Manual de MetaInfo

Equipo de Desarrollo

01/05/2025

Contents

1 MetaInfo

1.1 Descripción General

MetaInfo es una aplicación especializada en el análisis y gestión de metadatos en archivos digitales. Permite detectar información sensible, generar informes detallados y limpiar metadatos para proteger la privacidad.

1.2 Navegación Rápida

Documentación Principal	Diagramas	Técnica
Índice Principal	Diagrama de Componentes	Actualizaciones Recientes
Casos de Uso	Diagrama de Clases	Instalación y Dependencias
Modelo de Dominio	Diagramas de Interacción	Ejemplos de Uso

1.3 Actualizaciones Recientes

1.3.1 Mejoras en Limpieza de Metadatos

- Exclusión automática de archivos `.txt` durante el proceso de limpieza
- Método de limpieza mejorado con múltiples estrategias y verificación
- Manejo de errores robusto con recuperación automática

1.3.2 Mejoras en Generación de Informes

- Rutas relativas en informes para mejor portabilidad y privacidad
- Mejor identificación y presentación de patrones sensibles

1.3.3 Mejoras en Pruebas

- Pruebas unitarias más claras y enfocadas

- Pruebas de integración más robustas y confiables

1.4 Instalación y Dependencias

Para utilizar MetaInfo, necesitas:

```
# Instalación básica
pip install -r requirements.txt

# Dependencias opcionales para generación de PDF
apt-get install pandoc texlive-xetex
```

Dependencias principales: - Python 3.7+ - ExifTool - Pandoc (opcional, para PDF)

1.5 Ejemplos de Uso

1.5.1 Analizar metadatos y generar informe completo

```
python metainfo.py --i ~/Documentos --report_all --o ~/Informes --pdf
```

1.5.2 Limpiar metadatos sensibles

```
python metainfo.py --i ~/Fotos/Privadas --wipe_sensitive --verbose
```

1.5.3 Eliminar todos los metadatos

```
python metainfo.py --i ~/Documentos/ParaPublicar --wipe_all
```

1.6 Estructura de la Documentación

La documentación sigue un enfoque progresivo, desde aspectos conceptuales hasta detalles de implementación:

1. **Índice** - Punto de entrada principal a toda la documentación
2. **Casos de Uso** - Escenarios de interacción usuario-sistema
3. **Diagrama de Componentes** - Estructura de alto nivel
4. **Diagrama de Clases** - Detalles de implementación

Para más información, consulte el Índice Principal.

Última actualización: 11/06/2024 # Índice de la Documentación de MetaInfo

1.7 Navegación Rápida

- **Actualizaciones Recientes** - Últimas mejoras implementadas
- **Índice Técnico** - Guía técnica detallada
- **Casos de Uso** - Escenarios de interacción usuario-sistema
- **Arquitectura** - Documentos sobre la estructura del sistema
- **Diagramas** - Representaciones visuales del sistema

1.8 Introducción

MetaInfo es una aplicación especializada en el análisis y gestión de metadatos en archivos digitales. Esta documentación está estructurada para proporcionar una comprensión progresiva del sistema, sus fundamentos teóricos y su implementación práctica.

1.9 Documentos de Arquitectura

- **Modelo de Dominio** - Conceptos clave y relaciones
- **Arquitectura por Capas** - Estructura general del sistema
- **Arquitectura Modular** - Componentes y sus interacciones
- **Diagrama de Componentes** - Visión de alto nivel de los módulos del sistema

1.10 Documentos de Diagramas

- **Diagrama de Clases** - Diseño estático y relaciones entre clases
- **Diagramas de Interacción** - Comportamiento dinámico del sistema
- **Diagrama de Flujo de Datos** - Procesamiento de información

1.11 Metodología de Desarrollo

La documentación sigue los principios del **Proceso Unificado (UP)** utilizando notación **UML**:

- **Dirigido por casos de uso**: La arquitectura se centra en resolver escenarios de uso específicos
- **Centrado en la arquitectura**: Estructura modular con separación clara de responsabilidades
- **Iterativo e incremental**: Desarrollo por fases con mejoras continuas
- **Dirigido por riesgos**: Análisis temprano de riesgos técnicos y de negocio
- **Verificación continua de la calidad**: Pruebas unitarias e integración automatizadas

1.12 Aspectos Funcionales

1.12.1 Gestión de Metadatos

MetaInfo ofrece capacidades para:

- **Detectar y extraer** metadatos de diversos tipos de archivos
- **Identificar información sensible** mediante patrones predefinidos y personalizables
- **Generar informes detallados** en formatos Markdown, HTML y PDF con rutas relativas
- **Limpiar selectivamente** metadatos que contienen información sensible
- **Manejo inteligente de archivos** omitiendo formatos sin metadatos relevantes
- **Recuperación de errores robusta** con múltiples estrategias de limpieza

1.12.2 Extensibilidad

El sistema está diseñado para facilitar:

- La incorporación de **nuevos formatos de archivo**
- La definición de **patrones adicionales** para detectar información sensible
- La implementación de **nuevos formatos de informe**
- La integración de **estrategias alternativas de limpieza**
- La exclusión de **tipos de archivo específicos** del procesamiento

1.13 Guía de Estudio Recomendada

Para comprender el sistema de manera efectiva, se recomienda seguir este orden:

1. Primero: **Actualizaciones Recientes** y **Modelo de Dominio**
2. Segundo: **Casos de Uso** y **Arquitectura por Capas**
3. Tercero: **Diagrama de Componentes** y **Diagrama de Clases**
4. Finalmente: **Diagramas de Interacción** y **Diagrama de Flujo de Datos**

Última actualización: 11/06/2024 # Índice de Documentación Técnica

1.14 Introducción

Este documento sirve como índice principal para la documentación técnica de MetaInfo, una herramienta especializada en el análisis y gestión de metadatos en archivos digitales. El índice está organizado siguiendo un enfoque metodológico que facilita la comprensión progresiva del sistema, desde sus aspectos conceptuales hasta los detalles de implementación.

1.15 Metodología de Desarrollo

Esta documentación técnica está estructurada siguiendo los principios del **Proceso Unificado (UP)** con notación **UML** (Unified Modeling Language), e implementa los patrones de diseño **GRASP** (General Responsibility Assignment Software Patterns) y **GoF** (Gang of Four) para garantizar un diseño orientado a objetos de alta calidad.

1.15.1 Contexto de Aplicación

MetaInfo es una herramienta especializada para el análisis y gestión de metadatos en archivos digitales, con enfoque particular en la identificación y tratamiento de información sensible. La implementación sigue un enfoque modular que separa claramente las responsabilidades entre:

- Extracción de metadatos
- Análisis de patrones sensibles
- Generación de informes en múltiples formatos
- Limpieza selectiva de metadatos

La arquitectura aprovecha los principios GRASP (Creador, Controlador, Polimorfismo, Alta Cohesión y Bajo Acoplamiento) para distribuir responsabilidades, y patrones GoF (principalmente Factory, Strategy, Template Method y Observer) para resolver problemas recurrentes de diseño.

1.16 Itinerario de Estudio Recomendado

Para comprender completamente la arquitectura y funcionamiento de MetaInfo, se recomienda seguir este itinerario de estudio:

1. **Modelo de Dominio**
   - Visión conceptual de las entidades principales
   - Relaciones entre conceptos clave
   - Glosario de términos técnicos específicos
2. **Casos de Uso**
   - Escenarios de uso principales
   - Flujos de interacción usuario-sistema
   - Precondiciones y postcondiciones
3. **Arquitectura por Capas**
   - Estructura general de la aplicación
   - Separación de responsabilidades
   - Patrones de diseño arquitectónicos
4. **Diagrama de Clases**
   - Estructura estática del sistema
   - Aplicación de patrones GRASP
   - Implementación de patrones GoF
5. **Diagramas de Interacción**
   - Diagramas de secuencia para operaciones clave
   - Diagramas de colaboración
   - Comportamiento dinámico del sistema
6. **Arquitectura Modular**
   - Componentes principales
   - Mecanismos de extensibilidad

- Gestión de dependencias

7. **Diagrama de Flujo de Datos**
   - Procesamiento de metadatos
   - Flujos de información
   - Puntos de integración

1.17 Aspectos Funcionales Destacados

1.17.1 Gestión de Metadatos

La aplicación permite la detección, análisis y manipulación de metadatos en diversos tipos de archivos, centrándose en:

- **Extracción de metadatos**: Identificación y extracción de metadatos de múltiples formatos de archivo mediante herramientas especializadas
- **Detección de información sensible**: Identificación de patrones sensibles en los metadatos mediante expresiones regulares y análisis heurístico
- **Limpieza selectiva**: Capacidad para eliminar solo los metadatos que contienen información sensible
- **Generación de informes**: Creación de reportes detallados en formatos Markdown, HTML y PDF

1.17.2 Extensibilidad del Sistema

El diseño modular facilita la incorporación de:

- Nuevos formatos de archivo para análisis
- Patrones adicionales para la detección de información sensible
- Formatos de informe personalizados
- Estrategias alternativas de limpieza de metadatos

1.18 Patrones GRASP Implementados

- **Creador**: Asignación de responsabilidades de creación a clases específicas (ej: `Main` crea instancias de `Reporter`).
- **Controlador**: Centralización de la lógica de control en clases como `MetaInfo` y `Main`.
- **Alta Cohesión**: Agrupación de funcionalidades relacionadas (ej: separación entre `Messages` y `ParameterValidator`).
- **Bajo Acoplamiento**: Minimización de dependencias entre módulos.
- **Polimorfismo**: Implementación de interfaces comunes para comportamientos variables.

1.19 Patrones GoF Implementados

- **Factory Method**: Creación centralizada de informes en distintos formatos.
- **Strategy**: Diferentes estrategias para el procesamiento de metadatos según el tipo de archivo.
- **Template Method**: Estructura común para la generación de informes con pasos personalizables.
- **Observer**: Notificaciones durante el proceso de análisis y limpieza.
- **Singleton**: Utilizado para garantizar que solo existe una instancia de ciertas clases, como el gestor de mensajes.
- **Facade**: La clase `Main` proporciona una interfaz simplificada al subsistema complejo de gestión de metadatos.

