UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA INFORMÁTICA

EXTRACCION DE INFORMACIÓN DE ANÁLISIS DE ANTIVIRUS HECHOS SOBRE ARCHIVOS INFECTADOS POR EL VIRUS WANNACRY

INFORMATION EXTRACTION ON ANTIVIRUS ANALYTICS DONE ON FILES INFECTED BY THE VIRUS WANNACRY

Realizado por
ANDRÉS SEBASTIÁN CARRERA CAMACHO

Tutorizado por
EDUARDO GUZMÁN DE LOS RISCOS

Departamento
DEPARTAMENTO DE LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA INFORMÁTICA

Extracción de información de análisis de antivirus hechos sobre archivos infectados por el virus Wannacry.

Information extraction on antivirus analytics done on files infected by the virus Wannacry.

Realizado por

ANDRÉS SEBASTIÁN CARRERA CAMACHO

Tutorizado por

EDUARDO GUZMÁN DE LOS RISCOS

Departamento

DEPARTAMENTO DE LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2023

Resumen

En 2017, se liberó un ataque a escala global del virus Wannacry, que afectó a miles de equipos Windows cifrando todos sus archivos, con el fin de extorsionar al usuario para que, a cambio de un rescate, el ciberterrorista liberara sus archivos. Aunque se lanzó un parche de seguridad para detenerlo, muchos equipos siguen sin estar actualizados hoy en día y son vulnerables a sus ataques.

El principal objetivo de este TFG es, a partir de los resultados en formato JSON de varios antivirus que han diagnosticado archivos afectados por el virus Wannacry, extraer y transformar, de forma automática, información necesaria en un formato tabular (CSV) con la que poder facilitar a terceros la aplicación de modelos de ciencia de datos. Con estos modelos se espera obtener más información sobre el virus para poder determinar por qué es o no detectado por ciertos antivirus.

El principal reto de este TFG radica en la complejidad y el volumen de los datos en JSON que contienen la información sobre el virus y que, además, están almacenados en formato jerárquico con diversos niveles de anidamiento. Este TFG compila y procesa toda la información sobre el virus, que el usuario haya subido sobre el mismo a partir de una colección de ficheros en formato JSON y permite generar nuevos ficheros en formato tabular a través de un procesamiento de los datos. Este proceso se pone a disposición de los usuarios a través de una interfaz web donde los usuarios podrán

configurar las opciones de salida de los ficheros en formato tabular. Finalmente, en este TFG se han realizado pruebas con un conjunto de 1318 ficheros sobre Wannacry en formato JSON con un tamaño total de 107 MB.

Palabras clave:

wannacry, JSON, Flask, ingeniería de datos

Abstract

In 2017, a global-scale attack of the Wannacry virus was unleashed, affecting thousands of Windows computers by encrypting all their files. The purpose was to extort users into paying a ransom in exchange for the cybercriminal releasing their files. Despite the release of a security patch to halt its spread, many systems remain unupdated to this day, leaving them vulnerable to its attacks.

The main objective of this Bachelor's Thesis is to automatically extract and transform necessary information from JSON-formatted outputs of various antivirus software that have diagnosed files affected by the Wannacry virus. The goal is to present this information in a tabular format (CSV), facilitating third parties' implementation of data science models. Through these models, the aim is to gain more insights about the virus and understand why certain antivirus programs are able or unable to detect it.

The primary challenge of this Bachelor's Thesis lies in the complexity and volume of the JSON data containing virus-related information. Furthermore, this data is stored in a hierarchical format with varying levels of nesting. This thesis compiles and processes all available information about the virus, provided by users through a collection of JSON files. It enables the generation of new tabular-format files by processing the data. This entire process is accessible to users through a web interface, where they can configure the output options for the tabular-format files.

Keywords:

Wannacry, JSON, Flask, data engineering

Índice

Resumen 1

Abstract 1

Índice 1

Introducción 1

1.1 Motivación 1

1.2 Objetivos del proyecto 2

1.3 Antecedentes 3

1.4 Estructura de la memoria 5

Tecnologías y herramientas utilizadas 7

2.1 Metodología 8

2.2 Herramientas 9

2.1 Técnicas 11

Especificación y Análisis 13

3.1 Requisitos Funcionales 14

3.1 Requisitos No Funcionales 18

Diseño del Sistema 19

4.1 Diagrama de la Base de Datos 19

4.2 Diagrama de Despliegue 23

4.3 Diagramas de Secuencia 25

Implementación 31

5.1 Interfaces Comunes **31**

5.1 Pantallas del Administrador **35**

Pruebas **53**

6.1 Pruebas de Usabilidad **53**

6.2 Tests Unitarios **60**

Conclusiones y Trabajos futuros **69**

7.1 Dificultades encontradas **69**

7.1 Posibles ampliaciones **71**

7.1 Objetivos cumplidos y Conclusiones **72**

Referencias **75**

Manual de Instalación **77**

Requerimientos: **77**

Instalación: **77**

Manual de Usuario **79**

1

Introducción

1.1 Motivación

El formato de archivos JSON [1] permite almacenar información de forma jerárquica proporcionando un medio de representación de la información más rico que otros formatos como CSV [2] que únicamente permite almacenar los datos en forma de tabla. La complejidad de representación que permite JSON puede ser, sin embargo, un problema cuando queremos tratar la información a través de técnicas de Ciencia de Datos. En este ámbito normalmente se utiliza información representada de forma tabular y los algoritmos de Ciencia de Datos no suelen trabajar de forma directa con datos en formato JSON.

Por esta razón, cuando disponemos de datos en formato JSON es preciso realizar un proceso intensivo de transformación de los datos jerárquicos a un formato lineal para poder trabajar con técnicas de Ciencia de Datos.

Un caso concreto de esta problemática son los datos del virus Wannacry que proporciona la empresa VirusTotal a través de una API y que están en formato JSON. Se trata de un volumen de datos muy grande que consta de 1318 ficheros (con un tamaño total de 107 MB) en formato JSON con una jerarquía a varios niveles. Su procesamiento para poder extraer información significativa en formato de tabla es un proceso complejo que necesitaría poder ser automatizado.

1.2 Objetivos del proyecto

El principal objetivo de este Trabajo de Fin de Grado (TFG) es realizar una aplicación, que se puede emplear para extraer y simplificar la información de archivos JSON [1] para transformar a formato CSV [2] usando técnicas de ingeniería de datos, a la vez que se aplican conocimientos y técnicas aprendidas a lo largo de la carrera de Ingeniería Informática. Haciendo uso de los resultados de varios antivirus que han diagnosticado archivos afectados por el virus Wannacry, se propone crear una herramienta que facilite la posterior aplicación de modelos de Ciencia de Datos con el fin de intentar detectar el motivo por el que este virus es o no detectado por ciertos antivirus.

Para la creación de esta aplicación, se han considerado ciertos objetivos adicionales que se deben cumplir para considerar el desarrollo de la misma un éxito:

- Funcionalidad de inicio y cierre de sesión.
- Visualización de forma amigable y directa de los archivos en formato JSON.
- Realización de pantallas con una interfaz de usuario intuitiva y agradable.
- Subida y borrado de archivos JSON a la Base de Datos de aplicación.
- Creación, borrado y edición de las cuentas de usuario.

- Transformación de los archivos a CSV, tanto de forma completa como seleccionando tan sólo la información necesaria.

1.3 Antecedentes

Se ha realizado un estudio de mercado analizando aplicaciones que tengan elementos y funcionalidades similares a la aplicación desarrollada en este trabajo, con el fin de sacar a relucir sus principales características y los componentes que precisan de mejora o de los que carecen.

Aplicación	Descripción	Elementos a Mejorar
convertcsv	Herramienta Web bastante completa que permite convertir archivos JSON a CSV, ofreciendo varias opciones de personalización a la hora de realizar la transformación	-Solo permite transformar los elementos de uno en uno. -No permite extraer cierta información concreta del JSON. -No está preparada para tratar JSON con altos niveles de anidamiento.
data.page	Aplicación web que ofrece distintas utilidades para el tratamiento y conversión de archivos JSON y CSV y la exportación de los mismos a distintos formatos.	-Solo permite transformar los elementos de uno en uno. -Solo permite subir archivos que pesen hasta 1 MB. -No permite extraer cierta información concreta del JSON. -Algunas funciones requieren membresía premium.
gigasheet	Aplicación web bastante completa que permite subir múltiples archivos JSON para visualizarlos, editarlos y realizar transformaciones de datos sobre ellos.	-El proceso de descarga es algo confuso y, aunque se pueden descargar varios archivos a la vez, estos han de convertirse a CSV de uno en uno previamente. -No permite extraer cierta información concreta del JSON.
GroupDocs	Gestor de documentos que permite subir múltiples archivos JSON y transformarlos a formato CSV.	-No permite extraer cierta información concreta del JSON. -La conversión es considerablemente lenta con archivos de gran tamaño.

		-No está preparada para tratar JSON con altos niveles de anidamiento. -No permite hacer una previsualización de los archivos antes de convertirlo.
aconvert	Aplicación web que permite subir varios archivos JSON y transformarlos a formato CSV, eligiendo a posteriori cuáles de los elementos se van a descargar y permitiendo editar sus nombres.	-No permite extraer cierta información concreta del JSON. -No permite hacer una previsualización de los archivos antes de convertirlo. -No está preparada para tratar JSON con altos niveles de anidamiento.

Tabla 1.1. Análisis de posibles Competidores.

Tras analizar las características de estas herramientas (Tabla 1.1), llegamos a la conclusión de que no solo el mercado de aplicaciones para transformar información es bastante escaso, sino que ninguna de ellas presenta funcionalidades para extraer datos concretos de los conjuntos de archivos. En base al estudio realizado, se estiman que las características más importantes a incluir en la aplicación son las siguientes:

- Presentación agradable e intuitiva para el usuario.
- Poder transformar varios archivos a formato CSV al mismo tiempo y en un tiempo de respuesta aceptable.
- Ofrecer la posibilidad de visualizar los archivos JSON antes de la transformación.
- Extraer las claves que el usuario requiera de cada archivo, en lugar de realizar siempre una transformación completa.

- Preparar la aplicación para trabajar con altos niveles de anidamiento en el archivo JSON, así como para tratar correctamente los arrays [3] que se encuentren dentro del mismo.

1.4 Estructura de la memoria

A continuación se expondrá la organización y la estructura que presenta esta memoria:

Capítulo 2: Tecnologías y herramientas utilizadas

En este capítulo se describirán las principales tecnologías empleadas en la realización de la aplicación, así como las metodologías y procesos que se han empleado durante el desarrollo.

Capítulo 3: Especificación y Análisis

Este capítulo tratará sobre la especificación de los requisitos, tanto Funcionales como No Funcionales, que se deben satisfacer para considerar el desarrollo de la aplicación un éxito.

Capítulo 4: Diseño del sistema

En este capítulo se discutirá el diseño del sistema desde un punto de vista más técnico mediante el uso de diagramas de Diagramas estructurales, de despliegue, de casos de uso y de secuencia.

Capítulo 5: Implementación

El objetivo de este capítulo será diseccionar cada una de las funcionalidades de la aplicación para comprender el diseño de las mismas.

Capítulo 6: Pruebas

En este capítulo se comentarán las pruebas realizadas, así como los resultados de las mismas.

Capítulo 7:Conclusiones y Trabajos Futuros

En este capítulo final se reflexionará sobre el trabajo realizado durante el desarrollo de la aplicación, los problemas encontrados durante el mismo y si se han cumplido los objetivos presentados. También se hablará sobre posibles mejoras y funcionalidades añadidas con las que se podría expandir la herramienta.

Apéndices:

Manual de Instalación:

Breve documento en el que se detallarán los pasos a seguir para que el usuario promedio pueda descargar la aplicación y hacer que esta funcione de forma correcta.

Manual de Usuario:

Documento realizado con el fin de dar a conocer al usuario promedio las funcionalidades de la aplicación y el alcance de la misma.

2

Tecnologías y herramientas utilizadas

En este capítulo se definirá la metodología empleada durante el desarrollo de la aplicación así como las herramientas y técnicas de las que se han hecho uso para producir los resultados esperados.

2.1 Metodología

Durante el desarrollo de este trabajo se ha hecho uso de una metodología ágil, específicamente Scrum [4]. Esta es una metodología para la gestión de proyectos en los que se necesita obtener resultados de forma rápida en entornos susceptibles al cambio a lo largo de ciclos iterativos centrados en el cliente para crear un producto de forma incremental.

Se ha seleccionado esta metodología tanto por lo bien que se acomodaba al desarrollo (cambios incrementales sobre la aplicación que han de ser validados por el tutor), como por la familiaridad y experiencia con la que cuenta el autor respecto a ella, puesto que ya había sido empleada en múltiples asignaturas de la carrera de Ingeniería Informática.

Scrum posee varios elementos que han de conocerse para poder comprender de forma correcta su funcionamiento:

2.1.1 Fases:

Product Backlog: Formado por las tareas principales que constituyen el proyecto, y se corresponden con los requisitos del mismo. Estos han sido acordados al principio del desarrollo entre el tutor y el autor y se expande sobre ellos en el Capítulo 3.

Sprint: Intervalo de tiempo en el que se debe hacer el desarrollo de ciertas tareas que se han definido previamente en una reunión conocida como Sprint Planning. Durante el desarrollo, ha habido múltiples Sprints plannings entre el tutor y el autor para validar las funciones implementadas y concretar el próximo Sprint.

2.1.2 Roles:

Product Owner: Rol que se encarga de administrar las lista de tareas que componen el Product Backlog y de asegurarse de que las soluciones a las que se han llegado durante el desarrollo son correctas. Este rol ha sido desempeñado de forma conjunta entre el tutor y el autor.

Scrum Master: Este rol se encarga de dirigir los Sprints y asignar tareas, siendo análogo a un jefe de proyecto. Este rol ha sido desempeñado en su totalidad por el tutor.

Scrum Team: Grupo de personas que trabajan en el proyecto y desarrollan las tareas durante los sprints. Puesto que este trabajo es individual, este rol ha sido desempeñado únicamente por el autor.

2.2 Herramientas

Para el desarrollo de este trabajo, se han hecho uso de distintos lenguajes de programación, herramientas y librerías, las cuales se describen a continuación:

2.2.1 Flask

Flask [5] es un framework web de Python [6] para el desarrollo de aplicaciones web es conocido por su simplicidad y facilidad de uso. Proporciona herramientas y funcionalidades básicas para construir aplicaciones web, permitiendo la creación de aplicaciones de ámbito pequeño que requieran tanto front-end como back-end.

A diferencia de otros frameworks web más completos y con mayor grado de estructuración, Flask proporciona sólo lo esencial y, al estar construido de forma modular, permite a los desarrolladores agregar las funcionalidades que necesiten según sus requisitos específicos, como podrían ser autenticación o manejo de bases de datos.

Flask permite definir rutas y asociarlas con funciones que se ejecutarán cuando se acceda a dichas rutas, lo que permite manejar las diferentes páginas y funcionalidades de tu aplicación e incluye un servidor de desarrollo que permite probar tu aplicación localmente antes de implementarla en un entorno de producción.

Puesto que Flask es ideal para el desarrollo de aplicaciones web pequeñas y medianas debido la simplicidad y flexibilidad que proporciona a la hora de integrar la comunicación entre los distintos componentes de la misma, ha sido elegido para este proyecto frente a opciones más potentes pero más complejas como podrían ser Django o CubicWeb.

2.2.2 MongoDB

MongoDB [7] es una base de datos NoSQL de código abierto que se ha vuelto muy popular en el desarrollo de aplicaciones web y móviles que, a diferencia de las bases de datos relacionales tradicionales, utiliza un modelo de almacenamiento orientado a documentos.

Los datos se almacenan en forma de documentos BSON (Binary JSON), pueden tener diferentes campos y estructuras, lo que lo hace altamente flexible para aplicaciones con esquemas de datos variables o en constante evolución.

MongoDB está diseñado para ser altamente escalable y distribuido puesto que divide los datos en fragmentos (sharding) que se reparten en diferentes servidores para mantener un alto nivel de rendimiento. Además, se puede integrar de forma sencilla con Python mediante el uso de la librería Pymongo.

Tanto su fácil integración con Python como por las ventajas que ofrece el uso de una Base de Datos no relacional son motivos relevantes para justificar su uso, aunque el factor decisivo ha sido la velocidad. Puesto que se están tratando archivos grandes

y además en un número elevado, se necesita la mayor velocidad posible para que la subida y recuperación de datos se de en unos tiempos aceptables-

2.2.3 Otras Herramientas

Tecnologías Web

Para el desarrollo de la parte de front-end, se han hecho uso de las Tecnologías Web HTML y CSS para la construcción de las interfaces, así como de Javascript para afinar el comportamiento en muchas de ellas [8].

Jinja2

Motor de plantillas empleado por Flask para generar contenido HTML de manera más sencilla y flexible, permitiendo separar la lógica de presentación. Jinja2 se integra con el código HTML y permite realizar operaciones con la información obtenida de las funciones Python de la aplicación.

2.1 Técnicas

Para el desarrollo de la aplicación se ha trabajado con 1318 ficheros JSON resultantes de análisis de diversos antivirus sobre archivos afectados por el virus Wannacry, llegando a tener estos archivos un tamaño superior a 1.200 KB en algunos casos. Debido a la complejidad y al tamaño que poseen estos archivos, hay que realizar una serie de transformaciones sobre los mismos para poder tratarlos de forma eficiente. Puesto que los archivos JSON están estructurados como árboles, hacer una conversión directa a un formato plano como es CSV es imposible. Por ello, hay que realizar primero lo que se conoce como un aplanado a la hora de subir los archivos.

Este proceso de aplanado se refiere a la acción de convertir un dicho JSON anidado (con múltiples niveles de jerarquía) en una estructura plana, donde cada elemento se representa con una ruta única, extrayendo los elementos anidados y colocándolos

en una estructura lineal. Hay que tener en cuenta durante el proceso que puede haber arrays dentro de los JSON, los cuales han de ser tratados aparte para extraer los elementos en su interior y no perder información en el aplanado.

Para ser mostrados en el visualizador de archivos, los JSON aplanados han de ser devueltos de nuevo a su forma jerárquica, realizando el proceso inverso. Aunque existiría la opción de trabajar con los JSON estructurados durante todos los procesos y aplanarlos únicamente a la hora de la transformación a CSV, las operaciones serían mucho más costosas y se reducirían los tiempos de respuesta.

3

Especificación y Análisis

En este apartado se detallarán los requisitos Funcionales y No Funcionales que debe satisfacer la aplicación. Para facilitar su comprensión, los requisitos Funcionales han sido clasificados según el usuario que haga uso de los mismos, pudiendo ser este Administrador o Analista:

- Administrador: Este actor se encargará de realizar la subida de los archivos a la base de datos, así como de gestionar las distintas cuentas de usuario registradas en la aplicación.

- Analista: Una vez subidos los archivos por el Administrador, el usuario con perfil Analista podrá realizar las distintas transformaciones y conversiones a formato CSV.

3.1 Requisitos Funcionales

En este apartado se detallarán los requisitos Funcionales y No Funcionales que debe satisfacer la aplicación. Para facilitar su comprensión, los requisitos Funcionales han sido clasificados según el usuario que haga uso de los mismos.

3.1.1 Administrador

RF-1. Subida de archivos:

RF-1.1. El sistema permite la subida de los archivos subidos por el administrador.

RF-1.2. El sistema aplica la restricción de seleccionar al menos un archivo.

