR BROWSER | Tor Project | Tor Browser Manual*. Recuperado 8 de marzo de 2022, de https://tb-manual.torproject.org/about/

The Tor Project, Inc. (s. f.-b). *Tor Project | Download*. Recuperado 7 de marzo de 2022, de https://www.torproject.org/download/

The Tor Project, Inc. (s. f.-c). *Tor Project | History*. Recuperado 7 de marzo de 2022, de https://www.torproject.org/about/history/

The Tor Project, Inc. (s. f.-d). *Tor Project | How do Onion Services work?* Recuperado 3 de abril de 2022, de https://community.torproject.org/onion-services/overview/

The Tor Project, Inc. (s. f.-e). *Welcome to Tor Metrics*. Recuperado 12 de marzo de 2022, de https://metrics.torproject.org

Thelwall, M. (2001). A web crawler design for data mining. *Journal of Information Science*, *27*(5), 319-325. https://doi.org/10.1177/016555150102700503

Tomala, K., Plucar, J., Dubec, P., Rapant, L., & Voznak, M. (2013). The Data Extraction Using Distributed Crawler Inside Multi-Agent System. *Advances in Electrical and Electronic Engineering*, *11*(6), 455-460. https://doi.org/10.15598/aeee.v11i6.867

Uzun, E., Ozhan, E., Agun, H. V., Yerlikaya, T., & Bulus, H. N. (2020). Automatically Discovering Relevant Images From Web Pages. *IEEE Access*, *8*, 208910-208921. https://doi.org/10.1109/ACCESS.2020.3039044

Van Hattem, R. (2022, enero 5). *WoLpH/python-progressbar: Progressbar 2—A progress bar for Python 2 and Python 3—'pip install progressbar2'*. Github. https://github.com/WoLpH/python-progressbar

Van Hout, M. C., & Bingham, T. (2013). 'Silk Road', the virtual drug marketplace: A single case study of user experiences. *International Journal of Drug Policy*, *24*(5), 385-391. https://doi.org/10.1016/j.drugpo.2013.01.005

Van Rossum, G., Warsaw, B., & Coghlan, N. (2001, julio 5). *PEP 8—Style Guide for Python Code*. Python. https://peps.python.org/pep-0008/

Vignoli, R. G., & Monteiro, S. D. (2020). Deep Web e Dark Web: Similaridades e dissiparidades no contexto da Ciência da Informação. *Transinformação*, *32*. https://doi.org/10.1590/2318-0889202032e190052

Vyas, K., & Frasincar, F. (2020). Determining the most representative image on a Web page. *Information Sciences*, *512*, 1234-1248. https://doi.org/10.1016/j.ins.2019.10.045

Wang, D., & Liang, J. (2019). Research and Design of Theme Image Crawler Based on Difference Hash Algorithm. *IOP Conference Series: Materials Science and Engineering*, *563*(4), 042080. https://doi.org/10.1088/1757-899X/563/4/042080

Wang, J., Deng, S., & Wang, L. (2019). Multilingual Focused Crawler System based on Web Content Extraction and Path Configuration. *IOP Conference Series: Materials Science and Engineering*, *569*(5), 052030. https://doi.org/10.1088/1757-899X/569/5/052030

Webfp. (2021a, marzo 10). *webfp/tor-browser-crawler: A crawler based on Tor Browser and Selenium*. GitHub. https://github.com/webfp/tor-browser-crawler

Webfp. (2021b, diciembre 27). *webfp/tor-browser-selenium: Tor Browser automation with Selenium.* GitHub. https://github.com/webfp/tor-browser-selenium

Williams, R., Samtani, S., Patton, M., & Chen, H. (2018). Incremental Hacker Forum Exploit Collection and Classification for Proactive Cyber Threat Intelligence: An Exploratory Study. *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 94-99. https://doi.org/10.1109/ISI.2018.8587336

Winkler, I., & Gomes, A. T. (2016). *Advanced Persistent Security: A Cyberwarfare Approach to Implementing Adaptive Enterprise Protection, Detection, and Reaction Strategies*. Syngress.

Xu, Y., Chen, G., Wu, J., Xu, W., & Liu, Q. (2021). Research on Dark Web Monitoring Crawler Based on TOR. *2021 IEEE 2nd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA)*, *2*, 197-202. https://doi.org/10.1109/ICIBA52610.2021.9687954

Yu, Y.-B., Huang, S.-L., Tashi, N., Zhang, H., Lei, F., & Wu, L.-Y. (2018). A Survey about Algorithms Utilized by Focused Web Crawler. *Journal of Electronic Science and Technology*, *16*(2), 129-138. https://doi.org/10.11989/JEST.1674-862X.70116018

Zhou, Y., Qin, J., Lai, G., Reid, E., & Chen, H. (2005). Building Knowledge Management System for Researching Terrorist Groups on the Web. *AMCIS 2005 Proceedings*, 344. https://aisel.aisnet.org/amcis2005/344/

Zulkarnine, A. T., Frank, R., Monk, B., Mitchell, J., & Davies, G. (2016). Surfacing collaborated networks in dark web to find illicit and criminal content. *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 109-114. https://doi.org/10.1109/ISI.2016.7745452

Anexo A. Requisitos de la aplicación

La Tabla 28 incluye los requisitos que debe cumplir la aplicación. Se indica para cada uno de ellos la fase en la que se contempla su implantación. Los requisitos podrán ser:

- Obligatorios: incluyen la etiqueta 'DEBERÁ' o 'NO DEBERÁ'.
- Opcionales: incluyen la referencia 'PODRÁ'.

Nota: en el presente TFM únicamente se han implementado los requisitos correspondientes a las fases 1 y 2.

Tabla 28. Requisitos de la aplicación.

Id	Fase	Requisito
Generales		
GEN-01	1	La aplicación DEBERÁ permitir la configuración de, como mínimo, los siguientes parámetros mediante un fichero de texto denominado *darkfinder.conf*, ubicado en el mismo directorio que el programa principal: • Idioma utilizado. • Página(s) web semilla para comenzar el rastreo. • Condiciones para finalización del rastreo: número máximo de páginas y profundidad máxima de rastreo. • Estrategia de rastreo: 'bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best First Search). • Palabras clave relacionadas con la estrategia de rastreo. • Tamaño mínimo (ancho, alto) de las imágenes que van a ser analizadas. • Conexión a la base de datos: servidor, ruta de fichero en la que se encuentra usuario, contraseña.
GEN-02	1	La aplicación DEBERÁ permitir soporte multi idioma (español / inglés).
GEN-03	1	La aplicación DEBERÁ estar comentada en inglés y español. Con carácter general, se seguirán las pautas de codificación de código en Python de la guía PEP-8 especificada en (Van Rossum et al., 2001).
GEN-04	1	La aplicación DEBERÁ mostrar por pantalla y en fichero log los siguientes mensajes relativos a la marcha del proceso de rastreo: • Indicación del comienzo del proceso de rastreo. • Indicación del fin del proceso de rastreo. • Duración del proceso de rastreo. • Número de páginas web rastreadas.

Id	Fase	Requisito
GEN-05	2	La aplicación PODRÁ ser multiplataforma, funcionando en los siguientes sistemas operativos: • Microsoft Windows 10. • Linux, distribución Kali Linux.
GEN-06	2	La complejidad cognitiva[8] del conjunto de métodos y funciones de la aplicación NO DEBERÁ ser superior a 15.
GEN-07	3	La aplicación DEBERÁ eliminar automáticamente páginas web rastreadas con una antigüedad superior a un parámetro configurable.
Crawler		
CRA-01	1	La aplicación DEBERÁ recorrer de manera recursiva un conjunto de páginas web a partir de una dirección semilla y las páginas referenciadas como enlace en cada una de las páginas.
CRA-02	1	La aplicación DEBERÁ permitir configurar tanto la dirección semilla de partida como las condiciones para la finalización del rastreo. La aplicación permitirá incluir más de una dirección semilla, pudiendo pertenecer a redes independientes (web visible, TOR, I2P, Freenet).
CRA-03	1	La aplicación DEBERÁ analizar contenido de las siguientes redes: • Web visible. • TOR. • I2P. • Freenet.
CRA-04	1	La aplicación DEBERÁ permitir alternar enlaces entre las diferentes redes, permitiendo seguir un enlace de una página en una de las redes a otra página de una red diferente.
CRA-05	1	La aplicación DEBERÁ implantar las siguientes estrategias de *crawling*: • *Breadth-First Search* (BFS). • *Depth-First Search* (DFS).
CRA-06	1	La aplicación DEBERÁ evitar el rastreo de páginas que hayan sido rastreadas con anterioridad, estando almacenadas en la base de datos.
CRA-07	2	La aplicación PODRÁ realizar un cálculo de la frecuencia de aparición de un conjunto de palabras clave configurable.
CRA-08	2	La aplicación PODRÁ implantar una estrategia de *crawling* de tipo *Best-First Search* (BFS), proporcionando un ranking a las páginas localizadas en base a un algoritmo definido que tenga en consideración un conjunto de palabras clave predefinidas por el usuario.

[8] Medida de la complejidad del flujo de control de un método o función, calculada según la metodología indicada en (Campbell, 2021).

Id	Fase	Requisito
CRA-09	3	La aplicación PODRÁ eludir medidas de bloqueo de IP realizadas por servidores en la red TOR por medio de la rotación de la dirección IP en esta red.
CRA-10	3	La aplicación PODRÁ permitir el rastreo de formularios de la Deep Web mediante el relleno automático de campos de texto de formularios.
CRA-11	3	La aplicación PODRÁ permitir acceder a zonas de acceso restringido mediante usuario-contraseña en base a un listado de usuarios y contraseñas incluidos en ficheros de texto.
CRA-12	3	La aplicación PODRÁ permitir resolver de manera automática patrones CAPTCHA de texto.
CRA-13	3	La aplicación PODRÁ incorporar nuevas Dark Nets para el rastreo de patrones delictivos.
Imágenes		
IMA-01	1	La aplicación DEBERÁ obtener el hash MD5 y SHA1 relativos a las imágenes existentes en todas las páginas visitadas.
IMA-02	1	La aplicación DEBERÁ obtener metadatos EXIF relativos a las imágenes existentes en todas las páginas visitadas.
IMA-03	1	La aplicación DEBERÁ excluir del análisis aquellas imágenes cuyo tamaño (ancho, alto) sea inferior a un valor configurable.
IMA-04	1	La aplicación NO DEBERÁ almacenar en el disco duro las imágenes analizadas.
IMA-05	2	La aplicación PODRÁ obtener el *difference hash* relativo a las imágenes existentes en todas las páginas visitadas.
Bases de datos		
BD-01	1	La aplicación DEBERÁ almacenar la información en una base de datos no estructurada MongoDB.
BD-02	1	La aplicación DEBERÁ almacenar como mínimo la siguiente información relativa a una página web: • URL. • Red (visible, TOR, I2P, Freenet). • Fecha/Hora de rastreo. • Contenido HTML de la página web. • Información relativa a imágenes existentes en la página web (ver BD-03).
BD-03	1	La aplicación DEBERÁ almacenar como mínimo la siguiente información relativa a una imagen: • Nombre del fichero. • Hashes MD5 y SHA-1. • Metadatos EXIF.

Id	Fase	Requisito
BD-04	1	La aplicación DEBERÁ realizar la conexión a base de datos MongoDB mediante API específica estándar de conexión para minimizar los fallos.
BD-05	1	La aplicación DEBERÁ almacenar la información correspondiente a una página web y las imágenes asociadas de manera atómica.
BD-06	2	La aplicación PODRÁ almacenar la siguiente información relativa a una imagen: *Difference Hash*.
BD-07	2	La aplicación PODRÁ almacenar la información en un sistema Elasticsearch que permita realizar búsquedas avanzadas por el usuario.
BD-08	3	La aplicación PODRÁ cargar de manera automática la información rastreada en un sistema Elasticsearch mediante Logstash o mecanismo equivalente.
BD-09	3	La aplicación PODRÁ incorporar nuevos sistemas gestores de bases de datos para almacenar información: • MySQL.
Búsqueda		
BUS-01	1	La aplicación DEBERÁ incorporar un módulo de búsqueda de patrones delictivos en el contenido de la página web, incluyendo como mínimo las siguientes posibilidades: • Campo de texto. • Expresión regular (Regex).
BUS-02	1	La aplicación DEBERÁ incorporar un módulo de búsqueda de patrones delictivos en el contenido de una imagen, incluyendo como mínimo las siguientes posibilidades: • Hash MD5 / SHA1. • Expresión regular (Regex) asociada a metadatos.
BUS-03	1	La aplicación DEBERÁ mostrar como mínimo la siguiente información en caso de encontrar coincidencia: • URL. • Red (visible, TOR, I2P, Freenet). • Fecha de rastreo. En el caso de que se realice una búsqueda de imágenes, se mostrarán los siguientes campos: • Nombre del fichero imagen.
BUS-04	2	La aplicación PODRÁ incorporar un módulo de búsqueda de patrones delictivos en el contenido de una imagen, incluyendo: • *Difference Hash*, especificando tanto el hash referencia como la distancia (distancia Hamming). Además de la información mostrada en BUS-03, la aplicación deberá mostrar: • Distancia Hamming entre la imagen y el hash de referencia.

Id	Fase	Requisito
BUS-05	3	La aplicación PODRÁ permitir búsquedas relativas a las coordenadas geográficas reflejadas en los metadatos de las imágenes obtenidas.
Seguridad		
SEG-01	1	La aplicación DEBERÁ controlar los errores mediante bloques *try-except* en los accesos a: • Recepción de argumentos por línea de comandos. • Red. • Disco duro. • Base de datos.
SEG-02	1	La aplicación DEBERÁ almacenar logs con el siguiente contenido: • Fecha/hora. • Nivel del mensaje (*Debug, Info, Warning, Error, Critical*). • Usuario que ejecuta la aplicación. • Mensaje. • Módulo en el que se produce el log. • Función en la que se produce el log. La aplicación DEBERÁ registrar logs en las siguientes situaciones: • Errores. • Eventos informando de actuaciones del usuario y del software.

SEG-03	1	La aplicación DEBERÁ sanitizar el contenido de los parámetros del fichero de configuración.
		Asimismo, deberá realizar las comprobaciones de todos los parámetros de configuración del fichero *darkfinder.conf*. Se incluirán como mínimo:
		<u>Cadenas de caracteres</u>
		• Si es posible, verificación de valores permitidos (lista blanca).
		• En caso contrario, verificación de:
		  ○ Tipo (string).
		  ○ En su caso, longitud mínima y máxima.
		  ○ Filtrado de metacaracteres (“/”, “<”, “>”, etc.).
		<u>Numéricos</u>
		• Verificación de tipo (int / float).
		• En su caso, valor mínimo y máximo.

SEG-04	1	La aplicación DEBERÁ tener un acceso a la base de datos protegido mediante usuario y contraseña.
SEG-05	1	La aplicación DEBERÁ sanitizar la información de las webs rastreadas y que será almacenada en la base de datos MongoDB, incluyendo el contenido HTML de las páginas web almacenadas para evitar ataque de tipo *Cross-Site-Scripting* (XSS).

Id	Fase	Requisito
SEG-06	1	La aplicación DEBERÁ sanitizar el contenido de entradas introducidas por el usuario por línea de comandos, incluyendo aquellas que vayan a ser empleadas en consultas a bases de datos, para evitar inyecciones NoSQL.
SEG-07	1	La aplicación DEBERÁ contar con módulos de realización de pruebas automáticas, con una cobertura de código superior a 90%, incluyendo valores esperados (casos de uso) y no esperados (casos de abuso).
SEG-08	1	La aplicación DEBERÁ tratar los atributos de clases como privadas, proporcionando métodos para su lectura / escritura: • *get*: lectura del atributo. • *set*: escritura del atributo, realizando las comprobaciones de su validez, con las consideraciones establecidas en SEG-03.
SEG-09	1	La aplicación DEBERÁ instalarse y ejecutarse en entorno virtualizado.
SEG-10	1	La aplicación DEBERÁ instalarse en un sistema operativo con gestión de usuarios. Se ejecutará con un usuario SIN privilegios (NO root / administrador).
SEG-11	1	La aplicación DEBERÁ contemplar copia de seguridad de código fuente, fichero de configuración y datos de la base de datos.
SEG-12	1	La aplicación DEBERÁ contar con la configuración segura de: • Sistema operativo Windows Server según guía CCN-STIC-573 Implementación de Seguridad en Servidor de Ficheros Sobre Microsoft Windows Server 2016 / CIS Benchmark Microsoft Windows Server. • Sistema operativo Windows 10 según guía CCN-STIC-599 Configuración segura de Windows 10 / CIS Benchmark Microsoft Intune for Windows 10. • Sistema operativo Linux según guía CIS Benchmark Ubuntu Linux. • Sistema virtualización Hyper-V según guía CCN-STIC-578 Implementación de Seguridad en Microsoft Hyper-V sobre Windows server 2016. • Base de datos MongoDB según guía CIS Benchmark MongoDB.
SEG-13	1	La aplicación DEBERÁ pasar un análisis estático de código fuente.
SEG-14	1	La aplicación DEBERÁ pasar un análisis de librerías importadas vulnerables.
SEG-15	1	La aplicación DEBERÁ tener instalado software antimalware en cliente y servidor.
SEG-16	1	La aplicación NO DEBERÁ codificar contraseñas en el código fuente (*hardcoded password*).
SEG-17	1	La aplicación DEBERÁ realizar la lectura / escritura en el sistema de ficheros mediante librería estándar.

Fuente: Elaboración propia.

Anexo B. Análisis de modelado de amenazas

Se incluye en el presente anexo:

- Modelo de amenazas STRIDE.
- Criterios para la categorización de amenazas siguiendo el método DREAD.
- Listado de amenazas contempladas en el modelado realizado.
- Resultado e informe del modelado de amenazas.

Modelo de amenazas

La Tabla 29 recoge el modelo de amenazas STRIDE (Microsoft, 2022) empleado en el análisis.

Tabla 29. STRIDE.

Categoría	Descripción
Spoofing	Suplantación de la identidad del usuario. Conjunto de técnicas que buscan un compromiso de la gestión de la identidad, autenticación y autenticidad del sistema.
Tampering	Manipulación no autorizada. Motivación de afectar a la integridad de los elementos del sistema, modificándolo de manera maliciosa.
Repudation	Referente a no repudio. Cualidad de un sistema que permite tener cierta validez sobre la demostración de la autoría de una determinada acción.
Information Disclosure	Exposición de información. Brecha o fuga de información confidencial de la organización.
Denial of Service	Denegación de servicio. Ataques que buscan afectar a la capacidad del sistema para ofrecer servicio de manera temporal o indefinida.
Elevation of privilege	Escalada de privilegios. Motivación de realizar acciones para las que un usuario no está autorizado originalmente, elevando el nivel de permisos y disponibilidad sobre los recursos y funcionalidades del sistema.

Fuente: (Microsoft, 2022).

Criterios de categorización de amenazas

La Tabla 30 recoge los criterios de categorización de las amenazas seguido en base al método DREAD.

Tabla 30. Criterios de categorización de amenazas según el método DREAD.

Dimensión	Valor	Criterio
Damage (D) – Valoración del daño causado en caso de materialización de la amenaza	High (3)	• Impacto económico > 10% del presupuesto anual de la organización. • Acceso no autorizado a información sensible.
	Medium (2)	• Impacto económico entre un 4 y 10% del presupuesto anual de la organización. • Acceso no autorizado a información privada que no tiene el carácter de sensible.
	Low (1)	• Impacto económico inferior a un 4% del presupuesto anual de la organización. • No supone acceso a información privada.
Reproducibility (R) – Valoración de la facilidad en reproducir la explotación de la amenaza	High (3)	Sólo requiere ejecución de aplicación maliciosa, sin requerir autenticación.
	Medium (2)	Se requieren uno o dos pasos, pudiendo requerir autenticación.
	Low (1)	La secuencia de pasos es tremendamente complicada o imposible, incluso para el administrador del sistema.
Explotability (E) – Valoración de las herramientas y conocimientos necesarios para explotar la amenaza	High (3)	Explotable mediante malware o aplicación nativa.
	Medium (2)	Malware o *exploits* disponibles en Internet.
	Low (1)	Se requieren conocimientos avanzados de programación y sistemas, con herramientas de ataque avanzados y particularizadas.
Affected Users (A) – Valoración del número de usuarios afectados	High (3)	Más del 75% de la plantilla.
	Medium (2)	Entre el 33% y 66% de la plantilla.
	Low (1)	Menos del 33% de la plantilla.
Discoverability (DI) – Valoración de la facilidad para descubrir la vulnerabilidad	High (3)	Detalles accesibles en Internet o de manera pública que pueden ser fácilmente localizables.
	Medium (2)	Se puede obtener mediante sencillo análisis de peticiones y respuestas a la aplicación.
	Low (1)	Requiere acceso al código fuente o permisos administrativos.

Fuente: Elaboración propia.

El riesgo se calcula en función de los valores obtenidos según la fórmula:

$$riesgo = (R + E + DI) \times (D + A)$$

La Tabla 31 recoge los criterios de categorización de la amenaza en función del valor de riesgo obtenido.

Tabla 31. Criterios de categorización de la amenaza en función del riesgo.

Valor	Criterio
High (Alto)	Riesgo > 48
Medium (Medio)	Riesgo entre 24 y 48
Low (Bajo)	Riesgo < 24

Fuente: Elaboración propia.

Listado de amenazas

La Tabla 32 recoge el conjunto de amenazas contemplado en el análisis de modelado realizado, incluyendo tanto la denominación original en la herramienta como la traducción en español.