1.20 Principios de Diseño Adicionales

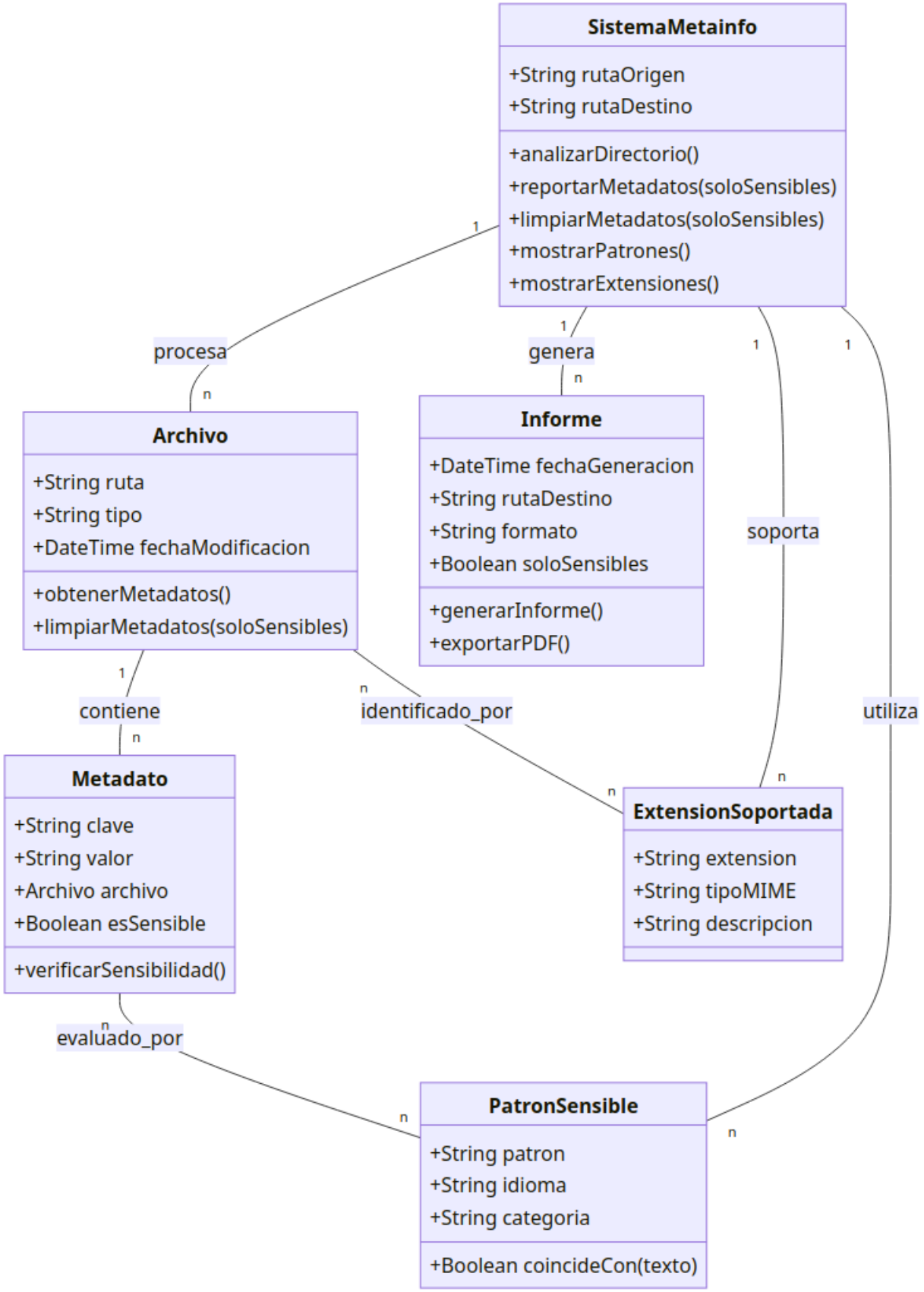
- **Principio de Responsabilidad Única (SRP)**: Cada clase tiene una única razón para cambiar.
- **Principio Abierto/Cerrado (OCP)**: El sistema está abierto a extensiones pero cerrado a modificaciones.
- **Inyección de Dependencias**: Reducción del acoplamiento mediante paso de dependencias.

- **Programación por Contrato**: Precondiciones y postcondiciones definidas para operaciones clave.

La aplicación de estos principios y patrones ha permitido crear una herramienta robusta, extensible y mantenible que cumple con los estándares de la ingeniería de software moderna. Este enfoque facilita la comprensión del código, la corrección de errores y la incorporación de nuevas funcionalidades. # Modelo de Dominio de Metainfo

Este documento presenta el modelo de dominio de la aplicación Metainfo, mostrando las entidades principales y sus relaciones.

1.21 Diagrama de Modelo de Dominio



1.22 Explicación de las Entidades

1.22.1 Archivo

Representa un archivo físico en el sistema que puede ser procesado por Metainfo.

1.22.2 Metadato

Representa un par clave-valor de metadatos extraídos de un archivo.

1.22.3 PatronSensible

Define los patrones que se utilizan para identificar información potencialmente sensible.

1.22.4 ExtensionSoportada

Define las extensiones de archivo que son soportadas por la aplicación.

1.22.5 Informe

Representa un informe generado con los metadatos analizados.

1.22.6 SistemaMetainfo

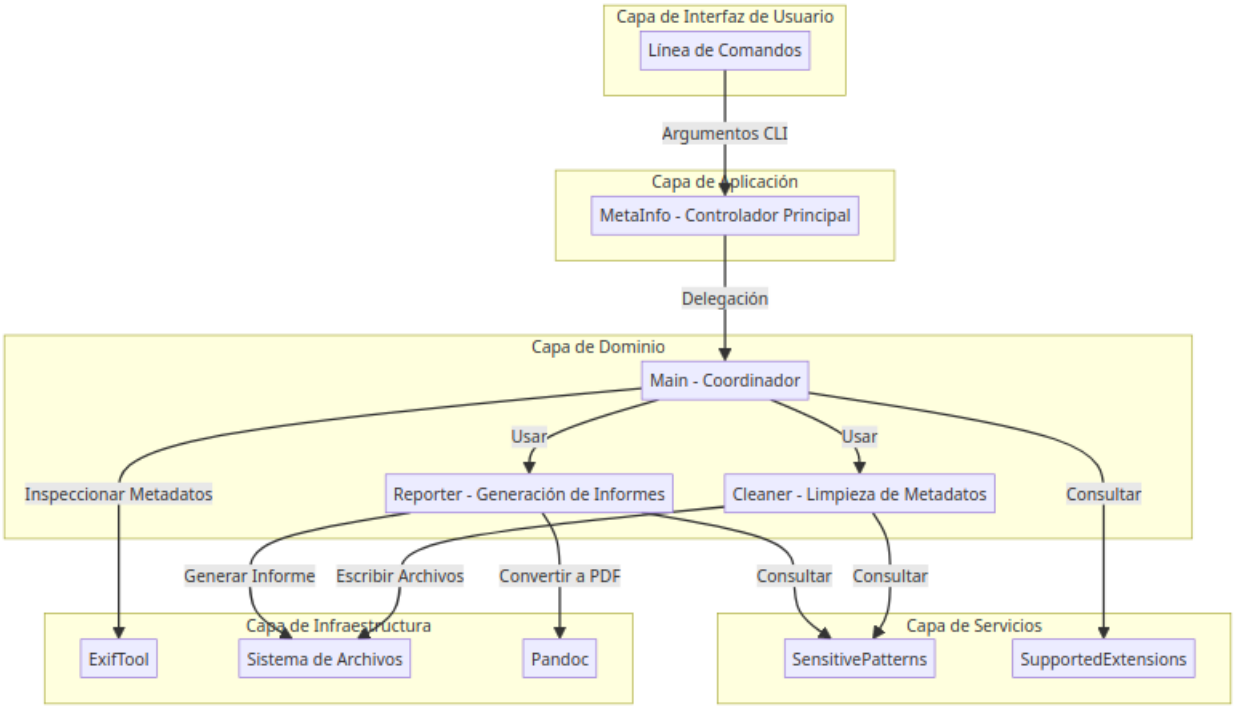
Representa el sistema completo que coordina todas las operaciones.

1.23 Relaciones Principales

- Un **Archivo** contiene múltiples **Metadatos**.
- Cada **Metadato** es evaluado por múltiples **PatronesSensibles** para determinar si contiene información sensible.
- Un **Archivo** es identificado por su **ExtensionSoportada**.
- El **SistemaMetainfo** procesa múltiples **Archivos**.
- El **SistemaMetainfo** genera múltiples **Informes**.
- El **SistemaMetainfo** utiliza **PatronesSensibles** y soporta **ExtensionesCompatibles**. # Arquitectura de Capas de Metainfo

Este documento presenta la arquitectura por capas de la aplicación Metainfo.

1.24 Diagrama de Arquitectura por Capas



1.25 Descripción de las Capas

1.25.1 Capa de Interfaz de Usuario

- **Línea de Comandos**: Proporciona la interfaz para interactuar con la aplicación mediante argumentos CLI.

1.25.2 Capa de Aplicación

- **MetaInfo**: Actúa como controlador principal de la aplicación, procesando argumentos y delegando a las clases de dominio.

1.25.3 Capa de Dominio

- **Main**: Coordina las operaciones principales, dirigiendo el flujo de trabajo entre componentes.
- **Reporter**: Responsable de generar informes con los metadatos analizados.
- **Cleaner**: Responsable de limpiar metadatos sensibles o todos los metadatos.

1.25.4 Capa de Servicios

- **SensitivePatterns**: Define y gestiona los patrones considerados sensibles.
- **SupportedExtensions**: Define y gestiona las extensiones de archivo soportadas.

1.25.5 Capa de Infraestructura

- **ExifTool**: Herramienta externa utilizada para extraer y manipular metadatos.
- **Sistema de Archivos**: Acceso a archivos y directorios para lectura y escritura.
- **Pandoc**: Herramienta externa para la conversión de documentos (Markdown a PDF).