RF-2. No se pueden seleccionar archivos repetidos.

RF-2.1. Se crean los distintos elementos que se van a guardar en la base de datos: Archivo JSON, Tipo, Datos JSON y Posibles valores.

RF-2.2. El sistema guarda los elementos creados en la base de datos.

RF-3. Borrado de archivos:

RF-3.1. El sistema pregunta al usuario si está seguro de querer borrar todos los archivos.

RF-3.2. El sistema borra todos los elementos de la base de datos.

RF-4. Administración de Cuentas de Usuario:

RF-4.1. El sistema mostrará la información de todas las cuentas de usuario registradas en la aplicación, tanto de analistas como de administradores.

RF-4.2. El usuario podrá crear otras cuentas de usuario, siempre que tenga suficientes permisos.

RF-4.3. El usuario podrá editar otras cuentas de usuario, siempre que tenga suficientes permisos.

RF-4.4. El usuario podrá borrar otras cuentas de usuario, siempre que tenga suficientes permisos.

3.1.2 Analista

RF-5. Transformación de archivos.

RF-5.1. El sistema permite al usuario seleccionar las claves que considere necesarias para la transformación a CSV.

RF-5.2. El sistema aplica la restricción de seleccionar al menos una clave.

RF-5.3. El sistema obtiene el archivo de la Base de Datos.

RF-5.4. El sistema aplica un formateo al archivo para que su presentación en CSV sea correcta y atractiva.

RF-5.5. El sistema guarda este conjunto formateado como CSV en la ruta destino.

RF-6. Visualización de archivos por lotes.

RF-6.1. El sistema obtiene el archivo de Datos JSON que se corresponde con el tipo de archivo seleccionado de la Base de Datos.

RF-6.2. El sistema obtiene los posibles valores para cada una de las claves de este archivo JSON de la Base de Datos.

RF-6.3. El sistema muestra una representación visual del archivo de Datos añadiendo a cada clave un desplegable con sus posibles valores.

RF-7. Transformación de archivos por lote.

RF-7.1. El sistema permite al usuario seleccionar las claves que considere necesarias para la transformación a CSV.

RF-7.2. El sistema aplica la restricción de seleccionar al menos una clave.

RF-7.3. El sistema obtiene los archivos necesarios de la base de datos según el tipo seleccionado y genera una tabla de previsualización para el usuario.

RF-7.4. Si el usuario quiere seguir adelante con la transformación, el sistema aplica un formateo a los elementos seleccionados para que su presentación en CSV sea correcta y atractiva.

RF-7.5. El sistema guarda este conjunto formateado como CSV en la ruta destino.

RF-8. Transformación binaria de archivos por lote.

RF-8.1. El sistema permite al usuario seleccionar las claves que considere necesarias para la transformación a CSV.

RF-8.2. El sistema aplica la restricción de seleccionar al menos una clave.

RF-8.3. El sistema crea una tabla con los posibles valores del actual tipo de archivo y crea una tabla con cada uno de los posibles valores en las columnas, los archivos en las filas, y 0 o 1 en las celdas en función de si ese elemento aparece o no en cada archivo.

RF-8.4. El sistema emplea esta información para generar una tabla de previsualización para el usuario.

RF-8.5 Si el usuario quiere seguir adelante con la transformación, el sistema aplica un formateo a los elementos seleccionados para que su presentación en CSV sea correcta y atractiva.

RF-8.6. El sistema guarda este conjunto formateado como CSV en la ruta destino.

RF-9. Comprobar archivos repetidos:

RF-9.1. El sistema comprueba si ya existe un archivo con el mismo nombre en la ruta destino y, en caso afirmativo, ofrece al usuario las opciones de reemplazarlos, ignorarlos o elegir cuales reemplazar.

RF-9.2 Cuando se ha seleccionado la opción de reemplazar todos los archivos, el sistema sobrescribe los elementos con el mismo nombre.

RF-9.3. Cuando se ha seleccionado la opción de ignorar todos los archivos, el sistema no sobrescribe los elementos con el mismo nombre y los deja intactos.

RF-9.4. Cuando se ha seleccionado la opción de elegir cuales reemplazar, el sistema sobrescribe tan sólo los elementos seleccionados por el usuario.

3.1.3 Comunes a ambos usuarios

RF-10. Inicio de sesión

RF-10.1. El sistema recibe las credenciales introducidas por el usuario y valida si son correctas.

RF-10.2. El inicio de sesión fallará si el usuario o la contraseña están vacíos o si la combinación de estos no existe en la base de datos.

RF-10.3. Si se ha marcado la opción de Recordarme, la sesión se debe mantener activa incluso cuando se cierre el navegador.

RF-10.4. El sistema crea las variables de sesión del usuario.

RF-10.5. El sistema inicia la sesión, cargando distintos elementos en función del tipo de usuario.

RF-11. Cierre de sesión

RF-11.1. El sistema borra las variables de sesión del usuario.

RF-11.2. El sistema cierra la sesión.

RF-12. Visualización de archivos

RF-12.1. El sistema obtiene el archivo JSON que se corresponde con el archivo seleccionado de la Base de Datos.

RF-12.2. El sistema muestra una representación visual del archivo JSON.

3.1 Requisitos No Funcionales

RNF-1. Tiempo de respuesta. El sistema debe funcionar de forma rápida a las peticiones realizadas por los usuarios. Se acepta entre 1 y 2 segundos entre la navegación entre páginas, y se estima un tiempo medio de 3 minutos por cada 300 archivos subidos.

RNF-2. Visualización. El sistema debe poseer interfaces gráficas intuitivas y bien formadas.

RNF-3. Fiabilidad. El sistema debe funcionar y responder correctamente a las peticiones del usuario.

RNF-4. Escalabilidad. El sistema debe estar preparado para facilitar la incorporación de nuevas funcionalidades en el futuro.

4

Diseño del Sistema

En este capítulo se expondrán varios diagramas que permitirán conocer mejor la estructura y el funcionamiento del sistema.

4.1 Diagrama de la Base de Datos

Un diagrama Entidad-Relación es una herramienta visual que representa cómo diferentes elementos del sistema a los que se llamará entidades se relacionan entre ellos. Estas se muestran como rectángulos con propiedades conocidas como atributos, y las relaciones entre ellas se expresan como líneas, que pueden ser de diferentes tipos (uno a uno, uno a muchos, muchos a muchos).

Aunque se ha utilizado un gestor de base de datos no relacional, más concretamente MongoDB, las entidades (o colecciones) definidas en nuestro modelo, están relacionadas entre sí y de ahí el uso de este tipo de representación. El principal objetivo de este diagrama es especificar cómo se ha organizado la información en una la Base de Datos y cómo se interrelacionan los diferentes componentes del modelo de datos empleado en el sistema.

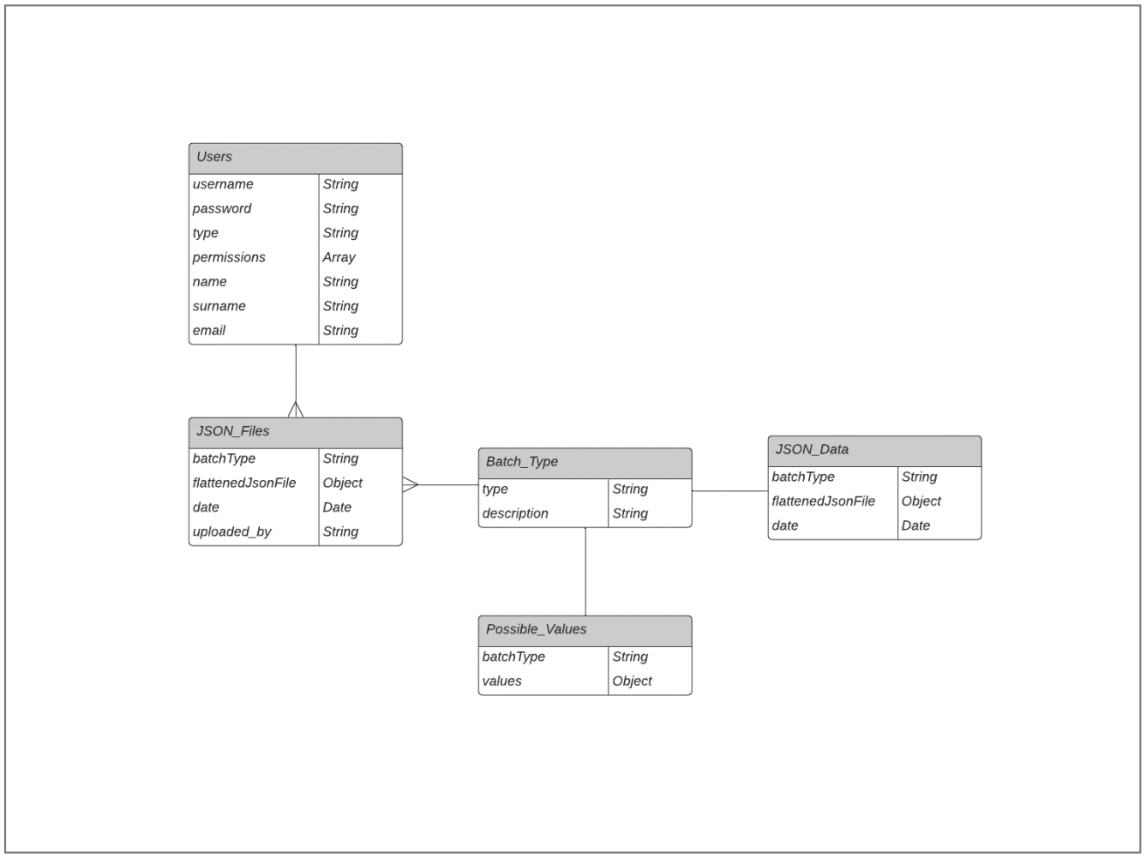

Figura 4.1. Diagrama Entidad-Relación de la Base de Datos.

Para el desarrollo de la aplicación se ha optado por un esquema de tamaño reducido (Figura 4.1), puesto que la mayor carga de base de datos, los archivos JSON, se pueden almacenar como un solo campo. Se procederá ahora a describir las entidades y los atributos que las componen:

- *Users:* Es la entidad que almacena todos los usuarios activos del sistema, que pueden ser de dos tipos distintos, Analista o Administrador. El usuario "admin" siempre se encuentra activo, puesto que es el que se emplea para crear el resto de usuarios que harán uso de la aplicación. Sus atributos son los siguientes:

    - *username:* Nombre del usuario.

    - *password:* Contraseña del usuario.

    - *type:* Tipo del usuario. Como se ha mencionado anteriormente, podrá tomar los valores "analyst" o "admin".

    - *permissions*: Permisos del usuario a la hora de administrar otras cuentas de Administrador, puesto que las cuentas de Analista no requieren autorización para ser modificadas. Como la administración es una función disponible sólo para el Administrador, el Analista tendrá siempre vacío este campo. Hay 4 posibles tipos de permisos:

        - *CREATE:* Permite crear otros Administradores.

        - *UPDATE:* Permite modificar otros Administradores.

        - *DELETE:* Permite borrar otros Administradores.

        - *GRANT:* Permite otorgar permisos a otros Administradores.

    - *name:* Nombre del usuario. Es un atributo opcional.

    - *surname:* Apellido o apellidos del usuario en caso de tener más de uno. Es un atributo opcional.

    - *email:* Dirección de correo electrónico del usuario. Es un atributo opcional.

- JSON_Files: Representa cada uno de los archivos JSON subidos a la Base de Datos. Para que su manejo resulte más sencillo durante la aplicación, se guarda una versión aplanada del mismo. Sus atributos son los siguientes:

    - *batchType*: Es el tipo del archivo JSON. El resultado de un análisis puede producir varios archivos, teniendo cada uno de ellos su propio tipo.

    - *flattenedJsonFile:* Objeto que guarda el archivo JSON ya aplanado. Contiene 2 arrays, uno con las claves y otro con los valores obtenidos tras el proceso de aplanamiento. Puesto que se trabaja con elementos de gran tamaño, este atributo puede llegar a ser muy pesado.

    - *date:* Fecha y hora en la que se ha subido el archivo.

    - *uploaded_by:* Guarda el nombre del administrador que ha subido cada archivo para llevar una trazabilidad.

- Batch_Type: Guarda los posibles tipos de los archivos. Al subir un nuevo archivo JSON, se comprobará si existe en esta tabla y se añadirá a ella en caso de no hacerlo. Sus atributos son los siguientes:

    - *type:* Es el tipo del archivo JSON.

    - *description:* Breve descripción del tipo de archivo que sirve para aclarar los elementos que contiene el mismo. Se obtiene de unos valores haciendo uso de una función en el código.

- JSON_Data: Esta tabla almacena los Datos JSON, que no son más que una combinación de todos los JSON subidos que pertenezcan al mismo tipo, puesto que es necesario tener todas las claves para realizar la transformación por lotes. Sus atributos son los siguientes:

- *batchType:* Es el tipo del archivo JSON.

- *flattenedJsonFile:* Objeto que guarda el archivo JSON ya aplanado. A diferencia del atributo que se guarda en la tabla "JSON_Files", el campo de claves dentro de este objeto representa todas las claves que puede tener un archivo que pertenezca al tipo correspondiente.

- *date:* Última fecha de modificación del archivo de datos.

- Possible_Values: Guarda todos los posibles valores que puede tener una clave dentro de un mismo tipo de archivo y se emplea, al igual que son los elementos de "JSON_Data", en la conversión de archivos por lote. Son almacenados aparte para no exceder la limitación de tamaño máximo de MongoDB. Sus atributos son los siguientes:

- *batchType:* Es el tipo del archivo JSON.

- *values:* Este objeto guarda múltiples arrays que contienen todos los posibles valores para cada una de las claves de los JSON del tipo correspondiente.

4.2 Diagrama de Despliegue

Un diagrama de despliegue es una representación visual utilizada para mostrar cómo los componentes de un sistema se distribuyen en el hardware y en el entorno de ejecución, mostrando cómo se ejecutan los diferentes elementos del software en los recursos físicos, como servidores, dispositivos y redes.

Puesto que la aplicación que se ha desarrollado es de tamaño reducido y no interactúan muchos nodos, el Diagrama de Despliegue (Figura 4.2) es bastante sencillo:

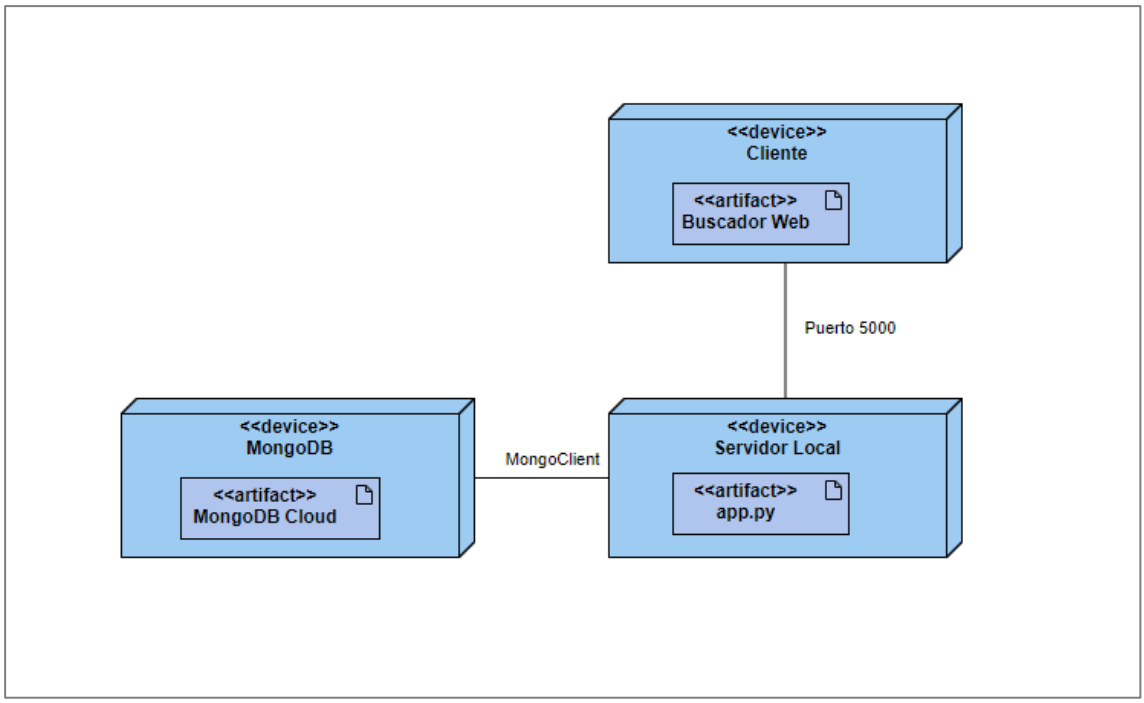

Figura 4.2. Diagrama de Despliegue.

Se procederá ahora a explicar brevemente cada uno de los nodos:

- *Cliente:* Representa el dispositivo mediante el cual el usuario interactúa con la aplicación, siendo representado por el navegador web que renderiza las diferentes pantallas de la herramienta. Puesto que el despliegue se ha realizado de forma local con Flask, el puerto del mismo es el 5000.

- *Servidor Web:* Representa el dispositivo que se encarga de alojar la herramienta y de procesar las distintas peticiones que se reciben. El artefacto que se encarga de ello es la clase app.py, que es en la que Flask realiza la configuración y el despliegue del servidor.

- *MongoDB:* Representa el dispositivo que se encarga de almacenar la información necesaria para el funcionamiento de la aplicación, siendo este la Base de Datos MongoDB. El artefacto que se encarga de la gestión de este nodo es MongoDB Cloud, al cual se conecta haciendo uso de la librería Pymongo y la clase MongoClient, que

instancia el clúster donde se encuentra la información en el código para poder interaccionar con ella.

4.3 Diagramas de Secuencia

Un diagrama de secuencia es una herramienta visual utilizada en el diseño y modelado de sistemas para representar la interacción entre objetos y el flujo de mensajes o acciones en una secuencia temporal. Este tipo de diagrama se enfoca en mostrar cómo diferentes objetos colaboran entre sí para llevar a cabo una funcionalidad específica.

Los principales elementos que se han de conocer para entender su funcionamiento son los siguientes:

- Actores: Es un elemento que representa a una entidad externa al sistema que interactúa con dicho sistema. En todos los diagramas de esta aplicación, el actor siempre será el usuario de la misma.
- Componentes: Son los distintos elementos de la aplicación que realizan las distintas funcionalidades de la misma y se comunican entre ellos mediante el uso de mensaje.
- Línea de vida: Representan la duración de existencia de un objeto a lo largo del tiempo y se dibujan como líneas verticales.
- Mensajes: Representan las comunicaciones o interacciones entre los objetos, indicando la acción que debe realizarse mediante el envío de mensajes.

Se han realizado los siguientes diagramas de secuencia para definir el comportamiento de las principales funcionalidades que ofrece la aplicación al usuario.

Inicio de sesión.

En este diagrama (Figura 4.3) se describe el proceso que sigue un usuario para iniciar sesión en la aplicación, y sirve como un sencillo ejemplo de cómo interactúan los distintos elementos de la misma.

