Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Desarrollo de un Plugin de FOCA para extraer cuentas de usuario en Redes Sociales.

Autor:
Iván García Hurtado

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

Desarrollo de un Plugin de FOCA para extraer cuentas de usuario en Redes Sociales.

Autor:
Iván García Hurtado

Tutores:
Amador Aparicio de la Fuente
Mercedes Martínez González

Resumen

Este Trabajo de Fin de Grado consiste en el diseño y desarrollo de un plugin para FOCA. FOCA es una herramienta de extracción de metadatos presentes en documentos ofimáticos accesibles desde Internet. El plugin aprovechará los servicios de inicio de sesión de distintas redes sociales para obtener información sobre las direcciones de correo electrónico encontradas por FOCA. El plugin se ha construido en C# e integrado en FOCA mediante la API de plugins de la propia herramienta.

La idea principal de este proyecto surge del concepto de *Profiling* (Perfilado de usuarios) en Ciberseguridad. La información sobre qué redes sociales está usando una persona es muy valiosa y está expuesta a Internet en la mayoría de servicios. Actualmente existen herramientas que explotan esta fuga de información, pero ninguna utiliza el correo electrónico, un dato que no puede ser considerado privado hoy en día, como punto de partida.

Índice general

1. Introducción **13**

1.1. Contexto 13
1.2. Objetivos 13
1.3. Alcance 13

2. Plan de proyecto **15**

2.1. Objetivos del proyecto 15
2.2. Metodología de proyecto 15
2.3. Fases del proyecto inicial 16
2.3.1. Diagrama de Gantt inicial 21
2.4. Replanificación del proyecto 22
2.4.1. Diagrama de Gantt replanificado 25
2.5. Plan de riesgos 26
2.6. Coste del proyecto 28

3. Preparación del proyecto **29**

3.1. FOCA 29
3.1.1. Esquema de Datos 32
3.1.2. Código Fuente 33
3.1.3. Instalación 36
3.1.4. API de Plugins 37
3.2. SQL Server 38
3.2.1. Instalación 39
3.2.2. SQL Server Management Studio 40
3.2.3. Instalación 41
3.3. Microsoft Visual Studio 43
3.3.1. Instalación 44
3.4. Microsoft .NET Framework 45
3.5. Servicio de recuperación de contraseña de Gmail 45
3.5.1. Funcionamiento 45
3.5.2. Extracción de la información expuesta 48
3.6. Estudio de Redes Sociales 54
3.7. Toma de Requisitos 55
3.7.1. Requisitos Funcionales 55
3.7.2. Requisitos no Funcionales 55
3.7.3. Requisitos de Información 56
3.7.4. Requisitos de Seguridad y Privacidad 56

4. Análisis **57**
4.1. Casos de uso . . . 57
4.1.1. Diagrama de casos de uso . . . 57
4.1.2. CU1 - Comprobar redes sociales de un correo electrónico . . . 58
4.1.3. CU2 - Mostrar historial de comprobaciones de redes sociales . . . 59
4.2. Modelo de dominio . . . 61
4.3. Diagrama de clases de análisis . . . 61
4.4. Realización de Casos de Uso de Análisis . . . 62
4.5. Flujo de información . . . 64

5. Diseño **65**
5.1. Arquitectura . . . 65
5.2. Interfaz de usuario . . . 66
5.3. Diagrama de Clases de Diseño . . . 70
5.4. Realización de los Casos de Uso de Diseño . . . 70
5.5. Persistencia de la información . . . 72
5.5.1. Bases de datos . . . 72
5.5.2. Exportación CSV . . . 73

6. Implementación **75**
6.1. Extracción de información de redes sociales . . . 75
6.1.1. Selenium . . . 75
6.2. Interfaz gráfica . . . 79
6.3. Acceso a datos . . . 81

7. Pruebas **83**
7.1. Batería de pruebas . . . 83

8. Manual de instalación y mantenimiento **91**
8.1. Manual de instalación . . . 91
8.2. Estructura de archivos del proyecto . . . 93

9. Conclusiones y trabajo futuro **97**
9.1. Conclusiones . . . 97
9.2. Trabajo futuro . . . 97

Bibliografía **99**

Índice de figuras

2.1. Fases del modelo de desarrollo en cascada . . . 16
2.2. Diagrama de Gantt - Preparación . . . 17
2.3. Diagrama de Gantt - Análisis . . . 18
2.4. Diagrama de Gantt - Diseño . . . 19
2.5. Diagrama de Gantt - Implementación . . . 20
2.6. Diagrama de Gantt - Pruebas . . . 20
2.7. Diagrama de Gantt - Mantenimiento . . . 21
2.8. Diagrama de Gantt de la planificación inicial . . . 21
2.9. Diagrama de Gantt - Análisis (II) . . . 23
2.10. Diagrama de Gantt - Diseño (II) . . . 23
2.11. Diagrama de Gantt - Implementación (II) . . . 24
2.12. Diagrama de Gantt - Pruebas (II) . . . 25
2.13. Diagrama de Gantt - Mantenimiento (II) . . . 25
2.14. Diagrama de Gantt de la replanificación . . . 26

3.1. Logotipo de la herramienta FOCA . . . 29
3.2. Metadatos estándar de una imagen JPEG . . . 30
3.3. Ejemplo de extracción de metadatos de documentos en un dominio . . . 30
3.4. Ejemplo de análisis de red . . . 31
3.5. Esquema de la base de datos de FOCA (I). Obtenido con SSMS (ver apartado 3.2.2). . . . 33
3.6. Esquema de la base de datos de FOCA (II). Obtenido con SSMS (ver apartado 3.2.2). . . . 33
3.7. Entidades y controladores de Entity Framework . . . 34
3.8. Contenido de `./SearcherCore` . . . 35
3.9. Contenido de `./MetadataExtractCore` . . . 36
3.10. Contenido de `./FOCA/Analysis` . . . 36
3.11. Ejemplo de plugin FOCA (Social Scan) en el menú contextual . . . 38
3.12. Aplicaciones y Características - Windows . . . 39
3.13. Instalador de SQL Server 2016 . . . 40
3.14. Conexión a una instancia - SQL Server Management Studio . . . 41
3.15. Interfaz de SQL Server Management Studio. . . . 41
3.16. Localización del ejecutable de SQL Server Management Studio . . . 42
3.17. Herramienta de edición del Registro de Windows (*regedit*) . . . 43
3.18. Eliminar una clave - *regedit* . . . 43
3.19. Instalación de Visual Studio 2019 . . . 45
3.20. Inicio de sesión de Gmail . . . 46
3.21. *Challenges* con información personal de los usuarios . . . 47
3.22. Peticiones HTTP realizadas al iniciar sesión. . . . 47

3.23. Peticiones HTTP realizadas al iniciar los *challenges*. 48
3.24. Pantalla de rechazo de *bots* de Google 50
3.25. Visitar una página - Selenium Firefox 50
3.26. Inicio de sesión de Google con el controlador Chrome - Selenium 51
3.27. Cambio de User-Agent - Selenium Firefox 51
3.28. Modo headless - Selenium Firefox 51
3.29. Esperas - Selenium Firefox 52
3.30. Visitar una página - Selenium Chrome 52
3.31. Configuración de Puppeteer Sharp 53
3.32. Configuración de Selenium Stealth 53
3.33. Captcha de inicio de sesión 54

4.1. Diagrama de casos de uso 57
4.2. Diagrama de secuencia - Caso de Uso 1 59
4.3. Diagrama de secuencia - Caso de Uso 2 60
4.4. Modelo de dominio 61
4.5. Diagrama de Clases de Análisis (BCE) 62
4.6. CUA1 - Comprobar redes sociales de un correo electrónico 63
4.7. CUA2 - Mostrar historial de comprobaciones de redes sociales 63
4.8. Flujo de información del sistema. 64

5.1. Arquitectura del sistema 66
5.2. Interfaz de FOCA (I) 67
5.3. Interfaz de FOCA (II) 67
5.4. Interfaz del plugin DNS Snooping [46] 68
5.5. Interfaz del plugin HaveIBeenPwned 68
5.6. Interfaz de usuario (I) 69
5.7. Interfaz de usuario (II) 69
5.8. Diagrama de Clases de Diseño 70
5.9. CUD1 - Comprobar redes sociales de un correo electrónico 71
5.10. CUD2 - Mostrar historial de comprobaciones de redes sociales 72
5.11. Tabla *EmailsItems* en el esquema de datos de FOCA (obtenido con SSMS, ver apartado 3.2.2) 73

6.1. Logo de Selenium 75
6.2. Configuración del *WebDriver* de Firefox 76
6.3. Esperas soportadas - Selenium 77
6.4. Visitar una página y buscar un elemento - Selenium 78
6.5. Aviso de uso de *Cookies* - Instagram 78
6.6. Manejo de los avisos de *Cookies* 78
6.7. Interfaz gráfica del historial 80
6.8. Interfaz gráfica principal 81
6.9. Interfaz gráfica principal durante un proceso de comprobación de redes sociales 81
6.10. Implementación de la consulta con *SQLClient* 82

8.1. Compilación del código fuente del plugin en DLL usando Visual Studio 91
8.2. Contenido de `./FOCA/bin/Release` 92
8.3. Interfaz principal de FOCA con el menú *Plugins* 92

8.4. Carga de un plugin - FOCA 93
8.5. Menú *Plugins* con el plugin cargado 93
8.6. Estructura de archivos del proyecto 95

Índice de cuadros

2.1. Costes del proyecto 28

3.1. Requisitos técnicos del sistema - SQL Server 2016 Express [12] 39
3.2. Requisitos técnicos del sistema - Visual Studio 2019 Community [24] 44

4.1. Caso de Uso 1 - Comprobar redes sociales de un correo electrónico 58
4.2. Caso de Uso 2 - Mostrar historial de comprobaciones de redes sociales 60

6.1. Elementos sintácticos de XPath [49] 77

7.1. Caso de Prueba 1 83
7.2. Caso de Prueba 2 84
7.3. Caso de Prueba 3 84
7.4. Caso de Prueba 4 84
7.5. Caso de Prueba 5 85
7.6. Caso de Prueba 6 85
7.7. Caso de Prueba 7 85
7.8. Caso de Prueba 8 86
7.9. Caso de Prueba 9 86
7.10. Caso de Prueba 10 86
7.11. Caso de Prueba 11 87
7.12. Caso de Prueba 12 87
7.13. Caso de Prueba 13 87
7.14. Caso de Prueba 14 88
7.15. Caso de Prueba 15 88
7.16. Caso de Prueba 16 88
7.17. Caso de Prueba 17 89

Capítulo 1

Introducción

1.1. Contexto

Dentro del ámbito de la Ciberseguridad y Privacidad, toda información que caracterice el comportamiento de un usuario es muy importante. Habiendo cada vez mas interacción entre las personas e Internet, el límite entre información pública y privada es cada vez más borroso. Existen herramientas que explotan este hecho, como FOCA, especializada en la extracción de metadatos presentes en documentos ofimáticos accesibles desde Internet, o los numerosos servicios de *tracking* de nombres de usuario.
El TFG nace en este contexto: desarrollar un plugin para FOCA que, utilizando los correos electrónicos presentes en los metadatos de documentos ofimáticos extraídos por FOCA, compruebe si esas cuentas de correo electrónico se encuentran presentes en redes sociales.

1.2. Objetivos

El fin último de este TFG es el desarrollo de un plugin FOCA que utilice los servicios de inicio de sesión de distintas redes sociales para obtener información sobre direcciones de correo electrónico. En base a esto, se proponen los siguientes objetivos:

- **Estudio de la herramienta FOCA:** Será necesario entender el funcionamiento de FOCA, su programación (es *open source*), arquitectura, y en especial su modelo de datos y su API de plugins.
- **Desarrollo del plugin:** Especificación, análisis, diseño e implementación de las funciones que debe proveer el plugin, su modelo de datos y las herramientas que utilizará.

Inicialmente la funcionalidad del plugin propuesta era recopilar la información expuesta por el servicio de recuperación de contraseña de Gmail sobre un correo electrónico dado, pero debido a un problema durante la implementación se tuvieron que cambiar los objetivos a los mencionados anteriormente.

1.3. Alcance

En base a los objetivos, el proyecto se dará por finalizado al haber completado las siguientes actividades clave:

- Estudio de la funcionalidad y arquitectura de FOCA.
- Estudio de su esquema de datos y de la API de FOCA.
- Estudio sobre qué redes sociales o servicios deberían ser utilizadas en la búsqueda de cuentas de usuario.

- Desarrollo de un plugin que utilice las direcciones de correo electrónico extraídas por FOCA para comprobar si tienen una cuenta en ciertas redes sociales y servicios.
- Redacción de un manual del programador que describa el proceso de instalación y mantenimiento del plugin desarrollado.
- Redacción de una memoria que recopile el trabajo realizado.

Capítulo 2

Plan de proyecto

En este capítulo se explicará la planificación que se ha realizado del proyecto: objetivos, metodologías utilizadas y una breve descripción de sus fases y tareas correspondientes. Primero se desglosará la planificación del proyecto inicial para después explicar la nueva planificación causada por cambio de objetivos.

2.1. Objetivos del proyecto

El objetivo del proyecto es el desarrollo de un plugin para la herramienta FOCA que determine en qué redes sociales aparece una cuenta de correo electrónico extraída de los documentos ofimáticos encontrados por FOCA.

Inicialmente el objetivo era utilizar el servicio de recuperación de contraseña de Gmail para obtener información personal de un correo electrónico, pero durante el desarrollo se encontró un problema que llevó a un cambio de los objetivos (y posterior replanificación) del proyecto.

2.2. Metodología de proyecto

En esta sección se explicará qué metodología se ha seguido durante la planificación y el desarrollo del proyecto. Se optó por seguir una "metodología de desarrollo en cascada" [1], caracterizada por ser un modelo de desarrollo secuencial en el que los resultados de una fase son utilizados en la siguiente. Generalmente, los proyectos dirigidos mediante una metodología en cascada tienen la siguiente estructura:

Figura 2.1: Fases del modelo de desarrollo en cascada

Entonces siguiendo la estructura general del modelo, este proyecto tendrá las siguientes fases:

- **(Preparación):** Fase precursora del proyecto. En ella se realiza la planificación de cada fase y sus tareas, además de preparar todo el material necesario para su realización.
- **Análisis:** En esta fase se define la funcionalidad que ofrecerá el plugin. Aquí se incluyen el análisis de requisitos, casos de uso, modelo de dominio y modelo Entidad-Relación.
- **Diseño:** En esta fase se utilizan los modelos de información generados en la fase de Análisis para definir los elementos software necesarios en la implementación, así como su estructura.
- **Implementación:** Programación del software. En este caso trata sobre la construcción del plugin.
- **Pruebas y verificación:** Diseño y puesta en marcha de casos de prueba que verifiquen el cumplimiento de los requisitos y casos de uso definidos en la fase de análisis.
- **Mantenimiento:** Corrección de errores encontrados tras el despliegue y/o implementación de mejoras que puedan surgir en un futuro. En este caso se realizará un manual donde se explicará detalladamente el proceso de instalación y utilización del plugin, así como la estructura del código fuente para su mejor comprensión.

Se ha escogido esta metodología debido a que, al ser una metodología secuencial y disponer de fases bien diferenciadas, permite una planificación y organización del proyecto más clara. Esto permite realizar una estimación de los costes del proyecto más exacta. Además, debido a la comprobación de lo desarrollado en cada fase, también permite evaluar el progreso del proyecto de forma más precisa.

2.3. Fases del proyecto inicial

En este apartado se explica el desglose en fases y tareas, y la temporalización del proyecto inicial. De cada tarea se mencionan sus predecesoras, duración estimada y una breve descripción.

Preparación del proyecto

En esta fase se realiza la planificación del proyecto, temporalización de cada fase y sus tareas, y el estudio e instalación del material y/o herramientas necesarias para su realización.

ID: 01	**Planificación del proyecto**
Predecesoras:	-
Duración:	5 días
Planificación y temporalización de las fases y sus tareas, elaboración de los planes de riesgo y análisis de coste del proyecto	

ID: 02	**Estudio de FOCA**
Predecesoras:	01
Duración:	7 días
Instalación y puesta en marcha de FOCA (y sus dependencias). Estudio de su documentación, esquema de datos y API de plugins.	

ID: 03	**Pruebas de uso de FOCA**
Predecesoras:	02
Duración:	3 días
Realización de pruebas con la herramienta para familiarizarse con ella	

ID: 04	**Estudio del servicio de Gmail**
Predecesoras:	03
Duración:	3 días
Estudio del funcionamiento del servicio de recuperación de contraseña de Gmail, qué datos expone y cómo extraerlos.	

ID: 05	**Memoria - Capítulos 1, 2 y 3**
Predecesoras:	04
Duración:	2 días
Preparación de los capítulos 1, 2 y 3 de la memoria (correspondientes a la Introducción, Plan de Proyecto y Preparación del Proyecto, respectivamente).	

Figura 2.2: Diagrama de Gantt - Preparación

Análisis

En esta fase del proyecto se especifica la funcionalidad que debe ofrecer el plugin. También se incluye el estudio de las herramientas necesarias para cumplir con dicha funcionalidad.

ID: 06	Análisis de requisitos
Predecesoras:	05
Duración:	2 días
Obtención de requisitos en base a los objetivos del proyecto.	

ID: 07	Estudio de herramientas
Predecesoras:	06
Duración:	8 días
Estudio de las herramientas y librerías disponibles para determinar cuál cumple mejor con los requisitos extraídos.	

ID: 08	Análisis de Casos de Uso
Predecesoras:	07
Duración:	2 días
Identificación y desarrollo de los distintos Casos de Uso a partir de los requisitos.	

ID: 09	Modelo de Dominio
Predecesoras:	08
Duración:	2 días
Análisis de los requisitos y casos de uso para extraer las clases de análisis necesarias y formar los diagramas pertinentes.	

ID: 10	Memoria - Capítulo 4
Predecesoras:	09
Duración:	1 día
Preparación del capítulo 4 de la memoria (Análisis).	

		Análisis	**15 días**	**vie 12/02/21**	**jue 04/03/21**	
		Toma de requisitos	2 días	vie 12/02/21	lun 15/02/21	6
		Estudio de herramientas	8 días	mar 16/02/2:	jue 25/02/21	8
		Análisis de CCUU	2 días	vie 26/02/21	lun 01/03/21	9
		Modelo de Dominio	2 días	mar 02/03/2:	mié 03/03/2:	10
		Memoria - Capitulo 4	1 día	jue 04/03/21	jue 04/03/21	11

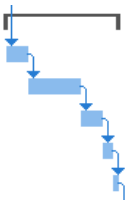


Figura 2.3: Diagrama de Gantt - Análisis

Diseño

Se especificará la arquitectura software a utilizar, diseño de la interfaz y desarrollo de los casos de uso en diseño.

ID: 11	Arquitectura del plugin
Predecesoras:	10
Duración:	2 días
Especificación de la arquitectura software a utilizar	

ID: 12	Diseño de la interfaz
Predecesoras:	11
Duración:	2 días
Diseño de la interfaz de usuario del plugin basándose en los requisitos y en los casos de uso.	

ID: 13	Casos de Uso en Diseño
Predecesoras:	12
Duración:	3 días
Desarrollar los Casos de Uso en análisis y realizar los correspondientes diagramas de secuencia en diseño.	

ID: 14	Diagrama de clases de Diseño
Predecesoras:	13
Duración:	1 día
A partir de las clases de diseño, realizar el diagrama de clases correspondiente.	

ID: 15	Memoria - Capítulo 5
Predecesoras:	14
Duración:	1 día
Preparación del capítulo 5 de la memoria (Diseño).	

	Diseño	10 días	vie 05/03/21	jue 18/03/21	
	Arquitectura del plugin	2 días	vie 05/03/21	lun 08/03/21	12
	Diseño de la interfaz	3 días	mar 09/03/2	jue 11/03/21	14
	Casos de Uso en Diseño	3 días	vie 12/03/21	mar 16/03/2	15
	Diagrama de clases de Diseñ	1 día	mié 17/03/2	mié 17/03/2	16
	Memoria - Capitulo 5	1 día	jue 18/03/21	jue 18/03/21	17

Figura 2.4: Diagrama de Gantt - Diseño

Implementación

En esta fase se aplicarán los análisis realizados en las anteriores fases para implementar la funcionalidad del plugin.

ID: 16	Esquema de Datos
Predecesoras:	15
Duración:	3 días
A partir de la información obtenida en la fase de diseño, implementar los esquemas de datos necesarios en la base de datos.	

ID: 17	Módulo de Acceso a Datos
Predecesoras:	16
Duración:	7 días
Implementar los métodos de conexión e interacción con la base de datos.	

ID: 18	Interfaz gráfica
Predecesoras:	17
Duración:	5 días
Implementación del diseño de interfaz gráfica realizado en la fase anterior.	

ID: 19	Módulo de Extracción
Predecesoras:	18
Duración:	14 días
Implementación de la funcionalidad encargada de extraer la información del servicio de Gmail.	

ID: 20	Memoria - Capítulo 6
Predecesoras:	19
Duración:	1 día
Preparación del capítulo 6 de la memoria (Implementación).	

Figura 2.5: Diagrama de Gantt - Implementación

Pruebas

En esta fase se comprueba que lo implementado en la fase anterior cumple con los requisitos y los casos de uso.

ID: 21	Realización de pruebas
Predecesoras:	20
Duración:	7 días
Realización de pruebas de funcionalidad del plugin y corrección de errores	

ID: 22	Memoria - Capítulo 7
Predecesoras:	21
Duración:	1 día
Preparación del capítulo 7 de la memoria (Pruebas)	

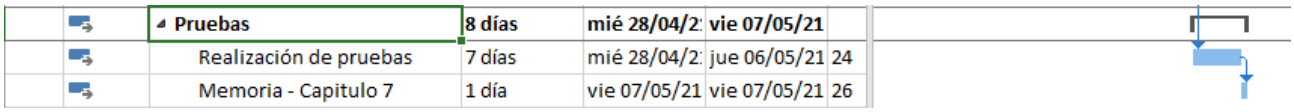


Figura 2.6: Diagrama de Gantt - Pruebas

Mantenimiento

Esta fase está dedicada a solucionar errores y/o implementar mejoras propuestas por el cliente. En este caso se realizará un manual del programador.

ID: 23	Manual del programador
Predecesoras:	22
Duración:	5 días
Redacción de un documento que explique la instalación, funcionamiento y estructura del código fuente para facilitar la comprensión y mantenimiento del plugin a terceros.	

ID: 24	Memoria - Capítulos 8 y 9
Predecesoras:	23
Duración:	3 días
Preparación de los capítulos 8 y 9 de la memoria (Mantenimiento y Conclusiones).	

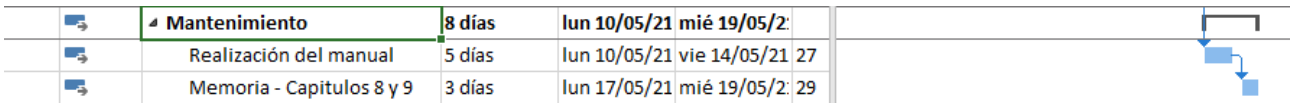


Figura 2.7: Diagrama de Gantt - Mantenimiento

2.3.1. Diagrama de Gantt inicial

El diagrama de Gantt resultante de la planificación del proyecto inicial. Muestra el desglose en tareas de cada fase y su temporalización. Iniciando el 18 de Enero, la fecha de finalización estimada es el 10 de Junio.

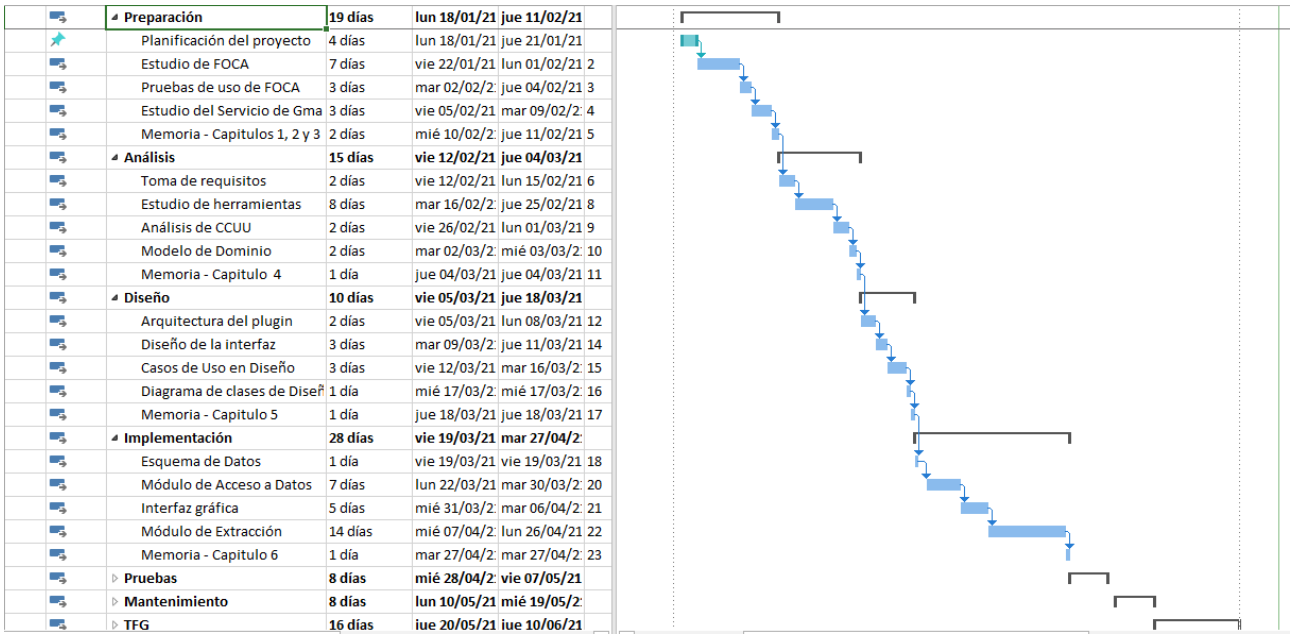


Figura 2.8: Diagrama de Gantt de la planificación inicial

2.4. Replanificación del proyecto

Como se ha mencionado anteriormente, el proyecto sufrió un cambio de objetivos que provocó la necesidad de volver a planificar el proyecto desde el principio.

La causa del cambio de objetivos fue un problema que surgió durante la fase de Implementación. Al programar el módulo de extracción (encargado de extraer la información del servicio de Gmail) la herramienta que se estaba utilizando no conseguía acceder al servicio debido al sistema de detección de software automatizado que Google tiene implementado. Tras probar con varias alternativas y no obtener resultados consistentes (el número de veces que era bloqueado era demasiado alto como para ser útil) se decidió cambiar los objetivos del proyecto a los mencionados en la introducción.

En esta sección se mostrará la nueva planificación del proyecto, partiendo de nuevo desde la fase de Análisis (no es necesario repetir la fase de planificación pues los recursos utilizados para el antiguo proyecto también son utilizados en el nuevo).

Análisis (II)

Definidos los nuevos objetivos, hay que repetir todo el proceso de desarrollo. En esta fase se repite el proceso de análisis funcional, desde la toma de requisitos hasta el modelado de clases.

ID: 01	**Selección de redes sociales**
Predecesoras:	-
Duración:	1 día
Se realiza una selección de redes sociales de entre las disponibles, basándose en su popularidad y utilidad	

ID: 02	**Redefinición de requisitos**
Predecesoras:	01
Duración:	1 día
Se vuelve a realizar la toma de requisitos en función de los nuevos objetivos.	

ID: 03	**Redefinición de Casos de Uso**
Predecesoras:	02
Duración:	1 día
Basándose en los nuevos requisitos, se identifican y desarrollan los nuevos casos de uso	

ID: 04	**Modelo de Dominio**
Predecesoras:	03
Duración:	1 día
Se identifican las nuevas clases de análisis	

ID: 05	**Memoria - Capítulo 4**
Predecesoras:	04
Duración:	1 día
Preparación del capítulo 4 de la memoria (Análisis)	

	Re-Analisis	5 días	vie 23/04/21	jue 29/04/21	
	Selección de RRSS	1 día	vie 23/04/21	vie 23/04/21	
	Redefinición de requisitos	1 día	lun 26/04/21	lun 26/04/21	2
	Redefinición de Casos de Uso	1 día	mar 27/04/2:	mar 27/04/2:	3
	Modelo de dominio	1 día	mié 28/04/2:	mié 28/04/2:	4
	Memoria - Capitulo 4	1 día	jue 29/04/21	jue 29/04/21	5

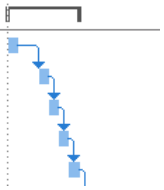


Figura 2.9: Diagrama de Gantt - Análisis (II)

Diseño (II)

Utilizando los datos sacados en la fase de Análisis, se diseñará una interfaz acorde y se desarrollarán los casos de uso en diseño.

ID: 06	**Arquitectura del plugin**
Predecesoras:	05
Duración:	2 días
Especificación de la arquitectura software a utilizar	

ID: 07	**Diseño de la interfaz**
Predecesoras:	06
Duración:	2 días
Diseño de la interfaz de usuario del plugin basándose en los requisitos y en los casos de uso.	

ID: 08	**Casos de Uso en Diseño**
Predecesoras:	07
Duración:	3 días
Desarrollar los Casos de Uso en diseño y realizar los correspondientes diagramas de secuencia en diseño.	

ID: 09	**Diagrama de clases de Diseño**
Predecesoras:	08
Duración:	1 día
A partir de las clases de diseño, realizar el diagrama de clases correspondiente.	

ID: 10	**Memoria - Capítulo 5**
Predecesoras:	09
Duración:	1 día
Preparación del capítulo 5 de la memoria (Diseño)	

	Re-Diseño	8 días	vie 30/04/21	mar 11/05/2:	
	Arquitectura del plugin	1 día	vie 30/04/21	vie 30/04/21	6
	Diseño de la interfaz	2 días	lun 03/05/21	mar 04/05/2:	8
	Casos de Uso en Diseño	3 días	mié 05/05/2:	vie 07/05/21	9
	Diagrama de clases de Diseño	1 día	lun 10/05/21	lun 10/05/21	10
	Memoria - Capitulo 5	1 día	mar 11/05/2:	mar 11/05/2:	11

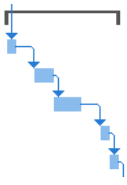


Figura 2.10: Diagrama de Gantt - Diseño (II)

Implementación (II)

En esta fase se realiza la implementación del plugin.

ID: 11	Módulo de Escáner
Predecesoras:	10
Duración:	4 días
Implementación de la funcionalidad del módulo encargado de recorrer las redes sociales buscando cuentas de usuario.	

ID: 12	Módulo de Acceso a Datos
Predecesoras:	11
Duración:	2 días
Implementación del módulo encargado de la interacción con la base de datos.	

ID: 13	Interfaz gráfica
Predecesoras:	12
Duración:	2 días
Implementación de la interfaz diseñada en la fase de Diseño.	

ID: 14	Memoria - Capítulo 6
Predecesoras:	13
Duración:	1 día
Preparación del capítulo 6 de la memoria (Implementación)	

Figura 2.11: Diagrama de Gantt - Implementación (II)

Pruebas (II)

En esta fase se comprueba que lo implementado en la fase anterior cumple con los requisitos y los casos de uso.

ID: 15	Realización de pruebas
Predecesoras:	14
Duración:	2 días
Realización de pruebas de funcionalidad del plugin y corrección de errores	

ID: 16	Memoria - Capítulo 7
Predecesoras:	15
Duración:	1 día
Preparación del capítulo 7 de la memoria (Pruebas)	

	Pruebas	3 días	lun 24/05/21	mié 26/05/2	
	Realización de pruebas	2 días	lun 24/05/21	mar 25/05/2	17
	Memoria - Capitulo 7	1 día	mié 26/05/2	mié 26/05/2	19

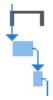


Figura 2.12: Diagrama de Gantt - Pruebas (II)

Mantenimiento (II)

Esta fase está dedicada a solucionar errores y/o implementar mejoras propuestas por el cliente. En este caso se realizará un manual del programador.