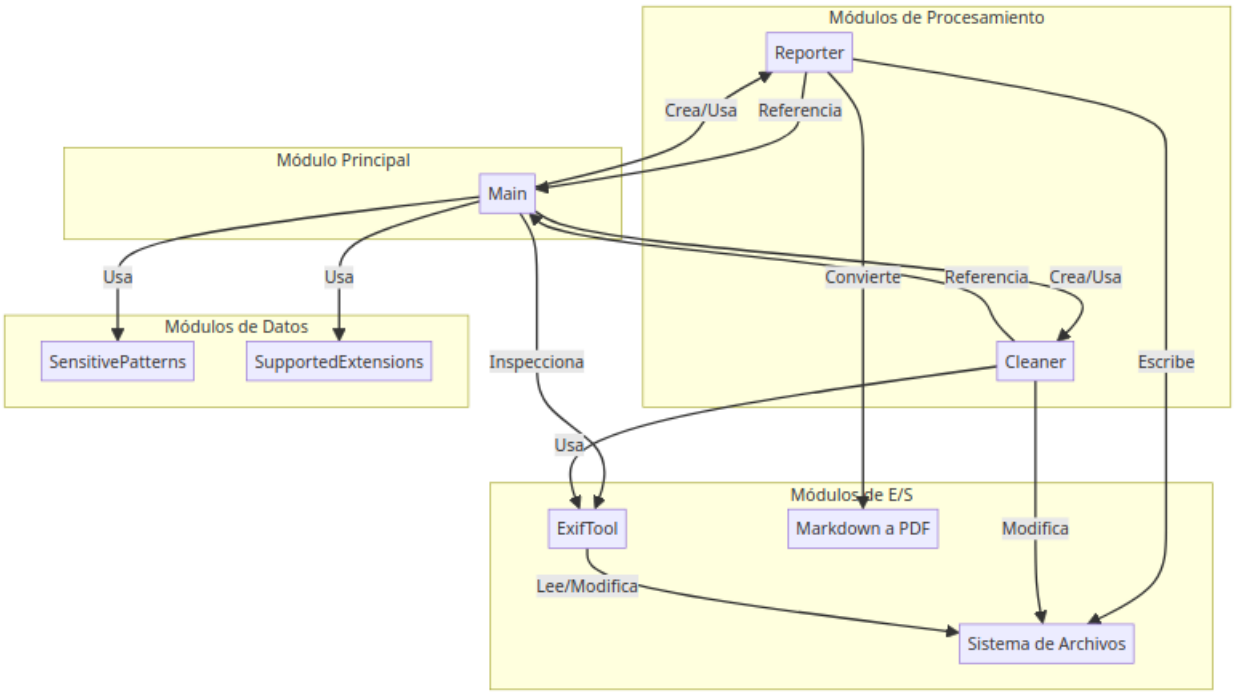
1.26 Flujo de Información

1. Los argumentos del usuario se reciben a través de la línea de comandos.
2. El controlador principal (MetaInfo) procesa estos argumentos y configura el entorno.
3. Las capas de dominio implementan la lógica principal, coordinando las operaciones solicitadas.
4. La capa de servicios proporciona funcionalidades especializadas.
5. La capa de infraestructura gestiona el acceso a herramientas y recursos externos.

Esta arquitectura por capas facilita la separación de responsabilidades y mejora la mantenibilidad del sistema.
Arquitectura Modular del Sistema MetaInfo

Este documento presenta la arquitectura modular del sistema MetaInfo, mostrando la organización de sus componentes y las interacciones entre ellos.

1.27 Diagrama de Arquitectura Modular



1.28 Descripción de la Arquitectura Modular

1.28.1 Módulo Principal

- **Main**: Componente central que coordina todas las operaciones del sistema. Actúa como punto de entrada y controlador principal.

1.28.2 Módulos de Procesamiento

- **Reporter**: Responsable de la generación de informes de metadatos. Transforma los datos recopilados en formatos legibles (Markdown, PDF).
- **Cleaner**: Encargado de la limpieza de metadatos. Puede eliminar todos los metadatos o solo aquellos considerados sensibles.

1.28.3 Módulos de Datos

- **SensitivePatterns**: Define los patrones considerados sensibles para la detección en metadatos.

- **SupportedExtensions**: Especifica las extensiones de archivo que el sistema puede procesar.

1.28.4 Módulos de Entrada/Salida

- **Sistema de Archivos**: Gestiona la lectura y escritura de archivos y directorios.
- **ExifTool**: Herramienta externa utilizada para la extracción y manipulación de metadatos.
- **Markdown a PDF**: Funcionalidad para convertir los informes de Markdown a PDF.

1.29 Características de la Arquitectura

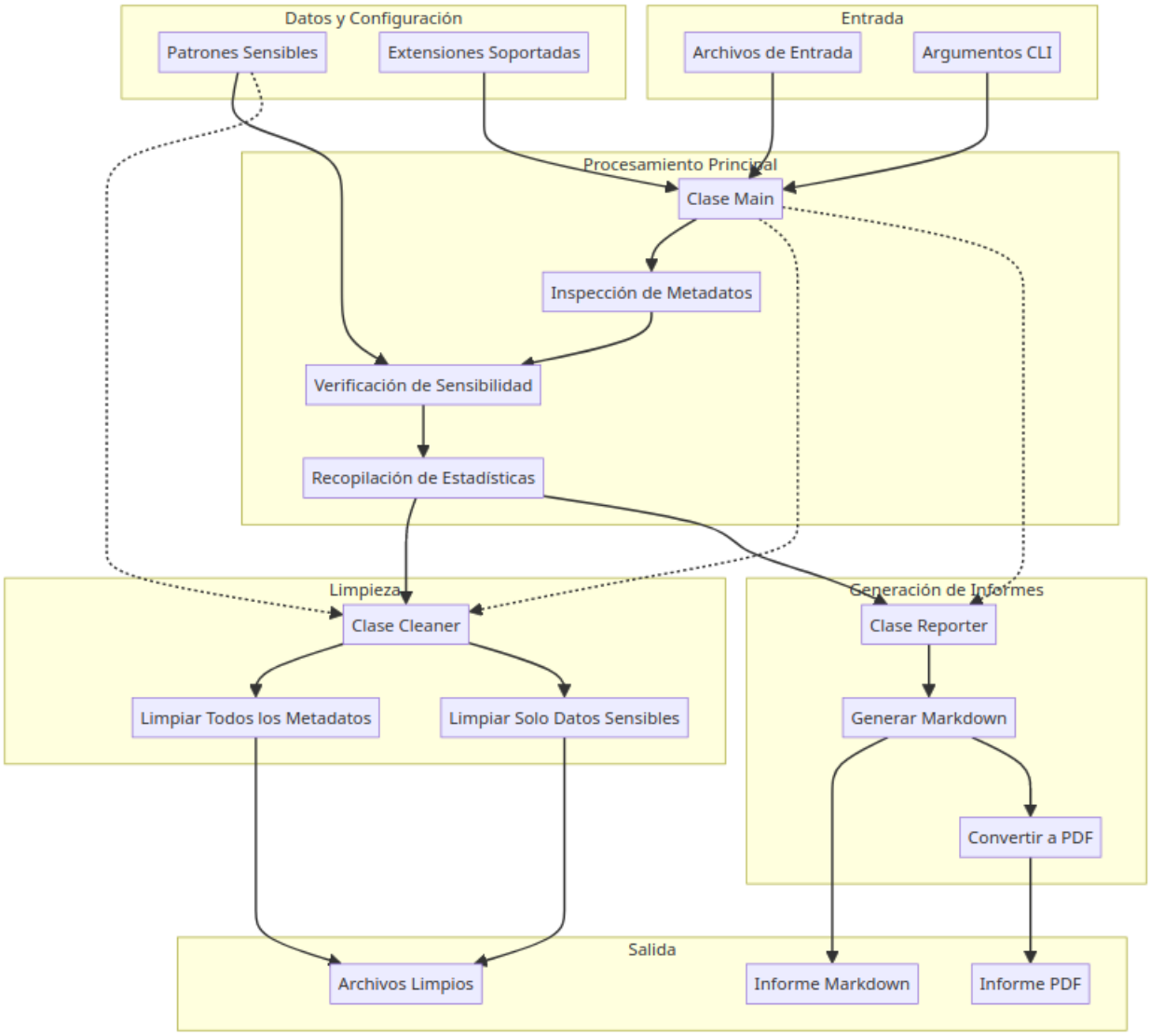
1. **Diseño Modular**: La aplicación está organizada en módulos con responsabilidades claramente definidas.
2. **Alta Cohesión**: Cada módulo agrupa funcionalidades relacionadas, maximizando la cohesión interna.
3. **Bajo Acoplamiento**: Los módulos se comunican a través de interfaces bien definidas, minimizando las dependencias directas.
4. **Separación de Responsabilidades**: Cada módulo tiene una responsabilidad única y bien definida.
5. **Reutilización de Código**: Los módulos de datos (SensitivePatterns, SupportedExtensions) son utilizados por múltiples componentes.
6. **Integración con Herramientas Externas**: La arquitectura incorpora herramientas externas (ExifTool) de manera modular, facilitando posibles reemplazos o actualizaciones.

1.30 Ventajas de esta Arquitectura

- **Facilidad de Mantenimiento**: La organización modular facilita la localización y corrección de errores.
- **Escalabilidad**: Nuevas funcionalidades pueden agregarse como módulos adicionales sin afectar significativamente el sistema existente.
- **Testabilidad**: Cada módulo puede probarse de forma independiente, facilitando la implementación de pruebas unitarias.
- **Flexibilidad**: Los componentes pueden evolucionar de forma independiente, siempre que mantengan sus interfaces.
- **Claridad**: La estructura modular mejora la comprensión del sistema para nuevos desarrolladores. # Diagrama de Flujo de Datos - MetaInfo

Este documento presenta el flujo de datos en la aplicación MetaInfo, mostrando cómo se procesa la información desde su entrada hasta la generación de informes o limpieza de metadatos.

1.31 Diagrama de Flujo de Datos



1.32 Explicación del Flujo de Datos

1.32.1 Entrada

- **Archivos de Entrada**: Los archivos y directorios que serán procesados por la aplicación.
- **Argumentos CLI**: Los parámetros proporcionados por el usuario a través de la línea de comandos.