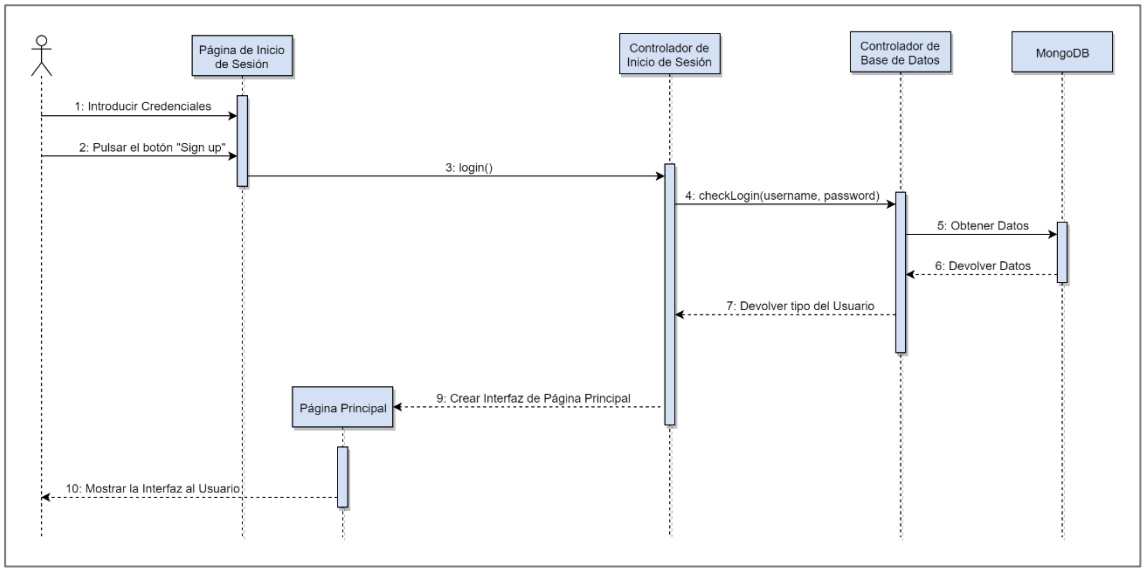

Figura 4.3. Diagrama de Secuencia "Inicio de sesión".

Los elementos de este diagrama son los siguientes:

- *Página de Inicio de Sesión:* Página de inicio de sesión del programa, con la que el usuario interactúa para iniciar la secuencia. Tras introducir sus credenciales y pulsar el botón “Sign up”, se comunicará con el controlador de Inicio de Sesión haciendo uso de la función login().

- *Controlador de inicio de sesión:* Se encarga de gestionar el comportamiento del inicio de sesión. Para ello, comunicará con el controlador de Base de Datos haciendo uso de la función checkLogin(), a la que le pasará el nombre de usuario y la contraseña.

- *Controlador de Base de Datos y MongoDB*: Controlador de la Base de Datos, que se encarga de realizar llamadas a MongoDB y de gestionar ciertos elementos de la lógica.

- *Página Principal:* Una vez terminado todo el proceso de inicio de sesión, se renderiza la Página Principal de la aplicación y se le muestra al usuario.

Subida de Archivos

En este diagrama (Figura 4.4) se describe el proceso que sigue un usuario Administrador para subir los archivos a la aplicación. Se ha considerado necesario definirlo puesto que es una tarea compleja en la que se realizan múltiples llamadas entre los componentes.

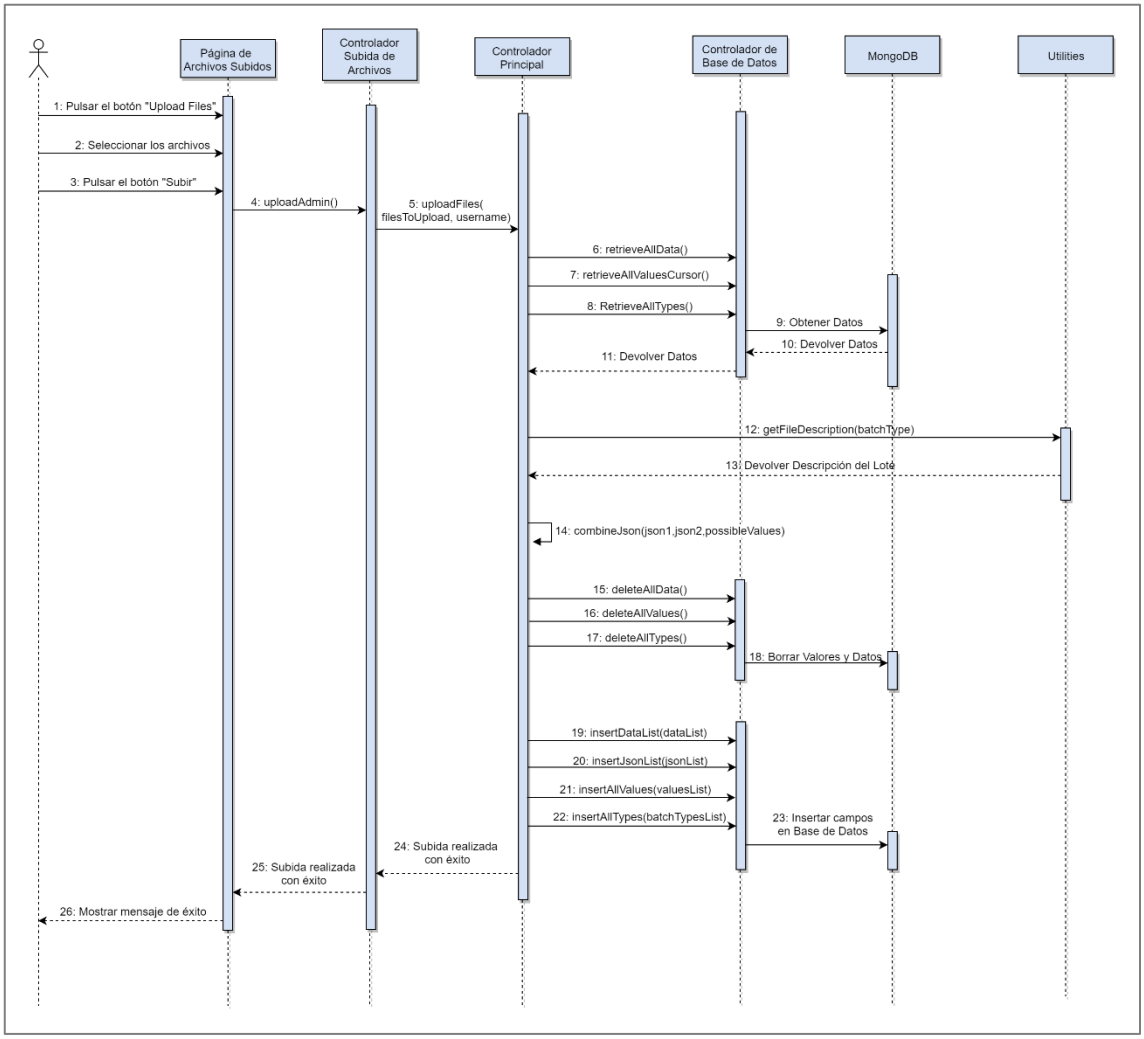

Figura 4.4. Diagrama de Secuencia "Subida de Archivos".

Los elementos de este diagrama son los siguientes:

- *Página de Archivos subidos:* Página de Archivos Subidos de la aplicación, con la que el usuario interactúa para iniciar la secuencia. El usuario deberá pulsar el botón "Upload Files", seleccionar los archivos que quiera subir, y darle al botón "Upload", lo cual iniciará la secuencia.

- *Controlador de Subida de Archivos:* Controlador llamado por la función uploadAdmin() por la página de Archivos Subidos. Esta funcionalidad delega todas sus funciones en el controlador principal, que se encargará de realizar la subida. Cuando esta esté terminada, este controlador se encargará de actualizar la página de subida de archivos y de mostrar el mensaje de éxito.

- *Controlador Principal:* Controlador que contiene varias funciones que se usan en distintas partes de la aplicación. Se encargará de realizar la subida de archivos haciendo varias llamadas a Utilities y al controlador de la Base de datos. También será necesaria una llamada a un método del propio controlador llamado combineJson(), que será esencial a la hora de crear los Posibles Valores y los Datos JSON.

- *Utilities:* Componente que contiene pequeñas funcionalidades como puede ser formateo de datos o transformación de ciertos tipos de archivos.

- *Controlador de Base de Datos y MongoDB:* Sirven la misma función que en el diagrama anterior, sólo que en este caso además de obtener información de la Base de Datos, se están realizando inserciones y borrados de datos.

Transformación de Archivos

En este último diagrama (Figura 4.5) se describe el proceso mediante el cual un usuario Analista realizará la transformación de archivos a CSV. Se ha elegido la transformación por lotes puesto que tiene más complejidad que la transformación

individual. No se ha incluido un diagrama de la transformación binaria por lotes puesto que debido a su similitud con este diagrama, resultaría redundante.

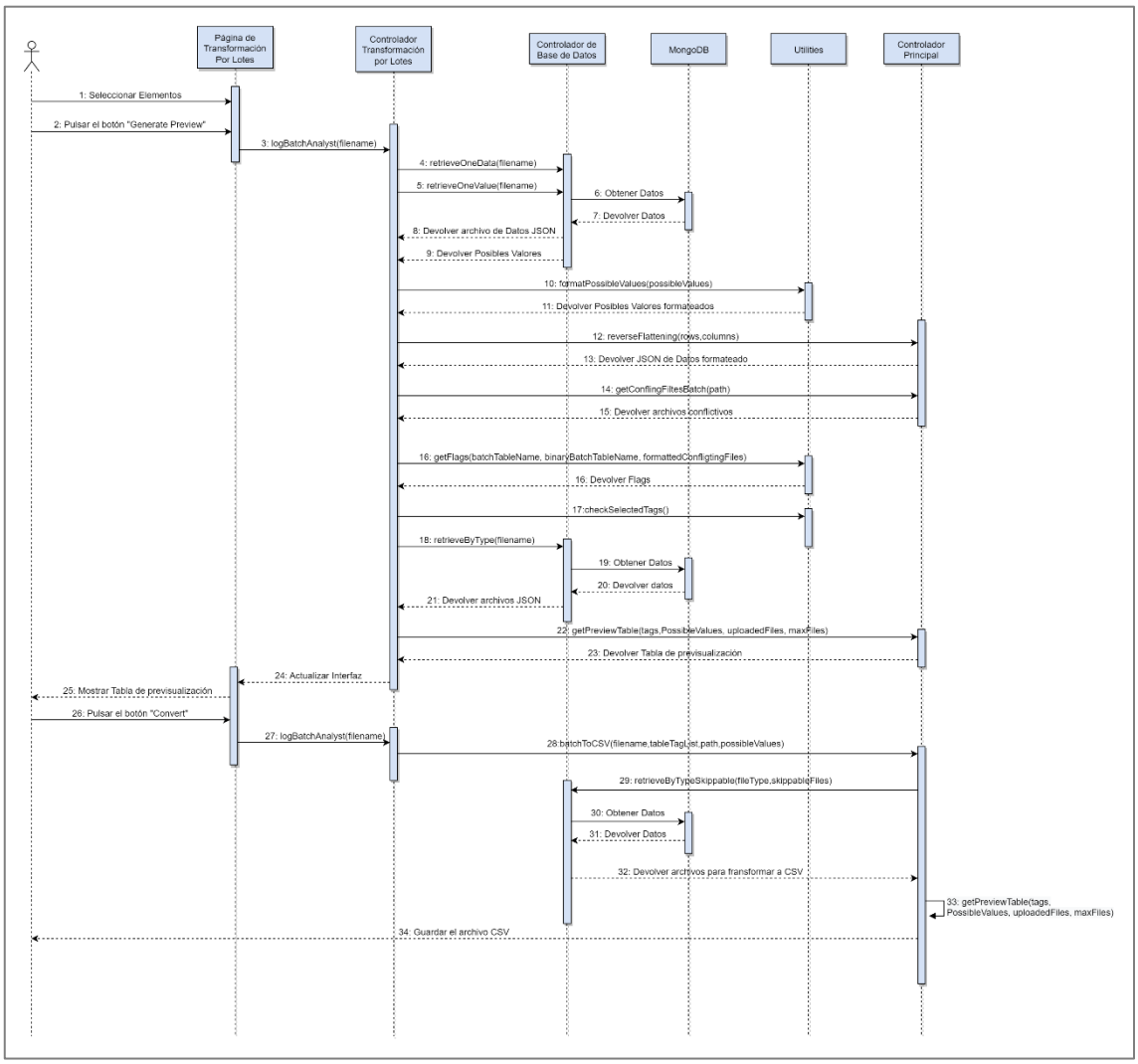

Figura 4.5. Diagrama de Secuencia "Transformación de Archivos".

Los elementos de este diagrama son los siguientes:

- *Página de Transformación por Lotes:* Página de Transformación por Lotes de la aplicación. Tras seleccionar las claves que el usuario quiere transformar de los archivos de ese tipo de lote y pulsar el botón de “Generate Preview”, se iniciará la secuencia. Esta página vuelve a ser mostrada al usuario tras generar la tabla de previsualización, siendo necesario que este pulse el botón “Convert” para terminar el proceso.

- *Controlador de Transformación por Lotes:* Controlador llamado por la función logBatchAnlyst() por la página de Archivos Subidos, pasándole el tipo de archivo. A diferencia del diagrama anterior, casi toda la funcionalidad de esta secuencia se encuentra en este controlador, puesto que requiere más interacciones con el usuario. También se encarga de actualizar la interfaz de la Página de Transformación por Lotes para mostrarle la tabla de previsualización y que este pueda elegir si quiere seguir adelante con la transformación o no.

- *Controlador Principal:* En este caso el Controlador Principal toma un papel más secundario, aportando principalmente las funciones que permiten la creación de la tabla de previsualización y la creación del archivo CSV.

- *Utilities, Controlador de Base de Datos y MongoDB:* Sirven la misma función que en los diagramas anteriores.

5

Implementación

En este capítulo se va a describir en profundidad el funcionamiento de la lógica de la aplicación. A continuación se expondrán cada una de las pantallas que la componen con el fin de detallar la lógica que hay debajo de ellas, separadas por usuario para ofrecer una visión más clara del sistema.

5.1 Interfaces Comunes

Pantalla login.html

Esta es la primera pantalla (Figura 5.1) a la que tendrá acceso el usuario en el momento de conectarse a la aplicación, y se encargará de permitir el inicio de sesión

al mismo. Para que se pueda realizar el inicio de sesión, es necesario que un administrador cree primero una cuenta de usuario, ya que debido a sus especificaciones iniciales, se carece de un sistema de registro.

Una vez que el usuario haya introducido sus credenciales, se llamará al endpoint "/login". Este comprobará si el usuario y contraseña introducidos haciendo uso de la función checkLogin(), que devolverá el tipo de usuario en caso de que el proceso haya sido un éxito, y lanzará la correspondiente excepción en caso contrario (Figura 5.2).

Una vez obtenido el tipo del usuario, se procederá a asignar las variables de sesión del mismo: Su nombre de usuario, su ya mencionado tipo de usuario, y sus permisos en caso de ser Administrador. Además, en caso de que el usuario haya seleccionado la opción de "Remember me", su sesión se hará permanente, y se mantendrá activa incluso cuando se cierre el Navegador Web que ha utilizado para iniciarla.

Finalmente, el sistema se encargará de redireccionar al usuario a la pantalla de inicio o "index.html". Si el usuario no ha cerrado su sesión e intenta acceder a esta pantalla, será redirigido preventivamente a la página de inicio sin necesidad a que vuelva a introducir sus credenciales.

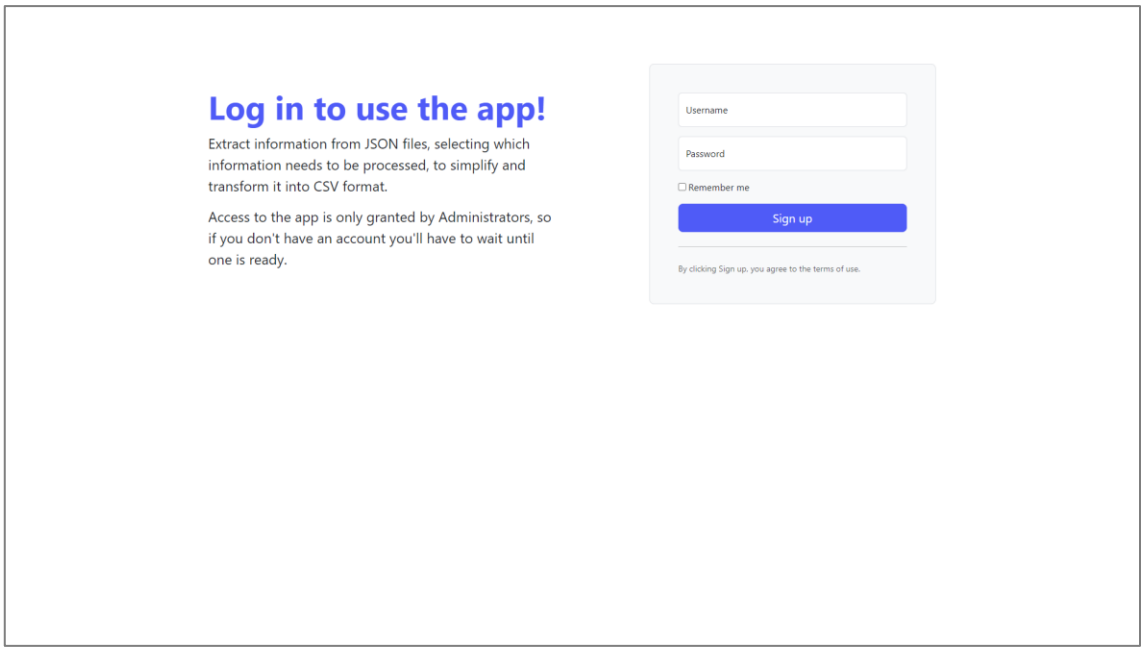

Figura 5.1. Pantalla de inicio de sesión.

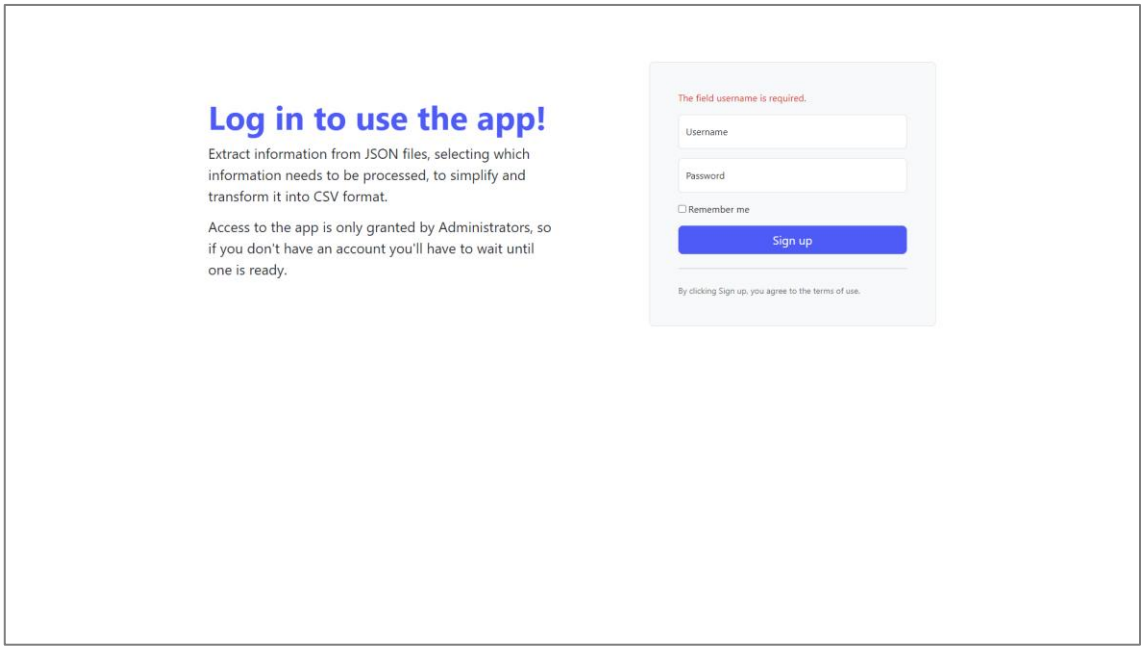

Figura 5.2. Error al tratar de iniciar sesión.

