Sistema de Recuperación de información para la asociación semi-automática de archivos ejecutables a su aplicación software correspondiente

Jalbersson Guillermo Plazas Londoño

Director: PhD. Carlos Alberto Cobos Lozada

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Línea de Investigación en Gestión de la Información y Sistemas Inteligentes
Popayán, noviembre de 2018

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

LISTADO DE ANEXOS

Anexo 1: Prototipo Software, código fuente y ejecutables relacionados con el SRI.

Anexo 2: Descripción detallada de la base de datos usada en el SRI.

Anexo 3: Mockups de los diseños de la Herramienta para el experto.

Anexo 4: Manual de usuario de la Herramienta para el experto.

Anexo 5: Resultados del test de usabilidad y funcionalidad aplicado.

Anexo 6: Artículo Sistema de Recuperación de información para la asociación semi-automática de archivos ejecutables a su aplicación software correspondiente

Dedicatoria

Dedico este trabajo a mi familia que siempre ha estado apoyándome en las buenas y las malas.

Agradecimientos

Agradezco a todos aquellos que ayudaron a lograr el desarrollo de este proyecto con su conocimiento, críticas, pruebas e incluso su apoyo moral, en especial al PhD. Carlos Alberto Cobos que me guio en todo momento y siempre tuvo algo constructivo que opinar acerca de lo que se estuviera realizando en el proyecto. También a Aranda Software S. A. S., en particular a Álvaro Carmona y Hugo Armando Castellanos que me ayudaron no sólo a lograr un buen resultado final sino también a crecer profesionalmente.

Resumen. Dentro de las actividades que suele realizar una empresa legalmente constituida, existen unas relacionadas con la revisión de las licencias del software usado en sus equipos de cómputo, esto con el fin de evitar problemas relacionados con la violación de derechos de autor, sanciones monetarias y problemas de seguridad, entre otros. Para poder realizar la revisión de las licencias de software las empresas deben hacer una revisión del software instalado y que se está ejecutando en determinado momento en cada uno de los equipos. Realizar esta tarea de forma manual puede volverse una actividad muy compleja, demorada o simplemente inmanejable y costosa dependiendo del número de computadores y aplicaciones software con los que cuente la empresa. En un esfuerzo por encontrar una solución en este proyecto se propone crear un Sistema de Recuperación de Información (SRI) que permita realizar la gestión de recursos software. Dentro de los componentes que tiene el SRI está primero, un repositorio de ejecutables y las aplicaciones a las cuales pertenecen, un algoritmo de clasificación basado en K-NN y en caso de que no se pueda realizar una clasificación automática se propone un esquema de visualización de resultados por grupos y un ranking dentro de los grupos para que un experto realice la clasificación de forma manual. Los componentes fueron desarrollados y complementados con otros componentes que se consideraron necesarios para el correcto funcionamiento del SRI. Se hicieron pruebas con datos de 20 computadores para comprobar la precisión del clasificador y se aplicó un test de usabilidad y funcionalidad sobre el SRI con nueve personas que dieron sus opiniones sobre lo que está bien y lo que podría mejorar en la propuesta.

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

En la actualidad, las empresas medianas y grandes tienen una gran cantidad de equipos de cómputo para soportar sus labores diarias y para estas tareas utilizan diferentes aplicaciones software dependiendo del área de negocio en la que se desenvuelven. Dentro de estas actividades que suele realizar una empresa, son de especial interés para este proyecto, las relacionadas con la revisión de las licencias del software usado en sus equipos de cómputo, esto con el fin de evitar problemas relacionados con la violación de derechos de autor, sanciones monetarias y problemas de seguridad, entre otros.

Para poder realizar la revisión de las licencias de software las empresas deben hacer una revisión del software instalado y que se está ejecutando en determinado momento en cada uno de los equipos. Realizar esta tarea de forma manual puede volverse una actividad muy compleja, demorada o simplemente inmanejable y costosa dependiendo del número de computadores y aplicaciones software con los que cuente la empresa.

En un proceso de asociación manual ejecutable/aplicación se deben tener en cuenta diferentes situaciones, a saber:

- Una aplicación software puede contar con diversos archivos como librerías de enlace dinámico (ddl), archivos XML o archivos ejecutables (exe), siendo estos últimos los de mayor relevancia en el control de las licencias.
- Al inspeccionar los archivos ejecutables haciendo uso de un editor hexadecimal, se puede conocer información de este como por ejemplo la empresa que lo desarrolló, su versión e información referente a sus derechos de autor, datos que no se encuentran expuestos a simple vista.
- Una aplicación puede tener varias versiones, como es el caso de Visual Studio 2010 y Visual Studio 2012.
- Un ejecutable puede pertenecer a diferentes suites o soluciones, por ejemplo winword.exe puede pertenecer a la solución Home and Student o a la Professional Plus de Microsoft Office.
- El software instalado o en ejecución puede o no incluir información en el registro del sistema.
- Cuando un usuario instala una aplicación puede cambiar la ubicación por defecto de esta.
- Contar con una firma (hash del contenido del archivo ejecutable) aumenta la precisión en la identificación de ejecutables que se tengan previamente registrados, pero no aporta información en la identificación de los que no están

registrados y por eso se requiere de un proceso con información complementaria que permita su identificación.

Para realizar el proceso de inventariar el software instalado y usado en los equipos de una compañía, las empresas buscan desarrollar o adquirir un sistema que asocie de forma semi-automática (es virtualmente imposible conocer todo el software existente, pues está en constante crecimiento y evolución) los ejecutables hallados en un computador, a la aplicación a la cual pertenecen.

Aranda Software Andina S.A.S. considera este tipo de funcionalidad clave para sus clientes y por esto se interesó en incluirla en uno de sus productos, esta funcionalidad es conocida en la industria de software como ***Gestión de Recursos Software (GRS)***. En este orden de ideas, Aranda a la fecha ha hecho un desarrollo en el que usando las características de los archivos ejecutables hace una asociación mediante fórmulas ponderadas de dichas características con el software instalado que es reportado por el sistema operativo y de esta forma evita la dependencia de una base de datos externa que es costosa para la compañía, sin embargo, los resultados alcanzados a la fecha no cumplen con el criterio de éxito (tasa de aciertos y disminución de intervención humana) definido por la compañía. Como una forma de solución a esto se plantó en este proyecto crear un repositorio propio de asociaciones ejecutables/aplicaciones para que sirva como conocimiento base para un algoritmo de clasificación que realice la asociación de un ejecutable con su respectiva aplicación.

Como no es posible garantizar que el algoritmo de clasificación haga la asociación automática de cualquier ejecutable y software que reciba como parámetro de entrada, se hace necesaria la intervención de un humano experto en cierto punto del proceso, bien sea en la clasificación o en la actualización del repositorio.

Debido a lo anterior, en el presente trabajo de grado se definieron las siguientes preguntas ¿Cuál es la mejor estructura de una base de datos (repositorio) de ejecutables y aplicaciones que permita soportar el proceso semi-automático de clasificación? Si un ejecutable que se necesita asociar con una aplicación es totalmente desconocido ¿qué estrategias se pueden usar para solventar la falta de información? Y ¿cuál es la mejor manera de apoyar el trabajo de un experto en caso de ser necesaria su intervención en el proceso de asociación?

Para resolver estas preguntas, en este proyecto se buscó crear un Sistema de Recuperación de Información que permita la gestión de recursos software. Dentro de los componentes que tiene el SRI está primero, un repositorio de ejecutables y las aplicaciones a las cuales pertenecen. Estos datos son el componente principal para la asociación de los ejecutables que se encuentren en un computador y el software del que hacen parte. Como algoritmo de búsqueda se usó una adaptación del algoritmo K-nn, que sirve para la clasificación semi-automática de los ejecutables cuyos datos ya se encuentren en el repositorio. En caso de que no se pueda realizar una clasificación automática se usa un esquema de visualización de resultados por grupos y un ranking dentro de los grupos para que un experto

realice la clasificación de forma manual y esto alimente el repositorio principal, para que en casos futuros su clasificación pase a ser automática. La interfaz gráfica de usuario muestra sólo los casos que requieran la intervención del experto.

En este trabajo, una necesidad empresarial se convirtió en un proyecto de investigación aplicada en el que las técnicas de recuperación de información, clasificación y agrupación se integraron y fueron puestos a prueba para evaluar su función en el problema de la gestión de recursos software.

1.2 APORTES

Desde la perspectiva de investigación las contribuciones de este proyecto se encaminaron a generar nuevo conocimiento, centrado en definir los criterios de similitud que hacen que un algoritmo de clasificación y un componente de agrupación trabajen en conjunto en un sistema de recuperación de información que, en este caso, permite asociar un archivo ejecutable con su aplicación software correspondiente. Para definir estos criterios de similitud se evaluaron diferentes atributos o características de un archivo ejecutable como su tamaño, nombre, ubicación, contenido y otra información del registro del sistema que generalmente se puede observar en el panel de control.

Por otro lado, también se realizó una revisión e implementación de un mecanismo que permite organizar y ordenar los resultados encontrados en el SRI, usando un algoritmo sencillo de agrupación, para facilitar la visualización de los resultados que no se puedan clasificar automáticamente y de esta forma proporcionar herramientas que ayuden al experto en su tarea de asociación ejecutable/software.

Desde la perspectiva de innovación, el SRI ha sido desarrollado e implementado como una nueva funcionalidad que se podrá incluir en una solución/producto de Aranda Software u otra compañía haciendo que este producto sea el primero en Colombia en incluir conceptos de Gestión de Recursos Software y de esta forma, además, permitirle a Aranda incursionar en un nuevo nicho de mercado a nivel internacional para su solución/producto.

1.3 OBJETIVOS

A continuación, se presentan los objetivos como fueron aprobados en el anteproyecto por parte del Consejo de Facultad de la Facultad de Ingeniería Electrónica y Telecomunicaciones.

1.3.1 Objetivo General

Proponer un sistema de recuperación de información para la asociación semi-automática de archivos ejecutables encontrados en un computador con la aplicación software a la cual pertenece buscando facilitar el inventario y licenciamiento de software en una organización.

1.3.2 Objetivos Específicos

- Modelar un repositorio de suites de software, aplicaciones y ejecutables en una base de datos relacional, cargarla con datos obtenidos de una base provista por Aranda Software S.A.S. y de un proceso de web scraping[1] en sitios web relacionados con el tema y establecer los mecanismos apropiados para su actualización en el marco del sistema de recuperación.
- Diseñar un algoritmo de clasificación que establezca automáticamente la asociación entre un archivo ejecutable encontrado en un computador con la aplicación software a la que pertenece y establecer la precisión del mismo (porcentaje de instancias correctamente clasificadas) usando el repositorio previamente definido e información extraída de por lo menos 20 computadores[2].
- Desarrollar una herramienta basada en visualización por grupos (clustering) para ayudar a un experto humano en la clasificación de los archivos ejecutables que el algoritmo no logró clasificar en forma automática.
- Integrar los componentes previamente mencionados en el marco de un sistema de recuperación de información desarrollado en VS.NET mediante SCRUM en un entorno cliente/servidor y aplicar un test de usabilidad y funcionalidad que permita medirlo.

1.4 RESULTADOS OBTENIDOS

- Monografía con el resumen del proyecto: corresponde al presente documento que contiene el estado del arte sobre los sistemas para la gestión de recursos software y la descripción de los componentes del sistema de recuperación de información, es decir, el repositorio, los algoritmos usados para la clasificación automática de los archivos ejecutables, la revisión del registro del sistema, el análisis de los directorios y la herramienta de visualización de grupos de aplicaciones que facilita el trabajo del experto. Finalmente, presenta la evaluación del sistema, las conclusiones y recomendaciones para el desarrollo de trabajos futuros en el área.
- Prototipo Software, código fuente, documentación y ejecutable del prototipo de Sistema de Recuperación de información desarrollado en el presente proyecto.
- Un artículo con los resultados del trabajo de grado a publicar en la página web del Departamento o en evento nacional o internacional. Para su elaboración se siguió el formato de la IEEE.

1.5 ORGANIZACIÓN DEL RESTO DEL DOCUMENTO

A continuación, se describe de manera general el contenido y organización de la presente monografía:

[1] Extracción automática de datos de una página web.

[2] Esta cantidad de 20 computadores se define para mantener el proyecto dentro del alcance definido para un trabajo de pregrado de la Universidad del Cauca, sin detrimento de la calidad de los resultados del trabajo a desarrollar.

CAPITULO 1: INTRODUCCIÓN: Hace referencia al presente capitulo que introduce el tema de investigación, presenta las preguntas de investigación que originaron el trabajo, los aportes del proyecto, también los objetivos (general y específicos) definidos en el anteproyecto, un breve resumen de los resultados obtenidos y finalmente la organización de la monografía.

CAPITULO 2: CONTEXTO TEÓRICO Y ESTADO DEL ARTE: En este capítulo se resume el contexto teórico de las temáticas integradas en la investigación, incluidas los Sistemas de Recuperación de Información, la clasificación como una de las tareas principales de la minería de datos, el algoritmo K-nn para clasificación de datos, las fuentes de información que se usaron para el proyecto incluyendo ProccessList.com que fue procesada con herramientas de web scraping, el registro de Windows y el Change Journal y algunos conceptos técnicos usados en el desarrollo del proyecto. Finalmente, en la sección del estado del arte se presentan algunos productos relacionados con la gestión de recursos software, tanto libres como propietario.

CAPITULO 3: PROPUESTA: Este capítulo presenta a nivel general, los diferentes componentes del SRI propuesto para resolver el problema de la asociación semiautomática de ejecutables con el software al que pertenecen. Se presenta la arquitectura y el despliegue de dichos componentes y los diferentes escenarios del proceso de clasificación que se soportan con el sistema.

CAPITULO 4: REPOSITORIO PRINCIPAL: Este capítulo presenta el trabajo realizado para obtener los datos del Repositorio principal, además de cómo se determinó cuáles eran los datos que aportaban más al proceso de clasificación. Se muestra el diseño de la base de datos usada para soportar el repositorio, la creación de una base de datos de apoyo y detalles de su implementación.

CAPITULO 5: CLASIFICADOR PRINCIPAL: En este capítulo se presenta el diseño e implementación del clasificador encargado de asociar los ejecutables con su software relacionado de manera automática. También se muestran algunos de los métodos más importantes para el clasificador y el resultado de su evaluación en un dataset creado para tal fin.

CAPITULO 6: HERRAMIENTA PARA EL EXPERTO: En este capítulo se presenta el diseño e implementación de la herramienta para el experto. Se muestra cómo el experto interactúa con la herramienta y cómo fue implementada la herramienta haciendo uso del patrón Modelo Vista Controlador.

CAPITULO 7: IMPLEMENTACIÓN DE LOS DEMÁS COMPONENTES DEL SRI: En este capítulo se detallan los demás componentes del SRI que fueron introducidos en el capítulo 3, a saber: el recolector de información (detallando el uso del change journal, el IRSTaskScheduler, el ExeDataExtractor y el IRSWCFWSClient) y el IRSWCFWebService. También se explica la forma como todos los componentes se unen para formar el sistema y se muestra aquellos detalles de la metodología de desarrollo Scrum aplicados durante el proyecto.

CAPITULO 8: EXPERIMENTACIÓN: En este capítulo se presentan los resultados de la aplicación del test de usabilidad y funcionalidad sobre el SRI.

CAPITULO 9: CONCLUSIONES Y TRABAJOS FUTUROS: En este capítulo se presentan las conclusiones obtenidas al finalizar el trabajo de grado e ideas que el grupo de investigación espera abordar como trabajo futuro.

CAPITULO 10: BIBLIOGRAFIA: Este último capítulo contiene las referencias bibliográficas de los artículos y libros consultados para la realización del proyecto.

CAPÍTULO 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

En esta sección se presentan algunos conceptos utilizados en este proyecto con el propósito de facilitar un mayor entendimiento de este. Luego, se presentan las principales herramientas software existentes a la fecha que soportan la gestión de recursos software.

2.1 SISTEMA DE RECUPERACIÓN DE INFORMACIÓN

En los últimos años se ha dado un incremento exponencial en el volumen y variedad de información que hay disponible, principalmente gracias a internet, y de la misma forma ha crecido la demanda por información precisa de acuerdo a los intereses de los usuarios [1]. Cumplir con las necesidades de información hace necesario el uso de métodos para obtener esa información de forma ágil y eficiente, es aquí donde el campo de la Recuperación de Información aparece para soportar dichas necesidades. Algunas de las muchas definiciones que se le han dado a la Recuperación de información incluyen:

- La recuperación de información intenta resolver el problema de encontrar y ordenar documentos relevantes que satisfagan la necesidad de información de un usuario [2].
- Recuperación de Información es el conjunto de tareas mediante las cuales el usuario localiza y accede a los recursos de información que son pertinentes para la resolución de un problema específico [3].
- Es la localización y presentación a un usuario de información relevante a una necesidad de información expresada como una pregunta [4].

Para este proyecto se tomó la segunda definición, debido a que es más completa y acorde con lo que se planea realizar.

A pesar de su reciente notoriedad, el campo de la Recuperación de Información no nació con el internet y los motores de búsqueda (Web Search Engine) como Google o Yahoo!. Este campo empezó con la búsqueda de publicaciones de artículos científicos pero pronto se extendió a otras formas de contenido, particularmente aquellas de profesionales de la información como reporteros, abogados y médicos [5]. Mucha de la investigación en Recuperación de información se ha hecho en estos campos y mucha de la práctica continua en Recuperación de Información está enfocada en proveer acceso a información no estructurada en varios dominios corporativos y gubernamentales.

Los Sistemas de Recuperación de Información (SRI) son sistemas que aplican métodos para facilitar el hallazgo de información en grandes conjuntos de datos y documentos, que se encargan de presentar los resultados a los usuarios de

acuerdo con ciertos criterios como, por ejemplo, la similitud de los resultados respecto a los términos de búsqueda, de acuerdo con el contexto, entre otros [2].

La arquitectura básica de un SRI se presenta en la **Figura 1**. El proceso inicia cuando un usuario hace una consulta a través de la interfaz de usuario. El sistema cuenta inicialmente con un repositorio en donde se encuentran los datos o documentos en los cuales se puede encontrar la respuesta a la necesidad de información del usuario, normalmente expresada en una consulta por palabras clave. Los datos/documentos del repositorio por lo general están indexados para acelerar su acceso. Esta indexación puede ser creada usando diferentes unidades (tokens, palabras, lemas, oraciones, ngramas, entre otros) y usando diferentes esquemas de ponderación, los cuales incluyen componentes como la frecuencia de las palabras, el orden de aparición de estas, la cantidad de veces que aparece una palabra en la colección de documentos, la longitud de los documentos, etc.

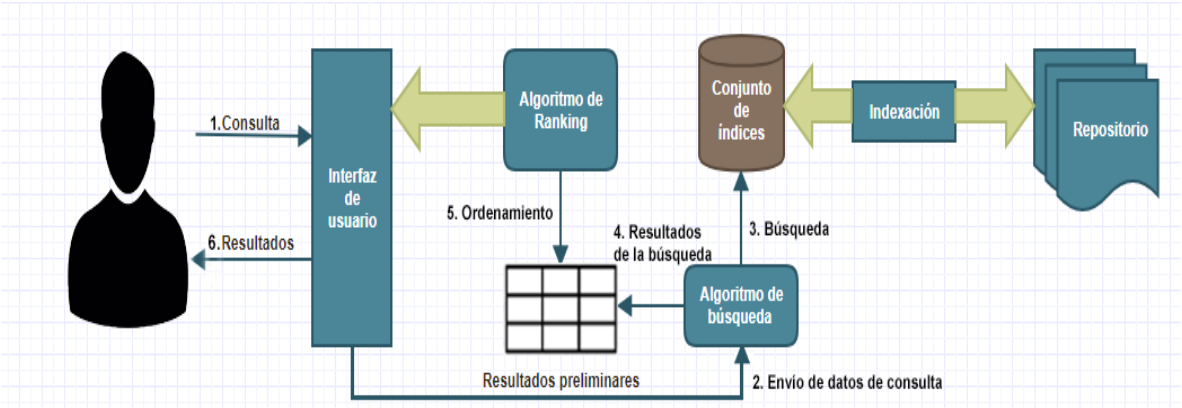

Figura 1 *Arquitectura básica de un Sistema de Recuperación de Información (Fuente propia)*

Una vez que los datos del repositorio se encuentran indexados, estos pueden ser gestionados por un algoritmo de búsqueda que genera unos resultados preliminares. Estos resultados preliminares son procesados por un algoritmo de ranking que organiza los datos/documentos en cierto orden y estos resultados ordenados se presentan al usuario en un esquema de visualización (interfaz de usuario) de acuerdo con la similitud de estos con las palabras clave con las que el usuario estableció la búsqueda.

Las investigaciones relacionadas con Sistemas de Recuperación de Información se encaminan en muchas direcciones, normalmente se toma cada uno de los componentes o combinaciones de estos en un SRI como un objeto de estudio y se establecen diferentes formas de mejorarlos. En [6] se habla de que en general un SRI debe arrojar los mismos resultados a una búsqueda sin importar quién sea el que hace la consulta, pero muchas empresas con el fin de satisfacer mejor a sus clientes y generar mayor lealtad a la marca, han optado por un enfoque de Recuperación de Información personalizada, en la que tienen en cuenta ciertos aspectos del usuario como su ubicación y sus costumbres para que los resultados se adapten al usuario y de esta forma se sienta mejor atendido.

Por otra parte, en [7] se hace una investigación que se centra en el repositorio de información. Por lo general los repositorios de información son un conjunto de documentos de texto que se encuentran en un disco duro de un computador o a través de la web usando las características de los documentos HMTL [8], pero debido a nuevas formas de interacción de los usuarios como las redes sociales o las plataformas de video y streamming se ha creado la necesidad de que los SRI sean capaces de gestionar otros tipos de archivos (fotos, música y videos). En [9] se enfocan en el algoritmo de búsqueda y ranking, tratando de plantear la forma en que los resultados de una búsqueda de un usuario no estén basados solamente en medidas de similitud sino también que se tenga en cuenta aspectos a nivel semántico de la información en el repositorio de datos y en la consulta hecha por el usuario.

Otras investigaciones van dirigidas hacia la forma en que interactúan los usuario con un SRI, por ejemplo en [8] se hace un análisis de los SRI usados por los niños, la información que encuentran a su disposición y su comportamiento al momento de realizar una búsqueda, esto con el fin de definir estrategias para proteger a los niños de todas las cosas malas que puedan encontrarse en internet. En [10] se busca mostrar nuevos paradigmas para las interacciones con la información del usuario que se enfaticen en su contexto y la interactividad con el fin de facilitar la exploración, interpretación y la asimilación de la información.

Por último se pueden encontrar otros enfoques relacionados a la extracción de los datos como en [11], en donde se busca aprovechar las características de los sistemas Peer-to-Peer, como el modelo de distribución en forma de grafo, para encontrar los resultados de búsqueda en ubicaciones más cercanas para optimizar los tiempos de respuesta.

2.2 CLASIFICACIÓN

La clasificación es una de las tareas más importantes en el campo de la minería de datos y también es conocida como aprendizaje supervisado en el campo del aprendizaje de máquina. Es un proceso mediante el cual se etiqueta un elemento o registro nuevo en ciertas categorías o clases [12] teniendo en cuenta un modelo que ha aprendido previamente las diferentes características de elementos similares, por ejemplo, cuando se aplica la clasificación en los seres vivos se puede hablar de que hay animales que tienen 2 patas, 4 patas, 6 patas, 8 patas, muchas más patas o ninguna, cada una de estas etiquetas puede tomarse como una clase o categoría.

Las características o propiedades de los datos permiten etiquetar ciertos registros dentro de una clase o categoría. Este tipo de problemas supervisados, donde la relación entre las diferentes características y su etiqueta tiene que ser aprendida se puede automatizar con un algoritmo denominado clasificador. Los datos usados para el aprendizaje de estas relaciones se conocen como datos de entrenamiento. Una vez se determina la relación entre las características de un registro se crea lo

que se llama un modelo que sirve para etiquetar nuevos registros que estén sin etiquetar [13].

La definición formal de clasificación en minería de datos es la siguiente: Dada una matriz D de n*d datos de entrenamiento y un valor de etiqueta de clase entre {1……k} asociado a cada una de las n filas o registros de D, se crea un modelo de entrenamiento M el cual puede ser usado para predecir la etiqueta de clase de un nuevo registro d-dimensional $\bar{Y} \notin D$ [14].

Dentro de los algoritmos de clasificación que se suelen usar se encuentran las redes Bayesianas que es un modelo probabilístico que representa un conjunto de variables aleatorias y sus dependencias condicionales a través de un grafo acíclico dirigido [15], funciones como la regresión logística o las usadas dentro de un perceptrón multicapa, algoritmos de aprendizaje perezoso como K-NN, algoritmos basados en reglas como OneR y los árboles de decisión como CART y C4.5.

2.2.1 K-nn

K-NN (K Nearest Neighbors o los K vecinos más cercanos) es un algoritmo que se basa en una heurística muy simple pero muy efectiva en diversos contextos de aplicación. Si un elemento que tiene ciertas características se encuentra cerca a unos vecinos que tienen características similares y estos vecinos son de la clase A en su mayoría, lo más posible es que el elemento también sea de la clase A y no de otra clase. La letra K hace referencia a la cantidad de vecinos más cercanos que se tienen en cuenta a la hora de tomar la decisión de etiquetar el elemento en cierta clase. La definición formal de K-NN es la siguiente:

- Sea $\varepsilon = \{e_1, \ldots, e_m\}$ un conjunto de datos con m ejemplos o instancias etiquetadas, donde cada ejemplo e_j contiene n atributos $(x_{j1}, \ldots, x_{jn})$ pertenecientes al espacio métrico Ω, y es clasificado con una etiqueta y_f que pertenece a Y $\{y_1, \ldots, y_k\}$. Sea además $\Gamma: \Omega \rightarrow Y$ una aplicación tal que $\Gamma(e_j) = y_f$, la clasificación de una nueva instancia e' cumple que:

  $$\Gamma(e') = \Gamma(e_i) \leftrightarrow \forall j \neq i.\ d(e', e_i) < d(e', ej)$$

  Donde d expresa una distancia o función de similitud definida en el espacio n-dimensional Ω. De esta manera una instancia de prueba es etiquetada según la clase de sus vecinos más cercanos [16].

La elección de la función de similitud es determinante ya que puede llegar a modificar el concepto usado de “vecindad” y por lo tanto es posible que, usando funciones de similitud distintas, una instancia de prueba termine con una etiqueta diferente.

2.3 FUENTES DE INFORMACIÓN

Teniendo en cuenta que el proyecto requería contar con un repositorio de archivos ejecutables y la relación con su correcta aplicación, se estudiaron soluciones de web scraping para obtener dicha información de páginas web que tienen parte de esta información disponible en internet. Además, se estudió el registro de Windows con el objetivo de conocer la información almacena allí y como podía ser usada para enriquecer la descripción de un archivo ejecutable. Otra información que se requirió estudiar fue el "change journal" o "el diario de cambios", ya que la búsqueda de ejecutables nuevos, modificados o borrados del sistema de archivos de un computador no se podía hacer en un solo momento buscando en todos los discos duros, por el alto costo computacional y el exceso de tiempo que esto puede tomar, y crear un agente de software que vigilará los cambios agregaba costos en procesamiento no deseables para los computadores que usan las empresas.