1.32.2 Procesamiento Principal

- **Clase Main**: Coordina todo el flujo de trabajo.
- **Inspección de Metadatos**: Extrae los metadatos de cada archivo usando ExifTool.
- **Verificación de Sensibilidad**: Comprueba si los metadatos contienen información sensible.
- **Recopilación de Estadísticas**: Agrupa y procesa los metadatos para generar estadísticas.

1.32.3 Datos y Configuración

- **Patrones Sensibles**: Lista de patrones que se consideran sensibles.

- **Extensiones Soportadas**: Lista de extensiones de archivo que pueden ser procesadas.

1.32.4 Bifurcación del Flujo

Después del procesamiento principal, el flujo se divide según la operación solicitada:

1.32.4.1 Ruta de Limpieza

- **Clase Cleaner**: Gestiona el proceso de limpieza.
- **Limpiar Todos los Metadatos**: Elimina todos los metadatos de los archivos.
- **Limpiar Solo Datos Sensibles**: Elimina solo los metadatos considerados sensibles.
- **Archivos Limpios**: Los archivos resultantes después del proceso de limpieza.

1.32.4.2 Ruta de Reporte

- **Clase Reporter**: Gestiona la generación de informes.
- **Generar Markdown**: Crea un informe en formato Markdown.
- **Convertir a PDF**: Opcionalmente convierte el informe Markdown a PDF.
- **Informe Markdown**: El informe generado en formato Markdown.
- **Informe PDF**: El informe generado en formato PDF (opcional).

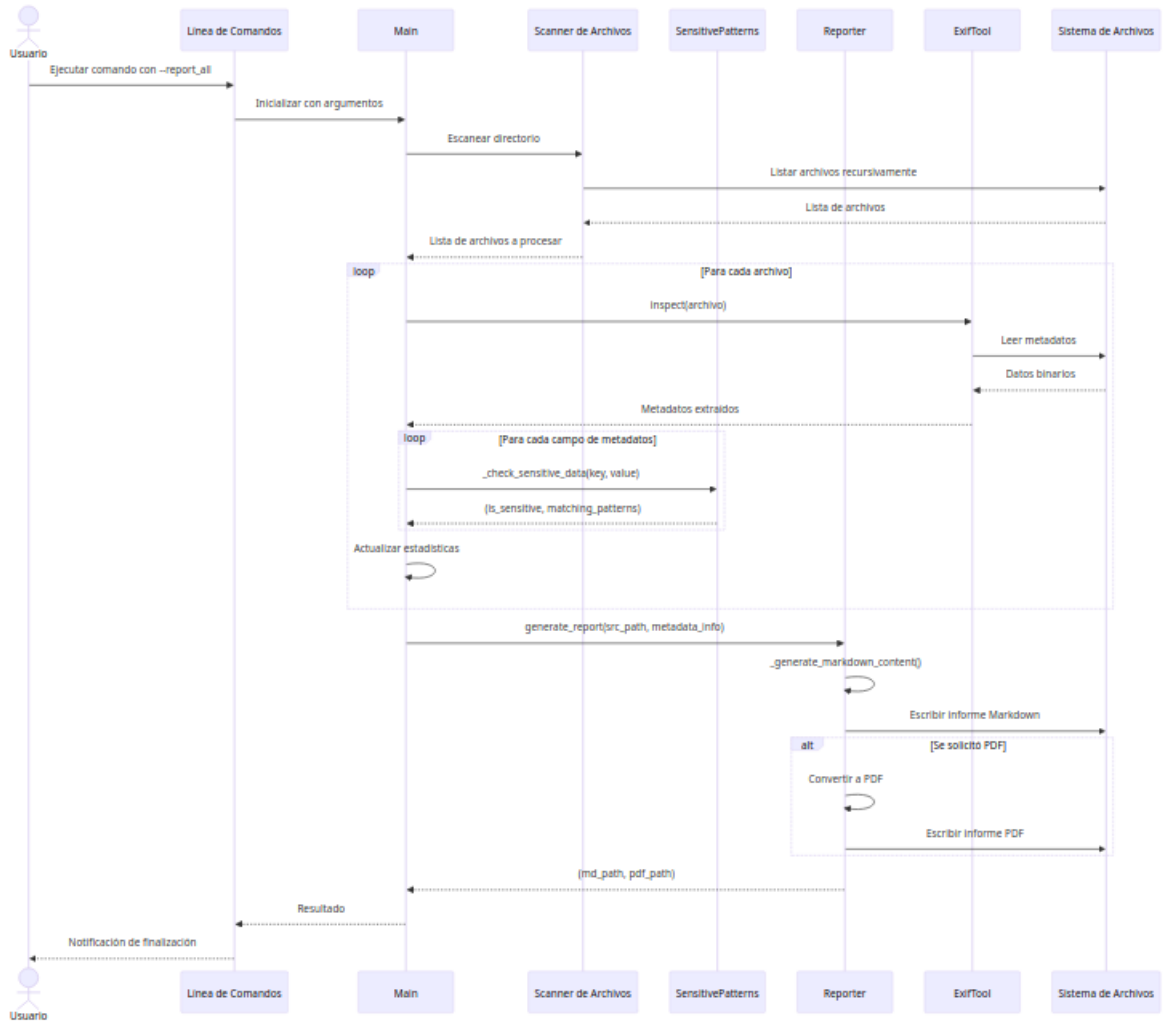
1.33 Características del Flujo de Datos

1. **Procesamiento Recursivo**: La aplicación procesa directorios de forma recursiva, examinando todos los archivos con extensiones soportadas.
2. **Procesamiento Condicional**: Dependiendo de los argumentos de entrada, se activan diferentes rutas de procesamiento.
3. **Verificación en Dos Pasos**: La sensibilidad se verifica tanto en las claves como en los valores de los metadatos.
4. **Procesamiento Estadístico**: Se recopilan estadísticas sobre los archivos procesados, lo que permite generar informes detallados.
5. **Salidas Múltiples**: El sistema puede generar diferentes tipos de salida (archivos limpios, informes en diferentes formatos) según las necesidades del usuario. # Diagramas de Interacción - MetaInfo

Este documento presenta diagramas de interacción que muestran el comportamiento dinámico del sistema, ilustrando cómo los distintos componentes de MetaInfo colaboran para realizar las tareas principales.

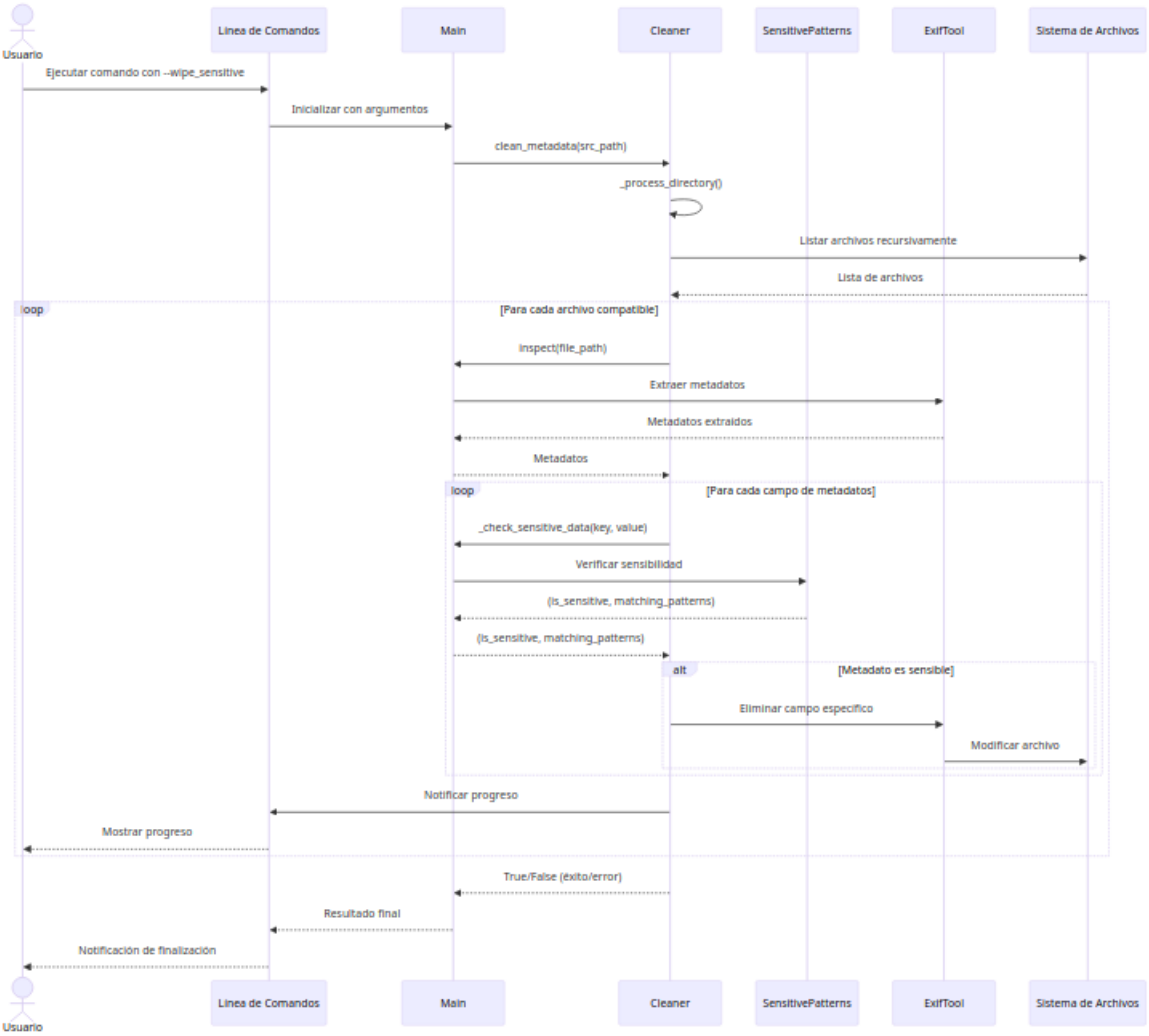
1.34 Diagrama de Secuencia: Generación de Informe

El siguiente diagrama muestra la secuencia de interacciones entre componentes para generar un informe de metadatos.