Pantalla index.html

Página principal de la aplicación (Figura 5.3), que sirve como bienvenida y puerta de enlace al resto de las funcionalidades. Su complejidad lógica es mínima, siendo su principal utilidad redireccionar al usuario al resto de pantallas y aportar información sobre la utilidad principal de la herramienta.

Figura 5.3. Pantalla principal.

common_components.html

Este elemento de la aplicación no se corresponde con una pantalla como tal, sino que contiene distintos elementos comunes al resto de pantallas de la aplicación, y son invocados por las mismas cuando es necesario. Los componentes que lo componen son los siguientes:

- *Barras de navegación:* Uno de los componentes principales son las barras de navegación que son empleadas por el Analista y el Administrador, cuya principal función es permitir la navegación por las distintas pantallas de la

aplicación. Puesto que algunas de las pantallas a las que tiene acceso cada tipo de usuario son distintas, se necesita hacer uso de dos barras diferentes.

- *Pie de página:* Pie de página común a todas las pantallas, por lo que se ha decidido crearlo como componente común para simplemente ser invocado por las pantallas de la aplicación.

- *Sistema de Cierre de Sesión:* Aparece como un botón en las barras de navegación, pero como posee cierta lógica de negocio se va a discutir en este punto. Al pulsar el botón, sea cual sea el tipo de usuario, se llamará al endpoint "/logout", el cual se encargará de cerrar la sesión haciendo uso para ello el Nombre de Usuario y de redireccionar de nuevo a la página de login.

5.1 Pantallas del Administrador

Pantalla uploaded_files.html

Esta pantalla (Figura 5.4) se encarga de mostrar al usuario todos los archivos que están subidos en la Base de datos, además de permitirle realizar ciertas funcionalidades que dependen del tipo de usuario. El mensaje explicativo que aparece en la pantalla también es variable, y el texto dependerá de nuevo de si el usuario es un Administrador o un Analista.

Los archivos subidos se encuentran agrupados en función de si están relacionados, obtenidos de la Base de Datos haciendo uso de la función listDirectoryGroup(), la cual se encargará también de agrupar todos los archivos que sean parte del mismo resultado de Análisis de Antivirus, bajo el mismo desplegable para facilitar su representación por pantalla. También aparece indicada la fecha de subida de cada uno de ellos, y se puede acceder a cada uno de los archivos para ir a la pantalla " individual_file.html" para ver la representación JSON de los mismos.

La principal funcionalidad de la pantalla como administrador es la subida de archivos (Figura 5.5), a la cual se accede llamando al endpoint "/uploadAdmin". Debido a su complejidad, se va a explicar su funcionamiento en detalle:

- Se pide al administrador que seleccione los archivos que quiere subir, ya sea uno o varios. En caso de que ningún archivo sea seleccionado, o que se intente subir alguno que ya se encuentre en la Base de Datos, se lanzará la excepción correspondiente.

- Una vez seleccionados los archivos, se llama a la función uploadFiles(), que será la que se encargue de realizar toda la lógica.

- Tratará de obtener, en caso de que existan, todos los registros de las tablas de Batch_Type, JSON_Data y de Possible_Values, creando un diccionario vacío para ellos en caso de no existir valores en la Base de Datos. También se creará una lista vacía para almacenar los archivos JSON seleccionados por el usuario.

- Por cada archivo a subir, se realizan una serie de transformaciones y conversiones.

- Se crea un Documento para insertar en la tabla JSON_Data el archivo JSON, para lo cual será necesario realizar un aplanado del mismo, proceso explicado detalladamente en el Capítulo 2. Una vez creado el registro, se añade a la lista para ser insertado en la colección cuando acabe todo el proceso.

-También se crea un Documento para cada tipo de archivo distinto que se ha subido a la Base de Datos, al cual se le agrega una breve descripción del mismo y se añadirá a un diccionario para su posterior inserción en la tabla Batch_Type.

- Ahora se tratarán los elementos de Datos y Posibles Valores. Para ello, se hará uso de la función combinarJson(), que se encargará de anexionar las claves que no existieran previamente en el registro de Datos del tipo de archivo actual, con el fin de crear un elemento que contenga todas las claves.

- Una operación similar se realizará con los Posibles Valores, solo que en lugar de combinar las claves, se combinarán los valores de todos los archivos JSON subidos.

- Ambos elementos se agregan a sus diccionarios correspondientes para ser insertados en Base de Datos. Para los posibles valores, debido al gran tamaño al que llegan a crecer, es necesario hacer una transformación previa, pues que MongoDB tiene un tamaño máximo de Documento. Para trabajar alrededor de esta restricción, se dividirá cada uno de los Documentos en fragmentos, de forma que el máximo número de posibles valores en cada uno de ellos sea 3000.

- Finalmente, haciendo uso de las funciones insertJsonList(), insertTypeList(), insertDataList() e insertAllValues(), se subirán todos los Documentos a la Base de Datos, tras lo cual el sistema redireccionará de nuevo a "uploaded_files.html" y mostrará un mensaje de que la subida ha sido un éxito.

El administrador también tiene acceso a la funcionalidad de borrar todos los archivos de la Base de datos. En caso de hacer clic en el botón de "Delete All Files", el sistema le preguntará si está seguro de que quiere seguir adelante. En caso de hacerlo, se borrarán todos los documentos de las colecciones de Datos JSON, archivos JSON, tipo y Posibles Valores, tras lo cual el sistema redireccionará de nuevo a "uploaded_files.html" y mostrará un mensaje de que el borrado de archivos ha sido un éxito.

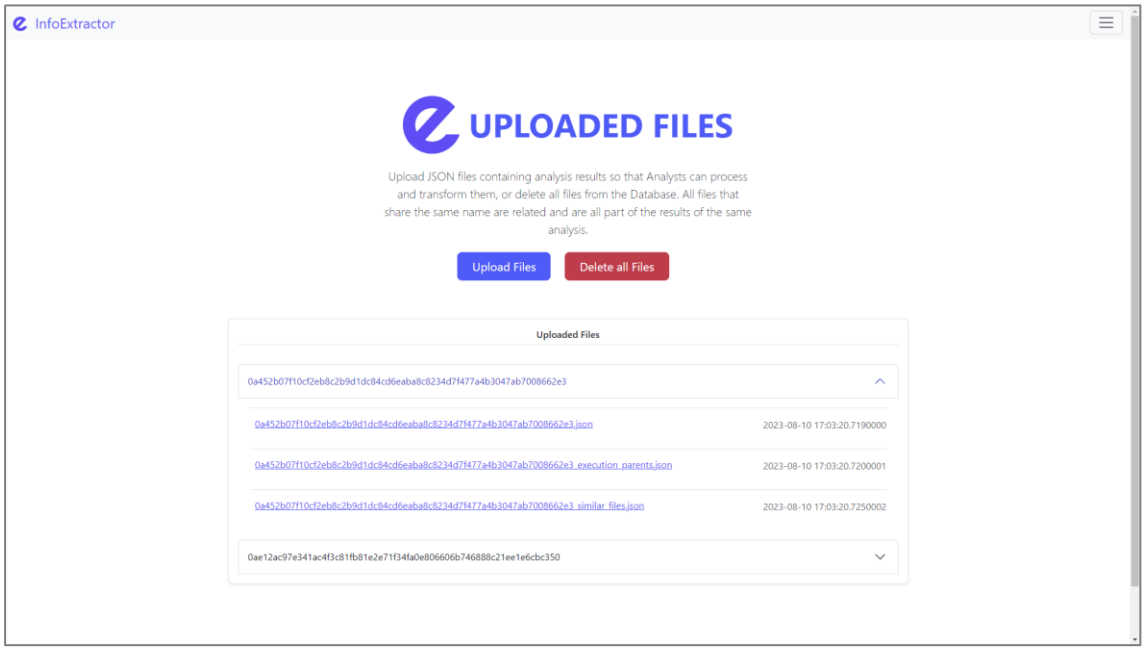

Figura 5.4. Pantalla de Archivos Subidos del Administrador.

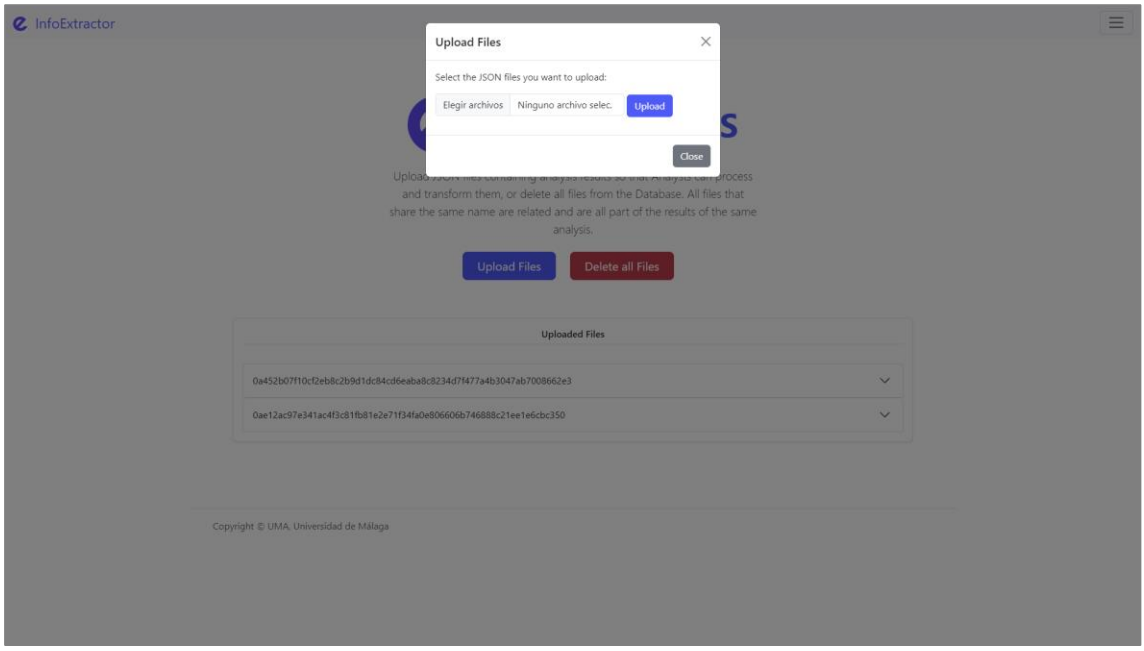

Figura 5.5. Subir Archivos.

Pantalla individual_file.html

Para acceder a esta pantalla (Figura 5.6), se debe hacer clic en cualquiera de los archivos subidos en la pantalla "uploaded_files.html", lo que mostrará una representación en forma de árbol del mismo. Puesto que el Administrador no puede hacer transformaciones de datos sobre los archivos, la única función de la pantalla para el mismo será ver de forma amigable e intuitiva el archivo, apareciendo también un texto explicativo para definir la funcionalidad de la página.

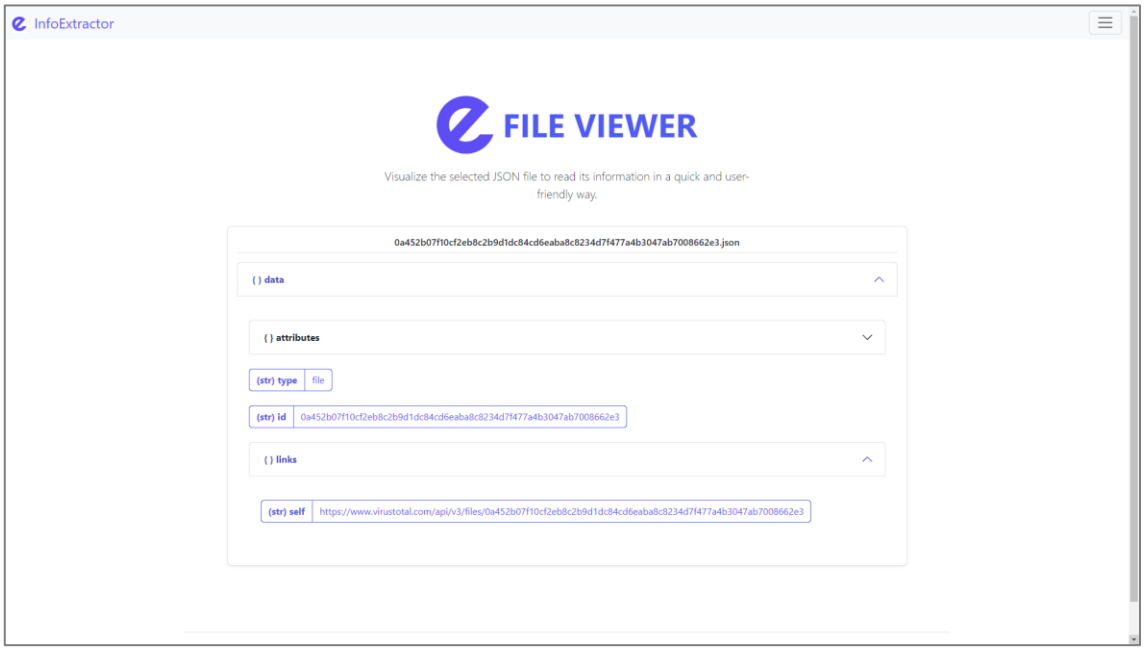

Figura 5.6. Pantalla de Archivo Individual del Administrador.

Pantalla admin_manage.html

Esta pantalla (Figura 5.8) es exclusiva del Administrador, y su función principal es la de gestionar las cuentas de los usuarios, independientemente del tipo de los mismos. También se muestra toda la información sobre las cuentas de los usuarios que se encuentran registrados en la aplicación, a excepción de sus contraseñas, haciendo uso para ello de la función retrieveAllUsers(), los cuales serán clasificados en función de si son Analistas o Administradores para representarlos en la pantalla.

Dependiendo de los permisos con los que cuente el Administrador que ha accedido a ella, tendrá acceso a ciertas funcionalidades sobre el resto de los Administradores, aunque siempre podrá realizar operaciones sobre los Analistas independientemente de sus permisos.

Las principales acciones que se pueden llevar a cabo son las siguientes:

- *Crear Usuario:* Se accede a esta funcionalidad (Figura 5.8) llamando al endpoint "/newAccount", el cual recoge toda la información introducida por el Administrador sobre la creación del usuario. Comprueba el tipo del usuario que se está creando y llama a la función addUser() con los parámetros correspondientes. Si el nombre de usuario ya existía con anterioridad, se lanza la excepción correspondiente, mientras que en caso contrario se redirige de nuevo a "admin_manage.html" y se muestra un mensaje de éxito.

- *Modificar Usuario:* Se accede a esta funcionalidad llamando al endpoint "/updateAccount", el cual recoge toda la información introducida por el Administrador sobre la creación del usuario, guardando de forma preventiva el anterior nombre de usuario en caso de que este se quiera modificar. Comprueba el tipo del usuario que se está creando y llama a la función updateUser() con los parámetros correspondientes. Como la única forma de modificar el nombre de usuario, que hace de id en la colección de datos, es creando un registro nuevo, se borra primero el usuario usando el nombre anterior del mismo que se había guardado y se crea un registro nuevo con la información actualizada. Se comprueba si el nombre de usuario ya existía con anterioridad para lanzar la excepción correspondiente, mientras que en caso contrario se redirige de nuevo a "admin_manage.html" y se muestra un mensaje de éxito.

- *Borrar Usuario:* Se accede a esta funcionalidad llamando al endpoint "/deleteAccount", el cual recoge el nombre del usuario que se ha de eliminar. Llama a la función deleteUser(), que se encargará de eliminarlo de la Base de Datos, tras lo cual se redirige de nuevo a "admin_manage.html" y se muestra un mensaje de éxito.

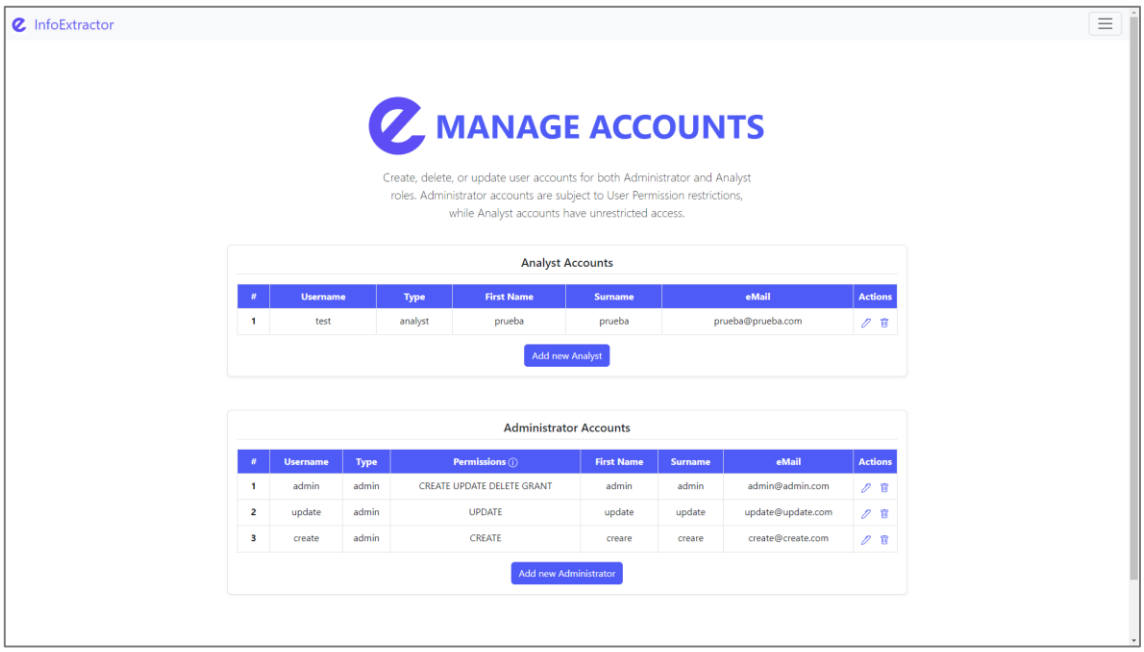

Figura 5.7. Pantalla de Administración de Cuentas.