ID: 17	Manual del programador
Predecesoras:	16
Duración:	3 días
Redacción de un documento que explique la instalación, funcionamiento y estructura del código fuente para facilitar la comprensión y mantenimiento del plugin a terceros.	

ID: 18	Memoria - Capítulos 8 y 9
Predecesoras:	17
Duración:	1 día
Preparación del capítulos 8 y 9 de la memoria (Mantenimiento y Conclusiones).	

	Mantenimiento	4 días	jue 27/05/21	mar 01/06/2	
	Manual del programador	3 días	jue 27/05/21	lun 31/05/21	20
	Memoria - Capitulos 8 y 9	1 día	mar 01/06/2	mar 01/06/2	22

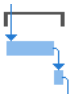
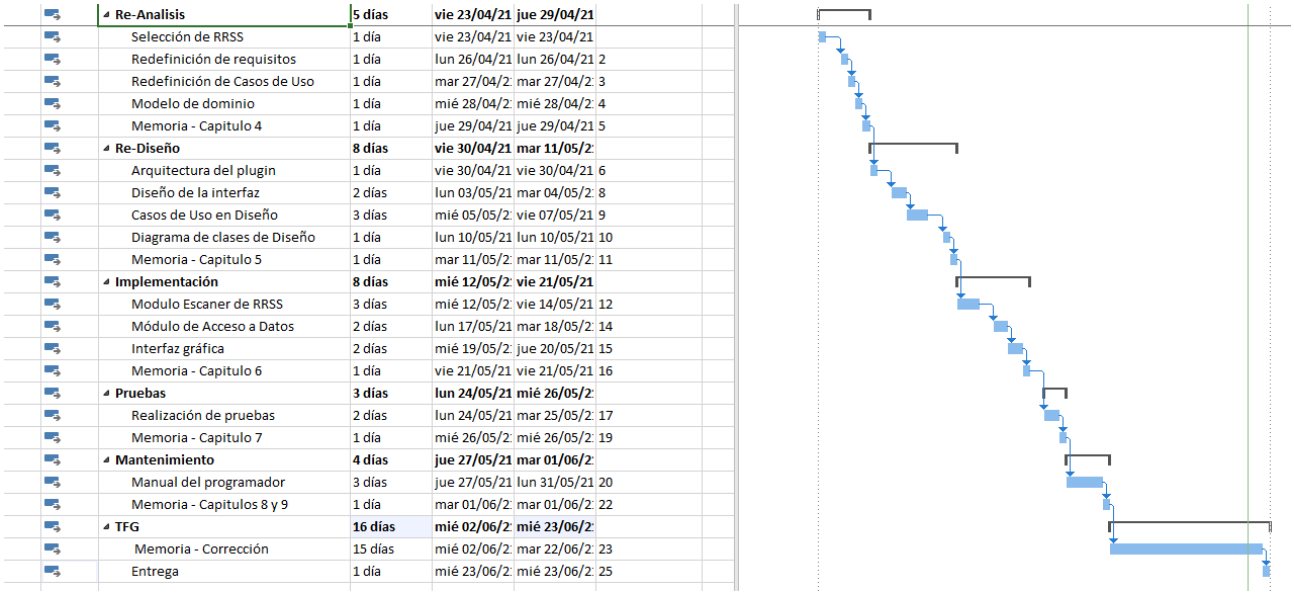


Figura 2.13: Diagrama de Gantt - Mantenimiento (II)

2.4.1. Diagrama de Gantt replanificado

El diagrama de Gantt resultante de la replanificación del proyecto provocada por el cambio de objetivos. Muestra la temporalización de las tareas del nuevo proyecto. Iniciando el 23 de Abril (fecha de la replanificación), se estima que el proyecto finalizará el 23 de Junio.



Re-Analisis	5 días	vie 23/04/21	jue 29/04/21	
Selección de RRSS	1 día	vie 23/04/21	vie 23/04/21	
Redefinición de requisitos	1 día	lun 26/04/21	lun 26/04/21	2
Redefinición de Casos de Uso	1 día	mar 27/04/2	mar 27/04/2	3
Modelo de dominio	1 día	mié 28/04/2	mié 28/04/2	4
Memoria - Capitulo 4	1 día	jue 29/04/21	jue 29/04/21	5
Re-Diseño	8 días	vie 30/04/21	mar 11/05/2	
Arquitectura del plugin	1 día	vie 30/04/21	vie 30/04/21	6
Diseño de la interfaz	2 días	lun 03/05/21	mar 04/05/2	8
Casos de Uso en Diseño	3 días	mié 05/05/2	vie 07/05/21	9
Diagrama de clases de Diseño	1 día	lun 10/05/21	lun 10/05/21	10
Memoria - Capitulo 5	1 día	mar 11/05/2	mar 11/05/2	11
Implementación	8 días	mié 12/05/2	vie 21/05/21	
Modulo Escaner de RRSS	3 días	mié 12/05/2	vie 14/05/21	12
Módulo de Acceso a Datos	2 días	lun 17/05/21	mar 18/05/2	14
Interfaz gráfica	2 días	mié 19/05/2	jue 20/05/21	15
Memoria - Capitulo 6	1 día	vie 21/05/21	vie 21/05/21	16
Pruebas	3 días	lun 24/05/21	mié 26/05/2	
Realización de pruebas	2 días	lun 24/05/21	mar 25/05/2	17
Memoria - Capitulo 7	1 día	mié 26/05/2	mié 26/05/2	19
Mantenimiento	4 días	jue 27/05/21	mar 01/06/2	
Manual del programador	3 días	jue 27/05/21	lun 31/05/21	20
Memoria - Capitulos 8 y 9	1 día	mar 01/06/2	mar 01/06/2	22
TFG	16 días	mié 02/06/2	mié 23/06/2	
Memoria - Corrección	15 días	mié 02/06/2	mar 22/06/2	23
Entrega	1 día	mié 23/06/2	mié 23/06/2	25

Figura 2.14: Diagrama de Gantt de la replanificación

2.5. Plan de riesgos

En este apartado se exponen los distintos riesgos que podrían ocurrir durante la ejecución del proyecto. También, para cada uno de ellos se evalúa qué posibilidad de ocurrencia tiene y que impacto podría tener para el proyecto.

R01 - Pérdida de datos

Pérdida del trabajo realizado (código fuente, herramientas, documentación).

Impacto Muy alto

Probabilidad Bajo

Plan de mitigación Creación y comprobación de copias de seguridad periódicamente o utilizar un sistema de control de versiones.

Plan de contingencia Evaluación de pérdidas y replanificación del proyecto en casos en los que sea necesario.

R02 - Problema con el hardware de desarrollo

Indisposición del hardware de desarrollo, conllevando la pérdida del trabajo realizado.

Impacto Muy alto

Probabilidad Bajo

Plan de mitigación Disponer de un segundo equipo de trabajo, junto con copias de seguridad en la nube (o sistema de control de versiones) para poder acceder a ellas desde el otro equipo.

Plan de contingencia Evaluación de pérdidas y replanificación del proyecto en casos en los que sea necesario.

R03 - Indisposición del personal

El personal encargado del proyecto no puede realizar su función.

Impacto Alto

Probabilidad Bajo

Plan de mitigación Ninguna.

Plan de contingencia Replanificación del proyecto para compensar el tiempo perdido.

R04 - Fallo en la temporalización del proyecto

El proyecto no puede ser acabado antes del plazo de finalización.

Impacto Medio

Probabilidad Bajo

Plan de mitigación Ajustarse a las fechas de vencimiento de las tareas y replanificar el proyecto si es necesario.

Plan de contingencia Realizar un análisis de viabilidad del proyecto y proponer otra fecha de entrega si es posible.

R05 - Problema durante la implementación

El proyecto no puede cumplir con los requisitos debido a un problema durante la implementación.

Impacto Alto

Probabilidad Medio

Plan de mitigación Realización de pruebas rigurosas a la hora de evaluar las herramientas a utilizar en el proyecto.

Plan de contingencia Realizar un análisis de viabilidad del proyecto y un cambio de objetivos si es necesario.

R06 - Cambios de requisitos

Los requisitos sufren un cambio durante la ejecución del proyecto.

Impacto Medio

Probabilidad Bajo

Plan de mitigación Realización de un análisis riguroso de los requisitos y dejarlos especificados antes de iniciar el desarrollo.

Plan de contingencia Replanificar el proyecto teniendo en cuenta los nuevos requisitos, incluso realizar un análisis de viabilidad.

2.6. Coste del proyecto

Según la planificación, la duración estimada del proyecto son de 4 meses y 3 semanas. Teniendo en cuenta que el salario medio de un Ingeniero Informático son 25000€ anuales mas 8500€ de beneficios sociales, y que generalmente se trabajan 8h al día, 5 días a la semana, se estima que el coste de este proyecto causado por el sueldo del personal (una persona) durante las 300h de duración estimada del TFG serán 3441.80€.

En el caso de costes de infraestructuras y hardware utilizado, suponiendo que el empleado se encuentra en condiciones de tele-trabajo, que la empresa provee de equipos a los empleados para realizar su función laboral (el precio promedio de un equipo portátil de gama media es 550€) y que la empresa debe sufragar los gastos asociados al desarrollo de la actividad laboral [5] (el coste de luz promedio es de 0.1 €/kWh [2], con un consumo promedio de 0.275 kWh por parte de un equipo portátil [3]; y el coste promedio de Internet por hora es de 0.05 €/h), el precio del tele-trabajo de un empleado durante 300h es de 574.75€.

No hay gastos asociados a licencias (tanto FOCA como el resto de herramientas auxiliares son *Open Source* o gratuitas), por tanto el coste del proyecto serán 4016.55€.

Cuadro 2.1: Costes del proyecto

Recurso	Precio por hora	Precio total
Equipo portátil	-	550€
Ingeniero informático	11.50€	3441.80€
Luz	0.02€	8.25€
Internet	0.05€	15€
TOTAL		4016.55€

Capítulo 3

Preparación del proyecto

En la fase de Preparación se incluyen tanto el estudio como la instalación de las herramientas necesarias para la ejecución del proyecto. También se explica el proceso de toma de requisitos realizado para el desarrollo del plugin.

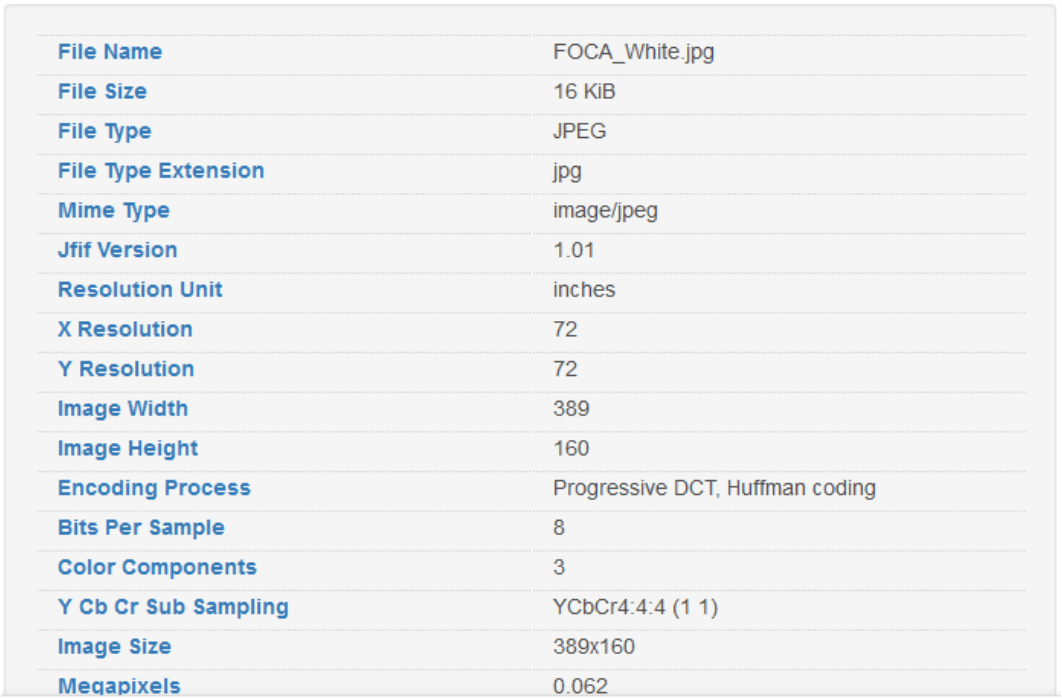
3.1. FOCA

Figura 3.1: Logotipo de la herramienta FOCA

FOCA (Fingerprinting Organisations with Collected Archives) es una herramienta *open-source* de extracción de metadatos encontrados en documentos ofimáticos desarrollada por ElevenPaths [6]. Es capaz de analizar documentos ofimáticos en formato Word, PDF, Open Office e imágenes, entre otros. Su principal utilidad es su capacidad de descargar y analizar todos los documentos dentro de un nombre de dominio de Internet dado. Esto es de gran importancia a la hora de auditar la seguridad de un dominio, pues es posible que haya información privada disponible para los posibles atacantes.

Antes de exponer las distintas funcionalidades que ofrece FOCA, primero hay que definir qué es un metadato: Según su etimología, un metadato es un "dato sobre un dato", refiriéndose a un fragmento de información que describe ciertas características de otro fragmento de información [7]. Actualmente, los metadatos se utilizan en todas partes, ya sea dentro del campo de la informática (fechas de creación y

modificación de un archivo, creador del archivo, permisos...) o fuera (géneros literarios de un libro, fechas de publicación, marca de un coche...). Estos datos se utilizan principalmente para facilitar la clasificación de los objetos a los que describen.



File Name	FOCA_White.jpg
File Size	16 KiB
File Type	JPEG
File Type Extension	jpg
Mime Type	image/jpeg
Jfif Version	1.01
Resolution Unit	inches
X Resolution	72
Y Resolution	72
Image Width	389
Image Height	160
Encoding Process	Progressive DCT, Huffman coding
Bits Per Sample	8
Color Components	3
Y Cb Cr Sub Sampling	YCbCr4:4:4 (1 1)
Image Size	389x160
Megapixels	0.062

Figura 3.2: Metadatos estándar de una imagen JPEG

La fuerza de FOCA como herramienta yace en la capacidad que tiene de extraer estos metadatos de los documentos y posteriormente analizarlos para relacionarlos entre sí y obtener una descripción general del dominio analizado en base a la información encontrada.

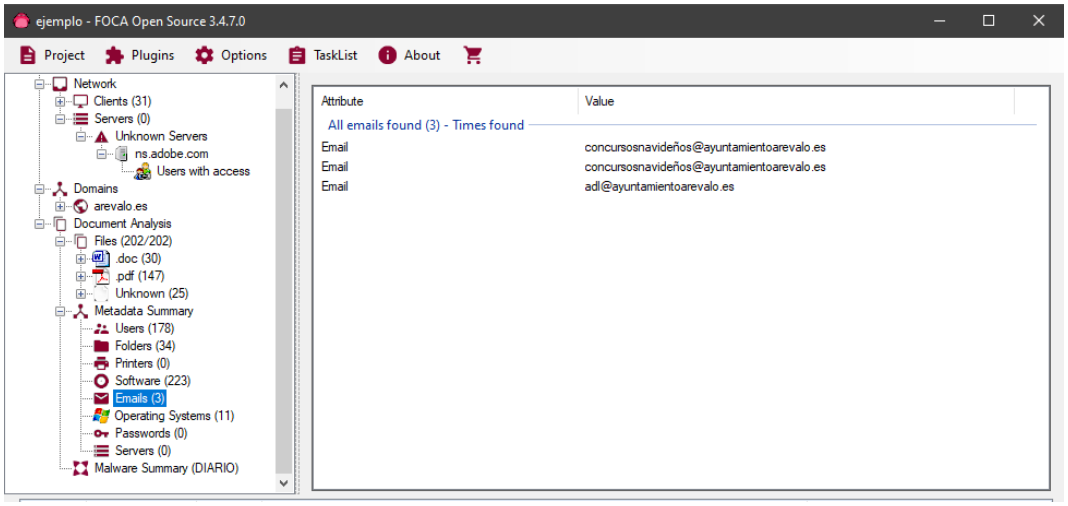


Figura 3.3: Ejemplo de extracción de metadatos de documentos en un dominio

Se pueden clasificar las funciones que ofrece esta herramienta en:

- **Búsqueda de documentos.** FOCA ofrece un conjunto de mecanismos de búsqueda automatizada de servidores dentro de un dominio, permitiendo recorrer recursivamente la red objetivo. Actualmente,

utiliza de las siguientes técnicas:

- Búsqueda Web: Descubrimiento de servidores en los servicios de búsqueda soportados (Google, Bing, DuckDuckGo) a partir de las URLs asociadas al dominio principal.
- Búsqueda DNS: Descubrimiento de servidores a partir de consultas a los registros DNS principales: NS (*Name server record*, un registro que indica qué servidor DNS consultar para resolver un nombre de dominio), MX (*Mail Exchange record*, indica qué servidores DNS se utilizan en el encaminamiento de correos electrónicos) y SPF (*Sender Policy Framework record*, un registro que indica qué servidores de correo electrónico están autorizados a enviar correos en nombre del dominio correspondiente).
- Resolución de Hosts: Resolver los nombres de host encontrados en los DNS para obtener sus direcciones IP.
- Escáner de registros PTR (también llamado consulta DNS inversa). Consultar los registros PTR para obtener los nombres de host de las IPs encontradas.
- Búsquedas de diccionario. Usar nombres utilizados comúnmente contra los servidores de DNS para obtener más servidores del dominio.
- Predicción DNS. En aquellos casos en los que se sospeche que los nombres de dominio siguen un patrón.
- Robtex. Un servicio online de análisis de dominios y direcciones IP.

- **Extracción de metadatos.** Permite extraer la información de los siguientes tipos de archivo:
  - PDF, INDDD (formato de estilo de documentos creado por Adobe InDesign).
  - Archivos de Office, OpenOffice y WordPerfect.
  - Imágenes (metadatos EXIF).
  - Archivos miscélaneos (XMP, OLE, RPD...)
- **Análisis de metadatos.** Procesado de los metadatos para formar una red de dispositivos encontrados en el dominio y su jerarquía.

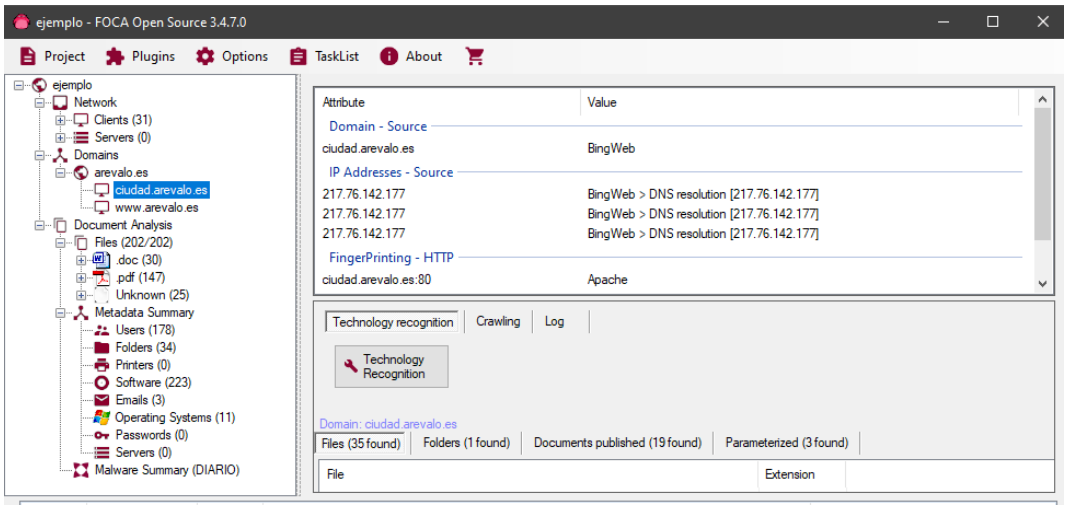


Figura 3.4: Ejemplo de análisis de red

3.1.1. Esquema de Datos

Respecto a su base de datos, FOCA toma una decisión de diseño muy característica. Normalmente las aplicaciones manejan su propia instancia de la base de datos de manera automática. En el caso de FOCA, se debe proporcionar una instancia ya creada de antemano. Esto permite a FOCA poder interactuar con bases de datos ya creadas anteriormente (ya sean locales como en red) o compartir instancias entre varias aplicaciones de FOCA.

Esta herramienta utiliza SQL Server como Sistema Gestor de Base de Datos, en específico una instancia ya configurada. Dentro de esa instancia FOCA genera su propio esquema de datos de manera automática (si no se ha generado anteriormente). La interacción con la base de datos se realiza mediante Entity Framework.

La parte del modelo de datos que nos interesa es todo lo relacionado con el almacenamiento de metadatos, en específico los correos electrónicos, de cara al plugin desarrollado en este proyecto. Teniendo el *connection string* de la instancia que se está utilizando, es posible consultar la información almacenada.

Respecto a este modelo de datos, los metadatos guardados se subdividen en 2 grupos, representados por sus respectivas tablas: *ComputersItems* y *MetaDatas*. Se explicará cada grupo por separado incluyendo el diagrama Entidad-Relación de cada parte (por facilitar la visualización de los diagramas, ya que el general es demasiado amplio) generado por SQL Server Management Studio (ver apartado 3.2.2).

En la tabla *ComputersItems* se referencian aquellos metadatos sobre los equipos encontrados (indicados con colores en la figura 3.5), estos son:

- Direcciones IPs (tabla *IPsItems*, en naranja).
- Usuarios (tabla *UserItems*, en verde).
- Contraseñas (tabla *PasswordItems*, en rosa).
- Dominios (tabla *DomainsItems*, en rojo).
- Impresoras (tabla *PrintersItems*, en amarillo).
- Rutas (tabla *PathItems*, en azul).
- Descripciones (tabla *DescriptionItems*, en morado).

En la tabla *MetaDatas* se recoge la información general sobre cada documento ofimático encontrado. En la tabla *MetaExtractors* se recogen los metadatos encontrados para cada documento guardado en la tabla *MetaDatas*, divididos por categoría de metadato. La información específica de cada categoría de metadato está guardada en su respectiva tabla (ver figura 3.6). Estas son:

- Usuarios (tabla *UserItems*, en rojo).
- Impresoras (tabla *PrintersItems*, en naranja).
- Correos electrónicos (tabla *EmailsItems*, en amarillo).

- Contraseñas (tabla *PasswordItems*, en morado).
- Historiales (tabla *HistoryItems*, en azul).
- Servidores (tabla *ServerItems*, en verde).
- Información de versiones (tabla *OldVersionItems*, en rosa).

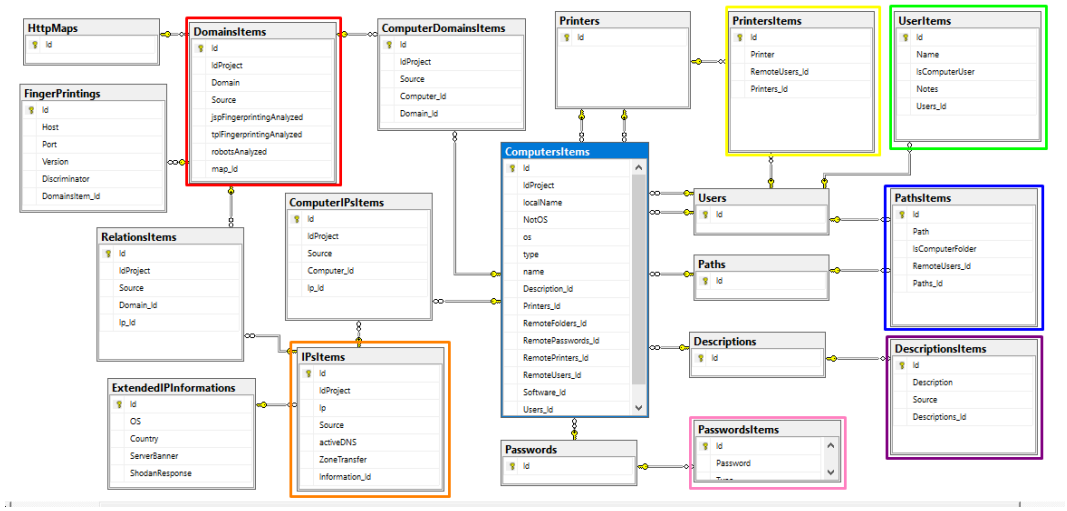


Figura 3.5: Esquema de la base de datos de FOCA (I). Obtenido con SSMS (ver apartado 3.2.2).

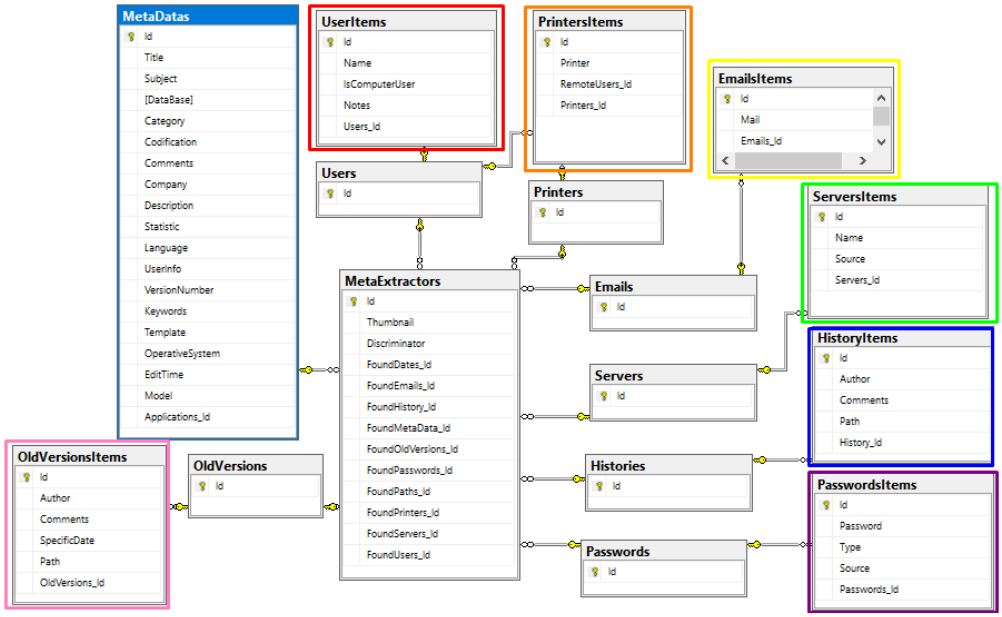


Figura 3.6: Esquema de la base de datos de FOCA (II). Obtenido con SSMS (ver apartado 3.2.2).

Se puede apreciar que hay categorías referenciadas tanto por *ComputersItems* como por *MetaExtractors*, esto es debido a que existen metadatos extraídos de documentos ofimáticos que se pueden vincular a equipos informáticos encontrados (usuarios, impresoras y contraseñas) durante el proceso de análisis de metadatos.

3.1.2. Código Fuente

En este apartado se explicará la estructura del código fuente en base a la función que desempeña. Este código está disponible en su repositorio GitHub [8].

Funcionalmente se puede dividir la aplicación en 5 bloques independientes:

- Acceso a datos. En este bloque se incluye toda la interacción con la base de datos.
- Búsqueda de subdominios. Este bloque se encarga del descubrimiento de nuevos subdominios bajo el dominio raíz y descargar sus documentos.
- Extracción y análisis de metadatos. Este bloque se encarga de la extracción de los metadatos en los documentos encontrados y su posterior análisis para formar la red de *hosts* del dominio raíz.
- Utilidades. Aquí se incluye el resto de recursos de la aplicación que no caen en las categorías anteriores, como por ejemplo interfaces, soporte de sistemas operativos, plugins, recursos visuales, etc.

Acceso a datos

Para implementar la funcionalidad de acceso a datos, FOCA utiliza Entity Framework. Entity Framework es un Object-Relational Mapper (ORM) disponible para Microsoft ADO.NET (incluido en .NET Framework). Un ORM permite gestionar una base de datos relacional utilizando la orientación a objetos de un lenguaje de programación (en el caso de Entity Framework, C#). Proporciona una capa de abstracción sobre la base de datos utilizada, ahorrando a la aplicación el tener que gestionar los detalles de la conexión, consultas, etc.

En el caso de FOCA, el código fuente dedicado a la gestión de la información persistente se encuentra en `./FOCA/Database`. Aquí se definen las entidades relacionales implementadas (`./FOCA/Database/Entities`), y las consultas que la aplicación puede realizar (`./FOCA/Database/Controllers`). Juntas definen la capa de abstracción implementada por Entity Framework.

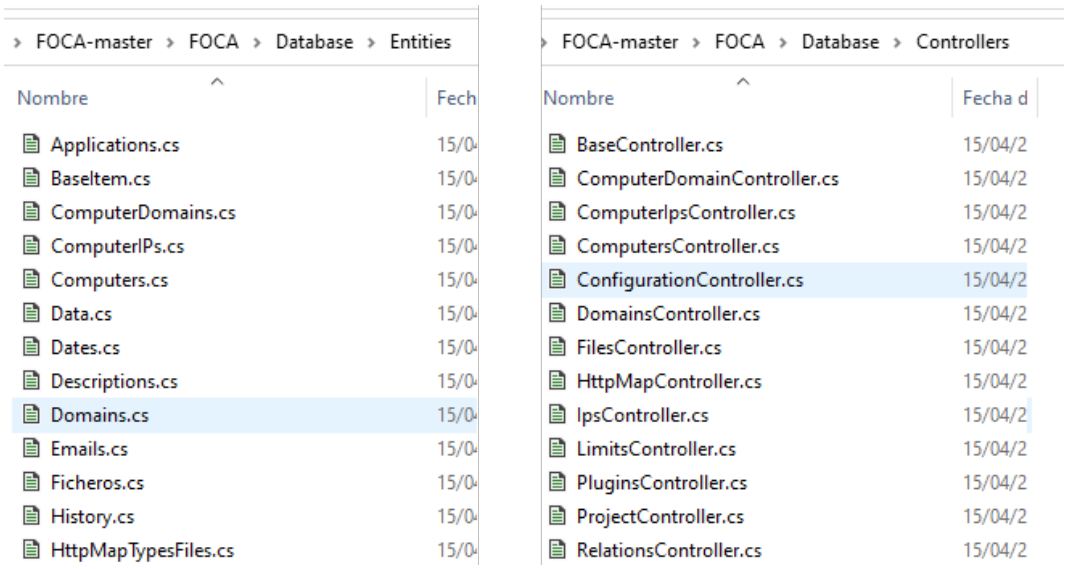


Figura 3.7: Entidades y controladores de Entity Framework

Búsqueda de documentos

Respecto a la búsqueda de subdominios y documentos, el código fuente se encuentra en un proyecto separado a FOCA, en `./SearcherCore`. Este proyecto se encarga de realizar las búsquedas de documentos

en los motores de búsqueda soportados. Estos motores de búsqueda disponen de sus propias APIs que permiten a distintas aplicaciones consumir sus servicios. Además de consumir esas APIs, FOCA también realiza una búsqueda convencional manipulando los parámetros GET de las URL de los buscadores y utilizando *web scraping*.