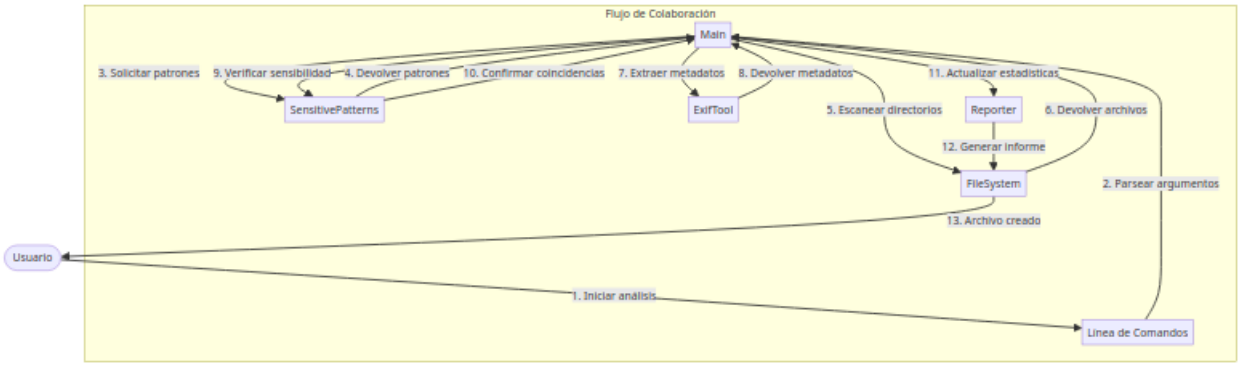
1.35 Diagrama de Secuencia: Limpieza de Metadatos Sensibles

El siguiente diagrama muestra la secuencia de interacciones para eliminar metadatos sensibles de los archivos.



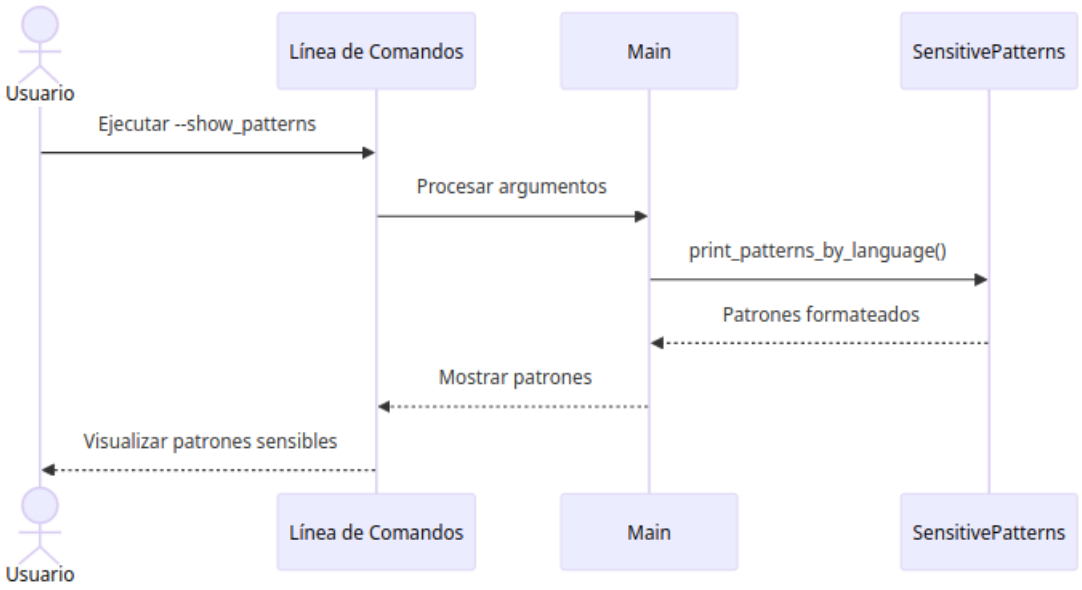
1.36 Diagrama de Colaboración: Detección de Datos Sensibles

El siguiente diagrama muestra cómo colaboran los objetos durante el proceso de detección de datos sensibles.



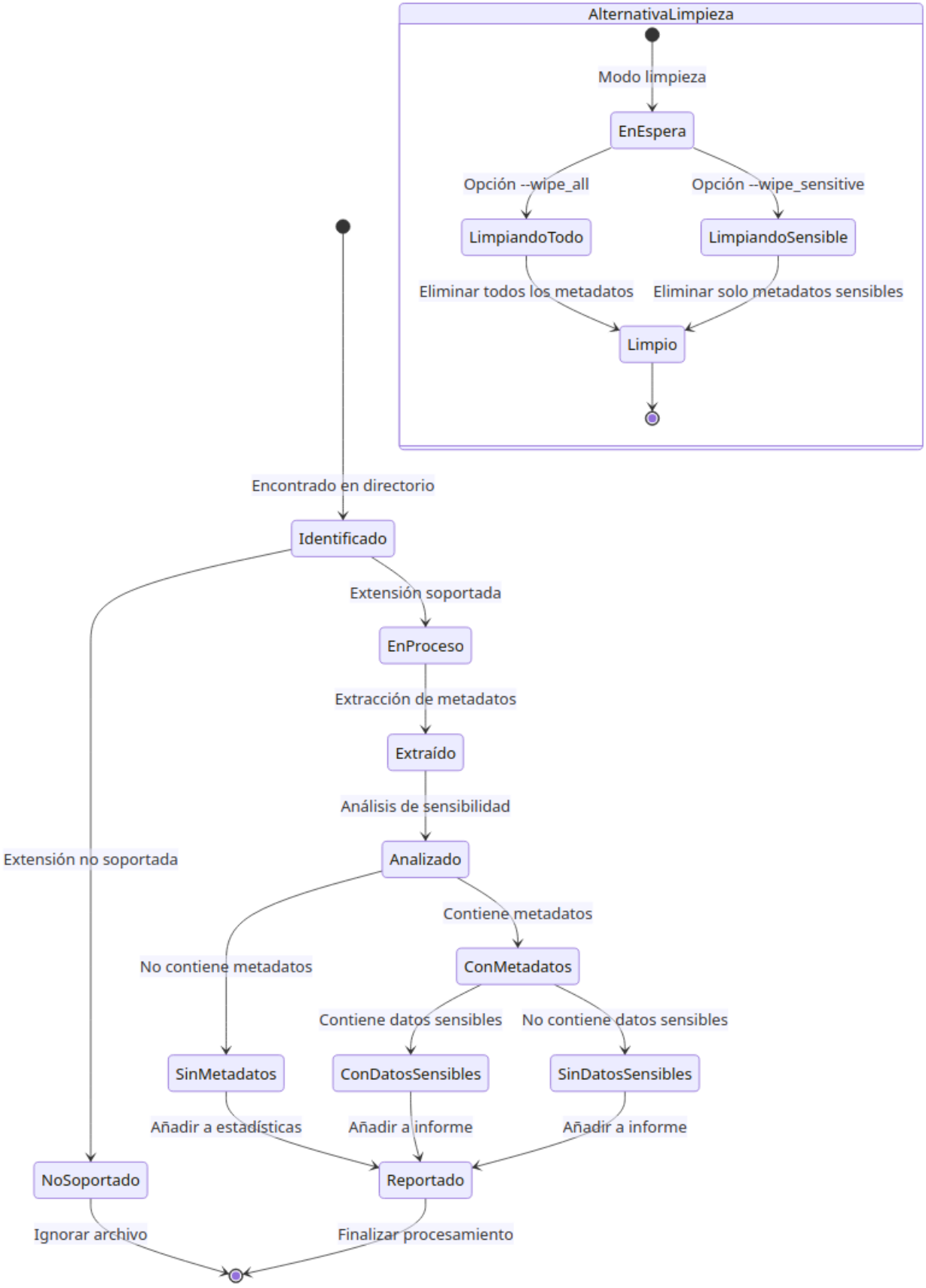
1.37 Diagrama de Secuencia: Consulta de Patrones Sensibles

El siguiente diagrama muestra la interacción cuando un usuario consulta los patrones sensibles.



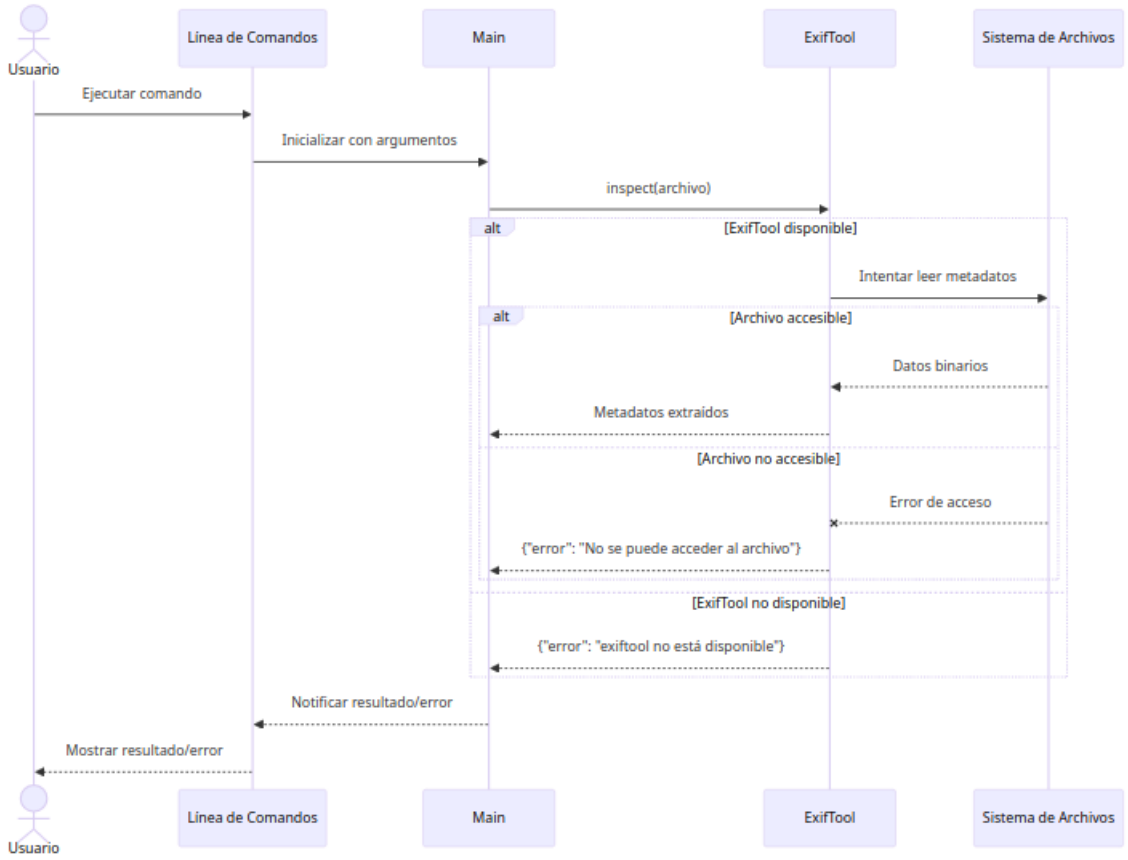
1.38 Diagrama de Estados: Procesamiento de un Archivo

El siguiente diagrama muestra los estados por los que pasa un archivo durante su procesamiento.


AlternativaLimpieza
Modo limpieza
EnEspera
Opción --wipe_all
Opción --wipe_sensitive
LimpiandoTodo
LimpiandoSensible
Eliminar todos los metadatos
Eliminar solo metadatos sensibles
Limpio
Encontrado en directorio
Identificado
Extensión soportada
EnProceso
Extracción de metadatos
Extraído
Análisis de sensibilidad
Analizado
Extensión no soportada
Contiene metadatos
No contiene metadatos
ConMetadatos
Contiene datos sensibles
No contiene datos sensibles
SinMetadatos
ConDatosSensibles
SinDatosSensibles
Añadir a estadísticas
Añadir a informe
Añadir a informe
NoSoportado
Reportado
Ignorar archivo
Finalizar procesamiento

1.39 Diagrama de Secuencia: Caso de Error en la Extracción de Metadatos

El siguiente diagrama muestra cómo el sistema maneja situaciones de error durante la extracción de metadatos.



1.40 Explicación de los Diagramas

1.40.1 Generación de Informe

Este diagrama muestra cómo el sistema procesa una solicitud para generar un informe. Ilustra el flujo desde la entrada del comando hasta la generación del archivo de informe, mostrando cómo se escanean los archivos, se extraen y analizan los metadatos, y se genera el informe.

1.40.2 Limpieza de Metadatos Sensibles

Este diagrama detalla el proceso de limpieza selectiva de metadatos sensibles. Muestra cómo se identifica la información sensible y cómo se elimina mientras se preserva el resto de los metadatos.

1.40.3 Detección de Datos Sensibles

Este diagrama de colaboración muestra las interacciones entre los diferentes componentes durante el proceso de detección de datos sensibles, destacando el flujo de la información y la colaboración entre objetos.

1.40.4 Consulta de Patrones Sensibles

Este diagrama simple muestra la interacción cuando un usuario solicita ver los patrones considerados sensibles por el sistema.

1.40.5 Estados de Procesamiento de un Archivo

Este diagrama de estados ilustra los diferentes estados por los que puede pasar un archivo durante su procesamiento, desde su identificación inicial hasta el informe final o la limpieza.

1.40.6 Manejo de Errores

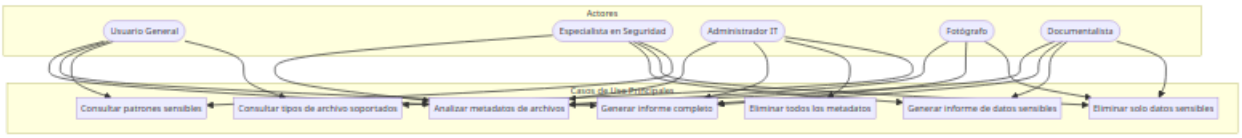
El último diagrama muestra cómo el sistema maneja situaciones de error durante la extracción de metadatos, ya sea por la falta de disponibilidad de ExifTool o por problemas al acceder a los archivos. # Casos de Uso - MetaInfo

Este documento describe los principales casos de uso de la aplicación MetaInfo, mostrando cómo los diferentes tipos de usuarios interactúan con el sistema para lograr sus objetivos.

1.41 Navegación de la Documentación

- ← Volver al Índice
- Ver Diagrama de Componentes
- Ver Diagrama de Clases

1.42 Diagrama de Casos de Uso



1.43 Descripción Detallada de Casos de Uso

1.43.1 UC1: Analizar metadatos de archivos

Actores principales: Todos los usuarios

Descripción: El usuario desea analizar los metadatos contenidos en archivos de un directorio para conocer qué información contienen.

Precondiciones: - El usuario tiene instalado Python y las dependencias de MetaInfo - El usuario tiene acceso a los archivos a analizar

Flujo principal: 1. El usuario ejecuta el comando con la ruta del directorio a analizar 2. El sistema escanea recursivamente todos los archivos con extensiones soportadas 3. Por cada archivo, el sistema extrae los metadatos 4. El sistema analiza cada campo de metadatos para detectar información sensible 5. El sistema genera estadísticas sobre los archivos procesados

Flujos alternativos: - Si no se encuentra ExifTool, el sistema muestra un mensaje de error adecuado - Si no hay archivos con extensiones soportadas, el sistema lo notifica al usuario

Postcondiciones: - El sistema ha recopilado la información de metadatos de los archivos

Ejemplo:

```
python metainfo.py --i ~/Documentos --verbose
```

1.43.2 UC2: Generar informe completo

Actores principales: Usuario General, Administrador IT, Fotógrafo, Documentalista

Descripción: El usuario desea generar un informe completo que incluya todos los metadatos encontrados en los archivos, sean sensibles o no.

Precondiciones: - Se ha completado el análisis de metadatos (UC1)

Flujo principal: 1. El sistema compila toda la información de metadatos recopilada 2. El sistema genera un informe en formato Markdown 3. Si se solicita, el sistema convierte el informe a formato PDF 4. El sistema guarda el informe en la ubicación especificada

Flujos alternativos: - Si se solicita PDF pero no está disponible Pandoc, se genera solo el informe Markdown con un aviso

Postcondiciones: - Se ha generado un informe detallado con todos los metadatos

Ejemplo:

```
python metainfo.py --i ~/Fotos/Vacaciones --report_all --pdf --o ~/Informes
```

1.43.3 UC3: Generar informe de datos sensibles

Actores principales: Especialista en Seguridad, Documentalista

Descripción: El usuario desea generar un informe que incluya solo los metadatos considerados sensibles, para evaluar riesgos de privacidad y seguridad.

Precondiciones: - Se ha completado el análisis de metadatos (UC1)

Flujo principal: 1. El sistema filtra los metadatos para incluir solo aquellos identificados como sensibles 2. El sistema genera un informe en formato Markdown con los datos sensibles 3. Si se solicita, el sistema convierte el informe a formato PDF 4. El sistema guarda el informe en la ubicación especificada

Flujos alternativos: - Si no se encuentran datos sensibles, se genera un informe indicando que no hay información sensible

Postcondiciones: - Se ha generado un informe específico sobre los datos sensibles encontrados

Ejemplo:

```
python metainfo.py --i ~/Documentos/Confidencial --report_sensitive --o ~/Informes/Seguridad
```

1.43.4 UC4: Eliminar todos los metadatos

Actores principales: Administrador IT

Descripción: El usuario desea eliminar todos los metadatos de los archivos para prepararlos para su distribución o publicación.

Precondiciones: - El usuario tiene permisos de escritura en los archivos a procesar - ExifTool está instalado en el sistema

Flujo principal: 1. El sistema identifica todos los archivos con extensiones soportadas 2. El sistema omite archivos de texto (.txt) que no tienen metadatos relevantes 3. Para cada archivo válido, el sistema elimina todos los metadatos utilizando estrategias apropiadas 4. El sistema verifica que la limpieza se haya realizado correctamente 5. El sistema notifica el progreso y resultado de la operación

Flujos alternativos: - Si algún archivo está protegido contra escritura, el sistema lo notifica y continúa con los demás - Si ExifTool no está disponible, el sistema muestra un error y detiene el proceso - Si un método de limpieza falla, el sistema intenta métodos alternativos

Postcondiciones: - Los archivos procesados ya no contienen metadatos - Se mantiene un registro de cualquier error ocurrido durante el proceso

Ejemplo:

```
python metainfo.py --i ~/Documentos/ParaPublicar --wipe_all --verbose
```

1.43.5 UC5: Eliminar solo datos sensibles

Actores principales: Especialista en Seguridad, Fotógrafo, Documentalista

Descripción: El usuario desea eliminar solo los metadatos clasificados como sensibles, manteniendo el resto de la información.

Precondiciones: - El usuario tiene permisos de escritura en los archivos a procesar - ExifTool está instalado en el sistema

Flujo principal: 1. El sistema identifica todos los archivos con extensiones soportadas 2. El sistema omite archivos de texto (.txt) que no tienen metadatos relevantes 3. Para cada archivo válido, el sistema extrae los metadatos 4. El sistema identifica qué campos contienen información sensible 5. El sistema elimina solo los campos sensibles mediante técnicas apropiadas 6. El sistema verifica que se hayan eliminado correctamente los campos sensibles 7. El sistema notifica el progreso y resultado de la operación

Flujos alternativos: - Si no se encuentran datos sensibles en un archivo, el sistema lo notifica y continúa - Si ExifTool no está disponible, el sistema muestra un error y detiene el proceso - Si un método de limpieza falla, el sistema intenta métodos alternativos

Postcondiciones: - Los archivos procesados no contienen metadatos sensibles, pero conservan el resto de la información - Se mantiene un registro de cualquier error ocurrido durante el proceso

Ejemplo:

```
python metainfo.py --i ~/Fotos/Portfolio --wipe_sensitive --verbose
```

1.43.6 UC6: Consultar patrones sensibles

Actores principales: Usuario General, Especialista en Seguridad

Descripción: El usuario desea conocer qué patrones son considerados sensibles por el sistema.