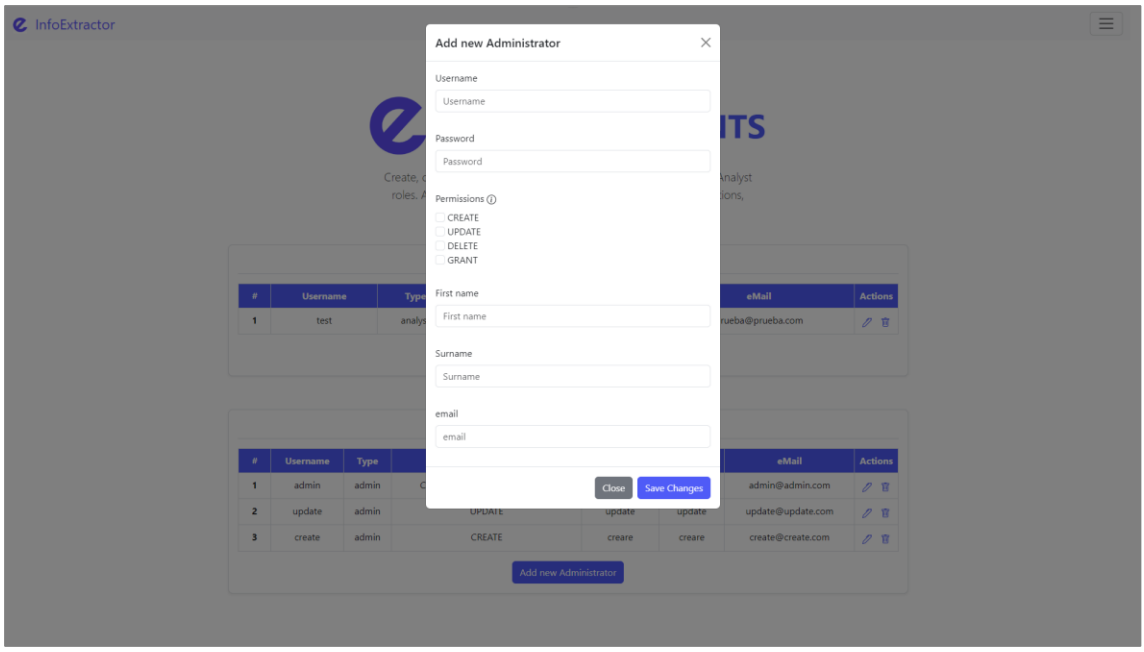

Figura 5.8. Creación de un usuario Administrador.

5.1 Pantallas del Analista

Pantalla uploaded_files.html

En esta pantalla (Figura 5.9) se muestra una información similar a la que aparece en la versión del Administrador, pero cuenta con ciertas diferencias tanto en la representación como en las funcionalidades, como el mensaje explicativo, que se ha actualizado para reflejar el uso del Analista. Para mostrar el archivo JSON, se ha de realizar un proceso inverso al aplanado, que permitirá recuperar la estructura en forma de árbol que necesitamos para la representación.

En la representación de la pantalla, se puede apreciar que se han añadido checkboxes a los desplegables, a cada uno de los archivos subidos, y una checkbox extra para seleccionar todos los archivos. Esto será de utilidad a la hora de realizar la conversión de múltiples elementos sin tener que seleccionarlos uno a uno manualmente.

Dicha conversión es el elemento principal que ofrece esta pantalla. Se presenta una conversión simple a formato CSV, transformando todas las vales de cada uno de los archivos y guardándolos en el sistema de forma individual. El proceso que se realiza es el siguiente:

- Se seleccionan los elementos que se quieran convertir y se hace clic en el botón de “Convert Files”. En caso de no haber seleccionado ninguno, se mostrará el mensaje de error correspondiente.

- Se añadirán estos elementos a una lista, y se llamará a la función encargada de realizar la conversión, transformToCSV(). Está a su vez llamará a la función auxiliar toCSV() con cada uno de los archivos, la cual se encargará de realizar un formateo a cada uno de los pares clave-valor para que se muestren de forma correcta en el CSV. Se ha decidido emplear la notación de columnas “padre_de_la_clave.clave”, con el fin de que no se pierda tanta información por el proceso de aplanado.

- Una vez listo el formateo del archivo, se guardará en formato CSV en la ruta seleccionada por el sistema.

- En caso de que alguno de los archivos seleccionados por el usuario ya existan en dicha ruta, se le ofrecerá al mismo la posibilidad de reemplazarlos todos, ignorarlos todos o seleccionar cuales quiere reemplazar. Independientemente de su elección, los que no hayan generado conflicto de entre los seleccionados serán transformados sin inconveniente.

- Finalmente, se redirigirá al usuario de nuevo a la página “uploaded_files.html” y se mostrará un mensaje que indique que la conversión ha sido un éxito.

La pantalla también ofrece la posibilidad de borrar todos los archivos que se encuentren transformados a CSV por el usuario. Al pulsar en el botón “Delete all Files” se le preguntará al usuario si está seguro de que quiere continuar. En caso afirmativo, se llamará al endpoint “/deleteAnalyst”, el cual borrará todos los elementos CSV del sistema, tanto los individuales como los obtenidos por la transformación por lotes.

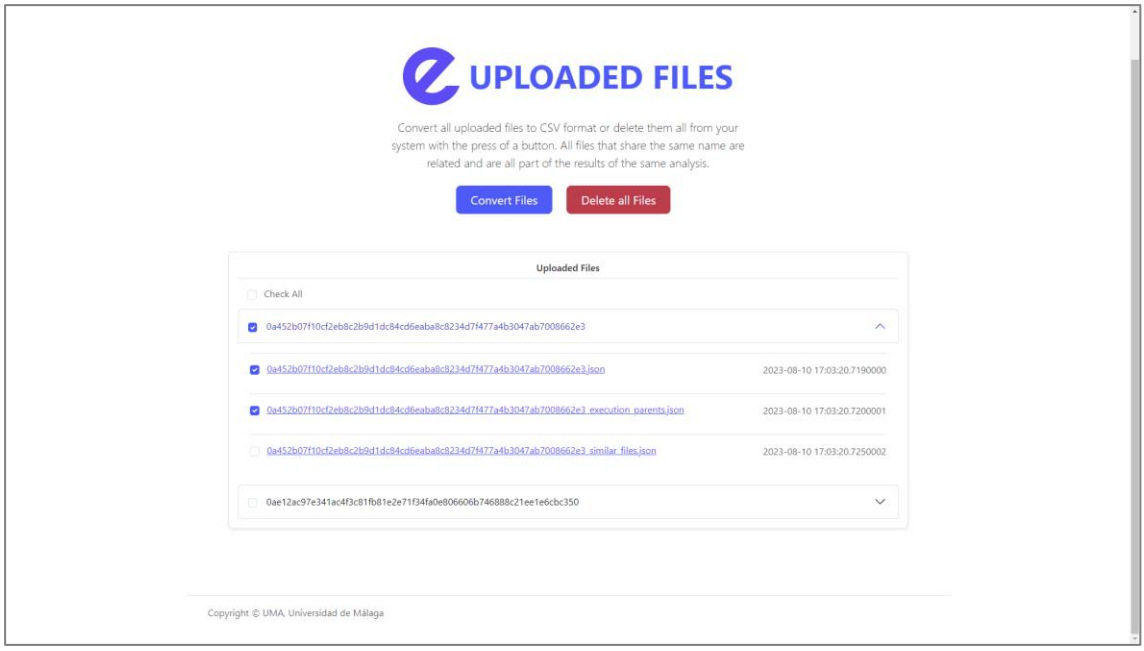

Figura 5.9. Pantalla de Archivos Subidos del Analista.

Pantalla individual_file.html

Al igual que ocurre con la pantalla anterior, en esta (Figura 5.10) también se muestran diferentes elementos y funcionalidades en relación a su contraparte a la que tienen acceso los Administradores. Se han añadido checkboxes a cada una de las claves de la representación JSON del archivo, de forma que al seleccionar una de estas cajas, todos los elementos contenidos dentro serán también seleccionados. Esto hace que el proceso de extracción de las claves sea más sencillo y rápido para el usuario.

La función principal que ofrece esta página es la transformación mediante el uso de claves o etiquetas. Seleccionando las claves que el usuario quiera extraer del archivo, podrá generar un CSV con la información que considere necesaria. El proceso se explica con mayor detalle a continuación:

- Se seleccionan las etiquetas que se quieran extraer el archivo JSON. En caso de no haber seleccionado ninguna, el sistema lanzará el mensaje de error correspondiente.

- A continuación toCSV(), pasándole como parámetro la lista que contiene las claves seleccionadas. Esta se encargará de realizar un formateo a cada uno de los pares clave-valor para que se muestren de forma correcta en el CSV, de forma que sólo se añadirán los elementos que se encuentren en la lista a la conversión. Una vez terminado este proceso, se guardará el archivo en formato CSV en la ruta seleccionada por el sistema.

- Si el archivo ya existía en la ruta seleccionada, se preguntará al usuario si quiere reemplazarlo o ignorarlo.

- Finalmente, se redirigirá al usuario de nuevo a la página " individual_file.html" y se mostrará el correspondiente mensaje que indique que la transformación ha sido un éxito.

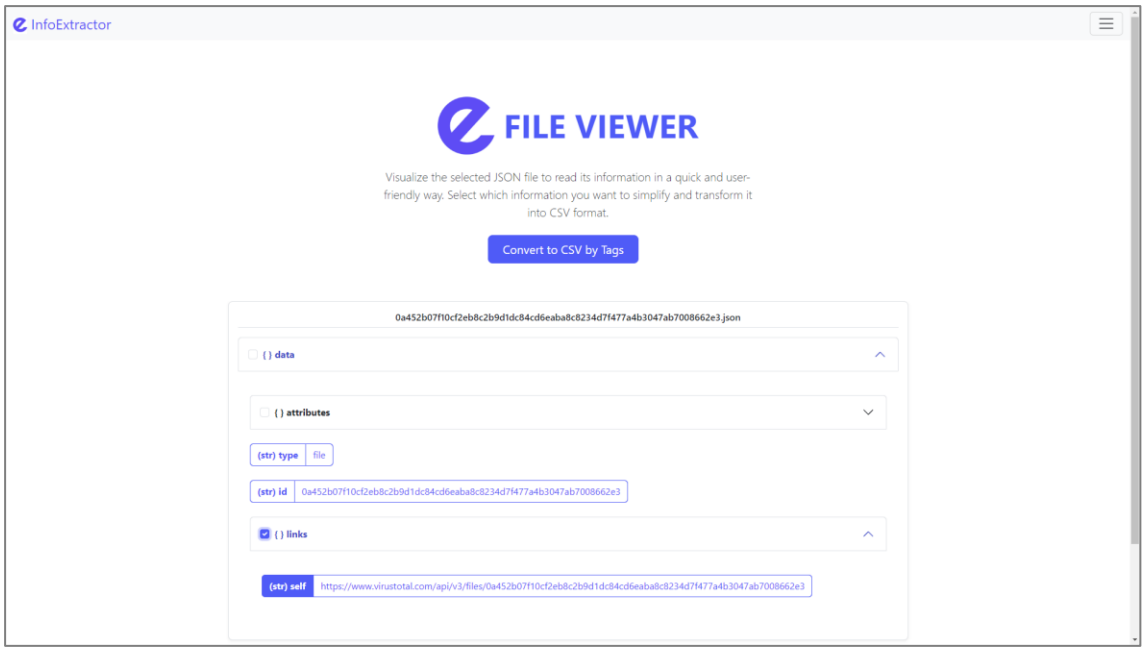

Figura 5.10. Pantalla de Archivo Individual del Administrador.

Pantalla analyst_batch.html

Página (Figura 5.11) de poca complejidad lógica cuya principal funcionalidad es la de permitir al usuario elegir el tipo de lote que desea para hacer la conversión por lotes. Los distintos tipos de lotes disponibles se obtienen de la tabla TBD haciendo uso de la función TBD, la cual extraerá el nombre de cada uno así como su descripción para ser mostrados por pantalla. Hacer clic en el botón “to CSV” de cada uno de ellos redirigirá a la pantalla “analyst_batch_file.html”, mediante la cual se podrá transformar el lote entero a la vez que se realizan las transformaciones necesarias.

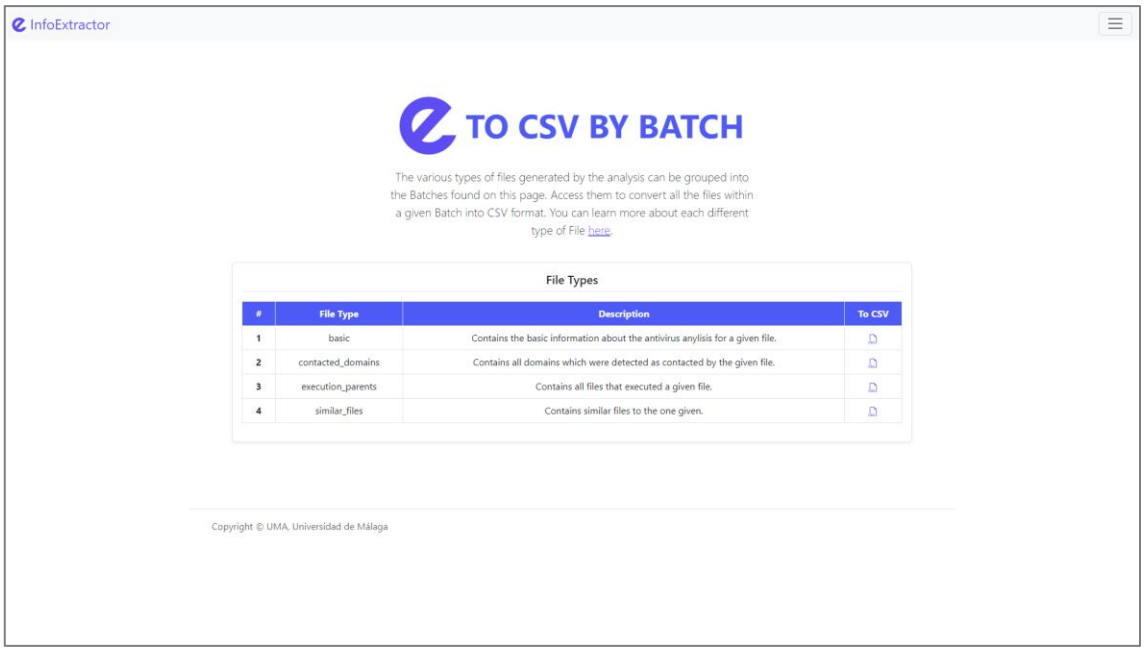

Figura 5.11. Pantalla Selección de Lote.

Pantalla analyst_batch_file.html

En esta pantalla (Figura 5.12) se muestra una visión similar a la que se obtiene en la página "individual_file.html", generando una representación del archivo JSON y ofreciendo al usuario la posibilidad de seleccionar las etiquetas necesarias para la extracción de la información. La primera diferencia se encuentra en que, como este archivo es una representación de todos los del mismo lote, las claves no tienen asociado un solo valor, sino una lista de todos los posibles valores que puede tomar la misma. Estos se obtienen de la tabla de Posibles Valores haciendo uso de la función retrieveOneValue(), que obtendrá los posibles valores para este tipo de lote, tras lo cual serán formateados para ser mostrados por pantalla gracias a la función formatPossibleValues().

Esta pantalla ofrece dos tipos nuevos de transformación, que serán detallados a continuación:

1. Transformación por Lotes (Figura 5.13): Este tipo de transformación generará un solo archivo CSV usando todos los JSON que pertenezcan al mismo lote. Se asociarán las etiquetas seleccionadas por el usuario en las columnas con el nombre del archivo en las filas, siendo la relación que se encuentra en las celdas el valor de cada archivo en dicha etiqueta. En caso de no existir, se mostrará un guion. El funcionamiento detallado es el siguiente:

- Se seleccionan las etiquetas que se quieran extraer el archivo JSON. En caso de no haber seleccionado ninguna, el sistema lanzará el mensaje de error correspondiente.

- Se generará una tabla de previsualización llamando a la función getPreviewTable(). Esta función itera sobre cada uno de los archivos del lote seleccionado y creará un diccionario que relacione las claves y los valores del archivo aplanado del mismo. Como la tabla de previsualización tiene un tamaño máximo de 10 elementos para no sobrecargar la pantalla, el bucle se detendrá al llegar a este número.

- Se itera sobre este diccionario y, si la clave aparece en la lista de etiquetas seleccionadas por el usuario y el valor se encuentra en la lista de Posibles Valores, se añade a la tabla de resultados. En caso de que no esté en esta última lista, se añade un guion, lo que significa que este elemento no posee dicho valor.

- Se formatean los elementos de la tabla para mostrarlos por pantalla y se renderiza la misma para que el usuario la visualice. En caso de querer seguir adelante, el usuario podrá transformar la tabla a CSV haciendo clic en el botón “convert”.

- Se llama ahora a la función batchToCSV(), que generará de nuevo los datos necesarios para crear la tabla de previsualización llamando

getPreviewTable(), sólo que esta vez en vez de 10 elementos como máximo, se le pasará el valor máximo del sistema, para asegurar que todos los archivos van a convertirse a CSV.

- Una vez obtenidos estos datos, se formatearán para ser mostrados de forma correcta, se generará un DataFrame con ellos y se convertirán a formato CSV en la ruta predefinida por el sistema.

2. Transformación Binaria por Lotes (Figura 5.14): Al igual que en el caso anterior, este tipo de transformación generará un solo archivo CSV usando todos los JSON que pertenezcan al mismo lote. En este caso asociarán los posibles valores de las etiquetas seleccionadas por el usuario en las columnas con el nombre del archivo en las filas, siendo la relación que se encuentra en las celdas 0 o 1, en función de si ese valor está presente en el archivo correspondiente. El funcionamiento detallado es el siguiente:

- Se seleccionan las etiquetas que se quieran extraer el archivo JSON. En caso de no haber seleccionado ninguna, el sistema lanzará el mensaje de error correspondiente.

- Se generará una tabla de previsualización llamando a la función getBinaryPreviewTable().

- Genera una lista de elementos uniendo el par clave valor de todos los Posibles Valores.

- Se genera una lista similar con los pares clave valor que se encuentran en la lista de archivos del lote seleccionado, aprovechando para seleccionar tan solo los que se encuentran en la lista de etiquetas. Como la tabla de

previsualización tiene un tamaño máximo de 10 elementos para no sobrecargar la pantalla, el bucle se detendrá al llegar a este número.

- Se itera sobre cada uno de los archivos formateados y comprobamos si coincide con alguno de los posibles valores. Si coincide, se añade un 1 al resultado, mientras que en el caso contrario se añade un 0.

- Se formatean los elementos de la tabla para mostrarlos por pantalla y se renderiza la misma para que el usuario la visualice. En caso de querer seguir adelante, el usuario podrá transformar la tabla a CSV haciendo clic en el botón “convert”.

- Se llama ahora a la función binaryBatchToCSV(), que generará de nuevo los datos necesarios para crear la tabla de previsualización llamando getBinaryPreviewTable(), sólo que esta vez en vez de 10 elementos como máximo, se le pasará el valor máximo del sistema, para asegurar que todos los archivos van a convertirse a CSV.

- Una vez obtenidos estos datos, se formatearán para ser mostrados de forma correcta, se generará un DataFrame con ellos y se convertirán a formato CSV en la ruta predefinida por el sistema.

En cualquiera de los dos casos, si el archivo ya existía en la ruta seleccionada, se preguntará al usuario si quiere reemplazarlo o ignorarlo y, en caso de que la operación haya sido un éxito, se redirigirá al usuario de nuevo a la página “ analyst_batch.html” y se mostrará el correspondiente mensaje que indique que la transformación ha sido un éxito.