Tabla 32. Listado de amenazas contempladas en el modelado de amenazas.

Amenaza	Tipo
Spoofing of Destination Data Store Suplantación de almacén de datos destino	Spoofing
Spoofing of Source Data Store Suplantación de almacén de datos origen	
Spoofing the Entity Suplantación de entidad	
Spoofing the Process Suplantación de proceso	
Potential No SQL Injection Vulnerability Potencial vulnerabilidad de inyección No SQL	Tampering
Potential Lack of Input Validation Potencial falta de validación de entrada	
Cross Site Scripting	
Risks from logging Riesgos derivados de registro de eventos	

Amenaza	Tipo
Data Logs from an Unknown Source Logs de datos de fuente desconocida	Repudation
Insufficient Auditing Auditoría insuficiente	
Lower Trusted Subject Updates Logs Sujeto con bajo nivel de privilegios actualiza logs	
Potential Data Repudiation Repudio de datos potencial	
Potential Weak Protections for Audit Data Protecciones débiles potenciales para datos de auditoría.	
Data Flow Sniffing Captación de flujo de datos	Information Disclosure
Weak Access Control for a Resource Control de acceso débil para un recurso	
Data Flow Is Potentially Interrupted El flujo de datos es potencialmente interrumpido	Denial of Service
Potential Excessive Resource Consumption Potencial consumo de recursos excesivo	
Potential Process Crash or Stop Potencial quiebra de proceso o parada	
Application May be Subject to Elevation of Privilege Using Remote Code Execution La aplicación puede estar sujeta a elevación de privilegios usando ejecución remota de código	Elevation Of Privilege
Elevation by Changing the Execution Flow Elevación por cambio en el flujo de ejecución	
Elevation Using Impersonation Elevación usando suplantación	

Fuente: Elaboración propia a partir de (Microsoft, 2022).

Resultado del modelado de amenazas

Se han analizado un total de 38 amenazas, con la distribución recogida en la Tabla 33:

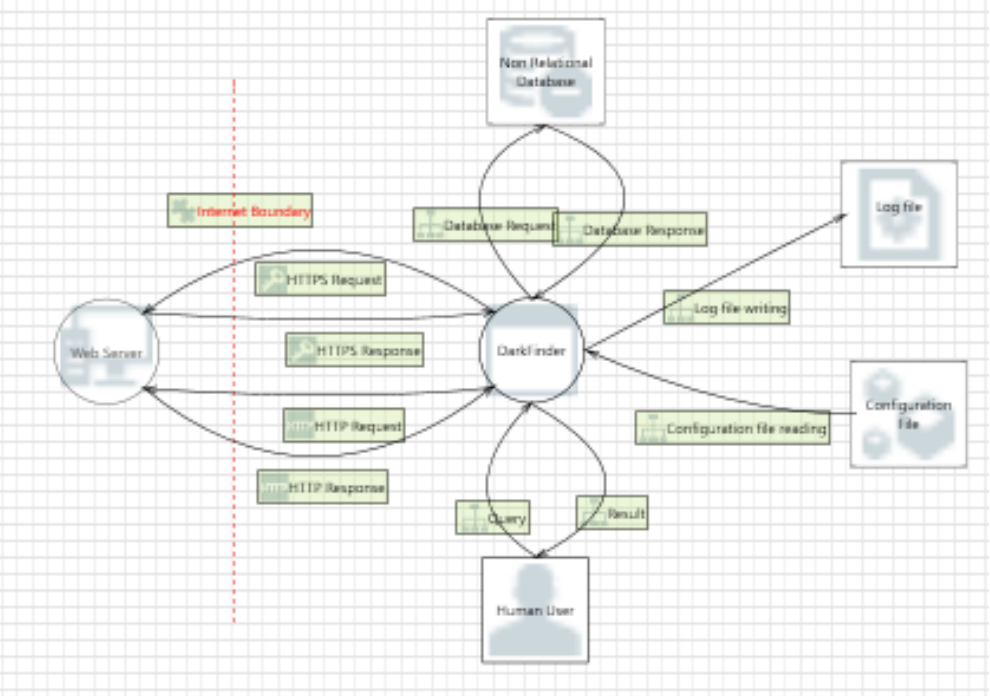
Tabla 33. Distribución de tipos de amenazas en el modelado.

Tipo amenaza	Valor
No aplicables	11
Mitigadas	25
No comenzadas	2[9]
Total	**38**

Fuente: Elaboración propia.

La Figura 50 recoge el diagrama modelado de amenazas realizado mediante la aplicación Microsoft Threat Modelling Tool. Para la mitigación de las amenazas se contemplan un conjunto de medidas de seguridad, recogidas en la Tabla 34. Para cada una de ellas se hace referencia al requisito correspondiente según la especificación realizada en el "Anexo A. Requisitos de la aplicación".

Figura 50. Diagrama de modelado de amenazas.



Fuente: Elaboración propia.

Nota: por simplificar el modelo de amenazas no se incluye el módulo DarkSearch por estar sus amenazas contempladas en el módulo DarkFinder.

[9] Correspondiente a la amenaza "*Data Flow Is Potentially Interrupted*" que tiene un valor de riesgo relativamente bajo (10) y para la que se prevé implementación de mitigación en fase 3 según lo establecido en el "Anexo A. Requisitos de la aplicación".

Tabla 34. Medidas de seguridad para mitigación de amenazas.

Medida de seguridad	Requisito de la aplicación software
Sanitización de las entradas	SEG-03, SEG-05, SEG-06
Sanitización de las entradas / salidas	SEG-03, SEG-05, SEG-06
Sanitización de entradas que van a ser incluidas en consultas No SQL	SEG-06
Controles de autenticación y autorización proporcionados por el sistema operativo	SEG-10
Controles de autenticación y autorización proporcionados por el sistema gestor de la base de datos	SEG-04
Conexión a la base de datos mediante librerías estándar	BD-04
La aplicación se ejecuta en entorno virtualizado	SEG-09
Análisis estático de código	SEG-13
Pruebas realizadas tras la codificación	SEG-07
Rotación de direcciones IP	CRA-09
Las excepciones se capturan en bloques *try-catch*	SEG-01
Logging y auditoría de errores software	SEG-02
Los ficheros se escriben utilizando librerías estándar	SEG-17

Fuente: Elaboración propia.

La Tabla 35 recoge las amenazas cuyo valor de riesgo es 'Medio' ('*Medium*') o 'Alto' ('*High*'). Para cada una de ellas se especifica:

- Vulnerabilidades asociadas a la amenaza mediante el listado MITRE CWE (*Common Weakness Enumeration*).
- Modelos de ataque recogidos en el listado MITRE CAPEC (*Common Attack Pattern Enumeration and Classification*).

Tabla 35. Vulnerabilidades CWE y patrones de ataque CAPEC asociados a amenazas.

Amenaza	Interacción(es)	Riesgo	Vulnerabilidad CWE	Patrón ataque CAPEC
Potencial vulnerabilidad de inyección No SQL– Base de datos	• Petición a la base de datos	30 (Medio)	CWE-943: Improper Neutralization of Special Elements in Data Query Logic CWE-1286: Improper Validation of Syntactic Correctness of Input	CAPEC-676: NoSQL Injection

Amenaza	Interacción(es)	Riesgo	Vulnerabilidad CWE	Patrón ataque CAPEC
Potencial falta de validación de entrada – Aplicación DarkFinder	• Consulta • Respuesta HTTP • Respuesta HTTPS • Fichero configuración	24 (Medio)	CWE-20: Improper Input Validation	CAPEC-23: File Content Injection CAPEC-28: Fuzzing CAPEC-63: Cross-Site Scripting (XSS) CAPEC-153: Input Data Manipulation
Cross Site Scripting	• Respuesta HTTP • Respuesta HTTPS	24 (Medio)	CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') CWE-80: Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	CAPEC-18: XSS Targeting Non-Script Elements CAPEC-63: Cross-Site Scripting (XSS) CAPEC-592: Stored XSS

Fuente: Elaboración propia a partir de (The MITRE Corporation, 2022b) y (The MITRE Corporation, 2022a).

Nota: si bien el contenido de las páginas HTML almacenado en base de datos no se muestra en un navegador web, se contempla su sanitización por si pudiera realizarse una copia a fichero HTML.

Informe de modelado de amenazas

Se adjunta a continuación el resultado del modelado de amenazas proporcionado por la herramienta Microsoft Threat Modelling Tool.

Threat	Interaction	Priority	State	Risk	Safeguards
Cross Site Scripting	HTTPS Response HTTP Response	Medium	Mitigated	24	Input / output sanitization
DarkFinder May be Subject to Elevation of Privilege Using Remote Code Execution	HTTP Response HTTPS Response	Low	Mitigated	18	Static Code Analysis Test carried out after coding The application is run in a virtualized environment
Data Flow Is Potentially Interrupted	HTTP Response HTTPS Response	Low	Not Started	10	Rotation of IP address
Data Flow Sniffing	HTTP Response	Low	Not Applicable	-	-
Data Logs from an Unknown Source	Log file writing	Low	Not Applicable	-	-
Elevation by Changing the Execution Flow in DarkFinder	HTTP Response HTTPS Response	Low	Mitigated	18	Static Code Analysis Test carried out after coding The application is run in a virtualized environment
Elevation Using Impersonation	Query	Low	Mitigated	18	Static Code Analysis Test carried out after coding The application is run in a virtualized environment
	HTTP Response HTTPS Response	Low	Not Applicable	-	-
Insufficient Auditing	Log file writing	Low	Mitigated	18	Logging and auditing software errors
Lower Trusted Subject Updates Logs	Log file writing	Low	Mitigated	6	Authentication and authorization control provided by the operating system

Threat	Interaction	Priority	State	Risk	Safeguards
Potential Data Repudiation by DarkFinder	HTTP Response	Low	Not Applicable	-	-
	HTTPS Response				
Potential Excessive Resource Consumption	Log file writing	Low	Mitigated	6	Files are written using standard libraries Exceptions are captured by try-catch blocks
	Database Request	Low	Mitigated	18	Connection to database through standard libraries Exceptions are captured by try-catch blocks
Potential Lack of Input Validation for DarkFinder	Query	Medium	Mitigated	24	Input sanitization
	HTTP Response				
	HTTPS Response				
	Configuration file reading				
Potential No SQL Injection Vulnerability for Non Relational Database	Database Request	Medium	Mitigated	30	Sanitization of inputs that are going to be included in NoSQL Queries
Potential Process Crash or Stop for DarkFinder	HTTP Response	Low	Mitigated	18	Exceptions are captured by try-except blocks
	HTTPS Response				
Potential Weak Protections for Audit Data	Log file writing	Low	Mitigated	6	Authentication and authorization control provided by the operating system
Risks from Logging	Log file writing	Low	Not Applicable	-	-
Spoofing of Destination Data Store	Log file writing	Low	Not Applicable	-	-
	Database Request	Low	Mitigated	15	Authentication and Authorization protection provided by the database management system

Threat	Interaction	Priority	State	Risk	Safeguards
Spoofing of Source Data Store	Configuration file reading	Low	Mitigated	12	Authentication and authorization control provided by the operating system Input sanitization
	Database Response	Low	Mitigated	15	Authentication and authorization protection provided by the database management system
Spoofing the DarkFinder Process	HTTP Response	Low	Not Applicable	-	-
Spoofing the Human User External Entity	Query	Low	Mitigated	12	Authentication and Authorization protection provided by the database management system
Spoofing the Web Server Process	HTTP Response	Low	Not Applicable	-	-
	HTTPS Response				
Weak Access Control for a Resource	Database Response	Low	Mitigated	6	Authentication and authorization protection provided by the database management system Authentication and authorization control provided by the operating system
	Configuration file reading	Low	Mitigated	6	Authentication and authorization control provided by the operating system

Fuente: Elaboración propia.

Anexo C. Análisis de riesgos arquitectónico

Se incluye en el presente anexo:

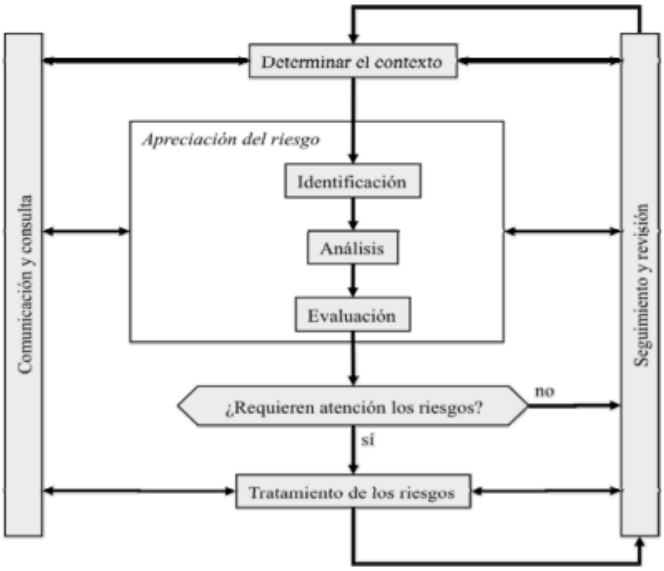
- Metodología.
- Determinación del contexto.
- Apreciación del riesgo, incluyendo:
  - Listado de activos identificados y dependencias.
  - Valoración del impacto.
  - Listado de amenazas.
  - Listado de riesgos residuales resultantes.
- Tratamiento del riesgo.

Metodología

En el análisis de riesgos se emplea la metodología MAGERIT (Ministerio de Hacienda y Administraciones Públicas, 2012a), siguiendo los pasos establecidos en la Figura 51. Las acciones realizadas incluyen:

- Determinación del contexto.
- Apreciación del riesgo, en el que se realizan los siguientes pasos:
  - Identificar el conjunto de activos correspondientes a la aplicación instalada.
  - Determinar las dependencias entre activos.
  - Identificar el conjunto de amenazas sobre los activos.
  - Analizar y evaluar el riesgo, por medio de la evaluación del impacto que un incidente de seguridad puede ocasionar en la organización, así como la probabilidad de que ocurra.
- Tratamiento del riesgo, estableciendo alguna acción para gestionarlo adecuadamente.

Figura 51. Metodología de análisis de riesgos.



Fuente: (Ministerio de Hacienda y Administraciones Públicas, 2012a).

Determinación del contexto

Se determina el contexto externo e interno siguiendo los criterios establecidos en la norma UNE-ISO 31000 Gestión del riesgo. Directrices.

Apreciación de riesgo

Para la **identificación de activos** se toma como referencia el estándar TOGAF (*The Open Group Architecture Framework*) (The Open Group, 2018), marco de trabajo que proporciona un enfoque para el diseño, planificación, implementación y gobierno de una arquitectura empresarial de sistemas de información. Para la definición se consideran tres dimensiones:

- **Capa de negocio**: hace referencia a la estrategia del negocio, gobierno, estructura y procesos clave de la organización.
- **Capa de aplicaciones**: hace referencia a los sistemas de información de la organización. Dentro de este nivel se incluye la definición de la arquitectura de datos, describiendo la estructura de datos físicos y lógicos de la aplicación.
- **Capa tecnológica**: describe la estructura de hardware, software y redes requerida para dar soporte a la implantación de los sistemas de información descritos en la capa de aplicaciones.

La Tabla 36 recoge los **criterios de categorización de impacto** sobre los activos de información y servicio empleados en el análisis de riesgos, tomando como referencia lo establecido en (Centro Criptológico Nacional, 2020) y (Ministerio de Hacienda y Administraciones Públicas, 2012b). Se definen cinco niveles de requisito: Muy bajo [MB], Bajo

[B], Medio [M], Alto [A] y Muy Alto [MA]; y dos criterios de impacto: negocio y legal (en base a posibles incumplimientos de la legislación de protección de datos).

Nota: no se evalúan los posibles impactos en base a criterios estatutario (exigidos por los estatutos de la compañía), regulatorio (normativa o regulación sectorial) y contractual (establecidos por contratos con clientes) por no resultar de aplicación en este caso.

Tabla 36. Criterios de categorización de activos de información y servicio en el análisis de riesgos.

Criterio	Nivel	Criterio de evaluación
Negocio	MB	• Pérdidas económicas despreciables. • Daño reputacional despreciable.
	B	• Mermas de ingresos (hasta 2% de facturación). • Pérdida menor de confianza con clientes.
	M	• Pérdidas económicas apreciables (hasta 5% de facturación). • Pérdida mayor de confianza con clientes.
	A	• Pérdidas económicas considerables (hasta 15% de facturación). • Publicidad negativa generalizada leve, con publicación puntual en medios de comunicación.
	MA	• Pérdidas económicas excepcionales (a partir de 15% de facturación). • Publicidad negativa generalizada grave.
Legal	MB	• Sin incumplimiento. • No afectando a datos personales.
	B	• Podría suponer incumplimiento legal leve. • Afecta a datos personales parciales de una persona.
	M	• Infracción con penas leves. • Afecta a datos personales completos de una persona.
	A	• Infracción con penas graves. • Afecta a datos personales parciales de múltiples personas.
	MA	• Infracción con penas muy graves. • Afecta a datos personales de categoría especial o datos completos de múltiples personas.

Fuente: Elaboración propia a partir de (Ministerio de Hacienda y Administraciones Públicas, 2012b).

La probabilidad se determina en función de los criterios establecidos en la Tabla 37.

Tabla 37. Criterios de evaluación de probabilidad en el análisis de riesgos.

Nivel	Criterio de evaluación
MB	Probabilidad estimada de ocurrencia de la amenaza muy baja: Repetición cada > 100 años (“Una vez en siglos”).
B	Probabilidad estimada de ocurrencia de la amenaza baja: Repetición entre 10 y 100 años (“Una vez en décadas”).
M	Probabilidad estimada de ocurrencia de la amenaza media: Repetición entre 1 y 10 años (“Una vez al año”).
A	Probabilidad estimada de ocurrencia de la amenaza alta: Repetición entre 1 y 12 meses (“Una vez al mes”).
MA	Probabilidad estimada de ocurrencia de la amenaza muy alta: Repetición < 1 mes (“Una vez al día”).

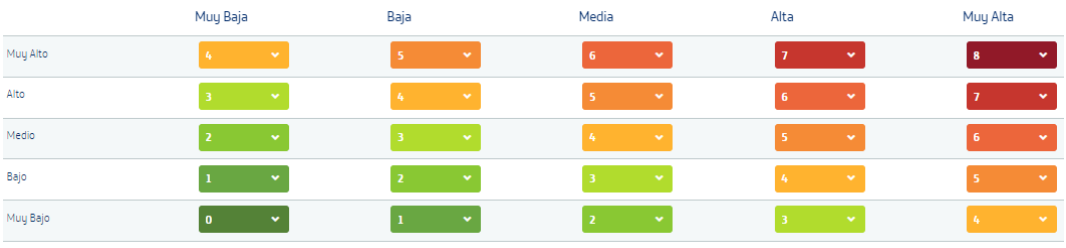
Fuente: Elaboración propia a partir de (Ministerio de Hacienda y Administraciones Públicas, 2012b).

El cálculo del análisis de riesgos se realiza en base a la determinación de los siguientes valores:

- **Probabilidad**, calculada en función de los siguientes parámetros:
  - Probabilidad inherente de que se materialice la amenaza (independientemente de los controles implantados), según los criterios de la Tabla 37.
  - Controles de seguridad implantados para hacer frente a la amenaza.
- **Impacto** sobre la organización en caso de materializarse la amenaza, calculado a partir de:
  - Valoración del activo, según los criterios de la Tabla 36.
  - Degradación del activo en caso de materialización de la amenaza, según los rangos: N/A, 0%, 0%-30%, 30%-70%, 70%-100%, 100%.

En función de estos parámetros se calcula el riesgo en base a la matriz representada en la Figura 52:

Figura 52. Matriz de cálculo de riesgos.



Fuente: Elaboración propia.

Sólo se aceptan aquellos riesgos cuyo valor resultante corresponda a un nivel 'Aceptable' según lo indicado en la Tabla 38:

Tabla 38. Criterios de aceptación del riesgo.

Nivel	Criterio de aceptación
Aceptable	Riesgo Bajo (≤ 3).
Tolerable	Riesgo Medio (entre 4 y 6).
No aceptable	Riesgo Alto (≥ 7).

Fuente: Elaboración propia.

Tratamiento del riesgo

Para los riesgos cuyo valor supere el umbral de aceptación de la organización se determina una medida de seguridad para su mitigación. Estas medidas se asocian con la especificación realizada en el "Anexo A. Requisitos de la aplicación".

Determinación del contexto

Como paso inicial de la realización del análisis de riesgos se realiza un estudio de la organización y su contexto.

Análisis de contexto externo

Del análisis de contexto y del estado del arte realizado (ver capítulos "1 Introducción" y "2 Contexto y estado del arte") pueden extraerse los siguientes principios:

- Incremento del número de ciberataques e importancia de las amenazas.
- Incremento de la actividad delictiva en la Dark Web durante la pandemia COVID-19.

Análisis de contexto interno

El análisis de la organización arroja las siguientes conclusiones:

- La organización dispone de una cultura de la seguridad de la información, con personal formado en la aplicación de medidas de seguridad y el desarrollo de software seguro.
- La entidad cuenta con medidas adecuadas de seguridad de la información, incluyendo protección perimetral que protegen frente a ataques procedentes del exterior.
- La entidad tiene un apetito de riesgo bajo, no deseando asumir riesgos que puedan suponer importantes impactos en el negocio.

La Tabla 39 recoge un cuadro DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) resumiendo el análisis de contexto externo e interno realizado.

Tabla 39. DAFO con el análisis del contexto externo e interno.

DEBILIDADES	FORTALEZAS
-	• F1. Cultura de la seguridad. • F2. Medidas de seguridad.
AMENAZAS	**OPORTUNIDADES**
• A1. Incremento de ciberataques. • A2. Incremento de la actividad delictiva en la Dark Web.	-

Fuente: Elaboración propia.

Apreciación del riesgo

En total se han identificado 15 activos, de los cuales 3 corresponden a activos esenciales (información o servicios), para los cuales se evalúa el impacto en las dimensiones de confidencialidad, integridad y disponibilidad con los criterios especificados en el presente anexo: Fichero configuración, Servicio *crawling* en la Dark Web e Información *crawling*. El resultado de la valoración se recoge en la Tabla 40:

Tabla 40. Impacto considerado en los activos esenciales por cada dimensión.

Dimensión	Impacto
Confidencialidad	Muy bajo
Integridad	Bajo
Disponibilidad	Bajo

Fuente: Elaboración propia.

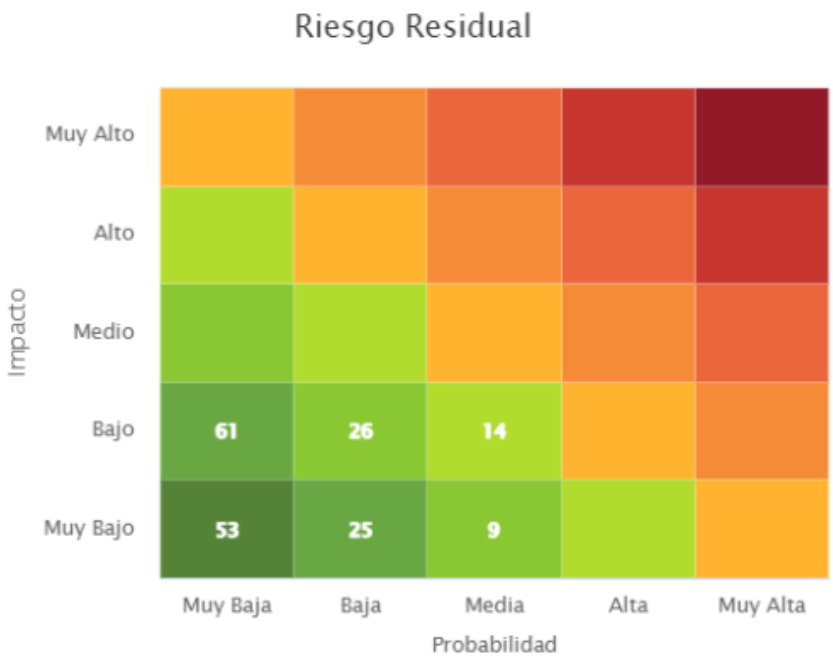
Como resultado del proceso de **identificación, análisis y evaluación de los riesgos** se han identificado un total de 188 riesgos arquitectónicos, con la distribución de riesgos residuales recogida en la Tabla 41 y Figura 53:

Tabla 41. Distribución de riesgos arquitectónicos residuales.

Tipo riesgo	Valor
No aceptables - Riesgo Alto (≥ 7)	0
Tolerables - Riesgo Medio (entre 4 y 6)	0
Aceptables - Riesgo Bajo (≤ 3)	188
Total	**188**

Fuente: Elaboración propia.

Figura 53. Distribución de riesgo arquitectónico residual.



Fuente: Elaboración propia.

Listado de activos identificados y dependencias

En base a la metodología identificada en el presente anexo, se han identificado los activos recogidos en la Tabla 42 y representados en la Figura 54:

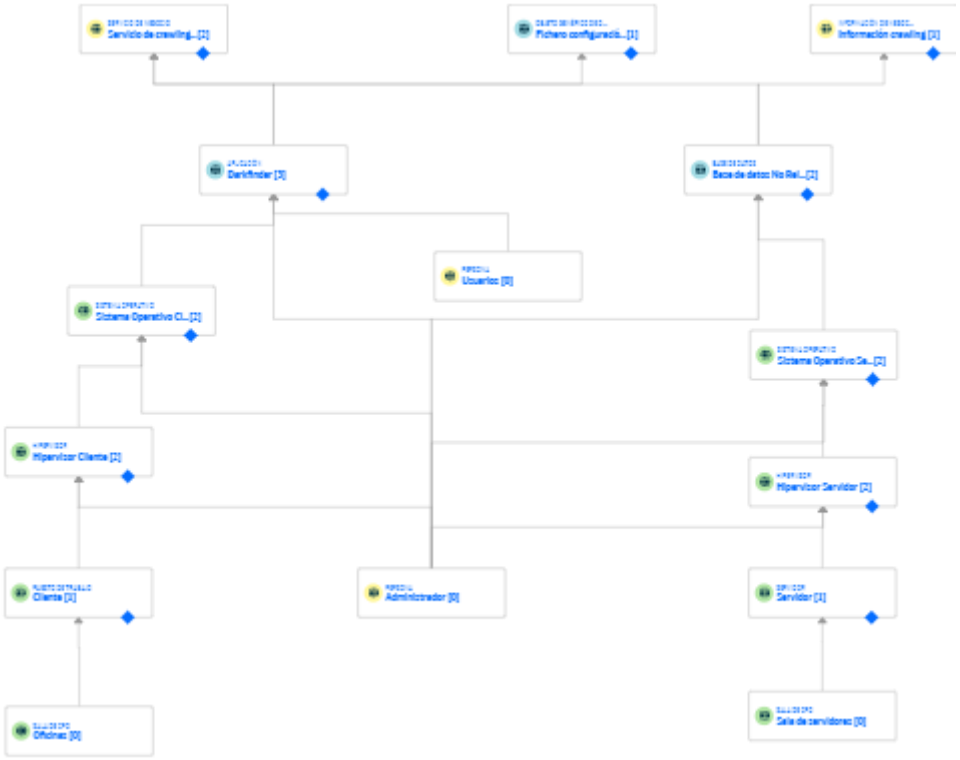
Tabla 42. Activos identificados en el análisis de riesgos por capa TOGAF.

Capa	Activos
Negocio	• **Fichero configuración**: fichero de configuración de la aplicación DarkFinder. • **Servicio *crawling* en la Dark Web**: servicio que permite realizar *crawling* en la Dark Web y almacenar la información encontrada. Requiere para su funcionamiento que tanto la base de datos no relacional que almacena la información como la aplicación DarkFinder estén operativas. • **Información *crawling***: información obtenida como consecuencia del *crawling* y almacenada en la base de datos no relacional. • **Usuarios**: personal que ejecuta la aplicación. • **Administrador**: personal encargado de configurar todo el software relacionado con la aplicación, incluyendo sistema operativo, virtualización y base de datos.
Aplicaciones	• **DarkFinder**: aplicación cliente que ejecuta un script para el rastreo de información en la web visible y Dark Web o realiza consultas en base a la información obtenida. • **Base de datos no relacional**: almacena la información obtenida como consecuencia del rastreo para ser consultada.

Capa	Activos
Tecnológica	**Cliente:** • **Sistema operativo cliente:** sistema operativo en el que se ejecutan los scripts Python de la aplicación DarkFinder. • **Hipervisor cliente**: virtualización empleada en el cliente. • **Cliente**: equipo cliente en el que se ejecuta la aplicación DarkFinder. • **Oficinas**: instalaciones en las que se encuentran los clientes que ejecutan la aplicación. **Servidor:** • **Sistema operativo servidor:** sistema operativo en el que se instala la base de datos no relacional. • **Hipervisor servidor**: virtualización empleada en el servidor. • **Servidor**: equipo servidor en el que se ejecuta la base de datos no relacional. • **Sala de servidores**: instalaciones en las que se encuentra el servidor que aloja la base de datos con los datos captados por la aplicación.

Fuente: Elaboración propia.

Figura 54. Activos identificados en el análisis de riesgos y dependencias



Fuente: Elaboración propia.

Valoración del impacto

La Tabla 43 recoge la valoración de impacto realizada para los activos esenciales (información / servicio) según los criterios especificados en el presente anexo.

Tabla 43. Valoración del impacto en los activos esenciales en el análisis de riesgos arquitectónico.

Activo	Criterio	Confidencialidad	Integridad	Disponibilidad
Información *crawling*	Negocio	Muy bajo	Bajo	Bajo
	Legal	Muy bajo	No asignado	No asignado
Fichero configuración	Negocio	Muy bajo	Bajo	Bajo
	Legal	No asignado	No asignado	No asignado
Servicio de *crawling* en la Dark Web	Negocio	No asignado	No asignado	Bajo
	Legal	No asignado	No asignado	No asignado

Fuente: Elaboración propia.

Listado de amenazas

La Tabla 44 recoge el conjunto de amenazas contemplado en el análisis de riesgos arquitectónico, tomando como referencia el catálogo establecido en (Ministerio de Hacienda y Administraciones Públicas, 2012b).

Tabla 44. Listado de amenazas contempladas en el análisis de riesgos arquitectónico.

Amenaza	Tipo
[A.03]-Manipulación de los registros de actividad (log)	[A] – Ataques intencionados
[A.04]-Manipulación de la configuración	
[A.05]-Suplantación de la identidad del usuario	
[A.06]-Abuso de los privilegios de acceso	
[A.07]-Uso no previsto	
[A.08]-Difusión de software dañino	
[A.09]-Re-encaminamiento de mensajes	
[A.10]-Alteración de secuencia	
[A.11]-Acceso no autorizado	
[A.12]-Análisis de tráfico	
[A.13]-Repudio	
[A.14]-Interceptación de la información (escucha)	
[A.15]-Modificación deliberada de la información	
[A.18]-Destrucción de información	
[A.19]-Divulgación de información	
[A.22]-Manipulación de los programas	
[A.23]-Manipulación de los equipos	
[A.24]-Denegación de servicio	
[A.25]-Robo	
[A.26]-Ataque destructivo	
[A.27]-Ocupación enemiga	
[A.28]-Indisponibilidad del personal	
[A.29]-Extorsión	
[A.30]-Ingeniería social (picaresca)	

Amenaza	Tipo
[E.01]-Errores de los usuarios	[E]-Errores y fallos no intencionados
[E.02]-Errores del administrador	
[E.03]-Errores de monitorización (logs)	
[E.04]-Errores de configuración	
[E.07]-Deficiencias en la organización	
[E.08]-Difusión de software dañino	
[E.09]-Errores de re-encaminamiento	
[E.10]-Errores de secuencia	
[E.14]-Escapes de información	
[E.15]-Alteración accidental de la información	
[E.18]-Destrucción de información	
[E.19]-Fugas de información	
[E.20]-Vulnerabilidades de los programas (software)	
[E.21]-Errores de mantenimiento / actualización de programas (software)	
[E.23]-Errores de mantenimiento / actualización de los equipos (hardware)	
[E.24]-Caída del sistema por agotamiento de recursos	
[E.25]-Pérdida de equipos	
[E.28]-Indisponibilidad del personal	
[I.*]-Desastres industriales	[I]-De origen industrial
[I.01]-Fuego	
[I.02]-Daños por agua	
[I.03]-Contaminación mecánica	
[I.04]-Contaminación electromagnética	
[I.05]-Avería de origen lógico o físico	
[I.06]-Corte del suministro eléctrico	
[I.07]-Condiciones inadecuadas de temperatura o humedad	
[I.08]-Fallo de los servicios de comunicaciones	
[I.09]-Interrupción de otros servicios y suministros esenciales	
[I.10]-Degradación de los soportes de almacenamiento de información	
[I.11]-Emanaciones electromagnéticas	
[N.*]-Desastres naturales	[N]-Desastres naturales
[N.01]-Fuego	
[N.02]-Daños por agua	

Fuente: (Ministerio de Hacienda y Administraciones Públicas, 2012b).

Listado de riesgos residuales resultantes

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Base de datos No Relacional	[A.03]	Manipulación de los registros de actividad (log)	[C.02] - Configuración segura del software (bastionado)	Baja	Bajo	100%	Bajo	2
Darkfinder				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Servidor				Baja	Bajo	100%	Bajo	2
Sistema Operativo Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Sistema Operativo Servidor				Baja	Bajo	100%	Bajo	2
Base de datos No Relacional	[A.04]	Manipulación de la configuración	[C.02] - Configuración segura del software (bastionado)	Baja	Bajo	100%	Bajo	2
Darkfinder				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Servidor				Baja	Bajo	100%	Bajo	2
Sistema Operativo Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Sistema Operativo Servidor				Baja	Bajo	100%	Bajo	2
Base de datos No Relacional	[A.05]	Suplantación de la identidad del usuario	[C.05] - Controles de autenticación y autorización proporcionados por el SO y/o SGBD	Muy Baja	Bajo	100%[10]	Muy Bajo	0
Sistema Operativo Cliente				Muy Baja	Bajo	30% - 70%[11]	Muy Bajo	0
Sistema Operativo Servidor				Muy Baja	Bajo	100%[12]	Muy Bajo	0

[10] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[11] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[12] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Base de datos No Relacional	[A.06]	Abuso de los privilegios de acceso	[C.05] - Controles de autenticación y autorización proporcionados por el SO y/o SGBD	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Sistema Operativo Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Sistema Operativo Servidor				Muy Baja	Bajo	100%	Bajo	1
Base de datos No Relacional	[A.07]	Uso no previsto	[C.05] - Controles de autenticación y autorización proporcionados por el SO y/o SGBD	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Hipervisor Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Hipervisor Servidor				Muy Baja	Bajo	100%	Bajo	1
Sistema Operativo Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Sistema Operativo Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente			-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Base de datos No Relacional	[A.08]	Difusión de software dañino	[C.04] - Software antimalware	Media	Bajo	100%	Bajo	3
Darkfinder				Media	Bajo	30% - 70%	Muy Bajo	2
Sistema Operativo Cliente				Media	Bajo	30% - 70%	Muy Bajo	2
Sistema Operativo Servidor				Media	Bajo	100%	Bajo	3
Base de datos No Relacional	[A.09]	Re-encaminamiento de mensajes	-	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Base de datos No Relacional	[A.10]	Alteración de secuencia	-	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Base de datos No Relacional	[A.11]	Acceso no autorizado	[C.05] - Controles de autenticación y autorización proporcionados por el SO y/o SGBD	Muy Baja	Bajo	100%[13]	Muy Bajo	0
Darkfinder				Muy Baja	Bajo	30% - 70%[14]	Muy Bajo	0
Sistema Operativo Cliente				Muy Baja	Bajo	30% - 70%[15]	Muy Bajo	0
Sistema Operativo Servidor				Muy Baja	Bajo	100%[16]	Muy Bajo	0
Oficinas			-	Baja	Bajo	0% - 30%[17]	Muy Bajo	1
Sala de servidores				Muy Baja	Bajo	70% - 100%[18]	Muy Bajo	0
Cliente				Muy Baja	Bajo	0% - 30%[19]	Muy Bajo	0
Servidor				Muy Baja	Bajo	70% - 100%[20]	Muy Bajo	0
Base de datos No Relacional	[A.12]	Análisis de tráfico	-	Baja	Bajo	100%[21]	Muy Bajo	1
Darkfinder				Baja	Bajo	30% - 70%[22]	Muy Bajo	1
Base de datos No Relacional	[A.13]	Repudio	-	Muy Baja	Bajo	0% - 30%	Muy Bajo	0
Darkfinder				Muy Baja	Bajo	0% - 30%	Muy Bajo	0

[13] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[14] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[15] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[16] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[17] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[18] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[19] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[20] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[21] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[22] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Base de datos No Relacional	[A.14]	Interceptación de la información (escucha)	-	Baja	Bajo	100%[23]	Muy Bajo	1
Darkfinder				Baja	Bajo	30% - 70%[24]	Muy Bajo	1
Base de datos No Relacional	[A.15]	Modificación deliberada de la información	[C.01] - Copia de seguridad	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Base de datos No Relacional	[A.18]	Destrucción de información	[C.01] - Copia de seguridad	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Base de datos No Relacional	[A.19]	Divulgación de información	-	Muy Baja	Bajo	100%[25]	Muy Bajo	0
Darkfinder				Muy Baja	Bajo	30% - 70%[26]	Muy Bajo	0
Base de datos No Relacional	[A.22]	Manipulación de los programas	-	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Baja	Bajo	100%	Bajo	2
Hipervisor Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Hipervisor Servidor				Muy Baja	Bajo	100%	Bajo	1
Sistema Operativo Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Sistema Operativo Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente	[A.23]	Manipulación de los equipos	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1

[23] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[24] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[25] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[26] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Base de datos No Relacional	[A.24]	Denegación de servicio	-	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Hipervisor Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Hipervisor Servidor				Muy Baja	Bajo	100%	Bajo	1
Sistema Operativo Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Sistema Operativo Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente	[A.25]	Robo	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Oficinas	[A.26]	Ataque destructivo	-	Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1
Cliente				Muy Baja	Bajo	100%	Bajo	1
Servidor				Muy Baja	Bajo	100%	Bajo	1
Oficinas	[A.27]	Ocupación enemiga	-	Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1
Administrador	[A.28]	Indisponibilidad del personal	-	Muy Baja	Bajo	100%	Bajo	1
Usuarios				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Administrador	[A.29]	Extorsión	-	Muy Baja	Bajo	100%	Bajo	1
Usuarios				Muy Baja	Bajo	30% - 70%	Muy Bajo	0

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Administrador	[A.30]	Ingeniería social (picaresca)	-	Muy Baja	Bajo	100%	Bajo	1
Usuarios				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Darkfinder	[E.01]	Errores de los usuarios	-	Media	Bajo	30% - 70%	Muy Bajo	2
Darkfinder	[E.02]	Errores del administrador	-	Media	Bajo	100%	Bajo	3
Base de datos No Relacional			[C.02] - Configuración segura del software (bastionado)	Baja	Bajo	100%	Bajo	2
Hipervisor Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Servidor				Baja	Bajo	100%	Bajo	2
Sistema Operativo Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Sistema Operativo Servidor				Baja	Bajo	100%	Bajo	2
Base de datos No Relacional	[E.03]	Errores de monitorización (logs)	-	Baja	Bajo	100%	Bajo	2
Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Darkfinder				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Servidor				Baja	Bajo	100%	Bajo	2
Servidor				Baja	Bajo	100%	Bajo	2
Sistema Operativo Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Sistema Operativo Servidor				Baja	Bajo	100%	Bajo	2

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Darkfinder	[E.04]	Errores de configuración	-	Media	Bajo	100%	Bajo	3
Base de datos No Relacional			[C.02] - Configuración segura del software (bastionado)	Baja	Bajo	100%	Bajo	2
Hipervisor Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Hipervisor Servidor				Baja	Bajo	100%	Bajo	2
Sistema Operativo Cliente				Baja	Bajo	30% - 70%	Muy Bajo	1
Sistema Operativo Servidor				Baja	Bajo	100%	Bajo	2
Administrador	[E.07]	Deficiencias en la organización	-	Muy Baja	Bajo	100%	Bajo	1
Usuarios				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Base de datos No Relacional	[E.08]	Difusión de software dañino	[C.04] - Software antimalware	Media	Bajo	100%	Bajo	3
Darkfinder				Media	Bajo	30% - 70%	Muy Bajo	2
Sistema Operativo Cliente				Media	Bajo	30% - 70%	Muy Bajo	2
Sistema Operativo Servidor				Media	Bajo	100%	Bajo	3
Base de datos No Relacional	[E.09]	Errores de re-encaminamiento	-	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Base de datos No Relacional	[E.10]	Errores de secuencia	-	Muy Baja	Bajo	100%	Bajo	1
Darkfinder				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Base de datos No Relacional	[E.14]	Escapes de información	-	Muy Baja	Bajo	100%[27]	Muy Bajo	0
Darkfinder				Muy Baja	Bajo	30% - 70%[28]	Muy Bajo	0

[27] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[28] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Base de datos No Relacional	[E.15]	Alteración accidental de la información	[C.01] - Copia de seguridad	Baja	Bajo	70% - 100%	Bajo	2
Darkfinder				Baja	Bajo	30% - 70%	Muy Bajo	1
Base de datos No Relacional	[E.18]	Destrucción de información	[C.01] - Copia de seguridad	Baja	Bajo	100%	Bajo	2
Darkfinder				Baja	Bajo	30% - 70%	Muy Bajo	1
Base de datos No Relacional	[E.19]	Fugas de información	-	Muy Baja	Bajo	100%[29]	Muy Bajo	0
Darkfinder				Muy Baja	Bajo	30% - 70%[30]	Muy Bajo	0
Base de datos No Relacional	[E.20]	Vulnerabilidades de los programas (software)	[C.03] - Revisión y actualización de software vulnerable	Media	Bajo	100%	Bajo	3
Darkfinder				Media	Bajo	100%	Bajo	3
Hipervisor Cliente				Media	Bajo	30% - 70%	Muy Bajo	2
Hipervisor Servidor				Media	Bajo	100%	Bajo	3
Sistema Operativo Cliente				Media	Bajo	30% - 70%	Muy Bajo	2
Sistema Operativo Servidor				Media	Bajo	100%	Bajo	3
Base de datos No Relacional	[E.21]	Errores de mantenimiento / actualización de programas (software)	[C.03] - Revisión y actualización de software vulnerable	Media	Bajo	100%	Bajo	3
Darkfinder				Media	Bajo	100%	Bajo	3
Hipervisor Cliente				Media	Bajo	30% - 70%	Muy Bajo	2
Hipervisor Servidor				Media	Bajo	100%	Bajo	3
Sistema Operativo Cliente				Media	Bajo	30% - 70%	Muy Bajo	2
Sistema Operativo Servidor				Media	Bajo	100%	Bajo	3

[29] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.
[30] Aplica sólo a la dimensión de confidencialidad, para la que el valor del activo es 'Muy Bajo'.

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Cliente	[E.23]	Errores de mantenimiento / actualización de los equipos (hardware)	-	Baja	Bajo	30% - 70%	Muy Bajo	1
Servidor				Baja	Bajo	100%	Bajo	2
Cliente	[E.24]	Caída del sistema por agotamiento de recursos	-	Baja	Bajo	30% - 70%	Muy Bajo	1
Servidor				Baja	Bajo	100%	Bajo	2
Cliente	[E.25]	Pérdida de equipos	-	Baja	Bajo	30% - 70%	Muy Bajo	1
Servidor				Muy Baja	Bajo	100%	Bajo	1
Administrador	[E.28]	Indisponibilidad del personal	-	Muy Baja	Bajo	100%	Bajo	1
Usuarios				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Cliente	[I.*]	Desastres industriales	-	Muy Baja	Bajo	100%	Bajo	1
Servidor				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1
Cliente	[I.01]	Fuego	-	Muy Baja	Bajo	100%	Bajo	1
Servidor				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1
Cliente	[I.02]	Daños por agua	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Cliente	[I.03]	Contaminación mecánica	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente	[I.04]	Contaminación electromagnética	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente	[I.05]	Avería de origen lógico o físico	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente	[I.06]	Corte del suministro eléctrico	-	Baja	Bajo	100%	Bajo	2
Servidor				Baja	Bajo	100%	Bajo	2
Cliente	[I.07]	Condiciones inadecuadas de temperatura o humedad	-	Baja	Bajo	30% - 70%	Muy Bajo	1
Servidor				Baja	Bajo	100%	Bajo	2
Cliente	[I.08]	Fallo de los servicios de comunicaciones	-	Baja	Bajo	100%	Bajo	2
Servidor				Baja	Bajo	100%	Bajo	2
Cliente	[I.09]	Interrupción de otros servicios y suministros esenciales	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Cliente	[I.10]	Degradación de los soportes de almacenamiento de información	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1

Activo	Cód. Evento	Evento	Controles	Probabilidad	Valor. Activo	Degradación	Impacto	Riesgo
Cliente	[I.11]	Emanaciones electromagnéticas	-	Muy Baja	Bajo	30% - 70%	Muy Bajo	0
Servidor				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	0%-30%	Muy Bajo	0
Sala de servidores				Muy Baja	Bajo	0%-30%	Muy Bajo	0
Servidor	[N.*]	Desastres naturales	-	Muy Baja	Bajo	100%	Bajo	1
Cliente				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1
Servidor	[N.01]	Fuego	-	Muy Baja	Bajo	100%	Bajo	1
Cliente				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1
Servidor	[N.02]	Daños por agua	-	Muy Baja	Bajo	100%	Bajo	1
Cliente				Muy Baja	Bajo	100%	Bajo	1
Oficinas				Muy Baja	Bajo	100%	Bajo	1
Sala de servidores				Muy Baja	Bajo	100%	Bajo	1

Fuente: Elaboración propia.

Tratamiento del riesgo

De un modo equivalente al realizado para el modelado de amenazas, se han considerado un conjunto de medidas de seguridad para la mitigación de los riesgos. La Tabla 45 recoge, para cada una de ellas, referencia al requisito correspondiente según la especificación realizada en el "Anexo A. Requisitos de la aplicación".

Nota: debe tenerse en cuenta que únicamente se recogen en este análisis de riesgos las medidas específicas para la aplicación desarrollada en el ámbito del presente TFM, no contemplándose el resto de medidas disponibles de manera transversal para toda la organización (como, por ejemplo, el sistema de protección perimetral).

Tabla 45. Medidas de seguridad para mitigación de riesgos arquitectónicos.

Medida de seguridad	Requisito de la aplicación software
[C.01] - Copia de seguridad	SEG-11
[C.02] - Configuración segura del software (bastionado)	SEG-12
[C.03] - Revisión y actualización de software vulnerable	SEG-14
[C.04] - Software antimalware	SEG-15
[C.05] - Controles de autenticación y autorización proporcionados por el sistema operativo y/o sistema gestor de base de datos	SEG-04, SEG-10

Fuente: Elaboración propia.

Anexo D. Diseño de clases

Se incluye a continuación el detalle de los atributos y métodos de las clases especificadas en el capítulo “4.2 Diseño”:

- *Configuration*.
- *Page*.
- *Image*.

Configuration

Tabla 46. Atributos Clase *Configuration*.

Atributo	Visibilidad	Tipo	Descripción
locale	Privada	String	Indicador del idioma seleccionado por el usuario: ‘es’ (español), ‘en’ (inglés).
trans	Privada	Objeto tipo gettext.Translation	Clase empleada para realizar las traducciones de los textos mostrados por la aplicación.
quiet	Privada	Booleano	Indica si se evita mostrar mensajes de tipo ‘*debug*’ y ‘*warning*’ (opción -q, --quiet de línea de comandos DarkFinder).
start_urls	Privada	Array de strings	Valores de URL de las semillas desde las que parte el *crawler* para realizar el rastreo.
max_depth_crawling	Privada	Integer	Máxima profundidad de rastreo.
max_crawled_web_pages	Privada	Integer	Máximo número de páginas web que son rastreadas.
crawling_strategy	Privada	String	Estrategia de rastreo: ‘bfs’ (*Breadth-First Search*), ‘dfs’ (*Depth-First Search*), ‘best-fs’ (*Best First Search*).
keywords	Privada	Array de strings	Conjunto de palabras clave que son utilizadas para priorizar webs en una estrategia de tipo *Best First Search.*
min_image_width	Privada	Integer	Anchura mínima de una imagen en una web para ser registrada.

Atributo	Visibilidad	Tipo	Descripción
min_image_height	Privada	Integer	Altura mínima de una imagen en una web para ser registrada.
db_user	Privada	String	Usuario para la base de datos.
db_password	Privada	String	Contraseña para la base de datos.
db_server	Privada	String	Dirección IP del servidor de base de datos.

Fuente: Elaboración propia.

Tabla 47. Métodos Clase *Configuration*.

Método	Parámetro(s) entrada	Descripción
get_locale	-	Método que devuelve el valor del atributo *locale*.
set_locale	String (*locale*)	Método que establece el atributo *locale*. Sólo son válidos los valores 'es', 'en'.
get_quiet	-	Método que devuelve el valor del atributo *quiet*.
set_quiet	Boolean (*quiet*)	Método que establece el atributo *quiet*. Sólo son válidos valores *True/False*.
get_start_urls	-	Método que devuelve el valor del atributo *start_urls*.
set_start_urls	Array de strings (*start_urls*)	Método que establece el atributo *start_urls*.
get_max_depth_crawling	-	Método que devuelve el valor del atributo *max_depth_crawling*.
set_max_depth_crawling	Integer (*max_depth_crawling*)	Método que establece el atributo *max_depth_crawling*. Sólo son válidos valores enteros ≥ 0.
get_max_crawled_web_pages	-	Método que devuelve el valor del atributo *max_crawled_web_pages*.
set_max_crawled_web_pages	Integer (*max_crawled_web_pages*)	Método que establece el atributo *max_crawled_web_pages*. Sólo son válidos valores enteros ≥ 0.

Método	Parámetro(s) entrada	Descripción
get_crawling_strategy	-	Método que devuelve el valor del atributo *crawling_strategy*.
set_crawling_strategy	String (*crawling_strategy*)	Método que establece el atributo *crawling_strategy*. Sólo son válidos los valores 'bfs', 'dfs' 'best-fs'.
get_keywords	-	Método que devuelve el valor del atributo *keywords*.
set_keywords	Array de strings (*keywords*)	Método que establece el atributo *keywords*.
get_min_image_width	-	Método que devuelve el valor del atributo *min_image_width*.
set_min_image_width	Integer (*min_image_width*)	Método que establece el atributo *min_image_width*. Sólo son válidos valores enteros ≥ 0.
get_min_image_height	-	Método que devuelve el valor del atributo *min_image_height*.
set_min_image_height	Integer (*min_image_ heigth*)	Método que establece el atributo *min_image_height*. Sólo son válidos valores enteros ≥ 0.
get_db_user	-	Método que devuelve el valor del atributo *db_user*.
set_db_user	String (*db_user*)	Método que establece el atributo *db_user*.
get_db_password	-	Método que devuelve el valor del atributo *db_password*.
set_db_password	String (*db_password*)	Método que establece el atributo *db_password*.
get_db_server	-	Método que devuelve el valor del atributo *db_server*.
set_db_server	String (*db_server*)	Método que establece el atributo *db_server*.

Método	Parámetro(s) entrada	Descripción
translate	String (*text*)	Método que traduce un texto pasado como parámetro de entrada (*text*) al idioma correspondiente al atributo *locale*. Realmente contiene un enlace a un método *gettext* de un objeto tipo *translate* (atributo *trans*) que realiza la traducción.
ntranslate	String (*text1*) String (*text2*) Integer (*num*)	Método que traduce un texto con un campo numérico diferenciando entre valores singulares y plurales. Permite traducir por ejemplo '1 día' y '2 días'. Realmente contiene un enlace a un método *ngettext* de un objeto tipo *translate* (atributo *trans*) que realiza la traducción.

Fuente: Elaboración propia.

Page

Tabla 48. Atributos Clase *Page*.

Atributo	Visibilidad	Tipo	Descripción
crawling_date_time	Privada	DateTime	Fecha/hora en la que se realizó el rastreo.
images	Privada	Array de objetos tipo *Image*	Array de objetos de tipo *Image* conteniendo información relativa a las imágenes de la página web cuya información debe ser almacenada[31].
url	Privada	String	URL de la página web rastreada.
network	Privada	String	Tipo de red correspondiente al atributo *url*: 'tor', 'i2p', 'freenet', 'visible'.
web_html_content	Privada	String	Contenido HTML de la web rastreada.

[31] Sólo se almacenan aquellas imágenes cuyo tamaño supere un tamaño mínimo configurable por el usuario.

Atributo	Visibilidad	Tipo	Descripción
conf	Privada	Objeto tipo *Configuration*	Objeto de tipo *Configuration* para poder acceder en la clase a datos relacionados con la configuración de la aplicación.

Fuente: Elaboración propia.

Tabla 49. Métodos Clase *Page*.

Método	Parámetro(s) entrada	Descripción
get_crawling_date_time	-	Método que devuelve el valor del atributo *crawling_date_time*.
set_crawling_date_time	DateTime (*crawling_date_time*)	Método que establece el atributo *crawling_date_time*.
get_images	-	Método que devuelve el valor del atributo *images*.
append_image	Objeto tipo Image (*image*)	Método que añade al array de objetos tipo *Image* un nuevo elemento conteniendo información relativa a una imagen contenida en la página web rastreada.
get_url	-	Método que devuelve el valor del atributo *url*.
set_url	String (*url*)	Método que establece el atributo *url*. También establece el valor del atributo *network* en función de dicho atributo *url*.
get_network	-	Método que devuelve el valor del atributo *network*.
get_web_html_content	-	Método que devuelve el valor del atributo *web_html_content*.
set_web_html_content	String (*web_html_content*)	Método que establece el atributo *web_html_content*.

Fuente: Elaboración propia.

Image

Tabla 50. Atributos Clase *Image*.

Atributo	Visibilidad	Tipo	Descripción
filename	Privada	String	Nombre del fichero correspondiente a la imagen.
md5_hash	Privada	String	Hash MD5 de la imagen.
sha1_hash	Privada	String	Hash SHA1 de la imagen.
diff_hash	Privada	String	*Difference hash* de la imagen.

Atributo	Visibilidad	Tipo	Descripción
exif_metadata	Privada	String	String conteniendo los metadatos de la imagen en formato 'clave': valor.
conf	Privada	Objeto tipo *Configuration*	Objeto de tipo *Configuration* para poder acceder en la clase a datos relacionados con la configuración de la aplicación.

Fuente: Elaboración propia.

Tabla 51. Métodos Clase *Image*.

Método	Parámetro(s) entrada	Descripción
get_filename	-	Método que devuelve el valor del atributo *filename*.
set_filename	String (*filename*)	Método que establece el atributo *filename*.
get_md5_hash	-	Método que devuelve el valor del atributo *md5_hash*.
set_md5_hash	String (*md5_hash*)	Método que establece el valor del atributo *md5_hash*. Debe ser un string de 16 caracteres de longitud.
get_sha1_hash	-	Método que devuelve el valor del atributo *sha1_hash*.
set_sha1_hash	String (*sha1_hash*)	Método que establece el valor del atributo *sha1_hash*. Debe ser un string de 20 caracteres de longitud.
get_diff_hash	-	Método que devuelve el valor del atributo *diff_hash*.
set_diff_hash	String (*diff_hash*)	Método que establece el valor del atributo *diff_hash*.
get_exif_metadata	-	Método que devuelve el valor del atributo *exif_metadata*.
set_exif_metadata	Diccionario (*exif_metadata*)	Método que establece el valor del atributo *exif_metadata*. Para facilitar la búsqueda posterior, el parámetro de entrada *exif_metadata* se transforma de diccionario a string.

Fuente: Elaboración propia.

Anexo E. Plan de pruebas

Se incluye en el presente anexo el plan de pruebas realizado, en el que se verifican todos los aspectos contemplados en el "Anexo A. Requisitos de la aplicación". Según lo indicado en el capítulo "4.4 Pruebas" para realizar la verificación se han realizado un conjunto de pruebas automatizadas, por medio del fichero *test-darfinder.py*, cuyo contenido se muestra en el "Anexo I. Código fuente".

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
Generales		º			
GEN-01	1	Configuración de parámetros mediante un fichero de texto denominado *darkfinder.conf*.	Automática	✔	El script test-darkfinder.py (función *test_configuration*) incluye la lectura de todos los ficheros de configuración contemplados.
GEN-02	1	Soporte multi idioma (español / inglés).	Manual	✔	Modificación manual de la clave '*locale*' en el fichero *darkfinder.conf*. Verificación manual de los resultados. Mostrado en "Anexo H. Resultados".
GEN-03	1	Comentarios en inglés y español. Seguimiento de las pautas de codificación de código en Python de la guía PEP-8.	Manual	✔	Verificación manual mediante aplicación Spyder versión 5.1.5.
GEN-04	1	Mostrar por pantalla y en fichero log los mensajes relativos a la marcha del proceso de rastreo: Comienzo, fin, duración del proceso de rastreo y número de páginas rastreadas.	Manual	✔	Verificación en pantalla y fichero log de los mensajes. Mostrado en "Anexo H. Resultados".

[32] Manual / automática

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
GEN-05	2	Funcionamiento en sistemas operativos Microsoft Windows 10 y Kali Linux.	Manual	✔	Verificación manual de la ejecución de los programas *darkfinder* y *darksearch* en sistemas operativos Windows y Kali Linux. Mostrado en “Anexo H. Resultados”.
GEN-06	2	La complejidad cognitiva del conjunto de métodos y funciones de la aplicación no deberá ser superior a 15.	Automática	✔	Verificación automática mediante el análisis de código estático realizado[33].
GEN-07	3	Eliminar automáticamente páginas web rastreadas con una antigüedad superior a un parámetro configurable.	-	-	Funcionalidad no implantada.
Crawler					
CRA-01	1	Recorrer de manera recursiva un conjunto de páginas web a partir de una dirección semilla.	Manual / Automática	✔	Verificación manual del rastreo de páginas web, tanto de la semilla incluida en el fichero *darkfinder.conf* como de los enlaces incluidos en las sucesivas web rastreadas. El script *test-darkfinder.py* (función *test_darkfinder*) incluye el rastreo de diversas URLs a partir de una semilla. Mostrado en “Anexo H. Resultados”.

[33] En caso de que no se cumpla la condición la herramienta Sonarqube lo etiqueta como ‘*Code Smell*’ (ver “Anexo F. Análisis estático de código fuente”).

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
CRA-02	1	Configuración tanto de la dirección semilla de partida como las condiciones para la finalización del rastreo. Incluir más de una dirección semilla, pudiendo pertenecer a redes independientes	Manual	✔	Verificación manual del funcionamiento del rastreo introduciendo múltiples direcciones (de redes diferentes) en la clave *start_url* y número máximo de páginas rastreadas en la clave *max_crawled_web_pages* en el fichero *darkfinder.conf.* Mostrado en "Anexo H. Resultados".
CRA-03	1	Analizar contenido de las siguientes redes: web visible, TOR, I2P, Freenet.	Manual / Automática	✔	Verificación manual del funcionamiento del rastreo en cada una de las redes, modificando la clave *start_url* del fichero *darkfinder.conf.* El script *test-darkfinder.py* (función *test_crawl*) incluye el rastreo en las redes visible, TOR, I2P y Freenet. Mostrado en "Anexo H. Resultados".
CRA-04	1	Alternar enlaces entre las diferentes redes, permitiendo seguir un enlace de una página en una de las redes a otra página de una red diferente.	Manual	✔	Verificación manual del funcionamiento del rastreo secuencial de diferentes redes, modificando la clave *start_url* del fichero *darkfinder.conf.* Mostrado en "Anexo H. Resultados".
CRA-05	1	Implantar las estrategias de *crawling Breadth-First Search* (BFS) y *Depth-First Search* (DFS).	Manual	✔	Verificación manual del funcionamiento de ambos algoritmos de rastreo, modificando la clave *crawling_strategy* del fichero *darkfinder.conf.* Mostrado en "Anexo H. Resultados".
CRA-06	1	Evitar el rastreo de páginas que hayan sido rastreadas con anterioridad, estando almacenadas en la base de datos.	Manual / Automática	✔	Verificación manual repitiendo ejecución del *crawler* y comprobando que no se realiza el rastreo de páginas almacenadas previamente. El script *test-darkfinder.py* (función *test_database_manager*) incluye la comprobación de rastreo previo de páginas web (función *url_crawled*).

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
CRA-07	2	Realizar un cálculo de la frecuencia de aparición de un conjunto de palabras clave configurable.	Manual	✔	Verificación manual de que la función *webpage_link_priority* del módulo Crawler realiza un cálculo de la frecuencia de aparición de los literales especificados en la clave *keywords* del fichero *darkfinder.conf.* Verificación manual de la aparición de los literales especificados en la clave *keywords* del fichero *darkfinder.conf* usando diferentes estrategias de rastreo. Mostrado en “Anexo H. Resultados”.
CRA-08	2	Implantar una estrategia de *crawling* de tipo *Best-First Search* (BFS), proporcionando un ranking a las páginas localizadas en base a un algoritmo definido que tenga en consideración un conjunto de palabras clave predefinidas por el usuario.	Manual	✔	Verificación manual del funcionamiento del algoritmo de rastreo Best-First Search (BFS), modificando las claves *crawling_strategy* y *keywords* del fichero *darkfinder.conf.* Mostrado en “Anexo H. Resultados”.
CRA-09	3	Eludir medidas de bloqueo de IP realizadas por servidores en la red TOR por medio de la rotación de la dirección IP en esta red.	-	-	Funcionalidad no implantada.
CRA-10	3	Permitir el rastreo de formularios de la Deep Web mediante el relleno automático de campos de texto de formularios.	-	-	Funcionalidad no implantada.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
CRA-11	3	Permitir acceder a zonas de acceso restringido mediante usuario-contraseña en base a un listado de usuarios y contraseñas incluidos en ficheros de texto.	-	-	Funcionalidad no implantada.
CRA-12	3	Permitir resolver de manera automática patrones CAPTCHA de texto.	-	-	Funcionalidad no implantada.
CRA-13	3	Incorporar nuevas Dark Nets para el rastreo de patrones delictivos.	-	-	Funcionalidad no implantada.
Imágenes					
IMA-01	1	Obtener el hash MD5 y SHA1 relativos a las imágenes existentes en todas las páginas visitadas.	Manual / Automática	✔	Verificación manual del cálculo de los hashes MD5/SHA1 de las imágenes de las páginas rastreadas. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación de los hashes MD5 y SHA1 de imagen rastreada. Mostrado en “Anexo H. Resultados”.
IMA-02	1	Obtener metadatos EXIF relativos a las imágenes existentes en todas las páginas visitadas.	Manual / Automática	✔	Verificación manual de la obtención de los metadatos de las imágenes de las páginas rastreadas. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación de los metadatos EXIF de imagen rastreada. Mostrado en “Anexo H. Resultados”.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
IMA-03	1	Excluir del análisis aquellas imágenes cuyo tamaño (ancho, alto) sea inferior a un valor configurable.	Manual / Automática	✔	Verificación manual de que no se procesan imágenes cuyas dimensiones sean inferiores a lo determinado en las claves *min_image_width* y *min_image_height* del fichero *darkfinder.conf.* El script *test-darkfinder.py* (función *test_crawl*) comprueba que no se almacena imagen de tamaño inferior al configurado en una web de la red TOR.
IMA-04	1	No almacenar en el disco duro las imágenes analizadas.	Manual	✔	Verificación manual de que no se almacena ninguna imagen en disco duro tras el proceso de rastreo.
IMA-05	2	Obtener el *difference hash* relativo a las imágenes existentes en todas las páginas visitadas.	Manual / Automática	✔	Verificación manual del cálculo del *difference hash* de las imágenes de las páginas rastreadas. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación del *difference hash* de imagen rastreada. Mostrado en “Anexo H. Resultados”.
Bases de datos					
BD-01	1	Almacenar la información en una base de datos no estructurada MongoDB.	Manual / Automática	✔	Verificación manual del almacenamiento de la información captada en base de datos MongoDB. El script *test-darkfinder.py* (funciones *test_crawl, test_database_manager*) incluye la verificación del almacenamiento en base de datos MongoDB. Mostrado en “Anexo H. Resultados”.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
BD-02	1	Almacenar la siguiente información relativa a una página web: • URL. • Red (visible, TOR, I2P, Freenet). • Fecha/Hora de rastreo. • Contenido HTML de la página web. • Información relativa a imágenes existentes en la página web (ver BD-03).	Manual / Automática	✔	Verificación manual de la obtención de la información especificada de las páginas web rastreadas. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación del almacenamiento en base de datos MongoDB de los parámetros especificados. Mostrado en “Anexo H. Resultados”.
BD-03	1	Almacenar la siguiente información relativa a una imagen: • Nombre del fichero. • Hashes MD5 y SHA-1. • Metadatos EXIF.	Manual / Automática	✔	Verificación manual de la obtención de la información especificada de las imágenes de las páginas web rastreadas. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación del almacenamiento en base de datos MongoDB de los parámetros especificados relativos a las imágenes. Mostrado en “Anexo H. Resultados”.
BD-04	1	Realizar la conexión a base de datos MongoDB mediante API específica estándar de conexión para minimizar los fallos de conexión.	Manual	✔	Verificación manual en el código fuente del uso de la librería *pymongo* para interacciones con la base de datos MongoDB.
BD-05	1	Almacenar la información correspondiente a una página web y las imágenes asociadas de manera atómica.	Manual	✔	Verificación manual en el código fuente del establecimiento de relación tipo *Embedded* entre las entidades *Page* e *Image* en base de datos MongoDB.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
BD-06	2	Almacenar la siguiente información relativa a una imagen: *Difference Hash*.	Manual / Automática	✔	Verificación manual del almacenamiento del *difference hash* de las imágenes de las páginas web rastreadas. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación del almacenamiento en base de datos MongoDB del *difference hash* de imagen rastreada. Mostrado en “Anexo H. Resultados”.
BD-07	2	Almacenar la información en un sistema Elasticsearch que permita realizar búsquedas avanzadas por el usuario.	Manual	✔	Carga manual de datos rastreados en la aplicación Elasticsearch y verificación de la realización de consultas mediante Kibana. Mostrado en “Anexo H. Resultados”.
BD-08	3	Cargar de manera automática la información rastreada en un sistema Elasticsearch mediante Logstash o mecanismo equivalente.	-	-	Funcionalidad no implantada.
BD-09	3	Incorporar nuevos sistemas gestores de bases de datos para almacenar información: MySQL.	-	-	Funcionalidad no implantada.
Búsqueda					
BUS-01	1	Incorporar un módulo de búsqueda de patrones delictivos en el contenido de la página web, incluyendo las siguientes posibilidades: • Campo de texto. • Expresión regular (Regex).	Manual / Automática	✔	Verificación manual de la búsqueda por campo de texto y/o expresiones regulares. El script *test-darkfinder.py* (función *test_darksearch*) incluye la verificación de la búsqueda de páginas web almacenadas. Mostrado en “Anexo H. Resultados”.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
BUS-02	1	Incorporar un módulo de búsqueda de patrones delictivos en el contenido de una imagen, incluyendo las siguientes posibilidades: • Hash MD5 / SHA1. • Expresión regular (Regex) asociada a metadatos.	Manual / Automática	✔	Verificación manual de la búsqueda de imágenes mediante hash MD5/SHA1 y/o expresiones regulares asociadas a metadatos. El script *test-darkfinder.py* (función *test_darksearch*) incluye la verificación de la búsqueda de imágenes mediante hash MD5/SHA1 y expresión regular asociada a metadatos. Mostrado en “Anexo H. Resultados”.
BUS-03	1	Mostrar la siguiente información en caso de encontrar coincidencia: • URL. • Red (visible, TOR, I2P, Freenet). • Fecha de rastreo. En el caso de que se realice una búsqueda de imágenes, mostrar los siguientes campos: • Nombre del fichero imagen.	Manual	✔	Verificación manual de la impresión de URL, red, fecha de rastreo y en su caso nombre del fichero imagen en la realización de búsquedas. Mostrado en “Anexo H. Resultados”.
BUS-04	2	Incorporar un módulo de búsqueda de patrones delictivos en el contenido de una imagen, incluyendo: • *Difference Hash*, especificando tanto el hash referencia como la distancia (distancia Hamming). Además de la información mostrada en BUS-03, mostrar: • Distancia Hamming entre la imagen y el hash de referencia.	Manual / Automática	✔	Verificación manual de la búsqueda de imágenes mediante establecimiento de *difference hash* de referencia y distancia de Hamming. Verificación de que la información mostrada en la búsqueda de imagen incluye la distancia de Hamming entre la imagen y el hash de referencia. El script *test-darkfinder.py* (función *test_darksearch*) incluye la verificación de la búsqueda de imágenes mediante *difference hash*. Mostrado en “Anexo H. Resultados”.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
BUS-05	3	Permitir búsquedas relativas a las coordenadas geográficas reflejadas en los metadatos de las imágenes obtenidas.	-	-	Funcionalidad no implantada.
Seguridad					
SEG-01	1	Controlar los errores mediante bloques *try-except* en los accesos a: • Recepción de argumentos por línea de comandos. • Red. • Disco duro. • Base de datos.	Manual / Automática	✔	Verificación manual en el código fuente del uso bloques try-except. La librería *scrapy* incorpora gestión de errores en el acceso a webs. El script *test-darkfinder.py* incluye la verificación de diversas situaciones de error, comprobando que el valor devuelto por las funciones es 'False'. Mostrado en Figura 40.
SEG-02	1	Almacenar log con el siguiente contenido: fecha/hora, nivel del mensaje (*Debug, Info, Warning, Error, Critical*), usuario que ejecuta la aplicación, mensaje, módulo en el que se produce el log, función en la que se produce el log. Registrar logs en las siguientes situaciones: errores, eventos informando de actuaciones del usuario y del software.	Manual	✔	Verificación manual de que el log almacenado comprende los campos especificados. Mostrado en Figura 39.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones

SEG-03	1	Sanitizar el contenido de los parámetros del fichero de configuración.
		Realizar las comprobaciones de todos los parámetros de configuración del fichero *darkfinder.conf,* incluyendo:
		<u>Cadenas de caracteres</u>
		• Verificación de valores permitidos (lista blanca).
		• Verificación de:
		  ○ Tipo (string).
		  ○ Longitud mínima y máxima.
		  ○ Filtrado de metacaracteres (“/”, “<”, “>”, etc.).
		<u>Numéricos</u>
		• Verificación de tipo (int / float).
		• Valor mínimo y máximo.
			Manual / Automática	✔	Verificación manual de la adecuada carga en un objeto tipo *Configuration* de los parámetros cargados en fichero *darkfinder.conf*. Modificación manual de los parámetros, con valores válidos y maliciosos/erróneos.
					El script *test-darkfinder.py* (función *test_configuration*) incluye la verificación de la carga adecuada en un objeto *Configuration* de valores correctos y maliciosos/erróneos.

SEG-04	1	Acceso a la base de datos protegido mediante usuario y contraseña.	Manual / Automática	✔	Verificación manual de que la base de datos MongoDB no permite la conexión con usuario y/o contraseña erróneos. El script *test-darkfinder.py* (función *test_database_manager*) incluye la verificación de que se produce un error al introducir usuario/contraseña erróneos.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
SEG-05	1	Sanitizar la información de las webs rastreadas y que será almacenada en la base de datos MongoDB, incluyendo el contenido HTML de las páginas web almacenadas para evitar ataque de tipo Cross-Site-Scripting (XSS).	Manual / Automática	✔	Verificación manual de que la información de las páginas almacenadas en la base de datos MongoDB ha sido sanitizada, no incluyendo metadatos como '<', '>', etc. El script *test-darkfinder.py* (función *test_crawl*) incluye la verificación de que en caso de que el texto HTML tenga metacaracteres (ej. '<script>') se procede a su sanitización.
SEG-06	1	Sanitizar el contenido de entradas introducidas por el usuario por línea de comandos, incluyendo aquellas que vayan a ser empleadas en consultas a bases de datos, para evitar inyecciones NoSQL.	Manual / Automática	✔	Verificación manual de sanitización de argumentos de entrada por línea de comandos. Verificación de que al realizar intentos de inyección NoSQL, introduciendo valores del tipo "*valor', otro_campo=otro_valor*", el filtro aplicado a la consulta MongoDB sólo se aplica al campo original, con lo que no se produce error ni resultado no previsto. El script *test-darkfinder.py* (función *test_database_manager*) incluye la verificación de intento de inyección NoSQL.
SEG-07	1	Realización de pruebas automáticas, con una cobertura de código superior a 90%, incluyendo: • Valores esperados (casos de uso). • Valores no esperados (casos de abuso).	Manual / Automática	✔	Verificación de la cobertura de pruebas automatizadas mediante aplicación Sonarqube. Realización manual de pruebas de valores esperados (casos de uso) y no esperados (casos de abuso, recogidos en el presente bloque de pruebas). El script *test-darkfinder.py* incluye la realización de pruebas de casos de uso y abuso. Mostrado en Tabla 26.

Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
SEG-08	1	Tratar los atributos de clases como privadas, proporcionando métodos para su lectura / escritura: • *get*: lectura del atributo. • *set*: escritura del atributo, realizando las comprobaciones de su validez, con las consideraciones establecidas en SEG-03.	Manual / Automática	✔	Verificación manual en el código fuente del manejo de atributos de clases como privadas, proporcionando métodos *get/set* para lectura/escritura. Verificación manual de comprobación de tipo de variables en métodos *set* previa al establecimiento del atributo. El script *test-darkfinder.py* (funciones *test_configuration* y *test_crawl)* incluye la inyección de parámetros maliciosos en métodos *set*.
SEG-09	1	Instalación y ejecución de la aplicación en entorno virtualizado.	Manual	✔	Verificación manual de que el software se ejecuta en entorno virtualizado Microsoft Hyper-V.
SEG-10	1	Instalación de la aplicación en un sistema operativo con gestión de usuarios. Ejecución con un usuario SIN privilegios (NO root / administrador).	Manual	✔	Verificación manual de que el software se ejecuta en sistemas operativos con gestión de usuarios: • Microsoft Windows 10. • Kali Linux. Verificación de ejecución mediante usuario sin privilegios[34].
SEG-11	1	Contemplar copia de seguridad de código fuente, fichero de configuración y datos de la base de datos.	Manual	✔	Verificación manual la realización de copia de seguridad de los elementos identificados.

[34] Algunos servicios deben ejecutarse con permisos de administrador.

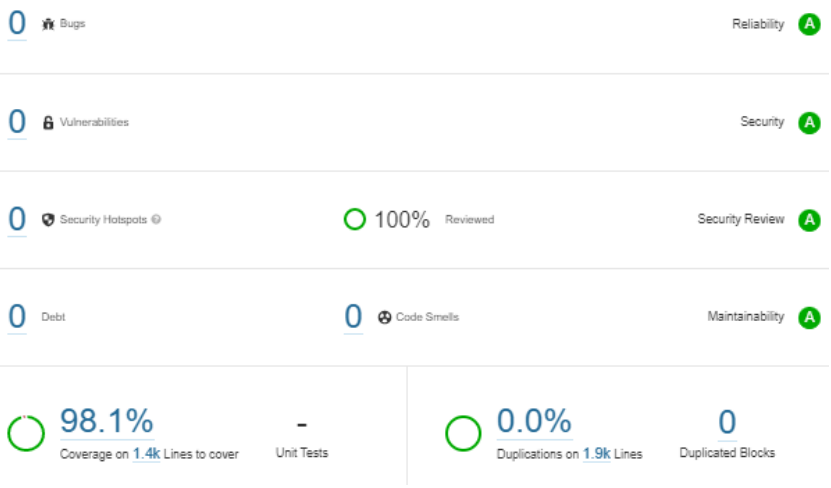
Id	Fase	Requisito	Tipo prueba [32]	Resultado	Observaciones
SEG-12	1	Configuración segura de sistemas operativos, sistema de virtualización y base de datos en base a las guías especificadas.	Manual	✔	Verificación manual de la configuración del sistema operativo conforme a las reglas de bastionado especificadas.
SEG-13	1	Análisis estático de código fuente.	Automática	✔	Realización de un análisis estático de código fuente mediante aplicación Sonarqube. Mostrado en Tabla 26 y “Anexo F. Análisis estático de código fuente”.
SEG-14	1	Pasar un análisis de librerías importadas vulnerables.	Automática	✔	Verificación de la realización de un análisis de librerías importadas mediante aplicación dependency-check. Mostrado en Figura 48.
SEG-15	1	Instalación de software antimalware en cliente / servidor.	Manual	✔	Verificación manual de la instalación de software antimalware en los equipos.
SEG-16	1	No codificar contraseñas en el código fuente (*hardcoded password*).	Manual	✔	Verificación manual en el código fuente de la ausencia de codificación de contraseñas.
SEG-17	1	Realizar la lectura / escritura en el sistema de ficheros mediante librería estándar.	Manual	✔	Verificación manual en el código fuente comprobando que la lectura / escritura de ficheros se realiza mediante librería estándar (módulos Configuración y Log).

Fuente: Elaboración propia.

Anexo F. Análisis estático de código fuente

En el presente anexo se muestra el resultado del análisis estático de código fuente realizado con la herramienta Sonarqube.

Figura 55. Resumen del análisis estático de código fuente.



Fuente: Elaboración propia.

Figura 56. Resultados del análisis estático de código fuente en los scripts principales de la aplicación.

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
darkfinder							
modules	900	0	0	0	0	98.2%	0.0%
darkfinder.py	108	0	0	0	0	90.1%	0.0%
darksearch.py	199	0	0	0	0	94.8%	0.0%
test-darkfinder.py	700	0	0	0	0	100%	0.0%

Fuente: Elaboración propia.

Figura 57. Resultados del análisis estático de código fuente en módulos de la aplicación.

	Lines of Code	Bugs	Vulnerabilities	Code Smells	Security Hotspots	Coverage	Duplications
modules	900	0	0	0	0	98.2%	0.0%
__init__.py	0	0	0	0	0	—	0.0%
auxiliary.py	27	0	0	0	0	100%	0.0%
configuration.py	233	0	0	0	0	100%	0.0%
crawler.py	384	0	0	0	0	95.9%	0.0%
database_manager.py	142	0	0	0	0	100%	0.0%
logger.py	77	0	0	0	0	100%	0.0%
settings.py	37	0	0	0	0	100%	0.0%

Fuente: Elaboración propia.

Anexo G. Manual de instalación de la aplicación

En el presente anexo se describen los pasos para el funcionamiento de la aplicación DarkFinder en los sistemas operativos Windows y Kali Linux, incluyendo la instalación de las siguientes herramientas:

- Intérprete Python.
- Servicio de conexión a red TOR.
- Servicio de conexión a red I2P.
- Servicio de conexión a red Freenet.
- Base de datos MongoDB.
- Elasticsearch. Si bien no es necesario para el funcionamiento de la aplicación DarkFinder, puede constituir un complemento para la realización de análisis y búsquedas.

Windows

Se describen a continuación los pasos para instalar la herramienta DarkFinder en sistema operativo Windows.

Python

Los pasos a seguir para instalar el intérprete Python en un sistema operativo Windows son los siguientes:

1. Descargar el asistente de instalación para la versión instalable Python (v3.10.2) de la página web de descarga de Python.
2. Ejecutar el asistente de instalación de Python.
3. Ejecutar el siguiente comando en la ruta en la que se encuentra el código con permisos de administrador para la instalación de las librerías externas de Python requeridas:

pip install -r requirements.txt

4. Ejecutar los comandos *darkfinder* y *darksearch* por medio de los siguientes comandos en la ruta en la que se encuentra el código:

python darkfinder.py [argumentos]

python darksearch.py [argumentos]

Siendo *[argumentos]* los argumentos especificados en los capítulos “4.2.2 DarkFinder” y “4.2.6 DarkSearch”.

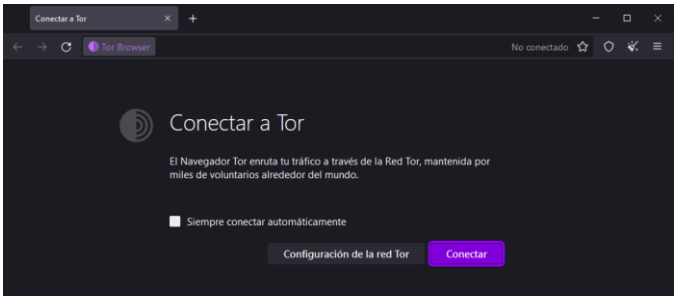
Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 4.

TOR

Los pasos a seguir para instalar el servicio de conexión a esta Dark Net en un sistema operativo Windows son:

1. Descargar el asistente de instalación Tor Browser (v11.0.3) de la página web de descarga de TOR Project y proceder a su instalación.
2. Abrir la aplicación "*Start TOR Browser*" y pulsar en el botón "*Conectar*" según lo mostrado en la Figura 58.

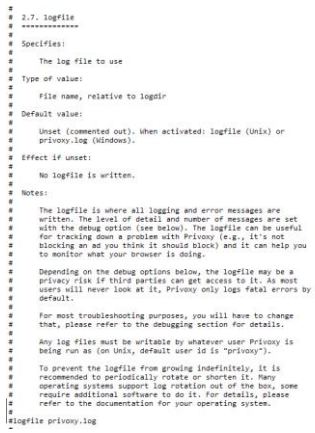
Figura 58. Conexión a red TOR.



Fuente: Elaboración propia.

3. Descargar el asistente de instalación de Privoxy (v3.0.33) de la página web de descarga de Privoxy y proceder a su instalación.
4. Actualizar el fichero de configuración de *Privoxy* (C:\Program Files (x86)\Privoxy\config.txt) en el apartado "2.4 logdir" de modo que apunte a un directorio en el que el usuario tenga permisos de lectura/escritura o bien comentar la referencia al fichero de log en el apartado "2.7 logfile", del modo indicado en la Figura 59.

Figura 59. Configuración Privoxy (config.txt). Logfile.



```
#
#  2.7. logfile
#  ============
#
#  Specifies:
#
#      The log file to use
#
#  Type of value:
#
#      File name, relative to logdir
#
#  Default value:
#
#      Unset (commented out). When activated: logfile (Unix) or
#      privoxy.log (Windows).
#
#  Effect if unset:
#
#      No logfile is written.
#
#  Notes:
#
#      The logfile is where all logging and error messages are
#      written. The level of detail and number of messages are set
#      with the debug option (see below). The logfile can be useful
#      for tracking down a problem with Privoxy (e.g., it's not
#      blocking an ad you think it should block) and it can help you
#      to monitor what your browser is doing.
#
#      Depending on the debug options below, the logfile may be a
#      privacy risk if third parties can get access to it. As most
#      users will never look at it, Privoxy only logs fatal errors by
#      default.
#
#      For most troubleshooting purposes, you will have to change
#      that, please refer to the debugging section for details.
#
#      Any log files must be writable by whatever user Privoxy is
#      being run as (on Unix, default user id is "privoxy").
#
#      To prevent the logfile from growing indefinitely, it is
#      recommended to periodically rotate or shorten it. Many
#      operating systems support log rotation out of the box, some
#      require additional software to do it. For details, please
#      refer to the documentation for your operating system.
#
#logfile privoxy.log
```

Fuente: Elaboración propia.

5. Actualizar el fichero de configuración de *Privoxy* (C:\Program Files (x86)\Privoxy\config.txt) añadiendo la información indicada a continuación.

forward-socks5t / 127.0.0.1:9150 .

keep-alive-timeout 600

default-server-timeout 600

socket-timeout 600

6. Ejecutar la aplicación Privoxy.

Para ejecutar la aplicación al iniciar el ordenador, deberán realizarse los pasos 2 y 6.

I2P

Los pasos a seguir para instalar el servicio de conexión a I2P en un sistema operativo Windows son:

1. Descargar el asistente de instalación I2P (v1.8.0) de la página web de descarga de Invisible Internet Project (I2P) y proceder a su instalación.
2. Ejecutar la aplicación "*Start I2P (restartable)*" por medio de la barra de búsqueda de Windows.
3. Realizar la configuración del servicio de navegación I2P. Tras la ejecución se mostrará pantalla relativa al router I2P.

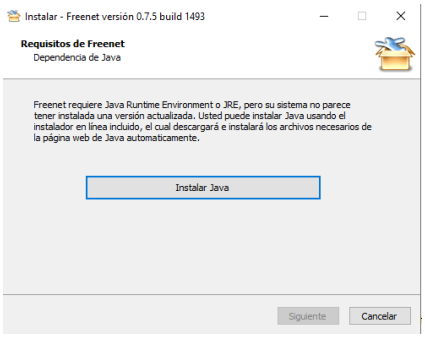
Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 2.

Freenet

Los pasos a seguir para instalar el servicio de conexión a esta Dark Net en un sistema operativo Windows son:

1. Descargar el asistente de instalación Freenet (v0.7.5) de la página web de descarga de The Freenet Project Inc y proceder a su instalación. Si se solicita, seleccionar la opción de instalar Java, del modo mostrado en la Figura 60.

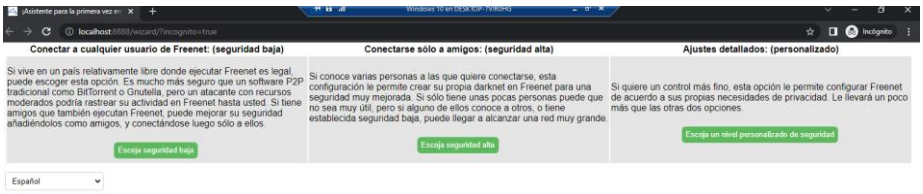
Figura 60. Asistente de instalación de Freenet. Instalación de Java.



Fuente: Elaboración propia.

2. Ejecutar la aplicación “Freenet” por medio de la barra de búsqueda de Windows.
3. Aparecerá el asistente de configuración en la primera conexión, mostrado en la Figura 61. Se podrá seleccionar la opción ‘*Conectar a cualquier usuario de Freenet (seguridad baja)*’. Escoger las diferentes opciones de configuración. Como resultado, se mostrará un navegador con acceso a esta Dark Net.

Figura 61. Ejecución de Freenet. Asistente primera conexión.



Fuente: Elaboración propia.

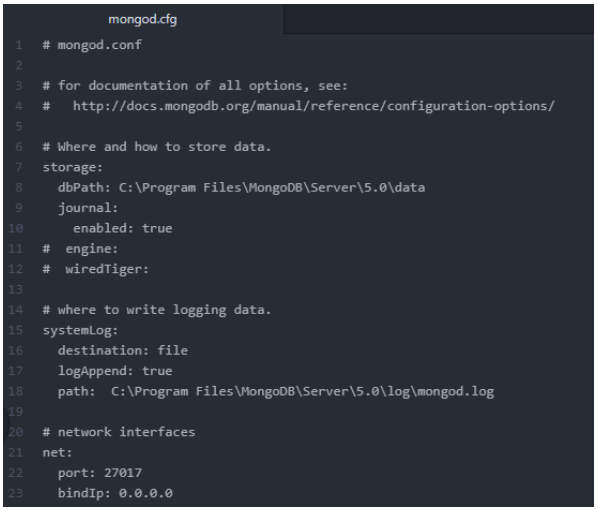
Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 2.

MongoDB

Los pasos a seguir para instalar esta base de datos no relacional en un sistema operativo Windows son:

1. Descargar el asistente de instalación MSI la versión instalable MongoDB Community Server (v5.0.8) de la página web de descarga de MongoDB y proceder a su instalación.
2. Instalar MongoDB Shell (versión 1.3.1), shell para interactuar con la base de datos, seleccionando el asistente de instalación ZIP descargable desde la página web de descarga de MongoDB Shell. Descomprimir el fichero ZIP.
3. Permitir conexiones remotas modificando el fichero “C:\Program Files\MongoDB\Server\5.0\bin\mongod.cfg”, reemplazando la línea bind_ip = 127.0.0.1 por bind_ip = 0.0.0.0, del modo mostrado en la Figura 62.

Figura 62. Fichero mongod.cfg.



Fuente: Elaboración propia.

4. Opcionalmente, instalar el GUI de MongoDB Compass (versión 1.31.2) seleccionando el asistente de instalación MSI descargable desde la página web de descarga de MongoDB Compass. Seguir el asistente de instalación del GUI MongoDB Compass.
5. Crear en el servidor la carpeta donde se van a alojar los datos de la base de datos (ej. "c:\data\db").
6. Arrancar el servicio de base de datos ejecutando el siguiente comando, donde la opción *--dbpath* apunta al directorio en el que se van a almacenar los datos.

"C:\Program Files\MongoDB\Server\5.0\bin\mongod.exe" --dbpath="c:\data\db"

Nota: la carpeta donde se alojen los datos debe estar previamente creada.

7. Ejecutar el comando *mongosh* en la carpeta donde se haya descomprimido el fichero zip y ejecutar el siguiente comando.

"C:\Program Files\MongoDB\mongosh-1.3.1-win32-x64\bin\mongosh"

8. Cambiar a la base de datos *admin*:

use admin

9. Crear un usuario administrador con el siguiente comando e introducir manualmente la contraseña:

```
db.createUser(
  {
    user: "admin",
    pwd: passwordPrompt(), // or cleartext password
    roles: [
      { role: "userAdminAnyDatabase", db: "admin" },
      { role: "readWriteAnyDatabase", db: "admin" }
    ]
  }
)
```

10. Apagar la instancia de base de datos con el comando:

db.adminCommand({ shutdown: 1 })

11. Reiniciar la base de datos siguiendo el paso 6.

12. Revisar que en el firewall del servidor se permite conexión remota al puerto 27017.

13. Conectar a la base de datos mediante el comando:

"mongosh –port 27017 --authenticationDatabase "admin" -u "admin" -p"

14. Crear un usuario mediante el siguiente comando:

```
use admin
db.createUser(
  {
    user: "darkfinder",
    pwd:  passwordPrompt(),   // or cleartext password
    roles: [ { role: "readWrite", db: "darkfinder" },
             { role: "read", db: "darkfinder" } ]
  }
)
```

15. Mediante la aplicación *mongosh* crear una colección denominada '*pages'* en una base de datos denominada '*darkfinder*'.

use darkfinder

db.createCollection('pages')

16. Realizar el resto de acciones de bastionado contempladas en la guía especificada en el capítulo "4.5 Producción".

17. Opcionalmente, realizar la conexión mediante el GUI MongoDB Compass.

18. En el fichero de configuración *darkfinder.conf* indicar en la clave *db_server* la dirección IP del servidor (localhost en caso de que sea la propia máquina).

19. Introducir en el fichero apuntado por la clave *database_user_password_filename* del archivo de configuración de la aplicación *darkfinder.conf* el usuario y contraseña para la base de datos configurado con anterioridad.

Figura 63. Sección de base de datos MongoDB en el fichero de configuración *darkfinder.conf*.

```
######################################################################
# EN: Parameters related to database                                 #
# ES: Parámetros relacionados con la base de datos                   #
######################################################################
[database]
# EN: Name of the file containing database user and password. Absolute path is recommended
# ES: Nombre del fichero que contiene el nombre y password del usuario de la base de datos. Se recomienda ruta absoluta
database_user_password_filename = C:\Users\Sergio\source\repos\DarkFinder\mongodb_user_passwd.conf
# EN: Database server IP address
# ES: Dirección IP del servidor
db_server = localhost
```

Fuente: Elaboración propia.

Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 6.

Elasticsearch

Los pasos a seguir para instalar esta herramienta en un sistema operativo Windows son:

1. Descargar Elasticsearch (v8.2.2) de la página web de descarga de Elasticsearch.
2. Descargar Kibana (v8.2.2) de la página web de descarga de Kibana.
3. Descomprimir los ficheros zip resultantes.
4. Abrir una ventana de ejecución de comandos (*cmd*) y cambiar la ruta a la carpeta donde se ha descomprimido el contenido de Elasticsearch. Arrancar la aplicación mediante el siguiente comando:

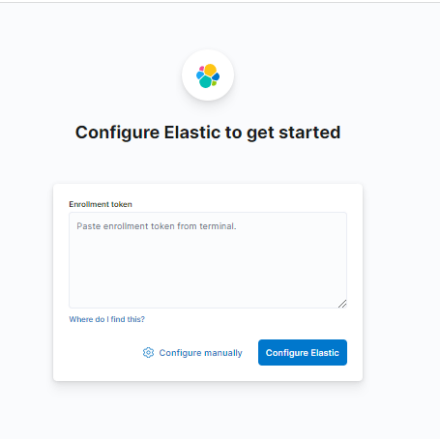
bin\elasticsearch.bat.

5. Anotar el password y el '*Enrollment token*' mostrados en pantalla para configurar Kibana en su arranque.
6. Abrir una ventana de ejecución de comandos (*cmd*) y cambiar la ruta a la carpeta donde se ha descomprimido el contenido de Kibana. Arrancar la aplicación mediante el siguiente comando:

bin\kibana.bat.

7. Hacer click en el enlace proporcionado en el terminal, o abrir en navegador web la dirección http://localhost:5601 y seguir las instrucciones para conectarse a Kibana.
8. Pegar el '*Enrollment token*' mostrado en pantalla al arrancar Elasticsearch por primera vez (referenciado en el paso 5) y pulsar en la opción '*Configure Elastic*' para la configuración automática de Elasticsearch y Kibana.

Figura 64. Configuración automática de Elasticsearch y Kibana.

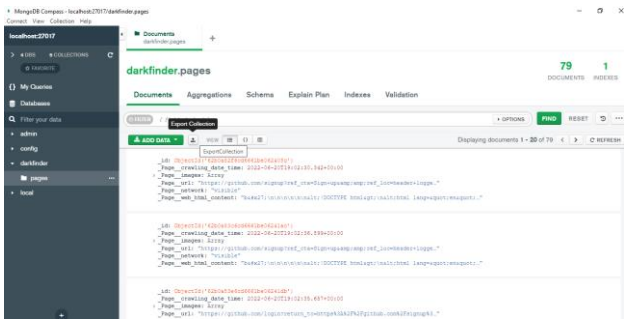


Fuente: Elaboración propia.

9. Conectarse a Kibana (http://localhost:5601/) e introducir el usuario '*elastic*' y la contraseña proporcionada al arrancar Elasticsearch por primera vez y referenciada en el paso 5.

10. Exportar mediante la aplicación MongoDB Compass (opción "*Export Collection*") en formato CSV la colección '*pages*' de la base de datos '*darkfinder*'. Seleccionar los campos a exportar.

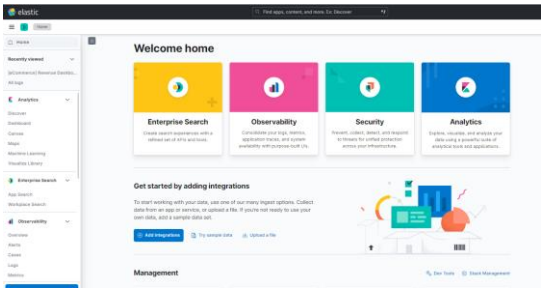
Figura 65. Exportación de colección MongoDB.



Fuente: Elaboración propia.

11. En Kibana seleccionar la opción de subir un fichero "*Upload a file*". Seleccionar el fichero CSV exportado en el paso 10.

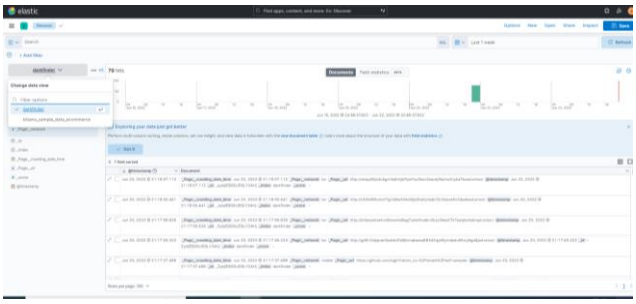
Figura 66. Importación de fichero CSV en Kibana.



Fuente: Elaboración propia.

12. En el menú situado en la parte izquierda de la pantalla seleccionar la opción "*All logs*" y seleccionar la base de datos '*darkfinder*', del modo mostrado en la Figura 67.

Figura 67. Selección de base de datos y visualización de resultados en Kibana.



Fuente: Elaboración propia.

13. Realizar las consultas y obtener los gráficos de consulta que se desee.

Para ejecutar la aplicación al iniciar el ordenador, deberán realizarse los pasos 4, 6 y 9.

Kali Linux

Se describen a continuación los pasos para instalar la herramienta DarkFinder en sistema operativo Kali Linux.

Python

El sistema operativo Kali Linux ya incorpora Python en su instalación por defecto, por lo que los únicos pasos requeridos son los siguientes:

1. Ejecutar los siguientes comandos para la actualización del sistema.

sudo apt-get update

sudo apt-get upgrade

sudo apt-get dist-upgrade

2. Ejecutar el siguiente comando en la ruta en la que se encuentra el código para instalación de librerías externas de Python requeridas por la aplicación.

pip install -r requirements.txt

3. Ejecutar los comandos *darkfinder* y *darksearch* por medio de los siguientes comandos en la ruta en la que se encuentra el código:

python darkfinder.py [argumentos]

python darksearch.py [argumentos]

Siendo *[argumentos]* los argumentos especificados en los capítulos "4.2.2 DarkFinder" y "4.2.6 DarkSearch".

Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 3.

TOR

Los pasos a seguir para instalar el servicio de conexión a esta Dark Net en un sistema operativo Kali Linux son:

1. Actualizar la base de datos de paquetes de descarga.

sudo apt update

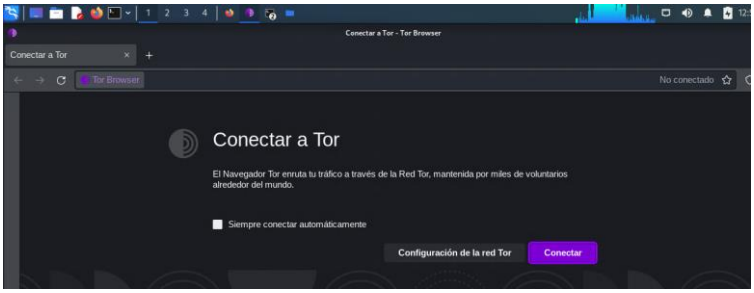
2. Instalar Tor Browser

sudo apt install -y tor torbrowser-launcher

3. Arrancar TOR Browser mediante el siguiente comando y pulsar en el botón “Conectar”, según lo mostrado en la Figura 68.

torbrowser-launcher

Figura 68. Conexión a red TOR.



Fuente: Elaboración propia.

4. Instalar Privoxy mediante el siguiente comando:

apt-get install privoxy

5. Actualizar el fichero de configuración de *Privoxy* (/etc/privoxy/config) añadiendo la información indicada a continuación al final del archivo.

forward-socks5t / 127.0.0.1:9150 .

keep-alive-timeout 600

default-server-timeout 600

socket-timeout 600

6. Arrancar el servicio Privoxy.

/etc/init.d/privoxy start

Para ejecutar la aplicación al iniciar el ordenador, deberán realizarse los pasos 3 y 6.

I2P

Los pasos a seguir para instalar el servicio de conexión a I2P en un sistema operativo Kali Linux son:

1. Comprobar si *apt-transport-https* y *curl* están instalados.

sudo apt-get update

sudo apt-get install apt-transport-https curl

2. Instalar la aplicación I2P.

sudo apt-get install i2p

3. Arrancar el servicio I2P.

i2prouter start

4. Realizar la configuración del servicio de navegación I2P. Tras la ejecución se mostrará pantalla relativa al router I2P.

Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 3.

Freenet

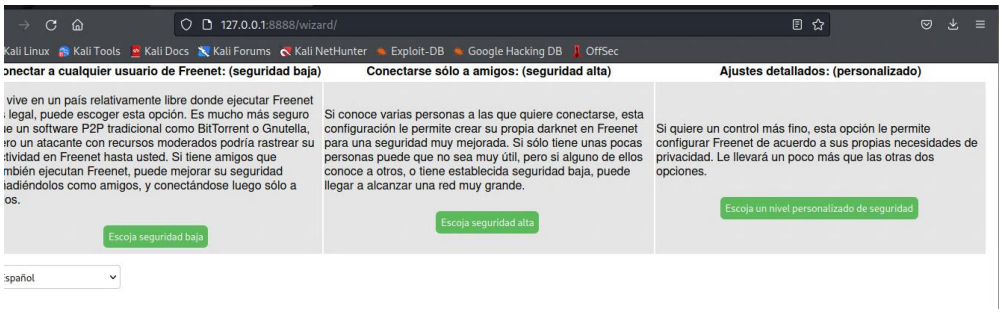
Los pasos a seguir para instalar el servicio de conexión a esta Dark Net en un sistema operativo Kali Linux son:

1. Descargar el asistente de instalación Freenet de la página web de descarga de The Freenet Project Inc. Ejecutar el instalador:

java -jar new_installer_offline_1493.jar

2. Aparecerá el asistente de configuración en la primera conexión, mostrado en la Figura 69. Se podrá seleccionar la opción '*Conectar a cualquier usuario de Freenet (seguridad baja)*'. Escoger las diferentes opciones de configuración. Como resultado, se mostrará un navegador con acceso a esta Dark Net.

Figura 69. Ejecución de Freenet. Asistente primera conexión.



Fuente: Elaboración propia.

Para ejecutar la aplicación al iniciar el ordenador, deberá ejecutarse el siguiente comando desde el directorio home del usuario:

./Freenet/bin/browse.sh

MongoDB

Los pasos a seguir para instalar esta base de datos no relacional en un sistema operativo Kali Linux son:

1. Importar la clave pública empleada por el sistema de gestión de paquetes.

wget -qO - https://www.mongodb.org/static/pgp/server-5.0.asc | sudo apt-key add –

2. Crear un fichero */etc/apt/sources.list.d/mongodb-org-5.0.list* para MongoDB.

echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/5.0 main" | sudo tee /etc/apt/sources.list.d/mongodb-org-5.0.list

3. Actualizar la base de datos local de paquetes.

sudo apt-get update

4. Instalar la última versión de MongoDB.

sudo apt-get install -y mongodb-org

5. Permitir conexiones remotas modificando el fichero "/etc/mongod.conf", reemplazando la línea bind_ip = 127.0.0.1 por bind_ip = 0.0.0.0.

6. Iniciar MongoDB.

sudo systemctl start mongod

7. Crear usuarios mediante el comando *mongosh* siguiendo los pasos especificados para el sistema operativo Windows.

8. Realizar las acciones de bastionado contempladas en las guías especificadas en el capítulo "4.5 Producción".

9. En el fichero *darkfinder.conf* indicar en la clave *db_server* la dirección IP del servidor (localhost en caso de que sea la propia máquina).

10. Introducir en el fichero apuntado por la clave *database_user_password_filename* del archivo de configuración *darkfinder.conf* el usuario y contraseña para la base de datos configurado con anterioridad.

Para ejecutar la aplicación al iniciar el ordenador, deberá realizarse el paso 6.

Elasticsearch

Los pasos a seguir para instalar esta herramienta en un sistema operativo Kali Linux son:

1. Descargar Elasticsearch (v8.2.2) de la página web de descarga de Elasticsearch.
2. Descargar Kibana (v8.2.2) de la página web de descarga de Kibana.
3. Descomprimir los ficheros zip resultantes.
4. Abrir una terminal de comandos y cambiar la ruta a la carpeta donde se ha descomprimido el contenido de Elasticsearch. Arrancar la aplicación mediante el siguiente comando:

./bin/elasticsearch

5. Anotar el password y el '*Enrollment token*' mostrados en pantalla para configurar Kibana en su arranque.
6. Abrir una terminal de comandos y cambiar la ruta a la carpeta donde se ha descomprimido el contenido de Kibana. Arrancar la aplicación mediante el siguiente comando:

./bin/kibana

7. Hacer click en el enlace proporcionado en el terminal, o abrir en navegador a la dirección http://localhost:5601 y seguir las instrucciones para conectarse a Kibana.
8. Pegar el '*Enrollment token*' mostrado en pantalla al arrancar Elasticsearch por primera vez (referenciado en el paso 5) y pulsar en la opción '*Configure Elastic*' para la configuración automática de Elasticsearch y Kibana.
9. Conectarse a Kibana (http://localhost:5601/) e introducir el usuario '*elastic*' y la contraseña proporcionada al arrancar Elasticsearch por primera vez y referenciada en el paso 5.
10. Seguir el resto de pasos indicados para sistema operativo Windows.

Para ejecutar la aplicación al iniciar el ordenador, deberán realizarse los pasos 4, 6 y 9.

Anexo H. Resultados

En el presente anexo se incluyen evidencias de los resultados expuestos en el capítulo "4.6 Resultados".

Mostrar mensajes de ayuda y versión de la aplicación

Los comandos *darkfinder* y *darksearch* muestran mensajes con la ayuda de la aplicación (opción -h) y versión (opción -v) según se muestra en la Figura 70 y la Figura 71:

Figura 70. Opción ayuda (-h) comandos *darkfinder* y *darksearch*.

```
C:\Users\Sergio\source\repos\DarkFinder>python darkfinder.py -h
uso : Crawler para patrones delictivos en la web visible y darknets
python darkfinder.py [opciones]

opciones:
  -h, --help     muestra este mensaje de ayuda y sale del programa
  -v, --version  muestra la versión de DarkFinder
  -q, --quiet    no muestra mensajes en pantalla

C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -h
uso : Crawler para patrones delictivos en la web visible y darknets
Comando para buscar páginas web rastreadas. Debe ser ejecutado después del script darkfinder
python darksearch.py [opciones]

opciones:
  -h, --help            muestra este mensaje de ayuda y sale del programa
  -v, --version         muestra la versión de DarkFinder
  -u URL, --url URL     busca páginas web rastreadas por su URL
  -t TEXT, --text TEXT  busca páginas rastreadas en función del contenido HTML body
  --hash HASH           busca páginas rastreadas que contienen imágenes cuyo hash MD5/SHA1 corresponde al
                        proporcionado como parámetro
  -d REF_HASH DISTANCE, --dhash REF_HASH DISTANCE
                        busca páginas rastreadas que contienen imágenes cuyo difference hash tiene una distancia de
                        Hamming DISTANCE a la referencia proporcionada REF_HASH
  -m METADATA, --metadata METADATA
                        busca páginas rastreadas que contienen imágenes cuyos metatados contienen un cierto texto
```

Fuente: Elaboración propia.

Figura 71. Opción versión (-v) comandos *darkfinder* y *darksearch*.

```
C:\Users\Sergio\source\repos\DarkFinder>python darkfinder.py -v
Versión : 1.0.0

C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -v
Versión : 1.0.0
```

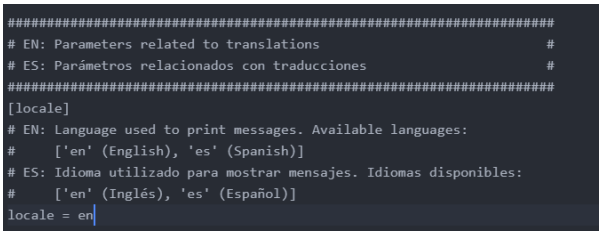
Fuente: Elaboración propia.

Funcionamiento en español e inglés

En el fichero de configuración se puede configurar el idioma español o inglés mediante la clave '*locale*'. Tanto los comandos *darkfinder* como *darksearch* funcionan en ambos idiomas, incluyendo tanto el menú de ayuda como los mensajes mostrados por pantalla.

Nota: en general, el número de páginas web rastreadas en todos los ejemplos mostrados en el presente anexo es 10, siendo únicamente ejemplos representativos, no pretendiendo realizar un rastreo real. En caso de que se ejecuten rastreos concurrentes (opción por defecto empleada por la librería scrapy) se pueden analizar más páginas de las indicadas por la clave 'max_crawled_web_pages' del fichero de configuración, si bien el número máximo de páginas almacenadas en base de datos será el indicado por este valor.

Figura 72. Fichero de configuración. Clave 'locale'.

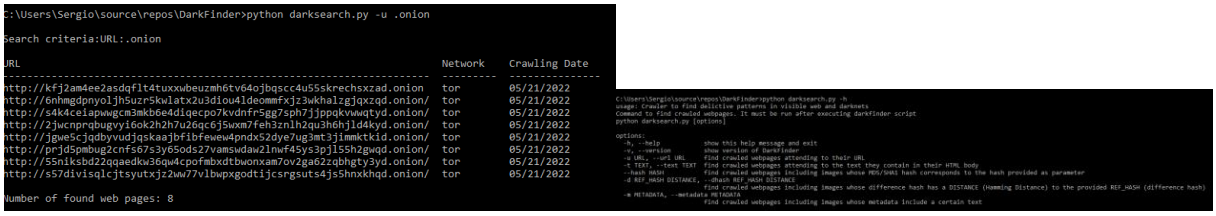


Fuente: Elaboración propia.

Figura 73. Funcionamiento comando *darkfinder* en inglés.

Fuente: Elaboración propia.

Figura 74. Funcionamiento comando *darksearch* en inglés.



Fuente: Elaboración propia.

Figura 75. Funcionamiento comando *darkfinder* en español.

Fuente: Elaboración propia.

Figura 76. Funcionamiento comando *darksearch* en español.

Fuente: Elaboración propia.

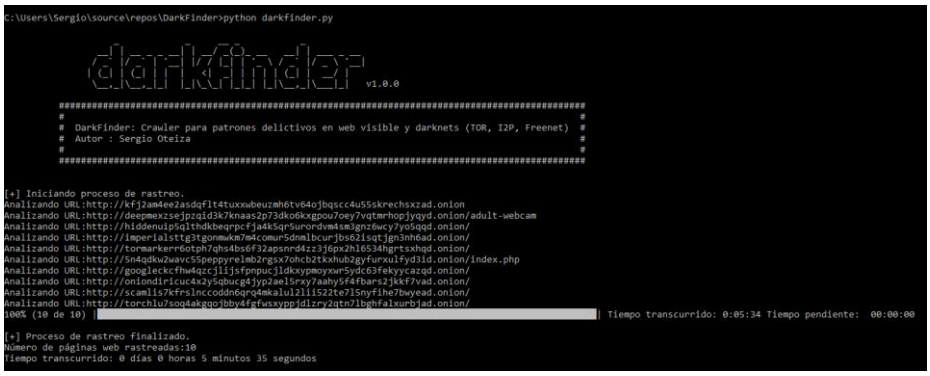
Capacidad de rastreo en red visible, TOR, I2P y Freenet

La aplicación es capaz de realizar rastreos en la web visible, red TOR, I2P y Freenet, del modo mostrado en la Figura 77, la Figura 78, la Figura 79 y la Figura 80 Asimismo, permite saltar entre diversas redes en el proceso de *crawling*, según se indica en la Figura 81:

Figura 77. Rastreo de páginas web en red visible.

Fuente: Elaboración propia.

Figura 78. Rastreo de páginas web en red TOR.



Fuente: Elaboración propia.

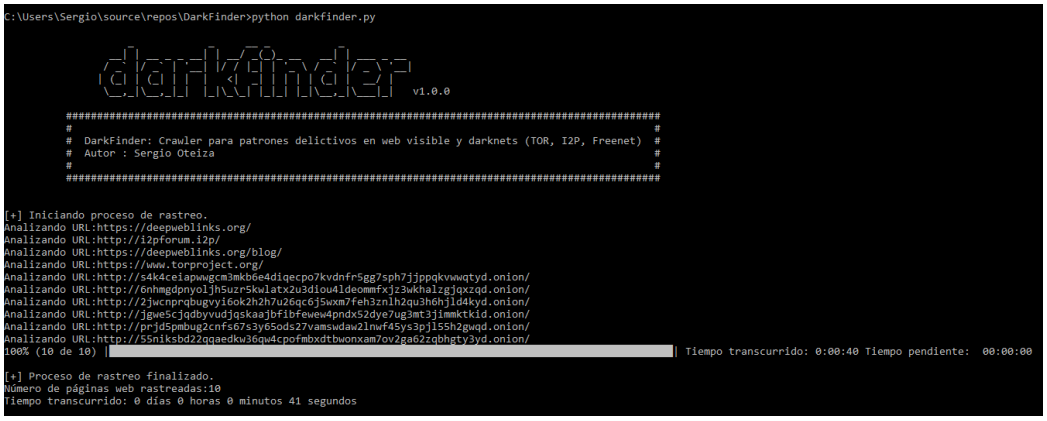
Figura 79. Rastreo de páginas web en red I2P.

Fuente: Elaboración propia.

Figura 80. Rastreo de páginas web en red Freenet.

Fuente: Elaboración propia.

Figura 81. Rastreo de páginas web en múltiples redes.



Fuente: Elaboración propia.

Por último, el comando *darkfinder* permite un rastreo en modo silencioso o '*quiet*' (opción -q), según lo mostrado en la Figura 82.

Figura 82. Funcionamiento comando *darkfinder* en español modo silencioso (opción -q).

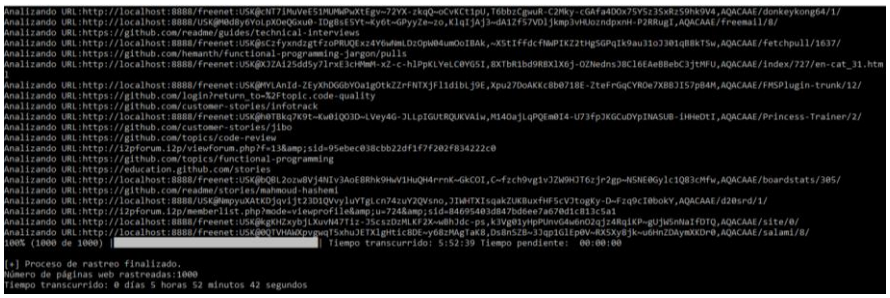
Fuente: Elaboración propia.

La Figura 83 y la Figura 84 muestran respectivamente el comienzo y fin del proceso de rastreo de 1.000 páginas web, las cuales se almacenan en la base de datos MongoDB, según lo mostrado en la Figura 85.

Figura 83. Comienzo del proceso de rastreo de 1.000 páginas web.

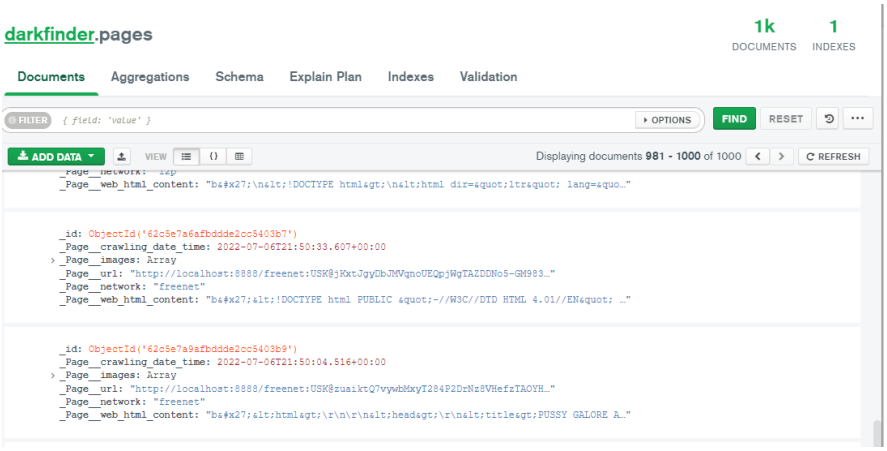
Fuente: Elaboración propia.

Figura 84. Fin del proceso de rastreo de 1.000 páginas web.



Fuente: Elaboración propia.

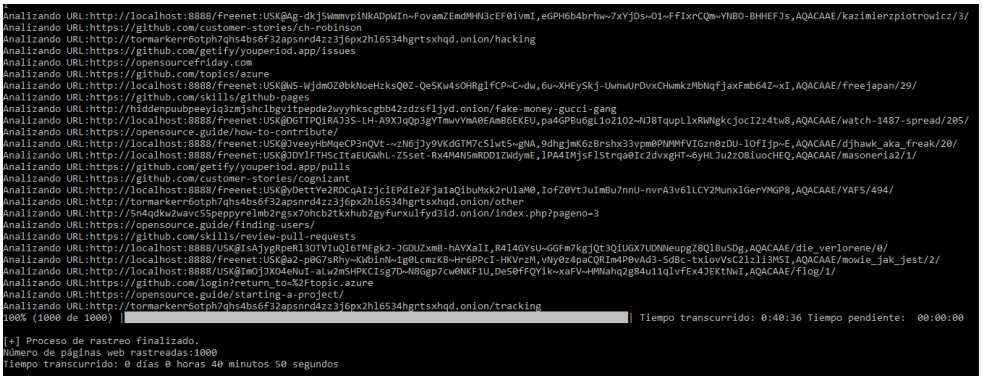
Figura 85. Base de datos MongoDB en el proceso de rastreo de 1.000 páginas web.



Fuente: Elaboración propia.

Una de las formas de reducir el tiempo de *crawling* es evitar el procesado de imágenes. Para ello, puede hacerse uso del parámetro WEBPAGE_CRAWL_IMAGES en el fichero *settings.py*, estableciendo su valor a *False.* El tiempo de rastreo resultante puede comprobarse en la Figura 86.

Figura 86. Rastreo de 1.000 páginas web sin procesar imágenes.



Fuente: Elaboración propia.

Almacenamiento de información en base de datos no relacional MongoDB

Se muestran a continuación extractos de la base de datos MongoDB con registros de las redes visible (Figura 87), TOR (Figura 88), I2P (Figura 89) y Freenet (Figura 90).

Figura 87. Registro MongoDB página red visible.

Fuente: Elaboración propia.

Figura 88. Registro MongoDB página red TOR.

Fuente: Elaboración propia.

Figura 89. Registro MongoDB página red I2P.

Fuente: Elaboración propia.

Figura 90. Registro MongoDB página red Freenet.

Fuente: Elaboración propia.

Implementación de diferentes algoritmos de rastreo

En este capítulo se incluyen los resultados de las 3 estrategias de rastreo contempladas en la herramienta:

- Breadth-First Search (BFS).
- Depth-First Search (DFS).
- Best-First Search.

Para cada una de ellas se muestra el fichero de configuración, el proceso de rastreo y el resultado de búsqueda de las palabras clave o *keywords* consideradas: '*sex*', '*porn*', '*drugs*', '*abuse*'.

Nota: la librería scrapy empleada en el proceso de rastreo tiene carácter asíncrono, por lo que es posible que se analicen varias URLs en paralelo y la secuencia de páginas web analizadas pueda variar respecto a la teórica esperada.

La Tabla 52 muestra una comparativa de los resultados obtenidos para cada uno de los algoritmos, contemplando las métricas especificadas en el capítulo "2.3.4 Métricas asociadas a los crawlers", considerando que las páginas relevantes son aquellas que contienen al menos una de las palabras clave seleccionadas. Teniendo en cuenta que se trata de un ejercicio de demostración, el número de páginas web existentes en la red que contienen al menos una de dichas palabras clave es un valor no conocido, muy superior a las diez páginas rastreadas (máximo número de documentos relevantes que se pueden localizar). Por este motivo,

únicamente se calcula la métrica correspondiente a la precisión, ya que los valores relativos a la exhaustividad serían en todo caso muy reducidos.

Tabla 52. Precisión de los algoritmos de rastreo.

Algoritmo	Precisión
Breadth-First Search (BFS)	0,6
Depth-First Search (DFS)	0,4
Best-First Search	0,9

Fuente: Elaboración propia.

Nota: este resultado se muestra únicamente para demostrar la posibilidad de funcionamiento de diversos algoritmos de rastreo. Dado el bajo número de webs rastreadas, no deberían extrapolarse los resultados obtenidos para comparar su desempeño.

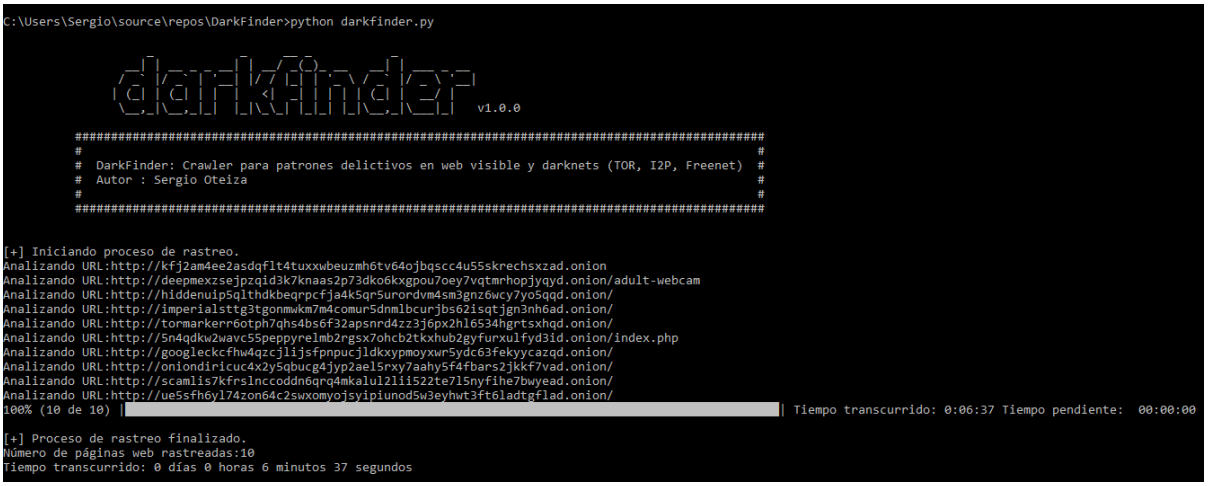
Breadth-First Search

Figura 91. Fichero de configuración rastreo tipo Breadth-First Search.

```
#################################################################
# EN: Parameters related to crawling process                    #
# ES: Parámetros relacionados con el proceso de rastreo         #
#################################################################
[crawling]
# EN: Array with the seed URL that launch crawling process
# ES: Array con las URL semilla que lanzan el proceso de crawling
start_url = ["http://kfj2am4ee2asdqflt4tuxxwbeuzmh6tv64ojbqscc4u55skrechsxzad.onion/doku.php"]

# EN: Maximum crawling depth
# ES: Máxima profundidad de rastreo
max_depth_crawling = 5
# EN: Maximum number of web pages that can be crawled
# ES: Número máximo de páginas que pueden ser rastreadas
max_crawled_web_pages = 10
# EN: Crawling strategy. Available strategies:
#     ['bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best-First Search)]
# ES: estrategia de rastreo. Valores permitidos:
#     ['bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best-First Search)]
crawling_strategy = bfs
```

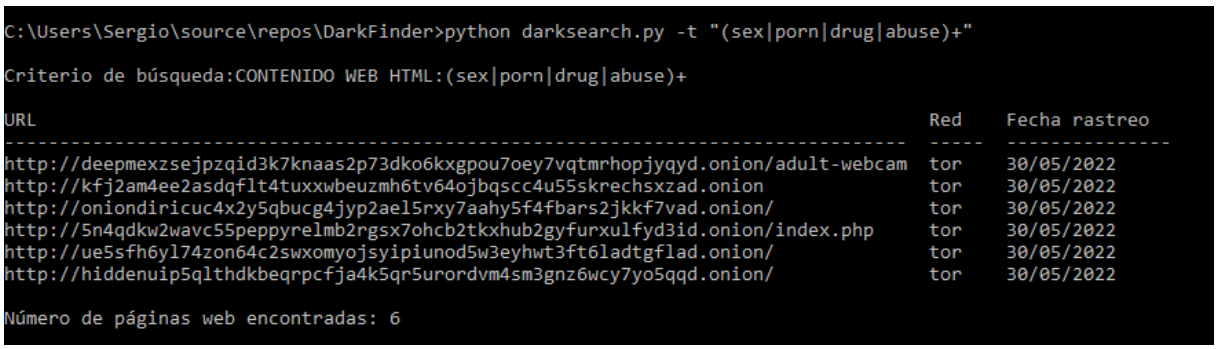
Fuente: Elaboración propia.

Figura 92. Proceso rastreo tipo Breadth-First Search.



Fuente: Elaboración propia.

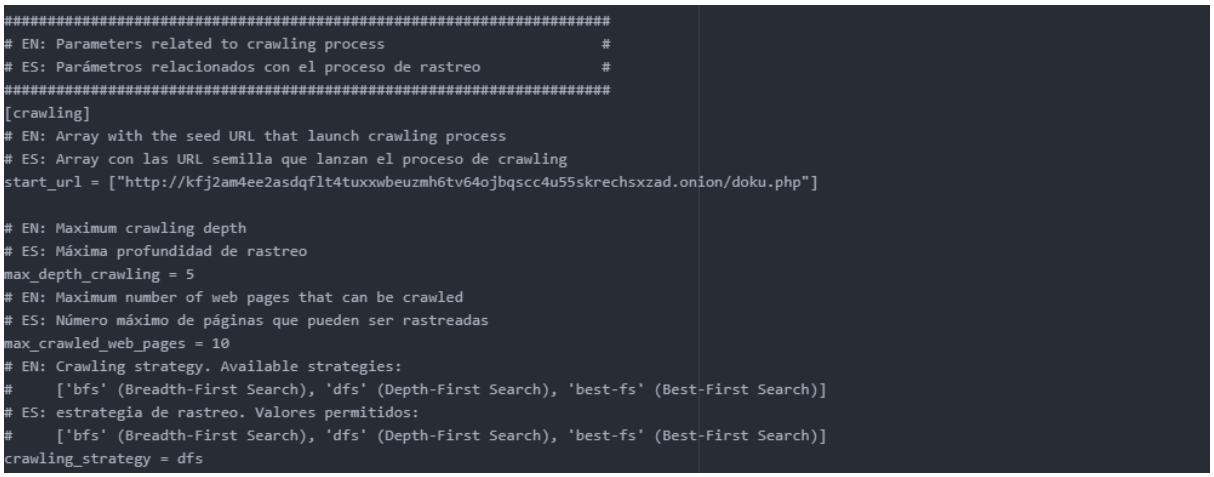
Figura 93. Búsqueda *keywords* rastreo tipo Breadth-First Search.



Fuente: Elaboración propia.

Depth-First Search

Figura 94. Fichero de configuración rastreo tipo Depth-First Search.



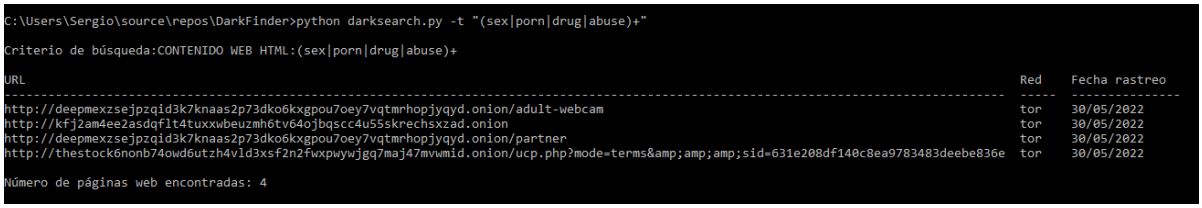
```
#####################################################################
# EN: Parameters related to crawling process                        #
# ES: Parámetros relacionados con el proceso de rastreo             #
#####################################################################
[crawling]
# EN: Array with the seed URL that launch crawling process
# ES: Array con las URL semilla que lanzan el proceso de crawling
start_url = ["http://kfj2am4ee2asdqflt4tuxxwbeuzmh6tv64ojbqscc4u55skrechsxzad.onion/doku.php"]

# EN: Maximum crawling depth
# ES: Máxima profundidad de rastreo
max_depth_crawling = 5
# EN: Maximum number of web pages that can be crawled
# ES: Número máximo de páginas que pueden ser rastreadas
max_crawled_web_pages = 10
# EN: Crawling strategy. Available strategies:
#     ['bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best-First Search)]
# ES: estrategia de rastreo. Valores permitidos:
#     ['bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best-First Search)]
crawling_strategy = dfs
```

Fuente: Elaboración propia.

Figura 95. Proceso rastreo tipo Depth-First Search.

Fuente: Elaboración propia.

Figura 96. Búsqueda *keywords* rastreo tipo Depth-First Search.



Fuente: Elaboración propia.

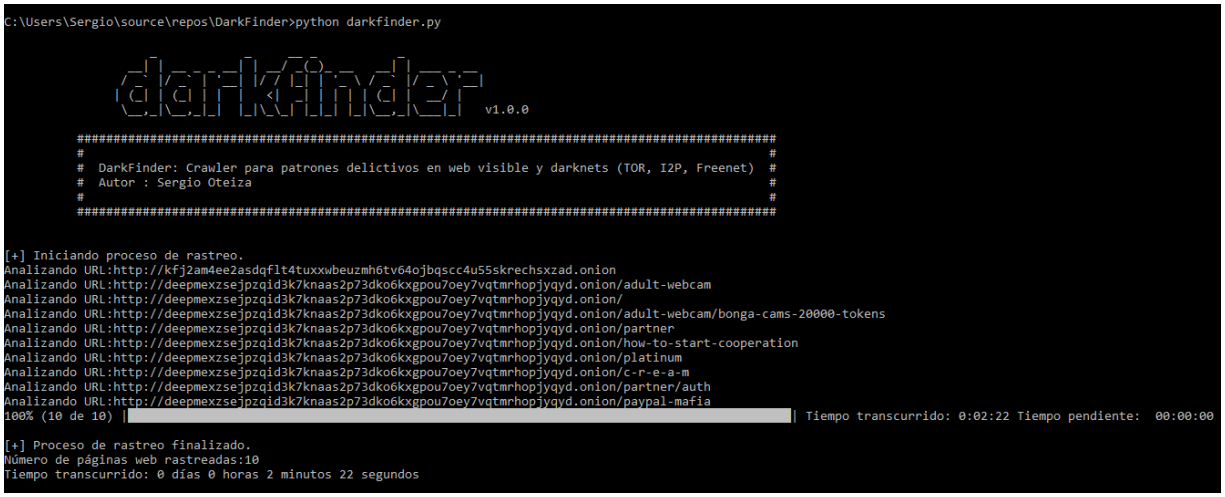
Best-First Search

Figura 97. Fichero de configuración rastreo tipo Best-First Search.

```
#########################################################################
# EN: Parameters related to crawling process                            #
# ES: Parámetros relacionados con el proceso de rastreo                 #
#########################################################################
[crawling]
# EN: Seed URL that launch crawling process
# ES: URLs semilla que lanzan el proceso de crawling
start_url = ["http://kfj2am4ee2asdqflt4tuxxwbeuzmh6tv64ojbqscc4u55skrechsxzad.onion/doku.php"]

# EN: Maximum crawling depth
# ES: Máxima profundidad de rastreo
max_depth_crawling = 5
# EN: Maximum number of web pages that can be crawled
# ES: Número máximo de páginas que pueden ser rastreadas
max_crawled_web_pages = 10
# EN: Crawling strategy. Available strategies:
#     ['bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best-First Search)]
# ES: Estrategia de rastreo. Valores disponibles:
#     ['bfs' (Breadth-First Search), 'dfs' (Depth-First Search), 'best-fs' (Best-First Search)]
crawling_strategy = best-fs
# EN: Keywords. Strings containing keywords that lead crawling process in Best-First Search strategy
# EN: Palabras clave. Strings que contienen las palabras clave que guían el proceso de rastreo en una estrategia Best-First Search
keywords = ["sex", "porn", "drugs", "abuse"]
```

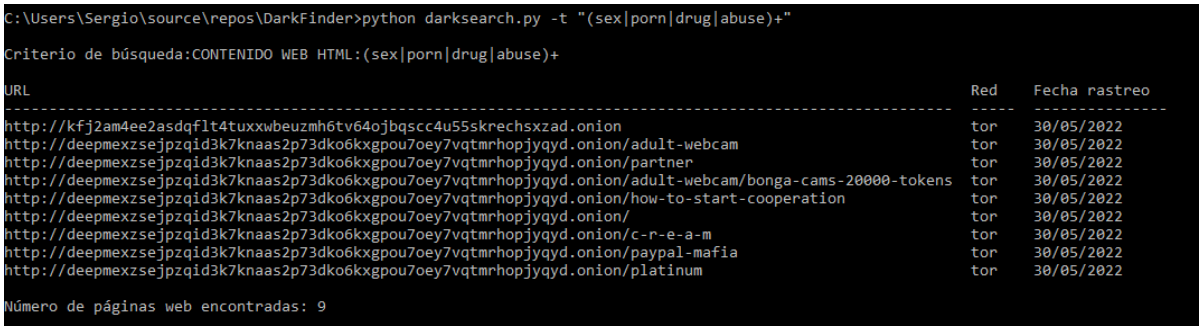
Fuente: Elaboración propia.

Figura 98. Proceso rastreo tipo Best-First Search.



Fuente: Elaboración propia.

Figura 99. Búsqueda *keywords* rastreo tipo Best-First Search.



Fuente: Elaboración propia.

Capacidad de búsqueda de páginas web

Se muestra a continuación diversas búsquedas de páginas rastreadas, incluyendo:

- Patrón de búsqueda de URL (Figura 100).
- Patrón de contenido mediante expresión en el cuerpo HTML (Figura 101).
- Hash MD5/SHA1 de imagen contenida en la página web (Figura 102 y Figura 103).
- Distancia de Hamming a un *difference hash* de referencia de una imagen pasado como parámetro (Figura 104).
- Patrón de búsqueda mediante expresión regular en los metadatos de una imagen (Figura 105).
- Búsqueda multicriterio (Figura 106).

Figura 100. Búsqueda de páginas web. Patrón de búsqueda URL.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -u .onion

Criterio de búsqueda:URL:.onion

URL                                                                          Red    Fecha rastreo
---------------------------------------------------------------------------  -----  ---------------
http://kfj2am4ee2asdqflt4tuxxwbeuzmh6tv64ojbqscc4u55skrechsxzad.onion        tor    22/05/2022
http://imperialsttg3tgonmwkm7m4comur5dnmlbcurjbs62isqtjgn3nh6ad.onion/       tor    22/05/2022
http://tormarkerr6otph7qhs4bs6f32apsnrd4zz3j6px2hl6534hgrtsxhqd.onion/        tor    22/05/2022
http://5n4qdkw2wavc55peppyrelmb2rgsx7ohcb2tkxhub2gyfurxulfyd3id.onion/index.php  tor    22/05/2022
http://googleckcfhw4qzcjlijsfpnpucjldkxypmoyxwr5ydc63fekyycazqd.onion/        tor    22/05/2022
http://scamlis7kfrslnccoddn6qrq4mkalul2lii522te7l5nyfihe7bwyead.onion/        tor    22/05/2022
http://ue5sfh6yl74zon64c2swxomyojsyipiunod5w3eyhwt3ft6ladtgflad.onion/        tor    22/05/2022
http://torchlu7soq4akgqojbby4fgfwsxyppjdlzry2qtn7lbghfalxurbjad.onion/        tor    22/05/2022

Número de páginas web encontradas: 8
```

Fuente: Elaboración propia.

Figura 101. Búsqueda de páginas web. Patrón de contenido mediante expresión regular en el cuerpo HTML.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -t [\w]{2,50}@+[\w]{2,50}\.+[a-zA-Z]{2,4}

Criterio de búsqueda:CONTENIDO WEB HTML:[\w]{2,50}@+[\w]{2,50}\.+[a-zA-Z]{2,4}

URL                                                                  Red      Fecha rastreo
-------------------------------------------------------------------  -------  ---------------
http://digdeep4orxw6psc33yxa2dgmuycj74zi6334xhxjlgppw6odvkzkiad.onion/  tor      31/05/2022
https://ar.al                                                        visible  31/05/2022

Número de páginas web encontradas: 2
```

Fuente: Elaboración propia.

Figura 102. Búsqueda de páginas web. Hash MD5 de imagen contenida en la página web.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py --hash e593c04fba8d761284a588f701dbbed0

Criterio de búsqueda:HASH IMAGEN:e593c04fba8d761284a588f701dbbed0

URL                                                                  Red    Fecha rastreo    Fichero imagen
-------------------------------------------------------------------  -----  ---------------  ---------------
http://imperialsttg3tgonmwkm7m4comur5dnmlbcurjbs62isqtjgn3nh6ad.onion  tor    31/05/2022       img-header-1.png

Número de páginas web encontradas: 1

Número de imágenes encontradas: 1
```

Fuente: Elaboración propia.

Figura 103. Búsqueda de páginas web. Hash SHA1 de imagen contenida en la página web.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py --hash f7dc44ab114bb5d64948371bdbe9e93a00cf4e77

Criterio de búsqueda:HASH IMAGEN:f7dc44ab114bb5d64948371bdbe9e93a00cf4e77

URL                                                                  Red    Fecha rastreo    Fichero imagen
-------------------------------------------------------------------  -----  ---------------  ---------------
http://imperialsttg3tgonmwkm7m4comur5dnmlbcurjbs62isqtjgn3nh6ad.onion  tor    31/05/2022       img-header-1.png

Número de páginas web encontradas: 1

Número de imágenes encontradas: 1
```

Fuente: Elaboración propia.

Figura 104. Búsqueda de páginas web. Distancia de Hamming a un *difference hash* de referencia de una imagen pasado como parámetro.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -d b0b8b630de86c071 25

Criterio de búsqueda:REFERENCIA DIFFERENCE HASH:b0b8b630de86c071 Y DISTANCIA HAMMING DIFFERENCE HASH:25

URL                                                                          Red    Fecha rastreo    Fichero imagen      Distancia Hamming
---------------------------------------------------------------------------  -----  ---------------  ----------------  -----------------
http://amazon44exbjrzdh4ysbq4tlkuh2w56l5xkpqelhign7xtp5fow55said.onion/      tor    10/07/2022       amzn.png                          0
http://amazon44exbjrzdh4ysbq4tlkuh2w56l5xkpqelhign7xtp5fow55said.onion/      tor    10/07/2022       safe.png                         23
http://imperial2tmx26sfzhkr5dcvbrjdq7qao3zc3d3qahpbptbfghxbbjid.onion        tor    10/07/2022       imp4.jpg                         24
http://imperial2tmx26sfzhkr5dcvbrjdq7qao3zc3d3qahpbptbfghxbbjid.onion        tor    10/07/2022       review-2.png                     23
http://imperial2tmx26sfzhkr5dcvbrjdq7qao3zc3d3qahpbptbfghxbbjid.onion        tor    10/07/2022       review-3.png                     23

Número de páginas web encontradas: 2

Número de imágenes encontradas: 5
```

Fuente: Elaboración propia.

Figura 105. Búsqueda de páginas web. Patrón de búsqueda mediante expresión regular en los metadatos de una imagen.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -m gps

Criterio de búsqueda:METADATOS DE IMAGEN:gps

URL                                                                          Red      Fecha rastreo    Fichero imagen
---------------------------------------------------------------------------  -------  ---------------  ----------------
https://github.com/ianare/exif-samples/blob/master/jpg/gps/DSCN0010.jpg      visible  22/05/2022       DSCN0010.jpg

Número de páginas web encontradas: 1

Número de imágenes encontradas: 1
```

Fuente: Elaboración propia.

Figura 106. Búsqueda de páginas web. Patrón de búsqueda mediante múltiples criterios.

```
C:\Users\Sergio\source\repos\DarkFinder>python darksearch.py -u github -t !DOCTYPE --hash 80b4903d0deec6d9f598b06649563e03 -d 313c1d66e2e4e595 5 -m GPS

Criterio de búsqueda:URL:github Y CONTENIDO WEB HTML:!DOCTYPE Y HASH IMAGEN:80b4903d0deec6d9f598b06649563e03 Y REFERENCIA DIFFERENCE HASH:313c1d66e2e4e595 Y DISTANCIA H
AMMING DIFFERENCE HASH:5 Y METADATOS DE IMAGEN:GPS

URL                                                                          Red      Fecha rastreo    Fichero imagen      Distancia Hamming
---------------------------------------------------------------------------  -------  ---------------  ----------------  -----------------
https://github.com/ianare/exif-samples/blob/master/jpg/gps/DSCN0010.jpg      visible  10/07/2022       DSCN0010.jpg                      0

Número de páginas web encontradas: 1

Número de imágenes encontradas: 1
```

Fuente: Elaboración propia.

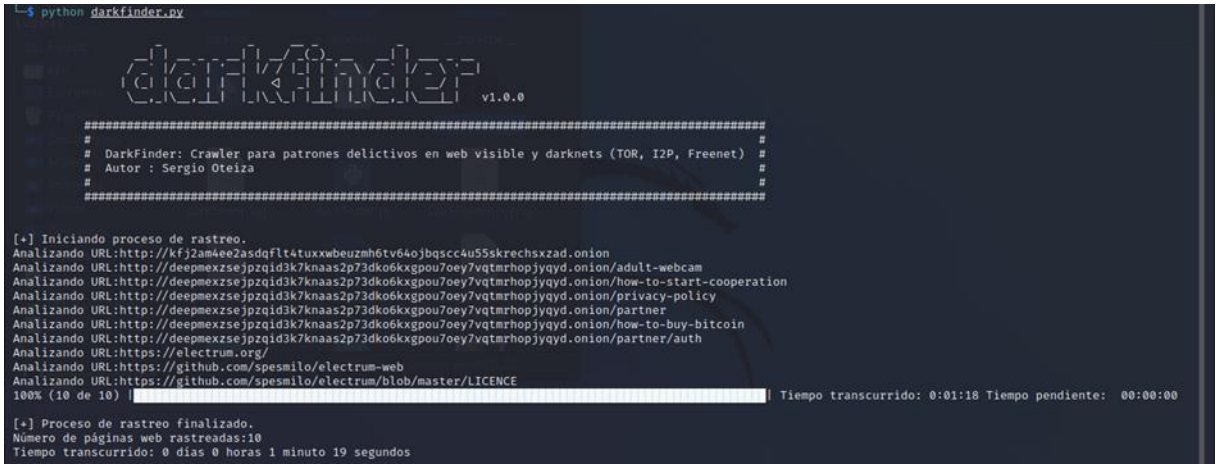
Funcionamiento en sistema operativo Kali Linux

Se recoge a continuación, para sistema operativo Kali Linux, la ejecución del comando *darkfinder* en red visible (Figura 107), TOR (Figura 108), I2P (Figura 109) y Freenet (Figura 110).

Figura 107. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red visible.

Fuente: Elaboración propia.

Figura 108. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red TOR.



Fuente: Elaboración propia.

Figura 109. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red I2P.

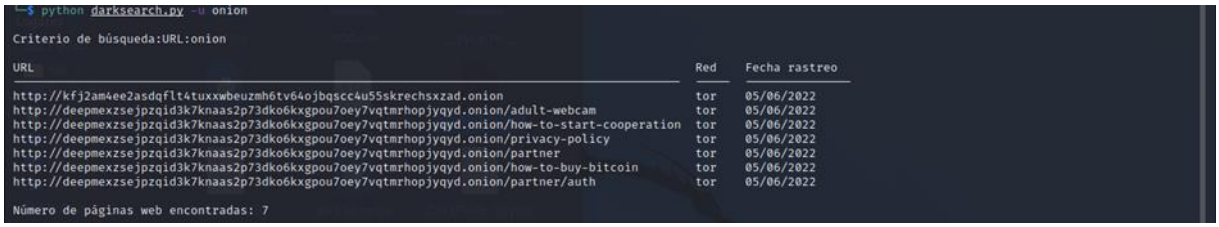
Fuente: Elaboración propia.

Figura 110. Ejecución de comando *darkfinder* en sistema operativo Kali Linux. Red Freenet.

Fuente: Elaboración propia.

La Figura 111 muestra la ejecución de comando *darksearch* en sistema operativo Kali Linux.

Figura 111. Ejecución de comando *darksearch* en sistema operativo Kali Linux.

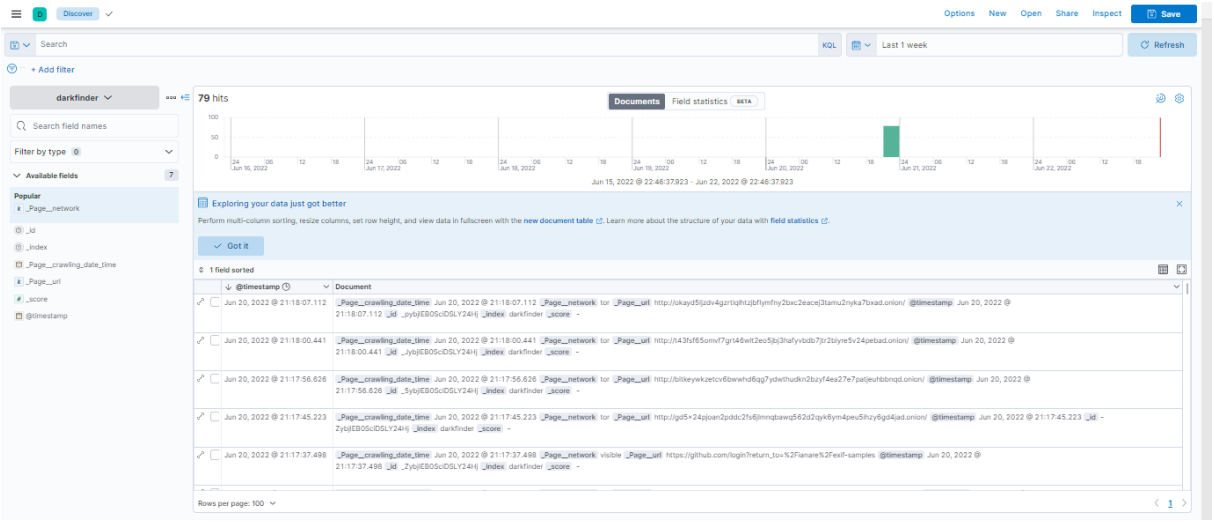


Fuente: Elaboración propia.

Carga de datos de rastreo en aplicación Elasticsearch y visualización con Kibana

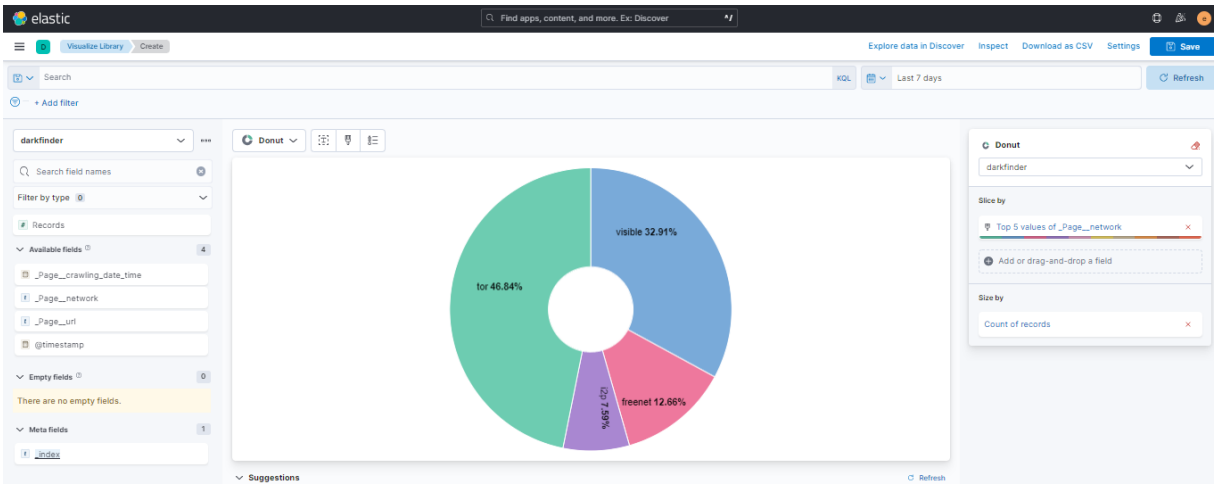
La Figura 112 muestra una pantalla de Kibana para el análisis en Elasticsearch de la información de rastreo cargada. La Figura 113 recoge un ejemplo de análisis de los datos de rastreo obtenidos, mostrando la distribución de las redes a las que pertenecen las webs detectadas.

Figura 112. Análisis de información cargada y procesada por Elasticsearch mediante Kibana.



Fuente: Elaboración propia.

Figura 113. Análisis de redes rastreadas mediante Kibana.



Fuente: Elaboración propia.

Anexo I. Código fuente

Se adjunta en fichero zip independiente el código fuente de la aplicación desarrollada. La Tabla 53 recoge la descripción de los ficheros de código fuente y su correspondencia con los módulos especificados en el capítulo “4.2 Diseño”.

Tabla 53. Descripción de ficheros de código fuente.

Fichero	Descripción
darkfinder.py	Código fuente que implementa el módulo DarkFinder.
darksearch.py	Código fuente que implementa el módulo DarkSearch.
auxiliary.py	Código fuente que implementa el módulo Auxiliar.
configuration.py	Código fuente que implementa el módulo Configuración.
crawler.py	Código fuente que implementa el módulo Crawler.
database_manager.py	Código fuente que implementa el módulo Gestor de base de datos.
logger.py	Código fuente que implementa el módulo Log.
settings.py	Fichero con definiciones de constantes utilizadas a lo largo del programa.
darkfinder.conf	Fichero de configuración de la aplicación.
test-darkfinder.py	Fichero con las pruebas automatizadas.

Fuente: Elaboración propia.