Precondiciones: - Ninguna específica

Flujo principal: 1. El usuario solicita la lista de patrones sensibles 2. El sistema muestra todos los patrones organizados por idioma y tipo

Postcondiciones: - El usuario conoce qué información es considerada sensible por el sistema

Ejemplo:

```
python metainfo.py --show_patterns
```

1.43.7 UC7: Consultar tipos de archivo soportados

Actores principales: Usuario General, Administrador IT

Descripción: El usuario desea conocer qué tipos de archivo son soportados por la aplicación.

Precondiciones: - Ninguna específica

Flujo principal: 1. El usuario solicita la lista de tipos de archivo soportados 2. El sistema muestra todas las extensiones soportadas organizadas por categoría

Postcondiciones: - El usuario conoce qué tipos de archivo puede procesar el sistema

Ejemplo:

```
python metainfo.py --show_mimes
```

1.44 Escenarios de Uso por Tipo de Usuario

1.44.1 Perfil: Fotógrafo Profesional

Escenario: Un fotógrafo profesional necesita preparar una colección de fotografías para entregar a un cliente, asegurándose de que no contengan información personal o sensible.

Secuencia típica: 1. Analizar los metadatos de la colección de fotos para identificar qué información contienen 2. Generar un informe completo para revisar en detalle los metadatos presentes 3. Eliminar selectivamente los metadatos sensibles (como ubicación GPS, número de serie de la cámara, etc.) 4. Verificar que la limpieza se ha realizado correctamente

Comandos utilizados:

```
# Paso 1 y 2: Analizar y generar informe
python metainfo.py --i ~/Fotos/ClienteXYZ --report_all --o ~/Informes --pdf

# Paso 3: Eliminar metadatos sensibles
python metainfo.py --i ~/Fotos/ClienteXYZ --wipe_sensitive --verbose

# Paso 4: Verificar limpieza
python metainfo.py --i ~/Fotos/ClienteXYZ --report_sensitive
```

1.44.2 Perfil: Especialista en Seguridad

Escenario: Un especialista en seguridad necesita auditar documentos corporativos para identificar posibles filtraciones de información sensible a través de metadatos.

Secuencia típica: 1. Consultar qué patrones son considerados sensibles para entender el alcance del análisis 2. Analizar los metadatos de los documentos corporativos 3. Generar un informe específico de datos sensibles para identificar riesgos 4. Recomendar o implementar la limpieza selectiva de información sensible

Comandos utilizados:

```
# Paso 1: Consultar patrones sensibles
python metainfo.py --show_patterns

# Paso 2 y 3: Analizar y generar informe de datos sensibles
python metainfo.py --i /datos/documentos_corporativos --report_sensitive --o /informes/auditoria --pdf

# Paso 4: Implementar limpieza selectiva
python metainfo.py --i /datos/documentos_corporativos --wipe_sensitive --verbose
```

1.44.3 Perfil: Documentalista

Escenario: Un documentalista necesita organizar una colección de documentos históricos, preservando la información de autoría y fechas pero eliminando datos personales sensibles.

Secuencia típica: 1. Analizar los metadatos de la colección para entender qué información contienen 2. Generar un informe completo para catalogar la información de autoría, fechas, etc. 3. Identificar qué documentos contienen información sensible 4. Eliminar selectivamente los datos sensibles preservando la información histórica relevante

Comandos utilizados:

```
# Paso 1 y 2: Analizar y generar informe completo
python metainfo.py --i /coleccion/documentos_historicos --report_all --o /catalogos/metadatos --pdf

# Paso 3: Identificar documentos con datos sensibles
python metainfo.py --i /coleccion/documentos_historicos --report_sensitive --o /catalogos/revision_sensi
```

```
# Paso 4: Eliminar selectivamente los datos sensibles
python metainfo.py --i /coleccion/documentos_historicos --wipe_sensitive --verbose
```

1.45 Relación con los Componentes del Sistema

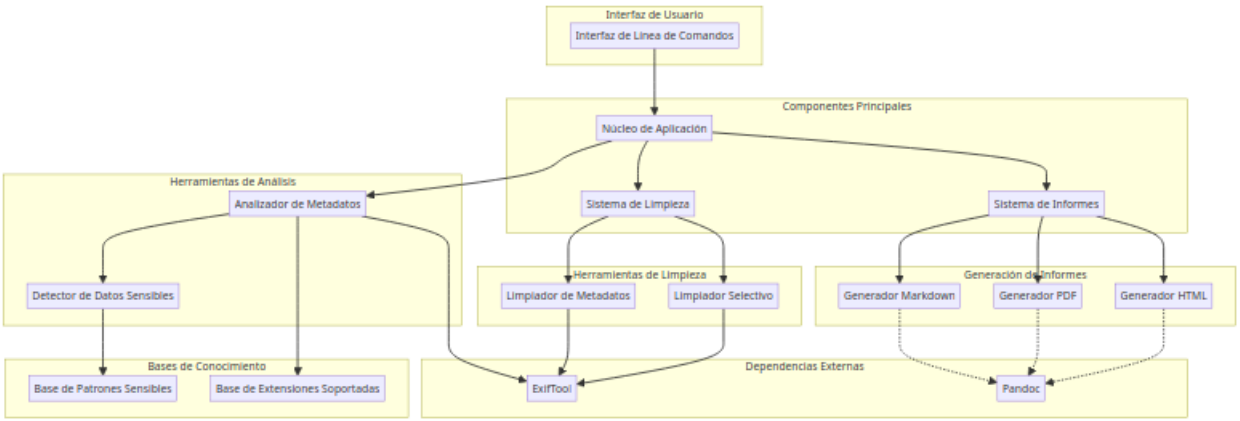
Los casos de uso aquí descritos se implementan mediante los componentes detallados en el Diagrama de Componentes:

- **UC1** utiliza principalmente el **Analizador de Metadatos** y el **Detector de Datos Sensibles**
- **UC2** y **UC3** utilizan el **Sistema de Informes** y sus generadores específicos
- **UC4** y **UC5** utilizan el **Sistema de Limpieza** con sus estrategias específicas
- **UC6** utiliza la **Base de Patrones Sensibles**
- **UC7** utiliza la **Base de Extensiones Soportadas**

Última actualización: 11/06/2024 # Diagrama de Componentes - MetaInfo

Este documento presenta una visión de alto nivel de los componentes principales del sistema MetaInfo y sus interacciones.

1.46 Diagrama de Componentes



1.47 Descripción de Componentes

1.47.1 Interfaz de Usuario

- **Interfaz de Línea de Comandos**: Componente que maneja la interacción con el usuario, procesa argumentos y muestra resultados.

1.47.2 Componentes Principales

- **Núcleo de Aplicación**: Coordina todos los componentes del sistema y gestiona el flujo de trabajo.
- **Sistema de Informes**: Maneja la generación de informes en diferentes formatos.
- **Sistema de Limpieza**: Gestiona los procesos de limpieza de metadatos.

1.47.3 Herramientas de Análisis

- **Analizador de Metadatos**: Extrae y procesa metadatos de diferentes tipos de archivos.
- **Detector de Datos Sensibles**: Identifica información sensible en los metadatos.

1.47.4 Generación de Informes

- **Generador Markdown**: Crea informes en formato Markdown.
- **Generador PDF**: Convierte informes Markdown a PDF para distribución formal.
- **Generador HTML**: Convierte informes Markdown a HTML para visualización web.

1.47.5 Herramientas de Limpieza

- **Limpiador de Metadatos**: Elimina todos los metadatos de los archivos.
- **Limpiador Selectivo**: Elimina solo los metadatos identificados como sensibles.

1.47.6 Dependencias Externas

- **ExifTool**: Herramienta externa utilizada para leer y modificar metadatos.
- **Pandoc**: Herramienta externa utilizada para convertir entre formatos de documento.

1.47.7 Bases de Conocimiento

- **Base de Patrones Sensibles**: Contiene patrones predefinidos para identificar información sensible.
- **Base de Extensiones Soportadas**: Define qué tipos de archivo son compatibles con el sistema.

1.48 Interacciones Principales

1. El usuario interactúa con el sistema a través de la **Interfaz de Línea de Comandos**.
2. El **Núcleo de Aplicación** interpreta los comandos y coordina el flujo de trabajo.
3. Para analizar metadatos:
   - El **Analizador de Metadatos** utiliza **ExifTool** para extraer información.
   - El **Detector de Datos Sensibles** compara los metadatos con la **Base de Patrones Sensibles**.
4. Para generar informes:
   - El **Sistema de Informes** recopila la información analizada.
   - Los generadores de formato (**Markdown**, **PDF**, **HTML**) crean los documentos solicitados.
5. Para limpiar metadatos:
   - El **Sistema de Limpieza** determina qué tipo de limpieza realizar.
   - El **Limpiador de Metadatos** o **Limpiador Selectivo** ejecuta la limpieza utilizando **ExifTool**.

1.49 Características de Arquitectura

- **Modularidad**: Los componentes están diseñados para funcionar de manera independiente con interfaces claras.
- **Extensibilidad**: Cada componente puede ser extendido sin afectar al resto del sistema.
- **Robustez**: Implementación de manejo de errores en cada componente para garantizar la estabilidad.
- **Bajo Acoplamiento**: Las dependencias entre componentes están minimizadas para facilitar mantenimiento y pruebas.

1.50 Relación con el Diagrama de Clases

Este diagrama de componentes proporciona una visión de alto nivel del sistema, mientras que el Diagrama de Clases ofrece un detalle más fino sobre la implementación concreta de estas funcionalidades en clases y sus relaciones.

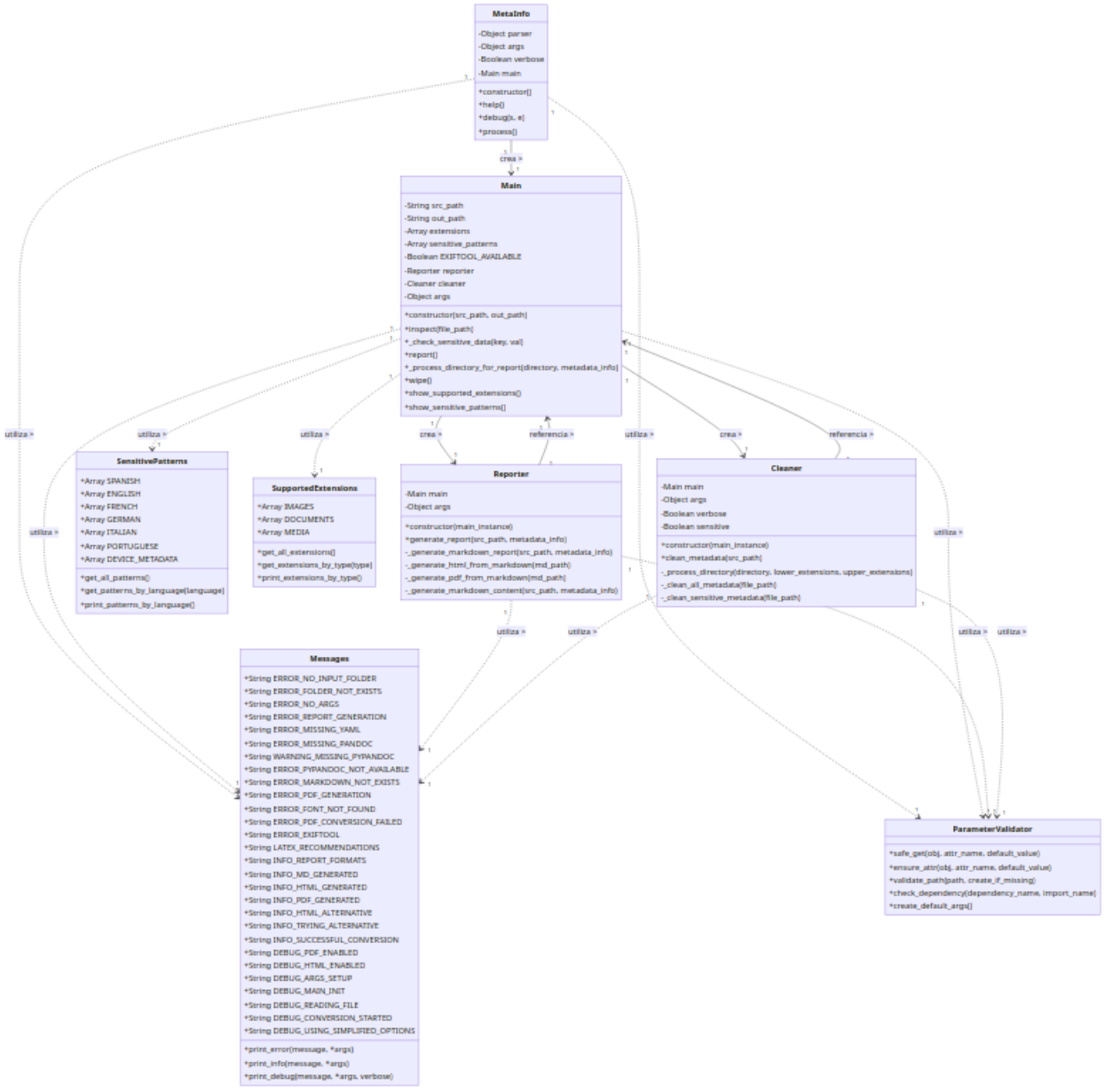
Última actualización: 11/06/2024 # Diagrama de Clases - MetaInfo

Este documento presenta el diseño estático de clases de la aplicación MetaInfo, describiendo sus atributos, métodos y relaciones.

1.51 Navegación de la Documentación

- ← Volver al Índice
- Ver Diagrama de Componentes
- Ver Diagramas de Interacción

1.52 Diagrama de Clases



1.53 Descripción Detallada de Clases

1.53.1 MetaInfo

La clase principal que maneja la entrada del usuario y coordina el procesamiento.

1.53.1.1 Atributos

- `parser`: Parser de argumentos
- `args`: Argumentos de línea de comandos
- `verbose`: Indica si se debe mostrar información detallada
- `main`: Instancia de la clase Main

1.53.1.2 Métodos

- `constructor`: Inicializa la aplicación, configura el parser y procesa argumentos
- `help`: Muestra la ayuda del programa
- `debug`: Muestra mensajes de depuración
- `process`: Procesa la solicitud del usuario según los argumentos

1.53.2 Main

La clase central que coordina todas las operaciones. Contiene referencias a las clases `Reporter` y `Cleaner`.

1.53.2.1 Atributos

- `src_path`: Ruta del directorio a procesar
- `out_path`: Ruta de salida para los informes
- `extensions`: Lista de extensiones de archivo soportadas
- `sensitive_patterns`: Lista de patrones sensibles
- `EXIFTOOL_AVAILABLE`: Indica si la herramienta ExifTool está disponible
- `reporter`: Instancia de la clase Reporter
- `cleaner`: Instancia de la clase Cleaner
- `args`: Argumentos de línea de comandos

1.53.2.2 Métodos

- `constructor`: Inicializa la clase con las rutas de origen y destino
- `inspect`: Inspecciona un archivo para extraer sus metadatos
- `_check_sensitive_data`: Verifica si una clave o valor contiene datos sensibles
- `report`: Genera un informe de metadatos
- `_process_directory_for_report`: Procesa recursivamente un directorio para el informe
- `wipe`: Limpia los metadatos de los archivos
- `show_supported_extensions`: Muestra las extensiones soportadas
- `show_sensitive_patterns`: Muestra los patrones sensibles

1.53.3 Reporter

Clase responsable de generar informes de metadatos en formatos Markdown, HTML y PDF.

1.53.3.1 Atributos

- `main`: Referencia a la instancia de Main
- `args`: Argumentos de línea de comandos

1.53.3.2 Métodos

- `constructor`: Inicializa la clase con una referencia a Main
- `generate_report`: Genera un informe basado en la información recopilada
- `_generate_markdown_report`: Genera el informe en formato Markdown
- `_generate_html_from_markdown`: Convierte el informe Markdown a HTML
- `_generate_pdf_from_markdown`: Convierte el informe Markdown a PDF con portada e índice
- `_generate_markdown_content`: Genera el contenido del informe en formato Markdown

1.53.4 Cleaner

Clase responsable de limpiar metadatos de archivos.

1.53.4.1 Atributos

- `main`: Referencia a la instancia de Main
- `args`: Argumentos de línea de comandos
- `verbose`: Indicador de modo detallado
- `sensitive`: Indicador de modo de limpieza sensitiva

1.53.4.2 Métodos

- `constructor`: Inicializa la clase con una referencia a Main
- `clean_metadata`: Método principal que coordina la limpieza de metadatos
- `_process_directory`: Procesa recursivamente directorios
- `_clean_all_metadata`: Elimina todos los metadatos de un archivo
- `_clean_sensitive_metadata`: Elimina solo los metadatos sensibles

1.53.5 Messages

Clase que centraliza todos los mensajes al usuario y proporciona métodos para mostrarlos.

1.53.5.1 Atributos (constantes)

- Múltiples constantes para diferentes categorías de mensajes

1.53.5.2 Métodos

- `print_error`: Imprime un mensaje de error formateado
- `print_info`: Imprime un mensaje informativo formateado
- `print_debug`: Imprime un mensaje de depuración formateado, solo si verbose es True

1.53.6 ParameterValidator

Clase que proporciona métodos para validar parámetros y obtener valores seguros.

1.53.6.1 Métodos

- `safe_get`: Obtiene de forma segura el valor de un atributo
- `ensure_attr`: Asegura que un objeto tenga un atributo, asignándole un valor por defecto si no existe
- `validate_path`: Valida que una ruta exista, opcionalmente creándola si no existe
- `check_dependency`: Verifica si una dependencia está instalada
- `create_default_args`: Crea un objeto con valores por defecto para los argumentos

1.53.7 SensitivePatterns

Clase que define los patrones considerados sensibles para detectar en metadatos.

1.53.7.1 Atributos (constantes)

- Listas de patrones sensibles en varios idiomas
- Lista de metadatos específicos de dispositivos

1.53.7.2 Métodos

- `get_all_patterns`: Obtiene todos los patrones sensibles
- `get_patterns_by_language`: Obtiene los patrones para un idioma específico
- `print_patterns_by_language`: Imprime los patrones agrupados por idioma

1.53.8 SupportedExtensions

Clase que define las extensiones de archivo soportadas.

1.53.8.1 Atributos (constantes)

- Listas de extensiones por categoría

1.53.8.2 Métodos

- `get_all_extensions`: Obtiene todas las extensiones soportadas
- `get_extensions_by_type`: Obtiene las extensiones para un tipo específico
- `print_extensions_by_type`: Imprime las extensiones agrupadas por tipo

1.54 Relaciones entre Clases

- `MetaInfo` crea y configura la instancia de `Main`, que es el núcleo del sistema.
- `Main` crea y mantiene instancias de `Reporter` y `Cleaner`.
- `Main`, `Reporter` y `Cleaner` utilizan `Messages` para mostrar mensajes al usuario.
- `Main`, `Reporter` y `Cleaner` utilizan `ParameterValidator` para validar y obtener valores seguros.
- `Main` utiliza las clases `SensitivePatterns` y `SupportedExtensions` para obtener datos constantes.
- `Reporter` y `Cleaner` mantienen referencias a la instancia de `Main` para acceder a sus métodos y atributos.
- No hay herencia entre clases, se utiliza composición para la reutilización de código.

1.55 Correspondencia con el Diagrama de Componentes

Este diagrama de clases implementa los componentes de alto nivel definidos en el Diagrama de Componentes:

- La clase `MetaInfo` implementa la **Interfaz de Línea de Comandos**
- La clase `Main` implementa el **Núcleo de Aplicación**
- La clase `Reporter` implementa el **Sistema de Informes**
- La clase `Cleaner` implementa el **Sistema de Limpieza**
- La funcionalidad dentro de `Main` implementa el **Analizador de Metadatos** y **Detector de Datos Sensibles**
- Las clases `SensitivePatterns` y `SupportedExtensions` implementan las **Bases de Conocimiento**

Última actualización: 11/06/2024