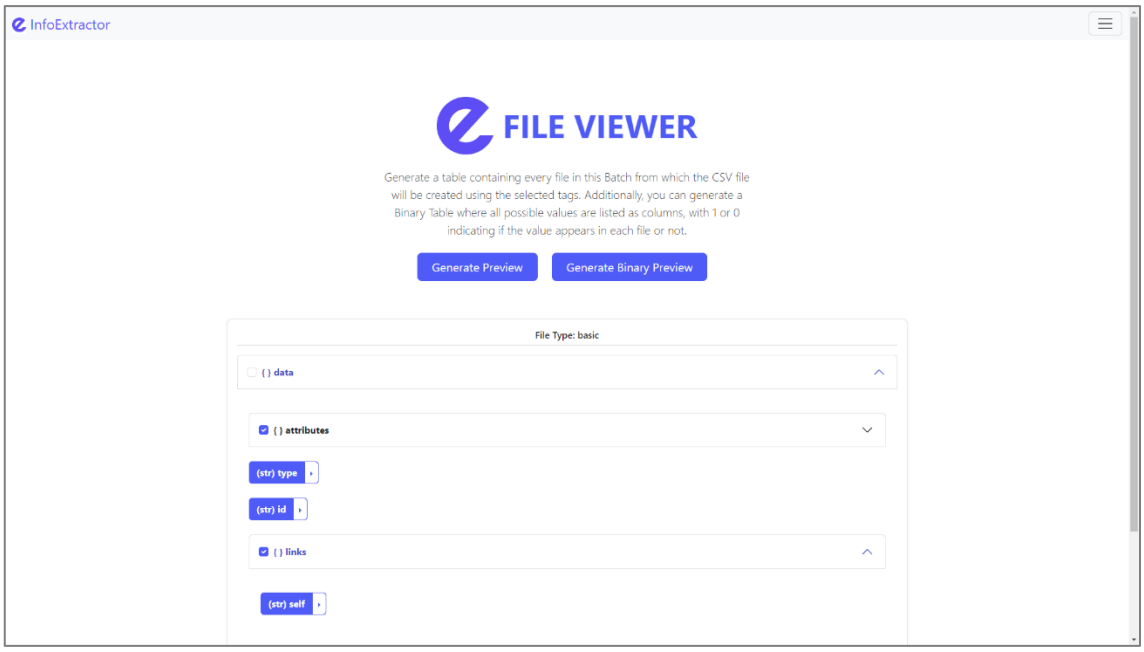

Figura 5.12. Pantalla de Transformación por Lote.

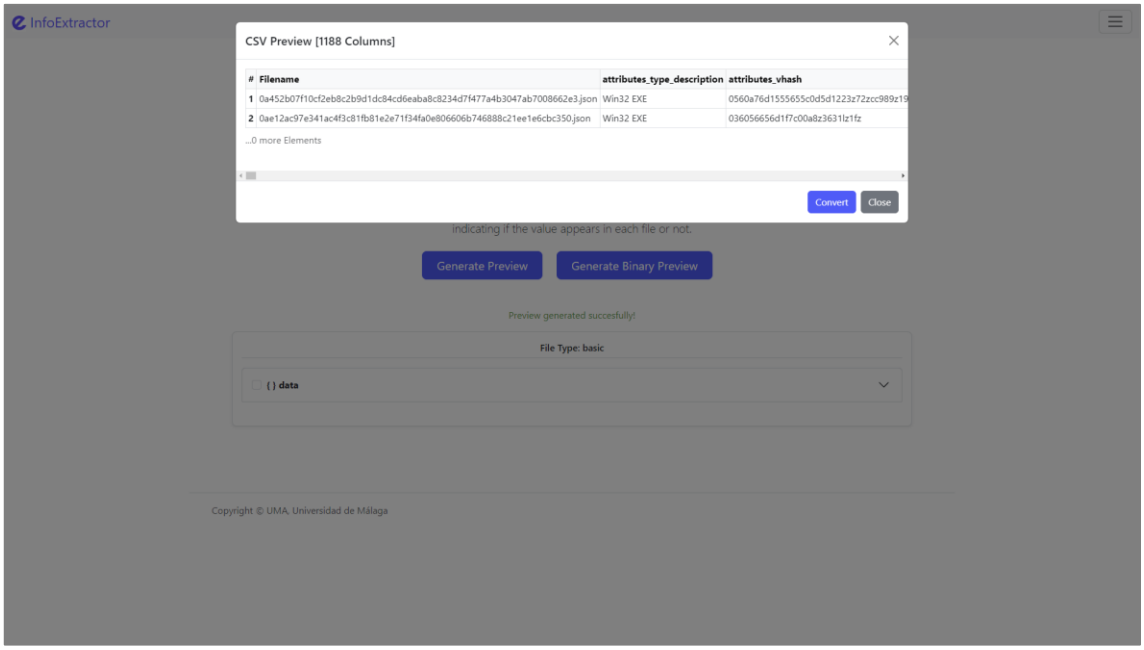

Figura 5.13. Tabla de Previsualización.

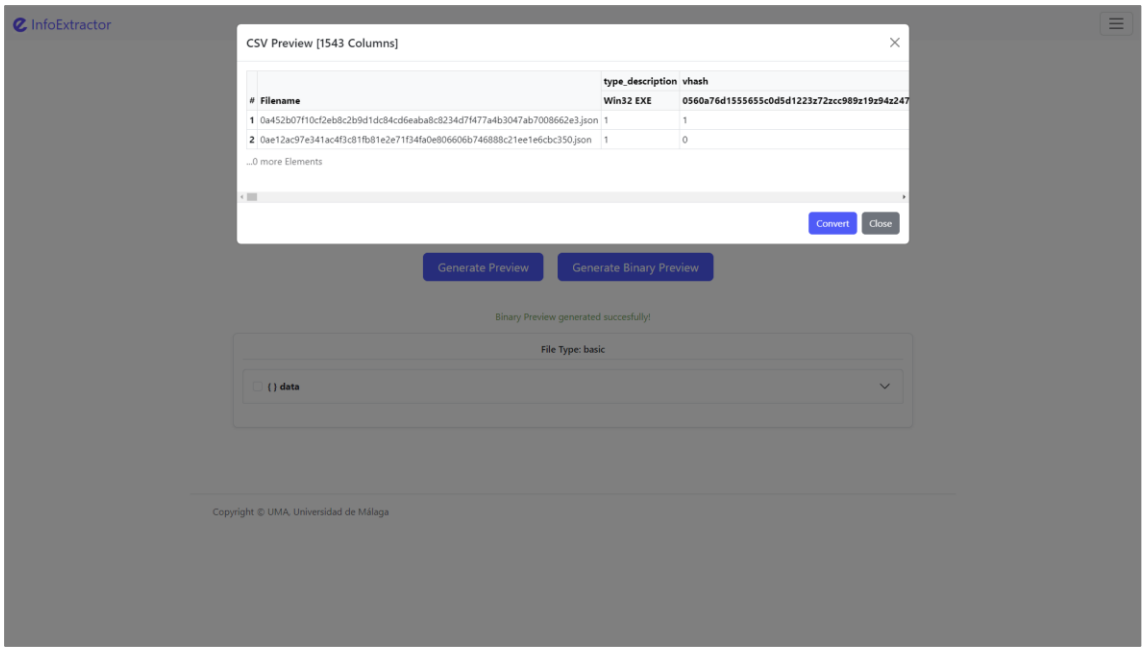

Figura 5.14. Tabla Binaria de Previsualización.

6

Pruebas

Este capítulo tratará sobre las pruebas que se han realizado sobre el sistema. Se van a realizar dos tipos de pruebas: Pruebas de Usabilidad para comprobar que la interfaz es agradable y fácil de usar y Tests Unitarios para asegurar que el funcionamiento del código es correcto.

6.1 Pruebas de Usabilidad

Son un método de evaluación en el campo del diseño de interfaces y experiencia del usuario que tiene como objetivo medir la eficacia y facilidad de uso de una aplicación a través de la observación y el análisis de cómo los usuarios interactúan con dicha herramienta.

Para la realización de las pruebas de usabilidad se ha usado el estándar de pruebas de usabilidad “10 Usability Heuristics for User Interface Design” [9] desarrollado

por Jakob Nielsen. Aunque no es un estándar formal como podrían ser las normas ISO, estas heurísticas son ampliamente reconocidas y utilizadas en el campo de la usabilidad y el diseño de interfaces de usuario.

Las 10 heurísticas en cuestión son las siguientes:

1. *Visibilidad del estado del sistema:* El sistema debe mantener informados a los usuarios sobre lo que está ocurriendo en todo momento mediante la retroalimentación necesaria dependiendo de la situación.

2. *Coincidencia entre el sistema y el mundo real:* El sistema debe emplear un lenguaje de comunicación con el usuario que este pueda entender y que le resulte familiar.

3. *Control y libertad del usuario:* En caso de que el usuario trate de deshacer o eliminar una tarea que pueda tener repercusiones a lo largo del sistema, se le debe informar antes de realizarla para poder cancelarla en caso de que sea necesario.

4. *Consistencia y estándares:* Se debe usar una misma terminología y diseño de la aplicación a lo largo de todas las pantallas de la misma.

5. *Prevención de errores:* Se debe tratar de prevenir y manejar los errores en lugar de mostrar al usuario un mensaje con el mismo, puesto que puede resultar confuso.

6. *Reconocimiento en lugar de recuerdo:* Las funcionalidades y tareas de la aplicación deben ser visibles y fácilmente accesibles por el usuario.

7. *Flexibilidad y eficiencia de uso:* Los usuarios expertos podrían optimizar el uso de la aplicación usando atajos de teclado para acciones comunes.

8. *Estética y diseño minimalista:* Las interfaces deben ser visualmente agradables, de forma que no contengan más información de la estrictamente necesaria para no sobrecargar ni confundir al usuario.

9. *Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores:* Se deben mostrar mensajes de error claros al usuario para que sepa cómo reaccionar ante las situaciones inesperadas que se encuentre durante el uso de la aplicación.

10. *Ayuda y documentación:* Se debe proporcionar información sobre el uso de la aplicación y documentación siempre que sea necesario para el usuario.

Se pueden clasificar las pruebas de usabilidad en 3 tipos:

Inspección.

La inspección es un proceso en el que un equipo de evaluadores o expertos examinan detenidamente la aplicación con el fin de determinar el grado de usabilidad de la misma. En este caso, el autor se encargará de tomar el rol del grupo de expertos, haciendo uso de las heurísticas definidas previamente. Cada heurística se aplica a diferentes aspectos del diseño y la interacción de la aplicación, con el objetivo de crear una interfaz más intuitiva y fácil de usar:

1. *Visibilidad del estado del sistema:* Se puede determinar que la aplicación informa de forma correcta al usuario cuando se ha completado una tarea, lanzando los mensajes de éxito correspondientes o mostrando de forma inmediata los cambios en las pantallas.

2. *Correspondencia entre el sistema y el mundo real:* La aplicación utiliza un lenguaje claro y familiar a la hora de comunicar todas las posibles acciones

que se pueden realizar en ella, aunque se debe mencionar que el texto se encuentra en inglés.

3. *Control y libertad del usuario:* En varias funciones de eliminación de datos o a la hora de sobrescribir archivos se muestra al usuario el correspondiente mensaje de confirmación, de forma que en caso de seguir adelante con la acción las consecuencias sean conocidas.

4. *Consistencia y estándares:* La posición de todos los botones y el texto de los mismos se mantiene de forma similar respecto a todas las pantallas. Las propias interfaces están diseñadas de forma similar para crear un sentimiento de consistencia a lo largo de la aplicación. En los sitios donde se emplean iconos, estos han sido elegidos de forma que sean reconocibles para la acción que presentan.

5. *Prevención de errores*: La aplicación está diseñada de forma que no se permite al usuario cometer errores, restringiendo el acceso a botones o funcionalidades que podrían ocasionarlos bajo ciertas circunstancias. Por ejemplo, si un Administrador trata de crear un cuenta de Administradores sin permisos de creación, en lugar de recibir un error de permisos no podrá darle al botón de crear en primer lugar.

6. *Reconocimiento en lugar de recuerdo:* Los elementos esenciales para cada una de las funcionalidades se encuentran siempre en la misma pantalla con la única excepción del botón de borrar todos los archivos, para el cual se puede argumentar que podría aparecer también en la pantalla de transformación por lotes.

7. *Flexibilidad y eficiencia de uso:* Al ser una aplicación de pequeño alcance, no se está haciendo uso de comandos ni atajos de teclado para facilitar el uso al usuario experto.

8. *Diseño estético y minimalista:* Cada una de las pantallas que componen la aplicación contienen la información necesaria, presentada de forma clara y centrándose en la visualización de tareas y acciones principales.

9. *Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores:* La aplicación evita mostrar errores al usuario, recogiendo las excepciones siempre que sea necesario con el fin de mostrar un mensaje de error explicativo que sirva al usuario como retroalimentación sobre lo que ha sucedido.

10. *Ayuda y documentación:* Todas las pantallas de la aplicación contienen un texto explicativo sobre el funcionamiento de la misma. Además, en los casos necesarios, se aportan enlaces externos para ampliar la información sobre ciertos elementos de la misma.

Indagación

El método de indagación consiste en la recopilación de información sobre cómo los usuarios interactúan con el diseño de la aplicación con el fin de obtener información detallada sobre la experiencia del usuario con la herramienta. Para ello, se va a hacer uso de entrevistas y cuestionarios realizados a varios usuarios, haciendo uso del estándar de Usabilidad de Jakob Nielsen que hemos definido previamente.

En primer lugar, se van a definir las 10 preguntas que se van a realizar a los usuarios coincidiendo cada de ellas con una de las heurísticas de Jakob Nielsen. Cada una de las preguntas se evaluará utilizando una escala de valoración del 1 al 5, donde 1 es

"Altamente en Desacuerdo" y 5 es “Altamente de Acuerdo". Las preguntas en cuestión son las siguientes:

1. *Visibilidad del estado del sistema:* ¿Proporciona el sistema suficiente información sobre lo que está ocurriendo en cada momento?

2. *Coincidencia entre el sistema y el mundo real:* ¿Son los mensajes y la terminología empleada fácil de comprender y coherentes con lo que esperabas de una aplicación?

3. *Control y libertad del usuario:* ¿Sentiste en algún momento que la aplicación no te detenía antes de realizar una acción que consideres delicada?

4. *Consistencia y estándares:* ¿Has notado una coherencia en la apariencia y el comportamiento de los elementos de la interfaz o en las propias pantallas a lo largo de tus interacciones?

5. *Prevención de errores:* ¿Te pareció que la interfaz te impedía de forma activa provocar errores durante la realización de las tareas?

6. *Reconocimiento en lugar de recuerdo:* ¿Has necesitado hacer uso de alguna funcionalidad en una pantalla pero no recordabas donde estaba?

7. *Flexibilidad y eficiencia de uso:* ¿Hiciste o trataste de hacer uso de atajos de teclado para optimizar tu uso de la aplicación?

8. *Estética y diseño minimalista:* ¿La interfaz te resultó visualmente agradable, fácil de usar y libre de elementos innecesarios o confusos?

9. *Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de errores:* ¿Consideras que la interfaz te proporcionó información útil para entender y resolver el problema cuando se haya producido un error?

10. *Ayuda y documentación:* ¿Tuviste algún problema haciendo uso de la aplicación por no encontrar información sobre las tareas o la documentación necesaria?

Se ha realizado un cuestionario con las preguntas establecidas a 5 usuarios. Los resultados de los mismos son los siguientes:

Resultado del Cuestionario	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5
Pregunta 1	4	3	5	4	5
Pregunta 2	3	4	4	3	5
Pregunta 3	5	4	5	4	4
Pregunta 4	5	3	5	3	4
Pregunta 5	4	3	4	5	5
Pregunta 6	5	4	5	4	5
Pregunta 7	2	3	2	1	2
Pregunta 8	5	4	5	5	4
Pregunta 9	4	5	4	3	4

Pregunta 10	3	5	4	5	5
Resultado Total	**8**	**7.6**	**8.6**	**7.4**	**8.6**

Tabla 6.1. Resultado del cuestionario.

Como se puede observar (Tabla 6.1), se ha obtenido una puntuación media de 8.04, por lo que podemos concluir que el diseño de la interfaz es lo suficientemente claro y fácil de usar como para satisfacer las necesidades del usuario promedio, siendo el principal inconveniente de la aplicación la carencia de atajos de teclado para agilizar la experiencia de usuarios expertos.

Test

En este tipo de método de usabilidad, uno o varios usuarios hacen uso de la aplicación mientras que los expertos o evaluadores analizan los resultados para ver el comportamiento de la aplicación. Se ha decidido descartar este método, puesto que genera algo de redundancia con los otros dos que ya se han empleado para las pruebas de usabilidad.

6.2 Tests Unitarios

Un test unitario es una técnica de pruebas en programación de software que se utiliza para evaluar individualmente los elementos que componen el código, que suelen ser las funciones, métodos o procedimientos más pequeños de un programa. El objetivo principal de los tests unitarios es garantizar que cada unidad de código funcione correctamente y produzca el resultado esperado, ya sea en caso de éxito o a la hora de lanzar las excepciones correspondientes.

En la siguiente tabla (Tabla 6.2) muestran los Test unitarios que se han realizado así como el resultado obtenido en los mismos:

Test Realizado	Descripción	Caso de Éxito	Resultado
Prueba de carga de la página de Inicio de Sesión.	Se accede a la página de inicio de sesión para comprobar que carga de forma correcta.	Se recibe un código de estado 200.	Satisfactorio.
Prueba de Inicio de sesión con usuario vacío.	Se intenta realizar un inicio de sesión con el nombre de usuario sin informar.	Se recibe una excepción personalizada de tipo “emptyUsernameException”.	Satisfactorio.
Prueba de Inicio de sesión con contraseña vacía.	Se intenta realizar un inicio de sesión con la contraseña sin informar.	Se recibe una excepción personalizada de tipo “emptyPasswordException”.	Satisfactorio.
Prueba de Inicio de sesión con un usuario erróneo.	Se intenta realizar un inicio de sesión con un usuario que no se encuentra en la Base de datos.	Se recibe una excepción personalizada de tipo “invalidUsernameException”.	Satisfactorio.

Prueba de inicio de sesión con una contraseña incorrecta.	Se intenta realizar un inicio de sesión con una contraseña que no coincide con la del usuario proporcionado.	Se recibe una excepción personalizada de tipo “invalidPasswordException”.	Satisfactorio.
Inicio de sesión exitoso.	Se realiza un inicio de sesión de forma exitosa usando un usuario y contraseña que se encuentren en la Base de datos.	Los parámetros de sesión usuario y contraseña coinciden con los que se han empleado durante el test.	Satisfactorio.
Prueba de carga de la página principal.	Se accede a la página principal para comprobar que carga de forma correcta.	Se recibe un código de estado 302, puesto que ha habido redirección.	Satisfactorio.
Prueba de carga de la página de Archivos Subidos.	Se accede a la página de Archivos Subidos para comprobar que carga de forma correcta.	Se recibe un código de estado 200.	Satisfactorio.

No se han seleccionado archivos para subir.	Se comprueba que la aplicación lanza una excepción al no seleccionar archivos para subir.	Se recibe una excepción personalizada de tipo “noFilesSelectedException”.	Satisfactorio.
Subir dos archivos con el mismo nombre.	Se tratan de subir dos archivos con el mismo nombre para comprobar que la aplicación lanza la excepción correspondiente.	Se recibe una excepción personalizada de tipo “duplicateElementException”.	Satisfactorio.
Borrar archivos de la base de datos.	Se accede a la funcionalidad de borrar archivos del Administrador para comprobar que el borrado es correcto.	Se recibe un código de estado 302, puesto que ha habido redirección.	Satisfactorio.
Convertir archivos.	Se accede a la funcionalidad de convertir archivos del	Se comprueba que el archivo de prueba que ha sido convertido a CSV se	Satisfactorio.