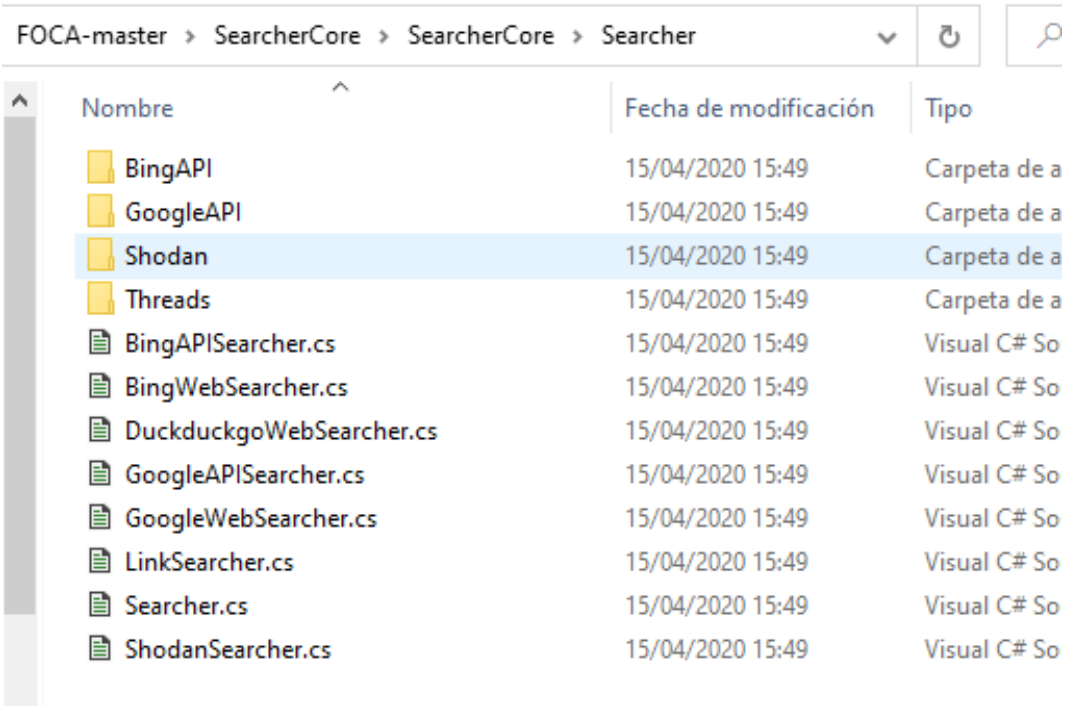


Figura 3.8: Contenido de `./SearcherCore`

Extracción y análisis de metadatos

En el caso de la extracción de metadatos, se encuentra también en un proyecto separado, en `./MetadataExtractCore`. Es el encargado de la extracción de los metadatos en cada documento soportado (mencionados anteriormente). Cada extensión soportada dispone de su propia clase en `./MetadataExtractCore/Metadata` que maneja la extracción de sus metadatos. También están recogidos en clases los distintos tipos de metadatos que FOCA extrae de los documentos (en `./MetadataExtractCore/Diagrams`).

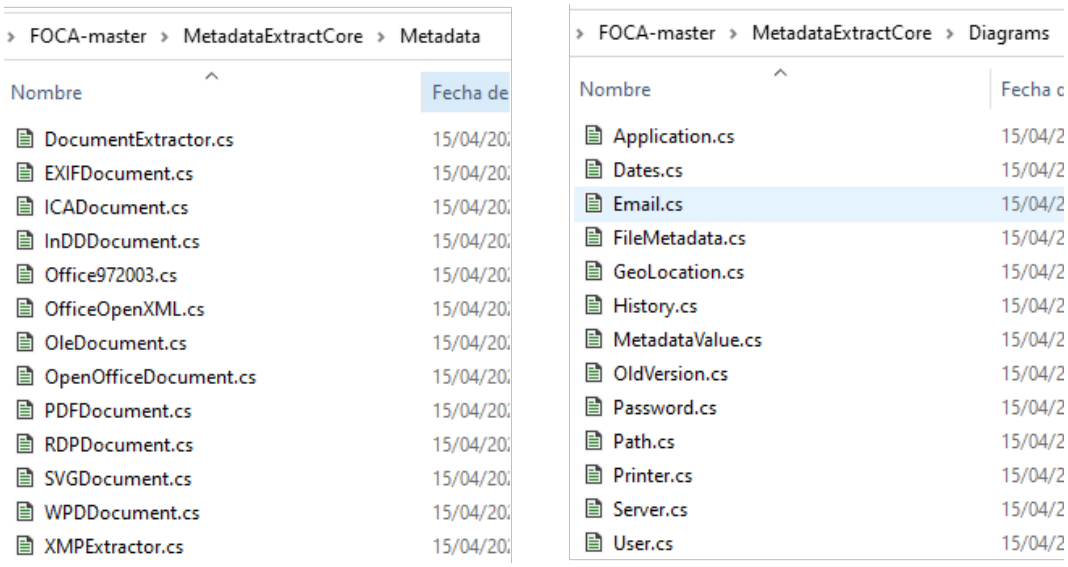


Figura 3.9: Contenido de `./MetadataExtractCore`

La parte de análisis de metadatos se encuentra en `./FOCA/Analysis` e incluye toda la funcionalidad para correlacionar los metadatos extraídos entre sí para obtener más información sobre el dominio (*Fingerprinting*, en Ciberseguridad). También incluye el análisis de malware de DIARIO.

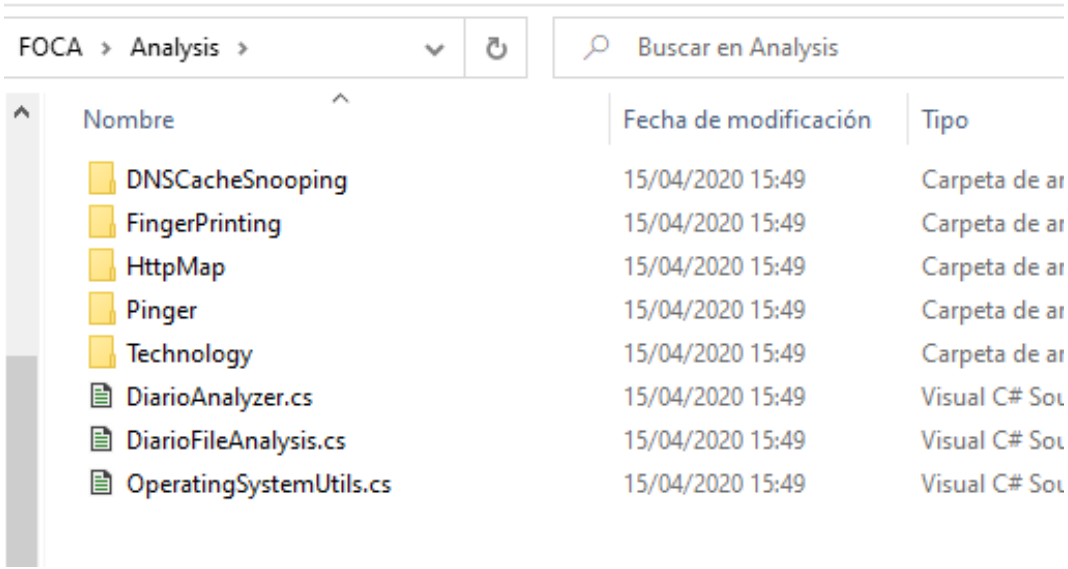


Figura 3.10: Contenido de `./FOCA/Analysis`

3.1.3. Instalación

Primero, hay que instalar las dependencias de la herramienta. FOCA esta construida sobre .NET Framework (versión 4.7.1), Microsoft Visual C++ (2010 o mayor) y utiliza SQL Server (2014 o mayor) como SGBD. El proceso de instalación de estas dependencias es descargarse los instaladores de las páginas oficiales y ejecutarlos, pero pueden ocurrir problemas durante la instalación debido a que la mayoría de usuarios ya tienen alguna versión de estos instalada, instalaciones defectuosas, archivos corruptos, etc (Los posibles errores de instalación están explicados en los apartados respectivos de cada dependencia).

El proceso de instalación de FOCA en sí es bastante sencillo. Es un proyecto *open source* disponible en el repositorio GitHub de ElevenPaths [8], por lo que simplemente hay que clonar el repositorio, compilar

el código fuente (ya viene con los binarios compilados, pero es recomendable recompilar el proyecto localmente) y ejecutarlo.

El repositorio también dispone de la solución de Visual Studio, por lo que para la compilación y desarrollo es muy recomendable usar este IDE. Es recomendable también recompilar el proyecto localmente para asegurar la compatibilidad entre las distintas herramientas y módulos utilizados por FOCA..

Al iniciar la herramienta por primera vez, FOCA buscará una instancia local de SQL Server local en el sistema. En caso de no encontrarla, el usuario tendrá que proveer el *connection string* de una instancia SQL Server para que FOCA cree su esquema de datos en ella. El *connection string* proporcionado será utilizado en las siguientes ejecuciones de la herramienta por defecto (guardado en `./bin/Release/FOCA.exe.Config`).

3.1.4. API de Plugins

La API de Plugins es un servicio ofrecido por FOCA que permite la ampliación de funcionalidad de la herramienta mediante plugins. Estos plugins son librerías de enlace dinámico (DLL) que pueden ser añadidas a la herramienta en tiempo de ejecución. En el repositorio de FOCA vienen incluidos el código fuente de la API (con su solución de Visual Studio), varios ejemplos de plugins compilados e incluso ejemplos de código fuente para facilitar el desarrollo de los mismos.

La integración con FOCA es manejada por la API de varias maneras [9]:

- *Interfaz de usuario.* La API expone un componente personalizado de Windows Forms que el plugin podrá rellenar con los elementos de su interfaz gráfica. Este panel será mostrado por FOCA en el apartado de plugins de la interfaz. También permite mostrar la interfaz en una ventana separada de FOCA.
- *Importar objetos.* La API también permite enviar a FOCA datos encontrados por el plugin para analizarlos.
- *Eventos.* Dispone de un sistema de eventos a los que los plugins pueden suscribirse. Esto es útil en el caso de querer ampliar la funcionalidad de FOCA en el manejo de estos eventos.

La API define los siguientes elementos importables:

- PluginPanel: El panel con la interfaz del plugin mencionado anteriormente.
- PluginToolStripMenuItem: Elemento en el desplegable de Plugins de FOCA para cargar el panel.
- ContextMenus: Para añadir elementos a distintos menús contextuales de la aplicación. Incluye *endpoints* para añadir a cada menú por separado y uno global para añadir el elemento a todos los menús soportados.
- ImportElements: Los distintos tipos de dato que están soportados por FOCA junto con los *endpoints* para importarlos. Son:
  - Domain
  - IP
  - URL
  - Association Domain IP

- Computer
- Project
- AddBackUp
- AddProxy
- AddUser
- AddZoneTransfer

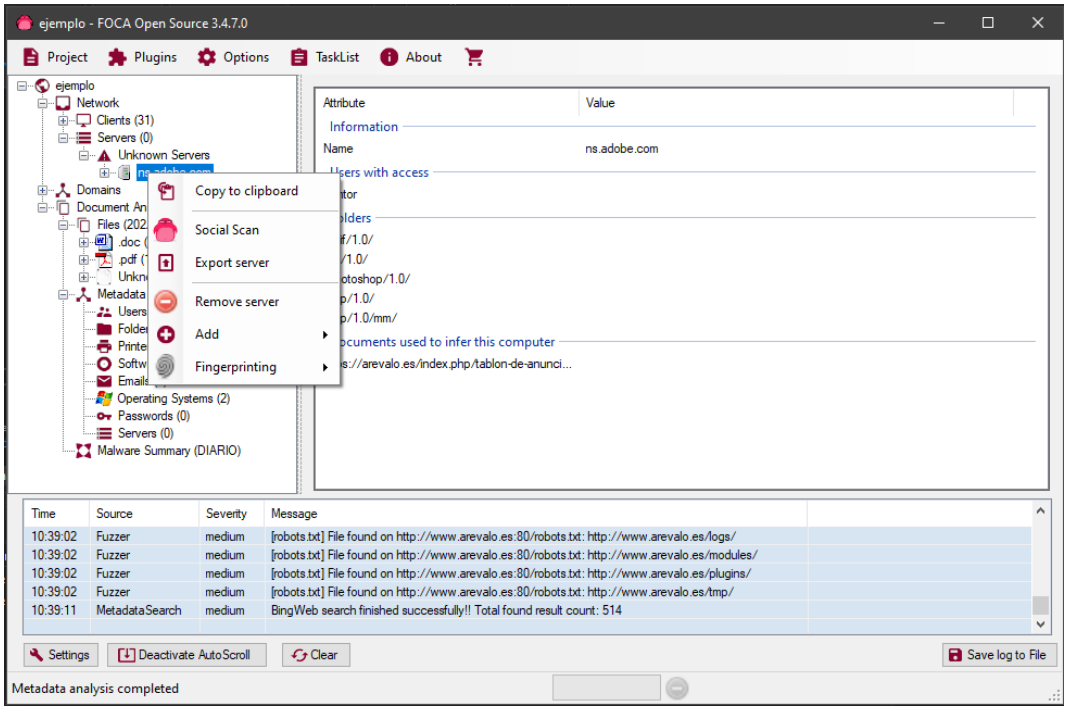


Figura 3.11: Ejemplo de plugin FOCA (Social Scan) en el menú contextual

Los plugins también pueden suscribirse a determinados eventos mantenidos por FOCA para recibir información cuando ocurren. Para suscribirse, hay que crear una función *void* con nombre igual al del evento deseado dentro de la clase *Plugin*. Los eventos soportados son:

- OnNewDomain
- OnNewURL
- OnNewIP
- OnNewProject
- OnNewRelation
- OnNewNetrange

3.2. SQL Server

SQL Server es un Gestor de Bases de Datos Relacional (RDBMS) desarrollado por Microsoft. Microsoft da soporte a muchos servicios de SQL Server (vía Microsoft Azure, *On-Premise*, para IoT, para desarrollo...), pero en este proyecto se utilizará la versión local. SQL Server 2016 Express es una versión de SQL Server empresarial gratuita dedicada a aplicaciones de pequeña envergadura que se instala localmente. Como FOCA soporta versiones a partir de SQL Server 2014, se decidió utilizar la versión 2016. En el caso

de no estar seguro de que opción escoger, en la página de SQL Server de Microsoft existe un ayudante que guiará a los usuarios por el proceso de selección.

Cuadro 3.1: Requisitos técnicos del sistema - SQL Server 2016 Express [12]

Característica	Mínimo	Recomendado
Sistema Operativo	Windows 8 o superior	Windows 10 o superior
Disco Duro	6 GB libres	8 GB libres
Memoria RAM	512 MB	1 GB
Velocidad de procesador	1.4 GHz	2 GHz
Tipo de procesador	x64	x64
.NET Framework	Versión 4.6	Versión 4.6

3.2.1. Instalación

Para la instalación y ejecución de FOCA, es necesario tener al menos la versión de SQL Server 2014. Para este proyecto se decidió utilizar la versión SQL Server 2016 Express. El primer paso en la instalación es comprobar si existe una versión ya instalada, y si no es la que se requiere, desinstalarla [16]. También puede ocurrir que la versión instalada de SQL Server esté corrupta, en cuyo caso es recomendable desinstalar. Esto se puede realizar desde Aplicaciones y Características, en el Panel de Control de Windows.

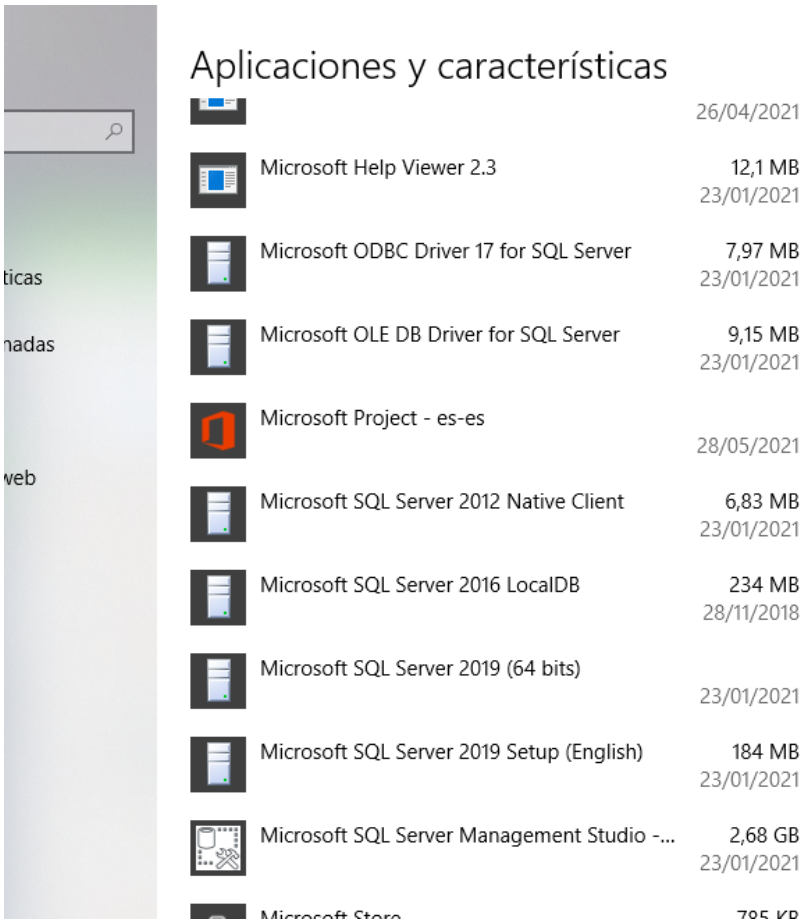


Figura 3.12: Aplicaciones y Características - Windows

Después de esto, se instala la nueva versión utilizando el instalador descargado de la página oficial [13].

A partir de ahí, un *Install Wizard* nos guiará en el proceso de instalación. Hay que indicar que se quiere una instalación de SQL Server independiente y que se usará una versión gratuita. Tras aceptar los acuerdos de licencia hay que elegir los componentes de SQL Server que se quiere instalar. En este caso se utilizará sólo el Motor de Bases de Datos, para el cual mostrará los requisitos del sistema necesarios. Por ultimo, hay que configurar la instancia de SQL Server (con nombre predeterminado o personalizado), servidor (credenciales de acceso, por defecto se usa la autenticación de Windows) y el Motor de Base de Datos (método de autenticación, cuentas de administración, memoria mínima y máxima disponible). Hay disponibles más páginas de configuración, pero estas dependen de los componentes que se vayan a instalar [14].

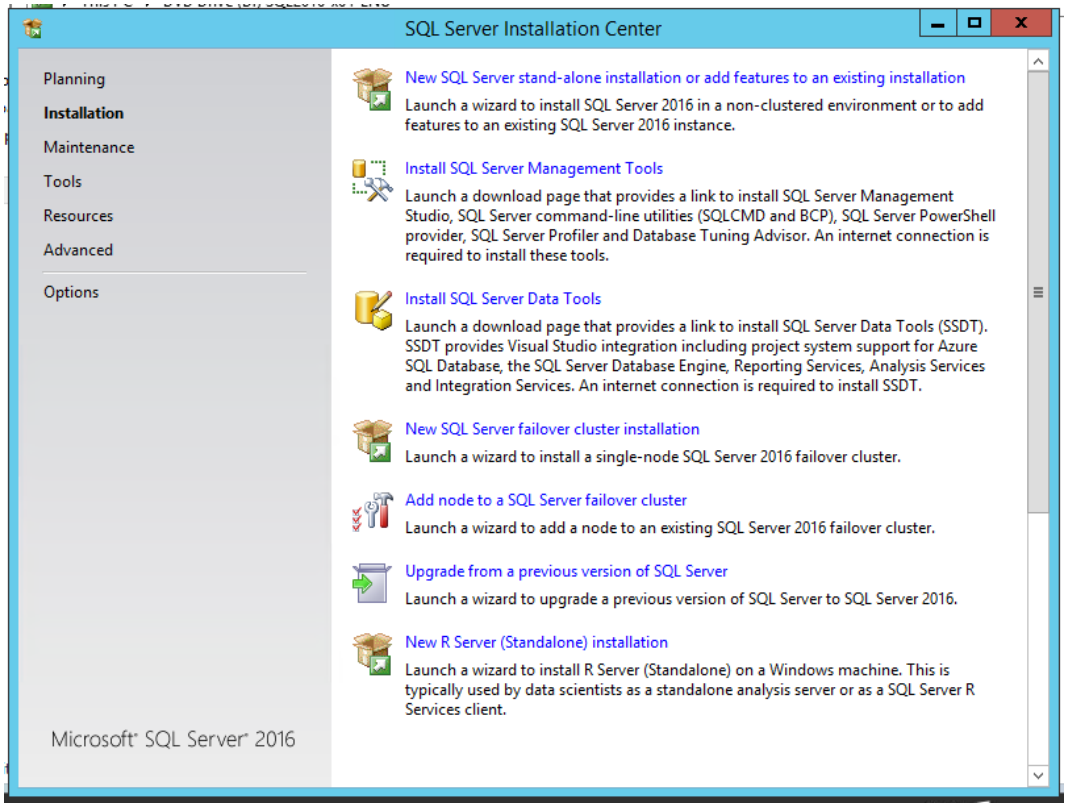


Figura 3.13: Instalador de SQL Server 2016

Una vez instalado se dispondrá de SQL Server en el sistema, con la instancia ya configurada. Para gestionar las instancias del sistema, SQL Server 2016 dispone de la herramienta SqlLocalDB [15].

3.2.2. SQL Server Management Studio

SQL Server Management Studio (SSMS) es un entorno de administración de instancias de SQL Server desarrollado por Microsoft [17]. Permite conectarse y gestionar instancias locales, desplegadas en un servidor, o de los servicios de Azure SQL Server en la nube. Esta herramienta solo está disponible para Windows, pero hay disponibles otras herramientas multiplataforma para gestión de instancias SQL Server como Azure Data Studio [18], que dispone de menos funcionalidades pero está disponible para Windows, Linux y macOS.

SSMS incluye todas las funciones típicas de un SGBD como crear bases de datos, tablas, vistas y realizar consultas. También permite generar diagramas Entidad-Relación de todas las tablas de una base de datos (o de una selección de ellas).

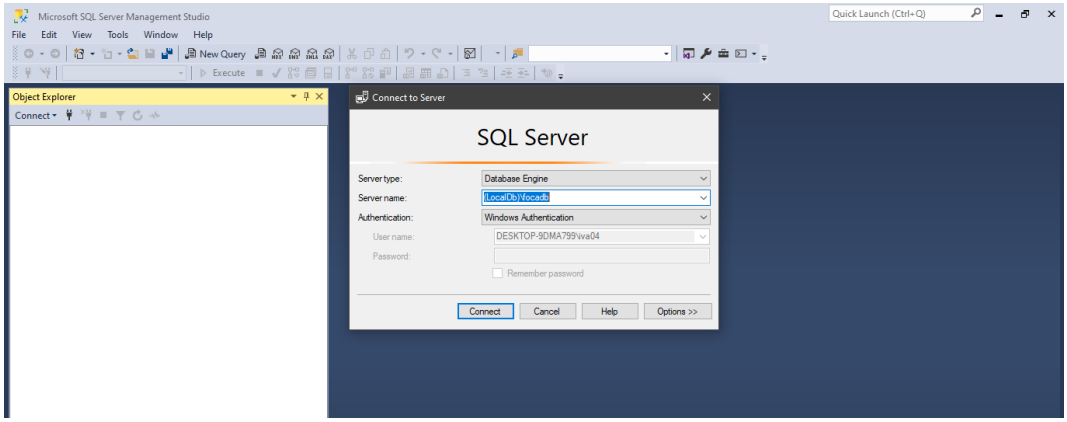


Figura 3.14: Conexión a una instancia - SQL Server Management Studio

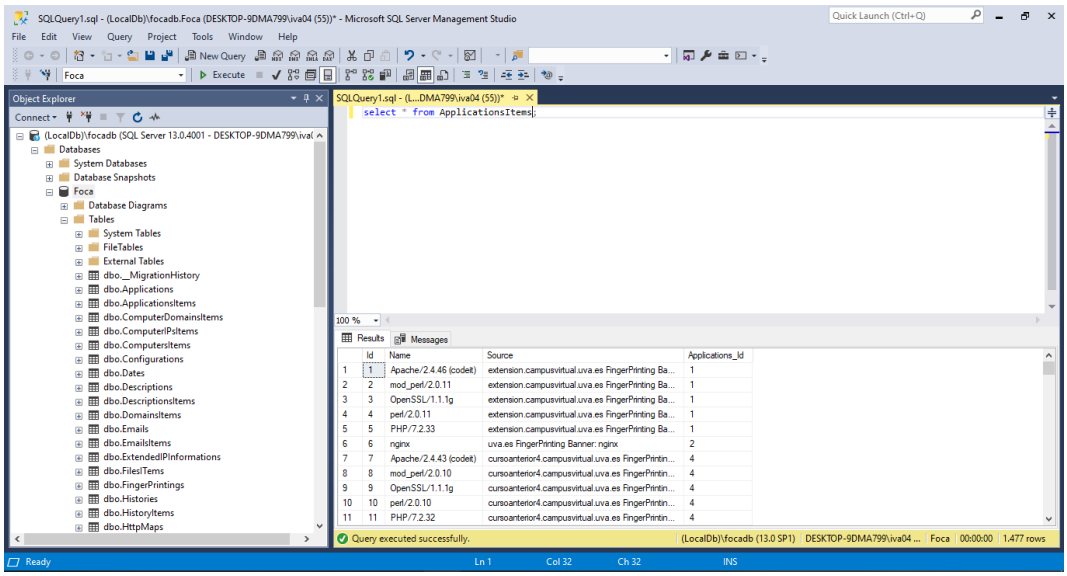


Figura 3.15: Interfaz de SQL Server Management Studio.

3.2.3. Instalación

El proceso de instalación sigue el mismo proceso que la instalación de SQL Server (y de la gran mayoría de herramientas disponibles para Windows): un instalador. En este caso las versiones anteriores no son un problema a la hora de instalar versiones nuevas, ya que es posible conservar ambas.

Al ejecutar el instalador, aparecerá la configuración de la instalación de SQL Server, donde se tendrá que indicar que se quiere añadir características a una instalación de SQL Server existente. Tras aceptar los términos de la licencia ofrecerá instalar actualizaciones de SQL Server (recomendable) para después elegir que características añadir (al ser el instalador de SSMS, éste viene seleccionado por defecto). Tras descargar los ficheros de configuración de SSMS necesarios, el instalador comprobará los requisitos de instalación (para asegurar que no habrá bloqueos durante la instalación) y la empezará. Al finalizar la instalación, la herramienta estará disponible en `C:/ProgramFiles(x86)/MicrosoftSQLServerManagementStudio18/Common7/IDE`.

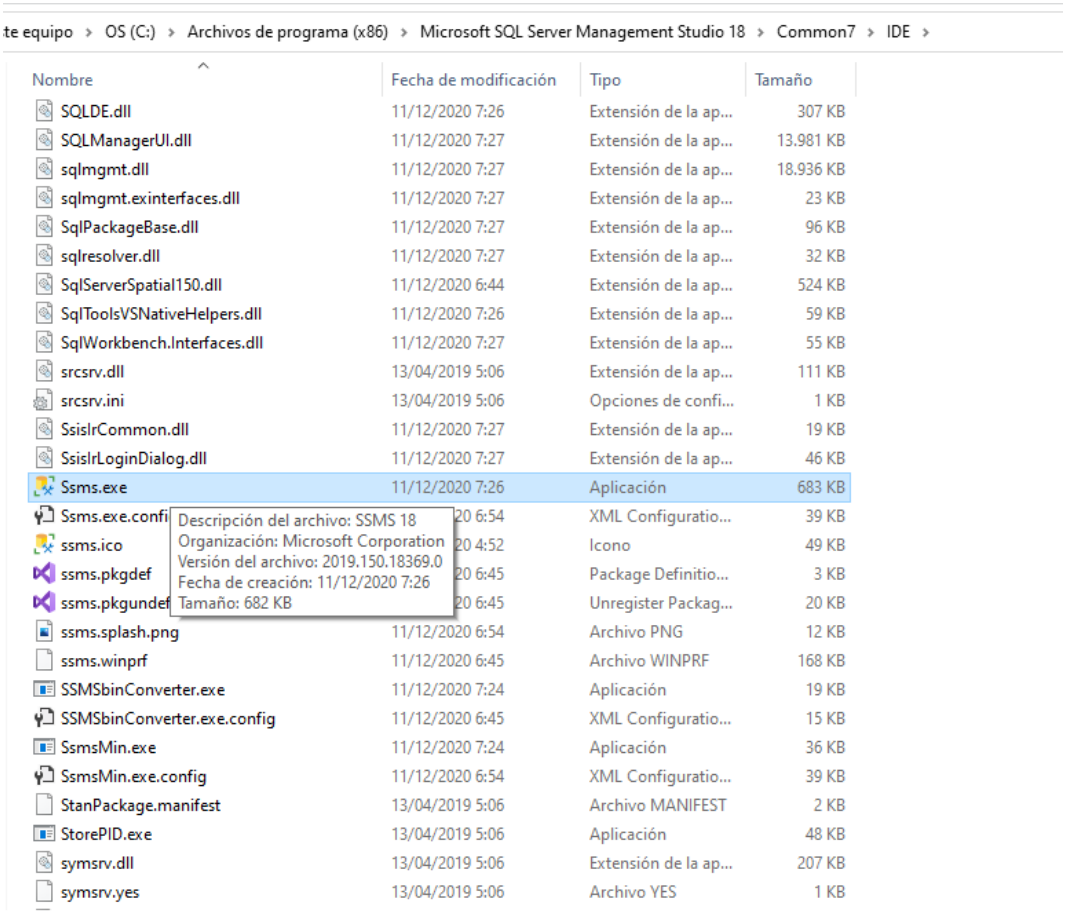


Figura 3.16: Localización del ejecutable de SQL Server Management Studio

Durante la instalación, es posible que el instalador informe de un error causado por una instalación anterior de SQL Server. Esto es debido a que esta instalación utiliza ciertas claves del Registro de Windows. Es posible que en algunas ocasiones el valor de estas claves no quede coherente con el estado del sistema. Esto suele ocurrir cuando se necesita un reinicio para completar la instalación. Entonces el usuario reinicia el ordenador, pero no se actualiza el registro, quedando la instalación incompleta. Una de las cosas que comprueba la instalación de SSMS es ese valor del Registro, impidiendo la instalación si no es el valor correcto.

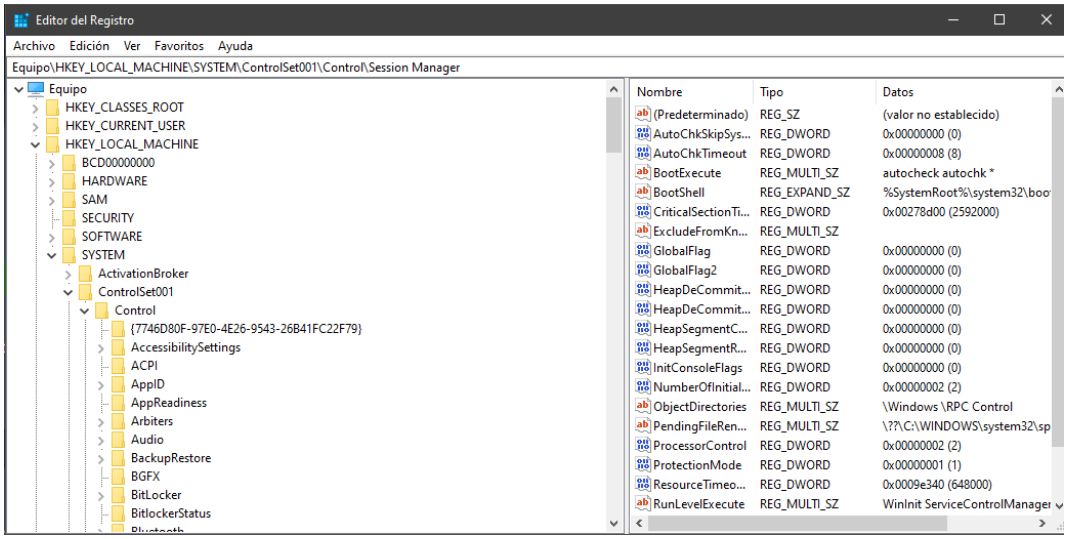


Figura 3.17: Herramienta de edición del Registro de Windows (*regedit*)

La solución de este problema es editar el Registro de Windows para volver a poner esos campos como corresponde [19]. Las rutas que contienen las claves en cuestión son:

- `HKEY_LOCAL_MACHINE/SYSTEM/CurrentControlSet/Control/SessionManager/`
- `HKEY_LOCAL_MACHINE/SYSTEM/ControlSet001/Control/SessionManager/`
- `HKEY_LOCAL_MACHINE/SYSTEM/ControlSet002/Control/SessionManager/`

El procedimiento a seguir es navegar hasta cada una de las rutas mencionadas anteriormente mediante la herramienta de edición del Registro de Windows (*regedit*), borrar la clave *PendingFileRenameOperations* (*Click* derecho en la clave y seleccionar *Eliminar*) y reiniciar el equipo.

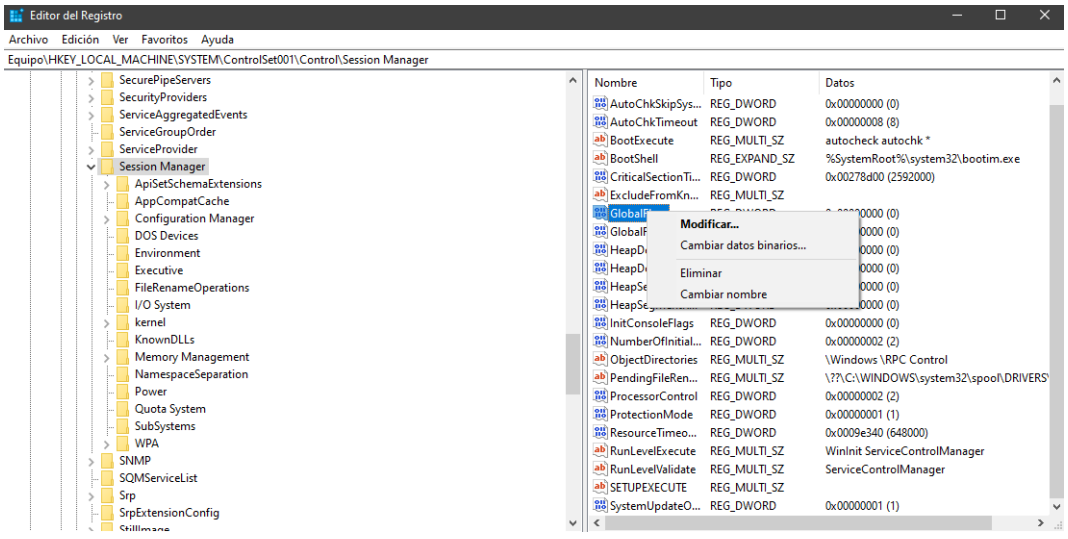


Figura 3.18: Eliminar una clave - *regedit*

3.3. Microsoft Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) desarrollado por Windows [23]. Ofrece soporte a gran cantidad de lenguajes de programación y tecnologías (como .NET Framework). Incluye amplias capacidades de desarrollo y despliegue, *debugging* y análisis de código fuente, gestión de casos de

prueba, soporte para control de versiones y funciones de colaboración e integración con Azure y otros servicios en la nube. En términos de IDEs, es uno de los más potentes a nivel de funcionalidad ofrecida (y consumo de recursos) de los disponibles. Esta herramienta es de pago, pero Microsoft ofrece versiones gratuitas (*Community*), aunque con funcionalidad reducida. Para este proyecto se utilizará este IDE ya que, además de ser el más recomendado para trabajar con .NET, en el repositorio de FOCA están disponibles también los archivos del proyecto de Visual Studio.

Cuadro 3.2: Requisitos técnicos del sistema - Visual Studio 2019 Community [24]

Característica	Mínimo	Recomendado
Sistema Operativo	Windows 7 o superior	Windows 10 o superior
Disco Duro	20 GB libres (SDD)	50 GB libres (SDD)
Memoria RAM	512 MB	1 GB
Velocidad de procesador	1.4 GHz	1.4 GHz
Tipo de procesador	x64	x64
Resolución de pantalla	720p	WXGA
.NET Framework	Versión 4.5.2	Versión 4.5.2

3.3.1. Instalación

Antes de instalar, es recomendable comprobar que el sistema dispone de los requisitos mínimos [24] necesarios para instalar y ejecutar Visual Studio. Para la instalación de Visual Studio es necesario disponer de .NET Framework 4.5.2 (esto es, para la ejecución del instalador, ya que durante el proceso de instalación se instalará la versión mas reciente de .NET Framework).

Comprobados todos los requisitos, se descarga y ejecuta el instalador [25]. Tras aceptar los acuerdos de licencia mostrará qué paquetes de funcionalidades están disponibles para instalar. Tener en cuenta que el espacio libre en el disco duro necesario depende directamente de cuantas características se quieren instalar (una instalación normal ocupa entre 20 GB y 50 GB, pero puede aumentar hasta 200 GB si añadimos más de la cuenta). Para este proyecto se necesita desarrollar una aplicación de escritorio con .NET, así que se elige esa opción. También se tiene la opción de instalar características por separado, lo cual es bastante útil pues FOCA requiere de .NET para ejecutarse, por lo que se seleccionará la versión de .NET que se desee (en este caso, FOCA requiere de versiones superiores a 4.7.1 por lo que se escogió la más reciente, 4.7.2). Por último, permitirá elegir qué paquetes de idiomas se quieren (opcional). Tras esto el instalador empezará con la descarga (el proceso puede ser bastante largo debido al alto volumen de datos a descargar e instalar).

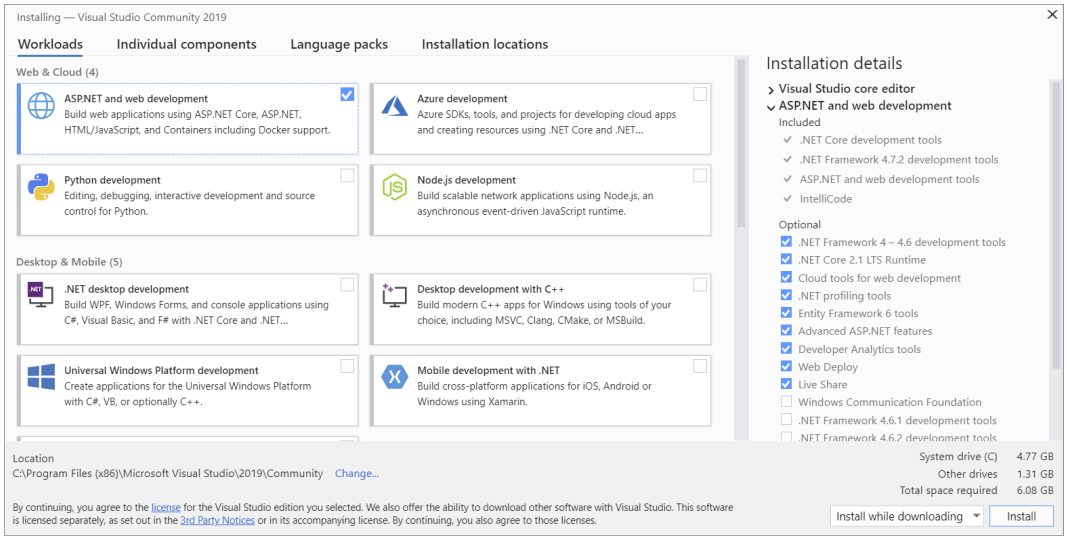


Figura 3.19: Instalación de Visual Studio 2019

3.4. Microsoft .NET Framework

Microsoft .NET es una estructura tecnológica y de software (un *framework*) que facilita el desarrollo de las aplicaciones y librerías construidas sobre él [20]. Se trata de un *framework* multiplataforma, disponible para la gran mayoría de versiones de Windows (XP en adelante), Linux y macOS. Incluye una amplia biblioteca de clases que manejan las distintas funcionalidades que soporta, como manejo de datos (ADO.NET), componentes Web (ASP.NET), integraciones con el sistema operativo (*Win32API*), etc. Mencionar también su soporte a Windows Forms y el ORM (Object-Relational Mapper) de ADO.NET, Entity Framework (ambos utilizados por FOCA).

FOCA requiere de una versión de .NET Framework superior a 4.7.1 y se aprovechará el instalador de Visual Studio, que te permite elegir qué versión de .NET Framework instalar (entre otros). Se escogerá la versión 4.7.2. Alternativamente, existen instaladores disponibles de esta versión en la página web de Microsoft [21].

3.5. Servicio de recuperación de contraseña de Gmail

Como se ha mencionado anteriormente, el objetivo inicial del proyecto es desarrollar un plugin capaz de recoger la información sobre cuentas de usuario expuesta por el servicio de recuperación de contraseña de Gmail.

En este apartado se explicará el funcionamiento de este servicio, qué información sobre las cuentas de los usuarios expone y qué métodos se han utilizado para intentar extraerla. También se expondrán los motivos que causaron el cambio de objetivos del proyecto, ya que está relacionado con los mecanismos de seguridad del propio servicio.

3.5.1. Funcionamiento

El servicio de recuperación de contraseña de Gmail es una funcionalidad añadida al inicio de sesión que permite a los usuarios recuperar la contraseña de su cuenta aportando información personal adicional para verificar su identidad.

Este servicio está basado en un mecanismo de *challenges* vía HTTP. El proceso de recuperación consta de un conjunto de preguntas de seguridad sobre tu cuenta sucediéndose una detrás de otra si no consigues responder.

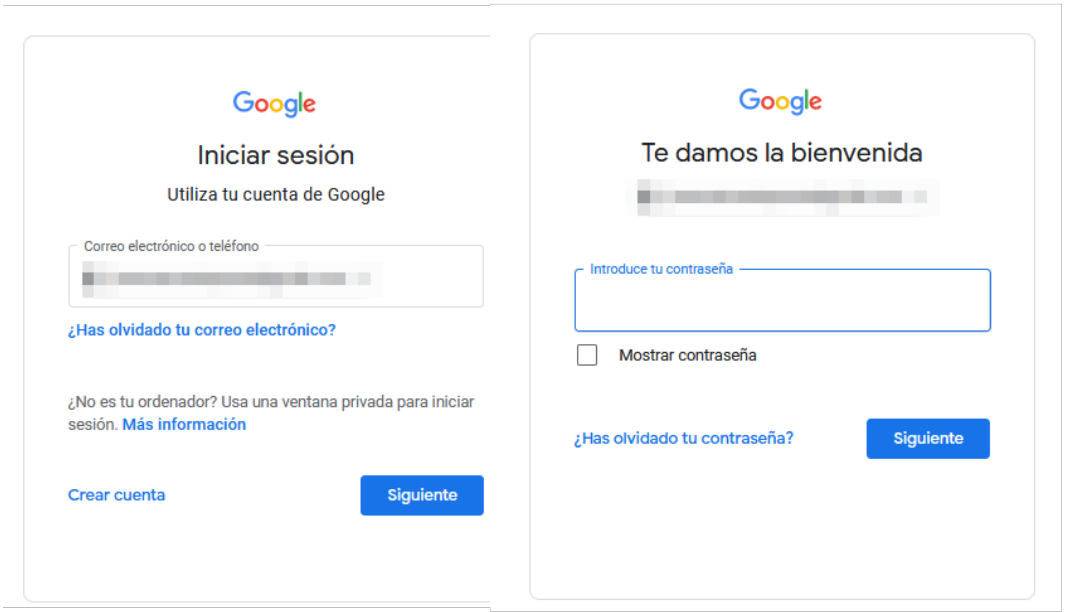


Figura 3.20: Inicio de sesión de Gmail

Este conjunto de preguntas es variable para cada correo electrónico pues depende de la cantidad de información personal contenida en la cuenta, y es esto lo que se pretende explotar con este proyecto. Este servicio depende en su totalidad de la cantidad de información personal que muestre al usuario para poder recuperar su cuenta. Mostrar muy poca información implicaría no poder autenticar al usuario, y demasiada información sería una filtración de datos personales demasiado grande.

De la información expuesta, se recogerán:

- Parte del correo de recuperación de la cuenta (solo muestran unos pocos caracteres)
- Estado de la cuenta (Si existe, ha sido desactivada o ha sido eliminada)
- Parte de los números de teléfono registrados en la cuenta (solo muestran unos pocos dígitos)
- Dispositivos vinculados a la cuenta.

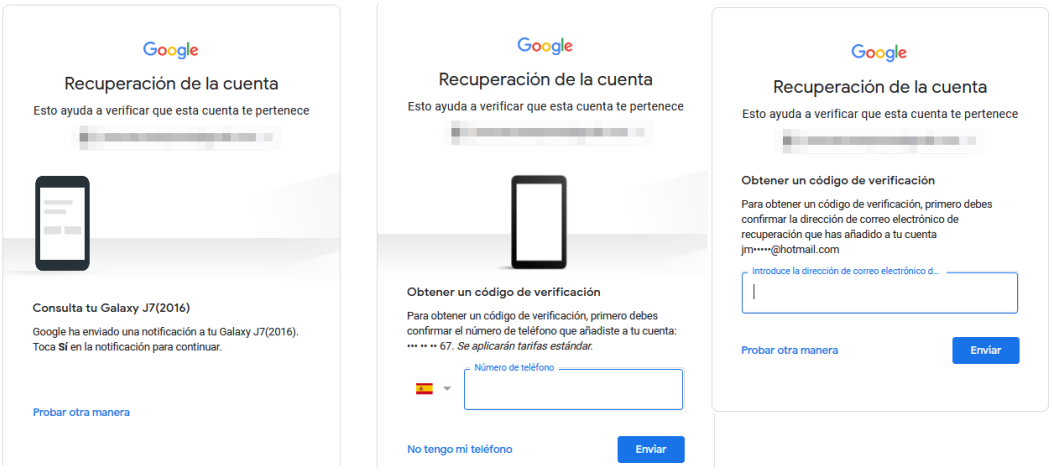


Figura 3.21: *Challenges* con información personal de los usuarios

Sobre la implementación técnica del servicio, es un flujo de vistas HTML basado en *cookies* y *tokens* de sesión. El proceso comienza en el servicio de inicio de sesión, tras introducir un correo electrónico y especificar que has olvidado la contraseña. En ese momento se genera un *token* de sesión que identificará el intento de recuperación de contraseña, y será utilizado durante todo el proceso (vía parámetros GET de las peticiones HTTP). Tras eso, iniciará el flujo de *challenges*, cada uno recibiendo todas las *cookies* del anterior y enviándoselas al siguiente. Entre las *cookies* utilizadas por el servicio de recuperación también se encuentran otras de servicios de Google como Google Analitics. Al final, si no se consiguió superar ningún *challenge* el flujo finaliza rechazando la solicitud de recuperación (por supuesto, al superar alguno de ellos se procede a la recuperación de la contraseña).

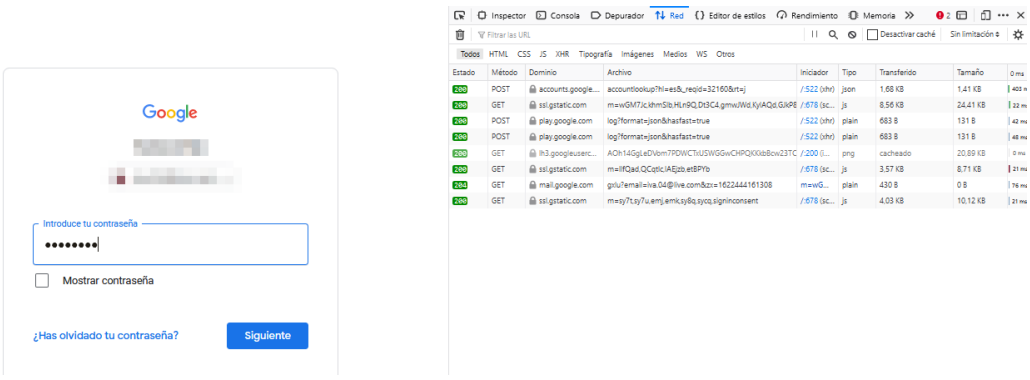


Figura 3.22: Peticiones HTTP realizadas al iniciar sesión.

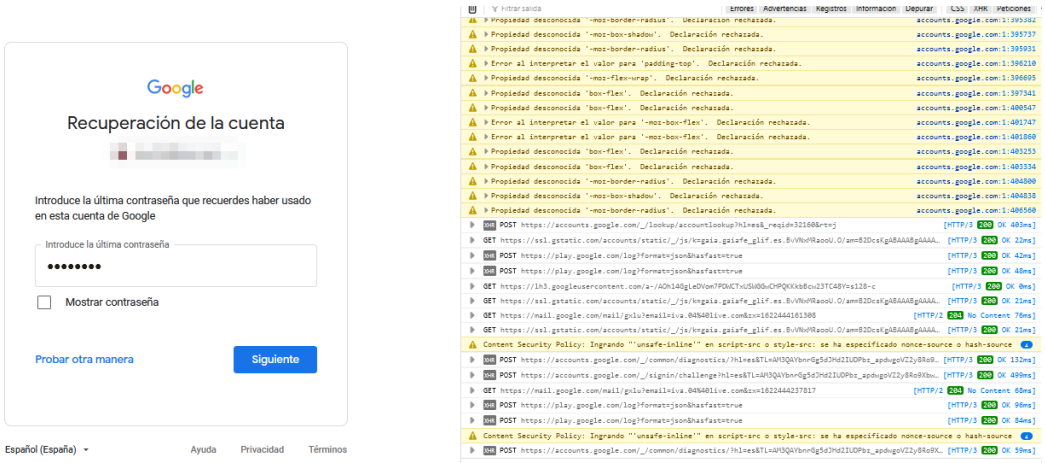


Figura 3.23: Peticiones HTTP realizadas al iniciar los *challenges*.

Debido a este intercambio de información tan denso, se decidió utilizar herramientas de *scraping* de alto nivel que manejan el tráfico de *cookies* y parámetros automáticamente.

3.5.2. Extracción de la información expuesta

La función clave que debe ofrecer el plugin es la capacidad de extraer la información personal expuesta en servicio de recuperación de contraseña de Gmail. Como se ha explicado anteriormente, es un servicio Web basado en *challenges* utilizados para verificar la identidad de los usuarios. Estas preguntas de seguridad exponen información sobre las cuentas de usuario para facilitar la autenticación del mismo (por ejemplo, enviándote un SMS al número de teléfono vinculado a la cuenta).

Al ser un servicio Web dedicado completamente a la interacción con usuarios reales (no está orientado a ser consumido por aplicaciones como otros servicios de Google como la búsqueda de Google, que dispone de una API), existen 2 maneras de interactuar con el y obtener la información que muestra de manera automática:

- *Web scraping*. Este es el enfoque estándar a la hora de recoger información de una página web. Enviar una petición HTTP al servidor Web destino, y procesar el HTML recibido.
- Herramientas de manejo de *Headless Browsers*. Como su propio nombre indica, son herramientas que utilizan navegadores sin interfaz gráfica, dedicados a ser utilizados por programas y no por usuarios finales.

Respecto al *Web scraping*, es bastante utilizado en la mayoría de casos de uso que necesitan obtener información de un sitio Web, en especial si ese servicio no dispone de una API con la que poder comunicarse. Además, es una operación bastante barata en términos de recursos y rendimiento (es sólo una petición HTTP). Pero esto conlleva una serie de desventajas. Al ser simplemente una petición, no realiza ningún tipo de ejecución del HTML recibido. Esto no es ningún problema si la información requerida se encuentra en el HTML base de la página, pero en bastantes ocasiones no es el caso. Con la propagación de las SPAs (*Single-page applications*) y las librerías de peticiones HTTP asíncronas (como AJAX [31] y el resto de herramientas construidas encima), la información no suele estar en el HTTP base (que es lo que envía el servidor Web a los navegadores y por extensión, a nuestro programa) sino que la envía el servidor en sucesivas peticiones realizadas tras cargar la página. Esto se consigue incluyendo en el HTML base el código para lanzar el resto de peticiones utilizando una librería como AJAX. Entonces un navegador, que

además de recibir el HTML lo ejecuta, lanza las peticiones sucesivas y obtiene la información. Utilizando *Web scraping*, se recibe el HTML pero no se ejecuta, impidiendo obtener la información deseada.

Una manera de solucionar esto sería que el programa maneje estas peticiones sucesivas por su cuenta, lo cual lleva a la siguiente desventaja: la complejidad. Existen muchos servicios Web que intercambian una gran cantidad de información con sus respectivos servidores, ya sea mediante peticiones asíncronas, parámetros en las peticiones, *cookies*, etc. Las librerías de *scraping* convencionales no manejan ninguno de estos intercambios, relegando la responsabilidad en los programas que las utilicen. Este es el caso del servicio de Gmail, cuyo flujo de *challenges* está basado en el intercambio de *cookies* y *tokens*, haciendo esta alternativa inviable.

Los *Headless Browsers* ofrecen las mismas posibilidades que el *scraping* sin sufrir de tales desventajas. Como su propio nombre indica, son herramientas que utilizan navegadores sin interfaz gráfica de usuario, pues están destinados a ser manejados en las aplicaciones mediante controladores. Al ser navegadores completamente funcionales, son capaces de ejecutar el código HTML que reciben y manejar el intercambio de *cookies* y parámetros de las peticiones, con la desventaja de ser más costosos a nivel de recursos y rendimiento. Existen disponibles varias librerías de *Headless browsers* disponibles para .NET, entre ellas Selenium y Puppeteer, que son las utilizadas en este proyecto.

Cambio de objetivos

El problema que causó el cambio de objetivos, fue la seguridad contra software automatizado específica del servicio de inicio de sesión de Google. Antes de iniciar el proceso de recuperación de contraseña, tras introducir una dirección de correo electrónico, el servicio comprueba si quien está interactuando con el servicio es un usuario final (una persona, lo que se espera) o se trata de software automatizado, en cuyo caso se rechaza inmediatamente.

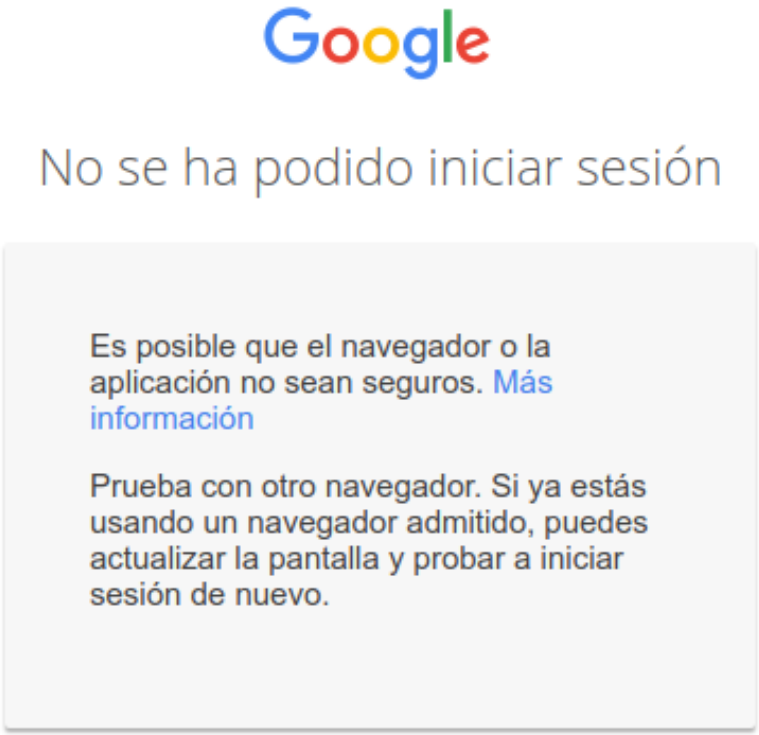


Figura 3.24: Pantalla de rechazo de *bots* de Google

En los siguientes apartados se describirán los intentos de evadir este bloqueo de seguridad.

Selenium

Selenium es un entorno de *software testing* para aplicaciones Web [32]. Permite la automatización de casos de prueba utilizando varios navegadores a través de sus respectivos controladores. Aunque esté dedicado al *testing*, puede ser utilizado para extraer información de sitios web gracias a las funcionalidades que ofrece la librería. Existe una versión para .NET en forma de paquete NuGet [33], el gestor de paquetes principal de dicho *framework*.

Teniendo los paquetes NuGet necesarios instalados, para navegar a una página web determinada, hay que instanciar el controlador que se desea usar (inicialmente Firefox) y llamar a la función *WebDriver.GoToUrl* pasándole la URL destino como parámetro.

```
var ffds = FirefoxDriverService.CreateDefaultService();
ffds.HideCommandPromptWindow = true;

IWebDriver driver = new FirefoxDriver(ffds, fopts);
driver.Navigate().GoToUrl(startUrl);
```

Figura 3.25: Visitar una página - Selenium Firefox

Inicialmente la preocupación principal fue que Google ni siquiera permitiese al software automatizado

alcanzar la página de inicio de sesión, pero al estar la comprobación de seguridad tras la primera fase del proceso de inicio de sesión (comprobación de correo electrónico) no hubo ningún problema. Fue al introducir el correo electrónico cuando el inicio de sesión fue rechazado.

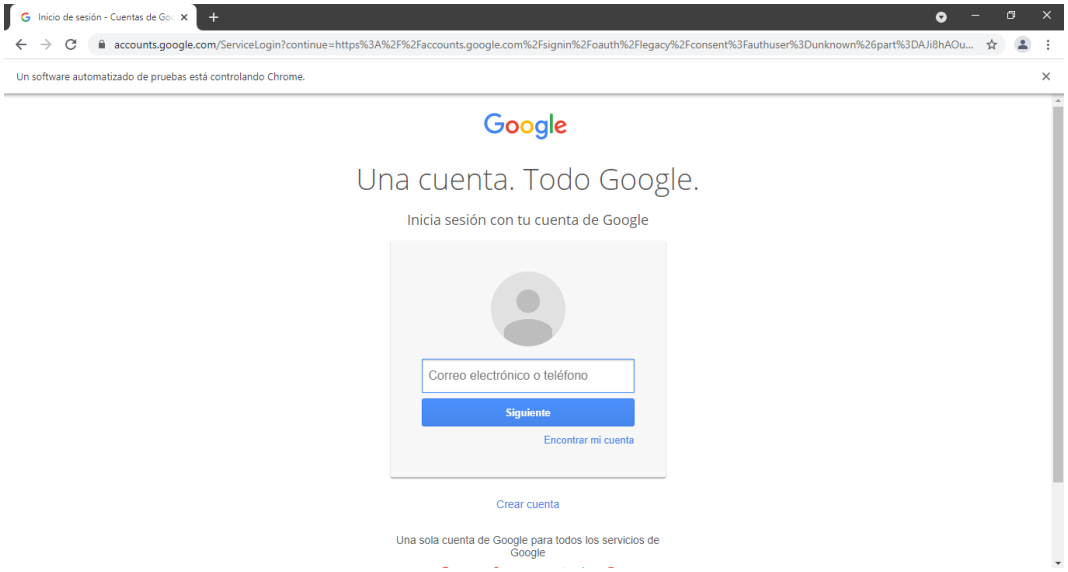


Figura 3.26: Inicio de sesión de Google con el controlador Chrome - Selenium

Para evitar este control de seguridad se han ido publicando varios métodos que lo conseguían, pero fueron arreglados por Google en su día. Se describen a continuación los métodos intentados:

- Cambiar el User-Agent utilizado.

  El controlador de Firefox permite la edición del perfil de usuario utilizado para editar propiedades como el *User-Agent*. Para ello hay que crear un perfil nuevo y editar las preferencias.

```
FirefoxProfile fp = new FirefoxProfile();
fp.SetPreference(
    "general.useragent.override",
    "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0"
);
```

Figura 3.27: Cambio de User-Agent - Selenium Firefox

- Utilizar el modo *headless*

  El controlador permite elegir al usuario si abrir el navegador controlado con Selenium en una ventana o sin interfaz gráfica. Para ello, hay que añadir el argumento *–headless* al controlador.

```
FirefoxOptions fopts = new FirefoxOptions();
fopts.AddArguments("--headless");
```

Figura 3.28: Modo headless - Selenium Firefox

- Introducir tiempos de pensar en el proceso de automatización para simular la actividad normal de un usuario.

Selenium soporta 2 tipos de esperas: espera implícita (esperar un tiempo determinado), y espera explícita (esperar a que ocurra un evento). Algunos sitios utilizan el tiempo que tarda el usuario en responder para determinar si se trata de un usuario realmente (el software automatizado es prácticamente instantáneo si no se controla esto).

```
//Espera implícita (esperar 10s)
driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);

//Espera explicita (esperar 10s a que cargue cierto elemento)
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(
    SeleniumExtras.WaitHelpers.ExpectedConditions.ElementExists(
        By.Id(target_id)
    )
);
```

Figura 3.29: Esperas - Selenium Firefox

- Probar desde distintas IP y con distintas cuentas de correo electrónico.

  Es posible que Google registre los intentos automatizados de inicio de sesión y utilice algún mecanismo de *blacklist* para bloquear los intentos de inicio de sesión de IPs reincidentes. Además, como la comprobación de seguridad se realiza tras introducir el correo electrónico de la cuenta objetivo, es posible que aplique un protocolo de seguridad mas estricto en aquellas cuentas en las que se intenta automatizar el inicio de sesión más frecuentemente.

- Utilizar el *WebDriver* de Chrome.

  Selenium también ofrece manejar el navegador de Google Chrome mediante un controlador.

```
IWebDriver driver = new ChromeDriver();
driver.Navigate().GoToUrl(startUrl);
```

Figura 3.30: Visitar una página - Selenium Chrome

- Acceder al servicio mediante los inicios de sesión alternativos de Google [35].

  En los inicios de sesión alternativos de Google (por ejemplo al iniciar sesión con Google desde Stack Overflow) se utilizaba un servicio más antiguo que el utilizado en los inicios de sesión directos desde cualquier aplicación de Google. Esto se reflejaba también en la seguridad, siendo la más antigua mucho menos estricta.

Puppeteer

Puppeteer es una librería de NodeJS desarrollada por Google que ofrece una API para interactuar con los navegadores de Chrome y Chromium [36]. Existe una versión *open source* de la librería [37], disponible en .NET como paquete NuGet [39]. Para este proyecto se utilizó la versión más reciente (PuppeteerSharp 4.0.0 en NuGet).

Con esta librería se intentaron las mismas configuraciones (soportadas) que con Selenium, sin éxito: inicio de sesión alternativo, falsear el *User-Agent* y usar el modo *Headless* [38].

```
//Iniciar el navegador
var browser = await Puppeteer.LaunchAsync(new LaunchOptions
{
    //Modo Headless
    Headless = true

});
var page = await browser.NewPageAsync();

//Visitar StackOverflow
await page.GoToAsync("https://stackoverflow.com/users/login");

//Falsificar User-Agent
await page.setUserAgent("Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0");
```

Figura 3.31: Configuración de Puppeteer Sharp

Selenium Stealth

Selenium Stealth es una versión de la librería de Selenium para Python con funcionalidad adicional dedicada a evadir controles de seguridad contra software automatizado [41]. Incluye la utilización de servidores proxy, utilización de *User-Agent* e información sobre el sistema (se envían en las peticiones HTTP) falsos. De las alternativas estudiadas es la única que ha mostrado resultados, consiguiendo evitar el bloqueo de Google, aunque de manera esporádica. La utilización de esta librería contra el servicio de Gmail no producía resultados con la consistencia necesaria como para ser de utilidad.

El escenario que mejor resultados ha producido fue utilizar el inicio de sesión alternativo de *StackOverflow*, falsificando el *User-Agent* y los datos sobre el sistema (idioma, sistema operativo, controladores gráficos, etc), e inserción de esperas en el proceso de automatización [40].

```
# Opciones del driver
options = webdriver.ChromeOptions()
options.add_argument("start-maximized")
if PROXY:
    options.add_argument('--proxy-server=http://%s' % PROXY)
options.add_experimental_option("excludeSwitches", ["enable-automation"])
options.add_experimental_option('useAutomationExtension', False)
options.add_experimental_option('excludeSwitches', ['enable-logging'])
driver = webdriver.Chrome(options=options, executable_path=PATH)

#Falsificación de parámetros
stealth(driver,
        user_agent='Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:87.0) Gecko/20100101 Firefox/87.0',
        languages=["en-US", "en"],
        vendor="Google Inc.",
        platform="Win32",
        webgl_vendor="Intel Inc.",
        renderer="Intel Iris OpenGL Engine",
        fix_hairline=True,
```

Figura 3.32: Configuración de Selenium Stealth

A pesar de conseguir evadir este control de seguridad algunas veces, no es suficiente como para ser considerado de utilidad (más de la mitad de los intentos son reconocidos como automatizados y bloqueados), además respecto al tema de protección contra software automatizado Google siempre está atento a las brechas que surgen en sus sistemas y al poco tiempo acaban arregladas (la brecha de seguridad utilizada fue publicada el 19 de Abril de 2021 [40]). En este caso, han añadido *captchas* que aparecen aleatoriamente para aumentar la tasa de fallo de aquellos *bots* que se basen en realizar numerosas peticiones contra su servicio.

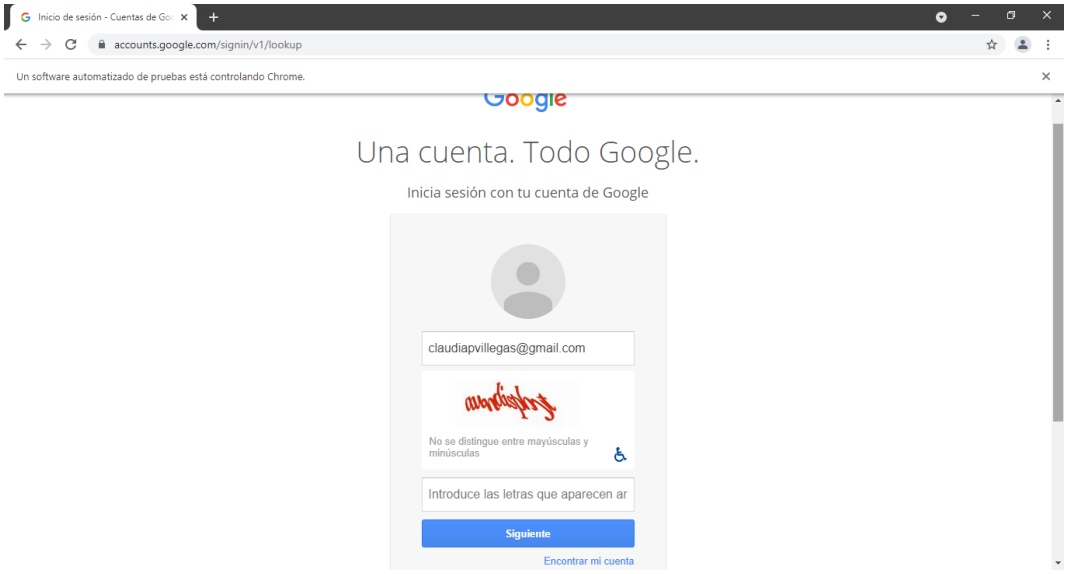


Figura 3.33: Captcha de inicio de sesión

3.6. Estudio de Redes Sociales

Tras el cambio de objetivos mencionado en el apartado 3.5, se replanteó la funcionalidad que deberá ofrecer el plugin (expuesta en la Introducción, ver apartado 1.2). Se deberán consultar los inicios de sesión de redes sociales para comprobar si una dirección de correo electrónico posee cuenta en esos servicios. En esta sección se detallarán los criterios utilizados para escoger qué servicios de redes sociales utilizar.

El conjunto de redes sociales y servicios utilizados tiene un gran impacto en la utilidad que tendrá la herramienta. Es necesario seleccionar aquellos servicios que sean lo suficientemente populares para que la información de sus usuarios sea valiosa. También es de interés cubrir la mayor cantidad de usuarios posibles, incluyendo aquellas redes sociales más extendidas entre la población adulta.

Según un estudio sobre los usuarios de redes sociales en España [42], las redes sociales más populares son:

- YouTube (gestionado por Google).
- Facebook.
- Instagram (propiedad de Facebook).
- Twitter.
- LinkedIn.
- Tumblr.

También se han añadido el servicio de Google, el de Amazon y el de Microsoft, que están muy extendidos entre los usuarios (YouTube utiliza las cuentas de Google como cuentas de usuario, y Microsoft gestiona los inicios de sesión de Windows y de Office, entre otros), además de servir como inicio de sesión alternativo en gran parte de las redes sociales.

Teniendo estos servicios, hay que determinar cuáles de ellos permiten a software automatizado realizar un intento de inicio de sesión. La característica que nos interesa aprovechar es que la gran mayoría de

servicios no impiden al software automatizado realizar el inicio de sesión. Algunos de ellos, como Google, realizan esa comprobación de seguridad después del inicio de sesión (pero si por ejemplo la cuenta no existe, te notifican de ello). Este es un problema de diseño de la seguridad del servicio: es posible que al servicio le interese dejar a los usuarios iniciar sesión mediante software automatizado (por ejemplo, para un programa que haga publicaciones automáticas en una red social), pero lo que no se debería permitir es que un programa compruebe si el usuario tiene una cuenta en ese sitio.

Del conjunto estudiado, LinkedIn es la única red social que protege su inicio de sesión contra ataques maliciosos de este tipo. Entonces, los servicios que permiten a software automatizado determinar si una dirección de correo electrónico tiene una cuenta en el sitio son:

- Google.
- Amazon.
- Facebook.
- Instagram.
- Twitter.
- Tumblr.
- Microsoft.

3.7. Toma de Requisitos

En este apartado se detallará el proceso de análisis de requisitos que se llevó a cabo sobre las premisas de funcionalidad del plugin. Estos requisitos serán utilizados como base del proceso de desarrollo.

3.7.1. Requisitos Funcionales

Los requisitos funcionales especifican que funciones y/o comportamientos deben de ser implementados en el sistema.

- RF-01: El plugin podrá comprobar si un correo electrónico tiene cuenta en las redes sociales objetivo.
- RF-02: El plugin podrá consultar los correos electrónicos guardados en FOCA para usarlos contra los servicios de inicio de sesión de las redes sociales objetivo.
- RF-03: El plugin dispondrá de un histórico exportable de las comprobaciones de redes sociales realizadas.

3.7.2. Requisitos no Funcionales

Los requisitos no funcionales recogen que características y/o restricciones deben de estar presentes en el sistema.

- RNF-01: El plugin estará desarrollado en C# (Microsoft .NET Framework).
- RNF-02: El plugin utilizará la API de FOCA.

- RNF-03: El plugin deberá poder conectarse a instancias de SQL Server.
- RNF-04: El plugin utilizará una interfaz basada en Windows Forms.
- RNF-05: Se utilizará el mismo idioma que el usado en FOCA (Inglés).
- RNF-06: El plugin estará desarrollado para el sistema operativo Windows.
- RNF-07: El sistema deberá disponer de conexión a Internet.

3.7.3. Requisitos de Información

Los requisitos de información especifican qué información será utilizada por el sistema.

- RI-01: El sistema requerirá la siguiente información:
  - Direcciones de correo electrónico extraídas de documentos ofimáticos por FOCA.
- RI-02: El sistema obtendrá la siguiente información de los servicios de inicio de sesión:
  - Dirección de correo electrónico
  - Estado de la cuenta en cada red social objetivo (existente o inexistente)

3.7.4. Requisitos de Seguridad y Privacidad

Los requisitos de seguridad y privacidad recogen las medidas de protección que se deben llevar a cabo para garantizar la seguridad del sistema y la información que guarda.

- RSP-01: No se utilizarán los nombres de usuario y contraseñas por defecto como credenciales de acceso al SGBD (SQL Server utiliza la autenticación del propio sistema operativo por defecto).
- RSP-02: No se utilizarán como credenciales del SGBD aquellos nombres de usuario y contraseñas presentes en los principales diccionarios de credenciales conocidos (*CrackStation* [43], *John The Ripper* [44], etc).
- RSP-03: Se dispondrá de la versión del SGBD más reciente y sin vulnerabilidades conocidas.

Capítulo 4

Análisis

En este capítulo se detallará el proceso de análisis realizado para especificar la funcionalidad del plugin a desarrollar y sus interacciones con los actores involucrados. Incluirá sólo el análisis realizado tras el cambio de objetivos de proyecto, ya que la versión inicial fue descartada.

4.1. Casos de uso

En esta sección se explicará en detalle la funcionalidad que debe ofrecer el plugin. Para cada caso de uso se expondrán precondiciones, una descripción, postcondiciones, excepciones y su desarrollo paso a paso, acompañado de su respectivo diagrama de secuencia.

4.1.1. Diagrama de casos de uso

Esquema general de la interacción entre el sistema y los distintos actores identificados.

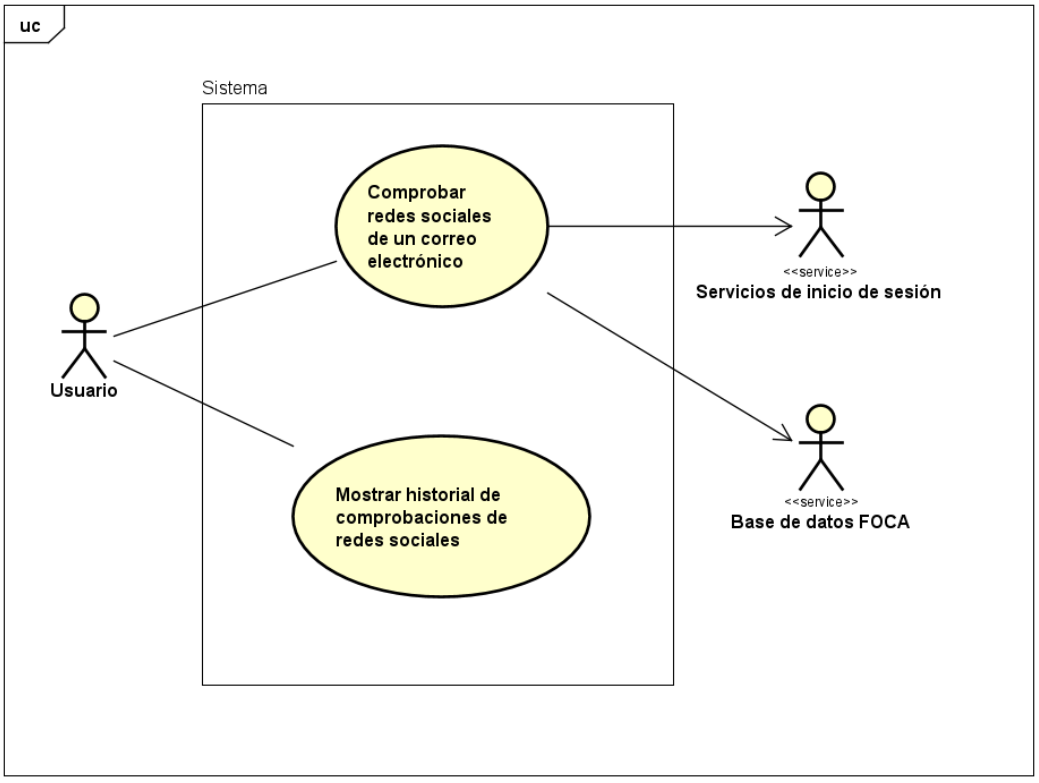


Figura 4.1: Diagrama de casos de uso

4.1.2. CU1 - Comprobar redes sociales de un correo electrónico

En este caso de uso se consultan los servicios de inicio de sesión de varias redes sociales para comprobar si una dirección de correo electrónico dada posee cuenta en estos servicios (referido como *Comprobación de redes sociales* de aquí en adelante).

CU-1	**Comprobar redes sociales de un correo electrónico**
Actor	**Usuario**
Descripción	Este caso de uso comprueba en distintos servicios de redes sociales si un correo electrónico dado posee cuenta de usuario en estos servicios.
Precondición	Plugin añadido y cargado con el gestor FOCA.
Secuencia Normal	
Paso	**Acción**
1	El usuario introduce un correo electrónico.
2	El sistema utiliza el correo electrónico obtenido contra los servicios de redes sociales para comprobar si tiene cuenta en ellos, notifica al usuario con el resultado de la comprobación en cada red social y actualiza el historial.
3	El caso de uso termina.
Secuencia alternativa	
Pasos	**Acción**
1b	El usuario pulsa en *Consultar*.
2b	El sistema muestra los correos electrónicos encontrados por FOCA en los metadatos de documentos ofimáticos.
3b	El usuario selecciona un correo electrónico de entre los obtenidos por la consulta.
4b	El caso de uso continúa en el paso 2.
Excepciones	
Pasos	**Acción**
1c	El correo electrónico introducido no es válido. Se cancela el caso de uso.
Postcondición	Los resultados obtenidos de la comprobación de redes sociales están disponibles para el usuario y actualizados en el historial.

Cuadro 4.1: Caso de Uso 1 - Comprobar redes sociales de un correo electrónico

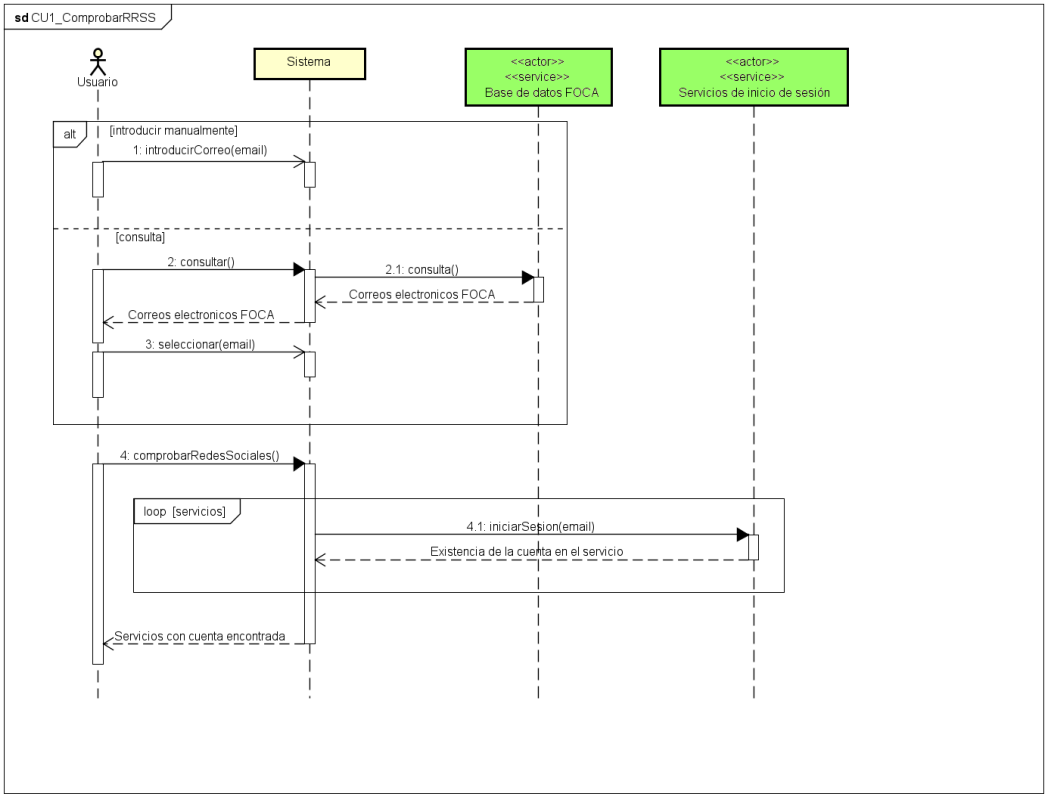


Figura 4.2: Diagrama de secuencia - Caso de Uso 1

4.1.3. CU2 - Mostrar historial de comprobaciones de redes sociales

En este caso de uso el usuario consulta la información sobre las comprobaciones de redes sociales (en qué redes sociales existen cuentas de usuario que utilizan un correo electrónico dado) realizadas anteriormente.

CU-2	Mostrar historial de comprobaciones de redes sociales
Actor	**Usuario**
Descripción	Este caso de uso consulta la información del historial de comprobaciones de redes sociales realizadas.
Precondición	Plugin añadido, cargado con el gestor FOCA y con comprobaciones de redes sociales realizadas anteriormente.
Secuencia Normal	
Paso	**Acción**
1	El usuario selecciona Historial.
2	El sistema muestra la información sobre las comprobaciones de redes sociales realizadas al usuario.
3	El caso de uso termina.
Secuencia alternativa	
Pasos	**Acción**
3b	El usuario selecciona Exportar.
4	El sistema exporta los contenidos del historial a un archivo CSV.
5	El caso de uso termina.
Excepciones	
Variación	**Acción**
2b	El historial no contiene ninguna información. El sistema muestra el historial vacío.
Postcondición	La información del historial está disponible para el usuario.

Cuadro 4.2: Caso de Uso 2 - Mostrar historial de comprobaciones de redes sociales

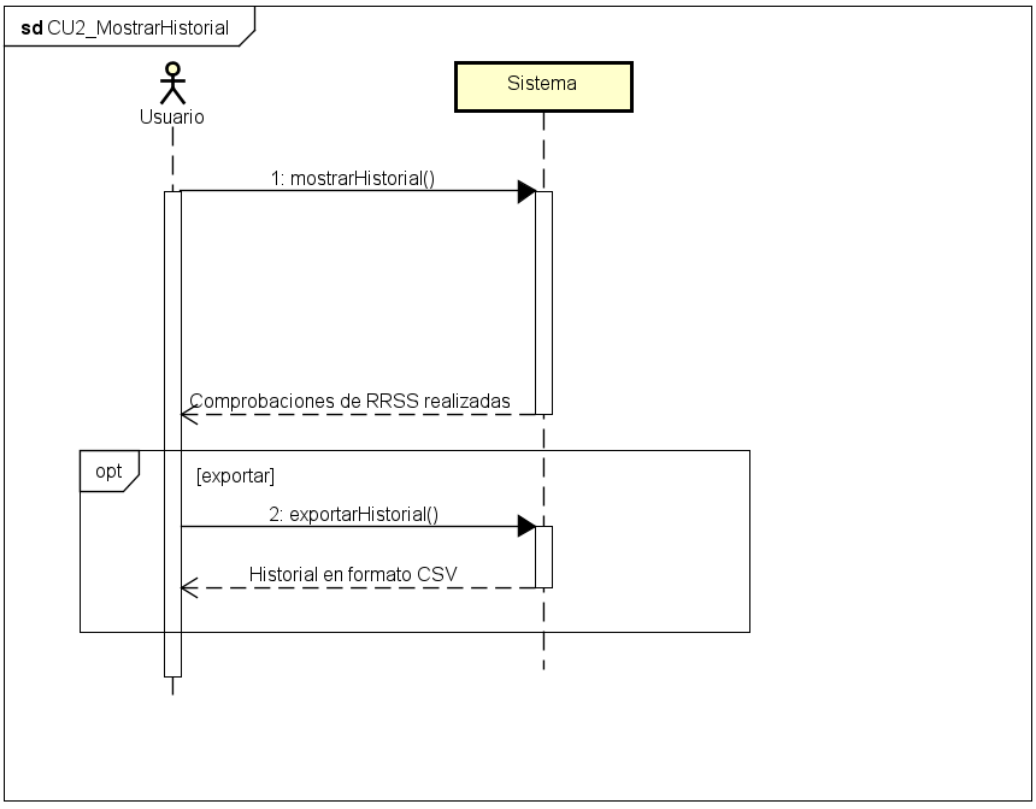


Figura 4.3: Diagrama de secuencia - Caso de Uso 2

4.2. Modelo de dominio

En esta sección se muestran las clases conceptuales pertenecientes al dominio del plugin, identificadas mediante un análisis de frases nominales [27] realizado sobre los requisitos y casos de uso.

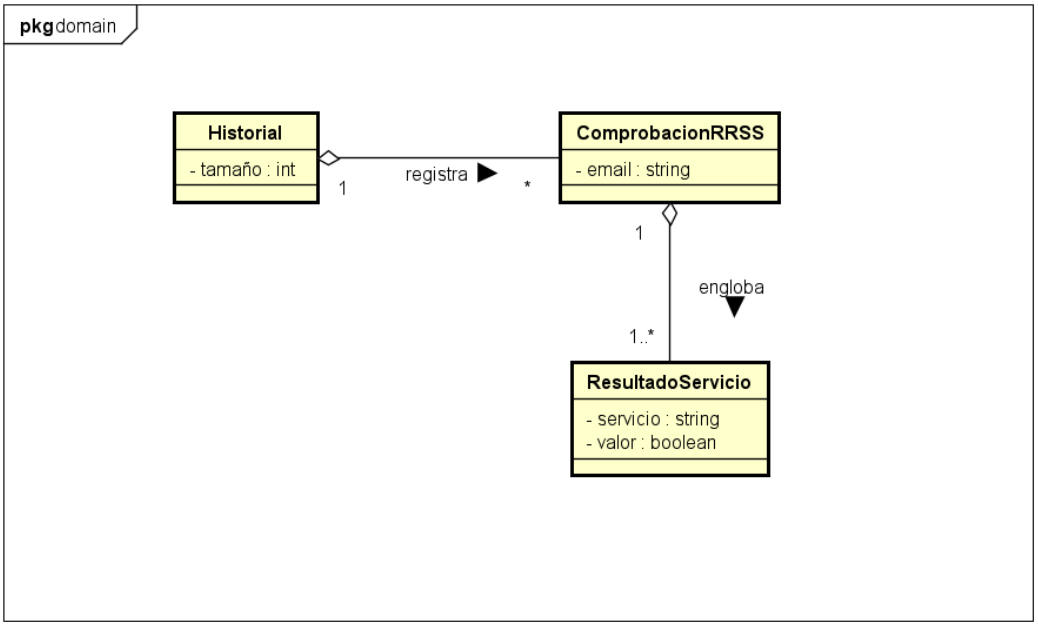


Figura 4.4: Modelo de dominio

4.3. Diagrama de clases de análisis

Desarrollo del diagrama de clases de análisis aplicando el enfoque Boundary-Control-Entity (BCE). Se utilizaron como base las clases conceptuales del modelo de dominio. Las clases de análisis identificadas son:

- **pluginMainForm.** Clase de tipo *Boundary* que modela todas las interacciones con el actor.
- **ControladorComprobacion.** Clase de tipo *Control* que maneja la lógica de la aplicación referente a las comprobaciones de redes sociales.
- **ControladorWeb.** Clase de tipo *Control* que maneja el proceso de interacción con los servicios web de inicio de sesión de las redes sociales y la extracción de su información .
- **ControladorHistorial.** Clase de tipo *Control* que maneja la lógica de la aplicación referente al historial de comprobaciones de redes sociales realizadas.
- **ControladorBD.** Clase de tipo *Control* que maneja las consultas de correos electrónicos realizadas contra la base de datos de FOCA.
- **ComprobacionRRSS.** Clase de tipo *Entity* que modela los resultados de una comprobación de redes sociales realizada usando un correo electrónico determinado (en qué servicios se han encontrado cuentas de usuario que utilizan ese correo electrónico)
- **ResultadoServicio.** Clase de tipo *Entity* que modela el resultado de la comprobación de redes sociales para un correo electrónico en un servicio en específico.
- **Historial.** Clase de tipo *Entity* que modela la información guardada en el historial de comprobaciones de redes sociales realizadas.

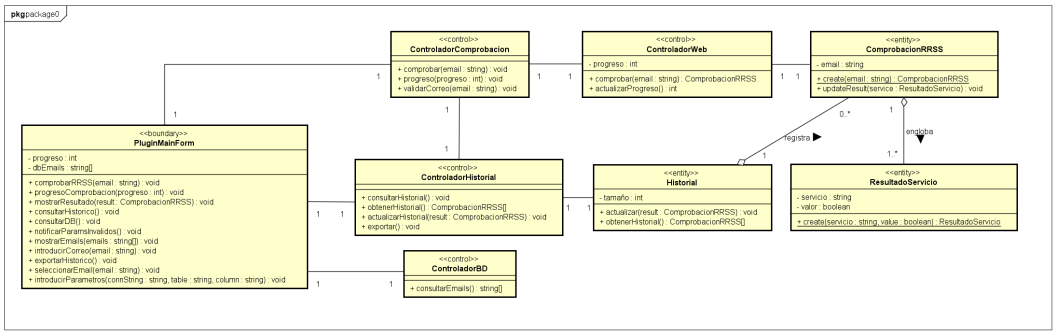


Figura 4.5: Diagrama de Clases de Análisis (BCE)

4.4. Realización de Casos de Uso de Análisis

En este apartado se detalla la realización de los casos de uso en análisis, aplicando un enfoque BCE. De cada caso de uso se expondrá una breve descripción y las clases de análisis participantes junto con su correspondiente diagrama de secuencia.

Caso de Uso 1 - Comprobar redes sociales de un correo electrónico

En este caso de uso se consultan los servicios de inicio de sesión de varias redes sociales para comprobar si una dirección de correo electrónico proporcionada posee cuenta en estos servicios (comprobación de redes sociales). Las clases de análisis participantes son (ver apartado 4.3):

- *pluginMainForm*, *ControladorComprobacion*, *ControladorWeb*, *ComprobacionRRSS* y *ResultadoServicio* para la comprobación de redes sociales.
- *ControladorHistorial* e *Historial* para actualizar el historial.
- *ControladorBD* para consultar los correos electrónicos encontrados por FOCA.
- *Servicios de inicio de sesión* y *Base de datos FOCA* como servicios externos (en verde).

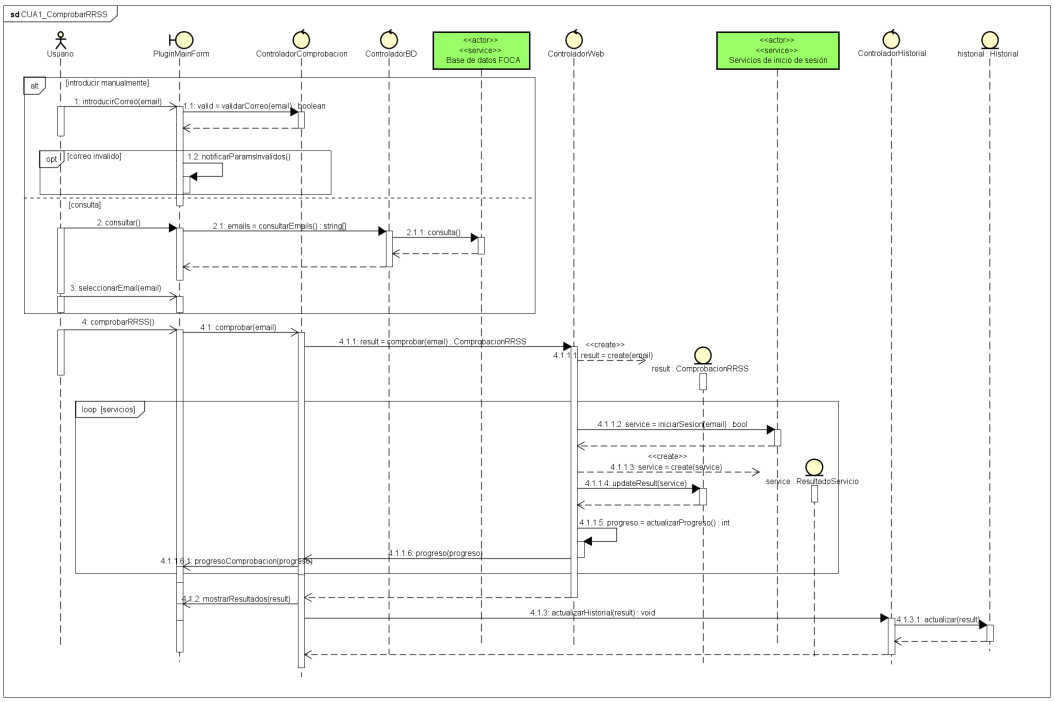


Figura 4.6: CUA1 - Comprobar redes sociales de un correo electrónico

Caso de Uso 2 - Mostrar historial de comprobaciones de redes sociales

En este caso de uso se muestra al usuario la información de las comprobaciones de redes sociales realizadas anteriormente, con la opción de exportarla a un archivo CSV. Las clases de análisis participantes son (ver apartado 4.3): *pluginMainForm*, *ControladorHistorial* e *Historial* para obtener y mostrar el historial.

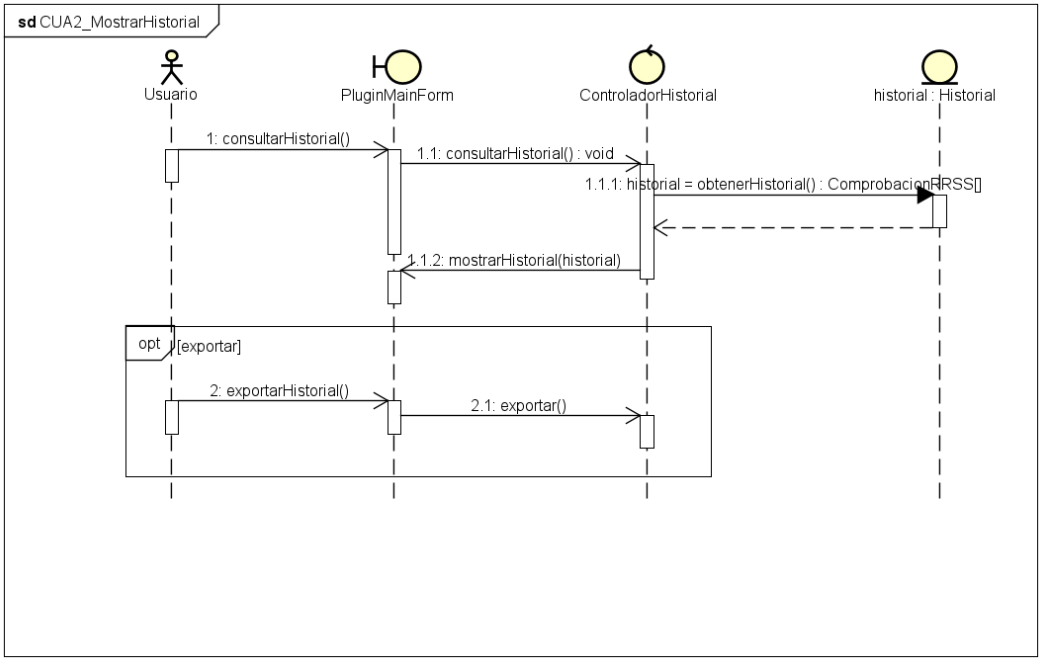


Figura 4.7: CUA2 - Mostrar historial de comprobaciones de redes sociales

4.5. Flujo de información

En esta sección se explicarán los orígenes y destinos de la información que el sistema debe manejar (definida en los Requisitos de Información, ver apartado 3.7.3). El plugin recoge información de 2 fuentes principales:

- **FOCA**. De este sistema se recogerán los correos electrónicos encontrados en los metadatos de documentos ofimáticos de Internet. Estos correos se utilizarán contra los servicios de inicio de sesión de redes sociales.
- **Servicios de inicio de sesión de redes sociales**. Utilizando los correos electrónicos de FOCA contra los servicios de inicio de sesión, se obtiene en qué redes sociales existe una cuenta de usuario con ese nombre (un conjunto de valores *boolean* por cada una de las redes sociales consultadas).

El plugin guardará el conjunto de valores *boolean* obtenido de los servicios de inicio de sesión junto con el correo electrónico utilizado (englobados bajo *Resultados* en la figura 4.8) en un archivo CSV (a petición del usuario).

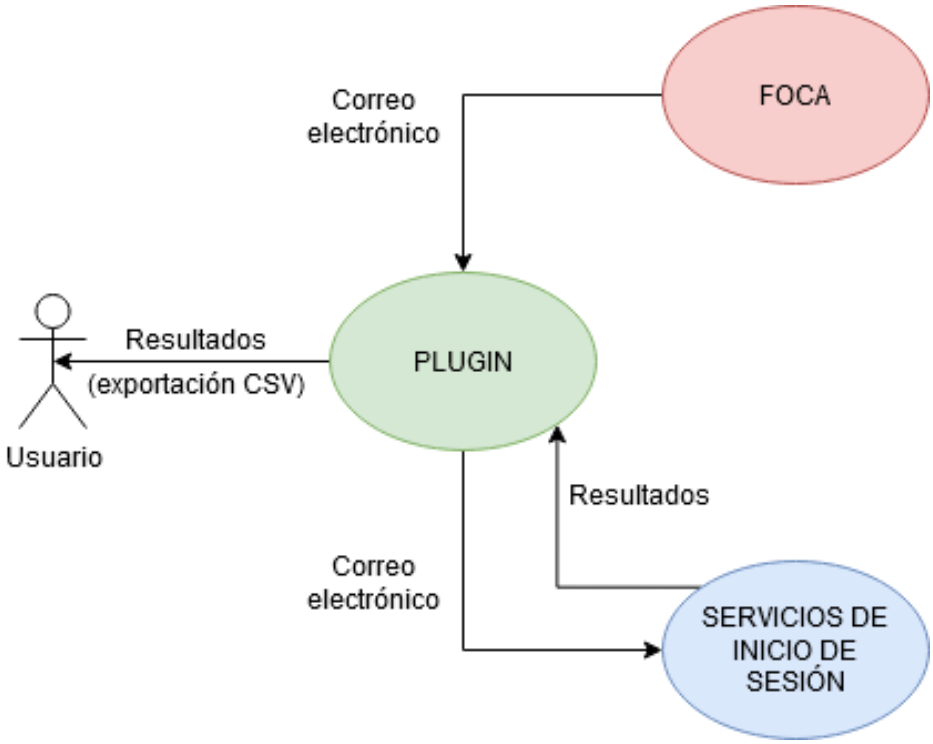


Figura 4.8: Flujo de información del sistema.

Capítulo 5

Diseño

En esta fase se realiza la especificación de la funcionalidad del software a desarrollar a un nivel más bajo que en la fase de análisis, profundizando más en los elementos software necesarios para su implementación.

5.1. Arquitectura

A la vista de los requisitos, casos de uso, y clases de análisis identificadas en el proceso de análisis anteriormente realizado, se utilizará una arquitectura de 3 capas [45]:

- Capa de Presentación: Esta capa gestiona las interacciones entre el usuario y la lógica del sistema. En este caso, la capa está compuesta por las interfaces de usuario embebidas en FOCA.
- Capa de lógica de aplicación: Maneja toda la lógica de la aplicación. Engloba tanto los elementos del dominio de la aplicación como los servicios que se ofrecen, actuando como intermediarios entre la interfaz y el acceso a datos.
- Capa de acceso a Datos: Gestionan las fuentes de información que utilizará el sistema. Aquí se incluyen el acceso a la base de datos y la interacción con los servicios de inicio de sesión.

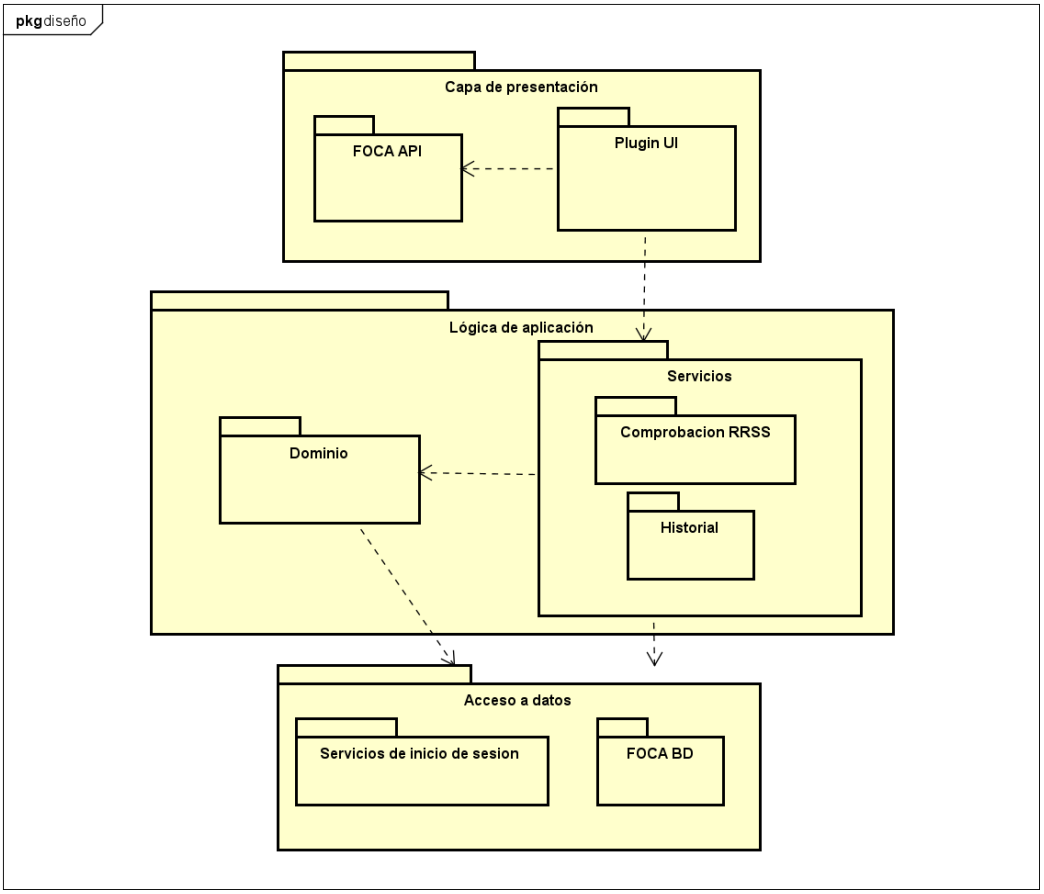


Figura 5.1: Arquitectura del sistema

5.2. Interfaz de usuario

Para realizar el diseño de la interfaz de usuario primero es necesario definir las características principales de los usuarios que van a utilizar la herramienta, así como sus necesidades a la hora de interactuar con la interfaz.

Este plugin está orientado a los usuarios de la herramienta FOCA, en la que está integrado. FOCA es utilizada en el campo de la Ciberseguridad como herramienta profesional en *Pentesting* o en Auditorías de Seguridad, entonces la eficiencia de trabajo será una necesidad esencial que se debe maximizar.

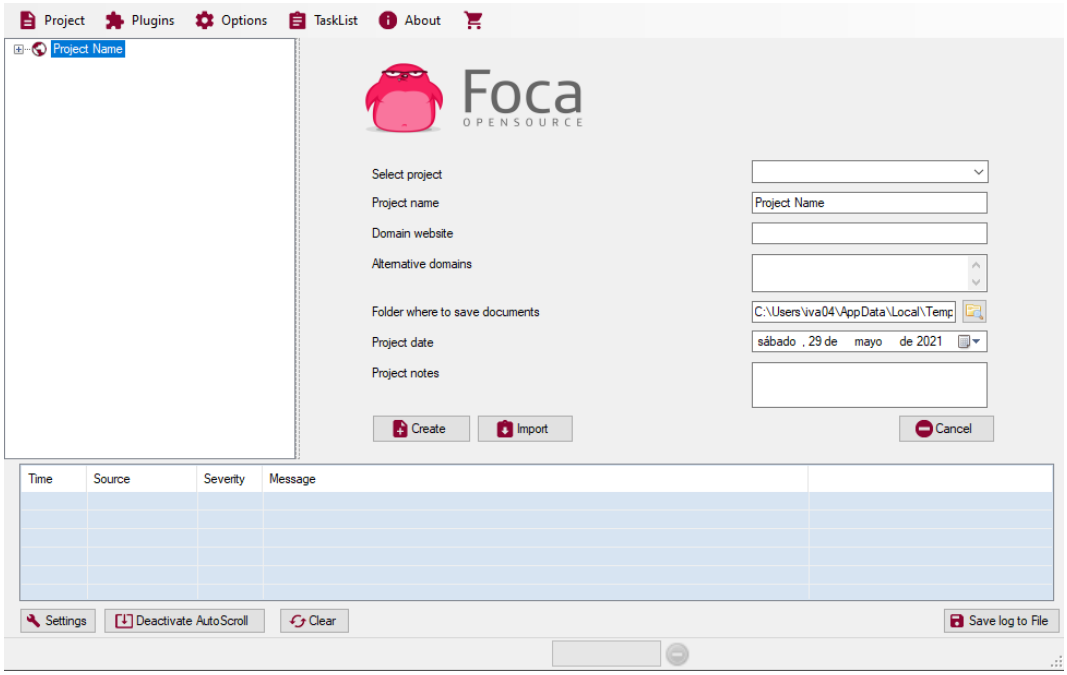


Figura 5.2: Interfaz de FOCA (I)

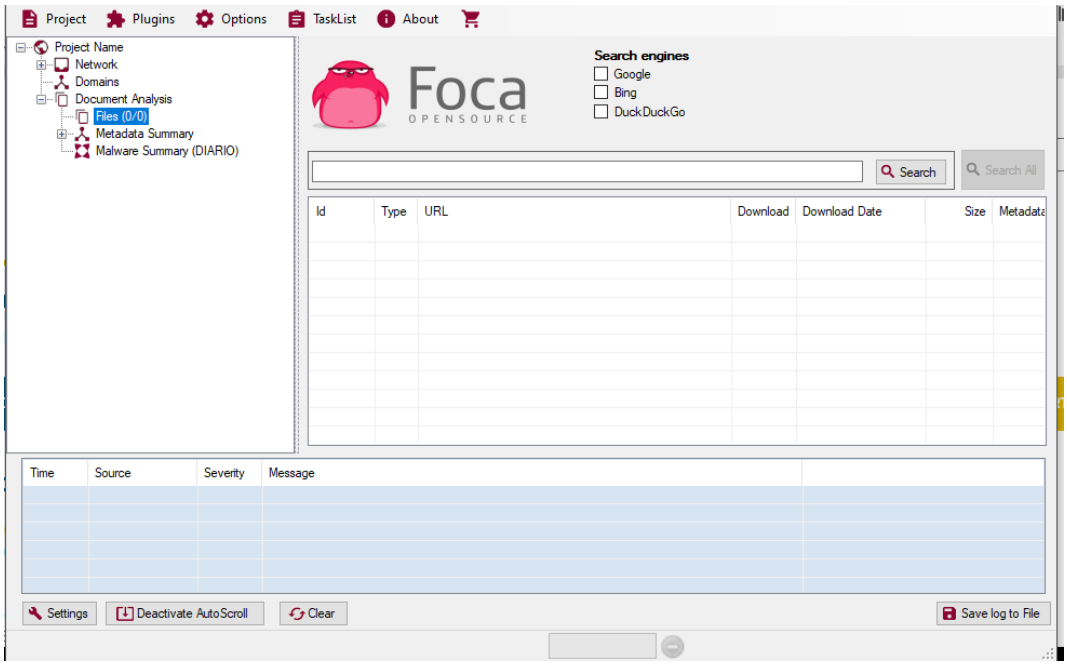


Figura 5.3: Interfaz de FOCA (II)

También facilitaría el uso de la interfaz el seguir los mismos patrones de diseño y formato que los utilizados en FOCA o en el resto de sus plugins.

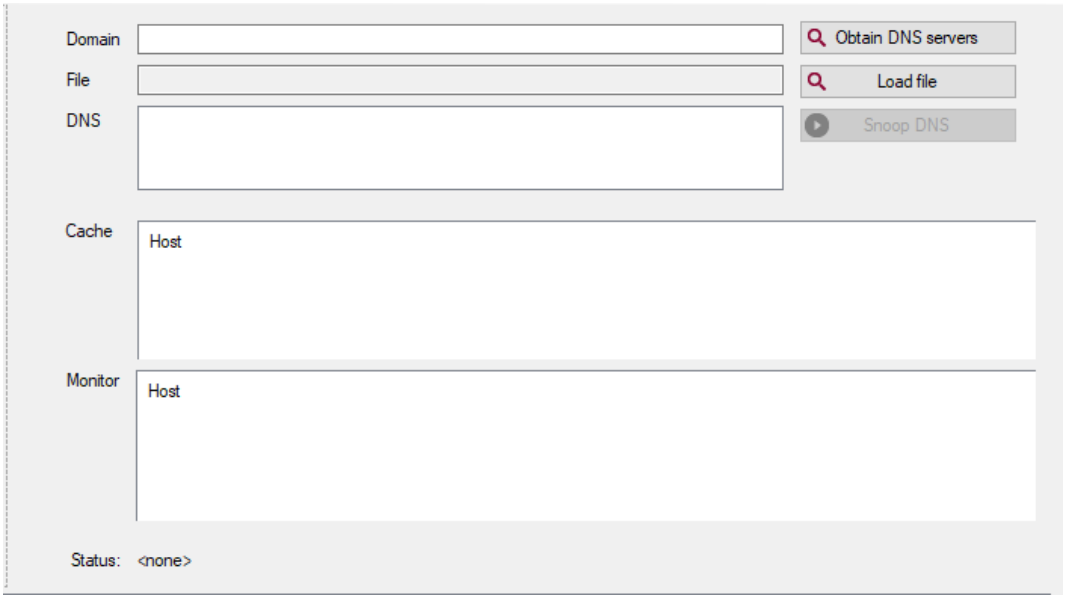


Figura 5.4: Interfaz del plugin DNS Snooping [46]

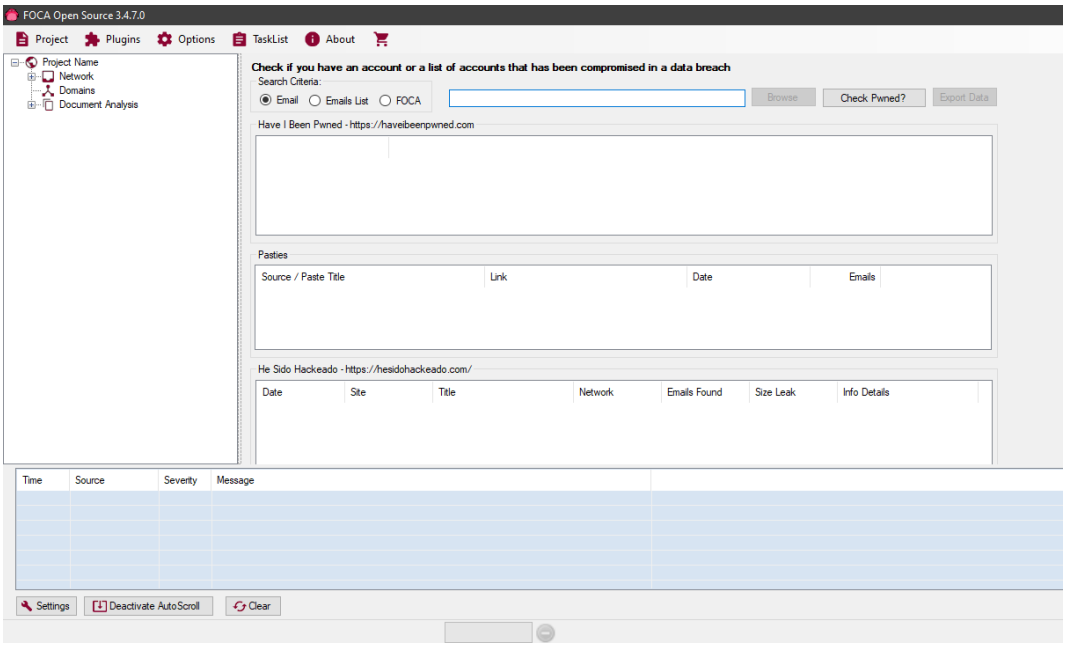


Figura 5.5: Interfaz del plugin HaveIBeenPwned

Teniendo los casos de uso y las necesidades de los usuarios en mente, se utilizó un patrón de diseño de pestañas para separar las partes de la interfaz relacionadas con las extracciones y las consultas a FOCA, de la interfaz del histórico. La razón detrás de esto es la posibilidad de consultar los correos electrónicos encontrados por FOCA antes de realizar un escaneo, ya que necesitas de una dirección de correo electrónico para iniciarlo. Para los bocetos de la interfaz he utilizado WireFrame [29].

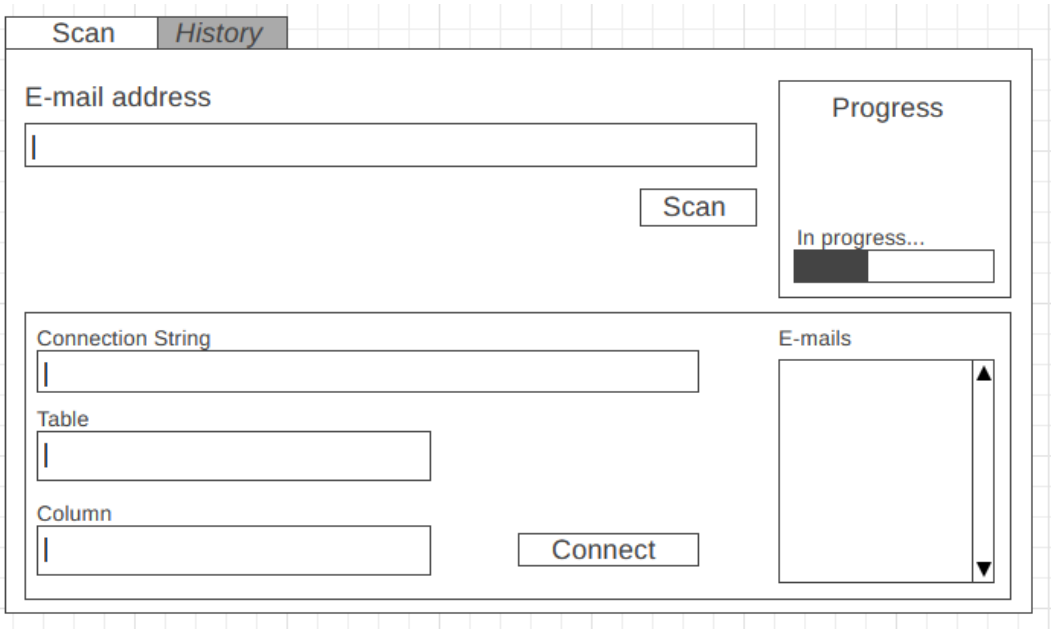


Figura 5.6: Interfaz de usuario (I)

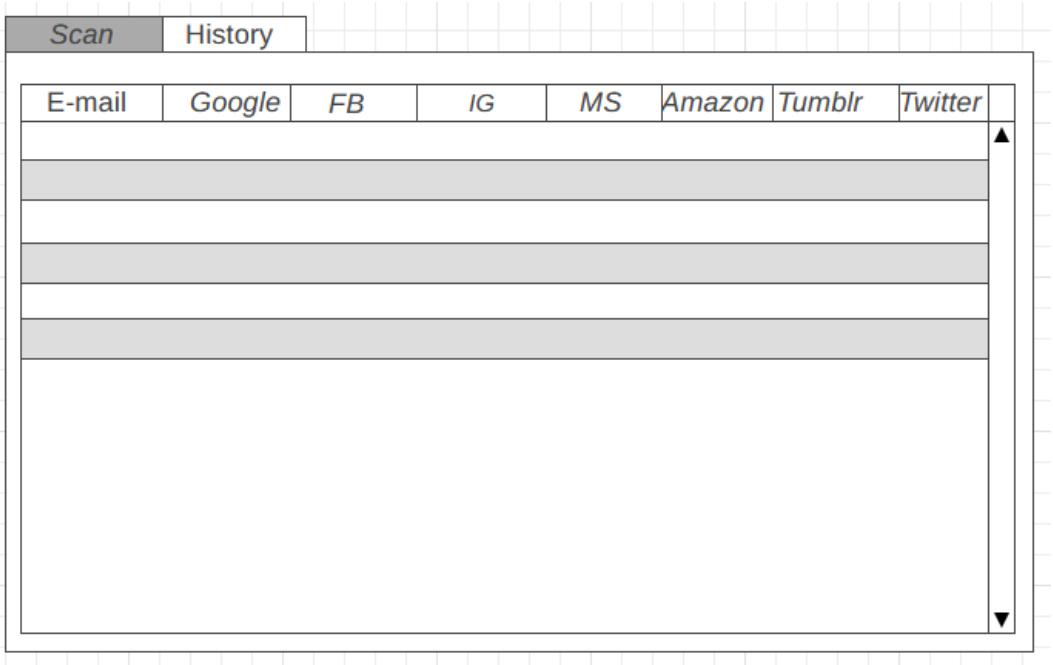


Figura 5.7: Interfaz de usuario (II)

5.3. Diagrama de Clases de Diseño

En esta sección detallaré las clases de diseño utilizadas. Estas clases han sido derivadas de las clases de análisis identificadas en el apartado 4.3. Las clases de diseño utilizadas son:

- **pluginForm.** Esta clase modela la interfaz de usuario. Se trata de un formulario que recoge toda interacción del usuario con el sistema e invoca la lógica de aplicación correspondiente.
- **RRSSController.** Esta clase se trata de un controlador GRASP [27] que recoge toda la funcionalidad del plugin referente a las comprobaciones de redes sociales.
- **HistoryController.** Esta clase se trata de un controlador GRASP que recoge la funcionalidad del plugin referente al historial de comprobaciones de redes sociales realizadas.
- **WebExtractor.** Esta clase se encarga de la interacción y extracción de información de los servicios Web de inicio de sesión.
- **DBAccess.** Esta clase se encarga de la conexión y las consultas realizadas contra la base de datos de FOCA.
- **ScanResult.** Clase perteneciente al dominio de la aplicación que modela los resultados de una comprobación de redes sociales realizada usando un correo electrónico determinado.
- **ServiceResult.** Clase perteneciente al dominio de la aplicación que modela el resultado de la comprobación de redes sociales para un correo electrónico en un servicio en específico.

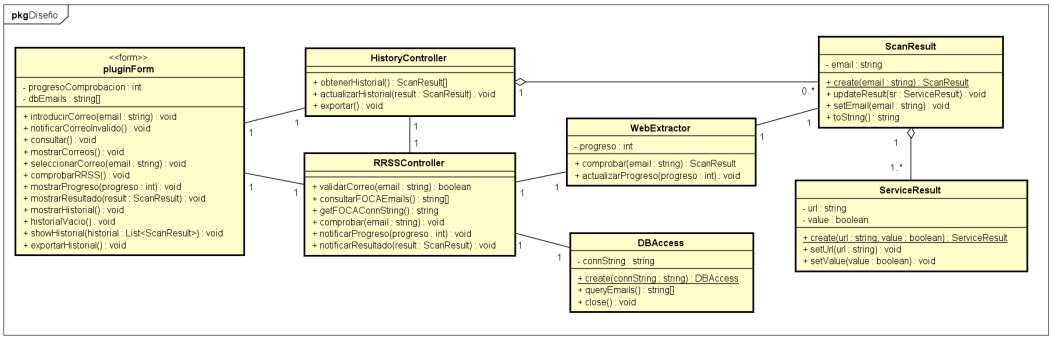


Figura 5.8: Diagrama de Clases de Diseño

5.4. Realización de los Casos de Uso de Diseño

En esta sección se detalla la realización de los casos de uso de diseño. Para cada caso de uso se expondrán una descripción y las clases de diseño participantes con su respectivo diagrama de secuencia.

Caso de Uso 1 - Comprobar redes sociales de un correo electrónico

En este caso de uso se consultan los servicios de inicio de sesión de varias redes sociales para comprobar si una dirección de correo electrónico proporcionada posee cuenta en estos servicios (comprobación de redes sociales). Las clases de diseño participantes son (ver apartado 5.3):

- *pluginForm*, *RRSSController*, *WebExtractor*, *ScanResult* y *ServiceResult* para la comprobación de redes sociales.

- *HistoryController* para actualizar el historial.
- *DBAccess* para consultar los correos electrónicos guardados en la base de datos de FOCA.
- *Servicios de inicio de sesión* y *Base de datos FOCA* como servicios externos (en verde).

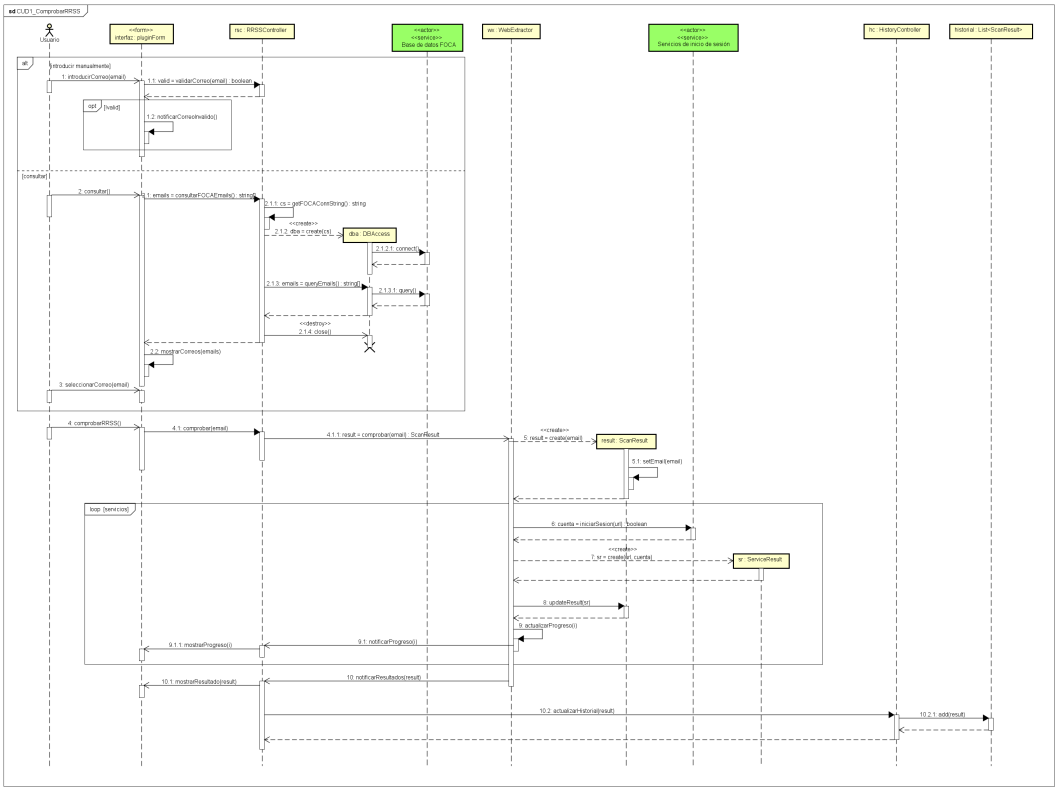


Figura 5.9: CUD1 - Comprobar redes sociales de un correo electrónico

Caso de Uso 2 - Mostrar historial de comprobaciones de redes sociales

En este caso de uso se muestra al usuario la información de las comprobaciones de redes sociales realizadas anteriormente, con la opción de exportarla a un archivo CSV. Las clases de diseño participantes son (ver apartado 5.3):

- *pluginForm* e *HistoryController* para obtener y mostrar el historial.
- *ScanResult* para obtener la información de cada comprobación de redes sociales para realizar la exportación.

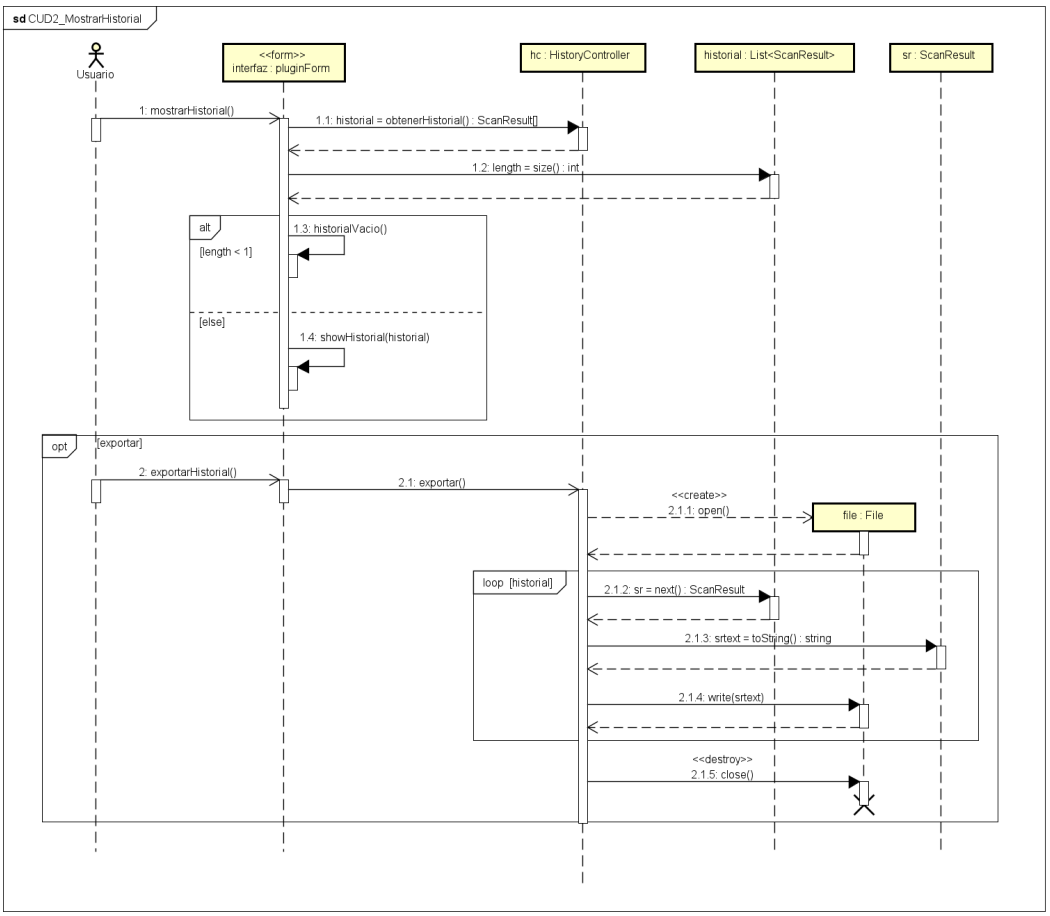


Figura 5.10: CUD2 - Mostrar historial de comprobaciones de redes sociales

5.5. Persistencia de la información

En esta sección se explicará el diseño de los distintos mecanismos de persistencia de información especificados en los requisitos y en los casos de uso: consultas a una base de datos y exportación de información al formato CSV.

5.5.1. Bases de datos

En el Caso de Uso 1 (ver apartado 4.1.2) se permite al usuario utilizar un correo electrónico de entre los guardados en la base de datos de FOCA para realizar una comprobación de redes sociales. En esta sección se describe la parte del esquema de datos donde reside esta información.

Dentro del esquema de datos de FOCA (ver apartado 3.1.1), es de especial interés la localización de los correos electrónicos que queremos consultar. Los correos electrónicos conforman su propia categoría dentro de los metadatos extraídos de documentos ofimáticos (tabla *EmailsItems*, columna *Mail* en azul, ver figura 5.11). Esta información, junto con el resto de metadatos, están referenciados en una tabla (*MetaExtractors*) que recoge la información sobre los metadatos encontrados en un documento ofimático. La información sobre los documentos ofimáticos guardados está en la tabla *MetaDatas*.

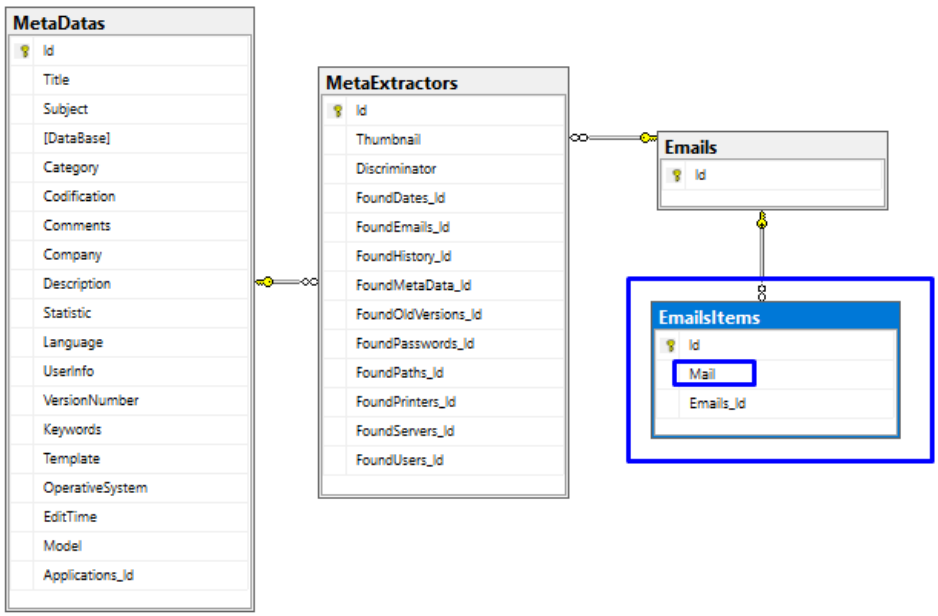


Figura 5.11: Tabla *EmailsItems* en el esquema de datos de FOCA (obtenido con SSMS, ver apartado 3.2.2)

5.5.2. Exportación CSV

En el Caso de Uso 2 (ver apartado 4.1.3) se permite al usuario exportar los contenidos del historial de comprobaciones de redes sociales a un archivo en formato CSV. CSV (*Comma Separated Values*) es un formato de texto abierto que permite representar información en forma de tabla, separando cada valor mediante un separador [47]. Dependiendo de las limitaciones de cada situación, este separador puede variar (coma, punto y coma, tabulaciones, etc). Debido a su simplicidad es muy fácil de implementar y procesar, y es soportado por gran cantidad de herramientas y aplicaciones. No tiene una especificación definida por lo que su implementación es de libre interpretación (aunque la mayoría de implementaciones comparten algunos conceptos básicos, como la coma como separador o los nombres de los campos en la primera línea). El formato utilizado es:

- Cada línea del archivo es una nueva entrada. En este caso, cada entrada corresponderá con una comprobación de redes sociales de un correo electrónico.
- Utilizar la coma (”,”) como separador.
- La primera linea del archivo son los nombres de cada campo definido (separados por el separador utilizado)
- Los campos recogidos son:
  - **Email:** El correo electrónico utilizado en la comprobación de redes sociales.
  - Un campo por cada servicio consultado en la comprobación de redes sociales, siendo en este caso: **Google, Facebook, Amazon, Twitter, Instagram, Microsoft, Tumblr**. Su valor sera *True* si el correo electrónico posee cuenta en ese servicio, y *False* en caso contrario.

Capítulo 6

Implementación

En este capítulo se explicarán los detalles de la implementación de la funcionalidad descrita en los capítulos de Análisis y Diseño.

6.1. Extracción de información de redes sociales

En esta sección se detallará el proceso de implementación de la extracción de la información de los servicios de inicio de sesión de las redes sociales.

Para interactuar con los servicios de inicio de sesión de las redes sociales y extraer su información, se decidió reutilizar una de las librerías que usé contra el servicio de Gmail, Selenium [32]. En este caso también es necesario disponer de una herramienta de más alto nivel que el *Web scraping* tradicional (extracción de información de una página web buscando en el código HTML obtenido mediante una petición HTTP) ya que estos inicios de sesión están orientados a usuarios reales y, aunque no bloqueen al software automatizado, en determinados servicios de inicio de sesión se pueden utilizar tecnologías o patrones de diseño que obstaculizan el *scraping* de manera indirecta, como por ejemplo una interfaz más responsiva mediante AJAX (provocando que la información no se encuentre en el HTML que recibimos mediante *scraping*). Las librerías como Selenium interactúan con la página de forma parecida a los usuarios reales, mediante *clicks*, rellenando formularios, etc. Así, el acceso a los servicios de inicio de sesión será mas uniforme (al menos en lo que a tecnología utilizada se refiere).

6.1.1. Selenium

Figura 6.1: Logo de Selenium

Selenium es un entorno de *software testing* para aplicaciones Web [32]. Permite la automatización de

casos de prueba utilizando varios navegadores a través de sus respectivos controladores. Aunque esté dedicado al *testing*, puede ser utilizado para extraer información de sitios web gracias a las funcionalidades que ofrece la librería [34]. Existe una versión para .NET en forma de paquete NuGet [33], el gestor de paquetes principal de dicho *framework*.

Para utilizar esta librería hay que instalar el respectivo paquete NuGet (*Selenium.WebDriver v3.141.0 - Latest*) junto con los paquetes correspondientes a los controladores que queramos utilizar desde el gestor de paquetes de Visual Studio. Se instalaron los controladores de Firefox (*Selenium.Firefox.WebDriver v0.27.0 - Latest*) y Google Chrome (*Selenium.Chrome.WebDriver v85.0.0 - Latest*) para la realización de este proyecto.

A la vista de los inicios de sesión de cada red social, es necesario realizar una implementación individual del proceso de extracción de información de cada una de ellas. La situación ideal sería poder generalizar el proceso de extracción para todas las redes sociales, pero cada página de inicio de sesión posee sus peculiaridades respecto a las demás, haciendo que la extracción sea muy distinta en algunos casos (por ejemplo, algunos servicios piden únicamente la dirección de correo electrónico antes de comprobar la existencia de la cuenta, mientras que otros requieren también una contraseña). Esto limita mucho la escalabilidad de la aplicación, ya que para cada red social que se quiera añadir es necesario programar su propio proceso de extracción de información.

En general, todos los procesos para extraer la información siguen estos pasos:

- (Configuración del controlador de Selenium).
- Visitar el sitio.
- Buscar y rellenar la entrada de texto correspondiente al correo electrónico.
- Esperar por la confirmación de la existencia de la cuenta.

Antes de poder visitar un sitio, primero hay que instanciar y configurar el *WebDriver* que se utilizará, en este caso Firefox. También se tendrá que especificar si usar el modo *headless* (sin interfaz gráfica) o no.

```
FirefoxOptions fopts = new FirefoxOptions();

//Headless
if (headless) { fopts.AddArguments("--headless"); }

var ffds = FirefoxDriverService.CreateDefaultService();
ffds.HideCommandPromptWindow = true;

//Inicializar el controlador
this.driver = new FirefoxDriver(ffds, fopts);
```

Figura 6.2: Configuración del *WebDriver* de Firefox

Una vez configurado el *driver* ya se puede interactuar con el. Si no se utilizó el modo *headless* aparecerá la ventana del navegador en blanco. Para visitar una página web determinada, se utiliza el método *WebDriver.Navigate().GoToUrl(url)*. Entonces la página comenzará a cargarse en el navegador. Es importante

mencionar que todas las interacciones con el navegador se realizan asíncronamente, es decir, el flujo de ejecución de la aplicación no va a esperar a que la página web haya terminado de cargar. Esto puede causar problemas a la hora de buscar elementos HTML en la página si aún no ha cargado del todo (no encontrará dicho elemento y lanzará *NoSuchElementException*). Para estos casos Selenium proporciona diversos mecanismos de espera: espera implícita (esperar un tiempo determinado), y espera explícita (esperar a que ocurra un evento). En este caso, como tras cargar la página se debe interactuar con un elemento, se puede utilizar una espera explícita para evitar problemas.

```
//Espera implícita (esperar 10s)
driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);

//Espera explicita (esperar 10s a que cargue cierto elemento)
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(
    SeleniumExtras.WaitHelpers.ExpectedConditions.ElementExists(
        By.Id(target_id)
    )
);
```

Figura 6.3: Esperas soportadas - Selenium

Para buscar elementos HTML, Selenium soporta varios tipos de búsqueda: por ID de elemento, nombre de la clase del elemento, nombre del elemento, *XPath*, etc. Todos los tipos de búsqueda soportados están recogidos en la clase *By* [48]. Entre estos, se ha utilizado *XPath*.

XPath es un lenguaje de búsqueda de nodos en árboles XML y JSON [49]. Selenium permite realizar búsquedas dentro del código HTML de la página (que es una especificación de XML) en tiempo de ejecución utilizando este lenguaje, siendo un método de búsqueda más general que buscar por un atributo determinado (es posible que el elemento en cuestión no disponga de ese atributo).

Cuadro 6.1: Elementos sintácticos de XPath [49]

Operador	Función
/	Buscar desde el nodo raíz
//	Buscar en todos los nodos
.	Buscar en el nodo actual
..	Buscar en el nodo padre del actual
@	Seleccionar atributos
*	Seleccionar todos los nodos
@*	Seleccionar todos los atributos
[]	Seleccionar atributos del nodo actual
elemento	Seleccionar el nodo "elemento"
|	Operador AND para combinar consultas

Entonces, para buscar dentro de la página un elemento *input* llamado *'account identifier'*, la consulta XPath sería: *//input[@name='account_identifier']*; o en el caso de buscar el elemento *div* que te notifica de que la cuenta no existe (por ejemplo, uno con la clase "o6cuMc"), sería: *//div[contains(@class, 'o6cuMc')]*. En este ultimo caso se utiliza la función *contains()* de XPath ya que el atributo de *class* puede incluir varios nombre de clase separados por espacios. Hay que tener en cuenta que cada página de inicio de

sesión de cada red social están programadas de distinta manera: cada una tiene sus propios elementos con sus propios atributos, por eso el proceso de extracción de la información tiene que ser específico para cada una de ellas.

```
this.driver.Navigate().GoToUrl(url);

IWebElement emailInput = this.driver.FindElement(By.XPath("//input[@name='account_identifier']"));
emailInput.SendKeys(email);
IWebElement buttonSiguiente = this.driver.FindElement(By.XPath("//input[@type='submit']"));
buttonSiguiente.Click();
```

Figura 6.4: Visitar una página y buscar un elemento - Selenium

Antes se ha mencionado que algunos patrones de diseño utilizados por redes sociales obstaculizan indirectamente las actividades de este tipo de software automatizado. Un claro ejemplo de esto son los avisos de uso de *Cookies*, muy utilizados en la actualidad. Dependiendo de su implementación, estos avisos son capaces de bloquear la interacción del usuario (y de Selenium) con la página hasta aceptarlos. Entonces, en las redes sociales que utilicen este tipo de *pop-up* hay que realizar pasos adicionales: buscar el botón de *Aceptar cookies* o similar y pulsarlo. Esto revela otro problema, los avisos de *Cookies* normalmente sólo aparecen una vez, por tanto cuando no aparezcan Selenium lanzará una excepción porque no encuentra dicho botón. Una manera efectiva de lidiar con esto es capturando las excepciones correspondientes.

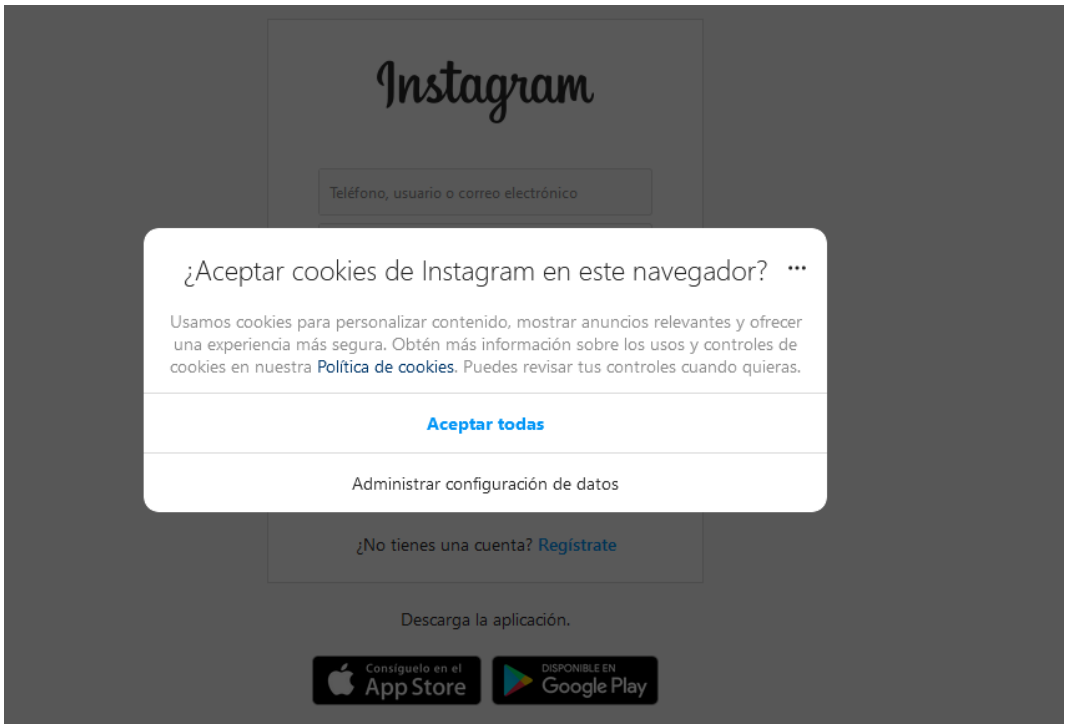


Figura 6.5: Aviso de uso de *Cookies* - Instagram

```
try
{
    IWebElement cookieButton = this.driver.FindElement(By.XPath("//button[contains(@class, 'aOOlW  bIiDR  ')]"));

    IJavaScriptExecutor ex = (IJavaScriptExecutor)driver;
    ex.ExecuteScript("arguments[0].click();", cookieButton);
}
catch (ElementClickInterceptedException) { } //Ignorar ClickIntercepted si no aparece el popup de cookies
catch (NoSuchElementException) { } //Y NoElement
```

Figura 6.6: Manejo de los avisos de *Cookies*

6.2. Interfaz gráfica

En este apartado se explicarán los detalles de la implementación de la interfaz diseñada en el capítulo anterior. FOCA permite a los plugins mostrar sus interfaces de usuario en un panel propio de la interfaz de FOCA dedicado a ellos. Es por esto que las interfaces de los plugins están sujetos a utilizar la misma API de interfaz gráfica que FOCA: Windows Forms.

Windows Forms es una interfaz de programación gráfica incluida dentro de Microsoft .NET Framework. Se trata de una API que permite el acceso a los elementos gráficos nativos del sistema operativo encapsulando las llamadas al mismo. Visual Studio provee de un diseñador de interfaces de Windows Forms interactivo que permite la compilación y despliegue de aplicaciones utilizando esta API de manera sencilla.

Tomando como referencia el diseño de interfaz realizado anteriormente, la interfaz consta de 3 partes:

- Pestaña de historial.
- Panel de consulta de correos electrónicos de FOCA.
- Panel de comprobaciones de redes sociales.

Historial

En la pestaña del historial es donde se muestran las operaciones realizadas anteriormente. Se trata de un *DataGridView*, un componente de Windows Forms que permite mostrar a los usuarios información en forma de tabla. Este componente es configurable desde el diseñador de Visual Studio para añadir las columnas convenientes (en este caso, el correo electrónico y una para cada red social) y editar el estilo de cada celda. El componente también se puede configurar en tiempo de ejecución utilizando los atributos y métodos de la clase *DataGridView*:

- *DataGridView.Rows:* Lista que contiene la información de cada fila guardada en la tabla.
- *DataGridViewRow.Cells:* Lista que contiene la información de cada celda de una fila de la tabla.
- *DataGridViewCell:* Clase que maneja la información referente a una celda de la tabla, incluyendo su valor, formato y estilo.

También incluye la función de exportar a CSV, para facilitar el análisis de los datos en herramientas como Microsoft Excel o relacionados.

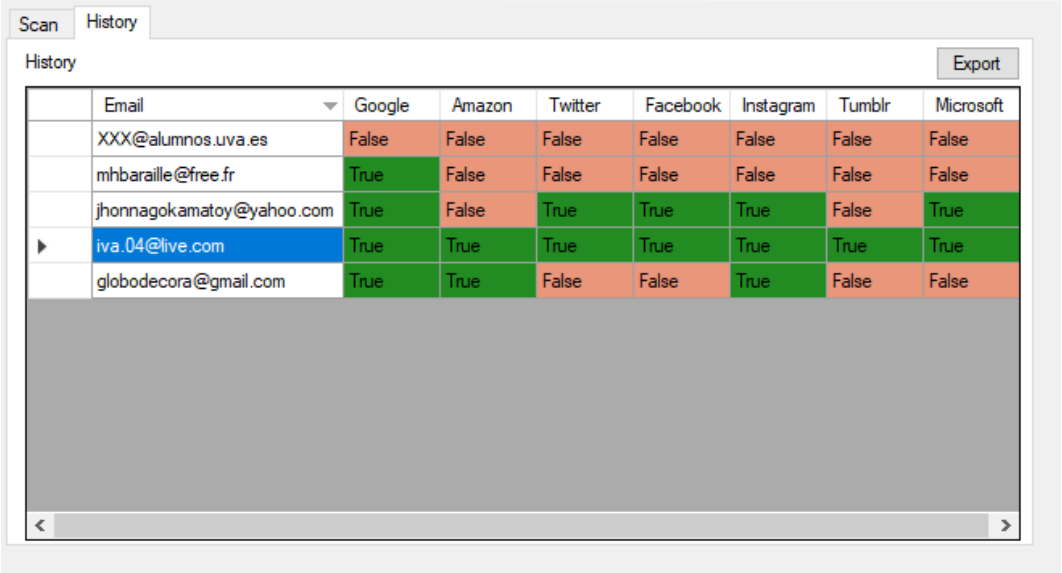


Figura 6.7: Interfaz gráfica del historial

Consulta de correos electrónicos de FOCA

En el panel de consultas de correos electrónicos se muestra al usuario los correos electrónicos guardados en la base de datos de FOCA. Incluye unas cajas de texto que informan al usuario de los parámetros de conexión de la instancia de base de datos que está siendo utilizada por FOCA, que son introducidos por el usuario la primera vez que lanza la herramienta y guardados por FOCA en `./FOCA/bin/Release/FOCA.exe.Config`.

Comprobación de cuentas de redes sociales

En el panel de comprobaciones de redes sociales es donde los usuarios pueden iniciar el proceso de comprobación de cuentas de usuario sobre una dirección de correo electrónico dada, configurar el *headless browser* y comprobar el estado del proceso si está en marcha. Una parte importante a la hora de implementar interfaces que interactúan con procesos de larga duración es mantener la funcionalidad de la interfaz durante la ejecución de dicho proceso. Si se utiliza el mismo hilo de ejecución que la interfaz para ejecutarlo, la interfaz no podrá responder a la interacción con el usuario hasta que la ejecución acabe. La solución para este problema es ejecutar el proceso de larga duración en un hilo separado al hilo de la interfaz. Windows Forms tiene un componente que permite mantener la funcionalidad de la interfaz, el *Background Worker*, que permite ejecutar tareas en un subproceso distinto. Para utilizarlo, sólo hay que buscarlo en la caja de herramientas del diseñador de interfaces y arrastrarlo a la interfaz en cuestión. Entonces el IDE preparará los *event listeners* (registros que indican que eventos ejecutan funcionalidad del componente) que necesita el componente para que el desarrollador pueda programar los *handlers* (fragmentos de código ejecutados al ocurrir un evento determinado).

La clase de *BackgroundWorker* provee 3 *listeners* principales, para los cuales habrá que implementar sus respectivos *handlers* con el código que queremos que se ejecute en cada caso:

- *DoWork.* El evento que se dispara cuando le indicamos al *worker* que inicie su tarea (mediante el método *BackgroundWorker.RunWorkerAsync()*).
- *ProgressChanged.* El evento que se dispara cuando desde el *worker* se notifica que ha habido progreso en la tarea (mediante el método *BackgroundWorker.notifyProgress(progress)*).

- *RunWorkerCompleted.* El evento que se dispara cuando el *worker* finaliza su tarea (teniendo disponible el resultado en el campo *Result* del propio evento).

Entonces, en el *handler* del evento *DoWork* ejecutamos el proceso de recogida de información, utilizando el método *notifyProgress()* para actualizar la barra de estado de la interfaz, y en el *handler* de *RunWorkerCompleted* preparar la interfaz para el final del proceso.

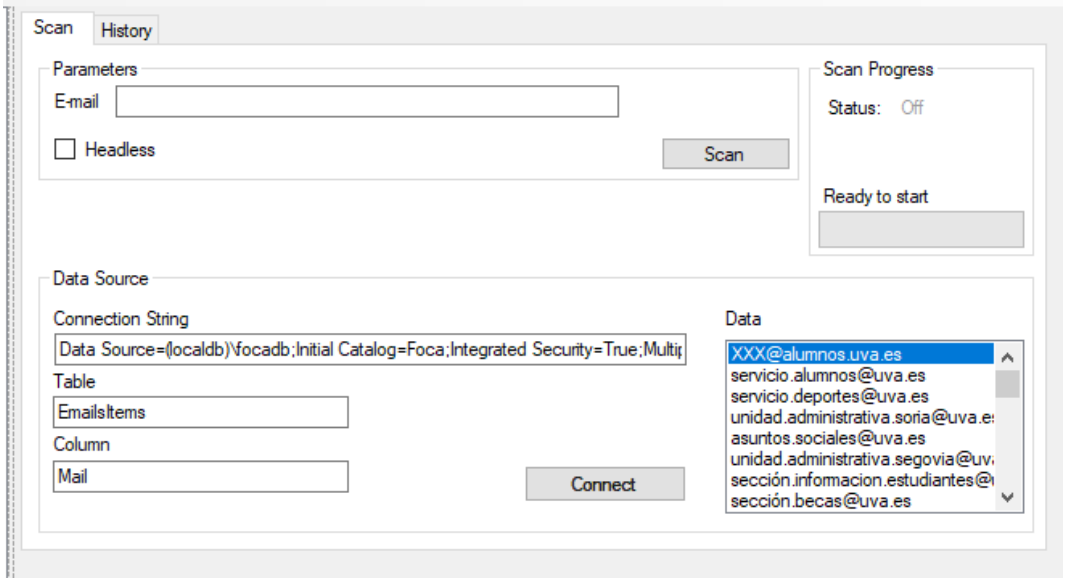


Figura 6.8: Interfaz gráfica principal

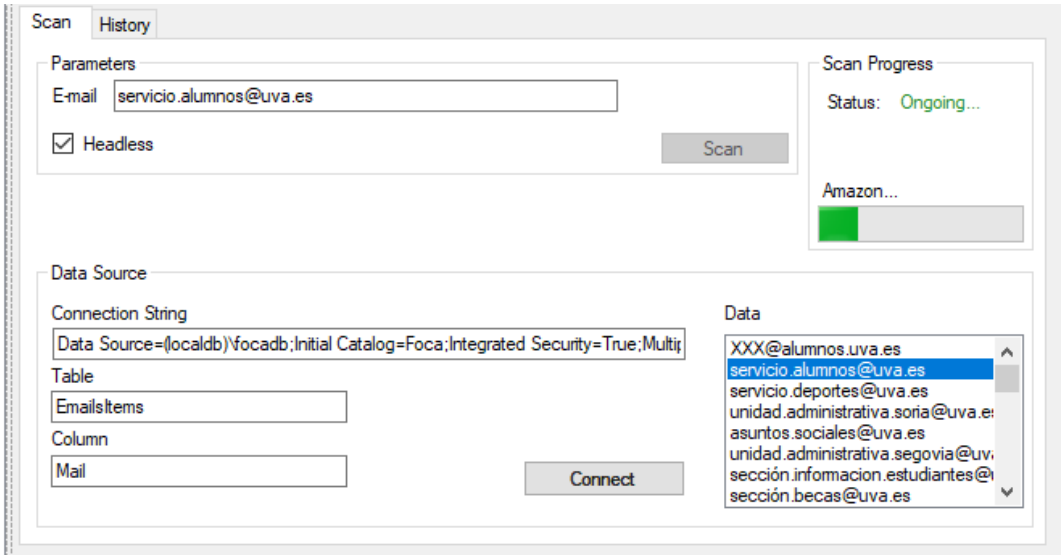


Figura 6.9: Interfaz gráfica principal durante un proceso de comprobación de redes sociales

6.3. Acceso a datos

Para implementar las interacciones del plugin y la base de datos de FOCA se utilizó la librería de *SQL-Client*, disponible como paquete en el gestor *NuGet* [53]. El plugin se conectará a la instancia de SQL Server que utiliza FOCA y buscará correos electrónicos en la tabla pertinente (*dbo.EmailsItems*). Para conectarse a la instancia de FOCA, es necesario disponer de su *connection string*. Este parámetro le proporciona el usuario la primera vez que inicia FOCA, que lo guarda en `./FOCA/bin/Release/FOCA.exe.Config` para evitar preguntar al usuario siempre que ejecute la herramienta. De esta manera, disponiendo de

esta información es posible conectarse a dicha instancia a través de cualquier conector SQL, en este caso *SQLClient*.

```
//Consulta SQL Parametrizada
SqlCommand cmd = new SqlCommand("SELECT "+column+" FROM "+table, this.db_conn);
SqlDataReader dr = cmd.ExecuteReader();
List<string> result = new List<string>();

//Procesado de resultados
while (dr.Read())
{
    string mail = dr.GetString(0);
    result.Add(mail);

}
dr.Close();
return result;
```

Figura 6.10: Implementación de la consulta con *SQLClient*

Capítulo 7

Pruebas

Este capítulo está dedicado a la documentación de los casos de prueba diseñados para asegurar que el plugin cumple con los requerimientos definidos anteriormente. En estas pruebas se asume que se dispone de todo lo necesario para operar FOCA con el mejor rendimiento posible (todas las dependencias de la herramienta, conexión a Internet, una instancia de SQL Server configurada, suficiente espacio en disco, memoria RAM libre y el sistema en un estado estacionario).

7.1. Batería de pruebas

De cada prueba diseñada se describirán: una breve descripción de la funcionalidad que se está probando, los pasos realizados, y los resultados que se esperan obtener de su ejecución. Las pruebas están numeradas por orden de ejecución.

CP-1	
Descripción	Comprobación de la existencia de cuentas de usuario en redes sociales de un correo electrónico.
Resultado esperado	Inicio del proceso de comprobación y al finalizar: un *pop-up* informativo y la actualización del histórico.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
Resultados obtenidos	
Ejecución	**Resultado**
1	No aparece el pop-up informativo.
2	Resultado esperado.

Cuadro 7.1: Caso de Prueba 1

CP-2	
Descripción	Comprobación del estado de la interfaz durante el proceso de comprobación de redes sociales.
Resultado esperado	Al iniciar el proceso, se desactivará el botón *Scan* y se irá actualizando el panel de progreso a medida que el proceso avanza.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
Resultados obtenidos	
Ejecución	**Resultado**
1	La interfaz no responde durante el proceso de extracción.
2	No se actualiza el panel de progreso.
3	Resultado esperado.

Cuadro 7.2: Caso de Prueba 2

CP-3	
Descripción	Comprobación del estado de la interfaz al finalizar el proceso de comprobación de redes sociales.
Resultado esperado	Al terminar la comprobación, el panel de progreso muestra que el proceso ha terminado y se activa el botón *Scan*.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
Resultados obtenidos	
Ejecución	**Resultado**
1	No se reactiva el botón de *Scan*.
2	Resultado esperado.

Cuadro 7.3: Caso de Prueba 3

CP-4	
Descripción	Comprobación de que la interfaz se mantiene funcional durante la ejecución del proceso de comprobación de redes sociales.
Resultado esperado	Durante el proceso se permite cambiar a la pestaña del historial sin problemas.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Se cambia a la pestaña del historial.
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.4: Caso de Prueba 4

CP-5	
Descripción	Prueba de coherencia de los resultados del proceso de comprobación de redes sociales.
Resultado esperado	Se demuestra que el correo electrónico posee cuenta en los servicios comprobados.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Se comprueba manualmente la dirección de correo introducida en los servicios utilizados.
Resultados obtenidos	
Ejecución	**Resultado**
1	El proceso no devuelve que exista ninguna cuenta (cuando se ha comprobado que sí).
2	No se identifica ninguna cuenta de Instagram (comprobado que sí hay).
3	Resultado esperado.

Cuadro 7.5: Caso de Prueba 5

CP-6	
Descripción	Prueba de consistencia de los resultados del proceso de comprobación de redes sociales.
Resultado esperado	Los resultados de sucesivas comprobaciones utilizando el mismo correo electrónico no varían.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Al finalizar el proceso, se pulsa de nuevo el botón *Scan*
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.6: Caso de Prueba 6

CP-7	
Descripción	Prueba del tratamiento de correos electrónicos inválidos.
Resultado esperado	Se notifica al usuario que el correo introducido no es válido.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico inválida en el campo de texto.
2	Se pulsa el botón *Scan*.
Resultados obtenidos	
Ejecución	**Resultado**
1	El proceso inicia con una dirección de correo electrónico inválida.
2	Resultado esperado.

Cuadro 7.7: Caso de Prueba 7

CP-8	
Descripción	Prueba del proceso de comprobación de redes sociales usando el modo gráfico.
Resultado esperado	Al iniciar el proceso se abre la ventana del navegador y muestra los distintos pasos del proceso.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
Resultados obtenidos	
Ejecución	**Resultado**
1	La opción gráfica en la interfaz no se activa.
2	Resultado esperado.

Cuadro 7.8: Caso de Prueba 8

CP-9	
Descripción	Comprobación de que el modo gráfico no influye en los resultados del proceso.
Resultado esperado	No hay diferencias en los resultados entre 2 ejecuciones sucesivas alternando el modo gráfico.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Al finalizar se marca la casilla *Headless* y se pulsa de nuevo en el botón *Scan*.
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.9: Caso de Prueba 9

CP-10	
Descripción	Comprobación de que el modo gráfico no influye en la funcionalidad de la interfaz durante la ejecución del proceso.
Resultado esperado	Se puede minimizar la ventana del navegador e interactuar con la interfaz de manera normal.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Al abrirse la ventana del navegador, se puede minimizar e interactuar con la interfaz del plugin.
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.10: Caso de Prueba 10

CP-11	
Descripción	Comprobación de la consulta de correos electrónicos a FOCA.
Resultado esperado	Se muestra al usuario los correos electrónicos encontrados por FOCA.
Secuencia de acciones	
Paso	**Acción**
1	Se pulsa el botón *Connect*.
Resultados obtenidos	
Ejecución	**Resultado**
1	No se encuentra la instancia referenciada por el *connection string* especificado.
2	Resultado esperado.

Cuadro 7.11: Caso de Prueba 11

CP-12	
Descripción	Comprobación de coherencia en consulta de correos electrónicos a FOCA.
Resultado esperado	Los correos electrónicos mostrados al usuario son los guardados en la instancia de base de datos que utiliza FOCA.
Secuencia de acciones	
Paso	**Acción**
1	Se pulsa el botón *Connect*.
2	Se comprueban los correos electrónicos en la instancia de FOCA.
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.12: Caso de Prueba 12

CP-13	
Descripción	Comprobación de consulta durante un proceso de comprobación de redes sociales.
Resultado esperado	La consulta se realiza correctamente.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Se pulsa el botón *Connect*.
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.13: Caso de Prueba 13

CP-14	
Descripción	Comprobación de la utilización de los correos electrónicos consultados en el proceso de comprobación de redes sociales.
Resultado esperado	Se extrae la información sobre el correo electrónico correctamente.
Secuencia de acciones	
Paso	**Acción**
1	Se pulsa el botón *Connect*.
2	Se selecciona un correo electrónico del listado (doble *click*).
3	Se pulsa el botón *Scan*
Resultados obtenidos	
Ejecución	**Resultado**
1	No se selecciona el correo electrónico elegido del listado.
2	Se selecciona el correo pero no se utiliza en el proceso de comprobación.
3	Resultado esperado.

Cuadro 7.14: Caso de Prueba 14

CP-15	
Descripción	Comprobación de actualización del historial.
Resultado esperado	La extracción realizada se muestra correctamente.
Secuencia de acciones	
Paso	**Acción**
1	Se introduce una dirección de correo electrónico en el campo de texto.
2	Se pulsa el botón *Scan*.
3	Al finalizar, se cambia a la pestaña del historial.
Resultados obtenidos	
Ejecución	**Resultado**
1	Resultado esperado.

Cuadro 7.15: Caso de Prueba 15

CP-16	
Descripción	Comprobación de la interfaz del historial con múltiples entradas.
Resultado esperado	La interfaz muestra correctamente la información al introducir múltiples comprobaciones de redes sociales en el historial.
Secuencia de acciones	
Paso	**Acción**
1	Se realizan 10 extracciones.
2	Al finalizar, se cambia a la pestaña del historial.
Resultados obtenidos	
Ejecución	**Resultado**
1	Se sobrescribía la primera entrada del histórico al realizar múltiples extracciones.
2	Resultado esperado.

Cuadro 7.16: Caso de Prueba 16

CP-17	
Descripción	Comprobación de la exportación del histórico a CSV.
Resultado esperado	El CSV recoge correctamente la información guardada en el histórico.
Secuencia de acciones	
Paso	**Acción**
1	Se realizan 10 extracciones.
2	Al finalizar, se cambia a la pestaña del historial.
3	Se pulsa el botón *Export*.
Resultados obtenidos	
Ejecución	**Resultado**
1	Solo se rellenaba 1 entrada en el archivo CSV.
2	Resultado esperado.

Cuadro 7.17: Caso de Prueba 17

Capítulo 8

Manual de instalación y mantenimiento

Este capítulo contiene una guía que orienta a los usuarios en el proceso de instalación del plugin, integración con FOCA y mantenimiento del software. Incluye también una explicación de la estructura del código fuente para facilitar su modificación.

8.1. Manual de instalación

En esta sección se describirá el proceso de instalación necesario para utilizar el plugin en FOCA. También detallará las pautas necesarias para realizar modificaciones en el código fuente y recompilar el proyecto. Este procedimiento asume que se disponen de las herramientas explicadas en el capítulo de Preparación del Proyecto (ver capítulo 3), especialmente Visual Studio. El procedimiento de instalación de FOCA también está descrito en el capítulo de Preparación del Proyecto (ver apartado 3.1.3)

Para poder utilizar el plugin dentro de FOCA es necesario que este esté compilado como una librería dinámica (ver figura 8.1). Además, es recomendable que esté compilado utilizando la misma versión de compilador utilizado para compilar el proyecto de FOCA. Es recomendable utilizar Visual Studio en la compilación y desarrollo, pues los archivos de solución de Visual Studio tanto de FOCA como del plugin están disponibles en el repositorio de la herramienta y en el código fuente, respectivamente.

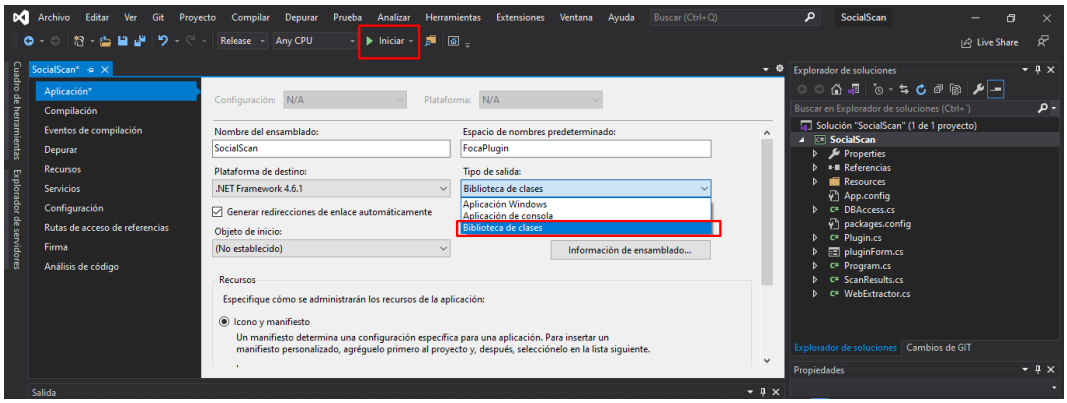


Figura 8.1: Compilación del código fuente del plugin en DLL usando Visual Studio

Además, el plugin utiliza Selenium, una librería que maneja diversos controladores de navegadores en formato ejecutable. Estos ejecutables están disponibles en la carpeta `./bin` del código fuente del plugin (colocado allí por Visual Studio al instalar los paquetes NuGet de Selenium) y hay que moverlos a la

carpeta *./FOCA/bin/Release*, donde el plugin (ya integrado en FOCA) podrá encontrarlos. Realizar los mismos pasos para los archivos `./bin/Release/WebDriver.dll` y `./bin/Release/WebDriver.xml` de Selenium. Aunque ya viene incluido por defecto, también hay que disponer de una copia de la librería dinámica de la API de plugins de FOCA (*PluginsAPI.dll*) disponible en el repositorio de la propia herramienta. En la figura 8.2 se encuentran los contenidos de `./FOCA/bin/Release` con los archivos necesarios para el funcionamiento del plugin resaltados.

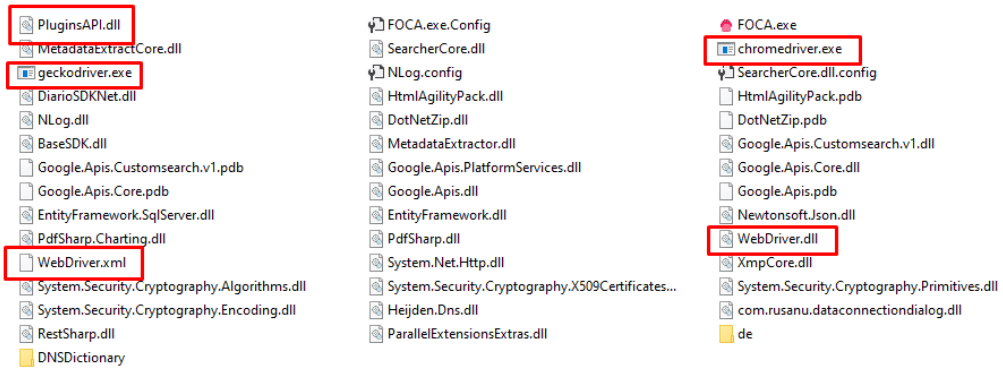


Figura 8.2: Contenido de `./FOCA/bin/Release`

Una vez resueltas todas las dependencias podemos ejecutar FOCA y cargar el plugin. El ejecutable de la herramienta se encuentra en `./FOCA/bin/Release/FOCA.exe`. Al ejecutar por primera vez, FOCA solicitará el *connection string* de una instancia SQL Server ya configurada (ver apartado 3.2.1) para crear en ella su esquema de datos. Tras esto aparecerá la interfaz principal de la herramienta. Para cargar el plugin en FOCA, hay que seleccionar *Load/Unload plugins* en el menú desplegable *Plugins* (ver figura 8.3), pulsar en *Load new plugin* y buscar el archivo DLL resultado de la compilación del plugin mediante el explorador de archivos que aparece (localizado dentro del proyecto del plugin en `./bin/Release`, ver figura 8.5). Tras cargar el plugin, aparecerá en el menú desplegable *Plugins* una nueva entrada con el nombre e icono del plugin, desde donde se podrá lanzar (ver figura 8.5).

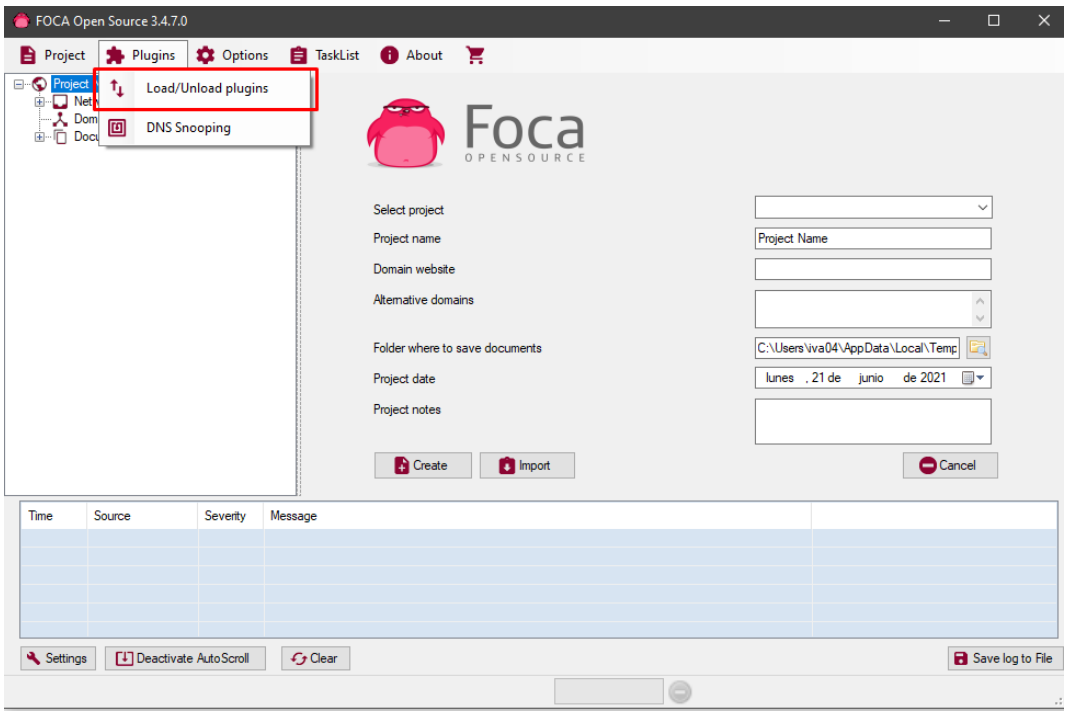


Figura 8.3: Interfaz principal de FOCA con el menú *Plugins*

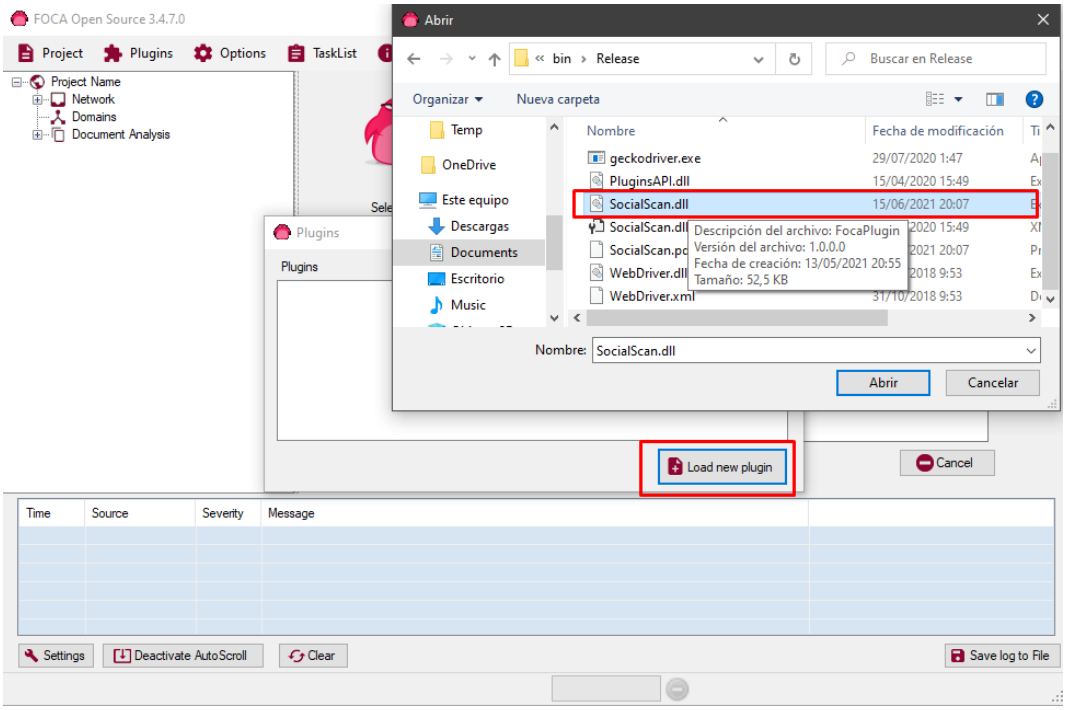


Figura 8.4: Carga de un plugin - FOCA

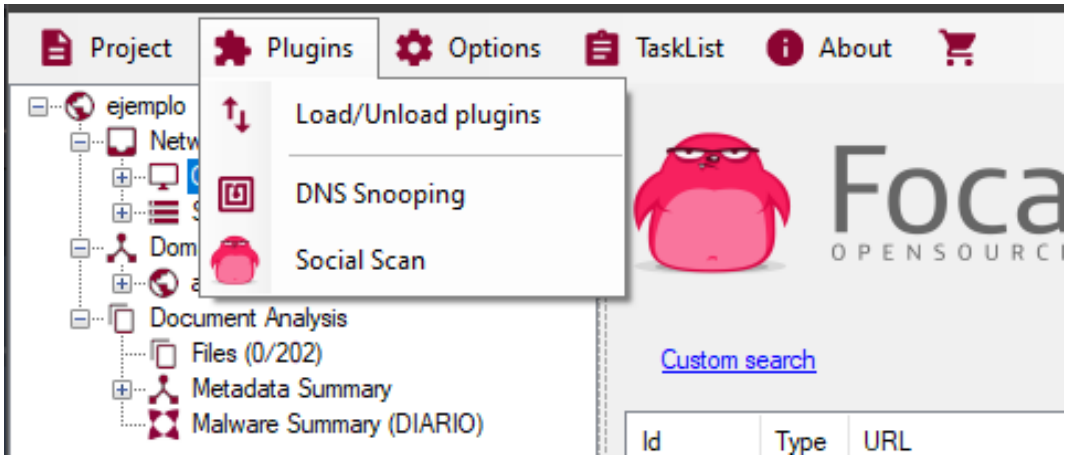


Figura 8.5: Menú *Plugins* con el plugin cargado

8.2. Estructura de archivos del proyecto

En este apartado se explicará la distribución del código fuente y qué funcionalidad del plugin se maneja en qué clase. Todo el código fuente se encuentra en `./SocialScan/src`.

- `UI/pluginForm.cs`
  Es la clase que contiene la interfaz gráfica. Maneja los componentes de Windows Forms (*pluginForm.cs.Designer*) y gestiona las llamadas a cada evento junto con sus *handlers*, incluidas las tareas en segundo plano.

- `Controladores/Program.cs`
  Es el punto de inicio de la aplicación. Se encarga de instanciar la clase que maneja toda la funcionalidad del plugin.

- `Controladores/Plugin.cs`
  La clase Plugin es la encargada de toda la interacción con la API de plugins de FOCA. Define el

nombre, descripción, icono del plugin así como los elementos que importa o exporta a la herramienta (por ejemplo, el panel con la interfaz gráfica del plugin).

- `Controladores/RRSSController.cs`
  Es la clase que recoge la lógica de la aplicación referente al proceso de comprobación de redes sociales.

- `Controladores/HistoryController.cs`
  Es la clase que recoge la lógica de la aplicación referente al histórico de comprobaciones de redes sociales realizadas.

- `Controladores/WebExtractor.cs`
  Es la clase que maneja la interacción con los servicios de inicio de sesión de las redes sociales y extrae la información. Incluye la configuración de los controladores de Selenium y el proceso de interacción con los servicios web paso a paso.

- `Data/ScanResults.cs`
  Se encarga del modelo de datos utilizado por la aplicación. Incluye la clase ScanResults (modelo de la información de cuentas de usuario en redes sociales encontradas sobre una dirección de correo electrónico) y ServiceResult (modelo de la información sobre la existencia de una cuenta de usuario de un correo electrónico determinado en una red social determinada).

- `Data/DBAccess.cs`
  Esta clase es la encargada de manejar la interacción con la base de datos de FOCA. Sirve de capa de abstracción sobre los esquemas de datos implementados en SQL Server.

- `Resources`
  En esta carpeta están guardados los recursos gráficos utilizados por el plugin, en este caso, solamente iconos.

- `Properties` y `Referencias`
  Apartados generados por Visual Studio que contienen la configuración de la solución y sus librerías.

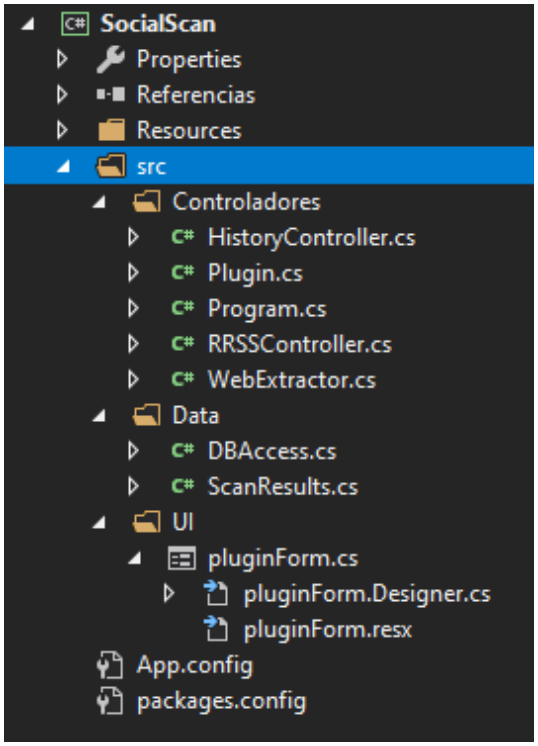


Figura 8.6: Estructura de archivos del proyecto

Capítulo 9

Conclusiones y trabajo futuro

En este capítulo se hará una recapitulación del trabajo realizado. Se expondrán los objetivos cumplidos durante el desarrollo del TFG y se detallarán las ampliaciones de funcionalidad que se podrían llevar a cabo a futuro.

9.1. Conclusiones

Durante la realización de este Trabajo de Fin de Grado se ha desarrollado un plugin para la herramienta FOCA capaz de utilizar los correos electrónicos encontrados por dicha herramienta para recopilar información sobre sus cuentas de usuario en distintas redes sociales y servicios. Pese a haber sufrido un cambio de objetivos, la finalidad de este TFG se sigue manteniendo: explotar la capacidad de FOCA de encontrar metadatos dentro de documentos ofimáticos de Internet para usarlos como punto de partida en la búsqueda de más información (un recurso muy utilizado en técnicas de *Profiling*). A continuación se resume el trabajo realizado:

- Planificación y ejecución de un proyecto de desarrollo software.
- Estudio de las funcionalidades que ofrece FOCA y su programación, incluyendo su API de plugins.
- Estudio del modelo de datos implementado por FOCA en las instancias anfitrionas.
- Desarrollo de un plugin para FOCA que añade funcionalidad a la herramienta.
- Familiarización con las distintas herramientas de desarrollo para aplicaciones de Windows.
- Manejo de varias librerías y herramientas de *Web scraping* y *Headless browsers*.

9.2. Trabajo futuro

En este aparado se exponen las posibles mejoras o ampliaciones de funcionalidad que se podrían implementar en versiones futuras del plugin. Las principales son:

- Mejorar la integración del plugin con FOCA, implementando otras maneras de utilizar las funciones ofrecidas desde la propia herramienta.
- Añadir soporte para más fuentes de correos electrónicos, no sólo bases de datos SQL Server (otros DBMS, ficheros CSV, JSON, XML, etc).

- Ampliar el catálogo de redes sociales utilizadas. Pese a ser un proceso muy individualizado, la capacidad de permitir a los usuarios añadir nuevas redes sociales al proceso de extracción de información aumentaría mucho la utilidad del mismo.
- Utilizar un mecanismo de colas de operaciones como el que ofrece FOCA para mejorar la eficiencia de extracción de información (esto daría pie a ejecuciones *batch* para comprobar cuentas de redes sociales de varios correos electrónicos sin tener que interactuar con la interfaz para cada una de ellas).

Bibliografía

[1] Hughes, B., & Cotterell, M. (2009). Software Project Management (5th Revised ed.). McGraw-Hill Education.

[2] Precio promedio de luz en España. (s. f.). Tarifasgasluz. Recuperado 20 de enero de 2021, de https://tarifasgasluz.com/comparador/precio-kwh

[3] Uso promedio de luz en España. (s. f.). Tarifasgasluz. Recuperado 20 de enero de 2021, de https://tarifasgasluz.com/faq/cuanto-cuesta-luz-mes

[4] Cuánta electricidad consume un ordenador. (s. f.). CHC Energía. Recuperado 20 de enero de 2021, de https://chcenergia.es/blog/cuanto-consume-un-ordenador-o-pc/

[5] BOE.es - BOE-A-2020-11043 Real Decreto-ley 28/2020, de 22 de septiembre, de trabajo a distancia. (2020, 22 septiembre). Boletín Oficial del Estado. https://www.boe.es/eli/es/rdl/2020/09/22/28/con

[6] FOCA. (s. f.). ElevenPaths. Recuperado 24 de enero de 2021, de https://www.elevenpaths.com/innovation-labs/technologies/foca

[7] Riley, J. (s. f.). Understanding Metadata. NISO. Recuperado 24 de enero de 2021, de https://groups.niso.org/apps/group_public/download.php/17446/Understanding%20Metadata.pdf

[8] ElevenPaths/FOCA. (s. f.). GitHub. Recuperado 24 de enero de 2021, de https://github.com/ElevenPaths/FOCA

[9] Alonso, C. (s. f.). Foca API v0.1. Slideshare. Recuperado 26 de enero de 2021, de https://es.slideshare.net/chemai64/foca-api-v01

[10] Documentación de Entity Framework. (s. f.). Microsoft Docs. Recuperado 27 de enero de 2021, de https://docs.microsoft.com/es-es/ef/

[11] Microsoft. (s. f.). Plataforma de datos de Microsoft. Recuperado 27 de enero de 2021, de https://www.microsoft.com/es-es/sql-server

[12] SQL Server 2016 y 2017: Requisitos de hardware y de software - SQL Server. (s. f.). Microsoft Docs. Recuperado 27 de enero de 2021, de https://docs.microsoft.com/es-es/sql/sql-server/install/hardware-and-software-requirements-for-installing-sql-server?view=sql-server-ver15

[13] Microsoft. (s. f.-a). Descargas de SQL Server. Recuperado 27 de enero de 2021, de https://www.microsoft.com/es-es/sql-server/sql-server-downloads

[14] Instalación mediante la interfaz gráfica de usuario - SQL Server. (s. f.). Microsoft Docs. Recuperado 27 de enero de 2021, de https://docs.microsoft.com/es-es/sql/database-engine/install-windows/install-sql-server-from-the-installation-wizard-setup?view=sql-server-ver15

[15] SqlLocalDB (utilidad) - SQL Server. (s. f.). Microsoft Docs. Recuperado 30 de enero de 2021, de https://docs.microsoft.com/es-es/sql/tools/sqllocaldb-utility?view=sql-server-ver15

[16] Desinstalación de una instancia existente - SQL Server. (s. f.). Microsoft Docs. Recuperado 30 de enero de 2021, de https://docs.microsoft.com/es-es/sql/sql-server/install/uninstall-an-existing-instance-of-sql-server-setup?view=sql-server-ver15&tabs=Windows10

[17] SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS). (s. f.). Microsoft Docs. Recuperado 10 de febrero de 2021, de https://docs.microsoft.com/es-es/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15

[18] Descarga e instalación de Azure Data Studio - Azure Data Studio. (s. f.). Microsoft Docs. Recuperado 10 de febrero de 2021, de https://docs.microsoft.com/es-es/sql/azure-data-studio/download-azure-data-studio?view=sql-server-ver15

[19] SSMS Installation error (0x80070643). (s. f.). Microsoft Forums. Recuperado 11 de febrero de 2021, de https://social.msdn.microsoft.com/Forums/sqlserver/en-US/183f4353-b0d7-451f-aa7f-0764cfc6254e/ssms-installation-error-0x80070643?forum=sqltools

[20] Robledano, A. (s. f.). Qué es NET Framework. OpenWebinars.net. Recuperado 12 de febrero de 2021, de https://openwebinars.net/blog/que-es-net-framework/

[21] Microsoft. (s. f.-b). Download .NET Framework 4.7.2 — Free official downloads. Recuperado 12 de febrero de 2021, de https://dotnet.microsoft.com/download/dotnet-framework/net472

[22] Microsoft. (s. f.-a). Descargar Visual Studio 2019 para Windows y Mac. Visual Studio. Recuperado 13 de febrero de 2021, de https://visualstudio.microsoft.com/es/downloads/

[23] Microsoft. (s. f.-e). Visual Studio Community 2019 - Free IDE and Developer Tools. Visual Studio. Recuperado 13 de febrero de 2021, de https://visualstudio.microsoft.com/es/vs/community/

[24] Requisitos del sistema de Visual Studio 2019. (s. f.). Microsoft Docs. Recuperado 13 de febrero de 2021, de https://docs.microsoft.com/es-es/visualstudio/releases/2019/system-requirements

[25] Instalar Visual Studio. (s. f.). Microsoft Docs. Recuperado 14 de febrero de 2021, de https://docs.microsoft.com/es-es/visualstudio/install/install-visual-studio?view=vs-2019

[26] Marqués, J. M. (2020). Diapositivas de la asignatura Diseño, Integración y Adaptación de Software. [Diapositivas].

[27] Larman, C. (2001). Applying Uml and Patterns: An Introduction to Object-Oriented Analysis and Design, and the Unified Process (2.a ed.). Prentice Hall.

[28] Astah. (2020). [Herramienta de modelado UML]. Astah. https://astah.net/

[29] Wireframe.cc — The go-to wireframing tool. (s. f.). Wireframe.Cc. Recuperado 10 de marzo de 2021, de https://wireframe.cc

[30] Editor de diagramas en línea gratis. (s. f.). Visual Paradigm. Recuperado 10 de marzo de 2021, de https://online.visual-paradigm.com/es/diagrams/solutions/free-online-diagram-editor/

[31] jQuery.ajax() — jQuery API Documentation. (s. f.). JQuery API. Recuperado 7 de abril de 2021, de https://api.jquery.com/jquery.ajax/

[32] SeleniumHQ Browser Automation. (s. f.). Selenium. Recuperado 8 de abril de 2021, de https://www.selenium.dev/

[33] Selenium.WebDriver 3.141.0. (s. f.). NuGet. Recuperado 8 de abril de 2021, de https://www.nuget.org/packages/Selenium.WebDriver/3.141.0

[34] WebDriver - Table of Content. (s. f.). Selenium. Recuperado 8 de abril de 2021, de https://www.selenium.dev/selenium/docs/api/dotnet/

[35] GMail is blocking login via Automation (Selenium). (s. f.). Stack Overflow. Recuperado 15 de abril de 2021, de https://stackoverflow.com/questions/57602974/gmail-is-blocking-login-via-automation-selenium

[36] Puppeteer — Tools for Web Developers —. (s. f.). Google Developers. Recuperado 16 de abril de 2021, de https://developers.google.com/web/tools/puppeteer

[37] hardkoded/puppeteer-sharp. (s. f.). GitHub. Recuperado 16 de abril de 2021, de https://github.com/hardkoded/puppeteer-sharp

[38] Puppeteer Sharp. (s. f.). Puppeteer Sharp. Recuperado 16 de abril de 2021, de http://www.puppeteersharp.com/api/index.html

[39] PuppeteerSharp 4.0.0. (s. f.). NuGet. Recuperado 16 de abril de 2021, de https://www.nuget.org/packages/PuppeteerSharp/

[40] Selenium Google Login Blocked in Automation. (2021, 18 abril). Stack Overflow. https://stackoverflow.com/questions/67150869/selenium-google-login-blocked-in-automation-self-answered-bypassed-the-google

[41] selenium-stealth. (s. f.). PyPI. Recuperado 18 de abril de 2021, de https://pypi.org/project/selenium-stealth/

[42] Usuarios de redes sociales en España. (s. f.). EPData. Recuperado 23 de abril de 2021, de https://www.epdata.es/datos/usuarios-redes-sociales-espana-estudio-iab/382

[43] Password Cracking Dictionary. (s. f.). CrackStation. Recuperado 26 de abril de 2021, de https://crackstation.net/crackstation-wordlist-password-cracking-dictionary.htm

[44] John the Ripper password cracker. (s. f.). OpenWall. Recuperado 26 de abril de 2021, de https://www.openwall.com/john/

[45] Fowler, M. (s. f.). PresentationDomainDataLayering. martinfowler.com. Recuperado 30 de abril de 2021, de https://martinfowler.com/bliki/PresentationDomainDataLayering.html

[46] What is DNS cache snooping? (s. f.). Internet Systems Consortium (ISC). Recuperado 4 de mayo de 2021, de https://kb.isc.org/docs/aa-00509

[47] rfc4180. (s. f.). IETF. Recuperado 6 de mayo de 2021, de https://datatracker.ietf.org/doc/html/rfc4180

[48] By. (s. f.). Selenium. Recuperado 13 de mayo de 2021, de https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/By.html

[49] XML Path Language (XPath) 3.1. (s. f.). W3.Org. Recuperado 15 de mayo de 2021, de https://www.w3.org/TR/2017/REC-xpath-31-20170321/

[50] Documentación de Windows Forms para .NET. (s. f.). Microsoft Docs. Recuperado 20 de mayo de 2021, de https://docs.microsoft.com/es-es/dotnet/desktop/winforms/?view=netframeworkdesktop-4.8

[51] BackgroundWorker Clase (System.ComponentModel). (s. f.). Microsoft Docs. Recuperado 20 de mayo de 2021, de https://docs.microsoft.com/es-es/dotnet/api/system.componentmodel.backgroundworker?view=net-5.0

[52] SqlConnection Clase (System.Data.SqlClient). (s. f.). Microsoft Docs. Recuperado 21 de mayo de 2021, de https://docs.microsoft.com/es-es/dotnet/api/system.data.sqlclient.sqlconnection?view=dotnet-plat-ext-5.0

[53] System.Data.SqlClient 4.8.2. (s. f.). NuGet. Recuperado 21 de mayo de 2021, de https://www.nuget.org/packages/System.Data.SqlClient