2.3.1 Web Scraping

Un sistema de web scraping es un sistema que extrae información de una página o aplicación web de manera automática y repetida y entrega la información almacenada en archivos o base de datos para su posterior uso. La tarea de extracción de información web generalmente se divide en cinco diferentes funciones:

1) Interacción web, la cual compromete principalmente la navegación en objetivos predeterminados que contienen la información deseada.
2) Soporte para la generación y ejecución de empaquetadores que extraen la información deseada y la empaquetan en un formato adecuado.
3) Programación horaria para permitir la extracción de datos de forma repetida en objetivos que ya han sido usados.
4) Transformación de información lo cual incluye filtros, transformación, refinamiento e integración de información extraída de una o más fuentes.
5) Entrega de la información estructurada a aplicaciones externas como una base de datos, una bodega de datos, sistemas de inteligencia de negocios o servidores de email entre otros.

Los Web Scrapers pueden navegar a través de una página web y extraer contenido textual, sin embargo, estos carecen usualmente de una lógica de programación que supere futuras actualizaciones en la estructura de la página web de la que extraen la información [17]. Dentro de los Web Scrapers más conocidos se encuentran:

- Mozenda: es un software que permite extraer información no sólo de páginas web sino de documentos Pdf, Word, Excel, entre otros. También permite la condensación de datos de diferentes fuentes de forma conjunta, es decir, se puede tomar datos de un documento de Excel y de una página al tiempo para llenar los datos de una tabla. Permite la creación de proyectos y la actualización semanal de los datos extraídos. Es software propietario [18].

- Automation Anywhere: es una plataforma que permite programar robots que realicen tareas de diferente complejidad, entre ellas extracción de información. Es el resultado de más de una década de automatización de procesos robóticos. Dentro de sus principales productos se encuentran IQ Bot que es un robot que busca darles sentido a diferentes datos de una fuente, Bot Insight que se encarga de generar información para Inteligencia de negocios y Bot Store para gestionar los robots que se usen. Es software propietario [19].
- Scrapy: es un framework que se especializa en la creación de elementos software llamados arañas (Spider) que sirvan para la extracción de datos de páginas web. Es de código abierto [20]. Tiene soporte para selección y extracción de datos de archivos HTML y XML, soporte para generar reportes en varios formatos y un fuerte soporte para extensibilidad.
- Data Scraping Studio: es una herramienta para sistemas operativos Windows que permite extraer información de sitios web a través del análisis del CSS o el uso de expresiones regulares. Es software propietario y tiene un costo que varía desde los 29 a los 100 dólares dependiendo de la cantidad de características habilitadas [21].
- Octoparse: es un web scraper que puede manejar sitios web estáticos o dinámicos que usen AJAX, javascript, cookies, etc. Lo que diferencia a este software de otros es que permite no sólo extraer información que se encuentra visible en una página web sino también datos que se encuentren ocultos. Los datos extraídos son primero enviados a unos servidores en la nube y a través de un api se obtienen localmente. Es software libre [22].

2.3.2 Registro de Windows

Es una base de datos jerárquica presente en los sistemas operativos Windows que se usa para almacenar información sobre la configuración del sistema operativo para uno o más usuarios, sobre la configuración de algunas aplicaciones y dispositivos hardware.

El registro contiene información que Windows usa continuamente durante su ejecución tales como los perfiles de usuarios, los permisos que estos tienen para la manipulación del computador, los tipos de documentos que este puede crear o el hardware y los puertos por donde funciona. Aunque el registro se encuentra presente en la mayoría de los sistemas operativos Windows, hay algunas pequeñas diferencias en ellos [23].

Dentro del registro se encuentran las llamadas Colmenas de llaves (Hive Keys), subllaves y valores en el registro que contienen un conjunto de archivos de soporte que contienen información de recuperación. La extensión de los archivos puede indicar el tipo de información que almacena o también la ausencia de ella.

Las principales secciones del registro son las siguientes:

- HKEY_CURRENT_USER: contiene información de configuración del usuario actualmente autenticado en el sistema operativo. Presenta información como carpetas, fondos de pantalla y configuraciones del panel de control.
- HKEY_USERS: contiene la información de los perfiles existentes en el computador.
- HKEY_LOCAL_MACHINE: contiene configuraciones generales del computador disponibles para cualquier usuario.
- HKEY_CLASSES_ROOT: es una subllave de HKEY_LOCAL_MACHINE\Software y contiene información con la que se asegura que un archivo sea abierto por el programa apropiado dentro del explorador de archivos.
- HKEY_CURRENT_CONFIG: contiene información acerca del hardware que es usado en el computador desde el arranque del sistema operativo.

Un usuario administrador del computador puede hacer modificaciones en el registro usando la herramienta Regedit.exe que se encuentra por lo general en C:\Windows\ pero esta herramienta no es sencilla de usar. Si se modifican cosas importantes del sistema operativo se puede llegar a obtener resultados no deseados, por esto, no se recomienda modificar el registro en forma manual.

2.3.3 Change Journal

El sistema de archivos NTFS [24] mantiene un diario de cambios en archivos y directorios con un número de secuencia de actualización. Cuando ocurre un cambio en un archivo o directorio, el "**diario de cambios**" se actualiza para ese volumen, con una descripción del cambio y el nombre del archivo o carpeta involucrada.

Los diarios de cambios son necesarios para recuperar la indexación del sistema de archivos en ocasiones en las que ocurre un fallo, de esta forma para el sistema es mucho más fácil y rápido reindexar un volumen entero y por lo tanto disminuir la posibilidad de pérdida de datos.

A medida que se van agregando, cambiando o eliminando archivos y directorios (carpetas), el sistema de archivos NTFS incluye un registro en un stream correspondiente al volumen que fue afectado. Cada uno de los change journal tiene un identificador de 64 bits sin signo con el que se puede manipular (Update Sequence Number, USN) [25]

Gracias a la funcionalidad y eficiencia del Change Journal, este se usa para dentro del presente proyecto para hacer la revisión de los cambios en los diferentes directorios sin tener que mantener un proceso abierto (agente) que pueda consumir recursos y que hagan que el usuario note una reducción en la velocidad en que se ejecutan diariamente los demás programas, hecho que puede ser muy notorio en equipos con procesadores de baja velocidad o viejos y poca memoria RAM.

2.4 CONCEPTOS TÉCNICOS DE DESARROLLO

2.4.1 Servicio Windows

Los servicios Windows son programas que trabajan en segundo plano, que se puede configurar para iniciar cuando el sistema operativo arranca y cuya funcionalidad puede variar dependiendo del objetivo con el que haya sido creado [26]. Es un concepto similar a los demonios (Daemon) [27] en sistemas operativos basados en Unix y debe ajustarse a las normas y protocolos del Service Control Manager [28], que es el componente responsable de la gestión de servicios en Windows. Otra forma en la que puede arrancar un servicio Windows es mediante un evento o a través de la iniciación manual del servicio en el Panel de control siguiendo las opciones de Sistema y seguridad -> Herramientas administrativas -> Servicios.

Los servicios Windows se ejecutan en el contexto de tres cuentas de usuario: System, Network service y Local service. Esto permite que los servicios arranquen a pesar de que un usuario no haya iniciado sesión. El tipo de cuenta de usuario asociada con el servicio define el acceso que tiene el servicio a diferentes recursos del sistema operativo y la gestión que pueden hacer sobre ellos.

La lista de servicios instalados y sus detalles se pueden consultar dentro del administrador de tareas de Windows en la pestaña servicios. En los detalles se pueden encontrar su estado, si está activo o no, el tipo de inicio y una descripción del servicio.

2.4.2 Representational State Transfer

Representational State Transfer (REST) es un estilo de arquitectura de software desarrollado por uno de los autores de HTTP 1.1 (Roy Fielding) para sistemas distribuidos. En la actualidad se usa para describir cualquier interfaz entre sistemas que usen HTTP para obtener datos o indicar la ejecución de operaciones sobre datos [29]. Un servicio REST es un servicio que no tiene estado, esto quiere decir que no se puede llamar a un servicio REST, pasarle unos datos como por ejemplo un usuario y una contraseña y esperar que los recuerde en una siguiente petición o llamado.

En REST lo que se publica son recursos. Un recurso se puede considerar como una entidad que representa un concepto de negocio que puede ser accedido públicamente. Un ejemplo de recurso sería simplemente “EmpleadosDeLaEmpresa”, que retorna en un texto y con un formato específico, los empleados activos de la empresa.

En una API REST se cuenta con recursos accesibles por identificadores conocidas como URIs [30]. Sobre esos recursos se pueden realizar acciones, generalmente diferenciadas a través de distintas HTTP. Si se invoca una URI se debe obtener una representación del recurso correspondiente a esa URI, por ejemplo, si se quiere acceder a una URI relacionada con una manzana, al acceder a su URI

debería obtener una representación de la manzana por lo general mediante JSON o XML [31].

2.4.3 Simple Object Access Protocol

Simple Object Access Protocol (SOAP) es un protocolo para intercambiar información en internet y define cómo dos objetos en procesos distintos pueden comunicarse a través del intercambio de datos XML. SOAP permite simular el llamado a una función con un retorno de un tipo de dato nativo por eso es común asociarlo con RPC (Remote Procedure Call).

Un mensaje SOAP se conforma de tres partes (ver ejemplo en la **Figura 2**):

- Sobre: define qué hay en el mensaje enviado y cómo interpretarlo/procesarlo.
- Cabecera: contiene información adicional que puede ayudar a procesar el mensaje. Este es un campo opcional pero que está presente la mayoría de veces [32].
- Cuerpo: es el mensaje en sí. En esta sección se incluyen datos relativos a la petición o la respuesta. Por ejemplo, pueden aparecer datos como temperaturas, resultados de operaciones matemáticas o cadenas que indiquen el estado de un sensor.

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <namesp1:NDFDgenByDay
        xmlns:namesp1="http://weather.gov/forecasts/xml/DWMLgen/wsdl/ndfdXML.wsdl">
            <latitude xsi:type="xsd:float">44.52</latitude>
            <longitude xsi:type="xsd:float">-89.58</longitude>
            <startDate xsi:type="xsd:string">2005-04-23</startDate>
            <numDays xsi:type="xsd:int">5</numDays>
            <format xsi:type="xsd:string">12 hourly</format>
        </namesp1:NDFDgenByDay>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 2 *Ejemplo de mensaje SOAP (Fuente [32])*

SOAP se caracteriza por su extensibilidad (por ejemplo, en materia de seguridad), su neutralidad y su independencia ya que puede ser usado con protocolos como HTTP, SMTP y cualquier modelo de programación.

Cuando se hace conexión a un servicio web se necesita saber la dirección del servicio y con qué protocolo se transporta la información, para esto la dirección muestra no sólo la ubicación sino también cómo comunicarse con el servicio.

También se debe incluir el espacio de nombres del servicio web. Los parámetros requeridos para un servicio SOAP específico se definen en un documento WDSL (Web Service Definition Language) [33]. En este documento se describen las interfaces/contratos para un servicio web.

2.5 CONCEPTOS RELACIONADOS CON INGENIERÍA DE SOFTWARE

2.5.1 Scrum

SCRUM es una metodología de desarrollo ágil que en la actualidad es muy usada por ofrecer muchas ventajas como la rápida adaptabilidad y no ser tan estricta como otras metodologías en materia de documentación.

La metodología se desarrolla a través de una serie de bloques temporales llamados Sprint que se enfocan en entregar software funcional y que tienen una duración por lo general de dos a cuatro semanas. La metodología cuenta con una serie de artefactos y roles que se describen a continuación:

a) Artefactos:
   - **Product backlog**: es una lista de todo el trabajo que debe realizar el equipo para poder desarrollar el proyecto. Esta lista representa los productos que el cliente quiere o desea. Dentro de este documento se encuentran las historias de usuario. El Product Backlog es gestionado por el Product owner que es responsable de adicionar o eliminar historias de usuario de este. Los elementos del Product backlog son también priorizados por el cliente y el product owner de tal forma que cada uno de ellos aporte algo de valor al producto que se quiere desarrollar y que los elementos más críticos tengan mayor interés por parte del equipo.
   - **Historias de usuario**: es una tarjeta que describe una característica o un incremento de valor para el cliente. Las historias de usuario deben ser auto contenidas, es decir, que no dependan de otras historias de usuario en lo posible. Mientras una historia de usuario no haga parte de un Sprint puede ser modificada y reescrita. También es importante tener en cuenta que cada historia de usuario debe poderse probar para poder determinar su completitud.
   - **Sprint backlog**: Se puede ver como una versión resumida del Product backlog pero que contiene las historias de usuario a ser desarrolladas en un solo Sprint. Típicamente, cuando una historia de usuario es admitida para un Sprint, esta es dividida en tareas como por ejemplo desarrollar una interfaz gráfica o hacer alguna modificación a la base de datos. Estas tareas suelen ser ubicadas en un sitio visible para todo el equipo en donde se describe si está por desarrollarse, en desarrollo, en pruebas o ya está terminada y aprobada.
b) Roles:
   - **Scrum master**: es responsable por asegurar que el proceso Scrum sea entendido y seguido. Facilita que las reuniones del equipo se desarrollen y que no haya obstáculos para que el equipo pueda desempeñar sus tareas.

El Scrum master no es el jefe del equipo, pero siempre está disponible para permitir que el equipo logre sus objetivos y también se necesita que sea una persona con buenas habilidades de comunicación ya que suele hablar con todos los integrantes del equipo.

- **Product owner**: es una persona que tiene un conocimiento más profundo del producto y puede ser considerado un representante del cliente que busca que cada elemento que se desarrolle aporte para lograr lo que el cliente busca. También prioriza las historias de usuario y gestiona el Product backlog y el Sprint backlog.
- **Desarrollador**: son los que hacen las diferentes tareas que permiten completar o satisfacer cada una de las historias de usuario y como consecuencia de esto desarrollar el proyecto como tal. En las actividades de un desarrollador se encuentran el desarrollo de nuevas funcionalidades, la ejecución de pruebas, la corrección de bugs y dependiendo de las situaciones también realizar actividades de arquitectura como el modelado de base de datos.

c) Actividades:

- **Planeación del Sprint**: es un proceso que se desarrolla antes de empezar cada Sprint en el que se determina qué características o qué software será desarrollado en el mismo. De igual forma si hay que hacer mantenimiento y pruebas. Se revisa la prioridad de las actividades a desarrollar y se planifica, de acuerdo con un límite de tiempo determinado por las horas laborales que están enmarcadas por el Sprint, cuántas horas le va a dedicar cada persona a cada una de las actividades pendientes.
- **Reuniones diarias**: en el marco de la metodología Scrum se tiene como una de las principales herramientas las reuniones diarias, ya que gracias a esto es posible que el equipo se adapte rápidamente a diferentes eventualidades que se presenten como es el caso de cambios en los requisitos por parte del cliente o también para que los miembros del equipo expongan los problemas a los que se haya enfrentado en el día anterior y que puedan requerir del apoyo de un compañero para su solución. Esta reunión por lo general no dura más de 15 minutos y adicionalmente permite que todos los miembros tengan conocimiento sobre todas las actividades que se están desarrollando.
- **Revisión del Sprint**: cuando un Sprint ha terminado, se realiza una reunión para revisar qué es lo que se había planeado para el Sprint al inicio y qué es lo que se logró hacer al final de este. Se hace una presentación informal, demo, de los componentes que hayan sido desarrollados o reparados para que se pueda observar si el software cumple con lo que requiere el cliente.
- **Retrospectiva del Sprint**: se realiza al final del Sprint y es una oportunidad para que el equipo reflexione sobre los eventos ocurridos. En esta reunión se hacen las felicitaciones pertinentes a los miembros del equipo y se discuten los asuntos que resultaron confusos o problemáticos para el desarrollo normal de las actividades del Sprint con el fin de proponer soluciones y que su aparición se reduzca o elimine.

2.5.2 Modelo, Vista, Controlador

El Modelo, Vista Controlador (MVC) es un patrón de diseño que consiste en tres tipos de clases o componentes que se especializan en un tipo de funcionalidad. El término se usa desde finales de los 70's alrededor del proyecto Smalltalk [34] y se definió a partir de lo que se conoce como capas de una aplicación. Los componentes que hacen parte del Modelos son los que representan la información con la que los usuarios trabajan. Las Vistas son las interfaces gráficas de usuario que sirven para visualizar una parte o todo el Modelo y le muestra al usuario las diferentes acciones que puede llegar a ejecutar. Los componentes de la sección Controlador son aquellos que procesan las peticiones entrantes, realizan operaciones y hacen la selección de qué vista debe mostrarse al usuario en cada escenario que se presente [35].

Gracias a la separación en capaz de los componentes, las responsabilidades de cada uno quedan bien separadas y definidas. La lógica que manipula los datos se encuentra sólo en el Modelo, la que muestra la información en las Vistas y la lógica que gestiona las peticiones del usuario es gestionada por el Controlador. Esto también facilita el mantenimiento y la extensibilidad.

El modelo puede ser implementado con cualquier tecnología, desde una base de datos MySQL hasta simples objetos, no hay restricciones. Para la gestión de las vistas, ASP.NET (ver **Figura 3**) cuenta con un motor de vistas llamado Razor que se encarga de procesar una vista para mostrar unos resultados en el navegador. Dentro de algunas vistas se incluye algo de código C# para mostrar de forma dinámica cualquier cambio hecho en el modelo.

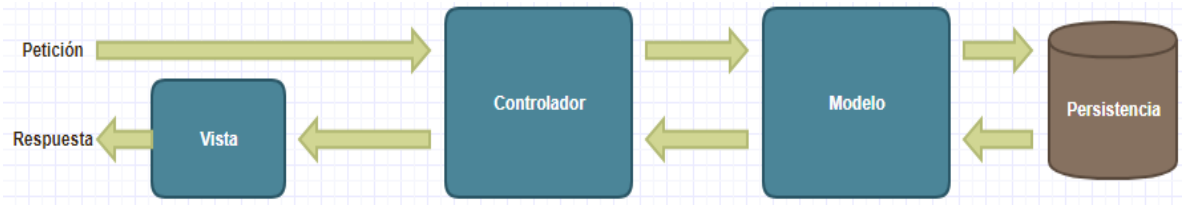

Figura 3 *Flujo de Petición/Respuesta en ASP.NET MVC (Fuente propia)*

2.6 ESTADO DEL ARTE

Muchas empresas hacen una gestión fuerte sobre los recursos con los que cuentan [36], como los recursos humanos o los recursos tecnológicos, esto con el fin de cuidarlos y aprovecharlos al máximo. Dentro de los recursos más importantes está el software que usan a diario y sobre todo las licencias de software. El control del software les permite reducir costos de software y el soporte, mejorar la productividad de los trabajadores, establecer políticas relacionadas con adquisición de software, evitar problemas de seguridad y además asegurar el cumplimiento de las normas relacionadas con las licencias, entre otros.

Dentro de las herramientas para la **gestión de recursos software (Software Asset Management)** se pueden encontrar:

- **Microsoft Software Inventary Analyzer**: es una herramienta gratuita que permite escanear e inventariar el software de Microsoft que está instalado en un PC o en varios que se encuentren en una misma red. Genera un informe que proporciona detalles de los productos como el tipo y número de licencia. Tiene unos requisitos muy bajos para funcionar, lo que es importante, pero su mayor desventaja es que se limita a los productos Microsoft que normalmente están bien registrados en el sistema [37].

- **ManageEngine ServiceDesk Plus**: es una suite en la cual se incluye una herramienta para gestionar el software instalado y sus licencias. Dentro de sus funcionalidades se encuentran el listar el software, la cantidad de instalaciones que tiene dicho software a nivel de una red de computadores, la plataforma en la que se encuentra instalado y permite que se asigne un software a una lista negra para que se le realice un seguimiento por cuestiones de seguridad y dado el caso se elimine de las estaciones de trabajo en las cuales se encuentra instalado. La asociación se hace de manera automática, sin embargo, pide en ciertos casos la confirmación de un experto para identificar un software específico y éste también puede agregar software instalado en una estación de trabajo de forma manual. Como un agregado interesante se encuentran la realización automática de inventarios de forma periódica si el cliente así lo desea. Este software es propietario, 100% web y cuenta con más de 10.000 empresas usuarias alrededor del mundo [38].

- **Snow License Manager**: presenta una visión unificada de los activos de software, autorizaciones de licencias y métricas de uso de aplicaciones. Según lo que se encuentra en la página de la aplicación, este gestor cuenta con información de más de 73.000 editores de software y promete reconocer cerca de 455.000 aplicaciones de manera directa, por lo cual dice identificar el 100% del software instalado en un equipo, algo que puede ser un poco difícil de cumplir a pesar de la amplia base de datos con la que cuentan. Tiene soporte para todos los tipos de licencias, GPL, BSD, etc., lo cual es una interesante adición al SRI que se propone en el presente trabajo, aunque requeriría de una investigación más a fondo en la materia. Es software propietario centrado específicamente en la gestión de licencias. La interfaz de usuario puede ser personalizada dependiendo de las necesidades de los administradores de recursos software y genera informes de gestión de forma sencilla [39].

- **FlexNet Manager Suite for Enterprises**: en el portal web de este producto se menciona que una empresa puede alcanzar cuatro niveles de madurez en materia de la gestión de recursos de software. El primer nivel consiste en simplemente descubrir los recursos de software usando tecnologías como agentes y recolectar información de uso de las aplicaciones. El segundo nivel se relaciona con nombrar y clasificar de forma consistente el software por

dispositivo. El tercer nivel se relaciona con calcular el nivel de cumplimiento de las licencias de software y verificar qué software no es realmente utilizado para aplicar políticas y gestionar de forma eficiente las licencias. Por último, el cuarto nivel consiste en realizar la gestión de licencias y el uso de software de forma automática. FlexNet Manager Suite for Enterprises afirma poder adaptarse a todos los niveles de madurez en gestión de recursos de software de las empresas ofreciendo servicios automatizados que faciliten la gestión de estos recursos. Este software es propietario y como característica diferenciadora frente a otras opciones en el mercado está la gestión de los recursos software a nivel monetario, permitiendo saber cuánto se ha invertido en un software en específico o cuánto se ha invertido en software de alguna compañía como por ejemplo Oracle [40].

- **Snipe-IT**: es una herramienta que permite la gestión de muchos aspectos relacionados con la gestión de recursos, entre ellos las licencias de software. Cuenta con algunos datos que no se suelen encontrar en otros sistemas gestores de licencias de software como lo son la llave del producto o su depreciación, esto es un buen soporte para la toma de decisiones a la hora de hacer compras o renovaciones de licencias. Es Open Source y a pesar de que ofrece muchas opciones, su interfaz no es muy intuitiva; sus reportes también son bastante simples y poco variados [41].

- **Symantec ExSP Program**: esta herramienta permite gestionar los recursos software al proveer una serie de aplicaciones de los partners de Symantec mediante un modelo de “alquiler mensual” en el que sólo se paga por el software que se utiliza. El contrato se renueva automáticamente a menos que Symantec o el comprador lo den por finalizado. Como ventaja tiene que permite el pago de las licencias a plazo vencido, es decir, no se requiere pago por adelantado, sino que se paga después de haber usado el software, pero como desventaja está la cantidad y la variedad del software que se tiene a disposición. Es una forma diferente e interesante de abordar el tema de las licencias de software y que también empieza a ser usado por algunas entidades como el Ministerio de Educación que reconocen sus cualidades. Este software es propietario [42].

CAPÍTULO 3

3 INTRODUCCIÓN A LA SOLUCIÓN PROPUESTA

En la presente sección se muestran los diferentes componentes que hacen parte del Sistema de recuperación de información para la asociación de ejecutables con su software correspondiente.

En la **Figura 4** se presentan los principales componentes del SRI y su interacción.

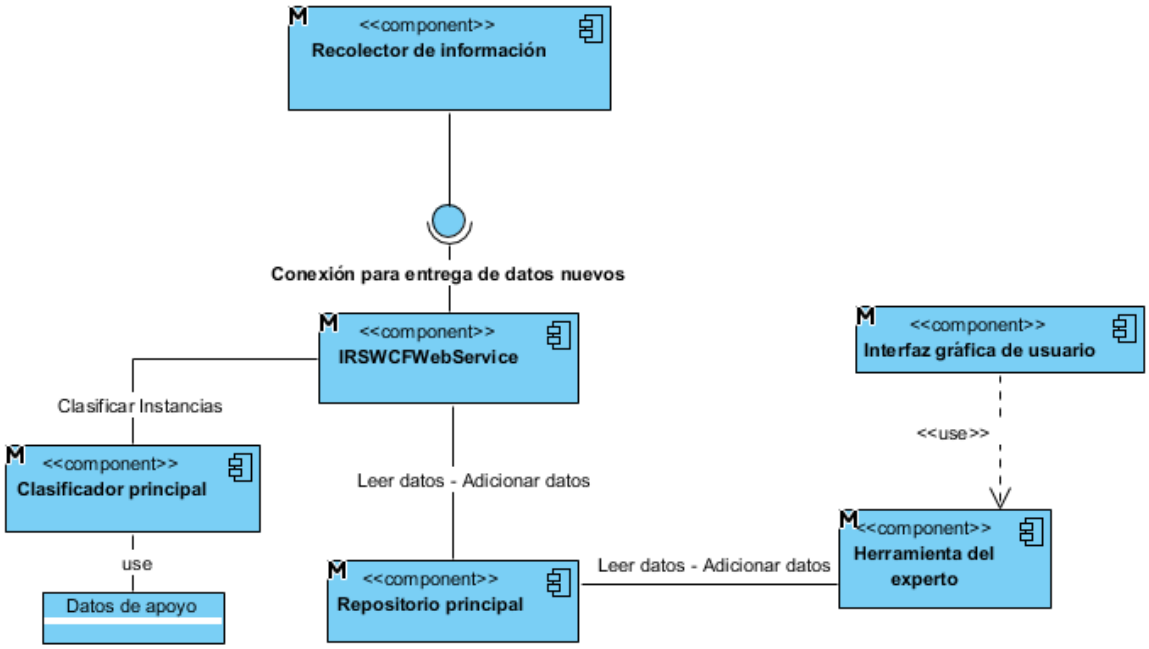

Figura 4 *Arquitectura de los componentes del SRI (Fuente propia)*

Dentro de un computador cliente existen un componente, el Recolector de información que hace la extracción de la información de los ejecutables y hace uso de tres componentes para lograr este fin, el Change Journal Reader que extrae la información básica de los ejecutables, el ExeDataExtractor que obtiene los datos completos de cada ejecutable a clasificar y el IRSWCFWSClient que es el cliente del IRSWCFWebService que se encuentra en el servidor.

En un servidor se encuentran otros componentes, el Repositorio principal que es el que tiene en una base de datos la información de los ejecutables anteriormente identificados y verificados, el Clasificador principal que se ocupa de determinar a qué software pertenece un ejecutable de forma automática, el IRSWCFWebService que controla el flujo de las operaciones y la invocación de los diferentes componentes del servidor en el momento en que sea requerido, unos datos de apoyo para soportar las clasificaciones no automáticas y producir

sugerencias al experto, y la Herramienta para el experto que le permite al mismo actuar en casos en que no se alcance un porcentaje de similitud adecuado para asociar un ejecutable con un software en específico. Estos componentes se describen en detalle más adelante en los capítulos 4, 5, 6 y 7.

La funcionalidad común (curso normal) del sistema de recuperación de información cuando va a realizar la clasificación de un nuevo archivo ejecutable encontrado en un computador se rige de acuerdo con el diagrama de secuencia de la **Figura 5**.

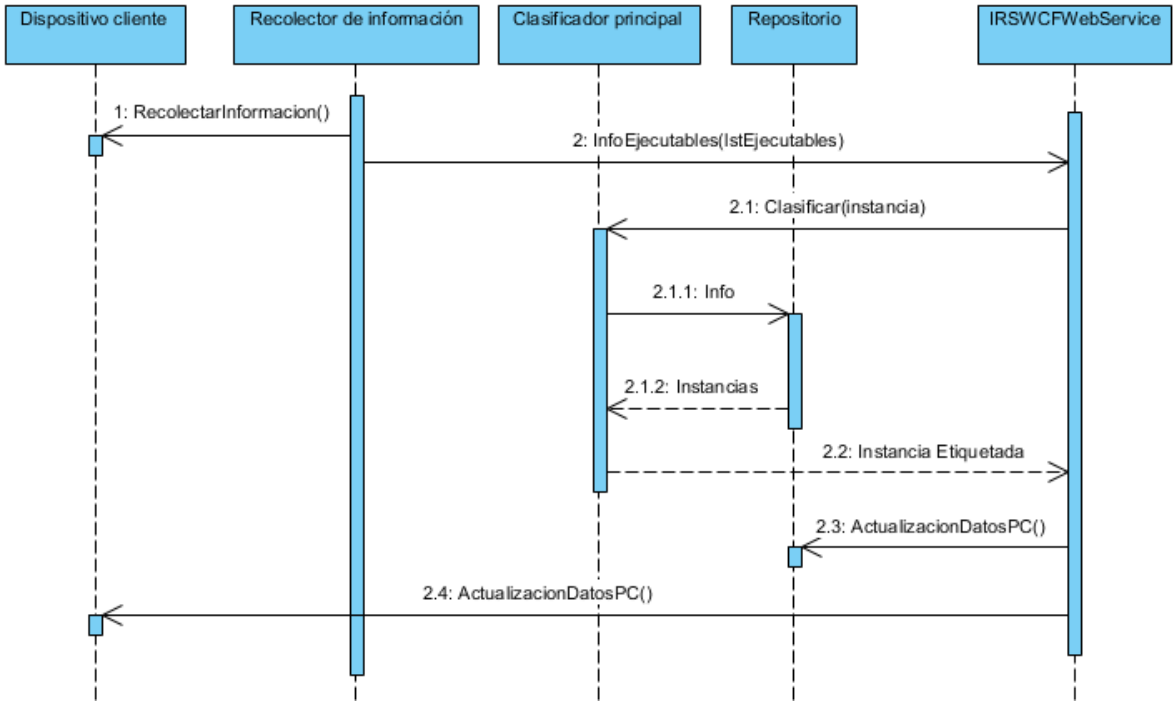

Figura 5 *Diagrama de secuencia de la clasificación de un nuevo ejecutable (Fuente propia)*

Este flujo normal de clasificación se realiza en forma totalmente automática, gracias a que la información que hay en el repositorio permite asociar la nueva instancia a ser clasificada con un software específico. El proceso arranca con la recolección de la información que se hace en el computador de un cliente y es un proceso que se realiza de acuerdo con una programación o agenda. Una vez se tiene la información de los nuevos ejecutables por clasificar, esta información es pasada por red al ISRWCFWebService. El ISRWCFWebService le envía al Clasificador principal cada instancia que se necesite etiquetar. Luego de esto el clasificador solicita al repositorio las instancias que tengan la mayor similitud con la instancia que se quiere clasificar y haciendo uso de un algoritmo basado en K-nn, decide cuál es la etiqueta más adecuada para esa instancia. Por último, el Clasificador le entrega al ISRWCFWebService la instancia con su respectiva etiqueta.

El proceso desde “clasificar” hasta “recibir la instancia etiquetada” se repite para cada instancia y el computador cliente mantiene un historial que le permite definir

qué nuevas instancias se deben enviar para clasificar posteriormente con base en los cambios presentados en el computador.

A partir de este escenario típico, se pueden presentar otros escenarios opcionales. Los escenarios opcionales que se tendrán en cuenta para el desarrollo del proyecto son los que ocurren cuando una instancia no se puede clasificar automáticamente.

Cuando se elimina un archivo ejecutable de un computador, el SRI realiza los pasos presentes en el diagrama de secuencia de la **Figura 6**.

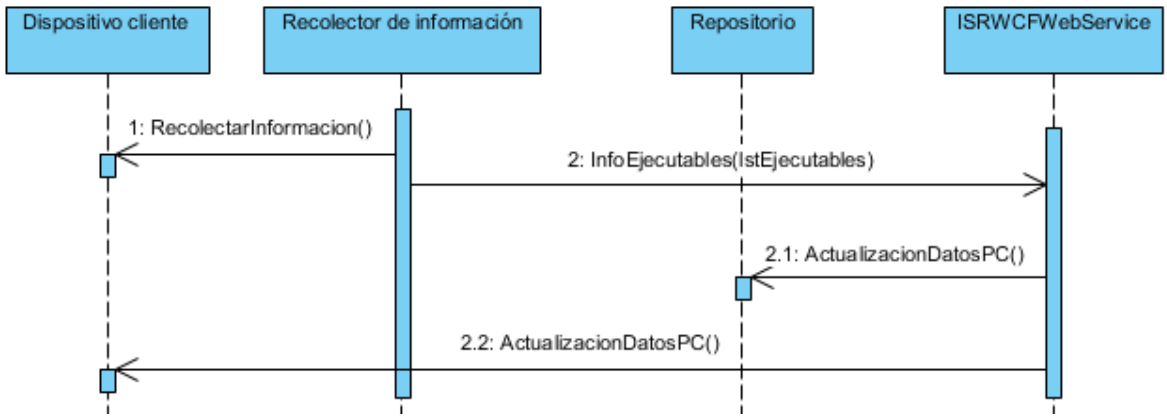

Figura 6 *Diagrama de secuencia de la eliminación de un ejecutable – flujo alterno (Fuente propia)*

Este también es un caso totalmente automático. Sólo es necesario actualizar los datos referentes al software presente en un dispositivo cliente, no hay que realizar una clasificación.

El primer caso alterno de clasificación que todavía es automático se presenta en el diagrama de secuencia de la **Figura 7**. Este escenario se presenta en los casos en que una instancia no cuenta con suficiente información que le permita al clasificador asociar la instancia con alguna de las que se encuentra en el repositorio. Para poder complementar su información, el IRSWCFWebService le pide al Inspector del registro del sistema y al analizador de directorios información adicional que pueda apoyar el proceso de clasificación. Una vez obtenida esta información adicional, se procede a hacer un nuevo intento de clasificación y en caso de ser exitoso, se agrega la nueva información al repositorio para que la próxima vez que aparezca una instancia igual o similar, pueda ser clasificada de forma automática.

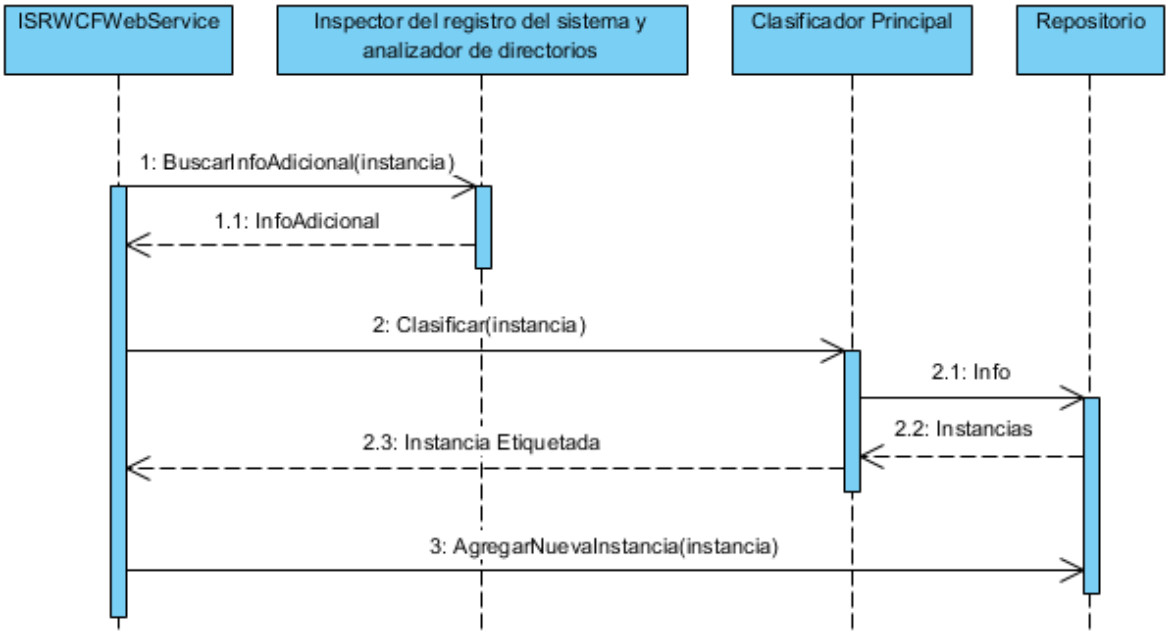

Figura 7 *Diagrama de secuencia de la clasificación de un ejecutable que requiere información adicional – flujo alterno (Fuente propia)*

Si el proceso de clasificación con el escenario anterior fue fallido, entonces se aplica el diagrama de secuencia presentado en la **Figura 8**, que ya requiere la intervención de un experto y por ello el proceso pasa a ser semiautomático.

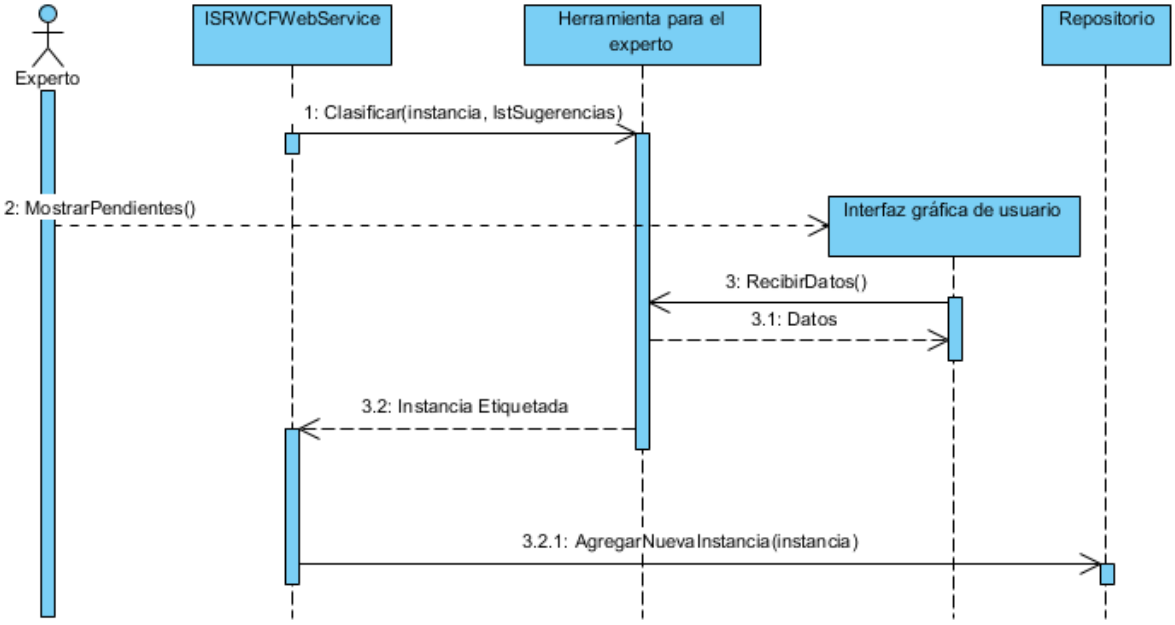

Figura 8 *Diagrama de secuencia de uso de la herramienta del experto (Fuente propia)*

Se llega a este escenario, debido a que no se obtuvo un porcentaje de similitud aceptable con alguna de las instancias en el repositorio a través de los procesos anteriormente descritos.

Como es ineficiente e inviable que el experto esté disponible todo el tiempo, vigilando si hay nuevas instancias por clasificar, el sistema con anterioridad prepara un paquete de instancias por clasificar y le notifica al experto a través de un correo electrónico. Posterior a esto, el Clasificador Principal le envía a la herramienta del experto las instancias a clasificar junto con un listado de sugerencias que le pueden ayudar al experto a tomar una decisión. El Clasificador Principal previamente revisa los datos de apoyo en busca de instancias que se parezcan a la instancia por clasificar. Cuando el experto lo decide, arranca la interfaz gráfica de la "herramienta para el experto" recibiendo información de los ejecutables que necesitan clasificación y la información específica de cada ejecutable. Cuando el experto ya ha decidido cuál será la etiqueta para cada ejecutable, la herramienta para el experto le devuelve las instancias etiquetadas al ISRWCFWebService y este agrega la nueva información al repositorio.

Muchos de los procesos mostrados en estos escenarios, el normal y los alternos, inician de manera asíncrona. Existen también otros escenarios en los que se requiere que los procesos sean síncronos, pero varios de esos escenarios no se tomaron en cuenta en el desarrollo del presente trabajo con el objetivo de no exceder el alcance del proyecto (9 meses según la reglamentación existente en la Facultad de Ingeniería Electrónica y Telecomunicaciones).

3.1 DESPLIEGUE DEL SRI

Los componentes del SRI se despliegan en cada máquina de acuerdo con el diagrama presentado en la **Figura 9**.

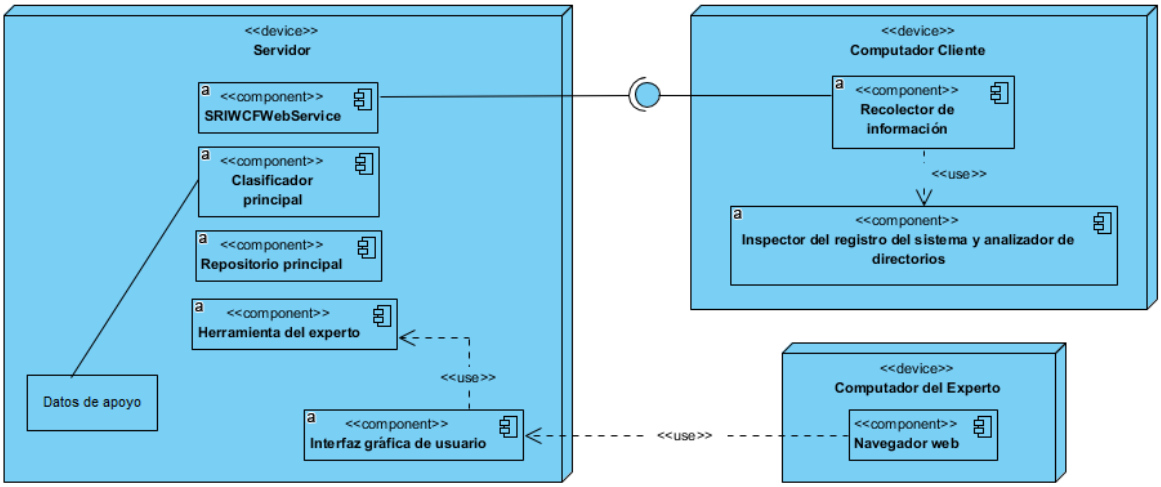

Figura 9 *Diagrama de despliegue del SRI (Fuente propia)*

Dentro de cada computador cliente se aloja el Recolector de información que tiene en su interior un Task Scheduler que arranca automáticamente cada cierto periodo los componentes que vigilan los cambios en los archivos y también el inspector del registro que se ejecuta sólo en algunas ocasiones. El Experto sólo necesita hacer uso de un navegador web para acceder a la interfaz gráfica de la "Herramienta para el experto", por lo cual lo puede hacer desde cualquier dispositivo.

El resto de las componentes se alojan en un servidor por cuestiones de facilidad de uso e implementación. Si se trabajase en un entorno más complejo que el de este trabajo de grado, sería necesario tener en cuenta otros aspectos como por ejemplo replicación de datos, que cambiarían un poco este diagrama de despliegue.

CAPÍTULO 4

4 REPOSITORIO PRINCIPAL

La construcción del repositorio principal inició con la búsqueda de una fuente externa que proporcionara los datos que se buscaban, en especial la asociación entre ejecutable y el software al que pertenece. Se buscaron varias fuentes gratuitas en internet ya que Aranda con anterioridad había hecho averiguaciones sobre bases de datos relacionadas con fuentes privadas y su costo resultó ser demasiado alto. Dentro de las búsquedas aparecieron páginas como SystemExplore.com, BDNA.com que ofrecen algunos datos pero que en general no se pueden tomar como una fuente confiable.

Aranda también contaba con una base de datos muy grande creada por otra empresa. Esta base de datos no sólo tenía información sobre los ejecutables sino además sobre las compañías desarrolladoras y lo que se conoce como suites de software que son paquetes que agrupan varios programas relacionados desarrollados por una misma compañía. Los registros de esta base de datos llegan a más de 600.000, un número bastante llamativo a simple vista, pero al revisar de forma minuciosa los datos se encontró que existían demasiados registros duplicados o con muy pocas diferencias y los ejecutables en ella por lo general eran de unas pocas empresas desarrolladoras, lo que la hace una fuente muy poco variada, a pesar de su tamaño.

Se tomó la decisión de crear un repositorio propio desde cero a partir de la información extraída de computadores de Aranda. Se obtuvo información de los ejecutables de varios dispositivos y se clasificaron de forma manual para asegurar la idoneidad de los registros que formarían parte del repositorio. En total se usaron los datos de diez (10) dispositivos con diferentes características con el fin de que el repositorio tuviera información variada y que contara con una cantidad mínima como punto de partida, ya que el objetivo es que este crezca de manera progresiva con su uso en el tiempo. Entre más registros se almacenen, más posibilidades hay de que una nueva instancia a clasificar sea clasificada automáticamente.

Si bien la cantidad de registros obtenidos de los computadores de Aranda ascienden a más de 370.000, con la cantidad de registros elegidos se llega a una no despreciable cifra de 10.000 registros con información completa y correcta.

La clasificación manual de estos registros tomó dos semanas de trabajo a una persona tiempo completo. A cada registro se le dio un tiempo de máximo cuatro minutos para encontrar a qué software pertenecía, ya que, si bien se le podría dedicar más tiempo a cada uno, esto haría que completar el repositorio tomara

demasiado tiempo. Este tiempo de cuatro minutos fue calculado como un tiempo adecuado para hacer esta asociación ya que en algunos casos un ejecutable puede ser conocido y encontrarse información con facilidad por lo que el proceso sería muy rápido y en otros casos habría registros poco conocidos que requieren más tiempo de exploración. El resultado es un repositorio confiable pero que tampoco se puede considerar infalible. A futuro se espera alimentar el repositorio con más registros y revisar más a fondo los existentes para que la fuente sea completamente confiable.

4.1 IMPLEMENTACIÓN DEL REPOSITORIO PRINCIPAL

La implementación del Repositorio principal se hizo a través de una base de datos en Microsoft SQL Server que tiene un conjunto de tablas para gestionar los datos del software, ejecutables, computadores e inventario encontrado en las máquinas además de otras tablas que sirven para el correcto funcionamiento de la "Herramienta para el experto". Esta decisión de implementación se tomó debido a que los datos que clasifique el experto siempre van a generar nuevos registros en el repositorio principal y los ejecutables identificados en un computador están relacionados a través de una llave foránea con un registro en la tabla central del repositorio principal a menos que aún no se haya realizado su clasificación.

El modelo Entidad Relación de los datos del repositorio principal en la base de datos se presentan en la **Figura 10** y continúan en la **Figura 11**.

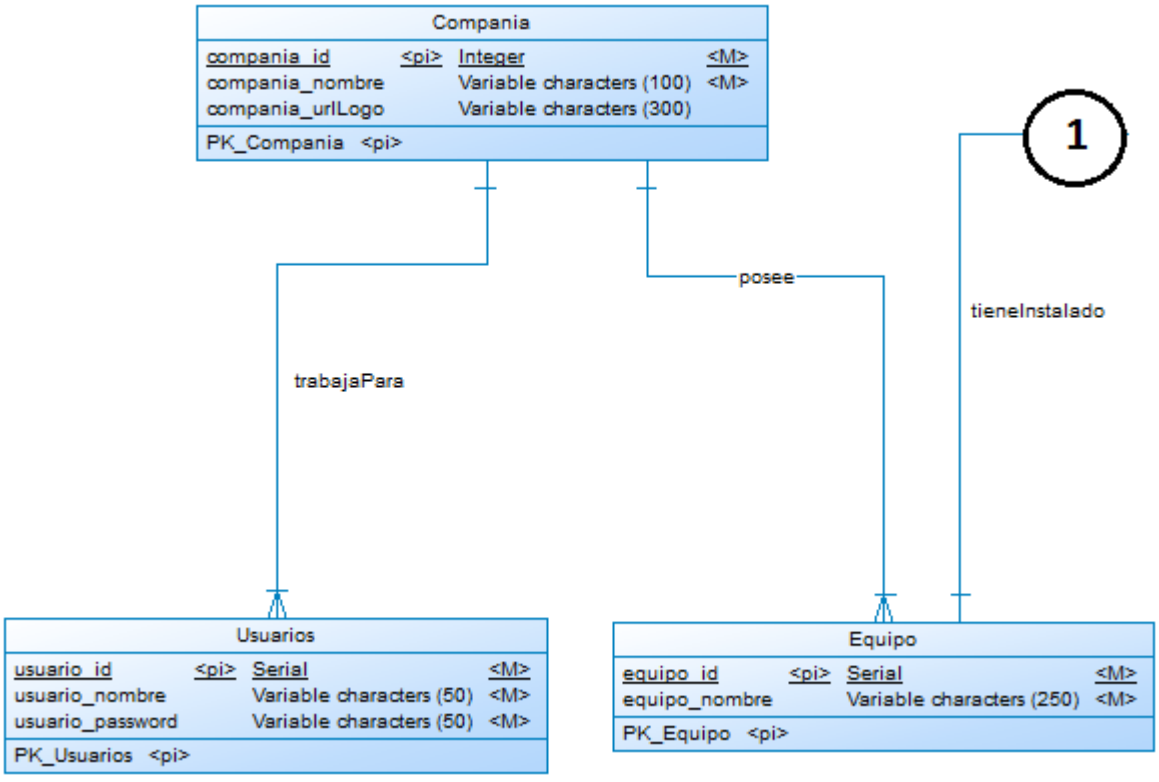

Figura 10 *Modelo Entidad/Relación Base de datos SRI (primera sección) (Fuente propia)*

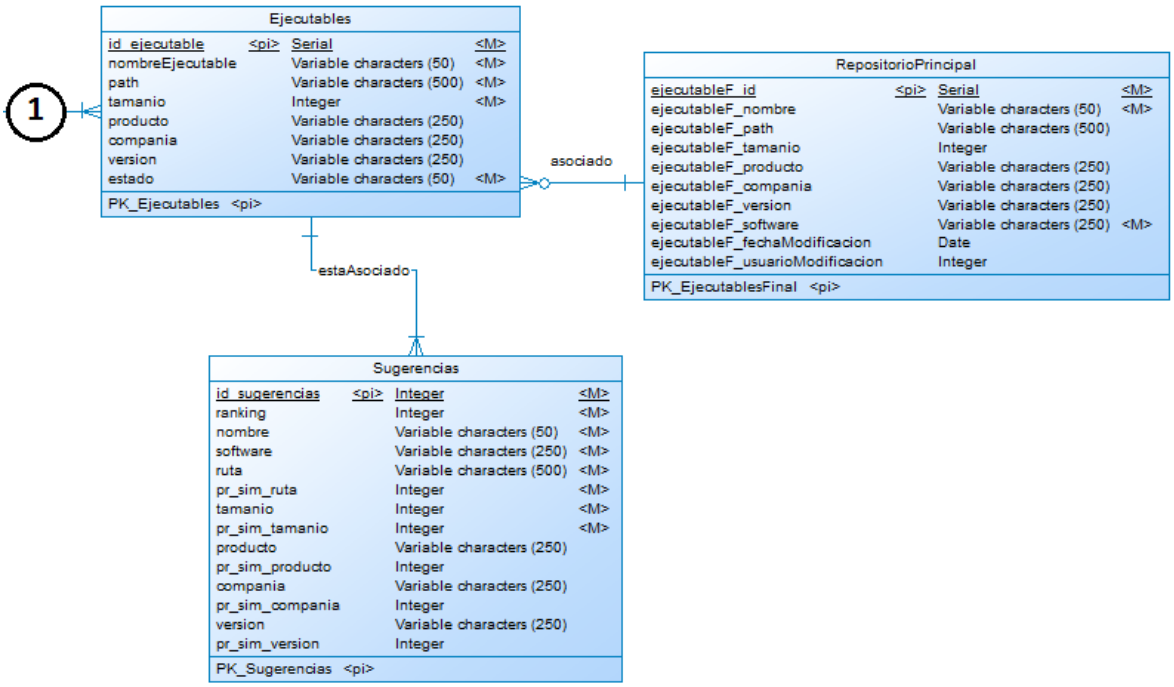

Figura 11 *Modelo Entidad/Relación Base de datos SRI (segunda sección) (Fuente propia)*

De las entidades que se pueden ver en las anteriores figuras, la entidad "RepositorioPrincipal" es la implementación del repositorio. Esta entidad tiene los siguientes atributos:

- **ejecutableF_id**: un identificador que sirve para que cada registro de esta entidad sea único.
- **ejecutableF_nombre**: el nombre del ejecutable, por ejemplo "PWPoint.exe".
- **ejeutableF_path**: la ruta asociada al archivo ejecutable. Este valor puede ser bastante extenso, por lo cual se le da un espacio amplio de almacenamiento. Este texto está asociado a la ubicación por defecto del ejecutable.
- **ejecutableF_tamanio**: el tamaño en bytes del archivo ejecutable.
- **ejecutableF_producto**: normalmente un archivo ejecutable tiene inmerso dentro de sí mismo un valor de producto al cual pertenece. Este valor está más asociado al programa que lo usa, más no a lo que se considera propiamente como el software dentro del proyecto.
- **ejecutableF_compania**: la empresa, casa o persona desarrolladora del ejecutable. Cabe aclarar que este valor no está relacionado con lo que se guarda en la entidad "Compania" que será explicada en detalle en la sección de la Herramienta para el experto.
- **ejecutableF_version**: versión de compilación del ejecutable. Este valor no se puede trabajar como un número directamente pues puede contener texto o tener varios puntos en medio, por ejemplo 2.45.4687.2.
- **ejecutableF_software**: el software al cual pertenece el archivo ejecutable.

- **ejecutableF_fechaModificacion**: la fecha y hora en que fue añadido o modificado el registro en la tabla.
- **ejecutableF_usuarioModificacion**: el ID del usuario que insertó un registro en específico en esta tabla.

Cabe destacar que los atributos de fecha de modificación y usuario de modificación son opcionales pues los registros que tiene el repositorio de forma inicial no cuentan con estos valores.

Los datos en el repositorio son dinámicos, inicialmente se puede hacer una carga de datos de alguna fuente (basados en equipos de una empresa o de información en internet, por ejemplo) y luego, mientras la herramienta se usa, se agrega nueva información en los procesos de clasificación que involucran al experto de alguna compañía. De esta forma el repositorio crece con el tiempo y de ser necesario se puede hacer una traza para saber qué usuario ha insertado un dato en el repositorio que no cumple con el nivel de calidad o confiabilidad deseado para posteriormente dar aviso y hacer sugerencias que eviten que suceda de nuevo. Básicamente el trabajo de mejoramiento del repositorio es un trabajo colaborativo pero coordinado por el prestador del servicio.

La tabla RepositorioPrincipal registra la información oficial o aceptada como correcta para cada ejecutable que puede clasificar automáticamente el sistema, a esta tabla la denominamos el Repositorio Principal. Las columnas o atributos de esta tabla son las que realmente aportan al proceso de clasificación y su unión con las demás tablas de apoyo, ofrecen una herramienta completa para lograr la asociación semiautomática de ejecutables con el software al cual pertenecen.

La tabla **Compania** permite tener un listado de ejecutables y software asociado para cada compañía. Gracias a su existencia es posible a futuro para el sistema generar reportes adaptados a las necesidades de cada compañía. Las columnas que contiene son:

- **compania_id**: es la llave primaria de la tabla. Sólo tiene como funcionalidad identificar cada registro de la tabla.
- **compania_nombre**: el nombre que caracteriza a una compañía. El tamaño de este campo no es muy grande ya que es muy raro que el nombre de una compañía supere los 100 caracteres.

En la tabla **Usuarios** se guardan los datos de autenticación de los expertos que están asociados a cada compañía en particular y que pueden gestionar la clasificación de ejecutables de equipos de su compañía y apoyar en el crecimiento y mejoramiento del repositorio principal. Las columnas que la componen son las siguientes:

- **usuario_id**: contiene el identificador de cada registro.
- **usuario_nombre**: contiene el nombre de usuario que un experto usa en el inicio de sesión en la herramienta para el experto.

- **usuario_password**: guarda la contraseña correspondiente al nombre de usuario. Su valor es guardado de forma cifrada para evitar problemas de seguridad en caso de que un usuario mal intencionado tenga acceso visual a los registros de esta tabla.
- **compania_id**: este es el id de la compañía a la que está asociado este usuario para que de esta forma no le sea posible ver registros de compañías ajenas y lo que esto conlleva.

La tabla **Equipo** permite guardar los datos básicos de los equipos pertenecientes a cada una de las compañías que usan el software. Las columnas que contiene son:

- **equipo_id**: llave primaria de la tabla, permite identificar cada uno de los registros de manera única.
- **equipo_nombre**: es generado a través de un algoritmo que usa el nombre identificador del equipo en el sistema operativo y un valor UUID que asegura que el nombre no se repita.
- **comp_id**: el identificador de la compañía a la cual pertenece el equipo.

La tabla **Ejecutables** es la que permite llevar el inventario de software de cada uno de los equipos de las compañías que usan el software, además de ella se extrae la lista de los ejecutables que el SRI no pudo clasificar automáticamente y que se deben pasar a la Herramienta para el Experto. Cada vez que un registro es clasificado de forma definitiva, queda asociado a un registro en la tabla EjecutablesFinal. Las columnas que la componen son:

- **id_ejecutable**: llave primaria de la tabla, permite identificar cada registro de manera única.
- **nombreEjecutable**: el nombre del archivo ejecutable, por ejemplo "devenv.exe".
- **path**: es la ruta asociada al archivo ejecutable. Este valor puede ser bastante extenso, por lo cual se le da un espacio amplio de almacenamiento.
- **tamanio**: el tamaño en bytes del archivo ejecutable.
- **producto**: normalmente un archivo ejecutable tiene inmerso dentro de sí mismo un valor de producto al cual pertenece. Este valor está más asociado al programa que lo usa, más no a lo que se considera propiamente como el software dentro del proyecto.
- **compania**: la empresa, casa o persona desarrolladora del ejecutable.
- **versión**: versión de compilación del ejecutable. Este valor no se puede trabajar como un número directamente pues puede contener texto o tener varios puntos en medio, por ejemplo 2.45.4687.2.
- **estado**: este es un campo que le permite a la “Herramienta para el Experto” determinar si es necesaria su clasificación y si se ha clasificado de forma temporal o definitiva. Los estados posibles son “P” (Por clasificar), “T” (Temporalmente Clasificado) y “C” (Clasificado de forma definitiva).
- **eq_id**: el id del equipo al que pertenece el archivo ejecutable.

- **ejecutableF_Id**: el id de uno de los registros en la tabla EjecutablesFinal. Es un valor que puede ser nulo ya que se llena una vez se confirma de forma definitiva la clasificación del ejecutable en la Herramienta para el experto.

La tabla **Sugerencias** contiene los registros asociados a un ejecutable en particular y que fueron generados durante el proceso de clasificación automática. Esta tabla es una tabla de soporte al proceso de clasificación que hace el experto. La existencia de sus registros se limita a un proceso de clasificación, es decir, una vez un ejecutable se ha clasificado de forma definitiva, todas sus sugerencias asociadas dejan de existir (se eliminan). Cuenta con las mismas columnas que la tabla Ejecutables, con la diferencia de que se adicionan las columnas que se refieren al porcentaje de similitud de cada una de las columnas respecto a la correspondiente a uno de los registros en la tabla Ejecutables, por ejemplo, en la tabla Ejecutables un registro tiene la columna compañía, en la tabla Sugerencias hay 5 registros relacionados al registro en la tabla Ejecutables, cada uno con un valor de compañía y un porcentaje de similitud entre la compañía del registro de la tabla Sugerencias y la compañía del registro de la tabla Ejecutables.

4.2 RECOLECCIÓN DE DATOS CON WEB SCRAPING

Inicialmente, se creó una base de datos usando un proceso de Web Scraping a la fuente más confiable y adecuada que se encontró. Con esto se buscaba una fuente que sirviera para llenar el repositorio principal del SRI y para esto se hizo una investigación de diferentes sitios web que ofrecían información referente a ejecutables y sus datos relacionados, obviamente con un énfasis en que estuviera el software al cual pertenecía el ejecutable del que mostraran información. Este proceso mostró resultados no muy alentadores, ya que muchas de las fuentes encontradas tenían datos genéricos, inconsistentes y poco confiables, sin embargo, hubo una fuente que no sólo contenía una gran cantidad de información, sino que la información es bastante confiable, esa fuente es ProccessList.com.

Este sitio web (ProccessList) sirve como un compendio de información no sólo de archivos ejecutables sino también de DLL's, siendo de estas últimas de las que más registros tiene. Su información se encuentra debidamente indexada y ordenada por orden alfabético y debido a la organización y estructura de las páginas, facilita el proceso de Web Scraping.

Los datos que ofrece sobre los ejecutables son el nombre del ejecutable, la versión, el tamaño, la compañía, una descripción del ejecutable y dos campos que llamaron la atención que son un código MD5 y un código SHA (ver ejemplo en la **Figura 12**). Estos códigos sirven "en teoría" para identificar el software, pero debido a la forma en que se generan, ante el más mínimo cambio en el archivo ejecutable antes de generarse cualquiera de estos códigos, estos cambian completamente.

Para poder ejecutar el proceso de Web Scraping es necesario tener los enlaces donde se encuentra la información que se quiere extraer o si estos son generados

usando alguna lógica en especial, la cual se debe conocer para incluirla en la lógica del Scraper. Como estos enlaces en ProcessList.com no siguen ningún patrón claro, fue necesario ejecutar un programa de Javascript en cada uno de los índices y subíndices de la página con el objetivo de extraer los enlaces del HTML y agruparlos en un archivo que posteriormente consultaría el Scraper.

Una vez se terminó de construir el archivo con los enlaces de cada una de las páginas con información sobre alguno de los ejecutables, se procedió a desarrollar el Scraper. Se hizo una investigación de las opciones disponibles para este desarrollo y se optó por implementar esta funcionalidad en Python inicialmente haciendo uso de una librería llamada Beautiful Soup [43] que es una librería especializada para Web Scraping, sin embargo, debido a la simplicidad del html de las páginas de información de los ejecutables, Beautiful Soup terminaba por complicar innecesariamente la extracción de la información así que sólo se usó una librería llamada “re” [44].

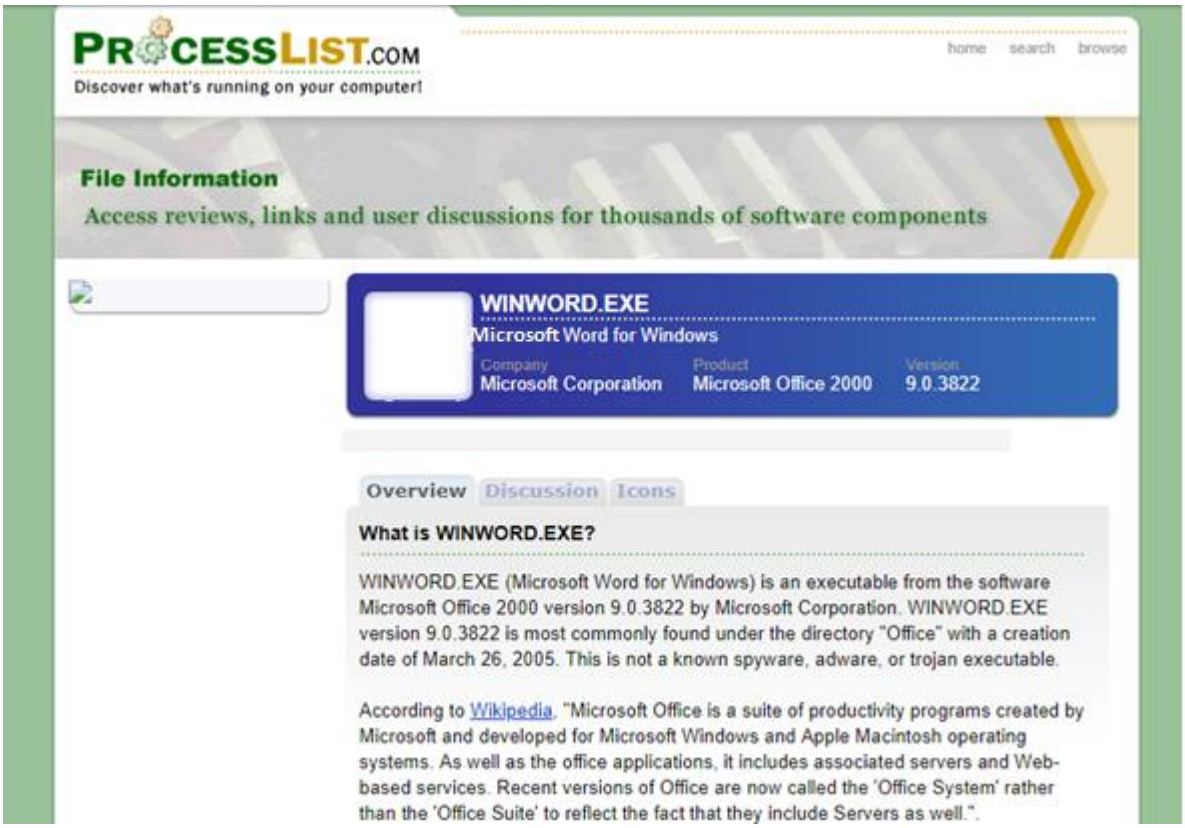

Figura 12 *Información del archivo WinWord.exe presentada en ProccessList.com*

Gracias a la ejecución del Scraper sobre los enlaces obtenidos de ProcessList.com se logró obtener información de aproximadamente 80.000 ejecutables. Estos datos fueron revisados y filtrados para omitir registros que no tuvieran información del software al que pertenecían o fueran en realidad registros de DLL’s; al final de la revisión se obtuvieron aproximadamente 28.000 registros.

Los registros resultantes, aunque en general están completos, tiene una gran debilidad y es que los registros son antiguos, los más recientes sólo datan del 2010, y por esta razón se decidió tomar esta base de datos como una apoyo y proceder a crear una nueva que si sirviera y fuera el repositorio principal del SRI.

Los registros obtenidos mediante este proceso son usados como los Datos de apoyo que se usan en la selección de sugerencias para el experto hecho por el Clasificador principal.

Con el desarrollo de la base de datos para el repositorio y la base de datos de ayuda se cumple con el primer objetivo establecido para el presente proyecto.

CAPÍTULO 5

5 CLASIFICADOR PRINCIPAL

Es el algoritmo encargado de realizar las clasificaciones de los archivos ejecutables de manera automática, en caso de poder hacerlo o de encontrar las diferentes sugerencias que pueda usar el experto cuando se trabaja con instancias difíciles de clasificar.

El funcionamiento básico del clasificador se rige por el pseudocódigo que se encuentra en la **Figura 13**. Primero, en la línea 1, se busca una instancia en el repositorio que tenga por lo menos el nombre del ejecutable igual al que se quiere clasificar. En la línea 3 se inicia una bandera en falso que sirve para identificar si en el proceso se logró hacer una clasificación automática o por el contrario ninguna de las instancias en el repositorio alcanzó el porcentaje de similitud deseado, en cuyo caso se entrega un listado con las sugerencias que pueden mostrársele al experto en la interfaz gráfica de la “Herramienta para el experto”.

Entrada: Instancia a clasificar
listado vacío de sugerencias
PCA (Porcentaje de Clasificación Automática)
PSS (Porcentaje de Selección de Sugerencia)

Salida: Instancia clasificada o sugerencias para el experto

```
1.  instanciaRepositorio = ObtenerInstanciaRepositorio ()
2.  bandera = falso
3.  Mientras instanciaRepositorio != Nulo hacer
4.      porcentajeSimilitud = CalcularSimilitud (instanciaClasificar, instanciaRepositorio)
5.      Si porcentajeSimilitud >= PCA entonces
6.          InstanciaClasificar.EjecutablesFina_Id = instanciaRepositorio.id
7.          bandera = verdadero
8.          Salir de Mientras
9.      Sino Si porcentajeSimilitud >=PSS entonces
10.         listadoSugerencias.Agregar (instanciaRepositorio)
11.     Fin Si
12.     instanciaRepositorio = ObtenerInstanciaRepositorio ()
13. Fin Mientras
14. Si bandera == falso entonces
15.     NotificarSugerencias (listadoSugerencias)
16. Fin si
17. Retornar instanciaClasificar
```

Figura 13 *Pseudocódigo del Clasificador Principal*

Luego, en la línea 4 se inicia un ciclo Mientras para cada una de las instancias del repositorio principal que coinciden con la instancia a clasificar, y se calcula una a una su porcentaje de similitud. El valor considerado adecuado para decir que la clasificación se acepta en forma automática viene dado por el parámetro PCA, para la implementación de este proyecto se consideró el 90% como un valor adecuado. Si bien lo ideal es que sea del 100%, con un 90% los resultados experimentales mostraron que se alcanza la confianza necesaria para realizar la asociación correcta entre ejecutable/software. Para el caso en que se va a crear una nueva sugerencia se opta por el porcentaje dado por el parámetro PSS que para la implementación de este proyecto se tomó el valor de 70% de similitud, esto debido a que, si se agregan sugerencias con un porcentaje bajo de similitud, en lugar de ayudar al experto lo que se logra es confundirlo o hacer que tienda a tomar una mala decisión al tener que revisar muchas opciones que no son las más apropiadas.

En el momento en que se quiera agregar una nueva sugerencia, si el listado de sugerencia tiene aún espacio, se agregada directamente, en caso contrario, esta nueva sugerencia debe ser contrastada con las actuales y sólo se agrega si es mejor que alguna de las que ya se tienen, de esta forma la lista de sugerencias contiene al final, la información más útil posible para el experto.

El porcentaje de similitud se calcula de acuerdo con el algoritmo que se utilice. Normalmente se hacen cálculos de distancias entre cadenas de caracteres y para obtener esta distancia existen algoritmos como la distancia Euclidiana [45], Brecha Afín [46], N-Gramas [47] o la distancia de Levenshtein [48], esta última es la que se usa en el desarrollo del clasificador de esta investigación.

El método NotificarSugerencias (listadoSugerencias) envía al IRSWCFWebService el listado de sugerencias para esa instancia a clasificar. Además de esto si se retorna en el método una instanciaClasificada en la que el campo de software sea vacío o nulo, se confirma por parte del IRSWCFWebService que la clasificación no se pudo hacer de manera automática.

Con el objetivo de vincular este componente al SRI se desarrolló una clase llamada Classifier con los siguientes métodos:

int SeparatorCounter (const std::string &inString): Teniendo en cuenta que el listado de ejecutables con su información asociada se encuentra en un archivo **CSV** (coma separated value), se hizo necesario unos métodos para trabajar con los datos en este tipo de archivo. En este caso se trata de un método que cuenta cuántas veces aparece el carácter separador ‘,’ que distingue cada uno de los campos y que es usado para cargar un vector dinámico con los datos de cada instancia a clasificar.

void Split (const std::string &line, std::vector<std::string> &instanceBuffer): Como la lectura de los datos del archivo se hace línea a línea, y que cada línea corresponde a una instancia a clasificar, esté método permite separar cada uno de

los campos asociados a un ejecutable para que se puedan hacer comparaciones con las instancias encontradas en el repositorio.

int LevenshteinDistance (const std::string &firstInString, const std::string &secondInString): Este método permite obtener el valor de la distancia de Levenshtein entre dos cadenas de entrada, para este caso serían el mismo campo en una instancia a clasificar y una instancia en el repositorio. La distancia entre dos cadenas, cadenaA y cadenaB, obtenida por este algoritmo corresponde a la cantidad de operaciones que hay que hacer para convertir la cadenaA en la cadenaB, esto incluye modificar, mover, eliminar o adicionar uno o más caracteres. Por ejemplo, si cadenaA fuera "Jose" y cadenaB fuera "Joselo", el resultado de aplicar la distancia de Levenshtein es 2, pues hay que hacer dos operaciones de adición para convertir "Jose" en "Joselo".

void Classification (): clasifica una instancia de acuerdo con la información hallada en el repositorio principal. También determina si una instancia no se pudo clasificar de forma automática y añade un conjunto de sugerencias de acuerdo con la menor distancia encontrada con las instancias del repositorio al invocar al método **AddSuggestion()**. En este método es donde se incluye el pseudocódigo de la **Figura 13**.

int GetFails (): es un método utilitario que permite conocer, en caso de desearlo, el total de instancias que no se pudieron clasificar automáticamente.

Void AddSuggestion (std::vector<std::string> &instanceBuffer): este método se utiliza para analizar si una instancia debe ser usada como sugerencia para la clasificación manual de un ejecutable, esto lo hace de acuerdo con las distancias en los campos respecto a instancias en el repositorio y a las sugerencias que ya existan del mismo ejecutable. El envío de una instancia como sugerencia no implica que esta sea agregada, además de esto se deja un campo para agregar sugerencias de los datos de apoyo.

void StartClassification (): es el que lleva el flujo principal del proceso de clasificación. Se encarga de leer el archivo donde se encuentran los ejecutables que deben ser clasificados e ir enviado cada instancia para saber a qué software pertenece o si debe ser enviada al experto para su posterior clasificación manual.

5.1 PROCESO DE SELECCIÓN DEL CLASIFICADOR PRINCIPAL

Para poder llegar a la definición del clasificador principal, fue necesario desarrollar unos experimentos que permitieran determinar entre los algoritmos de clasificación candidatos, cuál era el que presentaba mejores resultados y mejor desempeño. Estas pruebas usaron los datos extraídos manualmente de 8 computadores haciendo uso de una herramienta llamada FileAlyzer [49] para obtener un total de 331 instancias de prueba. Los datos obtenidos con la herramienta contaban con las características: nombre el ejecutable, tamaño, path, compañía, producto y un código CRC32 que se generaba al analizar cada ejecutable. Estos datos se

prepararon en un dataset para ser trabajados con los algoritmos presentes Weka (Waikato Environment for Knowledge Analysis) [50]. Una vez realizado este proceso, se hicieron pruebas de validación cruzada con cada algoritmo, los resultados se aprecian en la **Figura 14**.

Algoritmo	Instancias Correctamente Clasificadas	PICC	Estadística Kappa
KNN	267	80,66	0,79
Árbol J48	239	72,2	0,69
Naive Bayes	237	71,6	0,69
K-Star	200	60,42	0,58
AdaBoostM1	79	23,86	0,14
Zero-R	68	20,54	0

PICC: Porcentaje de instancias correctamente clasificadas

Figura 14 *Evaluación de diversos algoritmos de clasificación usando Weka (Fuente propia)*

De los resultados de la clasificación usando los diversos computadores, los valores más importantes corresponden al porcentaje de instancias correctamente clasificadas y el tiempo de ejecución. En cuanto al porcentaje de instancias correctamente clasificadas se observa claramente que la mejor opción es K-nn, además, teniendo en cuenta que el problema implica que las clases crecen, así como los datos de los ejecutables en los equipos, la elección de K-nn se soporta más, ya que no requiere crear modelos que se deban estar actualizando cada vez que llegue una nueva clase o cierta cantidad de registros nuevos.

Por el lado de los tiempos de ejecución las pruebas muestran que todos se desempeñan a muy buena velocidad. Es preciso tener claro que K-nn se ve afectado por la cantidad de registros existentes en el repositorio, por esta razón, se deben establecer las medidas apropiadas para que ellos se puedan acceder y manipular lo más rápidamente posible, haciendo uso por ejemplo de cache en memoria de la tabla “RepositorioPrincipal”.

También se hicieron experimentos para determinar cuál era el valor de K que podía entregar los mejores resultados. Los datos encontrados se presentan en la **Figura 15**. Como se puede apreciar la tendencia es que, con un mayor valor de K, los resultados pierden precisión, por esta razón el valor elegido es K = 1. Este resultado sigue la lógica del problema que se aborda.

Valor de K	Instancias Correctamente Clasificadas	PICC	Estadística Kappa
1	267	80,66	0,79
2	237	71,6	0,69
3	212	64	0,61
4	195	58,9	0,55

Figura 15 *Variación del valor de K en el algoritmo K-nn (Fuente propia)*

El pseudocódigo del algoritmo 1-nn usado en este proyecto se presenta en la **Figura 16**. Como se puede apreciar este algoritmo y el algoritmo que se presentó

previamente como el clasificador principal en la **Figura 13** son muy parecidos debido a que fue el que finalmente se seleccionó como componente del SRI.

Del algoritmo hay que tener en cuenta que se está usando la variación 1-nn de K-nn, es decir, se está usando sólo un vecino más cercano y aquellas instancias que también son lo suficientemente cercanas no son tenidas en cuenta para la decisión automática pero sí son tenidas en cuenta como sugerencias para ser enviadas al experto. Bajo circunstancias normales cuando se tiene un K-nn y se tienen varios vecinos cercanos, se hace una votación donde la etiqueta de clase dada a la instancia que se está clasificando sale de la etiqueta que tengan la mayoría de los vecinos cercanos usados o por una votación ponderada. Para este caso en concreto, como se está usando sólo un vecino más cercano, se está optando únicamente por aquel vecino que sea prácticamente idéntico (distancia = 0) a la instancia por clasificar; esto se hace debido a que es la forma en que se puede decir con seguridad que la clasificación es acertada y que el proceso se puede considerar automático.

La distancia entre una instancia por clasificar y una que se encuentra en el repositorio se calcula haciendo uso de la distancia de Levenshtein [48]. La razón de usar este método para calcular distancias se basa en los resultados encontrados en [51] donde se prueban cerca de 10 métodos distintos para calcular distancias de los más usados y se concluye que no se puede definir a uno de ellos como “mejor” que el otro. En este estudio se realizaron evaluaciones con diferentes tipos de entradas que facilitaran observar el comportamiento de cada método en distintos escenarios para calcular la distancia, sus fortalezas y debilidades. Como resultado algunos funcionan mejor en ciertos escenarios que otros y esos otros mejor en otros escenarios, pero para el escenario del problema tratado en el proyecto el método de la distancia de Levenshtein ofrece los mejores resultados.

Entrada: Instancia a clasificar
listado vacío de sugerencias
distancia menor

Salida: Instancia clasificada o sugerencias para el experto

```
1.  instanciaRepositorio = ObtenerInstanciaRepositorio ()
2.  distanciaActual = 0
3.  Mientras instanciaRepositorio != Nulo hacer
4.      distanciaActual = CalcularDistancias (instanciaClasificar, instanciaRepositorio)
5.      Si distanciaActual == 0 entonces
6.          instanciaClasificar. EjecutablesFinal_Id= instanciaRepositorio.Id
7.          distanciaMenor = 0
8.      Sino Si distanciaActual < distanciaMenor entonces
9.          listadoSugerencias.AgregarSugerencia (instanciaRepositorio)
10.         distanciaMenor = distanciaActual
11.     Fin Si
12.     instanciaRepositorio = ObtenerInstanciaRepositorio ()
```

```
13. Fin Mientras
14. Si distanciaMenor != 0 entonces
15.       NotificarSugerencias (listadoSugerencias)
16. Fin si
17. Retornar instanciaClasificar
18.
```

Figura 16 *Pseudocódigo de 1-NN adaptado en el SRI (Fuente propia)*

Algo que también se debe clarificar en el método de cálculo de distancia es que como la ruta en la que está un ejecutable puede diferir muy fácilmente de computador a computador y aun así referirse al mismo software, este campo es podado. La poda se realiza quitando de la ruta la sección que no pertenece propiamente al software, por ejemplo, si se tiene una ruta de entrada como esta "C:/Archivos de programa/NetBeans 8.2/ide.exe", el resultado después de la poda es "NetBeans 8.2/ide.exe". La realización de la poda facilita también el trabajo que posteriormente se hace al analizar los directorios.

5.2 Pruebas realizadas al Clasificador Principal

Como se planteó en el segundo objetivo específico, se hicieron pruebas con datos de 20 computadores para poder detallar los resultados del Clasificador principal ante diferentes situaciones. Estas pruebas fueron realizadas con el fin de identificar el nivel de intervención humana que se requeriría para el proceso de clasificación teniendo esta cantidad de datos. Las pruebas fueron realizadas en dos grupos de computadores con características distintas y se aplicó una adaptación de lo que se hace normalmente para probar la efectividad de algoritmos de clasificación, que es el uso de la validación cruzada. En este caso se tomaron grupos de 10 computadores. Para cada computador se tiene ya de antemano la asociación correspondiente a cada uno de los ejecutables y el software al cual pertenecen, es decir, ya se sabe qué resultados debería entregar el clasificador en un caso exitoso. Se procede a tratar de clasificar los ejecutables de un computador usando la información de las clasificaciones de todos los demás computadores sin incluirse a sí mismo. Luego se hace el mismo proceso con el siguiente computador y así sucesivamente de tal forma que los datos de cada uno de los computadores estuvieron ausentes como datos guía para la clasificación al menos una vez y el clasificador al final se evalúa con todos los computadores.

El primer grupo de pruebas corresponde a 10 computadores pertenecientes a Aranda Software, entre los que se encuentran algunos servidores y equipos con gran variedad y cantidad de software instalado. Los resultados de las pruebas sobre estos computadores se muestran en la **Figura 17**.

Dispositivo	Total ejecutables	Total fallidos	Porcentaje fallidos
D2	776	128	16,49484536
D3	1018	144	14,1453831
D10	232	43	18,53448276
D12	199	71	35,67839196
D14	1640	318	19,3902439
D17	1895	195	10,29023747
D19	1445	166	11,48788927
D20	1126	177	15,71936057
D21	596	39	6,543624161
D24	1076	172	15,98513011
	10003	**Promedio Fallidos:**	14,52564231

Figura 17 *Resultados Clasificación Computadores Aranda (Fuente propia)*

La primera columna muestra un nombre dado a cada uno de los equipos. La segunda el total de ejecutables a clasificar por el equipo. Luego el total de ejecutables que no pudieron ser clasificados automáticamente debido a que sus características no fueron suficientemente similares a algún registro dentro del repositorio principal. Por último, el porcentaje de fallidos para cada equipo y el porcentaje de fallidos a nivel global.

El segundo grupo de pruebas corresponde a 10 computadores que pertenecen a un local comercial de los que son conocidos como “Café Internet”. Estos computadores tienen muchas características distintas respecto a los que se probaron de Aranda. Por un lado, no tiene la cantidad tan grande de ejecutables en cada dispositivo, por otro lado, los computadores al ser de acceso público, por cuestiones de seguridad, son retenidos en un estado básico gracias al uso del software Deep Freeze [52], teniendo instalado sólo el software que la mayoría de las personas usa como los navegadores web y la suite de Microsoft Office además de una que otra utilidad. Los resultados de las pruebas en estos computadores se muestran en la **Figura 18**:

Dispositivo	Total ejecutables	Total fallidos	Porcentaje fallidos
PC1	37	31	83,78378378
PC2	82	53	64,63414634
PC3	36	31	86,11111111
PC4	62	49	79,03225806
PC5	16	12	75
PC6	38	26	68,42105263
PC7	24	19	79,16666667
PC8	12	9	75
PC9	49	32	65,30612245
PC10	213	202	94,83568075
	569	**Promedio Fallidos:**	81,54657293

Figura 18 *Resultados Clasificación Computadores Café Internet (Fuente propia)*

Como es evidente, los resultados distan muchísimo de los obtenidos con los computadores de Aranda Software. Esto se debe principalmente a dos factores, la cantidad y la variedad de ejecutables y software que está instalado en estos equipos. Esto evidencia la forma en que funciona el Clasificador principal. K-NN no necesita de un entrenamiento, es un algoritmo de clasificación que se alimenta de los datos que tiene como guía o ejemplo para así decidir qué etiqueta poner a una instancia. Entre más datos de guía haya, más fácil va a ser para el clasificador determinar, en este caso, a qué software pertenece un ejecutable.

En cuanto a lo que respecta a la intervención de un experto en el proceso de clasificación, se puede ver que la tendencia, a medida que el Repositorio Principal crezca en cantidad, calidad y variedad de datos, es a disminuir ya que el clasificador va a ser progresivamente más calificado para realizar una clasificación automática. Es prácticamente imposible no tener que usar la intervención de un experto en el proceso y los momentos donde más se necesita es cuando se realiza por primera vez la clasificación de los ejecutables de un computador, pero esto se reduciría enormemente para las siguientes veces ya que sólo tendría que lidiar con el nuevo software instalado en el periodo comprendido entre la anterior clasificación y la nueva.

Con el desarrollo del clasificador y la ejecución de las pruebas, se cumple con el segundo objetivo específico.

CAPÍTULO 6

6 HERRAMIENTA PARA EL EXPERTO

Cuando no hay forma de asegurar que un ejecutable pertenece a un determinado software a través del método automático presentado previamente, se hace necesaria la intervención de un experto para que el proceso de clasificación pueda llegar a feliz término. La herramienta para el experto consta de tres elementos principales:

- Un modelo que está asociado con las tablas de la base de datos del Repositorio y las tablas de apoyo para el proceso (Compañía, Usuarios, Equipo, Ejecutable, Sugerencia).
- Un componente controlador que se encarga de gestionar los procesos de la herramienta y de aplicar un algoritmo sencillo de agrupación por las características de los archivos para mostrar las sugerencias.
- Una interfaz gráfica que le permite al experto visualizar las sugerencias, agruparlas de acuerdo con diferentes criterios y poder así tomar su decisión final.

Esta herramienta se desarrolló aplicando el patrón de diseño MVC (ver **Figura 19**).

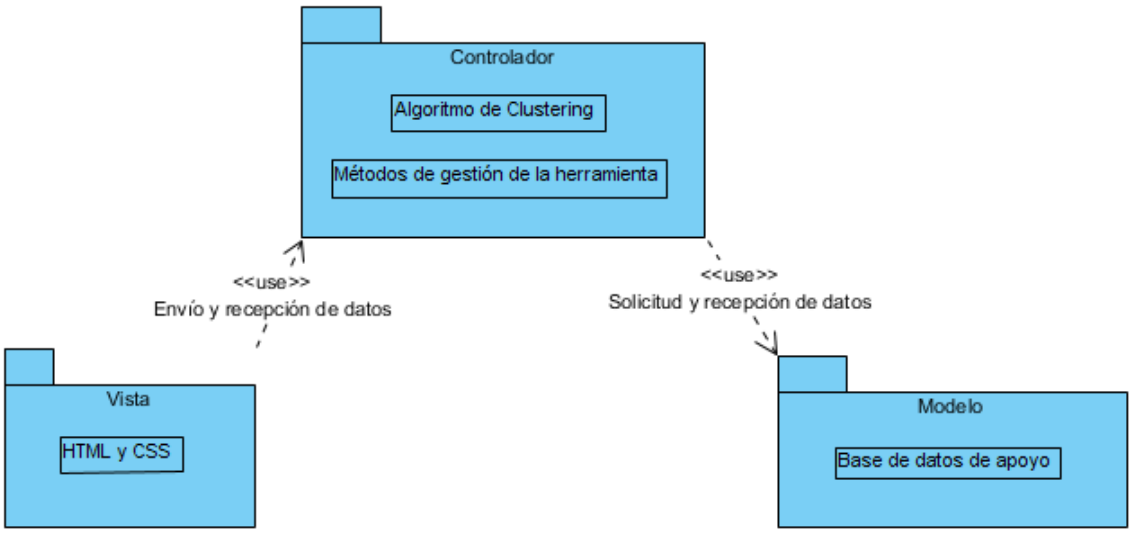

Figura 19 *Modelo MVC usado para el desarrollo del SRI (Fuente propia)*

Esto se desarrolló usando Visual Studio y el framework MVC .NET. La interfaz gráfica de usuario para la herramienta fue modelada haciendo uso del software Balsamiq Mockups 3, ya que facilita crear interfaces y permite además darles a los componentes visuales cierta funcionalidad principalmente de navegación. En la sección 6.2 se describe la interfaz gráfica usada para la herramienta del experto.

6.1 DESCRIPCIÓN DE LA GUI

Todo arranca por la pantalla inicial que se aprecia en la **Figura 20**. Acá se ve el listado de ejecutables que tiene el experto por clasificar, al seleccionar alguno se muestran sus detalles y se da la opción de clasificarlo. Luego de presionar el botón clasificar se entra al formulario de la **Figura 21**.

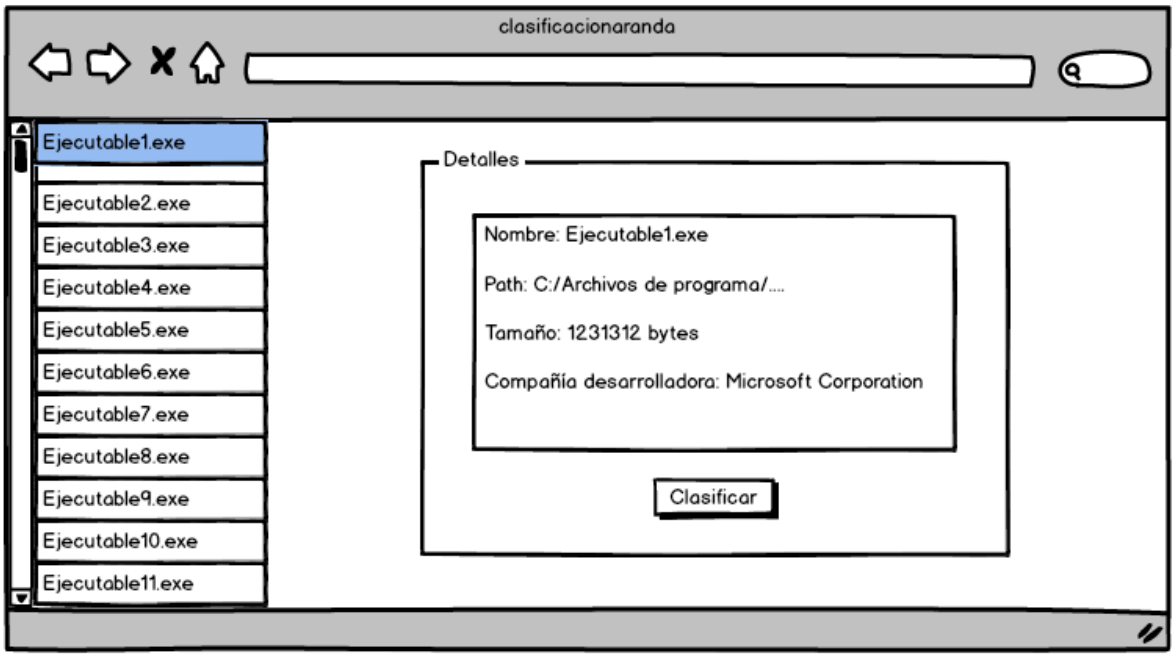

Figura 20 *Pantalla inicial del SRI (Fuente propia)*

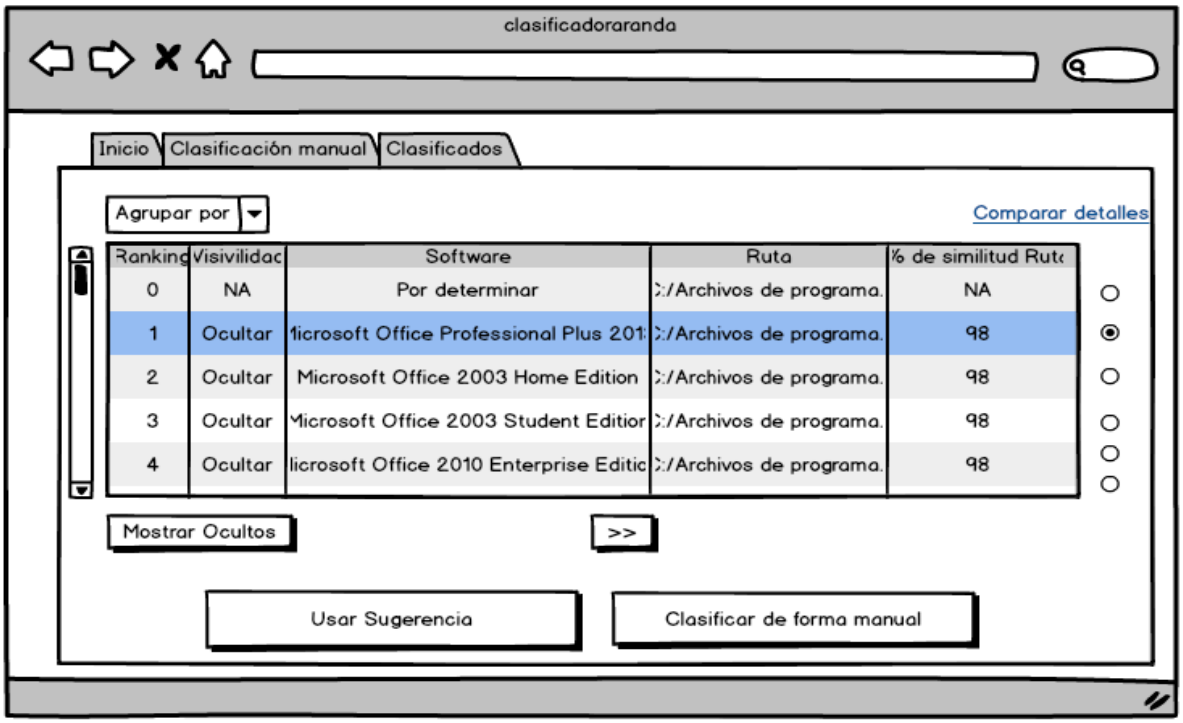

Figura 21 *Agrupación de nuevos ejecutables a clasificar manualmente por el experto (Fuente propia)*

En la parte superior de la **Figura 21** se pueden observar 3 pestañas, “Inicio” que hace referencia a la sección principal donde está la información del ejecutable que se está clasificando y las sugerencias para su clasificación, “Clasificación manual” que es la sección en la que se hace dicha clasificación, debido a que las sugerencias no son satisfactorias y se opta por llenar los datos de ese ejecutable manualmente y “Clasificados” en la que se encuentra un conjunto de ejecutables clasificados a manera de buffer que aún no han sido enviados a la base de datos, esto con el fin de que en caso de que se haya hecho una clasificación errada se pueda reclasificar.

En la sección central se encuentra el ejecutable que está en clasificación, el de Rank = 0, y las sugerencias para su clasificación. Estas sugerencias están organizadas de acuerdo con su similitud con el ejecutable a clasificar y se pueden ver todos sus datos y su similitud. En la tabla se encuentra un botón para ocultar filas, ubicado en la columna “Visibilidad”, para de esta forma poder comparar el ejecutable con sugerencias de manera personalizada y más sencilla, pudiendo por ejemplo tener en pantalla solamente el ejecutable a clasificar y la sugerencia de ranking 3 y 4. Debajo de la tabla a la izquierda está un botón que sirve para visualizar nuevamente aquellas filas que se hayan ocultado; esta funcionalidad de mostrar y ocultar filas no es posible mostrarla en los mockups por limitaciones del programa.

En la parte central bajo la tabla hay uno o más botones que permitirán navegar horizontalmente a través de los datos de las filas de la tabla. Esto se hizo de esta forma ya que los datos (columnas), son muchos y hacer scroll horizontal es algo incómodo.

Por encima de la tabla se encuentra un comboBox que tiene la funcionalidad de separar las sugerencias de acuerdo con uno de los campos como la compañía desarrolladora o la versión.

De esta forma se siguieron haciendo el resto de los mockups para poder desarrollar la interfaz gráfica.

6.2 FUNCIONALIDAD DE LA HERRAMIENTA PARA EL EXPERTO

La ejecución de la Herramienta para el experto inicia con una página de autenticación (ver **Figura 22**). Es una página de inicio similar a cualquier otro sistema con autenticación, pero con la diferencia de que no permite la opción de registrarse, pues los usuarios se agregan en la base de datos por parte de un usuario de nivel superior. Aunque este proceso de autenticación no se contempló en la etapa de diseño, se consideró como algo de vital importancia para proporcionarle a la herramienta un nivel de seguridad básico.

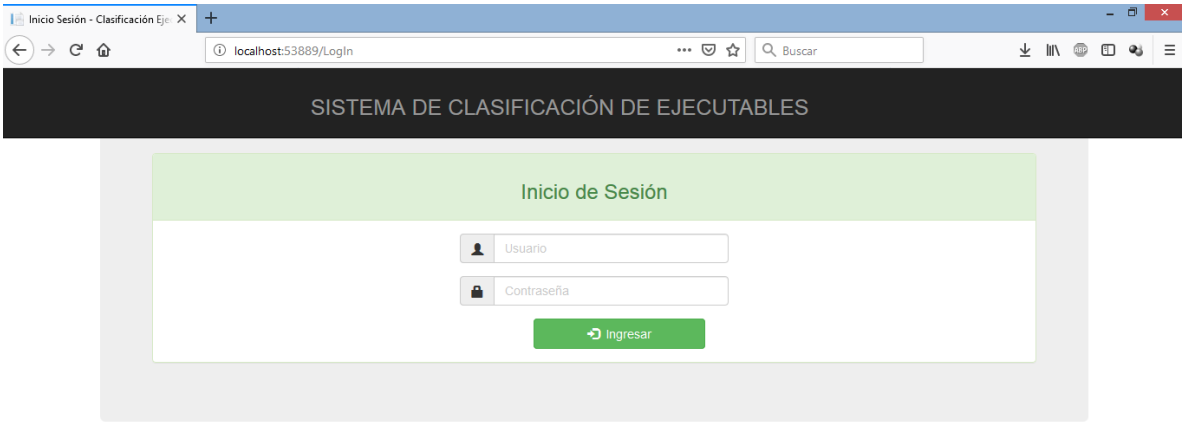

Figura 22 *Pantalla de inicio de sesión Herramienta para el experto (Fuente propia)*

Una vez el usuario ha iniciado sesión puede ingresar a la sección de inicio, donde puede empezar a trabajar. La pantalla que ve se aprecia en la **Figura 23**. En la parte superior se aprecia una sección con el título “Clasificación”. Esta sección está contemplada para adaptarse a cualquier empresa en particular, por ejemplo, agregando un logo que identifique a la empresa que usa la herramienta.

Debajo de esto se pueden ver tres pestañas:

- Inicio: donde se desarrolla el proceso de clasificación de forma normal, seleccionando el ejecutable que se quiere clasificar y usando alguna de las sugerencias entregadas por los resultados del clasificador principal.
- Clasificación manual: como su nombre lo indica, al dar click a esta pestaña se podrá hacer la clasificación manual de un ejecutable sin tener en cuenta las sugerencias dadas por el sistema.
- Clasificados: esta pestaña muestra el total de instancias que han sido clasificadas de manera “temporal” y que, gracias a eso, pueden volver a ser reclasificadas en caso de que se haya producido un error o no se esté satisfecho con el software escogido para asociar el ejecutable.

Luego se puede ver un selector con el listado de ejecutables que tiene el experto por clasificar. Este listado de ejecutables se encuentra separado por el computador al que pertenece de tal forma que para el experto sea más fácil identificar qué es lo que está revisando y pueda agregar algún otro criterio a su selección de serle posible. Esto se puede ver en la **Figura 23** que coincide en mayor medida con lo definido previamente en los mockups.

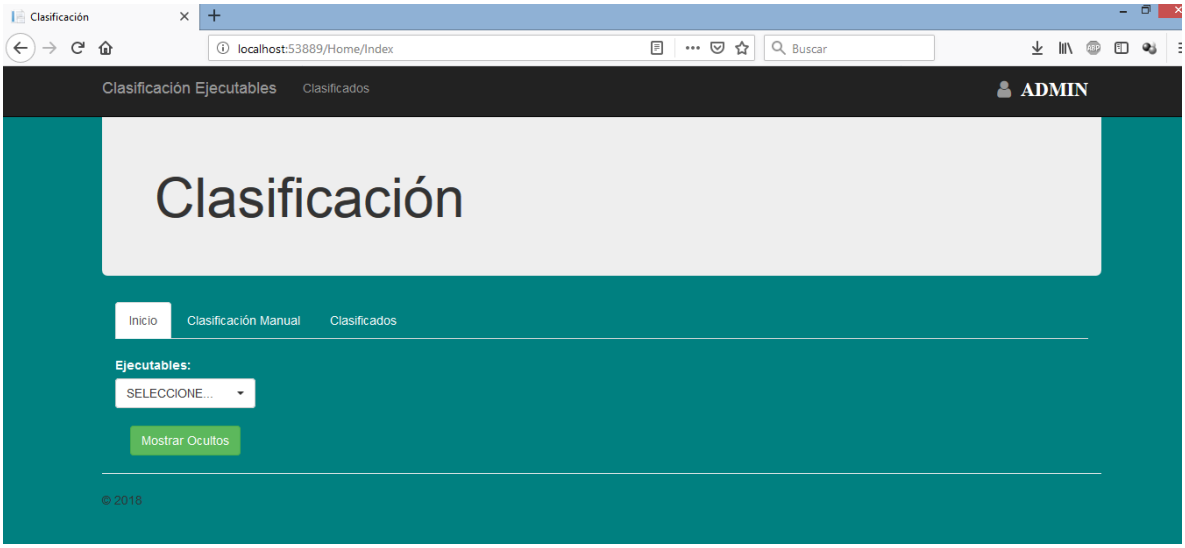

Figura 23 *Pantalla inicial Herramienta para el experto (Fuente propia)*

Una vez seleccionado uno de los ejecutables, el sistema hace una petición para obtener los datos relacionados con ese ejecutable en particular, es decir, todas sus sugerencias asociadas. Estas sugerencias aparecen ordenadas de acuerdo con el ranking, su nivel de similitud respecto a los datos del ejecutable por clasificar (ver **Figura 25**). Como se puede apreciar aparece una tabla con las sugerencias de clasificación. Dentro de esta tabla destacan las 3 primeras columnas que siempre son visibles para el experto. El Ranking que sirve como guía para la decisión de clasificación, la Visibilidad que se maneja de acuerdo con un botón “Ocultar” que oculta la fila seleccionada para permitirle al experto comparar las sugerencias que desee sin tener que ver datos que lo confundan o le generen ruido. A partir de ahí aparecen columnas correspondientes a cada característica de la sugerencia y su porcentaje de similitud respecto a ese mismo dato en el ejecutable que se desea clasificar.

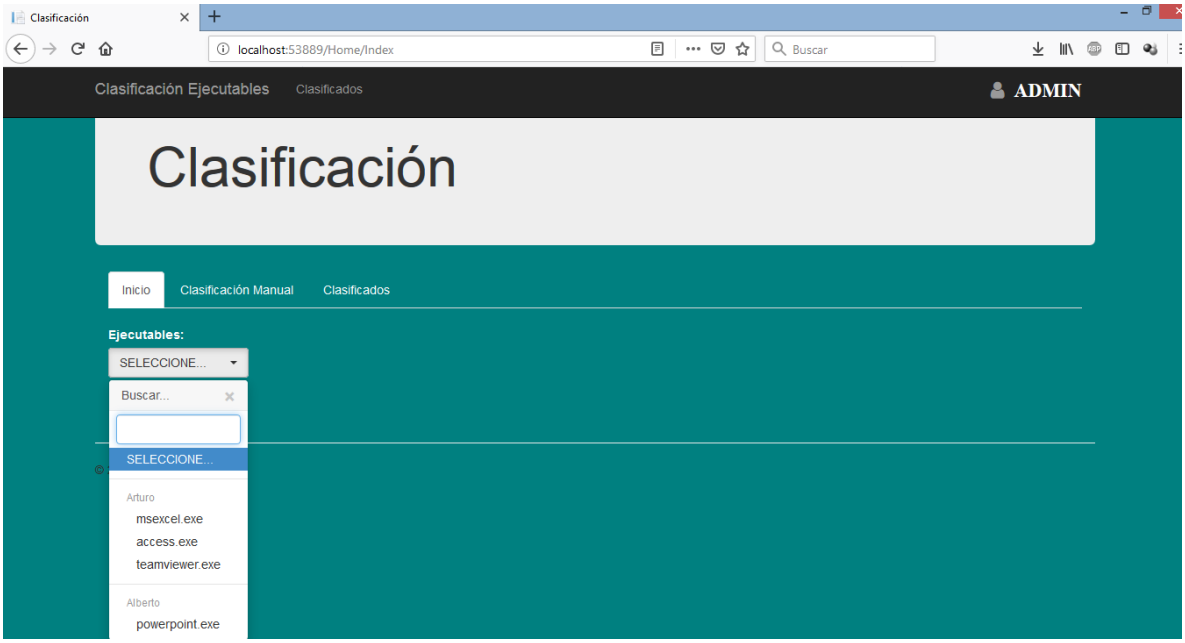

Figura 24 *Selección de ejecutable a clasificar Herramienta para el experto (Fuente propia)*

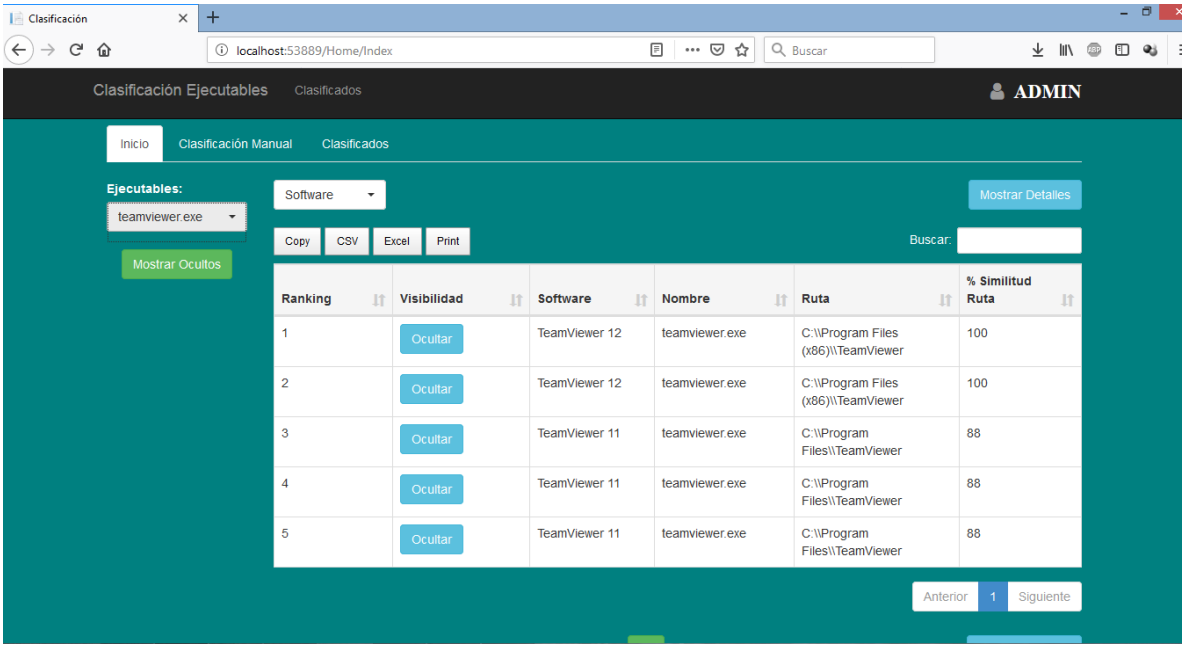

Figura 25 *Vista inicial de sugerencias Herramienta para el experto (Fuente propia)*

Como cada característica de una sugerencia viene acompañada por su porcentaje de similitud, la tabla termina teniendo demasiadas columnas para ser cómodas de leer en una sola pantalla, por lo cual en la parte inferior existe siempre un botón que permite ver los campos no visibles de una sugerencia en un momento determinado (ver **Figura 26**).

Figura 26 *Botón "Usar Sugerencia" Herramienta para el experto (Fuente propia)*

Al presionar este botón se adapta para que su texto cambie para indicar que se puede volver a ver los datos de antes. El resultado de avanzar en los datos usando el botón mencionado se puede apreciar en la **Figura 27**.

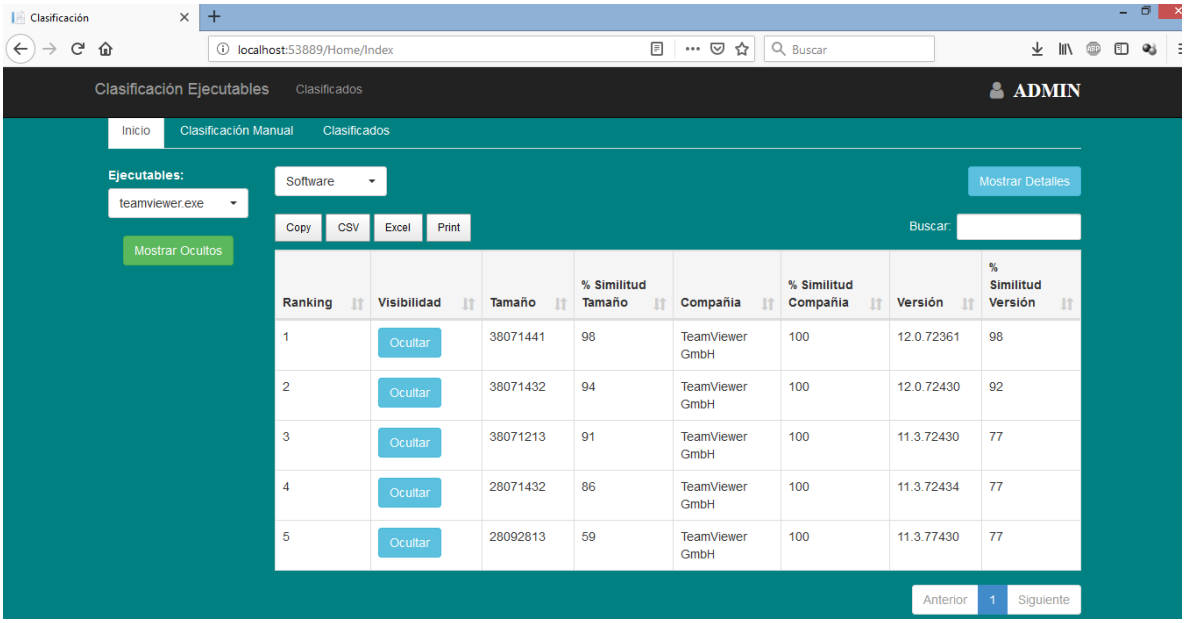

Figura 27 *Visualización de otras columnas de sugerencias Herramienta para el experto (Fuente propia)*

Con estos datos, si el experto así lo determina, se puede proceder a seleccionar una de las sugerencias para que el ejecutable por clasificar adopte su software y se proceda a clasificar un nuevo ejecutable. En caso de que desee hacer un análisis más extenso puede usar la funcionalidad de ocultar filas de la tabla para poder hacer comparaciones entre sugerencias como se ve en la **Figura 28**. Aquí se ocultaron las filas de sugerencias con ranking uno y cuatro. Para restaurar la vista a la tabla completa basta con presionar el botón verde que está a la izquierda que dice "Mostrar Ocultos".

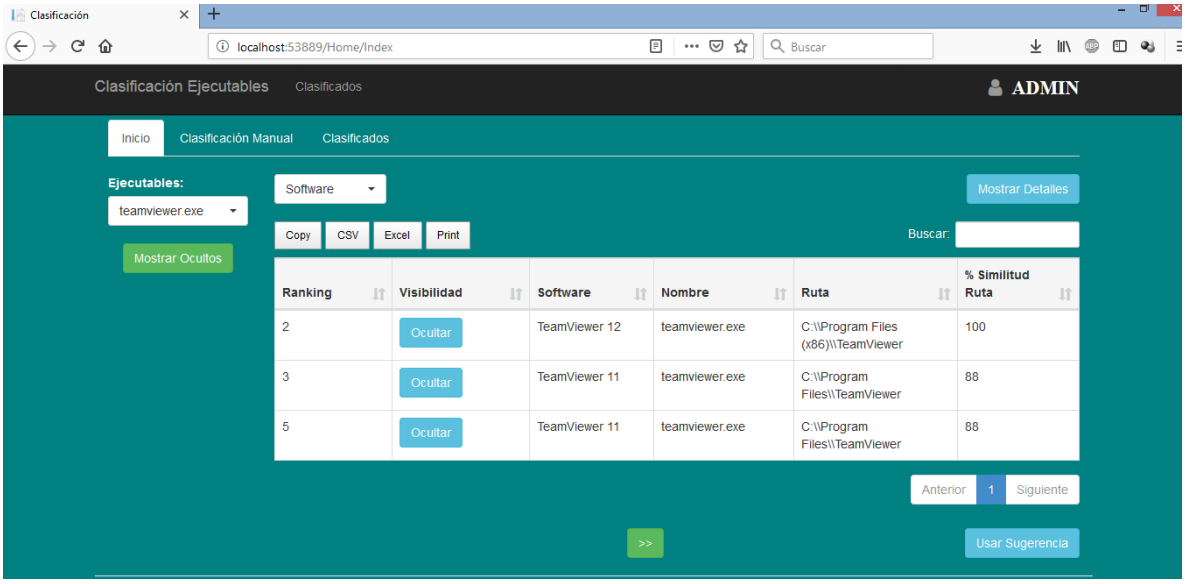

Figura 28 *Sugerencias ocultas Herramienta para el experto (Fuente propia)*

El experto tiene la opción de comparar las características de una sugerencia con las del ejecutable por clasificar sin que tenga que ver los porcentajes de similitud ni los datos de otras sugerencias. Para hacer esto basta con seleccionar la fila de la sugerencia que se quiere contrastar y presionar el botón azul que se encuentra en la parte superior derecha que dice “Mostrar Detalles”. El resultado de esta interacción se puede observar en la **Figura 29**.

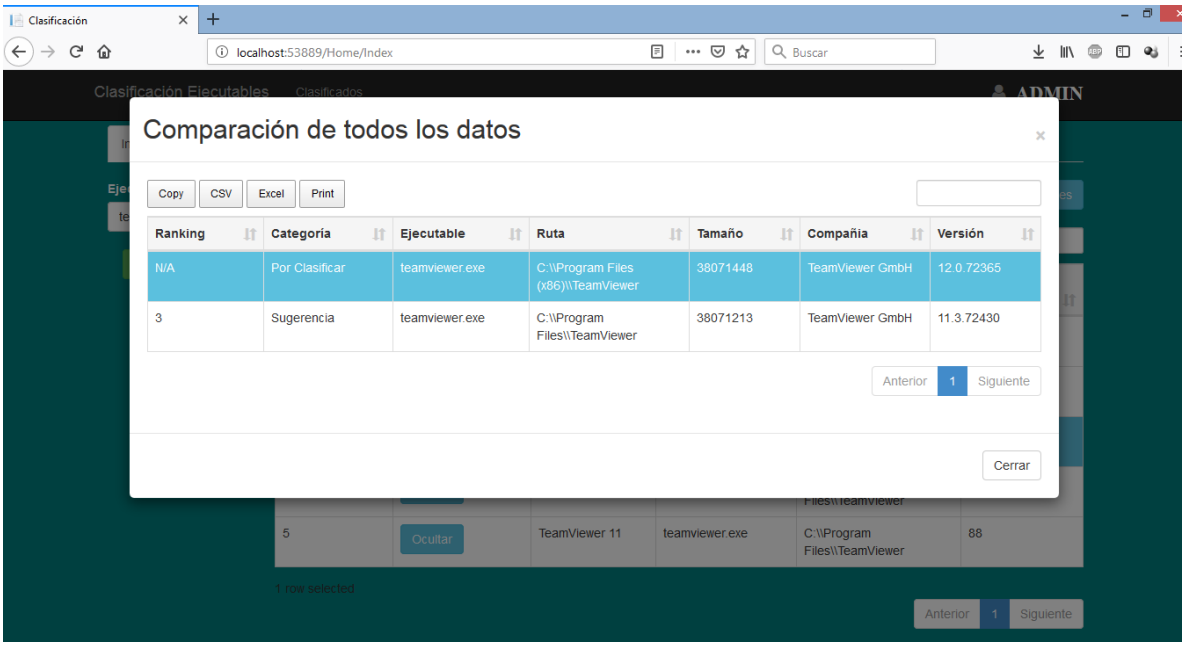

Figura 29 *Mostrar detalles Herramienta para el experto (Fuente propia)*

Otra herramienta que tiene el experto para comparar y analizar las sugerencias es la separación de las sugerencias en grupos de acuerdo con una de las columnas. Para generar los grupos de sugerencias, el experto debe dar click al selector que

se encuentra en la parte superior izquierda de la tabla y elegir la columna de acuerdo con la cual se generarán los grupos. Esta acción hace que aparezcan una serie de tablas dependiendo del número de valores diferentes en la columna seleccionada, por ejemplo, si se selecciona una columna donde todos los valores en cada sugerencia sean diferentes, entonces aparecerán tantas tablas como sugerencias se hayan hecho (cinco en este caso), pero si se selecciona una donde existan sólo dos valores diferentes para esa columna entre las sugerencias, entonces serán dos las tablas en aparecer y así sucesivamente. La **Figura 30** muestra un ejemplo de los resultados obtenidos al formar grupos.

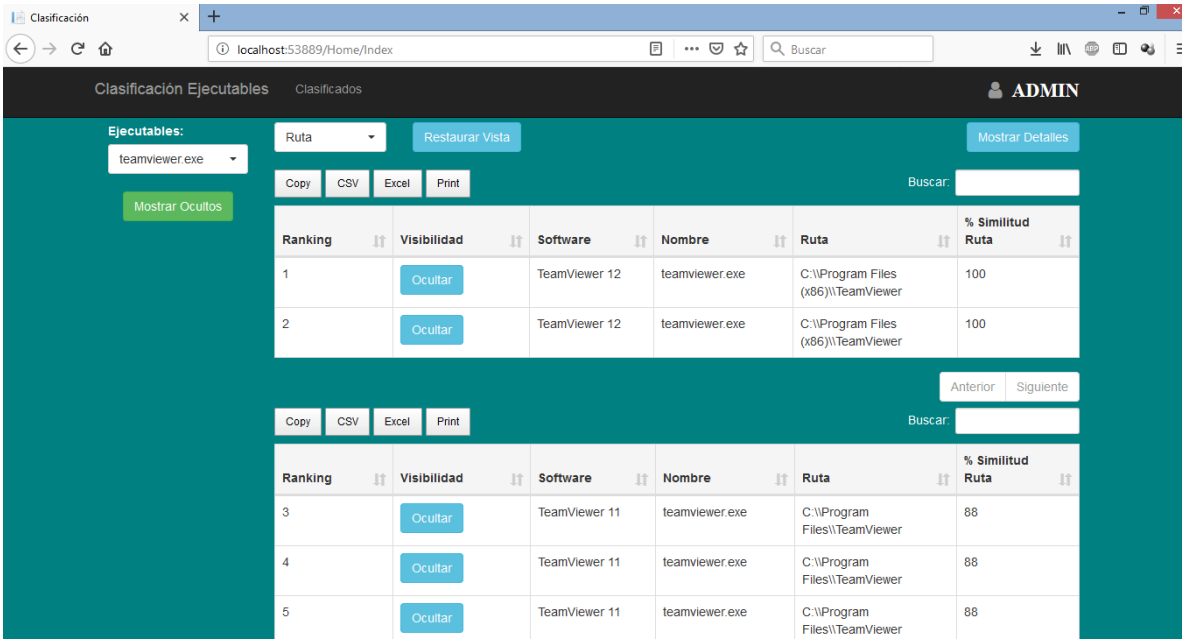

Figura 30 *Visualización de grupos por columna Herramienta para el experto (Fuente propia)*

Cuando se han separado las sugerencias en grupos se puede de igual forma seleccionar filas dentro de alguna de las tablas y usarla para la clasificación con el botón “Usar Sugerencia” o comparar sus detalles con el ejecutable por clasificar al presionar el botón “Mostrar Detalles”. Para regresar a la vista inicial basta con presionar el botón “Restaurar Vista” que está en la parte superior.

En caso de que un experto haya revisado todo y aun así no se sienta satisfecho con las opciones entregadas por el sistema, puede proceder a hacer la clasificación del ejecutable en forma manual. Para esto, el experto debe dar click en la pestaña que dice “Clasificación Manual”. En esta pestaña se hace una precarga de las diferentes características del ejecutable por clasificar de tal forma que se agilice este proceso y no tenga que ponerse a recordar estos datos y lo principal que debe llenar es el software al cual pertenece el ejecutable. A continuación, en la **Figura 31** se puede apreciar un ejemplo.

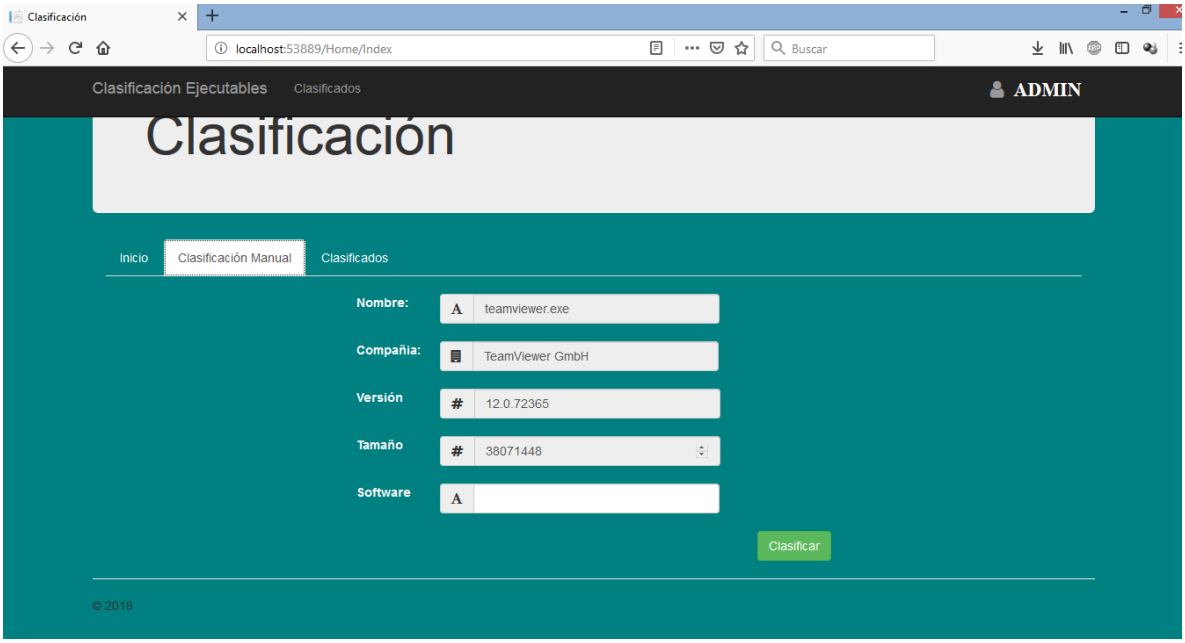

Figura 31 *Clasificación manual Herramienta para el experto (Fuente propia)*

Al llenar el campo de software y presionar el botón “Clasificar” de la parte inferior, este ejecutable pasa a la sección de clasificados y no aparece más dentro del selector a menos de que se quiera reclasificar. La **Figura 32** muestra un ejemplo de la vista en la pestaña “Clasificados”.

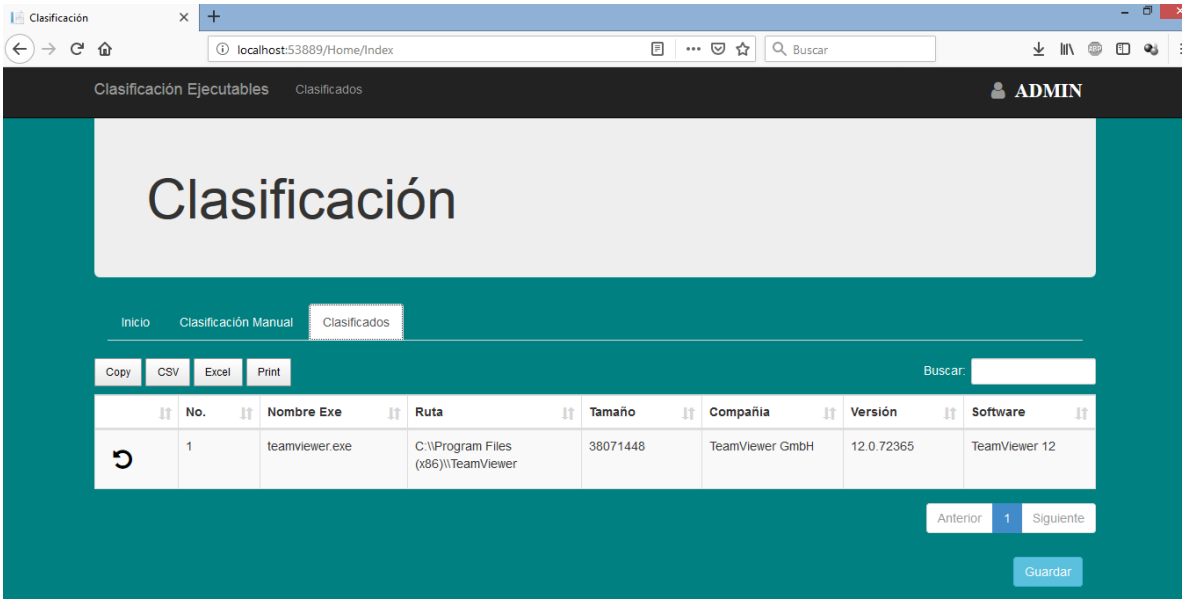

Figura 32 *Clasificados temporalmente Herramienta para el experto (Fuente propia)*

De esta sección destacan dos funcionalidades principales. La primera es la posibilidad de reclasificar el ejecutable en caso de algún error y la segunda es terminar el proceso de clasificación de un ejecutable de manera definitiva, lo cual hace que este registro desaparezca tanto de esta pestaña como del selector de la

parte inicial y su software asociado sólo podrá ser modificado accediendo directamente al repositorio. Al terminar el proceso de clasificación de un ejecutable se crea un nuevo registro en el repositorio con todos los datos y además de esto se añaden dos campos que son la fecha en que se creó el registro y el autor que hizo este cambio para que en casos de que se detecten registros erróneos, se pueda hacer fácilmente una traza y determinar acciones que ayuden a que el repositorio principal permanezca lo más correcto y completo posible.

6.3 IMPLEMENTACIÓN DE LA HERRAMIENTA PARA EL EXPERTO

Esta herramienta fue desarrollada en .Net MVC5 por lo cual dentro del proyecto desde el inicio se crean carpetas que separan las vistas, el modelo y los controladores que se usan. En la **Figura 33** se muestran las vistas en el entorno de VS.NET.

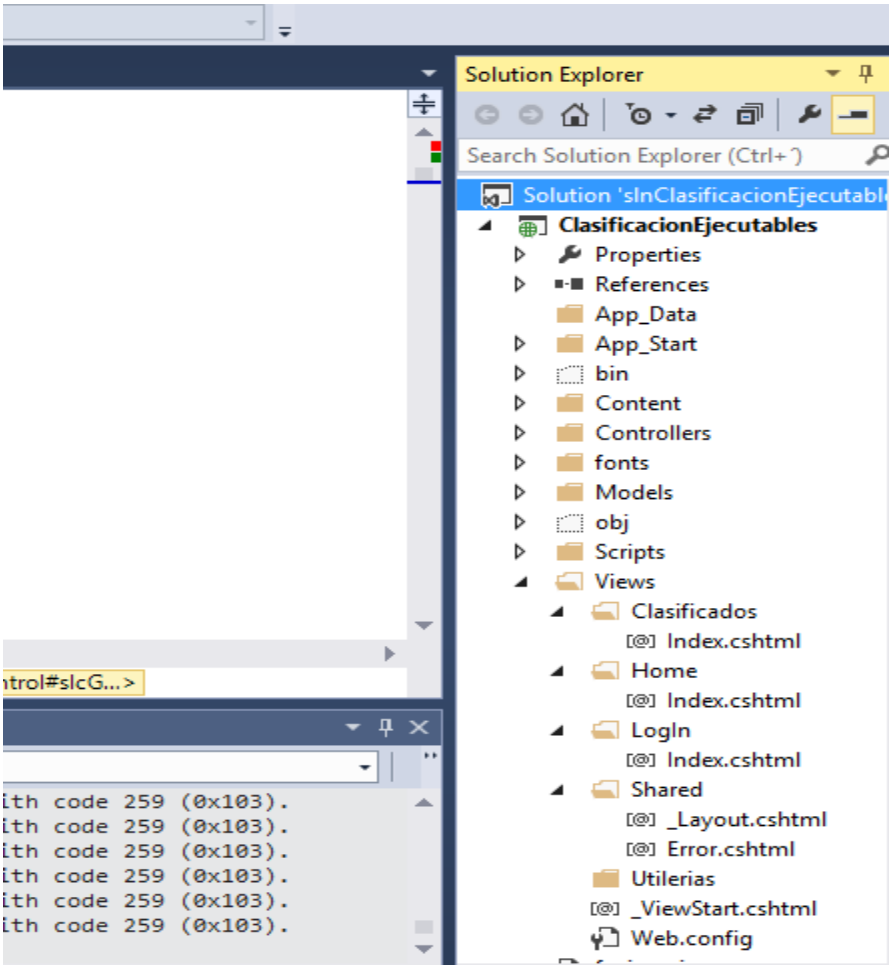

Figura 33 *Vistas en el proyecto Herramienta para el experto (Fuente propia)*

La vista de Clasificados corresponde a la vista que permite editar los registros del repositorio. La vista Home es con la que el experto interactúa de forma más constante pues es la que muestra las sugerencias, permite hacer la clasificación manual y muestra el listado de ejecutables clasificados temporalmente. LogIn

como su nombre lo indica es la vista a través de la cual un usuario se autentica e inicia sesión. En las vistas dentro de la carpeta Shared se encuentran las plantillas usadas para manejar los errores y la plantilla general usada en la aplicación.

A continuación, la **Figura 34** muestra los controladores implementados en VS.NET.

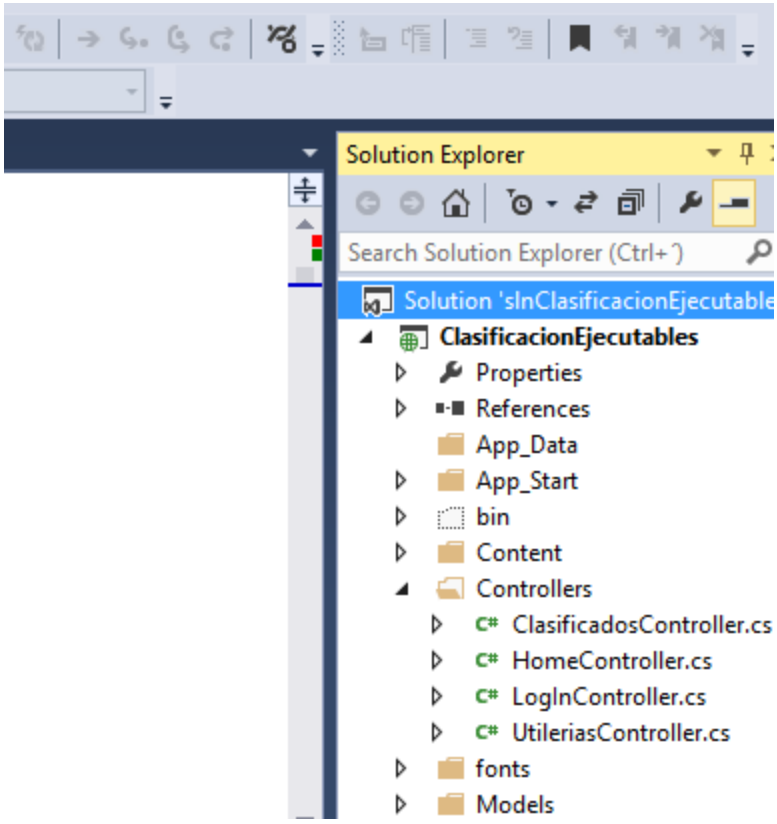

Figura 34 *Controladores en el proyecto Herramienta para el experto (Fuente propia)*

Hay un controlador que contiene funcionalidades asociadas a su correspondiente vista, de esta forma, ClasificadosController tiene las funciones relacionadas con los datos del repositorio principal, HomeController las funciones relacionadas a las sugerencias y ejecutables con los que se interactúa en la mayoría de la aplicación y LogInController las funciones asociadas a la autenticación de usuarios.

Para la construcción del modelo se hizo uso del Entity Framework [53] con el estilo “DataBase First”. Esto genera un modelo visible dentro del proyecto de las tablas existentes en la base de datos usada, además de generar clases en C# que mapean las tablas para que se puedan recibir datos en objetos de estas clases y manipularlos de tal forma que para el desarrollador sea más sencilla su gestión sin tener que involucrarse de forma muy profunda con la conexión e interacción con la base de datos. La **Figura 35** muestra el resultado del uso de Entity Framework en el proyecto de VS.NET.

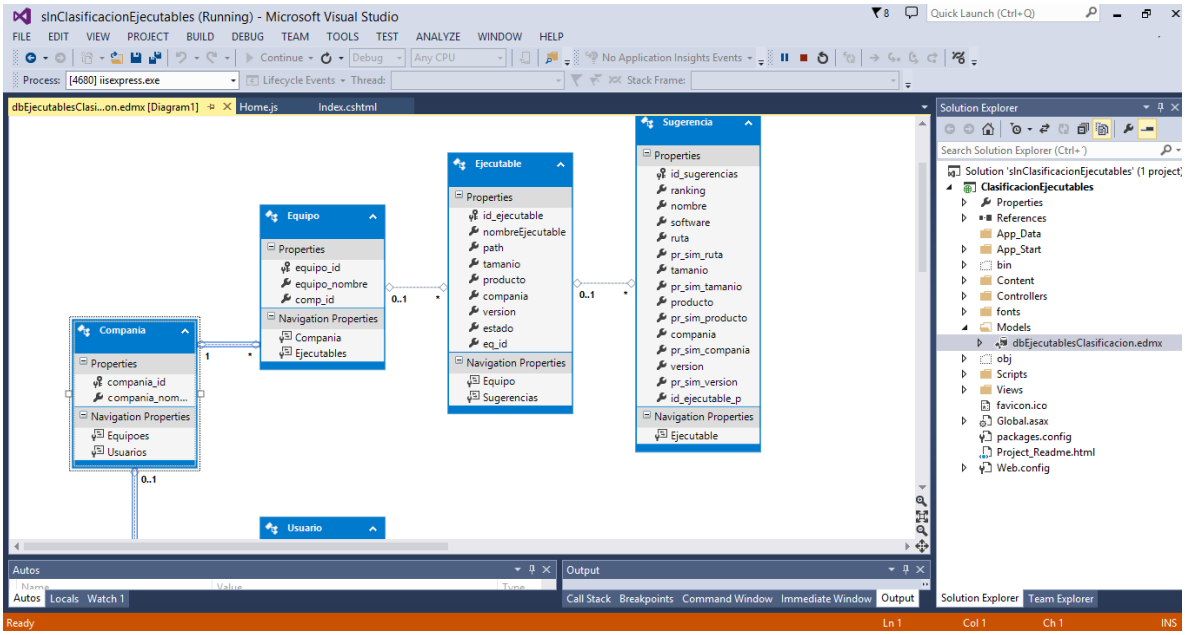

Figura 35 *Modelo en el proyecto Herramienta para el experto (Fuente propia)*

El resto de las carpetas no son de mayor relevancia pues son carpetas estándar en todo proyecto de .Net MVC5 entre las que se incluyen algunas para almacenar scripts útiles en cualquier aplicación como los de JQuery y Bootstrap, archivos para el enrutamiento de la aplicación y otra para la gestión de contenido multimedia como imágenes e íconos en caso de tenerlos.

Cuando se planteó el anteproyecto se definió una pregunta de investigación relacionada con qué estrategias se podían usar para solventar la falta de información de un ejecutable y cuál era la mejor manera de apoyar el trabajo de un experto en caso de ser necesaria su intervención en el proceso de asociación. A través de la “Herramienta para el Experto” que se implementó, se le da a este un listado de sugerencias de posibles registros en el Repositorio principal que son similares al que se quiere clasificar, de esa forma el experto no tiene que hacer la asociación ejecutable-software sin ningún dato de soporte. La Herramienta para el Experto también apoya su intervención en el proceso de asociación a través de la agrupación de las sugerencias de acuerdo con cada una de las columnas o características de los ejecutables, dando además la posibilidad de ocultar registros de sugerencias y comparar los detalles del ejecutable a clasificar y alguna de las sugerencias. Por último, le da la posibilidad de retractarse (deshacer una decisión) en alguna de las clasificaciones hechas para que pueda tomar decisiones con mayor libertad, pero teniendo siempre en mente que se debe buscar la exactitud, integridad y completitud del repositorio principal. Con lo presentado en este capitulo se cumple con el tercer objetivo especifico previamente definido en el anteproyecto.

CAPÍTULO 7

7 DESCRIPCIÓN DE LOS OTROS COMPONENTES DEL SRI

El Clasificador principal, el Recolector de información y el Analizador de directorios fueron desarrollados en C++ 11 usando Visual Studio 2017. Esto se hizo así debido a que Aranda Software trabaja sus componentes de tal forma que sean funcionales en sistemas operativos Windows, Linux y MacOS y C++ es un lenguaje que funciona bien en cualquiera de estos sistemas operativos. La decisión del uso de C++ 11 en específico obedece al hecho de que es la versión de C++ más moderno y que tiene mayor acogida y compatibilidad entre los sistemas actualmente, y el uso de una versión moderna de este lenguaje permite aprovechar las diferentes mejoras y optimizaciones que ha recibido el mismo.

Como se decidió trabajar con C++ 11 se usó Visual Studio 2017 porque es la versión que mejor soporta esta versión del lenguaje además del hecho de que Aranda ya cuenta con licencias para esta versión del IDE. Aunque en versiones anteriores de Visual Studio se puede llegar a trabajar con C++ 11 haciendo uso de un compilador diferente al que viene configurado por defecto, su soporte es bastante pobre y termina por entorpecer el desarrollo ya que no es claro qué se puede y qué no se puede usar del estándar de C++ 11. A continuación se describe la investigación, diseño y/o desarrollo de los otros componentes del SRI.

7.1 RECOLECTOR DE INFORMACIÓN

Es el componente encargado de extraer de un computador todos los ejecutables que se deseen clasificar y su información asociada como la versión o el tamaño de cada archivo, información que se usa para que el clasificador pueda determinar a qué software pertenece. También revisa qué archivos ejecutables se han eliminado. Es el primer componente en ejecutarse y tiene en cuenta las siguientes restricciones:

- Los ejecutables propios del sistema operativo no son de interés para la clasificación.
- El proceso de instalación de un software cualquiera no está amarrado a que deba ser realizado en las ubicaciones por defecto, normalmente “Archivos de programas”.
- Hay ejecutables que pertenecen a lo que se conoce como “programas portables”.
- Muchos ejecutables no cuentan con todos los datos asociados, por ejemplo, algunos ejecutables no tienen información de la compañía que los desarrolló (Posible riesgo de seguridad).

Debido a las restricciones con las que cuenta, el recolector de información tiene una sección en la que se agregan directorios que serán omitidos en la búsqueda de ejecutables, por ejemplo C:\Windows\System32.

La **Figura 36** muestra los elementos que hacen parte del Recolector de información.

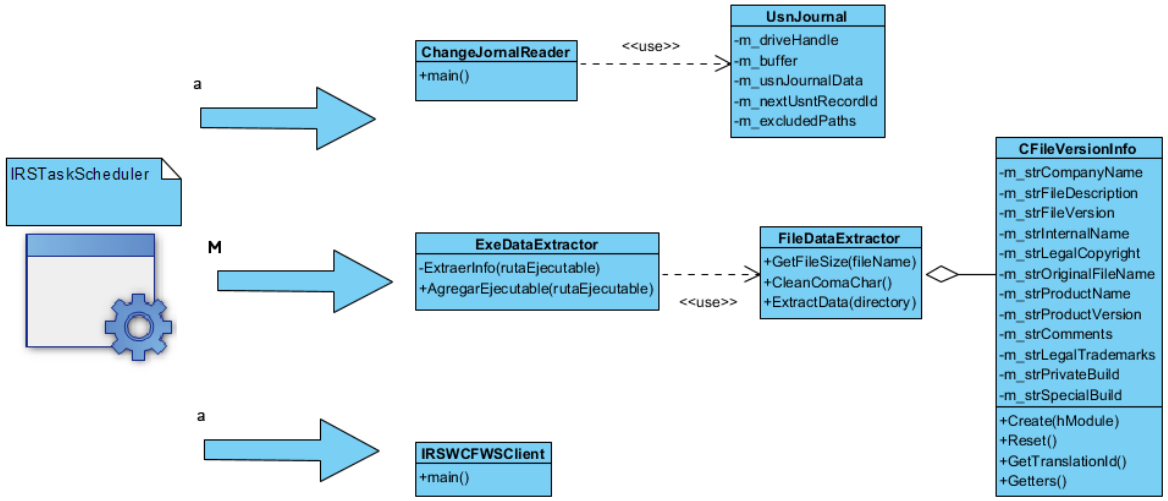

Figura 36 *Componentes del Recolector de Información (Fuente propia)*

El IRSTaskScheduler es un programa que se encarga de invocar a los demás componentes del Recolector de Información en un periodo determinado. Este componente arranca por invocar al Change Journal Reader que es el que hace la revisión de los ejecutables y en especial de la ruta en la que están ubicados. Luego invoca al ExeDataExtractor para que obtenga todos los datos relacionados a cada uno de los ejecutables hallados por el Change Journal Reader. Por último, se inicia el IRSWCFWSClient que es el cliente que le solicita al IRSWCFWebService que clasifique el listado de ejecutables que se requieran asociar.

Existen dos escenarios típicos de procesos de recolección de datos. El primer escenario es aquel en el que a un computador cliente nunca se le ha realizado el proceso de clasificación, por lo cual es necesario enviar para clasificación todos los ejecutables que se encuentren en el mismo omitiendo por supuesto los que se encuentren en directorios restringidos (m_excludedPaths en el UsnJournal). El segundo escenario es el que se produce luego de haber hecho el proceso de clasificación al menos una vez. En este escenario se ha creado un archivo con los ejecutables ya clasificados y cuando se cumple el periodo de ejecución del IRSTaskScheduler sólo se solicita clasificación de los nuevos ejecutables encontrados.

El pseudocódigo del recolector de información se resume en la **Figura 37**. El algoritmo arranca desde un directorio inicial obteniendo un elemento que puede ser un archivo para clasificar. En la línea 2 del pseudocódigo se inicia un ciclo

mientras que permite revisar todos los elementos hallados dentro de ese directorio. Luego de eso en el condicional de la línea 3 se hace una comprobación de si el elemento actual en revisión es de tipo ejecutable (.exe) y además de eso si la ruta que lo contiene no hace parte del conjunto de rutas restringidas, que no se deben revisar, esto se refiere a lo que realiza el Change Journal Reader. En caso positivo se realiza la extracción de la información del ejecutable y se envía como parámetro al método Agregar() que identifica también si el ejecutable es nuevo y necesita clasificarse o ya existe en la base de datos local y debe omitirse; esto corresponde a las acciones hechas por el ExeDataExtractor. Luego, en la línea 7, se avanza al siguiente elemento en el listado y se repite el proceso hasta que no quede ninguno por analizar y se devuelve el listado de ejecutables que se deben clasificar. Por último, en la línea 10, se solicita la clasificación de los ejecutables que lo necesiten, acción correspondiente al IRSWCFWSClient.

Entrada: Lista de rutas a omitir
directorio inicial

Salida: Conjunto de ejecutables con su información asociada

```
1.  ObtenerElementoActual ()
2.  Mientras elementoActual != Nulo hacer
3.      Si elementoActual.tipo == Ejecutable && !ListaRestringidos.Contiene
            (elementoActual.ruta) entonces
4.          ExtraerInformaciónAsociada (elementoActual)
5.          ListadoAClasificar.Agregar (elementoActual)
6.      Fin si
7.      ObtenerElementoActual ()
8.  Fin Mientras
9.  Retornar ListadoAClasificar
10. SolicitarClasificacion(ListadoAClasificar)
```

Figura 37 *Pseudocódigo del Recolector de información*

7.1.1 IRSTaskScheduler

Este componente es el orquestador del proceso de recolección de datos en un computador cliente en específico. Sus funciones son invocar a los demás componentes cada cierto tiempo. El periodo en que se ejecuta la lógica del IRSTaskScheduler viene determinado por un archivo de configuración en el que se puede ajustar cada cuántas horas, días o semanas se hace la recolección de datos y opcionalmente la clasificación de nuevos ejecutables encontrados.

Debido a su simple lógica este componente sólo realiza las siguientes acciones:

- Crear una nueva tarea para registrar.
- Adicionar la descripción y los disparadores para la nueva tarea que en este caso son la periodicidad en que se ejecutan las acciones de la tarea.

- Ajustar el nivel de ejecución. En este caso Administrador, para poder tener acceso a todos los recursos, sobre todo a los que usa el Change Journal Reader que son de muy bajo nivel.
- Adicionar las acciones de la tarea que sería conectar los ejecutables del Change Journal Reader, el ExeDataExtractor y el IRSWCFWSClient.
- Registrar la tarea en el sistema operativo.

Los valores creados por este componente se pueden visualizar en el sistema operativo accediendo al ***Panel de control -> Sistema y seguridad -> Herramientas Administrativas -> Programador de tareas*** en la sección ***"Biblioteca del Programador de tareas"***. Por ejemplo, en la **Figura 38** se muestra una tarea del IRSTaskScheduler a ejecutar a las 10:51 pm. del 7 de diciembre de 2018.

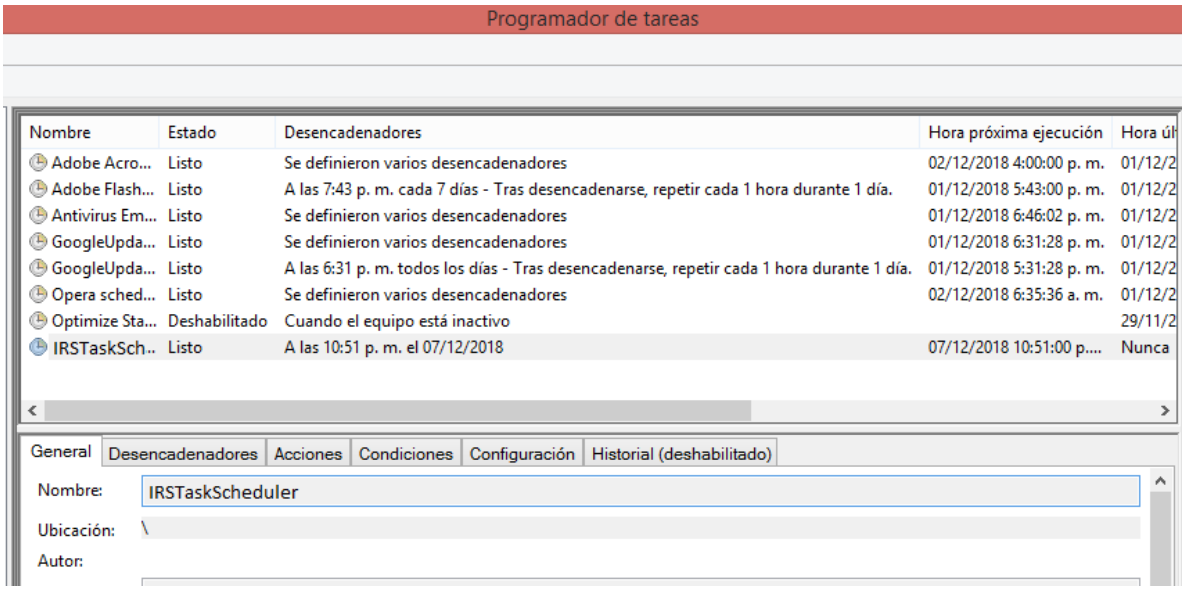

Figura 38 *Ejemplo de tarea programada en el Programador de Tareas (Fuente propia)*

7.1.2 Change Journal Reader

Este programa contiene la lógica que abre y lee el Change Journal y a partir de esto genera un archivo que contiene el path completo de aquellos ejecutables para su posterior clasificación.

El programa incluye una clase llamada UsnJournal que tiene las siguientes funciones:

UsnJournal (const wchar_t* driverName): El constructor parametrizado de la clase. Tiene como principal objetivo invocar la función **CreateFileW** que permite manipular el contenido de una partición, **driverName**, en un disco duro de formato NTFS. Adicional a esto se llena un vector que contiene el listado de los directorios a excluir del proceso de clasificación. Entre los directorios excluidos se encuentra "**C:\Windows**" que como es conocido contiene los principales componentes de un

sistema operativo de Windows. También se excluye de la clasificación cualquier ejecutable que se encuentre en el directorio "**C:\ProgramData**" ya que la información contenida en este directorio es temporal y es sólo de soporte para diferentes programas. Por ahora se cuenta con 13 directorios excluidos, pero como el vector que los contiene es dinámico, se puede agregar a futuro cualquier otro que se encuentre relevante.

RefreshJournal (): Es un método que prepara el Change Journal correspondiente a una partición en específico para su posterior lectura de datos. En su funcionalidad se destaca el llamado a la función **DeviceIoControl** con el código **FSCTL_QUERY_USN_JOURNAL** perteneciente a la librería **Windows.h** [54] que revisa la existencia del Change Journal y su correcta apertura.

FILE_ID_DESCRIPTOR GetFileIdDescriptor (const DWORDLONG fileId): este método permite obtener un descriptor de id de archivo [55] necesario para poder obtener el path completo de un ejecutable. El uso de este método se hizo necesario debido a que el Change Journal no tiene en los registros un campo que permita conseguir de forma directa el path. Se cuenta con un campo llamado **FileReferenceNumber** que es un código que identifica a todo archivo o carpeta en la partición y un campo llamado **ParentFileReferenceNumber [56]** que en teoría permitiría la reconstrucción a través de una función recursiva del path, sin embargo hacerlo con estos campos es bastante complejo y requiere recorrer el Change Journal más de una vez sólo para obtener este dato, lo que es muy ineficiente. En este caso se optó por usar el FileReferenceNumber para obtener el descriptor de id del archivo en cuestión y ahorrarse cualquier recorrido adicional del Change Journal.

FILE_ID_DESCRIPTOR GetFileIdDescriptor (const FILE_ID_128& fileId): la existencia de esta sobrecarga de métodos se debe a que como el formato de archivos NTFS y las librerías de Windows llevan bastantes años en el mercado y con el pasar del tiempo han cambiado. Existen tres clases distintas dentro de Windows.h que pueden almacenar registros del Change Journal y que su uso depende de la versión del sistema operativo en el que se ejecute el programa [57]. En este proyecto se le da soporte a las dos versiones más usadas debido a que la versión no soportada es muy vieja y muy poco usada.

string GetReason (DWORD reason): Es un método que permite hacer legible la razón por la cual un registro de un archivo fue creado en el Change Journal. El motivo de hacer esto es que el sistema hace uso de una serie de códigos para marcar los cambios hechos en un archivo o carpeta y que no son fácilmente legibles pues son códigos hexadecimales [58]. Entre las razones que motivan la inclusión de un registro de un archivo en el Change Journal están la creación de un archivo, la modificación de su contenido, su borrado o el cambio de su nombre.

string GetPath (FILE_ID_DESCRIPTOR file_id): haciendo uso de los resultados obtenidos con los métodos **GetFileIdDescriptor** entrega el path correspondiente a un archivo ejecutable. En su funcionalidad se hace un llamado a la función del

sistema **OpenFileById [59]** que recibe como parámetro el descriptor de id de un archivo y luego de esto se obtiene el path teniendo en cuenta el nivel de seguridad del archivo en cuestión.

bool PathIsExcluded (string path): Entrega un valor booleano que determina si el path perteneciente a un ejecutable está en la lista de directorios que no se deben revisar.

TCHAR* GetTimestamp (const LARGE_INTEGER * timestamp): transforma un timestamp entregado por las funciones de Windows.h a un formato legible ya que el entregado es un número que a simple vista no se puede entender.

void ProcessEntries (): este es el principal método dentro de la clase ya que es el que propiamente puede revisar todos los registros del Change Journal. Inicialmente invoca la función **DeviceIoControl** con el código **FSCTL_ENUM_USN_DATA [60]** que permite abrir el Change Journal de una partición para enumerar los registros que se encuentran en él. Después de esto se entra a un ciclo que acumula un buffer de información con un registro y sus datos asociados y que en caso de no estar dentro de los path excluidos y ser nuevo, se incluye en el proceso de clasificación. Una vez revisado un registro se avanza en el Change Journal una cantidad de bytes correspondiente a cada registro y se avanza en el ciclo hasta que no queden registros. Cabe destacar también que dentro de la lógica se analiza que el registro actual sea un ejecutable, ya que el Change Journal contiene muchísima información, no sólo de ejecutables, por lo cual es necesario aplicar un filtro.

7.1.3 ExeDataExtractor

Este componente, que toma como entrada un archivo de salida originado por el Revisor de Change Journal, se encarga de obtener los datos (campos o características previamente definidas al inicio del proyecto) asociados a un archivo ejecutable que son usados dentro del proceso de clasificación. Los campos son: nombre del ejecutable, compañía, path, tamaño y versión. También se hace la extracción de otros campos como la descripción del ejecutable que no son tenidos en cuenta por el clasificador principal pero que pueden ayudar al experto a tener una idea de a qué software pertenece el ejecutable en cuestión.

Dentro de este componente se encuentran dos clases principales:

- **CFileVersionInfo**: esta clase es de fuente externa, proveída por Sven Wiegand y ToolsCenter y tiene como finalidad proveer los métodos necesarios para obtener cada uno de los campos asociados a un ejecutable. Cuenta con un método getter correspondiente a cada uno de los campos que puede extraer, sin embargo, no todos son usados.
- **FileDataExtractor**: hace uso de la clase CFileVersionInfo y organiza la información obtenida para que se ajuste a lo que usan los demás componentes del SRI. En sus métodos destacan **long GetFileSize (std::string filename)**

que permite obtener el tamaño del ejecutable en el formato deseado, string **CleanComaChar (std::string inString)** que evita que en los campos como la versión del ejecutable aparezca el carácter ‘,’ que puede llegar a confundir al clasificador, es decir, funciona a manera de filtro y **void ExtractData (std::string directory)** que es el método que tiene la lógica principal para obtener los datos; como los parámetros que recibe el path de cada ejecutable del que se quieran extraer datos.

7.1.4 IRSWCFWSClient

Este es el cliente para los servicios prestados por el web IRSWCFWebService. Su lógica es la siguiente:

1. Leer los resultados del ExeDataExtractor para crear un listado de ejecutables a enviar para clasificación.
2. Enviar cada paquete a clasificar al web service de manera asíncrona.
3. Solicitar la clasificación del paquete enviado.
4. Solicitar de manera asíncrona los resultados de la clasificación al servidor.

Las solicitudes asíncronas se hacen con el fin de evitar que los procesos del Recolector de información obstaculicen las diferentes tareas que esté realizando el usuario del computador cliente en el momento en que se requiere de una nueva clasificación.

7.2 IRSWCFWebService

Es el componente que recibe las solicitudes de clasificación del cliente y gestiona su resultado. Dentro de este se define una interfaz, IWSClasificacion, que define las siguientes funciones para ser solicitadas por un cliente:

- **enviarPaqueteAClasificar**: Tiene como propósito recibir de parte del cliente un listado de ejecutables que necesitan ser clasificados.
- **clasificarPaqueteEnviado**: invoca al Clasificador Principal para delegarle la tarea de asociar los ejecutables anteriormente enviados por un cliente con el ejecutable que corresponda de forma automática y de crear el paquete para el experto en caso de necesitarse.
- **getEjecutablesClasificados**: Entrega los resultados de la clasificación al cliente.

Gracias a las ventajas de usar Windows Communication Foundation (WCF) y Visual Studio, cada una de las funciones cuenta con una versión asíncrona que permite que el servidor las ejecuta cuando tenga disponibilidad y no se bloquee para esperar los resultados de alguna. La versión asíncrona de una función en la interfaz se identifica por el sufijo Async, de esta forma la interfaz incluye también las funciones enviarPaqueteAClasificarAsync, clasificarPaqueteEnviadoAsync y getEjecutablesClasificadosAsync.

En el servicio web hay un DataContract que sirve para definir una clase que sería serializable y deserializable a través de la red y que permite la gestión de los ejecutables en el proceso. En la **Figura 39** se puede ver este DataContract.

```
[DataContract]
public class Ejecutable
{
    [DataMember(Name = "nombre")]
    public string nombre { get; set; }

    [DataMember(Name = "path")]
    public string path { get; set; }

    [DataMember(Name = "tamanio")]
    public int tamanio { get; set; }

    [DataMember(Name = "producto")]
    public string producto { get; set; }

    [DataMember(Name = "compania")]
    public string compania { get; set; }

    [DataMember(Name = "version")]
    public string version { get; set; }

    [DataMember(Name = "descripcion")]
    public string descripcion { get; set; }

    [DataMember(Name = "software")]
    public string software { get; set; }
}
```

Figura 39 *DataContract Ejecutable en el Web Service (Fuente propia)*

La implementación de la interfaz se encuentra en el archivo **WSEjecutables.svc.cs**. En este archivo además de implementar cada una de las funciones definidas en la interfaz, se hace uso de una función llamada **IsFileLocked (FileInfo file)** que permite determinar si un archivo se encuentra bloqueado actualmente, por ejemplo, por un proceso de escritura, ya que como las funciones usadas por el cliente son las versiones asíncronas, es posible que una función que deba ejecutarse luego de haberse terminado de escribir un archivo, quiera hacerlo cuando no debe y así se pueda hacer una espera mientras esto termina. El WSDL en un Web Service desarrollado usando WCF, no es visible por lo cual no se muestra en este documento o en un anexo, sin embargo, es importante anotar que cuando se construye el servicio se generan opciones para poder consumirlo bien sea como JSON (Rest) o como XML (SOAP) lo que le da una gran versatilidad al mismo.

7.3 INSPECTOR DEL REGISTRO DEL SISTEMA Y ANÁLISIS DE DIRECTORIOS

Este es un componente de apoyo pensado para aquellas situaciones en las que la clasificación de una instancia por parte del Clasificador principal, no se puede realizar de forma automática. Debía ser el primero en ser llamado luego del Clasificador por parte del IRSWCFWebService y su objetivo era revisar la información del registro de Windows del equipo emisor además de hacer un análisis de directorios para buscar indicios que permitieran agregar información sobre la instancia a clasificar. En caso de que encuentre información adicional que permita clasificar la instancia, le notifica al IRSWCFWebService para que continúe con la siguiente clasificación, en caso contrario le notifica para que este adicione la instancia por clasificar al paquete para el experto junto con sus sugerencias respectivas y luego continúe con la siguiente instancia por clasificar. Este componente se ubica en el lado del cliente.

Al inicio del proyecto se hizo una investigación del Registro del sistema de Windows con el fin de determinar qué información de la existente en estos registros podría aportar al proceso de clasificación. Se buscó en la documentación oficial de Microsoft y en otras fuentes externas información al respecto, pero lastimosamente la información que se encontró fue muy poca. Microsoft por su parte ofrece una escasa documentación que simplemente describe los principales elementos de estos registros, sin entrar en detalle y que además de esto se encuentra desactualizada [23]. Por otro lado, en fuentes hechas por usuarios se encontraba mejor información que la que ofrece Microsoft, aunque aún muy poca.

Como la documentación no cumplió los requerimientos, se procedió a hacer una inspección manual de todos los datos que hay en el registro. Algunos de estos datos eran legibles y otros no. A través de este proceso se pudo descubrir que este registro no sólo no se encuentra muy bien ordenado, sino que también es bastante incompleto. Esto sucede debido a que Microsoft no tiene políticas estrictas para el soporte de cada nueva aplicación en sus sistemas operativos como por ejemplo si lo hace Apple; literalmente cada fabricante de software es libre de usar o no este registro y además de eso decide cómo usarlo. Por otro lado, está la existencia de los conocidos “programas portables o directamente ejecutables” que por lo general no se registran y no guardan ningún tipo de información en este componente del sistema operativo.

Después de haber realizado esta investigación se encontró que, aunque posiblemente pudiera encontrarse algún dato que aportara para identificar un ejecutable con su software relacionado, las posibilidades de que esto sucediera eran muy bajas y como cada fabricante decide qué poner dentro del registro y cómo ponerlo, es imposible realizar un programa que pueda extraer nueva información valiosa en forma genérica o transversal para cada ejecutable. Finalmente se desarrolló solamente un Analizador de directorios en C++ como un pequeño algoritmo de apoyo para el Clasificador Principal, pero este realmente no tiene relevancia en el SRI.

7.4 TRABAJO CON LA METODOLOGÍA DE DESARROLLO SCRUM

La metodología Scrum se aplicó en el desarrollo del proyecto de una manera simplificada. Para empezar, están los roles:

- Producto Owner: Álvaro Carmona y Carlos Cobos
- Scrum Master: Hugo Castellanos
- Developer: Jalbersson Plazas

Las actividades del proyecto quedaron enmarcadas dentro de cada Sprint que se desarrolla dentro de la empresa Aranda Software de tal forma que el developer no sólo realizaba tareas del proyecto sino también otras relacionadas con los productos de la compañía.

El control de la metodología se hizo a través de la herramienta Team Foundation Server (TFS) de Microsoft. Dentro de esta herramienta se pueden gestionar los miembros de un equipo, las tareas que va a realizar en cada Sprint, su duración, el reporte de avances en cada tarea y entrega algunas gráficas relacionadas para hacer evaluación del proceso.

En cuanto a documentación realmente no hubo mucha relacionada con la metodología ya que todo se manejó de manera simple y concisa en el TFS. Sin embargo, sí se hizo un seguimiento del proyecto con las reuniones diarias con el Scrum Master y reuniones semanales con alguno de los Product Owner para mostrar los avances, hacer los análisis requeridos y escuchar las opiniones o sugerencias que se tuviera frente a lo presentado en la reunión.

En la **Figura 40** se aprecia la forma en cómo muestra TFS las tareas en un Sprint.

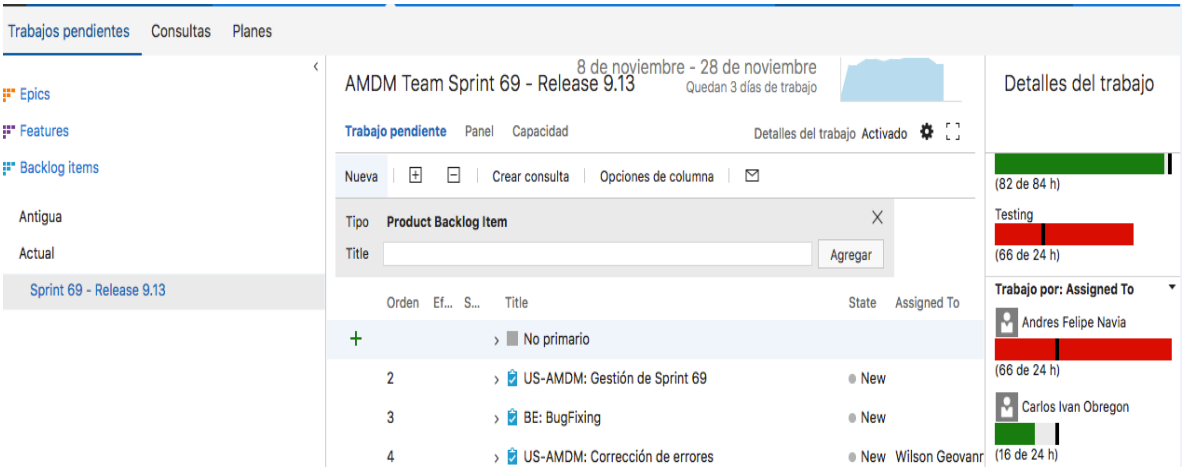

Figura 40 *Ejemplo tareas en el TFS (Fuente propia)*

Ya viendo en detalles las tareas asignadas a un integrante del equipo en particular se puede ver una vista como la de la **Figura 41**.

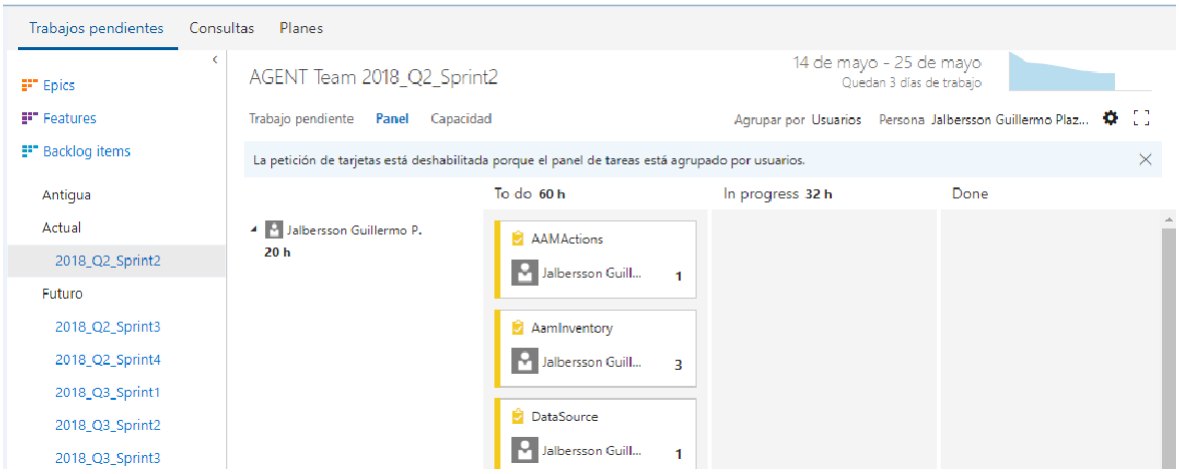

Figura 41 *Detalles de tareas integrante en TFS (Fuente propia)*

En cuanto a las actividades de Scrum, se realizaron todas, sin embargo, el trabajo de un Sprint no aplicaba solamente para el trabajo con el SRI. Cada Sprint dentro de Aranda Software tuvo una duración de dos semanas aproximadamente. Al inicio cada integrante hace la planeación de sus tareas dentro del Sprint ubicándolas con nombre y duración en el TFS. El Scrum Master revisaba lo propuesto por cada integrante y lo aprobaba o modificaba de acuerdo a las necesidades de la empresa. Diariamente se realizaron Daily Meetings para analizar el estado de avance de cada una de las tareas, hacer reportes de problemas encontrados y proponer soluciones entre todos los integrantes del equipo. Al final de cada Sprint se realizó una retrospectiva para ver en qué se podía mejorar o dónde hubo cosas que destacar y así sucesivamente con cada Sprint.

CAPÍTULO 8

8 EXPERIMENTACIÓN

8.1 Aplicación del test de usabilidad y funcionalidad

Con el propósito de poner a prueba el SRI, en especial la Herramienta para el experto, se aplicó un test de usabilidad y funcionalidad con nueve voluntarios entre los que se encuentran Ingenieros de Sistemas, estudiantes de la carrera y desarrolladores. La prueba fue tomada de [61] y busca evaluar las siguientes características:

- **Identidad**: se refiere a qué tanto se puede identificar una aplicación como asociada a una compañía como por ejemplo que estén presentes los logos, colores e imágenes distintivas.
- **Diseño**: tiene que ver con las formas y el look general de la aplicación en cada una de sus interfaces. Se evalúan aspectos como la consistencia en la apariencia y ubicación de los elementos.
- **Accesibilidad**: va encaminada a evaluar si una aplicación es apta no sólo para personas saludables sino también, por ejemplo, para personas con problemas de visión u otras dificultades físicas que requieran un trato especial en la aplicación.
- **Tiempo de acceso**: qué tan rápido se accede a las diferentes secciones y qué tan rápidos se hacen los procesos que se ejecutan.
- **Navegación**: busca evaluar la fluidez con la que un usuario se mueve por los diferentes componentes de la aplicación.
- **Operación**: básicamente se evalúa que todos los componentes funcionen correctamente.
- **Utilidad**: herramientas adicionales que le ayuden al usuario a tener una experiencia más agradable al usar la aplicación como por ejemplo buscadores o secciones de ayuda.

El proceso inició con una breve explicación acerca de lo que era la herramienta y qué buscaba solucionar. Posteriormente se siguió un guion que le permitió a cada uno de los usuarios visitar cada una de las secciones y funcionalidades presentes en la Herramienta para el experto. De igual forma si alguno lo deseaba podía ser libre de realizar diferentes acciones que pusieran a prueba temas como seguridad o funcionalidad en algún componente bajo situaciones extrañas o poco frecuentes sirviendo de cierto modo como testers de la aplicación.

Una vez terminado el uso de la aplicación se aplicó la encuesta disponible en [62] a través de Formularios de Google. Se usó la siguiente escala para la evaluación:

1 - Muy malo o no funciona: lo evaluado no realiza la actividad o no muestra el contenido que ofrece.

2 - Funciona, pero no sirve: lo evaluado desarrolla la actividad o muestra un contenido, pero en general, lo recibido por el evaluador no aporta a la experiencia general del sitio.
3 - Funciona, pero debe mejorar: lo evaluado desarrolla la actividad o muestra un contenido relativamente útil, pero podría ser mejor. Al respecto el evaluador debe aportar información para hacer mejoras al sitio.
4 - Cumple: lo evaluado desarrolla la actividad o muestra un contenido útil. El evaluador debe aportar información para hacer mejoras al sitio.
5 - Es lo que el usuario busca: lo evaluado desarrolla la actividad o muestra un contenido útil que cumple o excede la expectativa del usuario.

Y se aplicó sobre las siguientes preguntas:

Identidad
1. El sitio web entrega información corporativa de la organización
2. Entrega información para hacer consultas web o no-web (Ej.: Números. de teléfono)

Diseño
3. ¿El diseño es consistente en todas las pantallas del sitio?
4. ¿Las páginas tienen scroll adecuado?

Accesibilidad
5. ¿Tamaño de letras es adecuado? ¿Se pueden agrandar?

Tiempo de Acceso
6. ¿El tiempo en que se ingresa a la plataforma y se ejecutan las acciones es adecuado?

Navegación
7. ¿Los enlaces son claramente visibles?
8. ¿El menú del sitio es consistente en todo el sitio?
9. ¿Todos los vínculos funcionan?

Operación
10. ¿Presenta fallas de sistema?
11. ¿Se administra el error 404?
12. ¿Existe seguridad adecuada para el tipo de sitio?

Utilidad
13. ¿Tiene Buscador? ¿Funciona correctamente?
14. ¿Permite resolver las dudas básicas que el usuario tenga sobre sus contenidos?
15. ¿El sitio genera una experiencia que a uno llevaría a recomendarlo a otros?

Los resultados obtenidos de la aplicación del test son los siguientes:

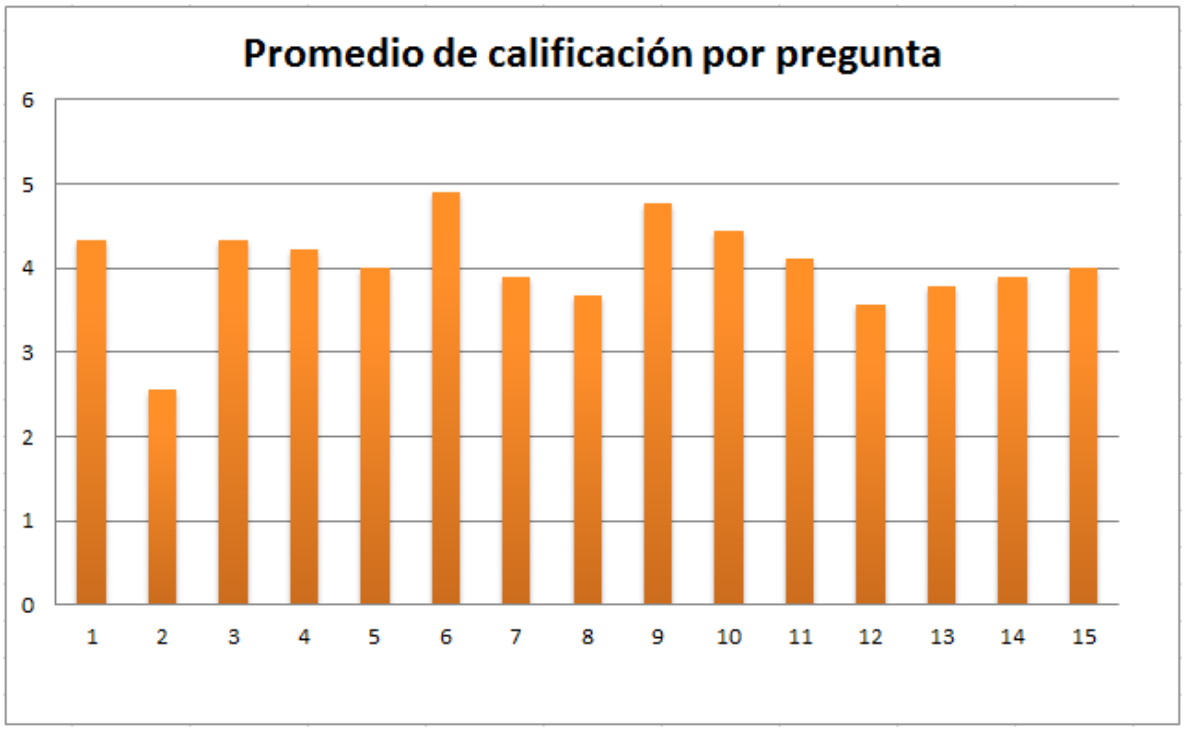

Figura 42 *Promedio de calificación por pregunta en el test de usabilidad y funcionalidad (Fuente propia)*

Cada una de las barras corresponde a una de las 15 preguntas hechas en la encuesta en el mismo orden en el que aparecen. De acuerdo a estos se puede ver una generalizada aceptación hacia la herramienta y su funcionamiento a excepción de la pregunta relacionada con la información proporcionada para consultas más allá de la parte web.

Al final de la encuesta se dejó una sección para comentarios y sugerencias. Dentro de estas destacan unas referentes a agregar un texto descriptivo en cada sección o tab explicando su función, otros referentes a los tipos de letra usados y al uso de un scroll horizontal en lugar de un botón para visualizar todos los campos de las sugerencias. También se habló de los colores usados en algunos componentes y algunos errores encontrados durante las pruebas. Los Test aplicados junto con el guion se encuentran en el **anexo 5** de este documento.

Con el desarrollo de los diferentes componentes con el apoyo del IDE Visual Studio y la ejecución de este test de usabilidad y funcionalidad se cumple con el último objetivo específico propuesto.

CAPÍTULO 9

9 CONCLUSIONES Y TRABAJOS FUTUROS

9.1 CONCLUSIONES

Al término del desarrollo del proyecto se llegó a las siguientes conclusiones:

- Se modelo una base de datos (repositorio) de ejecutables y aplicaciones que es sencillo y permite soportar efectivamente (sin normalizar datos para hacer más rápida la ejecución del algoritmo de clasificación) el proceso semi-automático de clasificación. Además, se modeló un repositorio de ejecutables y aplicaciones software en una base de datos relacional que se cargó con datos obtenidos por parte de Aranda Software S. A. S. y se obtuvieron datos complementarios a través de un proceso de Web Scraping al sitio web ProcessList.com.
- Los datos incluidos en el Repositorio Principal fueron sometidos a un riguroso proceso de asociación manual y revisión por parte de otro experto asegurando así la completitud y confiabilidad de aproximadamente 10.000 registros que sirven como base para continuar el crecimiento del mencionado repositorio.
- Se diseñó un algoritmo de clasificación, usando K-nn como base, que establece automáticamente la asociación entre un ejecutable encontrado en un computador con la aplicación software a la cual pertenece y se estableció la precisión de este usando los datos del Repositorio Principal y la información extraída de 20 computadores mostrando cómo mejora su capacidad de asociación a medida que existen más datos de ejemplo para el algoritmo.
- Como estrategias que se pudieran usar para solventar la falta de información en un ejecutable y que apoyaran el trabajo de un experto en caso de ser necesaria su intervención en el proceso de asociación se plantearon dos componentes, un Inspector de Directorios y del Registro de Windows y la Herramienta para el experto, siendo abandonada la primera debido a la poca información oficial y no oficial disponible al respecto y a su poca confiabilidad por el control exigido de parte de Microsoft.
- Se desarrolló una herramienta web de visualización por grupos, la Herramienta para el experto, que ayuda a un experto humano en la clasificación de los archivos ejecutables que el Clasificador Principal no pudo clasificar de manera automática.
- Se integraron los componentes anteriormente mencionados en el marco de un Sistema de Recuperación de Información junto con otros componentes de apoyo usando VS.NET mediante SCRUM y se aplicó un test de usabilidad y funcionalidad con nueve personas relacionadas con el desarrollo de software que permitió medir qué tan fácil de usar el SRI resulta para el usuario y qué cosas podría mejorar a futuro.
- Se propuso un Sistema de Recuperación de Información para la asociación semi-automática de archivos ejecutables con el software al cual pertenece

facilitando el inventario y gestión del licenciamiento de software en una organización.

9.2 TRABAJOS FUTUROS

Los trabajos futuros deberían encaminarse en el estudio con mayor profundidad de las medidas de distancia o similitud para determinar si se puede lograr aumentar el porcentaje de instancias correctamente clasificadas por el Clasificador Principal. Se puede explorar la posibilidad de hacer una ponderación de las características de un ejecutable que le permita al Clasificador tomar una mejor decisión en la asociación ejecutable/software.

También se pueden enfocar en el desarrollo o adaptación del SRI con software libre además de su aplicación con dispositivos de almacenamiento en formatos diferentes a NTFS y sistemas operativos diferentes a Windows (Linux, Android, IOS).

Otro punto que se debe abordar, es soportar la solicitud en tiempo real de clasificación de ejecutables de uno o más equipos, lo que requeriría tecnologías que mantengan una conexión abierta entre clientes y servidor como web sockets [63] tratando de optimizar al máximo los recursos de red y procesamiento en ambos lados (cliente y servidor).

CAPÍTULO 9

10 BIBLIOGRAFÍA

[1] S. Maleki-Dizaji, Siddiqi, J., Soltan-Zadeh, Y. et al., "Adaptive information retrieval system via modelling user behaviour," *Journal of Ambient Intelligence and Humanized Computing,* 2014.

[2] F. R. A. B. y. G. H. Tolosa, "Recuperación de información: Un área de investigación en crecimiento," 2007.

[3] W. B. Croft, "Approaches to intelligent information retrieval. Information Proccesing & Management," *Information Processing and Management: an International Journal - Artificial Intelligence and Information Retrieval,* vol. 23, pp. 249-254, 1987.

[4] R. R. Korfhage, *Information Storage and Retrieval*: Wiley.com, 1997.

[5] P. R. Christopher D. Manning, Hinrich Schütze, *An Introduction to Information Retrieval*: Cambridge UP, 2009.

[6] D. Z. M. Rami Ghorab, Alexander O'Connor, Vincent Wade, "Personalised Information Retrieval: survey and classification," 2012.

[7] M. S. Lew, "Special issue on visual information retrieval," *International Journal of Multimedia Information Retrieval,* 2016.

[8] T. Gossen, *Search Engines for Children*: Springer Vieweg, 2015.

[9] M. D. Francesco Corcoglioniti, Marco Rospocher, and Alessio Palmero Aprosio, "Knowledge Extraction for Information Retrieval," *The Semantic Web Latest Advances and New Domains,* pp. 317-333, 2016.

[10] Y.-W. H. Rahul Singh, Naureen Moon, "Multiple perspective interactive search: a paradigm for exploratory search and information retrieval on the web," *Multimedia Tools and Applications,* vol. 62, pp. 507–543, 2013.

[11] I. B.-G. S. W. Weinsberg, "Peer-to-peer information retrieval using shared-content clustering," *Knowledge and Information Systems,* vol. 39, pp. 383–408, 2014.

[12] M. K. Jiawei Han, Jian Pei, "Data Mining Concepts and Techniques," vol. 3, pp. 327-330, 2012.

[13] C. C. Aggarwal, "Data Mining The Textbook," ed, 2015, pp. 18-19.

[14] M. Bramer, "Principles of Data Mining," Second Edition ed, 2013, pp. 21-22.

[15] MathWorld. (2018). *Acyclic Digraph*. Available: http://mathworld.wolfram.com/AcyclicDigraph.html

[16] M. García, "Ponderación local evolutiva de la regla kNN," Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Sevilla, 2013.

[17] M. T. O. z. Ling Liu, in *Encyclopedia of Database Systems*, ed, 2009, pp. 1351-1354.

[18] I. Mozenda. (2018). *Mozenda*. Available: https://www.mozenda.com/software/

[19] A. Anywhere, "Automation Anywhere Platform," 2018.

[20] Scrapinghub. (2018). *Scrapy*. Available: https://scrapy.org

[21] Agenty. (2018). *Data Scraping Studio*. Available: https://www.agenty.com/download.aspx
[22] O. D. Inc. (2018). *Octoparse*. Available: https://www.octoparse.com
[23] M. Corporation. (2018). *Structure of the Registry*. Available: https://docs.microsoft.com/en-us/windows/desktop/sysinfo/structure-of-the-registry
[24] M. Corporation. (2000). *What is NTFS?* Available: https://web.archive.org/web/20080420104035/http://technet2.microsoft.com:80/windowsserver/en/library/59a9462a-cbdd-45e7-828b-12c6cd9ae4781033.mspx?mfr=true
[25] M. Corporation. (2018). *Change Journals*. Available: https://docs.microsoft.com/en-us/windows/desktop/fileio/change-journals
[26] P. Wilson, "Windows Services," in *The Definitive Guide to Windows Installer*, 1 ed: Apress, 2004, pp. 185-190.
[27] T. L. I. Project. (2005). *Daemon Definition*. Available: http://www.linfo.org/daemon.html
[28] M. Corporation. (2018). *Service Control Manager*. Available: https://docs.microsoft.com/en-us/windows/desktop/services/service-control-manager
[29] J. Newmarch, "REST," in *Network Programming with Go*, ed: Apress, 2017, pp. 221-223.
[30] T. I. Society. (2005). *Uniform Resource Identifier (URI): Generic Syntax*. Available: https://tools.ietf.org/html/rfc3986
[31] D. Crockford. (2018). *How JavaScript Works*. Available: https://www.json.org
[32] S. Suehring, "SOAP-Based Web Services," in *Beginning Web Development with Perl From Novice to Professional*, ed, 2006, pp. 137-153.
[33] W3C. (2007). *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language*. Available: https://www.w3.org/TR/wsdl/
[34] A. Bundy and L. Wallen, "Smalltalk," in *Catalogue of Artificial Intelligence Tools*, A. Bundy and L. Wallen, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 1984, pp. 123-124.
[35] S. Smith. (2018). *What is the MVC pattern?* Available: https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-2.1
[36] N. A. J. Hastings, "Asset Management Information Systems," in *Physical Asset Management*, ed: Springer International Publishing, 2015, pp. 223-235.
[37] Microsoft_Corporation. (2017). *Microsoft Software Inventary Analyzer*. Available: https://www.microsoft.com/es/sam/
[38] ManageEngine. (2017). *ServiceDesk Plus*. Available: https://www.manageengine.com/products/service-desk/?gclid=CKGxyIybi9QCFdFahgod228NLg
[39] Snow_Software. (2017). *Snow License Manager*. Available: https://www.snowsoftware.com/es/productos/snow-license-manager

[40] Flexera_Software. (2017). *FlexNet Manager Suite for Enterpirses*. Available: https://www.flexerasoftware.com/enterprise/products/software-license-management/flexnet-manager-suite-enterprises/
[41] Grokability_Inc. (2017). *Snipe-IT*. Available: https://snipeitapp.com
[42] Symantec. (2017). *Symantec ExSP Program*. Available: http://www.licenciasonline.com/co/es/cloud/productos/symantec/exsp-program
[43] L. Richardson. (2015). *Beautiful Soup Documentation*. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[44] P. S. Foundation. (2018). *Regular expression operations*. Available: https://docs.python.org/2/library/re.html
[45] N. Bourbaki, "Topological vector spaces over a valued division ring," in *Topological Vector Spaces*, ed, 1981, pp. 3-16.
[46] I. A. Juan Pablo Pérez, "Evaluación de la eficacia de la detección de duplicados utilizando Sql Server," *Revista en telecomunicaciones e informática,* vol. 3, p. 30, 2013.
[47] W. E. Yancey, "Evaluating String Comparator Performance for Record Linkage," in *Proceedings of the Fifth Australasian Conf. on Data mining and Analytics*, S. R. D. U. S. C. Bureau, Ed., ed, 2006, pp. 21-23.
[48] V. I. .Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions, and Reversals," *Soviet Physics Doklady,* vol. 10, pp. 707-710, 1966.
[49] S.-N. Ltd. (2018). *FileAlyzer*. Available: https://www.safer-networking.org/products/filealyzer/
[50] W. University. (2018). *Weka (Waikato Environment for Knowledge Analysis)*. Available: https://www.cs.waikato.ac.nz/ml/weka/
[51] C. J. Iván Amón, "Funciones de Similitud sobre Cadenas de Texto: Una Comparación Basada en la Naturaleza de los Datos," presented at the International Conference on Information Resources Management (Conf-IRM) Jamaica, 2010.
[52] Faronics. (2018). *Deep Freeze*. Available: https://www.faronics.com/es/products/deep-freeze/enterprise
[53] M. W. Douglas Laudenschlager, Genevieve Warren. (2018). *Entity Framework overview*. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview
[54] M. Corporation. (2018). *FSCTL_QUERY_USN_JOURNAL control code*. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa364583(v=vs.85).aspx
[55] D. B. H. Jesse Liberty, "Aprendiendo C++ para Linux en 21 días," ed: Pearsons Educación, 2000, p. 875.
[56] M. Corporation. (2018). *USN_RECORD_V2 structure*. Available: https://docs.microsoft.com/en-us/windows/desktop/api/winioctl/ns-winioctl-usn_record_v2
[57] M. Corporation. (2018). *USN_JOURNAL_DATA_V0 structure*. Available: https://docs.microsoft.com/en-us/windows/desktop/api/winioctl/ns-winioctl-usn_journal_data_v0

[58] M. Corporation. (2018). *Change Journal Records*. Available: https://docs.microsoft.com/en-us/windows/desktop/fileio/change-journal-records

[59] M. Corporation. (2018). *OpenFileById function*. Available: https://docs.microsoft.com/en-us/windows/desktop/api/winbase/nf-winbase-openfilebyid

[60] M. Corporation. (2018). *FSCTL_ENUM_USN_DATA control code*. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/aa364563(v=vs.85).aspx

[61] guiaweb.gob.cl. (2016). *Pauta para Evaluación Heurística de Sitios* Available: http://www.guiadigital.gob.cl/guia-v2/capitulos/05/anexos/pauta-evaluacion-heuristica.pdf

[62] J. G. Plazas. (2018). *Test de usabilidad y funcionalidad del SRI*. Available: https://goo.gl/forms/d89Xy1jaR5nvrHA23

[63] B. Joshi, "Using the Communication API and Web Sockets," in *HTML5 Programming for ASP.NET Developers*, ed Berkeley, CA: Apress, 2012, pp. 277-304.