	Analista para comprobar que su funcionamiento es correcto.	encuentra en el directorio correspondiente.	
No se han seleccionado archivos para convertir.	Se comprueba que la aplicación lanza una excepción al no seleccionar archivos para convertir.	Se recibe una excepción personalizada de tipo “noFilesSelectedException”.	Satisfactorio.
Borrar archivos CSV del sistema.	Se accede a la funcionalidad de borrar archivos del Analista para comprobar que el borrado del sistema es correcto.	Se recibe un código de estado 302, puesto que ha habido redirección.	Satisfactorio.
Prueba de carga de la página de Administración de Usuarios.	Se accede a la página de Administración de Usuarios para comprobar que carga de forma correcta.	Se recibe un código de estado 200.	Satisfactorio.

Prueba de creación de Usuario.	Se crea un usuario nuevo para comprobar que funciona de forma correcta.	Se recibe un código de estado 302, puesto que ha habido redirección.	Satisfactorio.
Prueba de nombre de usuario en uso.	Se intenta crear un usuario con un nombre ya existente para comprobar que la aplicación lanza la excepción correspondiente	Se recibe una excepción personalizada de tipo "UsernameAlreadyInUseException".	Satisfactorio.
Prueba de borrado de Usuario.	Se borra un usuario para comprobar que funciona de forma correcta.	Se recibe un código de estado 302, puesto que ha habido redirección.	Satisfactorio.
Prueba de actualización de Usuario.	Se modifica un usuario para comprobar que funciona de forma correcta.	Se recibe un código de estado 302, puesto que ha habido redirección.	Satisfactorio.
Prueba de carga de la página de	Se accede a la página de Archivo Individual para	Se recibe un código de estado 200.	Satisfactorio.

Archivo Individual.	comprobar que carga de forma correcta.		
Prueba de conversión del archivo individual.	Se convierte un archivo individual a CSV para comprobar que funciona de forma correcta.	Se comprueba que el archivo de prueba que ha sido convertido a CSV se encuentra en el directorio correspondiente.	Satisfactorio.
Prueba de conversión del archivo individual sin seleccionar etiquetas.	Se trata de convertir un archivo individual a CSV sin seleccionar etiquetas para comprobar que la aplicación lanza la excepción correspondiente.	Se recibe una excepción personalizada de tipo “noTagsSelectedException”.	Satisfactorio.
Prueba de carga de la página de Archivo por Lote.	Se accede a la página de Archivo de Lote para comprobar que carga de forma correcta.	Se recibe un código de estado 200.	Satisfactorio.
Prueba de conversión	Se convierte un archivo por lote	Se comprueba que el archivo de prueba que ha sido	Satisfactorio.

del archivo por lote.	a CSV para comprobar que funciona de forma correcta.	convertido a CSV se encuentra en el directorio correspondiente.	
Prueba de carga de la página de Archivo por Lote sin seleccionar etiquetas.	Se trata de convertir un archivo individual a CSV sin seleccionar etiquetas para comprobar que la aplicación lanza la excepción correspondiente.	Se recibe una excepción personalizada de tipo “noTagsSelectedException”.	Satisfactorio.

Tabla 6.2. Resultado de los Tests Unitarios.

7

Conclusiones y Trabajos futuros

7.1 Dificultades encontradas

Aunque el desarrollo de la aplicación se considera satisfactorio, se han encontrado durante el proceso diversas limitaciones y problemas que han alterado algunas de las funcionalidades:

Limitaciones de Flask

Aunque Flask sigue siendo la mejor elección posible para el desarrollo de esta aplicación, se ha encontrado un inconveniente con el mismo que, aunque no sea impedimento crítico, sí que ha requerido replantear el cómo estaban implementadas algunas de las funcionalidades. Por cómo está diseñado Flask, para poder intercambiar información entre front-end y back-end es necesario renderizar de nuevo la interfaz de usuario. Esto produce una mala experiencia de usuario a la hora de realizar tareas que requieran entradas de usuario dentro de la misma pantalla, como podría ser ordenar una tabla.

Debido a esto, algunas funcionalidades menores han debido ser descartadas o reimaginadas completamente mientras que otras han subido de complejidad, puesto que en lugar de ser calculadas cuando el usuario las necesita, estas debían estar listas desde el momento en que la pantalla era renderizada. Aunque no se ha conseguido conciliar en todas las funcionalidades, se ha tratado de minimizar este fenómeno todo lo posible, ocurriendo tan solo en los procesos de generación de tablas de previsualización en la transformación por lotes.

Tamaño Máximo de MongoDB

MongoDB tiene un tamaño máximo de documento de 16 MB, el cual era originalmente superado por algunos archivos de Posibles Valores debido al gran tamaño que alcanzaban algunos de estos.

Esta interacción fue solucionada detectando primero que archivos eran los que estaban superando el tamaño máximo, dividirlos en múltiples partes cuyo tamaño siempre fuera aceptado por MongoDB, para guardarlos posteriormente en la Base de Datos.

Tiempos de respuesta debido al tamaño de los archivos

Debido no sólo al gran tamaño que presentan muchos de los archivos JSON, sino al gran número de ellos con los que se puede operar de forma simultánea, la rapidez de ciertas funcionalidades se ha visto comprometida. Trabajar con los archivos aplanados en lugar de con las estructuras JSON anidadas al mismo tiempo que se guarda cierta información en la Base de Datos durante la subida de los mismos soluciona gran parte de los problemas de velocidad.

Cabe destacar que la subida de archivos siempre va a ser un proceso lento cuando hay un gran número de elementos, pero se ha concretado que la solución actual es la que ofrece mayor velocidad sin tener que mejorar la herramienta de Base de Datos a otra más potente.

7.1 Posibles ampliaciones

Puesto que la aplicación estaba pensada con la escalabilidad en mente, se podrían hacer ciertas mejoras o modificaciones en caso de considerarse necesario:

Más tipos de transformación.

Por el momento, la aplicación tan sólo ofrece dos tipos de transformación a parte de la simple conversión a CSV por selección de claves. Se podría estudiar que otros tipos de transformación resultan útiles para el análisis de datos para posteriormente ser implementados.

Conversión a otros tipos de archivos.

Actualmente la aplicación sólo transforma los resultados a CSV. En caso de considerarse necesario, se podría ampliar a otros formatos o el uso de otros separadores distintos a la coma.

Sistema de registro

Tal y como se encuentra ahora mismo y por cómo fue diseñada originalmente, la única forma de tener acceso a la aplicación es mediante el alta por un Administrador. Otra alternativa sería tener un sistema de registro, con posibilidad de registrar con Google o Facebook en caso de considerarse necesario.

Archivos relacionados cuando abres un archivo.

Puesto que cada archivo JSON tiene un subconjunto de archivos con los que está relacionado (es decir, que todos provienen del mismo análisis), se podría indicar a la hora de visualizar cualquiera de ellos cuáles son estos elementos relacionados.

7.1 Objetivos cumplidos y Conclusiones

El propósito principal de este Trabajo de Fin de Grado (TFG) radicaba en desarrollar una aplicación con la capacidad de extraer y simplificar información contenida en archivos JSON, y luego transformar esta información a formato CSV mediante la aplicación de técnicas de ingeniería de datos, haciendo uso de los resultados obtenidos de diversos programas antivirus que han evaluado archivos afectados por el virus Wannacry.

En resumidas cuentas, podría decirse que el proyecto ha sido un éxito al lograr cumplir cada uno de los objetivos propuestos. A lo largo del desarrollo, se han llevado a cabo las etapas planificadas de manera coordinada y efectiva, lo que ha permitido alcanzar resultados significativos en el plazo de tiempo acordado.

El desarrollo del mismo ha sido una gran experiencia de aprendizaje, especialmente en lo referente al lenguaje de programación Python, con el que el autor no tenía prácticamente ninguna experiencia, y con el desarrollo y funcionamiento de aplicaciones Web.

No solo ha sido una buena oportunidad de aplicar varios conceptos aprendidos durante la carrera de Ingeniería Informática, como podrían ser las metodologías ágiles o los distintos tipos de diagramas, sino que también es un buen acercamiento al desarrollo de un proyecto real, habiendo permitido así al autor obtener experiencia de cara al mundo empresarial.

Referencias

[1] JSON. (2023). *Lightweight data-interchange format.* https://www.json.org/

[2] RFC 4180. (2023). *Common Format and MIME Type for Comma-Separated Values (CSV) Files.* https://tools.ietf.org/html/rfc4180

[3] Python Software Foundation. (2023). *Lists.* https://docs.python.org/3/tutorial/introduction.html#lists

[4] Scrum.org. (2023). *Scrum Framework.* https://www.scrum.org/resources/scrum-framework

[5] Pallets Projects. (2023). *Flask - A Python Microframework*. https://flask.palletsprojects.com/

[6] Python Software Foundation. (2023). *Python Programming Language.* https://www.python.org/

[7] MongoDB. (2023). *The database for modern applications.* https://www.mongodb.com/

[8] MDN Web Docs. (2023). *Recursos para desarrolladores web.* https://developer.mozilla.org/es/

[9] Nielsen, J. (2023). *10 Usability Heuristics for User Interface Design*. https://www.nngroup.com/articles/ten-usability-heuristics/

Apéndice A

Manual de Instalación

Requerimientos:

- Python3
- Pymongo
- Pandas
- Flask
- Flask_wtf
- Jinja2

Instalación:

El primer paso para realizar la instalación de la aplicación será descargar e instalar en el sistema las aplicaciones y librerías mencionadas en el apartado de requerimientos. Una vez listo, tan sólo se deberá ir al directorio que contiene el archivo app.py y ejecutar el comando "python app.py". Esto hará que el servidor Flask sea lanzado, y se podrá a acceder a la ruta "http://127.0.0.1:5000/" en el navegador web para comenzar la interacción con la herramienta.

Apéndice B

Manual de Usuario

Pantalla de Inicio de Sesión

Lo primero que se debe realizar para interaccionar con la aplicación es iniciar sesión en la misma haciendo uso de la página de Inicio de Sesión (Figura B1). Si no se tiene una cuenta de usuario, se debe esperar a que un Administrador cree una que coincida con las necesidades del usuario, ya sea Analista o Administrador. En caso de ser la primera vez que se hace uso de la aplicación, se deberá iniciar sesión con la cuenta llamada “admin”, cuya contraseña es también “admin”. Una vez creadas las cuentas necesarias, como se explicará en el apartado de “Pantalla de Administración”, se podrán introducir sus credenciales para acceder a la pantalla principal.

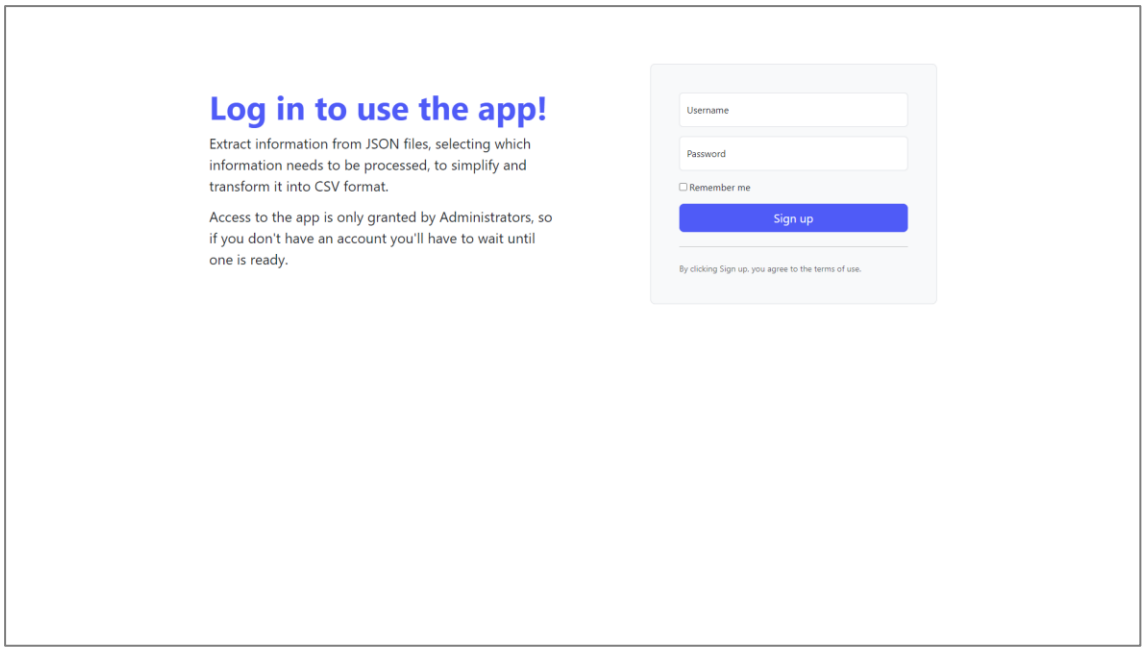

Figura B1. Pantalla de inicio de sesión.

Pantalla Principal

Desde la Pantalla Principal (Figura B2) se podrán acceder al resto de pantallas de la aplicación, ya sea haciendo uso de la barra de navegación o usando el botón de atajo "Get Started" para ir directamente a la pantalla de Archivos Subidos.

Figura B2. Pantalla Principal.

Pantalla de Administración

En la Pantalla de Administración (Figura B3), se podrán crear, borrar o modificar las distintas cuentas de usuario, siempre que se tengan los permisos necesarios. Si es la primera vez que se usa la aplicación, se recomienda, al entrar con el usuario "admin", crear otro usuario administrador que tenga todos los permisos y usar dicho usuario en lugar del predeterminado.

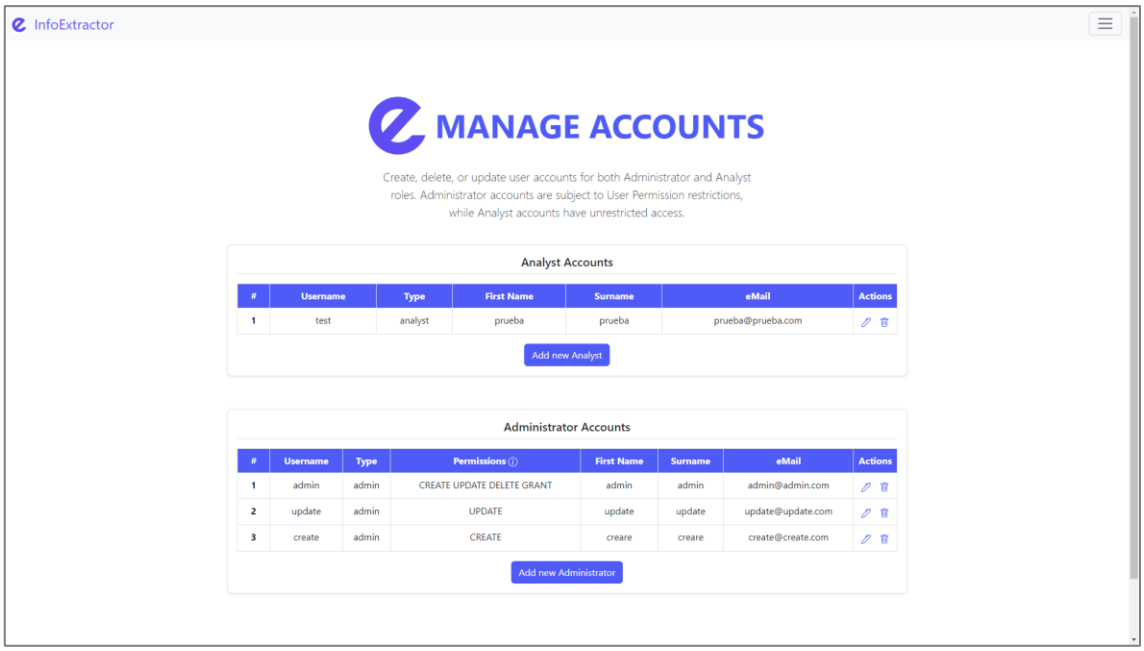

Figura B3. Pantalla de Administración de Cuentas.

Se podrá borrar un usuario haciendo clic en el botón de la papelera al lado de su nombre y se podrán modificar sus datos haciendo clic en el botón contiguo con el icono del bolígrafo. Para la creación o modificación de un usuario, bastará con rellenar los campos dentro de la pantalla correspondiente (Figura B4).

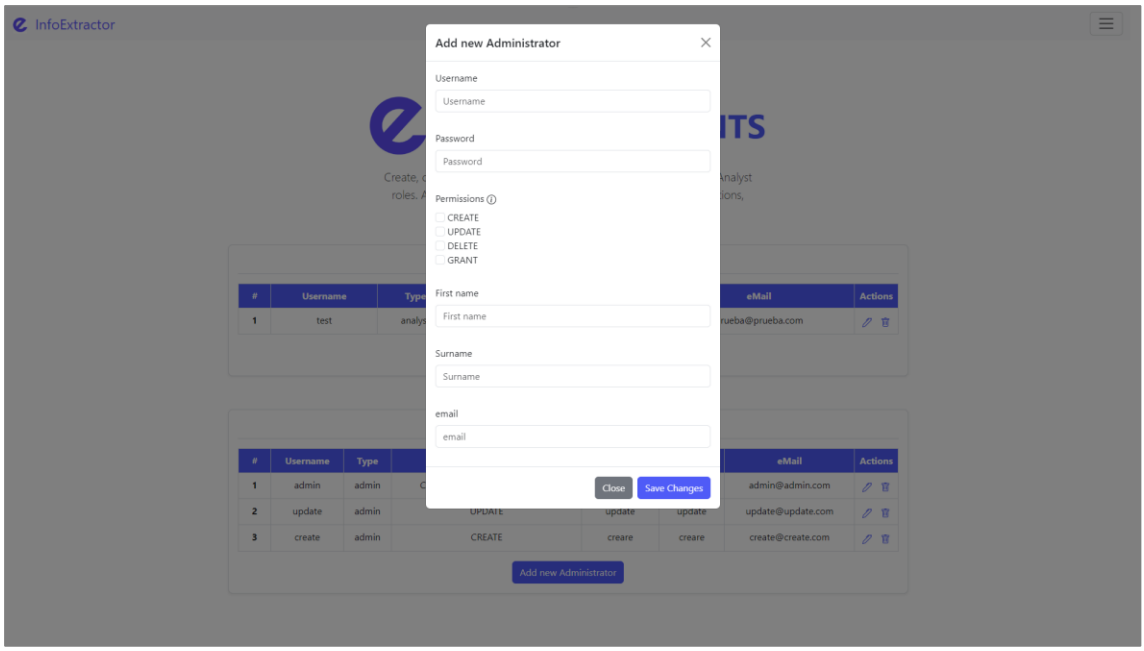

Figura B4. Creación de un usuario Administrador.

Pantalla de Archivos Subidos - Administrador

La pantalla de Archivos Subidos (Figura B5) sirve tanto para mostrar los archivos que se encuentran actualmente en la Base de datos como para subir nuevos archivos en caso de ser un Administrador. Para ello, simplemente se ha de hacer clic en el botón “Upload Files”, seleccionar los archivos que se quieren subir, y después darle al botón “Subir”. Si todo ha salido bien, se verá una lista en esta misma página la lista con los archivos que se han subido, siendo los que están bajo el mismo desplegable archivos que provienen del resultado del mismo análisis.

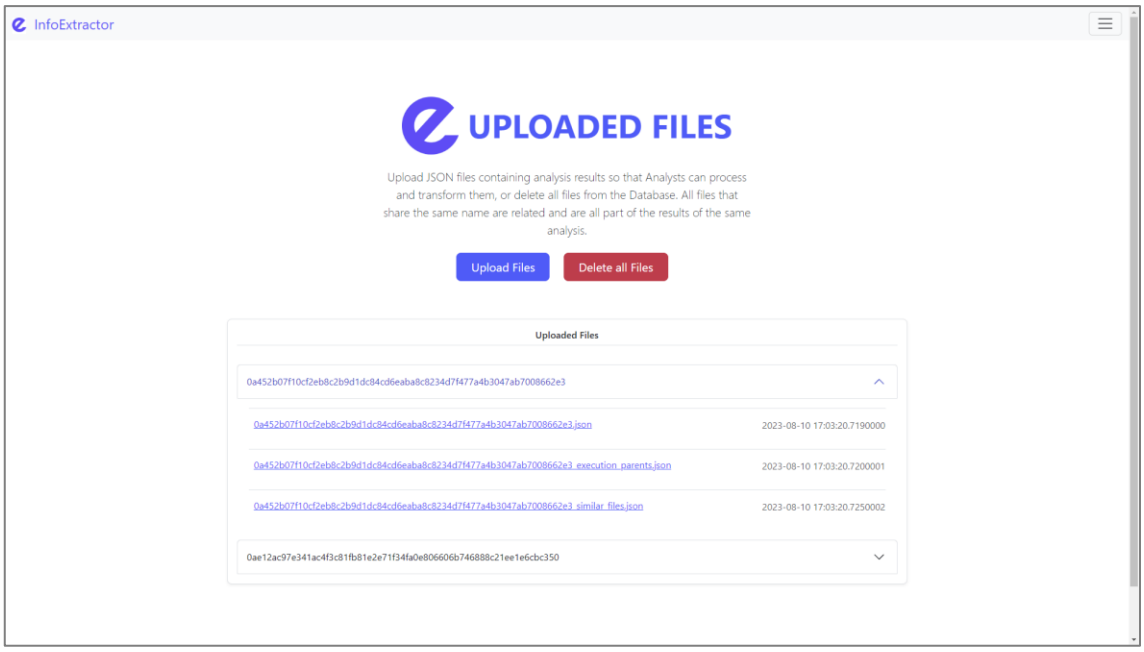

Figura B5. Pantalla de Archivos Subidos del Administrador.

Se puede hacer clic en cualquiera de estos archivos para ir a la Pantalla de Archivo Individual. También se pueden borrar todos los archivos que se encuentran subidos en la Base de Datos haciendo clic en el botón “Delete All Files”.

Pantalla de Archivos Subidos - Analista

Similar a la visión del Administrador, solo que en vez de la funcionalidad de subir archivos en la versión del Analista (Figura B6) se podrán seleccionar de entre los que se encuentran en la base de datos para transformarlos a CSV en su totalidad. Se puede seleccionar un desplegable entero para convertir todos los archivos que se encuentran en su interior pulsando el botón “Convert Files”. También se pueden borrar todos los archivos convertidos de la ruta de guardado de la aplicación haciendo clic en el botón “Delete All Files”.

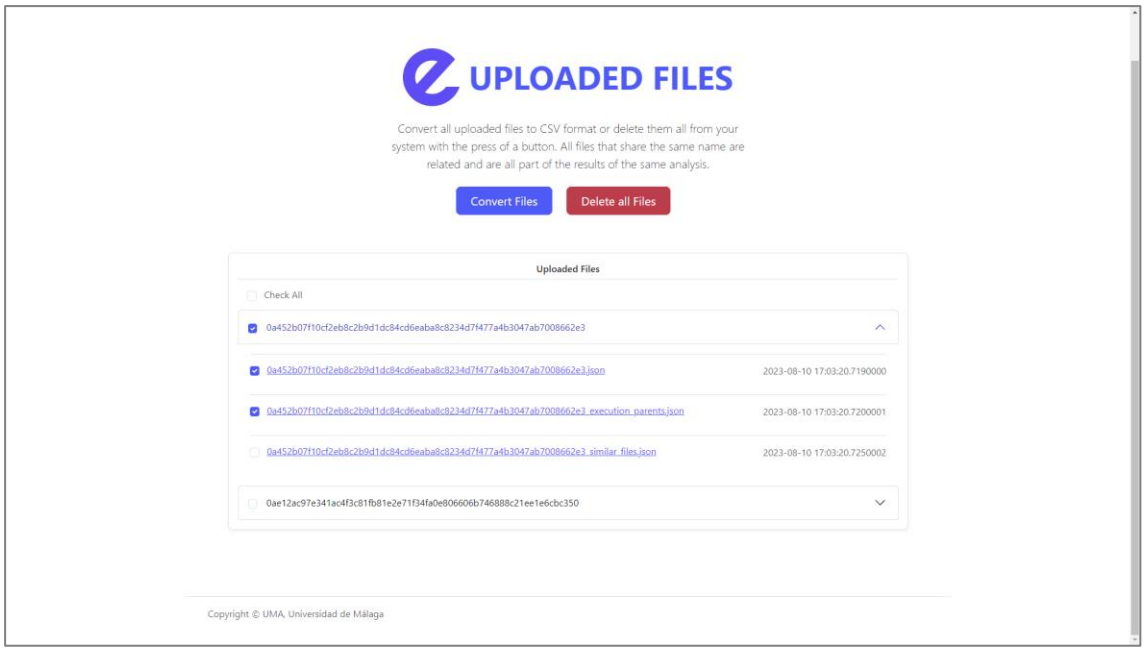

Figura B6. Pantalla de Archivos Subidos del Analista.

Pantalla de Archivo Individual.

Esta pantalla (Figura B7) ofrece una visión del archivo al que se ha accedido de forma estructurada, simulando su implementación en formato JSON. En caso de ser un analista, se podrán seleccionar los datos que se quieran convertir a CSV haciendo clic en los checkboxes. Seleccionar los que se encuentran en un desplegable hará que también se seleccionen todos los elementos que hay dentro del mismo. Una vez seleccionada toda la información necesaria, pulsar el botón “Convert to CSV By Tags” transformará el archivo a formato CSV y lo guardará en la ruta correspondiente de la aplicación.

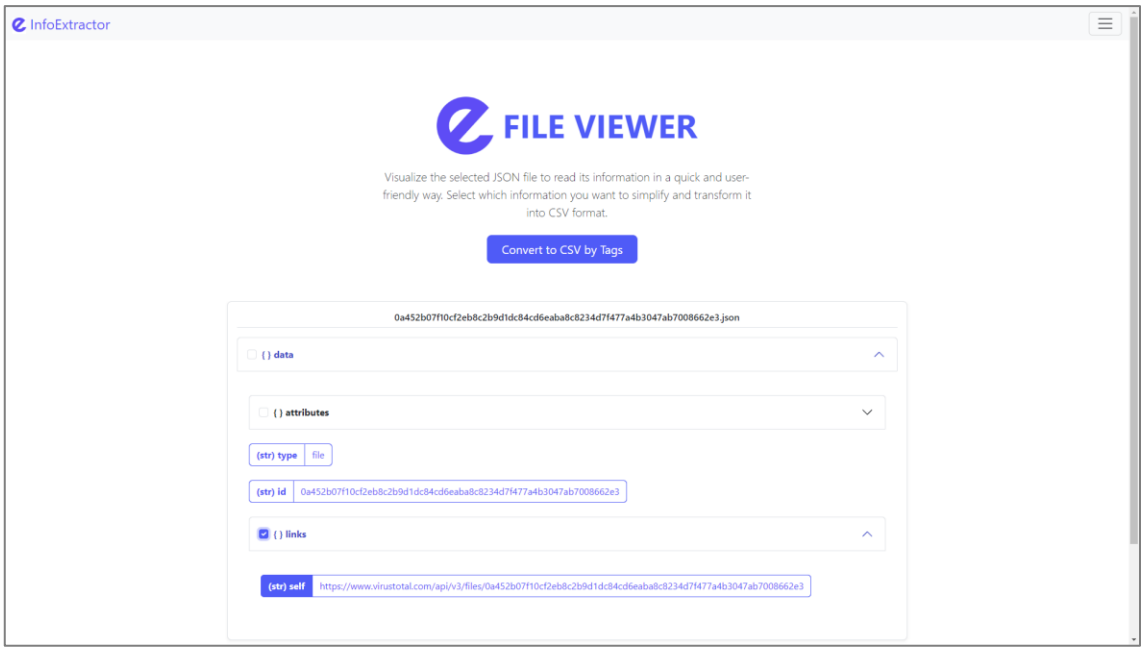

Figura B7. Pantalla

Pantalla de Lotes

En esta pantalla (Figura B8) se muestran los distintos tipos de archivos. Hacer clic en el icono de la columna “To CSV” redireccionará a la página de Subida por Lotes de dicho archivo.

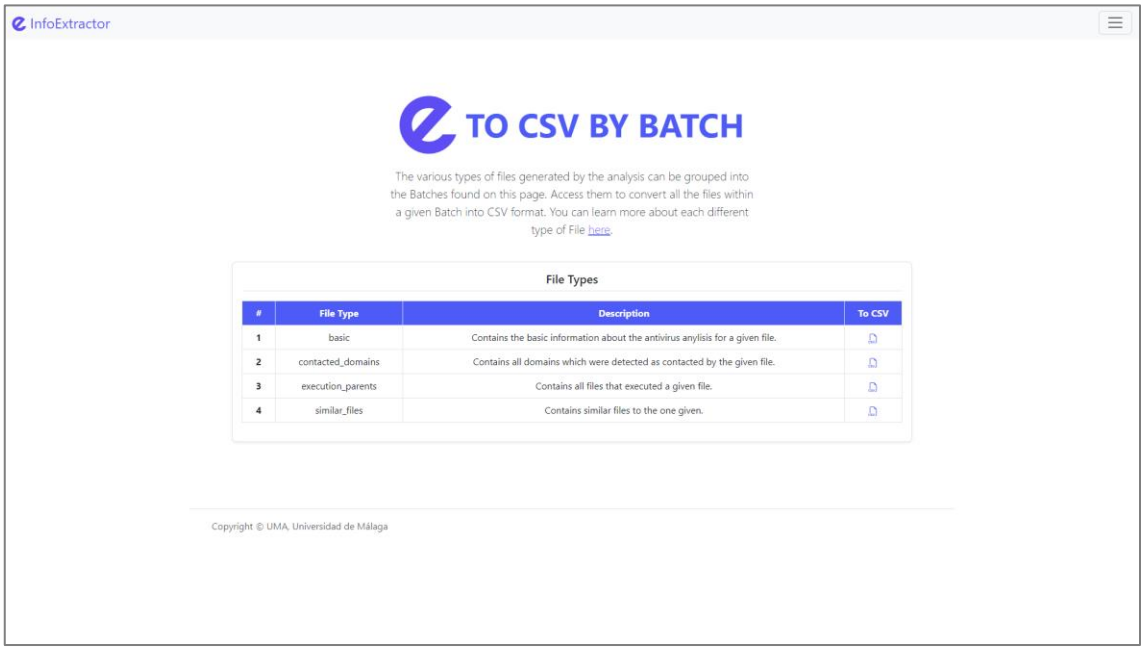

Figura B8. Pantalla Selección de Lote.

Pantalla de Subida por Lotes

En esta pantalla (Figura B9) se muestra una representación JSON de todos los archivos que pertenezcan al tipo de archivo que se ha seleccionado. Al igual que en la conversión que se realiza en la pantalla de "Archivo Individual", se podrá seleccionar la información necesaria para la conversión.

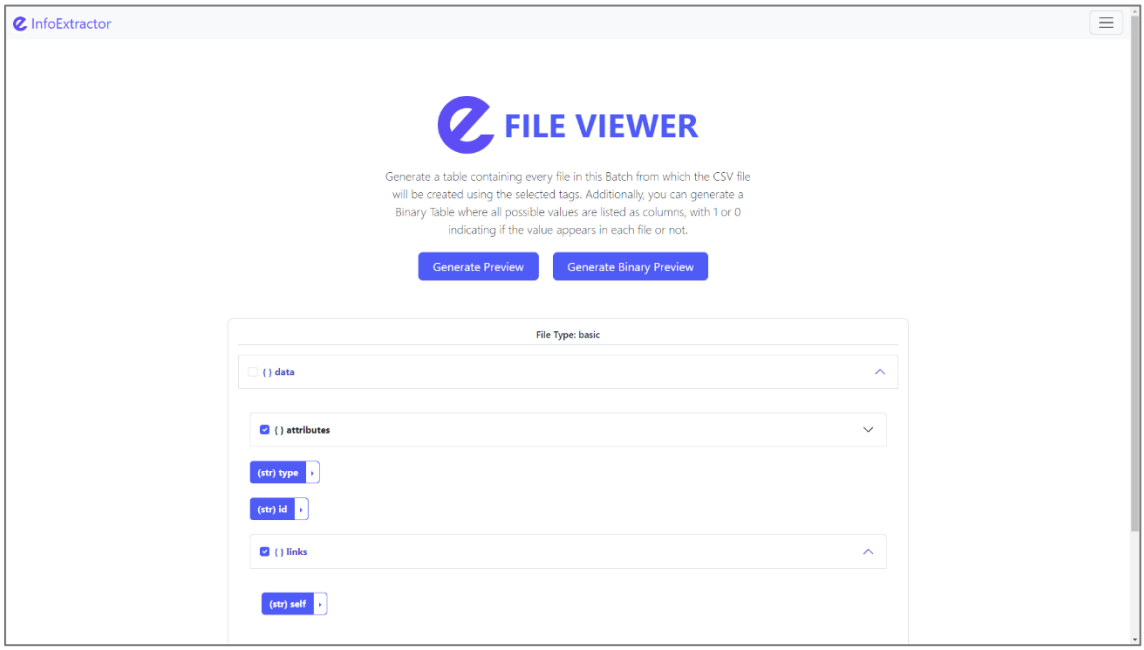

Figura B9. Pantalla de Transformación por Lote

Hacer clic en el botón "Generate Preview" o "Generate Binary Preview" actualizará la página para mostrar una tabla que contenga los datos seleccionados por el usuario (Figuras B10 y B11).

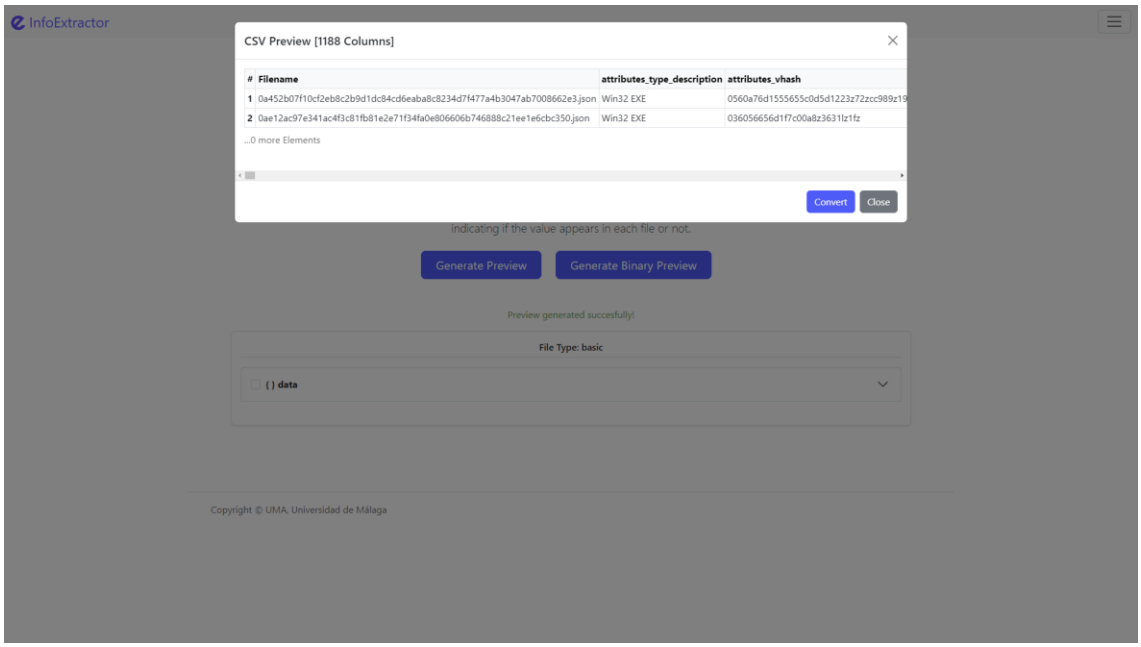

Figura B10. Tabla de Previsualización.

Hacer clic en el botón “Convert” generará el archivo CSV que se corresponde con la tabla que se ha mostrado anteriormente.

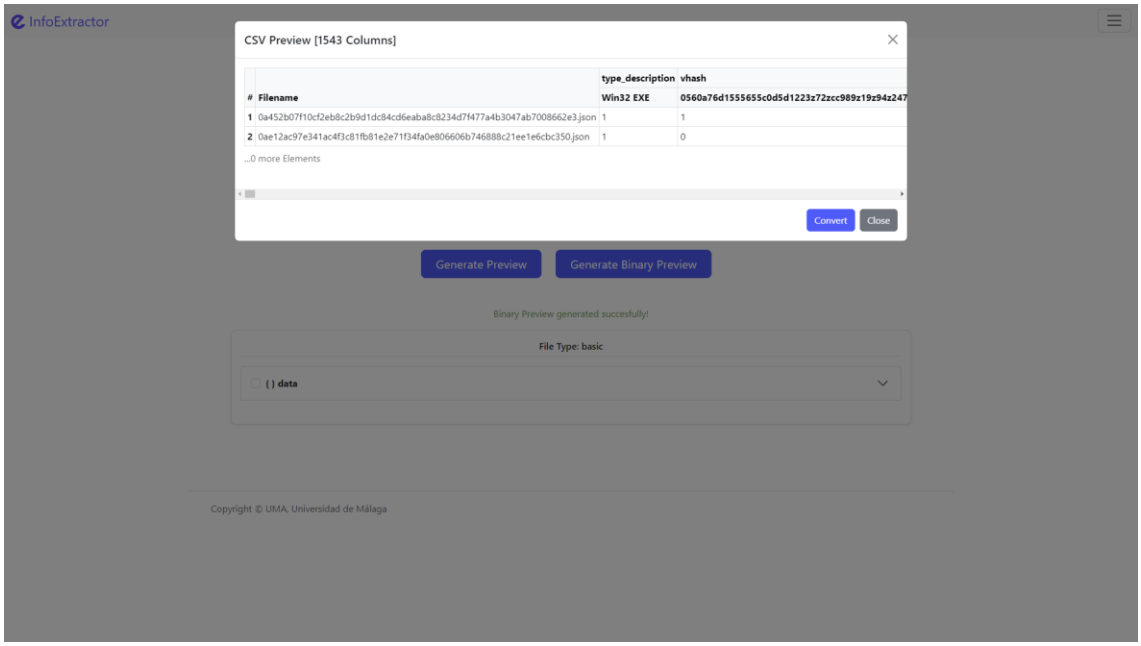

Figura B11. Tabla Binaria de Previsualización.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA