Universidad de las Ciencias Informáticas

Facultad 1

Título: Desarrollo del módulo de pre-procesamiento para Motor de Categorización Inteligente de Contenido

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autores: Alexis Rodriguez Reyes
Maidelín Prieto Guerra

Tutor: Ing. Kiuver Kaddiel Ibañez Castro

La Habana. 15 de junio de 2012

"Año 54 de la Revolución"

"Sé amable con los nerds. Hay muchas probabilidades de que termines trabajando para uno de ellos"

Bill Gates

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____	_____
Autor:	Autor:
Alexis Rodriguez Reyes	Maidelín Prieto Guerra

_____

Tutor:

Ing. Kiuver Kaddiel Ibañez Castro

Agradecimientos

Gracias a mi familia por su apoyo incondicional en todo momento, por apoyarme y por guiarme en el transcurso de mi vida.

A mami, por ser la mejor madre del mundo, por su apoyo, por su amor, por ser tan fuerte y criar a dos locos como mi hermano y yo.

A mi mejor cuñada por sus consejos y su apoyo en todo momento, un beso grande de tu "MC".

A mi hermano por ser mi amigo, mi padre, mi todo, por aconsejarme y guiarme en mi crecimiento y formación en la vida.

A Danielito, el miembro más joven de la familia por ser tan bello, tan precioso y correr tan rápido en el andador.

A papi por su apoyo continuo y sus consejos que me ayudaron a crecer y cumplir con todas mis metas.

A karel, por dedicarme su tiempo y ayudarme en todo con mucha paciencia.

A kiuver, por todo lo que ha sido capaz de sacrificar por mí, por ser mi amigo, por guiarme y cuidarme en todo momento, por aconsejarme, por enseñarme las cosas buenas y malas de esta vida.

Alexis Rodriguez Reyes

A mami, por ser mi consejera, mi amiga, mi protectora, mi guía y mi ejemplo a seguir. Por haberme dado tanto amor y cariño durante toda mi vida. Gracias por ser mi hada madrina, por enseñarme a enfrentar todos los problemas sin ningún temor. Por ser la mujer más especial y perfecta que he conocido.

A papi por estar junto a mí en todo momento, por enseñarme a luchar por lo que quiero y no rendirme. Por ser el hombre más sencillo y honrado del mundo. Por haberme dado tanto amor y atención en mi

vida, por ser mi maestro, mi protector.

A mis hermanas Maidelis y Maryelis que son mis tesoros, las quiero mucho.

A mis abuelos y a mi tío Pipi por apoyarme siempre y estar al tanto de mi vida, aun sin vivir cerca de mí, los adoro.

A mis amigas: Saily, Anaelys, Yiliam, Yoandra, Ariadna y Leinis con las que he vivido muchas experiencias y momentos divertidos en la universidad, en los cuales hemos reído y llorado. Muchas gracias, las quiero mucho.

A mi compañero de tesis, mi hermano y amigo Alexi, por todo el apoyo y ayuda que me brindó.

A mi tutor Kiuver por estar siempre pendiente a todo durante el desarrollo de la tesis y por ser un buen amigo.

A mis amigas de Sandino: Naila, Anily y Elizabeth que siempre me han apoyado.

A todos lo que de una forma u otra me han ayudado en esta etapa de mi vida, gracias por todo.

Maidelín Prieto Guerra

Dedicatoria

Dedico este trabajo a mi familia, especialmente a mi mamá para que pueda decir que todos en la familia somos ingenieros.

Alexis Rodriguez Reyes

A mis padres Maira y Jorge por estar junto a mí en todos los momentos de mi vida apoyándome incondicionalmente.

A mis hermanas Maidelis y Maryelis, espero que estén orgullosas de mí.

A mis Abuelos y mi tío Pipi por el apoyo que siempre me han dado.

Maidelín Prieto Guerra

Resumen

En la Universidad de las Ciencias Informáticas (UCI) se está desarrollando un proyecto llamado Motor de Categorización Inteligente de Contenido (MOCIC), el cual tiene como funcionalidad general, recibir información digitalizada para su categorización en categorías preestablecidas. Uno de los primeros pasos en el proceso de categorización es la extracción de contenidos de los documentos como son el texto y las imágenes. Este paso mencionado se realiza mediante un módulo que actualmente no presenta un funcionamiento adecuado por las siguientes razones. Se encuentra unido a otro módulo creando un alto acoplamiento en la arquitectura del sistema. No posee una arquitectura que permita la distribución del módulo en diferentes nodos ni la inclusión de nuevos tipos de documentos. No se integra de forma correcta al nuevo protocolo de comunicación definido en el proyecto.

Este trabajo presenta el desarrollo de un módulo capaz de extraer el texto, las imágenes y los enlaces de un documento. Se realiza un análisis de la problemática donde se identificaron las principales cuestiones que conllevaron a la solución que se expone. Se diseña una arquitectura distribuida y extensible. Se explica la estructura interna de la solución propuesta mediante los diagramas de clases del diseño, modelo de datos, diagrama de paquetes, diagrama de componentes y vista lógica de la arquitectura. Finalmente, se valida la solución propuesta a través de la realización de las pruebas funcionales diseñadas que propone la metodología *Scrum-Extreme Pogramming* (SXP).

PALABRAS CLAVE: Arquitectura distribuida, Categorización de texto, Categorización de imágenes, Extracción de contenidos.

Índice general

Introducción

El aumento de la capacidad de los dispositivos de almacenamiento ha permitido incrementar el volumen de información que existe en el universo digital [1]. Una computadora de un laboratorio de producción en la Universidad de las Ciencias Informáticas (UCI) tiene como capacidad de almacenamiento aproximado 160GB, dentro de los archivos que se guardan en ella como modelos, planillas o tesis; el tamaño medio de cada uno de estos, de acuerdo a un estudio realizado es de 2MB, ver Anexo A. Una persona tarda 49 segundos como promedio en categorizar de forma manual un documento como se explica en el Anexo B. Contando con una carpeta de muestra la cual posee 5000 archivos, se multiplica esta cantidad por el tiempo que tarda una persona en categorizar de forma manual (49 segundos) y se divide entre 3600 (cantidad de segundos en una hora), el resultado sería de 68.05, lo cual representa el tiempo en horas que se tardaría y si se divide este resultado entre 8 que son las horas por jornadas laborales, esto arrojaría como resultado 8.56, que serían las jornadas de trabajo necesarias para categorizar los 5000 documentos mencionados. Por lo anteriormente expuesto, categorizar la información existente, de forma manual en la mayoría de las empresas que generan documentación digital, es una tarea inviable y se necesita una herramienta que permita categorizar de forma automática toda esta información.

En la UCI se está desarrollando un proyecto llamado Motor de Categorización Inteligente de Contenido (MOCIC), el cual tiene como funcionalidad general, recibir información digitalizada para su categorización en categorías preestablecidas. El proceso de extracción de contenido de un documento es uno de los primeros pasos en la categorización, en el cual intervienen varios módulos en MOCIC.

Al llegar un nuevo documento a categorizar, el módulo *Recover* se encarga de recuperar el mismo y de realizar mediante un programa escrito en Python el proceso de extracción de contenidos. Seguidamente el módulo *Store* es el encargado de almacenar de forma estructurada en carpetas la información del documento, mediante una copia que hace el *Recover*. El módulo *Controller* es el principal componente para distribuir estos contenidos a los diferentes categorizadores según su tipo, ya sea enlace, texto o imágenes como se puede apreciar en la Figura 1.

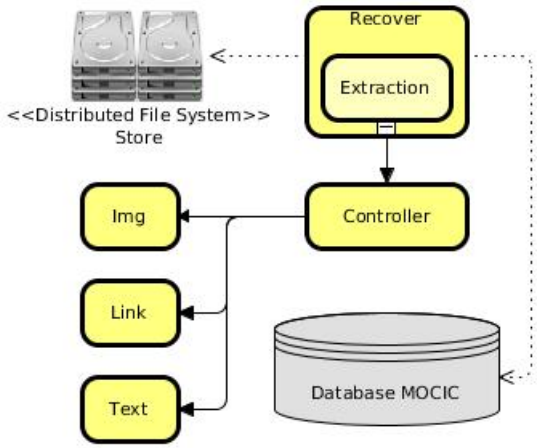

Figura 1: Diagrama de arquitectura de los componentes que interactúan en el proceso de extracción.

La extracción se realiza principalmente por el programa escrito en Python, el cual está unido al módulo *Recover*, lo que implica total dependencia hacia el mismo en el proceso. Como consecuencia de esto existe poca flexibilidad en la arquitectura de MOCIC, ya que la estructura del proceso de extracción evita un mejor despliegue de la misma. Los componentes que intervienen en la ejecución del proceso están obligados a ejecutarse en el mismo entorno por el alto nivel de acoplamiento que existe entre ellos. También, este programa utiliza herramientas que solucionan parcialmente el problema ya que poseen licencia *General Public License* (GPL), la cual es una licencia viral y su utilización implica que el sistema sea GPL. Esto no es conveniente para MOCIC por las políticas que sigue el proyecto, las cuales especifican que el sistema tiene como objetivo brindar solamente el servicio de categorización de contenidos, de modo que no se permitirá la adquisición ni publicación del código fuente del sistema. Tampoco permite la inclusión de otros formatos de documentos que actualmente el módulo no admite, por lo que no se le podrá realizar la extracción a estos. Por último, en el proyecto MOCIC han surgido nuevos cambios, dentro de ellos se encuentra la definición de una nueva arquitectura que posee un nuevo protocolo de comunicación, lo que conlleva a definir una integración correcta con el proceso de extracción.

Debido a la necesidad de un componente con características adecuadas para la extracción de los contenidos con el propósito de categorizar diferentes formatos de documentos, surge el siguiente problema a resolver: ¿Cómo extraer de los diferentes tipos de documentos, los elementos para la categorización de un contenido por MOCIC?

Se plantea como objeto de estudio: Herramientas para la extracción de elementos para categorizar un documento. El campo de acción se enmarca en: Herramientas para la extracción de elementos para categorizar un documento para MOCIC.

Para dar solución a la problemática descrita se establece como objetivo general: Desarrollar un módulo de pre-procesamiento para MOCIC que permita separar, de forma automática, los contenidos de un documento para su categorización.

El objetivo general se desglosa en los siguientes objetivos específicos:

- Caracterizar, mediante el estado del arte las aplicaciones similares existentes.
- Implementar un módulo que le permita a MOCIC separar los contenidos para su categorización.
- Integrar el módulo implementado a MOCIC.
- Validar el módulo implementado teniendo en cuenta su funcionamiento.

En el cumplimiento de las tareas se usarán los siguientes **Métodos teóricos:**

- **Analítico-Sintético:** Se aplicará para entender las herramientas de extracción de contenido de un documento a partir del análisis de las características y arquitectura que presentan, y para formular conclusiones a través de la síntesis de los conocimientos y resultados obtenidos.
- **Histórico-Lógico:** Permitirá una mayor comprensión del estado actual de las herramientas de extracción de contenido de un documento a partir del estudio de su evolución y las etapas principales por las que han transitado.
- **Modelación:** se utilizará el Lenguaje Unificado de Modelado (UML) para reflejar la estructura, relaciones internas y características de la solución a través de diagramas.

Estructura del Documento.

Capítulo 1: contiene la fundamentación teórica del tema, abordando básicamente los lenguajes de programación y las tecnologías que se utilizan en el desarrollo de la aplicación, seleccionando las más

apropiadas y argumentando el motivo de su utilización. Se exploran soluciones existentes similares al campo de acción para tener una guía de las posibles automatizaciones que se pueden realizar.

Capítulo 2: contiene la solución propuesta y los principales procesos que se realizan en el proyecto MOCIC, se realiza la planificación del tiempo requerido para el desarrollo de la aplicación mediante los artefactos generados por la metodología seleccionada. Aborda los aspectos funcionales para el desarrollo del sistema y definen los procesos fundamentales, para permitir la extracción de contenidos de los documentos para su categorización.

Capítulo 3: contiene la documentación de las pruebas que se le realizarán al sistema. Muestra una visión clara de la estructura del sistema en ejecución. Refleja las relaciones y las dependencias internas que poseen los componentes de la aplicación y estima el tiempo que se tomará en desarrollar el componente.

Para un mejor entendimiento y claridad en la lectura del documento se utilizará las siguientes reglas tipográficas:

- Los nombres de las clases se escribirán con fuente mono espaciada, ejemplo: `ClaseUno`
- Todas las palabras en otro idioma se escribirán en letra cursiva, ejemplo: *interface*
- Las secciones mostradas del código fuente se escribirán en inglés.

Capítulo 1

Fundamentación teórica

Introducción

En el presente capítulo se realiza una investigación sobre las herramientas de extracción de contenido. Se abordan las características principales de cada una de ellas tanto en el ámbito internacional como en el nacional. Se caracterizan y seleccionan las herramientas y tecnologías adecuadas para dar solución al problema planteado.

1.1. Herramientas de extracción de contenido a nivel internacional

A continuación se analizan las herramientas de extracción de contenidos que por sus características han logrado un importante espacio en el ámbito funcional o comercial, con el objetivo de estudiar sus características y extraer elementos que tributen al diseño de la solución.

Poppler-utils: es una biblioteca de software libre la cual posee varias herramientas para obtener información de los documentos PDF [2], dentro de las funcionalidades que ofrece esta biblioteca se encuentran:

- Pdffonts: posibilita analizar la fuente de un documento PDF.
- Información-PDF: brinda la información de un documento PDF.
- Funcionalidades que convierten de formato PDF a otros formatos:
  - Pdftohtml: convierte de formato PDF a HTML.
  - Pdfpm: convierte de formato PDF a *Portable Pixmap Format* (PPM), PNG o JPEG imagen.
  - Pdftops: convierte de formato PDF a PostScript (PS).
  - Pdftoabw: convierte de formato PDF a AbiWord.

- Pdftotext: extrae el texto del PDF.

Podofo: es una biblioteca libre escrita en el lenguaje de programación C++ para trabajar con el formato de archivo PDF, incluye clases para analizar un archivo PDF y modificar su contenido en la memoria. Los cambios se pueden volver a escribir en el disco fácilmente [3], esta incluye algunas herramientas como son:

- Popdofoencrypt: se encarga de descifrar cualquier archivo PDF y permite establecer permisos de PDF.
- Podofoimgextract: extrae todas las imágenes de un determinado archivo PDF.
- Podofoimpose: herramienta de imposición PDF, coloca las páginas de uno o más archivos PDF de origen en las páginas de un PDF nuevo.
- Podofomerge: se encarga de combinar dos archivos PDF en uno.
- Podofopdfinfo: proporciona información básica acerca de un PDF – metadatos y detalles de la página.
- Podofotxt2pdf: convierte un archivo de texto a un PDF.
- Podofotxtextract: herramienta que extrae todo el texto de un archivo PDF.
- Podofouncompress: elimina todos los filtros de compresión de un archivo PDF.

Pypdf: es una biblioteca escrita en el lenguaje de programación Python, construida como un conjunto de herramientas para la interacción con documentos PDF. Esta biblioteca es capaz de extraer información de documentos (título, autor), dividir documentos por página, además de la fusión de documentos página por página, de recorte por páginas y de la fusión de varias páginas en una sola página.

SGMLOP: es un módulo que proporciona un analizador rápido y sencillo para leer documentos XML. Analiza prácticamente cualquier archivo XML en una secuencia de etiquetas de inicio, entidades, las etiquetas de final y las secciones de texto [4], este módulo define una clase SGMLParser que sirve como base para el análisis de archivos de texto con formato *Standard Generalized Mark-up Language* (SGML); puede ser usado además para analizar los documentos *HyperText Mark Language* (HTML).

***Email*:** el paquete de correo electrónico es una biblioteca para la gestión de mensajes de correo electrónico [5]. La principal característica de identificación del paquete es que se divide el análisis de los mensajes de correo y la generación de la representación del objeto modelo interno de correo electrónico. Las aplicaciones que utilizan esta biblioteca con objetos, principalmente, pueden agregar subobjetos a los mensajes, eliminar mensajes de subobjetos y completamente reorganizar los contenidos. Cuenta con un generador de análisis independiente y separado que se encarga de la transformación de texto plano con el modelo de objetos y luego vuelve al texto plano otra vez. Lee el mensaje de correo electrónico como texto plano de un archivo o de otra fuente, el texto se analiza para producir la estructura del objeto de mensaje de correo electrónico.

Chardet: esta biblioteca es un puerto del código de autodetección de Mozilla, se utiliza para la codificación de un archivo en específico [6], los tipos de codificación consisten en tomar una secuencia de bytes en principales codificaciones de caracteres desconocidos e intentar determinar la codificación.

Realizando una valoración de las características de las herramientas ya mencionadas, expuestas en el presente epígrafe se llega a la conclusión de que se utilizarán Podofo, Pypdf, SGMLOP, *Email* y Chardet por ser estas herramientas libres. Permiten utilizar su código fuente y poseen funcionalidades que se encuentran bien documentadas, además de no encontrarse bajo términos de ninguna licencia GPL.

1.2. Herramientas de extracción de contenido a nivel nacional

En el ámbito nacional se estudiará Centro de Estudios de Reconocimiento de Patrones y Minería de Datos (CERPAMID) para extraer sus características fundamentales.

CERPAMID:

Se centran en la investigación básica y aplicada en el área del Reconocimiento de Patrones y su aplicación a la Minería de Datos y Textos. Realizan investigaciones que incluyen el desarrollo de algoritmos para el procesamiento y análisis de grandes volúmenes de información estructurada o textual.

En el desarrollo de la solución se tendrá en cuenta CERPAMID, ya que posee sistemas que realizan funciones similares al procesamiento y análisis de grandes volúmenes de información estructurada o

textual, sirviendo como guía en la realización del módulo *Pre-processor*.

1.3. Tecnologías a utilizar

Es de suma importancia al inicio del desarrollo de un software realizar la selección de las tecnologías a utilizar, previendo el exceso del tiempo requerido y de los costos, así como un descenso de la calidad del producto. Analizar las características particulares de la aplicación a desarrollar, así como los beneficios que aportaría el uso de las herramientas, son elementos importantes a tener en cuenta durante el proceso de selección, debido a que MOCIC es un software que utiliza como base Ubuntu GNU/Linux como sistema operativo.

1.3.1. Codificación de caracteres

¿Por qué es necesario?

La información de codificación inadecuada no sólo perjudica la capacidad de lectura de un texto que se visualiza, además puede significar que sus datos no se encuentren en una búsqueda y que no se puedan procesar de manera confiable.

¿Qué es la codificación de caracteres?

Las palabras y las oraciones de un texto se crean a partir de caracteres. Algunos ejemplos de caracteres incluyen la letra latina á. Los caracteres se agrupan en un conjunto de caracteres (también denominados repertorio). Luego, se denomina un conjunto de caracteres codificados cuando a cada carácter se le asigna un número en particular, denominado punto de codificación. Estos puntos de codificación se representarán en la computadora por uno o más bytes. La codificación de caracteres es la vía para descifrar el código. Es un conjunto de mapeos entre los bytes que representan los números en la computadora y los caracteres ubicados en el conjunto de caracteres codificados. Sin la codificación, los datos no salieran legibles [7]. Dentro de los tipos de codificación más utilizados se encuentran:

***American Standard Code for Information Interchange* (ASCII)**

Es un código de caracteres basado en el alfabeto latino, tal como se usa en inglés moderno y en otras lenguas occidentales. Utiliza 7 bits para representar los caracteres, aunque inicialmente empleaba un bit adicional (bit de paridad) que se usaba para detectar errores en la transmisión. El código ASCII define una relación entre caracteres específicos y secuencias de bits; además de reservar unos cuantos códigos de control para el procesador de textos, y no define ningún mecanismo para describir la estructura o la apariencia del texto en un documento; estos asuntos están especificados por otros lenguajes como los lenguajes de etiquetas.

ASCII Extendido

Se denomina ASCII extendido a cualquier juego de caracteres de 8 bits en el cual los códigos 32 a 126 (0×20 a $0\times7E$) coinciden con los caracteres imprimibles de ASCII, así como los caracteres comúnmente llamados “de espacio”, estos son los códigos de control de 8 a 13 (0×08 a $0\times0D$). Las codificaciones de ASCII extendido utilizan además parte o la totalidad de los códigos superiores a 128 para codificar caracteres adicionales a los caracteres imprimibles ASCII.

Unicode

Es un estándar industrial cuyo objetivo es proporcionar el medio por el cual un texto en cualquier forma e idioma pueda ser codificado para el uso informático. Se ha convertido en el más extenso y completo esquema de codificación de caracteres, siendo el más dominante en la internacionalización y adaptación local del software informático. El estándar ha sido implementado en un número considerable de tecnologías recientes, que incluyen XML, Java y sistemas operativos modernos.

Unicode define dos métodos de localización de caracteres:

- La codificación *Unicode Transformation Format* (UTF).
- La codificación *Universal Character Set* (UCS).

Dichas codificaciones incluyen:

- UTF-7: codificación relativamente poco popular de 7 bits, a menudo considerada obsoleta.

- UTF-8: codificación de 8 bits de longitud variable.
- UCS-2: codificación de 16 bits de longitud fija que solamente permite el “mapeo” o la búsqueda en la Plana Básica Multilengüe.
- UTF-16: codificación de 16 bits de longitud variable.
- UCS-4 y UTF-32: codificaciones de 32 bits de longitud fija que son funcionalmente idénticas.
- UTF-EBCDIC: codificación poco difundida creada para sistemas basados en EBCDIC.

Codificaciones de carácter popular, ver Anexo C.

1.3.2. Lenguaje de Programación

Es un conjunto de sintaxis y reglas semánticas que definen los programas del computador. Es una técnica estándar de comunicación para entregarle instrucciones al computador. Brinda la capacidad al programador de especificarle a la PC, que tipo de datos actúan y que acciones tomar bajo una variada gama de circunstancias, utilizando un lenguaje relativamente próximo al lenguaje humano [8]. La elección del lenguaje de programación para el desarrollo se sustentó en las particularidades de la solución que se va a desarrollar. En el desarrollo de aplicaciones en sistemas GNU/Linux algunos de los lenguajes de programación más usados son: C, C++ , Perl, Java, y Python.

C++

Entre las características para comprender la potencialidad de C++ y su elección por parte del grupo de desarrollo se destacan:

- Provee bibliotecas necesarias para el desarrollo de la solución las cuales se puede integrar con Python.
- Posee gran utilidad en proyectos grandes, debido a su estructura orientada a objetos, lo que favorece en gran medida el desarrollo.

- A su vez presenta una gran ventaja respecto a otros lenguajes como Perl o Python debido al hecho de ser compilado y la necesidad de búsqueda de un mayor rendimiento para satisfacer necesidades de los usuarios finales.
- La reutilización de código es otra de las potencialidades que ofrece C++, permitiendo disminuir en ocasiones el tiempo de desarrollo.

Python

Un paso del proceso de categorización de MOCIC es la extracción de contenidos de un documento o archivo. Los módulos que necesitan extraer contenidos utilizan un programa desarrollado en Python, que no posee una arquitectura correcta para integrarse con MOCIC. El desarrollo de un nuevo programa que solucione completamente este problema implica que se desarrolle utilizando el lenguaje de programación Python. Dicho lenguaje es de *scripting* independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

Características del lenguaje [9]

Propósito general: se pueden crear todo tipo de programas. No es un lenguaje creado específicamente para la web, aunque entre sus posibilidades sí se encuentra el desarrollo de páginas.

Multiplataforma: hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.

Interpretado: no se compila el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador. En ciertos casos, cuando se ejecuta por primera vez un código, se producen unos *bytecodes* que se guardan en el sistema y que sirven para acelerar la compilación implícita que realiza el intérprete cada vez que se ejecuta el mismo código.

Interactivo: dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudar a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.

Orientado a Objetos: la programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables.

Funciones y bibliotecas: dispone de muchas funciones incorporadas en el propio lenguaje, como el tratamiento de *strings*, números y archivos. Además, existen muchas bibliotecas que se pueden utilizar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red o cosas tan interesantes como crear archivos comprimidos en .zip.

Sintaxis clara: posee una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. En muchos lenguajes, para separar porciones de código, se utilizan elementos como las llaves o las palabras clave *begin* y *end*. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.

Ventajas [10]

- Es un lenguaje muy expresivo, los programas Python son muy compactos: un programa Python suele ser más corto que su equivalente en lenguajes como C.
- Legible. La sintaxis de Python es muy elegante y permite la escritura de programas cuya lectura resulta sencilla.
- Ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje.
- El entorno de ejecución de Python detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información abundante para detectarlos y corregirlos.
- Python puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objetos.
- Posee un abundante juego de estructuras de datos que se pueden manipular de modo sencillo.

1.3.3. Herramientas case

Visual Paradigm

Es una herramienta de diseño, que hace uso del UML, este admite todos los diagramas UML. Produce documentación del sistema en varios formatos como PDF y HTML. Los desarrolladores pueden diseñar documentación del sistema con una plantilla de diseño. Los analistas de sistemas pueden estimar las consecuencias de los cambios con los diagramas de análisis de impacto, tales como la matriz y el diagrama de análisis. Visual Paradigm no es sólo enfocarse en cómo muchos diagramas se pueden dibujar, pero lo fácil que puede crear, modificar y diseñar estos diagramas ayuda a aumentar la eficiencia del sistema de análisis y diseño de manera significativa. Proporciona una plataforma de modelado colaborativo para el trabajo en equipo. Con las características de colaboración en equipo, sus miembros pueden ver y editar el mismo proyecto, o el mismo esquema, incluso de forma simultánea. Todos los cambios se almacenan en el servidor de Visual Paradigm en función de revisión. Proporciona una plataforma ampliable para que los desarrolladores puedan agregar funciones al mismo, ellos pueden hacer referencia a los *plugin* de guiar el desarrollo, a construir sus propios *plugin* para leer, actualizar, recuperar y eliminar los diagramas y los elementos del modelo [11].

1.3.4. Sistema de control de versiones

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal característica es mantener la historia de los cambios y modificaciones que se han realizado sobre ellos a lo largo del tiempo. De esta forma, el sistema es capaz de “recordar” las versiones antiguas de los datos, lo que permite examinar el histórico de cambios o recuperar versiones anteriores de un fichero, incluso aunque haya sido borrado. Los sistemas de control de versiones son ampliamente utilizados en los proyectos de desarrollo de software, para mantener las versiones del código fuente. No obstante, su aplicación no está limitada a esta actividad, sino que permiten gestionar documentos, imágenes y ficheros de todo tipo [12].

Normalmente, consiste en una copia maestra en un repositorio central, y un programa cliente con el que cada usuario sincroniza su copia local. Esto permite compartir los cambios sobre un mismo conjunto de

ficheros. Además, el repositorio guarda registro de los cambios realizados por cada usuario, y permite volver a un estado anterior en caso de necesidad.

Características del SVN [12]

- Mantiene versiones no sólo de archivos, sino también de directorios.
- Mantiene versiones de los meta-datos asociados a los directorios.
- Mantiene la historia de todas las operaciones de cada elemento, incluyendo la copia, cambio de directorio o de nombre.
- Atomicidad de las actualizaciones. Una lista de cambios constituye una única transacción o actualización del repositorio. Esta característica minimiza el riesgo de que aparezcan inconsistencias entre distintas partes del repositorio.
- Permite tanto ficheros de texto como de binarios.
- Mejor uso del ancho de banda, ya que en las transacciones se transmiten sólo las diferencias y no los archivos completos.
- Mayor eficiencia en la creación de ramas y etiquetas que en *Concurrent Versions System* (CVS).

Ventajas que proporciona el SVN

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Maneja eficientemente archivos binarios (a diferencia de CVS que los trata internamente como si fueran de texto).
- Permite selectivamente el bloqueo de archivos. Se usa en archivos binarios que, al no poder fusionarse fácilmente, conviene que no sean editados por más de una persona a la vez.

- Cuando se usa integrado a Apache permite utilizar todas las opciones que este servidor provee a la hora de autentificar archivos (SQL, *Lightweight Directory Access Protocol* (LDAP), *Pluggable Authentication Modules* (PAM)).
- Protocolo WebDAV/DeltaV para el protocolo de red.
- Versiona todos los ficheros guardando comprimidas sus diferencias.
- Usa la biblioteca *Apache Portable Runtime*, que permite portar la capa de red a varios sistemas operativos.

Desventajas [12]

- El manejo de cambio de nombres de archivos no es completo. Lo maneja como la suma de una operación de copia y una de borrado.
- No resuelve el problema de aplicar repetidamente parches entre ramas, no facilita llevar la cuenta de qué cambios se han realizado. Esto se resuelve siendo cuidadoso con los mensajes de *commit*.

1.3.5. Metodología de desarrollo

El desarrollo de software no es una tarea fácil, por lo que una adecuada selección de la metodología de desarrollo a utilizar requiere meritoria atención para lograr el éxito final y cumplir con las expectativas planteadas. Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para la producción de software. En la actualidad su uso es imprescindible para obtener mejores resultados en la organización y planificación del trabajo, contribuyendo a la obtención de un producto de calidad, que satisfaga tanto a los clientes como a los desarrolladores. Las metodologías clásicas tradicionales, no facilitan el desarrollo rápido de aplicaciones, pues requieren de una excesiva cantidad de documentación y no se adaptan a los cambios por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos pueden variar fácilmente. Las metodologías no ágiles, son aplicables cuando el equipo de desarrollo es numeroso, mientras que las metodologías ágiles son usadas en equipos pequeños. Estas se centran en el factor humano y el producto de software; es decir, le dan mayor valor al individuo, a la colaboración entre el cliente y el equipo de desarrollo y definen un proceso incremental de desarrollo con iteraciones cortas [13].

Metodología Scrum:

Se basa en los principios de inspección continua, adaptación, autogestión e innovación. Para la implantación de una gestión ágil de proyecto con este modelo es de especial importancia que, el equipo trabaje en un espacio común para maximizar la comunicación [14]

Un proyecto administrado mediante Scrum se desarrolla de forma iterativa e incremental organizándose en iteraciones, llamadas *sprints*. Al iniciar cada iteración, un equipo multifuncional selecciona los requisitos del cliente de una lista priorizada (ProductBackLog) y propone los requisitos más prioritarios a desarrollar en ella y se comprometen a terminar los elementos al final de la iteración. Cada ciclo o iteración termina con una pieza de software ejecutable que incorpora una nueva funcionalidad [15].

Ventajas [16]

- **Cumplimento de expectativas:** el cliente establece sus expectativas indicando el valor que le aporta cada requisito-historia del proyecto, el equipo los estima y con esta información el *Product Owner* establece su prioridad. De manera regular, en las demos de *Sprint* el *Product Owner* comprueba que efectivamente los requisitos se han cumplido.
- **Flexibilidad a cambios:** alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- **Reducción del *Time to Market*:** el cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- **Mayor calidad del software:** la metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- **Mayor productividad:** se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- **Maximiza el retorno de la inversión:** producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.
- **Predicciones de tiempos:** mediante esta metodología se conoce la velocidad media del equipo por

sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el *Backlog*.

- **Reducción de riesgos:** el hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

Desventajas [17]

- No genera toda la evidencia o documentación de otras metodologías.
- No es apto para todos los proyectos.
- Tal vez sea necesario complementarlo con otros procesos *Extreme Pogramming* (XP).

Metodología *Extreme Pogramming* (XP)

Se basa en la simplicidad, comunicación y realimentación o reutilización del código desarrollado [18]. Promueve el trabajo en equipo, preocupándose en todo momento del aprendizaje de los desarrolladores. Esta metodología es la adecuada para los proyectos con requisitos imprecisos, volátiles y con un riesgo técnico excesivo.

El ciclo de vida ideal de la metodología XP está compuesto por las siguientes seis fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y muerte del proyecto. Tiene definidos los siguientes roles: programador, cliente, encargado de pruebas, encargado de seguimiento, entrenador, consultor y gestor. Sus características fundamentales son:

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto.

- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir una nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código: reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- Simplicidad en el código: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir una nueva funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.
- La simplicidad y la comunicación son extraordinariamente complementarias. Con más comunicación resulta más fácil identificar qué se debe y qué no se debe hacer. Cuanto más simple es el sistema, menos tendrá que comunicar sobre este, lo que lleva a una comunicación más completa, especialmente si se puede reducir el equipo de programadores.

Ventajas

- Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para los clientes, ya conocen las fechas de entrega de funcionalidades. Vital para su negocio.
- Permite definir en cada iteración cuales son los objetivos de la siguiente.
- Permite la retroalimentación.

- La presión está a lo largo de todo el proyecto y no en una entrega final.

Desventajas [19]

- Delimitar el alcance del proyecto con el cliente.

Metodología ágil SXP:

Es una fusión de las metodologías SCRUM y XP la cual permite mejorar la actividad productiva fomentando el desarrollo de la creatividad, aumentando el nivel de preocupación y responsabilidad de los miembros del equipo y ayudando al líder del proyecto a ejercer mejor control sobre el mismo. SXP está enfocada para proyectos de pequeños equipos de trabajo, con requisitos imprecisos, muy cambiantes, donde existe un alto riesgo técnico y se orienta una entrega rápida de resultados y una alta flexibilidad. Ayuda a que el equipo trabaje unido, en la misma dirección y con un objetivo claro. Permite además seguir el avance de las tareas a realizar, ayudando a los líderes a ver día a día cómo funciona el trabajo [20].

Consta de 4 fases principales: Planificación-Definición donde se establece la visión, se fijan las expectativas y se realiza el aseguramiento del financiamiento del proyecto. Desarrollo, es donde se realiza la implementación del sistema hasta que este listo para ser entregado. Entrega, puesta en marcha y por último Mantenimiento, donde se realiza la transferencia al cliente. De cada una de ellas se despliegan 7 flujos de trabajo: concepción inicial, captura de requisitos, diseño con metáforas, implementación, prueba, entrega de la documentación, soporte e investigación, el cual se utiliza por el equipo de desarrollo cuando sea necesario. De estos flujos se realizan numerosas actividades tales como el levantamiento de requisitos, la priorización de la lista de reserva del producto, definición de las historias de usuario, diseño, implementación, planificación de las iteraciones y las actividades que se van a realizar para lograr el producto, pruebas, además de las tareas necesarias para realizar las investigaciones para documentar todo el proceso, ver Figura 1.1.

Justificación de la Metodología Seleccionada

Realizando una valoración de las características de las metodologías de desarrollo expuestas en el presente epígrafe, se llega a la conclusión de que se utilizará SXP como metodología de desarrollo de

software. Esta metodología brinda una estrategia basada en la unión de dos de las metodologías ágiles más representativas actualmente, posibilitando una guía segura hacia el cumplimiento del objetivo trazado.

Figura 1.1: Fases y flujos de trabajo de la metodología SXP, tomada de [20].

1.3.6. Lenguaje de Modelado Unificado UML

Es el lenguaje gráfico para modelado de sistemas con tecnología orientada a objeto que permite especificar, visualizar, construir y documentar. Permite todo el ciclo de vida de desarrollo de software: especificaciones de analistas, arquitectura, diseño, implementación e implantación. Admite además distintas áreas de aplicación tales como: sistemas distribuidos, tiempo real, aplicaciones mono proceso, sistemas de información corporativos, Banca y Finanzas, Telecomunicaciones, Defensa y Espacio, Transporte, Distribución, Electro-medicina, Ciencia. Es un lenguaje de modelado visual fácil de aprender, estándar, estable

y configurable.

Es importante destacar que UML es un “lenguaje” para especificar y no un método o un proceso, se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir. Se puede aplicar en una gran variedad de formas para admitir una metodología de desarrollo de software (tal como el Proceso Unificado de Rational) pero no especifica en sí mismo qué metodología o proceso usar [21].

1.4. Arquitectura

La calidad final de una aplicación está directamente influenciada por la arquitectura de la misma. El pobre rendimiento, la insuficiente mantenibilidad y baja disponibilidad, son algunos de los problemas usualmente causados por arquitecturas defectuosas o mal diseñadas [22]. La arquitectura del software es una representación abstracta de la estructura de la aplicación y el comportamiento basado en las limitaciones de los atributos de calidad que se considera que producen operativamente. Todos los sistemas tienen una arquitectura. Consta de dos estados (tal como está, a-ser). Es la justificación de cómo las responsabilidades se reparten entre las partes, las políticas y mecanismos que coordinan las interacciones entre dichas partes, ya que colaboran para cumplir el propósito del sistema; Es a la vez la partición de un sistema en sus elementos significativos y la organización e integración de esos elementos en un todo coherente. Arquitectura de software debe apoyar variabilidad, la prescindibilidad, la portabilidad, escalabilidad y la reutilización en el sistema a desarrollar [23].

1.5. Sistema gestor de base de datos DBMS

Los *Database Management System* (DBMS) son un tipo de software muy específico, con la funcionalidad de servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. El propósito general de los DBMS es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

PostgresSQL

Es un programa de código abierto, que está dirigido por una comunidad de desarrolladores llamada *PostgreSQL Global Development Group* (PGDG). Comienza su desarrollo en el año 1982 con el proyecto Ingres en la Universidad de Berkeley. Entre sus principales características se tiene la alta concurrencia, la amplia variedad de tipos nativos, y diversas funciones más específicas [24].

Ventajas [24]

- **Instalación ilimitada:** no se puede demandar a una empresa por instalarlo en más ordenadores de los que la licencia permite, ya que no hay costo asociado a la licencia de software. Esto permite un negocio más rentable con instalaciones a gran escala.
- **Ahorros considerables de costos de operación:** PostgreSQL ha sido diseñado para tener un mantenimiento y ajuste menor que los productos de proveedores comerciales, conservando todas las características, estabilidad y rendimiento.
- **Estabilidad y confiabilidad:** no se han presentado caídas de la base de datos.
- **Extensible:** el código fuente está disponible de forma gratuita, para que quien necesite extender o personalizar el programa pueda hacerlo sin costes.
- **Multiplataforma:** está disponible en casi cualquier Unix, con 34 plataformas en la última versión estable, además de una versión nativa de Windows en estado de prueba.
- **Diseñado para ambientes de alto volumen:** utilizando una estrategia de almacenamiento de filas llamada *Multi Version Concurrency Control* (MVCC), consigue mejor respuesta en grandes volúmenes. Además, MVCC permite a los accesos de solo lectura continuar leyendo datos consistentes durante la actualización de registros, permitiendo copias de seguridad en caliente, herramientas gráficas de diseño y administración de bases de datos.
- Buen sistema de seguridad mediante la gestión de usuarios, grupos de usuarios y contraseñas.
- Gran capacidad de almacenamiento.
- **Buena escalabilidad:** es capaz de ajustarse al número de CPU y a la cantidad de memoria disponible de forma óptima, permitiendo una mayor cantidad de peticiones simultáneas a la base de datos de forma correcta.

Desventajas

- En comparación con MySQL es más lento en inserciones y actualizaciones, ya que cuenta con cabeceras de intersección que no tiene MySQL.
- Consultoría en línea: existen foros oficiales, pero no una ayuda obligatoria.
- Consume más recursos que MySQL.
- La sintaxis de algunos de sus comandos o sentencias no es intuitiva.

MongoDB

Es un sistema de base de datos multiplataforma orientado a documentos, de esquema libre. Está escrito en C++, lo que le confiere cierta cercanía a los recursos de hardware de la máquina, lo que permite que sea rápido a la hora de ejecutar sus tareas. Es un software de licencia libre. Funciona en sistemas operativos Windows, GNU/Linux, OS X y Solaris [25].

Ventajas [25]

- **Modelo de datos basado en documento:** la unidad básica de almacenamiento es análoga a JSON, Python diccionarios, hashes de Ruby. Se trata de una rica estructura de datos capaz de almacenar las matrices y otros documentos. Esto significa que a menudo puede representar en una sola entidad una construcción de lo que requeriría varias tablas para representar correctamente en una base de datos relacional.
- **Profunda capacidad de consulta:** MongoDB permite consultas dinámicas en los documentos que utilizan un lenguaje de consulta basado en documentos, que es casi tan poderoso como *Structured Query Language* (SQL).
- **No hay ninguna migración de esquema:** desde MongoDB es el esquema libre, el código define el esquema.
- **Mejor rendimiento:** el modelo de documento con frecuencia no necesita uniones, no los admite, utiliza archivos asignados en memoria y un modelo de consistencia diferente.
- Un camino claro a la escalabilidad horizontal.

Conclusiones

Con el objetivo de proveer un acercamiento al objeto de estudio y campo de acción definido en la introducción del documento, se realizó un análisis de las herramientas de extracción de contenido. Se seleccionaron las tecnologías adecuadas, basándose en sus características y utilidades para el desarrollo de la aplicación, sentando las bases para el comienzo de un desarrollo eficiente. En el presente capítulo se arribaron a las siguientes conclusiones.

- Las herramientas estudiadas servirán para resolver parcialmente el problema planteado ya que, con ellas se podrá extraer los contenidos de los documentos de tipo PDF y *Email Markup Language* (EML).
- No se identificó ninguna herramienta para la extracción de contenidos de algunos tipos de documentos como son: DOC, DOCX, ODT, PPT, PPTX, XLS, XLSX.
- El lenguaje de programación Python es potente para el trabajo con archivos y cadenas de texto siendo así el más adecuado para el desarrollo de la solución.
- La metodología SXP es la más adecuada para el desarrollo del presente trabajo.

Capítulo 2

Características del sistema.

Introducción

En el presente capítulo se abordan los aspectos relacionados con el diseño de la solución. Se identifican los requisitos funcionales y no funcionales que debe cumplir el módulo. Se exponen las características del módulo *Pre-processor*, a partir de la descripción de los diagramas de vista lógica de la arquitectura, diagrama de clase del diseño, diagrama de modelo de datos y diagrama de paquetes. Basándose en los principios y las reglas que sigue la metodología SXP, se definen las historias de usuario y tareas de ingeniería asociadas a la fase de desarrollo. Se sientan las bases para la implementación de la solución propuesta utilizando las tecnologías y herramientas definidas en el capítulo anterior.

2.1. Descripción del problema

La extracción de contenidos se realiza por un programa el cual está unido al módulo *Recover*, lo que implica total dependencia hacia el mismo. Como consecuencia de esto existe poca flexibilidad en la arquitectura de MOCIC. Existe un alto acoplamiento entre los componentes que intervienen en la extracción de contenidos. Algunas de las herramientas utilizadas poseen licencia GPL, la cual es una licencia viral. No permite la inclusión de otros formatos de documentos que actualmente el módulo no posee. No se acopla adecuadamente al protocolo definido por el proyecto MOCIC.

2.2. Solución propuesta

Con el desarrollo de un módulo que permita la extracción de los contenidos de un documento para MOCIC se alcanzaría un mejor funcionamiento en el proceso de categorización. Además de ser un componente

más de MOCIC, podría brindar una mayor flexibilidad a la arquitectura del mismo, debido a que el proceso de extracción sería extensible capaz de ejecutarse de una manera distribuida. Dicho proceso mostraría dependencia en su ejecución con respecto a todos los componentes que interactúan en el mismo. Agregando las mejoras de que en el proceso se utilicen herramientas adecuadas, un correcto funcionamiento y una mejor integración con la arquitectura de MOCIC.

Como se muestra en la Figura 2.1 el proceso de extracción comienza cuando el módulo *Recover* mediante el protocolo de comunicación de MOCIC, envía un mensaje al *Controller* con la ubicación del documento, luego dicho componente le envía el documento al *Pre-processor*. De acuerdo a la solución propuesta el módulo de pre-procesamiento procederá a realizar la extracción de los contenidos en cualquier servidor que utilice este servicio, debido a la ejecución distribuida que poseerá el mismo. Dicho componente estará compuesto por dos partes fundamentales, de las cuales dependen el éxito y la organización del proceso de extracción. Primeramente el programa de extracción de contenidos y en segundo lugar el *Controller* que se encargará de administrar y organizar el proceso de ejecución de dicho programa.

El proceso de extracción se realizará en distintos servidores y llevará a cabo las operaciones necesarias con la información del documento, las cuales son:

- Almacenar en el módulo *Store* de manera estructurada la información del documento procesado.
- Almacenar en la base de datos toda la información del documento procesado.

El proceso de extracción seguirá una configuración específica basada en parámetros variables que poseerá el módulo en su funcionamiento, esto será posible con la inclusión de un fichero de configuración para configurar dichos parámetros.

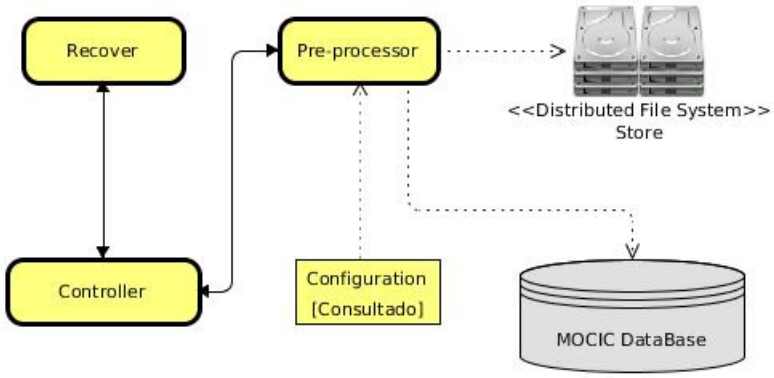

Figura 2.1: Diagrama de la arquitectura propuesta del componente *Pre-processor*.

2.3. Captura de requisitos

La captura de los requisitos es una parte fundamental en el desarrollo del software, ya que representa la necesidad del cliente en un lenguaje funcional para la construcción del mismo. Es esencial comprender perfectamente los requisitos del sistema, para que un desarrollo de software tenga éxito. Independientemente de lo bien diseñado o codificado que esté un programa, si se ha analizado y especificado pobremente, no se obtendrá un software bien desarrollado.

Un requisito funcional define el comportamiento interno del software: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas. Son complementados por los requisitos no funcionales, que se enfocan en cambio en el diseño o la implementación [18].

2.3.1. Requisitos funcionales

RF1 Realizar la extracción de texto, imágenes y enlaces de un documento.

RF2 Almacenar en la base de datos la información del documento procesado.

RF3 Almacenar en el módulo *Store* los contenidos del documento procesado.

RF4 Realizar trazas del proceso de extracción de contenido.

RF5 Comunicar el módulo a MOCIC mediante el módulo *Controller*.

2.3.2. Requisitos no funcionales

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que éstos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar [18].

- **Usabilidad:** debe ser intuitivo, requiriendo un mínimo de esfuerzo por parte del usuario.
- **Arquitectura:**
  - debe permitir incluir nuevos formatos a procesar.
  - debe ejecutarse como un servicio en Linux.
- **Flexibilidad:** debe cargar fichero de configuración con todos los parámetros de la aplicación.
- **Rendimiento:** debe mantener un rendimiento estable, utilizando un mínimo de recursos.
- **Portabilidad:** debe ser empaquetado en un .deb que permita la instalación de forma sencilla.
- **Entorno:** para ejecutar el software se requiere como mínimo un ordenador Pentium 4 con 1GB de RAM y Sistema Operativo Ubuntu 10.04.

2.4. Historias de usuarios y las tareas de ingeniería

La comunicación y el diseño simple son factores importantes que predominan en la metodología SXP. Para ello se crean las historias de usuario donde se representan los requerimientos del sistema. Cada historia de usuario incluye una o varias tareas que responden a la solución de un requerimiento, asignándole la prioridad que tienen y los usuarios que se encargan de desarrollarlas.

A continuación se muestran las historias de usuario y las principales tareas ingenieriles que dan solución a los requisitos funcionales **RF1**, **RF2**, **RF3** y **RF5**. La descripción de las restantes historias de usuario que complementan la solución del requisito **RF1** se encuentran en el Anexo D.

	Historia de Usuario
Número: PRE-1	**Nombre Historia Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo *Portable Document Format* (PDF).
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodriguez Reyes	**Iteración Asignada:**Sprint 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Dada la entrada del mensaje del protocolo de comunicación, se obtiene la dirección del archivo PDF, seguidamente se le extrae los contenidos, ya sea texto, imagen o enlace.	
Observaciones:	

Cuadro 2.1: Historia de Usuario 1.

	Tarea Ingeniería
Número: 1	**Número Historia Usuario:** PRE-1
Nombre Tarea: Investigar biblioteca de Podofo para la realización de la extracción de texto de un PDF.	
Tipo de tarea: Investigativa.	**Puntos Estimados:** 1
Fecha Inicio: 20/02/2012	**Fecha Fin:** 22/02/2012
Continúa en la próxima página	

Programador Responsable: Alexis Rodriguez Reyes
Descripción: Se realiza un estudio de la biblioteca podofo y sus funcionalidades.

Cuadro 2.2: Tarea de Ingeniería 1.

		Historia de Usuario
Número: PRE-2	**Nombre Historia Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo HTML.	
Modificación de historia de Usuario Número: Ninguna		
Usuario: Alexis Rodriguez Reyes	**Iteración Asignada:** Sprint 1	
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 1 Semana	
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1	
Descripción: Dada la entrada del mensaje del protocolo de comunicación, se obtiene la dirección del archivo HTML, seguidamente se le extrae los contenidos, ya sea texto, imagen o enlace.		
Observaciones:		

Cuadro 2.3: Historia de Usuario 2.

	Tarea Ingeniería
Número: 2	**Número Historia Usuario:** PRE-2
	Continúa en la próxima página

Nombre Tarea: Implementación del plugin HtmlExtract.	
Tipo de tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 02/10/2012	**Fecha Fin:** 02/14/2012
Programador Responsable: Alexis Rodriguez Reyes	
Descripción: Se implementa el plugin HtmlExtract para la extracción de contenidos de un archivo HTML.	

Cuadro 2.4: Tarea de Ingeniería 2.

		Historia de Usuario
Número: PRE-3	**Nombre Historia Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo DOC (Documento word).	
Modificación de historia de Usuario Número: Ninguna		
Usuario: Alexis Rodriguez Reyes		**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Muy Alta		**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alto		**Puntos Reales:** 1
Descripción: Dada la entrada del mensaje del protocolo de comunicación, se obtiene la dirección del archivo DOC (Documento word), seguidamente se le extrae los contenidos, ya sea texto, imagen o enlace.		
Observaciones:		

Cuadro 2.5: Historia de Usuario 3.

	Tarea Ingeniería
Número: 3	**Número Historia Usuario:** PRE-3
Nombre Tarea: Implementación del plugin DocExtract.	
Tipo de tarea: Desarrollo.	**Puntos Estimados:** 1
Fecha Inicio: 02/10/2012	**Fecha Fin:** 02/14/2012
Programador Responsable: Alexis Rodriguez Reyes	
Descripción: Se implementa el plugin DocExtract para la extracción de contenidos de un archivo DOC (Documento word).	

Cuadro 2.6: Tarea de Ingeniería 3.

	Historia de Usuario
Número: PRE-7	**Nombre Historia Usuario:** Almacenar en la base de datos la información brindada por el módulo *Controller*.
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodriguez Reyes	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 Semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2
Descripción: Se realizará la copia de la información brindada del documento en la base de datos, la configuración de la base de datos tiene que estar en un fichero de configuración.	
Observaciones:	

Cuadro 2.7: Historia de Usuario 7.

Tarea Ingeniería	
Número: 4	**Número Historia Usuario:** PRE-7
Nombre Tarea: Implementación del acceso a datos del módulo *Pre-processor*.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 21/03/2012	**Fecha Fin:** 27/03/2012
Programador Responsable: Alexis Rodriguez Reyes	
Descripción: Se implementaran las clases necesarias para el acceso a dato utilizando la biblioteca libpqxx para PostgresSQL.	

Cuadro 2.8: Tarea de Ingeniería 4.

Historia de Usuario	
Número: PRE-8	**Nombre Historia Usuario:** Almacenar en el módulo *Store* la información del documento a procesar.
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodriguez Reyes	**Iteración Asignada:** Sprint 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 Semanas
Riesgo en Desarrollo: Alto	**Puntos Reales:** 2
Descripción: Se realizará la copia de la información extraída del documento por los extractores en el *Store*, creando una estructura de carpetas, la cual va a contener una carpeta con el uid del lote del documento, dentro una carpeta con el id del documento y en la misma estarán todos los contenidos extraídos.	
Continúa en la próxima página	

Observaciones:

Cuadro 2.9: Historia de Usuario 8.

	Tarea Ingeniería
Número: 5	**Número Historia Usuario:** PRE-8
Nombre de Tarea: Investigar funcionamiento y características del nuevo protocolo de comunicación definido por MOCIC.	
Tipo de tarea: Investigativa	**Puntos Estimados:** 1
Fecha Inicio: 08/04/2012	**Fecha Fin:** 10/04/2012
Programador Responsable: Alexis Rodriguez Reyes	
Descripción: Se realizará un estudio del funcionamiento del nuevo protocolo de comunicación, sus principales características y los mensajes del mismo.	

Cuadro 2.10: Tarea de Ingeniería 5.

	Historia de Usuario
Número: PRE-9	**Nombre Historia Usuario:** Comunicar el módulo a MOCIC mediante el Controller.
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodriguez Reyes	**Iteración Asignada:** Sprint 2
Prioridad en Negocio: Alta	**Puntos Estimados:** 2 Semanas
Continúa en la próxima página	

Riesgo en Desarrollo: Alto	**Puntos Reales:** 2
Descripción: Se realizará una integración con MOCIC de manera que exista una comunicación entre los módulos que interactúan con el *Pre-processor*. El módulo *Controller* debe comunicarse con el *Pre-processor* mediante el protocolo de comunicación, el *Pre-processor* debe realizar una conexión con la base de datos de MOCIC y debe comunicarse con el módulo *Store*, para realizar copias de la información del documento procesado.	
Observaciones:	

Cuadro 2.11: Historia de Usuario 9.

		Tarea Ingeniería
Número: 6	**Número Historia Usuario:** PRE-9	
Nombre Tarea: Estudiar nueva arquitectura de MOCIC.		
Tipo de tarea: Investigativa		**Puntos Estimados:** 1
Fecha Inicio: 24/04/2012		**Fecha Fin:** 30/04/2012
Programador Responsable: Alexis Rodriguez Reyes		
Descripción: Se hará un estudio acerca de la nueva arquitectura definida por MOCIC viendo el funcionamiento de los procesos y las relaciones de los módulos existentes.		

Cuadro 2.12: Tarea de Ingeniería 6.

2.5. Vista lógica de la arquitectura propuesta

En la Figura 2.2 se muestra la vista lógica de la arquitectura propuesta para el módulo a desarrollar, la cual está compuesta por dos capas: capa de procesamiento de la aplicación y administración de datos.

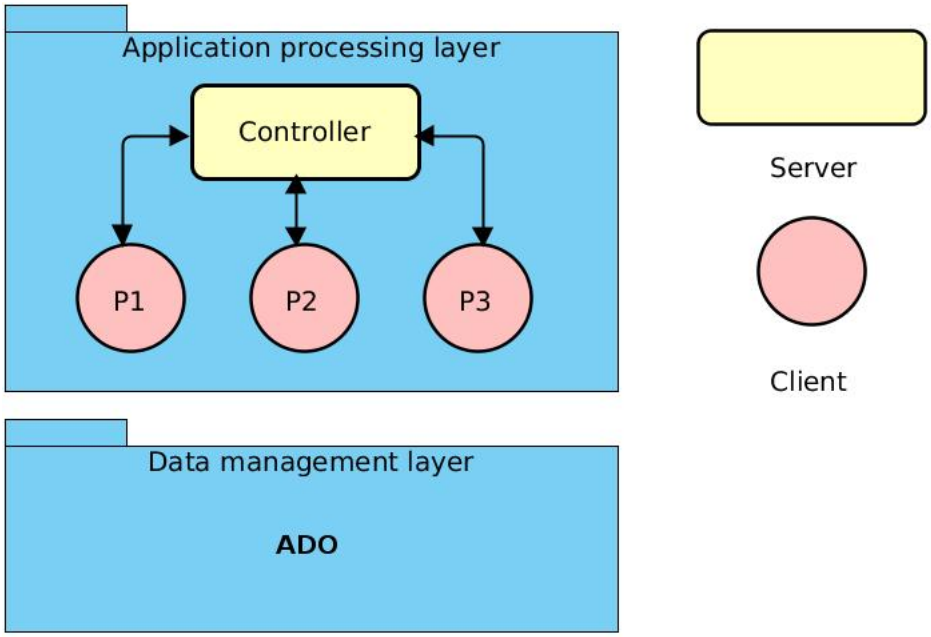

Figura 2.2: Vista lógica de la arquitectura propuesta.

La capa de procesamiento muestra la estructura lógica que va a seguir la aplicación. El modelo de una arquitectura cliente-servidor refleja el comportamiento que tendrán los componentes del módulo *Preprocessor*. Dicho modelo está compuesto por el *Controller* que será el servidor y los clientes (P) que serán servicios de extracción ejecutados en distintos nodos. El *Controller* y todos los clientes se ejecutarán concurrentemente ya que existirán varios clientes realizando el mismo proceso simultáneamente. Además existirá un intercambio de datos continuos en el proceso de ejecución del módulo. La capa de administración de datos engloba todas las operaciones relacionadas con la base de datos. Estas operaciones se llevan acabo independientemente del servidor, cada cliente o extractor copia la información necesaria de un documento en la base de datos de MOCIC y en el módulo *Store*.

2.6. Diagrama de paquetes

Con el objetivo de tener una visión de la estructura y organización que seguirá la solución propuesta en la Figura 2.3 se muestra una descomposición de la totalidad del módulo *Pre-processor* en paquetes y sus relaciones de dependencia.

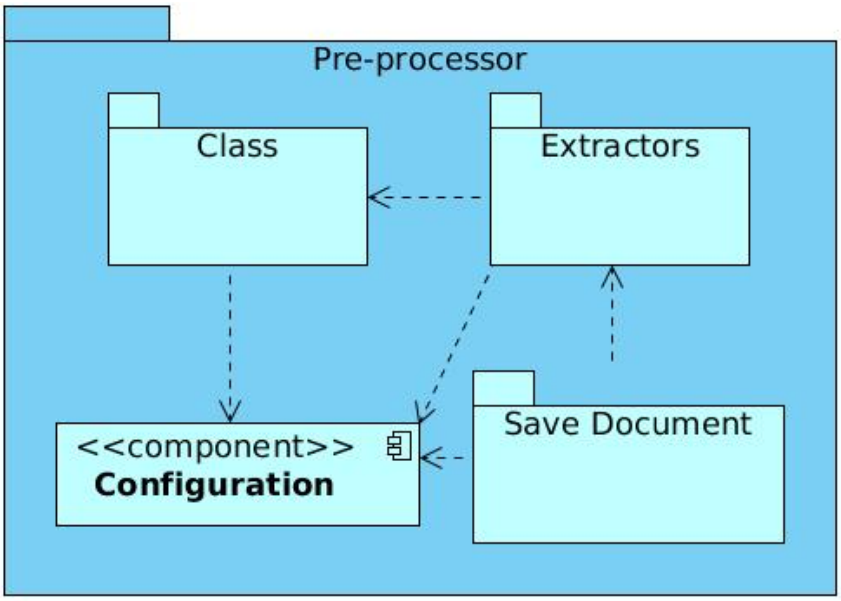

Figura 2.3: Diagrama de Paquetes.

- ***Class*:** contiene todas las clases del modelo.
- ***Extractors*:** contiene todas las clases que representan los extractores que se deseen implementar.
- ***Save Document*:** contiene todas las clases que representan los tipos de copias que se le realizarán al documento.
- ***Component Configuration*:** archivo de configuración el cual guarda todos los parámetros variables del módulo a desarrollar.

2.7. Arquitectura seleccionada

Arquitectura basada en componentes:

La arquitectura basada en componentes es una rama de la Ingeniería de Software. Trata con énfasis la descomposición del software en componentes funcionales. La descomposición permite convertir componentes preexistentes en piezas más grandes de software. Este proceso de construcción de una pieza de software, da origen al principio de reutilización del software. Esto promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro. La estructura de la arquitectura basada en componentes contempla tres partes [26]:

- **El nombre de los componentes:** el nombre de un componente debe ser la identificación de la funcionalidad y uso que tiene como software. Generalmente, los desarrolladores usan algún tipo de convención que facilite la identificación de componentes, especialmente, cuando se trabaja en proyectos de gran envergadura.
- **La interfaz de los componentes:** es el área de intercambio (*input-output*) entre el interior y el exterior de un componente de software. La interfaz es quien permite acceder a los datos y funcionalidades que estén especificadas en el interior del componente (acceder funcionalmente, no a su especificación). Adicional a la interfaz se encuentra la documentación que muestra la información sobre cómo utilizar un componente.
- **Cuerpo y código de implementación:** es la parte del componente que provee la forma (implementación) sobre la cual un fragmento del componente realiza sus servicios y funcionalidades. Este es el área que debe cumplir con el principio de encapsulación.

Se realizó la selección de una arquitectura basada en componentes principalmente por la necesidad de permitir incluir nuevos formatos de documentos a procesar. Dicha arquitectura permite tratar en el tiempo de ejecución todos los extractores[1] admitidos en forma de componentes, lo que permite evitar la repetida integración de cada extractor de un formato con la solución propuesta. Posibilitando en un futuro dada la existencia de un formato desconocido, una inclusión factible del mismo.

[1] Clases que implementan la extracción de contenidos de diferentes formatos de archivos.

2.8. Patrones de diseño

Con el objetivo de realizar un correcto diseño de las clases del sistema a desarrollar y basándose en estrategias a seguir se utilizaron los siguientes patrones de diseño:

***General Responsibility Asignment Software Patterns* (GRASP):**

Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

- **Experto:** la asignación de una responsabilidad a la clase que cuenta con la información necesaria para cumplir una funcionalidad en específico permite tener un código correcto y funcional a la hora de utilizarlo, este patrón se evidencia en algunas clases como `Extractor` y `DocExtractor`.
- **Creador:** en el desarrollo del módulo de pre-procesamiento, se utilizó dicho patrón para las clases que tienen como responsabilidad crear instancias de otras clases, por ejemplo: `DocExtractor` que tiene como responsabilidad crear una instancia de la clase `Document`, `Extractor` y `Message` para poder realizar completamente el funcionamiento del proceso de extracción.
- **Bajo Acoplamiento:** con la utilización de la arquitectura basada en componentes, se utilizó este patrón en las clases `Extractor` y `SaveDoc`, las cuales utilizan interfaces que poseen un comportamiento específico. Posibilitando la asignación de responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible.
- **Alta cohesión:** generalmente una clase que esté altamente cohesionada tiene bajo acoplamiento desde el punto de vista de clases, ya que posee la menor cantidad de dependencias posibles con otras clases y a su vez posee responsabilidades moderadas en un área funcional y colabora con las otras para llevar acabo las tareas. En las clases `Extractor` (Posee todos los métodos relacionados con respecto a la extracción), `Savedoc` (Posee todas las operaciones relacionadas con la copia de los contenidos de un documento) se evidencia la utilización de dicho patrón.

Patrones *Gan of Four* (GOF):

Proponen una forma de reutilizar la experiencia de los desarrolladores. Están basados en la recopilación del conocimiento de los expertos en desarrollo de software. Es una experiencia real, probada y que funciona. Es historia y ayuda a no cometer los mismos errores.

- ***Prototype*:** la especificación de los tipos de objetos a crear por medio de una instancia prototípica fue la condición necesaria para el uso de este patrón en la implementación de las clases del módulo de pre-procesamiento, específicamente en la clase `DocExtractor`, ya que la misma posee la funcionalidad de crear todos los extractores existentes, para de acuerdo a un documento en específico, realizar su tarea.
- **Singlenton:** asegura que solo exista una instancia de la clase que lo utilice y provee un método para acceder a la misma. Todas las clases o objetos que manejan una instancia de una clase que emplea el patrón Singlenton, hacen uso de la misma instancia. Esto se evidencia en la clase `Singlenton` de la cual se obtiene un único objeto de la configuración que sigue el módulo *Pre-procesor*. Las clases que manejan el objeto de configuración solo invocarán al objeto creado.
- ***Interface:*** garantiza un bajo acoplamiento y logra un comportamiento independientemente entre las clases que lo utilizan, dichas clases son: `Extractor` y `SaveDoc`.

2.9. Diagrama de clases del diseño

Con el objetivo de mostrar la estructura y organización que seguirá el módulo *Pre-processor* en su desarrollo, en la Figura 2.4 se refleja de una manera explícita las clases diseñadas para el desarrollo de la solución, así como las relaciones y dependencias que poseen.

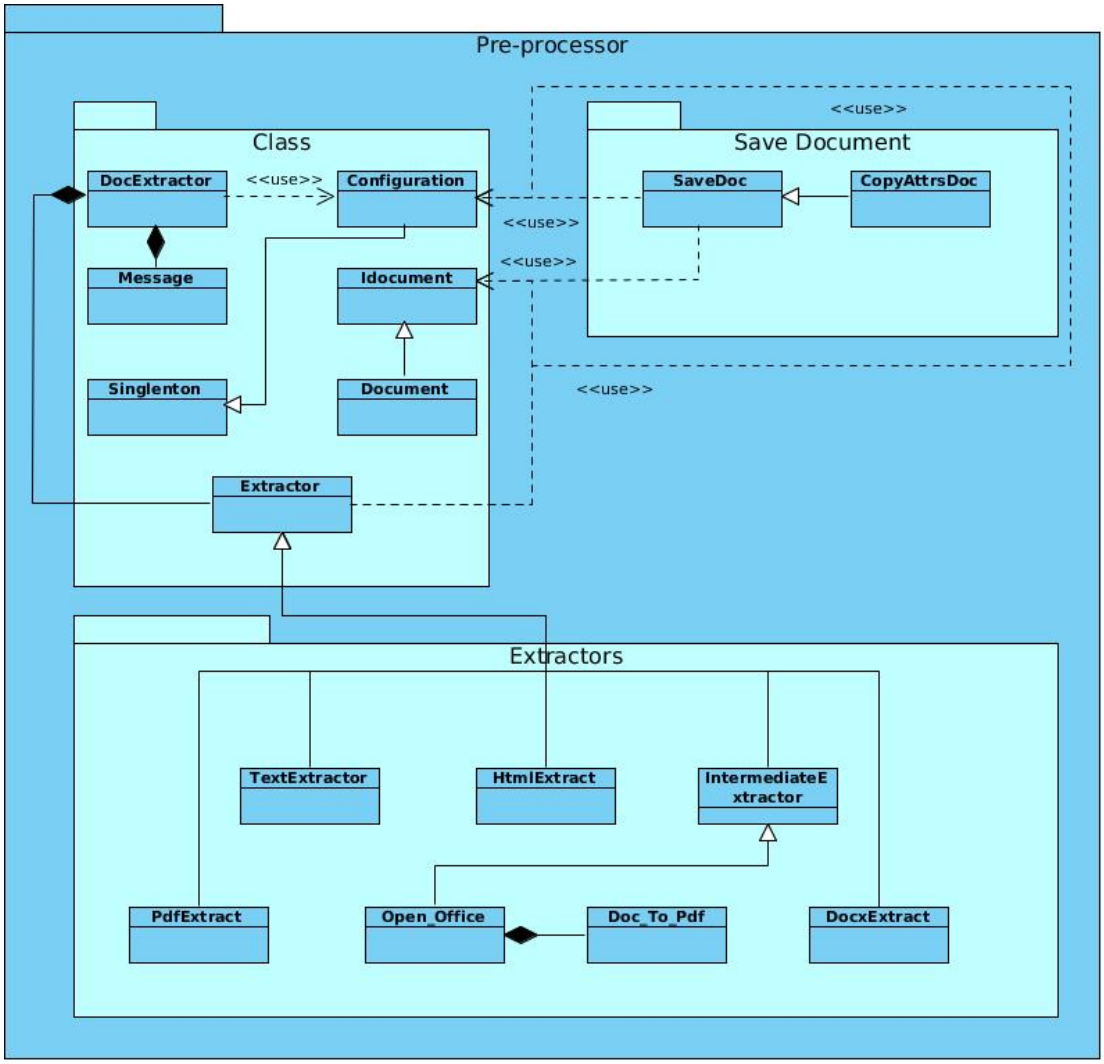

Figura 2.4: Diagrama de clases del diseño.

- `DocExtractor:` es la clase principal, encargada de realizar todo el proceso de extracción de contenidos de un documento.
- `Message:` se encarga de obtener el formato del mensaje pasado por el protocolo de comunicación, el cual contiene toda la información del documento a procesar.
- `Configuration:` encargada de obtener el objeto configuración, el cual tiene los parámetros configurables de la aplicación.
- `Singlenton:` implementa el patrón singlenton para la utilización de la clase `Configuration`.

- `Extractor:` es la clase encargada de realizar la extracción de contenidos de un documento, de ella heredan todos los tipos de extractores de formatos específicos que se deseen implementar como son: archivos HTML, PDF.
- `Idocument:` interfaz que posee una estructura y comportamiento para las clases que hacen uso de la misma, reduciendo el acoplamiento y dependencias entre ellas.
- `Document:` posee la estructura y los atributos específicos que posee un documento.
- `Savedoc:` esta clase es la encargada de realizar todas las operaciones de copia de un documento de la cual heredan su comportamiento, las clases copias como `CopyAttrsDoc`.
- `CopyAttrsDoc:` hereda el comportamiento de `SaveDoc`. Las clases que heredan de `SaveDoc` serían los tipos de copias a realizar.
- `HtmlExtract:` clase que realiza la extracción para archivos HTML.
- `TextExtractor:` clase que realiza la extracción para archivos de texto plano.
- `PDFExtract:` clase que realiza la extracción para archivos PDF.
- `DocxExtract:` clase que realiza la extracción para archivos comprimidos.
- `IntermediateExtractor:` se encarga de implementar un extractor, cuando en la extracción de documentos se necesita convertir el documento a un formato intermedio.
- `Open_Office:` se encarga de obtener los contenidos de los documentos que admita el Open Office.
- `Doc_To_Pdf:` se encarga de convertir un documento de un extractor intermedio a PDF.

2.9.1. Descripción de las clases

Nombre de la Clase: `DocExtractor`	
Tipo de Clase: controladora	
Atributo	**Tipo**
_cfg	ConfigParser()
_Log	String
Continúa en la próxima página	

_pathToCopyContents	String
Para cada responsabilidad:	
Nombre:	Descripción:
GetObjOfCopy(document,OutP,Msg)	Devuelve un objeto del fichero de configuración que contiene el tipo de copia que se realizará en el proceso de extracción.
GetMsgProtocol(argv[])	Devuelve un objeto de la clase `Message` dado el arreglo de parámetros pasados.
Get_Stractors(document)	Devuelve los extractores que permite dado el documento.
Begin()	Comienza el proceso de extracción.

Cuadro 2.13: Descripción de la clase `DocExtractor`.

Nombre de la Clase: `Message`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_timestamp	int
_uri	String
_communiti	String
_uid_bash	String
_id_doc	String
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(argv[])	Constructor de la clase, crea un objeto `Message`.
parser()	Devuelve un objeto con todos los parámetros del mensaje.

Cuadro 2.14: Descripción de la clase `Message`.

Nombre de la Clase: `Configuration`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_config	String
Para cada responsabilidad:	
Nombre:	Descripción:
Load()	Inicializa un objeto ConfigParser()(biblioteca de python) para guardar los parámetros variables de la aplicación.
Get_Attribute_Section()	Obtiene un atributo de una sección especificada del objeto de configuración.

Cuadro 2.15: Descripción de la clase `Configuration`.

Nombre de la Clase: `Singlenton`	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
__init__()	Constructor de la clase, crea un objeto `Singlenton`.
__call__()	Método de llamada para crear una instancia.

Cuadro 2.16: Descripción de la clase `Singlenton`.

Nombre de la Clase: `Extractor`
Tipo de Clase: controladora
Continúa en la próxima página

Atributo	**Tipo**
document	Document
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(Pdocument)	Constructor de la clase, crea un objeto `Extractor`.
get_mime()	Obtiene el mime(tipo de formato del documento) que permite procesar.
ExtractText()	Método que retorna el texto extraído de un documento.
ExtractLinks()	Obtiene los enlaces que posee el documento.
ExtractImages()	Obtiene las imágenes que posee el documento .
get_raw(Pdocument)	Devuelve una cadena con el contenido del documento que se pasa por parámetro.

Cuadro 2.17: Descripción de la clase `Extractor`.

Nombre de la Clase: `Idocument`	
Tipo de Clase: interfaz	
Atributo	**Tipo**
_document	String
_base_url	String
_links	list[]
_images	list[]
_text	String
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(Ppath,Pbase_url)	Constructor de la clase, crea un objeto `Idocument`.
get_mimeDoc()	Obtiene el mime(tipo de formato del documento) que permite procesar.
Continúa en la próxima página	

Cuadro 2.18: Descripción de la clase `Idocument`.

Nombre de la Clase: `Document`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_path	String
_base_url	String
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(Ppath,Pbase_url)	Constructor de la clase, crea un objeto `Document`.
get_mimeDoc()	Obtiene el mime(tipo de formato del documento) que se va a procesar.

Cuadro 2.19: Descripción de la clase `Document`.

Nombre de la Clase: `SaveDoc`	
Tipo de Clase: controlador	
Atributo	**Tipo**
_Idocument	Idocument
_OutPut	String
_Message	Message
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(document,OutPut,Message)	Constructor de la clase, crea un objeto `SaveDoc`.

Continúa en la próxima página

CopyText()	Método que copia el texto extraído de un documento.
CopyLinks()	Método que copia los enlaces que posee el documento.
CopyImages()	Copia las imágenes que posee el documento.

Cuadro 2.20: Descripción de la clase `SaveDoc`.

Nombre de la Clase: `CopyAttrsDoc`	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(document,OutPut,Message)	Constructor de la clase, crea un objeto `CopyAttrsDoc`.
CopyText()	Método que copia el texto extraído de un documento.
CopyLinks()	Método que copia los enlaces que posee el documento.
CopyImages()	Copia las imágenes que posee el documento.

Cuadro 2.21: Descripción de la clase `CopyAttrsDoc`.

Nombre de la Clase: `HtmlExtract`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_links	list[]
_images	list[]
_text	String
Para cada responsabilidad	
Continúa en la próxima página	

Nombre:	Descripción:
__init__(document)	Constructor de la clase, crea un objeto `HtmlExtract`.
ExtractText()	Método que retorna el texto extraído de un documento HTML.
ExtractLinks()	Obtiene los enlaces que posee el documento HTML.
ExtractImages()	Obtiene las imágenes que posee el documento.
get_mime()	Obtiene el mime(tipo de formato del documento) del documento.

Cuadro 2.22: Descripción de la clase `HtmlExtract`.

Nombre de la Clase: `TextExtractor`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_links	list[]
_images	list[]
_text	String
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(document)	Constructor de la clase, crea un objeto `TextExtractor`.
ExtractText()	Método que retorna el texto extraído de un documento de texto plano.
ExtractLinks()	Obtiene los enlaces que posee el documento de texto plano.
ExtractImages()	Obtiene las imágenes que posee el documento de texto plano.
get_mime()	Obtiene el mime(tipo de formato del documento) del documento de texto plano.
Continúa en la próxima página	

Cuadro 2.23: Descripción de la clase `TextExtractor`.

Nombre de la Clase: `PDFExtract`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_links	list[]
_images	list[]
_text	String
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(document)	Constructor de la clase, crea un objeto `PDFExtract`.
ExtractText()	Método que retorna el texto extraído de un documento PDF.
ExtractLinks()	Obtiene los enlaces que posee el documento PDF.
ExtractImages()	Obtiene las imágenes que posee el documento PDF.
get_mime()	Obtiene el mime(tipo de formato del documento) del documento PDF.

Cuadro 2.24: Descripción de la clase `PDFExtract`.

Nombre de la Clase: `DocxExtract`	
Tipo de Clase: entidad	
Atributo	**Tipo**
_links	list[]

Continúa en la próxima página

_images	list[]
_text	String
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(document)	Constructor de la clase, crea un objeto `DocxExtract`.
ExtractText()	Método que retorna el texto extraído de un documento comprimido.
ExtractLinks()	Obtiene los enlaces que posee el documento comprimido.
ExtractImages()	Obtiene las imágenes que posee el documento comprimido.
get_mime()	Obtiene el mime(tipo de formato del documento) del documento comprimido.

Cuadro 2.25: Descripción de la clase `DocxExtract`.

Nombre de la Clase: `IntermediateExtractor`	
Tipo de Clase: controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
__init__(document)	Constructor de la clase, crea un objeto `IntermediateExtractor`.
_get_intermediate(document)	Convierte el documento que se pasa por parámetro a un nuevo documento en el formato intermedio y devuelve un objeto extractor para el nuevo documento.
ExtractText()	Método que retorna el texto extraído del documento, dependiendo del objeto extractor intermedio.
Continúa en la próxima página	

ExtractLinks()	Obtiene los enlaces que posee el documento, dependiendo del objeto Extractor intermedio.
ExtractImages()	Obtiene las imágenes que posee el documento, dependiendo del objeto extractor intermedio.
get_mime()	Obtiene el mime(tipo de formato del documento) del documento dependiendo del objeto extractor intermedio

Cuadro 2.26: Descripción de `IntermediateExtractor`.

Nombre de la Clase: `Open_Office`	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	Descripción:
_get_intermediate(document)	Convierte el documento que se pasa por parámetro a un nuevo documento en el formato intermedio y devuelve un objeto extractor para el nuevo documento.
get_mime()	Obtiene los mimes(tipo de formato del documento) de los documentos que soporta el Open Office.

Cuadro 2.27: Descripción de la clase `Open_Office`.

Nombre de la Clase: `Doc_To_Pdf`	
Tipo de Clase: entidad	
Atributo	**Tipo**
Para cada responsabilidad:	
Continúa en la próxima página	

Nombre:	Descripción:
__init__(server, port, timeout)	Constructor de la clase, crea un objeto `Doc_To_Pdf`.
convert(document)	Convierte el documento pasado por parámetro a PDF .

Cuadro 2.28: Descripción de la clase `Doc_To_Pdf`.

2.10. Modelo de datos

Es un conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Comprenden aspectos relacionados con: estructuras y tipos de datos, operaciones y restricciones [27].

En la Figura 2.5 se muestra el modelo de datos correspondiente al módulo de pre-procesamiento para MOCIC. Se refleja la estructura y relaciones que poseen las tablas que almacenan las características necesarias requeridas por el sistema, de acuerdo a los requisitos funcionales.

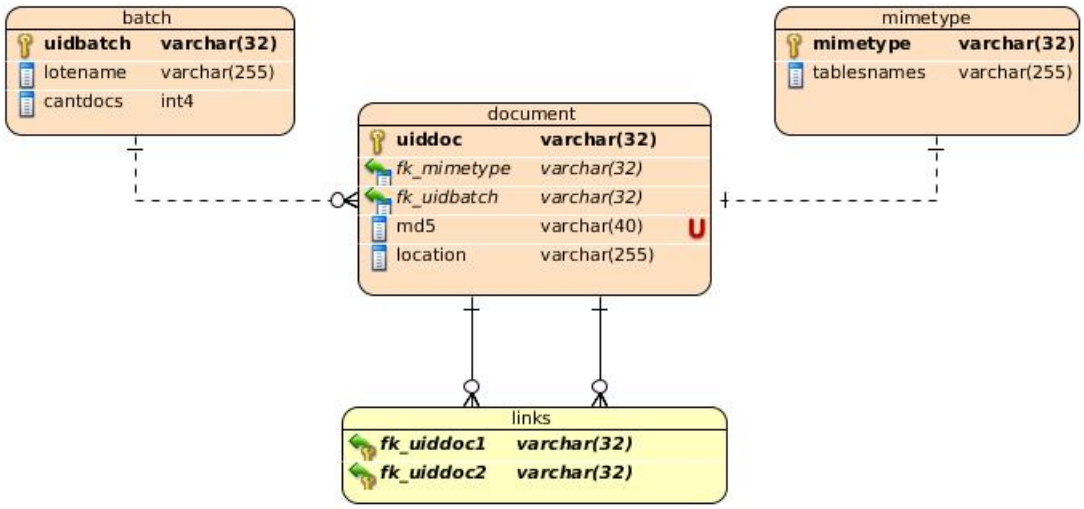

Figura 2.5: Modelo de datos.

2.10.1. Descripción de las clases

Nombre de la Clase: `document`		
Descripción: Tabla que guarda los datos del documento.		
Atributo	**Tipo**	**Descripción**
uiddoc	varchar(32)	Identificador único de la tabla.
fk_mimetype	varchar(32)	Llave foránea por la relación con la clase mimetype.
fk_uidbatch	varchar(32)	Llave foránea por la relación con la clase batch.
md5	varchar(40)	Código sha1.
location	varchar(255)	Uri donde se encuentra el documento.

Cuadro 2.29: Descripción de la clase `document`.

Nombre de la Clase: `links`		
Descripción: Tabla que guarda los enlaces extraídos del documento.		
Atributo	**Tipo**	**Descripción**
fk_uiddoc1	varchar(32)	Identificador del documento.
fk_uiddoc2	varchar(32)	Identificador del documento.

Cuadro 2.30: Descripción de la clase `links`.

Nombre de la Clase: `batch`		
Descripción: Contiene toda la información con respecto al lote.		
Atributo	**Tipo**	**Descripción**
uidbash	varchar(32)	Identificador de la tabla.
lotename	varchar(255)	Nombre del lote donde se encuentra el documento.

Continúa en la próxima página

cantdocs	int4	Cantidad de documentos.

Cuadro 2.31: Descripción de la clase `batch`.

Nombre de la Clase: `mimetype`		
Descripción: Contiene el formato del documento guardado documento.		
Atributo	**Tipo**	**Descripción**
mimetype	varchar(32)	Id.

Cuadro 2.32: Descripción de la clase `mimetype`.

Conclusiones

Con el objetivo de proveer una mayor comprensión de las características del componente a desarrollar, se realizó una descripción detallada de la solución propuesta. Se llevó a cabo un diseño utilizando las herramientas definidas en el primer capítulo, así como la planificación de entrega del software a partir de las historias de usuario definidas. Obteniéndose las siguientes conclusiones.

- Se obtuvo una visión del funcionamiento y estructura que tendrá el módulo a desarrollar en ejecución mediante la vista lógica de la arquitectura propuesta.
- Se realizó una correcta definición de las características del sistema mediante la captura de los requisitos funcionales y no funcionales.
- La arquitectura basada en componentes posibilita la inclusión de nuevos formatos de documentos abstrayéndose de la lógica de negocio existente.

- Mediante la utilización de los patrones adecuados se obtuvo un diseño de clases estructurado y organizado.

Capítulo 3

Implementación y pruebas.

Introducción

Para realizar la comprobación y validar las funcionalidades del sistema, y de esta forma saber si está apto para ser liberado, es necesario en el ciclo de vida realizarle pruebas al mismo. También para obtener un entorno de desarrollo satisfactorio es muy importante realizar pruebas desde el inicio de la implementación del software. Esto garantiza que la inclusión de una nueva funcionalidad no afecte el comportamiento esperado de otra ya implementada y con correcto funcionamiento.

Las pruebas funcionales, constituyen un conjunto de condiciones o variables que le permiten a un analista determinar si el requisito de una aplicación es parcial o totalmente satisfactorio. En estas pruebas, el cliente es el máximo responsable de verificar y priorizar la corrección de las inconformidades detectadas.

En el presente capítulo se documentan las pruebas realizadas al módulo de pre-procesamiento para MOCIC. En la sección siguiente se documentarán las pruebas funcionales que fueron realizadas en cada iteración, siendo necesario que cada prueba funcional fuera superada satisfactoriamente para la implementación de las siguientes funcionalidades.

3.1. Pruebas funcionales

	Prueba funcional
Código de prueba funcional: PRE-1-1	**Nombre de Historia de Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo PDF.
Nombre de la persona que realiza la prueba funcional: Alexis Rodriguez Reyes	
Continúa en la próxima página	

Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la extracción de todos los contenidos de un archivo PDF.
Condiciones de ejecución: ■ El módulo *Pre-processor* debe conocer los parámetros del mensaje del protocolo de comunicación. ■ El documento a realizar la prueba debe ser de tipo PDF.
Entradas / **Pasos de ejecución:** ■ Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros: • -t: Valor del timestamp de cuando se empezó a procesar el documento. • -u: Dirección del documento PDF a procesar. • -l: Identificador (uid) del lote. • -i: Identificador del documento (sha1). • -c: Comunidad (opcional). .
Resultado esperado: Se guarda en una carpeta los contenidos (textos, enlaces, imágenes) del archivo PDF.
Evaluación: Prueba satisfactoria

	Prueba funcional
Código de prueba funcional: PRE-2-1	**Nombre de Historia de Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo HTML.
Nombre de la persona que realiza la prueba funcional: Alexis Rodriguez Reyes	
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la extracción de todos los contenidos de un archivo HTML.	
Continúa en la próxima página	

Condiciones de ejecución:

- El módulo *Pre-processor* debe conocer los parámetros del mensaje del protocolo de comunicación.
- El documento a realizar la prueba debe ser del tipo de los formatos permitidos.

Entradas / **Pasos de ejecución:**

- Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros:
  - -t: Valor del timestamp de cuando se empezó a procesar el documento.
  - -u: Dirección del documento HTML a procesar.
  - -l: Identificador (uid) del lote.
  - -i: Identificador del documento(sha1).
  - -c: Comunidad (opcional).

.

Resultado esperado: Se guarda en una carpeta los contenidos (textos, enlaces, imágenes) del archivo HTML.

Evaluación: Prueba satisfactoria

	Prueba funcional
Código de prueba funcional: PRE-3-1	**Nombre de Historia de Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo DOC (Documento word).
Nombre de la persona que realiza la prueba funcional: Alexis Rodriguez Reyes	
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la extracción de todos los contenidos de un archivo DOC (Documento word).	

Continúa en la próxima página

Condiciones de ejecución: ■ El módulo *Pre-processor* debe conocer los parámetros del mensaje del protocolo de comunicación. ■ El documento a realizar la prueba debe ser del tipo de los formatos permitidos.
Entradas / Pasos de ejecución: ■ Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros: • -t: Valor del timestamp de cuando se empezó a procesar el documento. • -u: Dirección del documento DOC (Documento word) a procesar. • -l: Identificador (uid) del lote. • -i: Identificador del documento (sha1). • -c: Comunidad (opcional). .
Resultado esperado: Se guarda en una carpeta los contenidos (textos, enlaces, imágenes) del archivo DOC (Documento word).
Evaluación: Prueba satisfactoria

La descripción de las restantes pruebas funcionales del requisito funcional **Realizar la extracción de texto, imágenes, enlaces del documento a procesar** se encuentran en el Anexo E.

	Prueba funcional
Código de prueba funcional: PRE-7-1	**Nombre de Historia de Usuario:** Almacenar en la base de datos la información brindada por el módulo *Controller*.
Nombre de la persona que realiza la prueba funcional: Alexis Rodriguez Reyes	
Continúa en la próxima página	

Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la copia de toda la información necesaria de un documento en la base de datos.

Condiciones de ejecución:

- La base de datos debe estar accesible.
- Existencia de la tabla documento, meta-datos y las asociadas con las mismas.
- Para la conexión a la base de datos debe existir usuario y contraseña con permisos.

Entradas / Pasos de ejecución:

- Configurar el fichero de configuración para poner los datos.
- Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros:
  - -t: Valor del timestamp de cuando se empezó a procesar el documento.
  - -u: Dirección del documento a procesar.
  - -l: Identificador (uid) del lote.
  - -i: Identificador del documento (sha1).
  - -c: Comunidad (opcional).
- Verificar la inserción de los datos.

Resultado esperado: Se guarda en la base de datos la información del documento procesado.

Evaluación: Prueba satisfactoria

	Prueba funcional
Código de prueba funcional: PRE-8-1	**Nombre de Historia de Usuario:** Almacenar en el módulo *Store* la información del documento procesado.
Nombre de la persona que realiza la prueba funcional: Alexis Rodriguez Reyes	

Continúa en la próxima página

Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la copia del documento de forma estructurada en el *Store*.
Condiciones de ejecución: ▪ Acceso a la carpeta donde se guardarán los contenidos que se encuentra en el módulo *Store*.
Entradas / **Pasos de ejecución:** ▪ Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros: • -t: Valor del timestamp de cuando se empezó a procesar el documento. • -u: Dirección del documento a procesar. • -l: Identificador (uid) del lote. • -i: Identificador del documento (sha1). • -c: Comunidad (opcional). .
Resultado esperado: Se guarda en módulo *Store* una carpeta con el nombre del uid del lote donde se encuentra el documento, dentro una carpeta nombrada con el id del documento, en la cual se va a copiar todos los contenidos extraídos por el componente *Pre-processor* dependiendo el tipo de documento.
Evaluación: Prueba satisfactoria

	Prueba funcional
Código de prueba funcional: PRE-9-1	**Nombre de Historia de Usuario:** Comunicar el módulo a MOCIC mediante el módulo *Controller*
Nombre de la persona que realiza la prueba funcional: Alexis Rodriguez Reyes	
Descripción de la Prueba: Se comprueba el proceso de extracción integrado con los módulos que interactúan con el módulo *Pre-processor*, de acuerdo a la arquitectura definida por MOCIC.	
Continúa en la próxima página	

Condiciones de ejecución: ■ El módulo *Pre-processor* se debe ejecutar como un servicio en Linux. ■ El módulo *Controller* debe mandar un mensaje cuando exista un documento a procesar.
Entradas/ Pasos de ejecución: ■ Se recibe un mensaje mediante el protocolo de comunicación con los siguientes parámetros: • -t: Valor del timestamp de cuando se empezó a procesar el documento. • -u: Dirección del documento a procesar. • -l: Identificador (uid) del lote. • -i: Identificador del documento (sha1). • -c: Comunidad (opcional).
Resultado esperado: Se guarda en la base de datos toda la información del documento y en el módulo *Store* toda la estructura de carpetas correspondientes.
Evaluación: Prueba satisfactoria

Con la realización de las pruebas funcionales mencionadas anteriormente se comprobó la completa realización de las historias de usuarios definidas en el capítulo anterior, detectándose las siguientes no conformidades.

Iteración	Prueba funcional	No conformidad
1	PRE-1-1	No se puede mostrar el contenido de las imágenes extraidas del PDF.
	PRE-2-1	El texto extraído del HTML no es legible.
	PRE-5-1	No se extraen los enlaces de un archivo EXCEL.
2	PRE-7-1	No se verifica la existencia del documento que se está procesando, en la base de datos.

Continúa en la próxima página

	PRE-8-1	No se copian correctamente de manera estructurada en carpetas, la información extraída del documento.
3	PRE-9-1	No se verifica que se hace con los documentos que no se le pudieron extraer los contenidos.

Todas las no conformidades detectadas en cada iteración se solucionaron en la siguiente, sin afectar el continuo desarrollo del módulo *Pre-processor*. La realización periódica de las pruebas fue el principal factor para la completa y correcta validación de todas las funcionalidades del componente desarrollado.

3.2. Pruebas de estrés y rendimiento

Con el objetivo de realizar un análisis del comportamiento del componente desarrollado en cuanto a rapidez, desempeño y uso de recursos en condiciones extremas, se le aplicaron pruebas de estrés y de rendimiento.

Las pruebas de estrés evalúan el comportamiento del sistema cuando es sometido a situaciones anormales en demanda de recursos, frecuencia o volumen. Algunos ejemplos de las pruebas de estrés que pueden aplicarse a un sistema son [18]:

- Evaluar el desempeño del sistema al someterlo a cantidades superiores a las normales de interrupciones por segundo.
- Elevar el volumen de datos de entrada buscando evaluar el comportamiento de las funciones de entrada.
- Diseñar escenarios que necesiten niveles máximos de memoria.

Para el desarrollo de las pruebas de estrés y de rendimiento que a continuación se documentan se utilizó una PC con las siguientes características:

- Micro: Intel (R) Pentium(R) M processor 1.86GHz
- *Random-access memory* (RAM): 2GB
- Capacidad: 80GB

En la siguiente tabla se refleja el resultado que tuvo el componente *Pre-processor* en el proceso de extracción de contenidos ante una colección de 5000 archivos.

Cantidad de archivos	**RAM(MB)**	***Central Processing Unit* (CPU)(%)**	**Tiempo(H)**
5000	320	80	3

Según el tiempo que se demoró el proceso de extracción ante la colección de datos, se concluye que el componente desarrollado admite una cantidad de 5000 documentos a procesar, extrae como promedio 27 documentos por minuto, siendo este un resultado satisfactorio de acuerdo las características de la *Personal Computer* (PC) escogida para la prueba. El proceso de ejecución del componente fue estable en cuanto a uso de CPU y de RAM durante el tiempo que conllevó la prueba realizada.

Durante las pruebas de resistencia se incluyeron algunas pruebas de rendimiento con el objetivo de comprobar la rapidez y el máximo de recursos utilizados por el componente desarrollado, así como el tiempo que se demora en extraer información de colecciones de datos de distintos archivos.

Se tomaron como muestra tres carpetas, las cuales contienen 100 archivos HTML, PDF y Doc respectivamente. En la Figura 3.1 se muestran los resultados para el proceso de extracción de contenidos de un documento en cuanto a uso de CPU, en la Figura 3.2 con respecto al uso de memoria y por último en la Figura 3.3, el tiempo que se demora en realizarse el proceso de extracción.

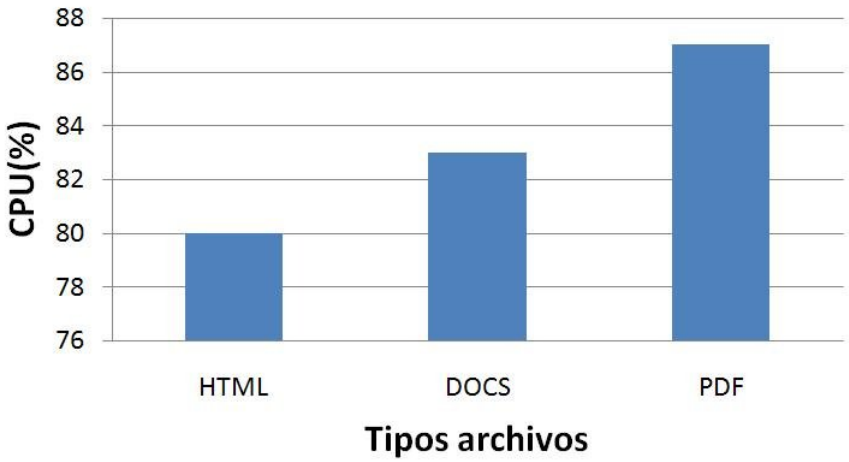

Figura 3.1: Uso de CPU.

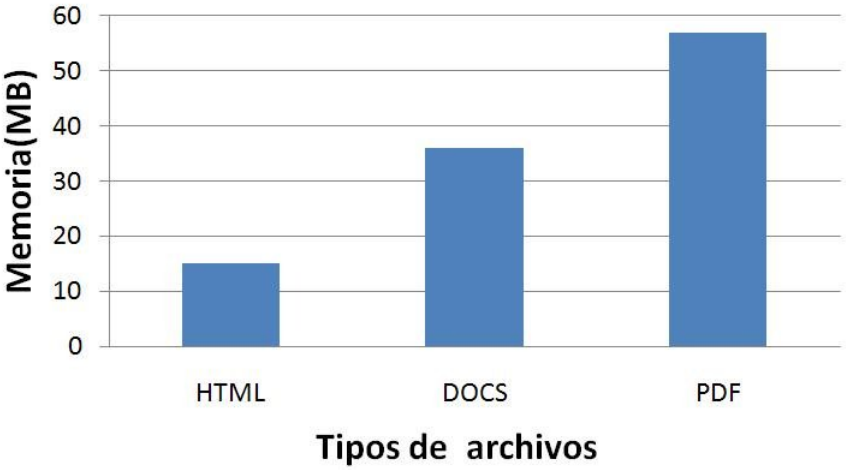

Figura 3.2: Uso de memoria.

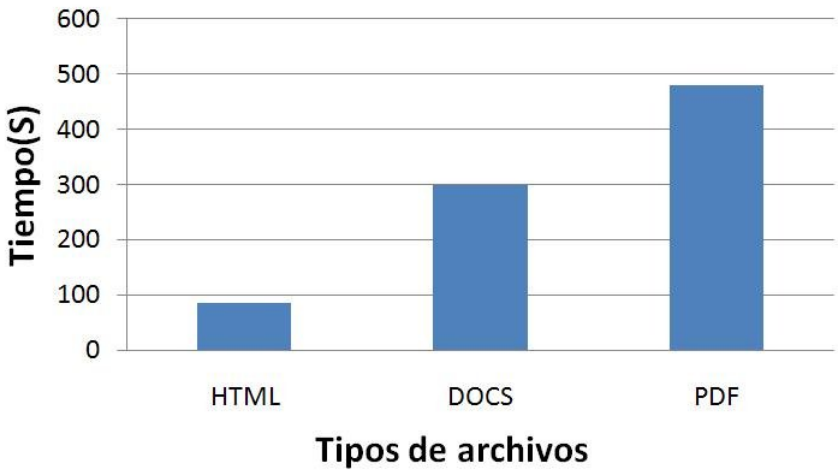

Figura 3.3: Tiempo.

Como se puede apreciar en las pruebas de rendimientos representadas anteriormente, el tiempo de extracción de contenidos es pequeño para cada tipo de archivo. Se mostró la rapidez y el desempeño que posee el proceso de extracción de archivos HTML con respecto a los otros archivos, ya que utiliza un mínimo de recursos.

3.3. Diagrama de componentes

Los diagramas de componentes modelan la vista estática de un sistema. Muestran las dependencias lógicas entre componentes de software, sean éstos código fuentes, binarios o ejecutables. Estas dependencias poseen varios tipos, que indican si son de enlace, ejecución o uso. Se consideran en este diagrama solo tipos de instancias específicas que se encuentran en el diagrama de ejecución. En la Figura 3.4 se muestra la organización interna que poseen los componentes del módulo *Pre-processor* y sus dependencias.

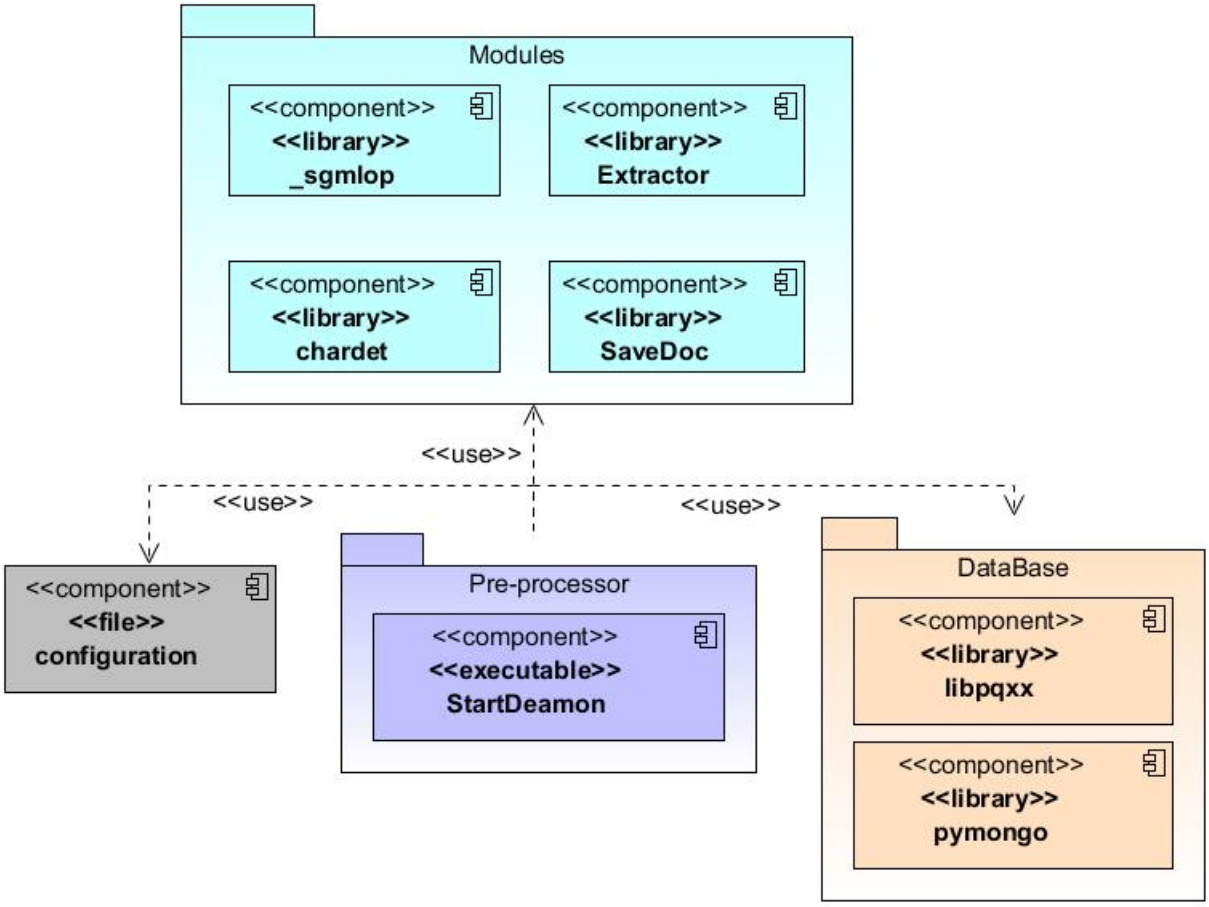

Figura 3.4: Diagrama de componentes del módulo de pre-procesamiento para MOCIC.

Configuration

Los parámetros variables que existen en el proceso de ejecución del *Pre-processor* son la causa fundamental de tener el componente de configuración, debido a que existen informaciones que pueden variar en un momento dado. Un ejemplo de esto es la ubicación exacta del módulo *Controller*, el cual posee una arquitectura cliente-servidor en conjunto con el *Pre-processor*, lo que implica que se realizará una conexión en cualquier momento ya sea de enviar o de recibir. El *Pre-processor* tiene que tener conocimiento de donde se encuentra exactamente el *Controller* para poder realizar la conexión. Existen otros parámetros como la dirección del módulo *Store*, donde se almacenará la información del documento pre-procesado y el tipo de copia que se realizará del componente SaveDoc, el cual se explicará más adelante.

El componente de configuración es un fichero que guarda los parámetros mencionados anteriormente para poder en un momento específico, en caso de que alguno varíe, modificarlo fácilmente sin afectar la implementación del módulo. Un ejemplo de esto sería: en caso de que exista un cambio de computadora, en la cual se encuentra desplegado el componente *Controller*, el *Pre-processor* necesita saber cual es el IP de la nueva computadora, para poder realizar las conexiones necesarias y continuar su funcionamiento. Pueden existir varios casos como el mencionado, que pueden variar por muchas causas. Para poder solucionar este problema sin tener que reimplementar el código, o realizar alguna modificación extensa, los ficheros de configuración permiten la actualización rápida y factible de cualquier parámetro variable que exista en el desarrollo del módulo.

Modules

Refleja los componentes utilizados para el desarrollo satisfactorio de las funcionalidades principales del *Pre-processor*, dichos componentes son:

- ***Extractor*:** realiza el proceso de extracción de contenido de acuerdo al formato que posea el documento, se utiliza para adoptar un comportamiento específico de acuerdo al tipo de documento que se le quiera extraer los contenidos, se implementarán tantos extractores como formatos de documento si es necesario, de esta forma dicho componente de acuerdo a los extractores disponibles que tenga, realizará el proceso de extracción de contenidos de un documento.
- **_sgmlop:** en el caso de algunos extractores existirán dependencias hacia otros módulos que complementen su extracción completamente, debido a que poseen funcionalidades específicas que ahorran en tiempo y desarrollo la construcción de un extractor. En el caso de _sgmlop provee un analizador de HTML, el cual posee funcionalidades que permiten utilizarlo satisfactoriamente en el desarrollo del extractor de dicho archivo, posibilitando así que sea más sencilla la implementación del mismo.
- **chardet:** como se mencionó en el Capítulo 1 existen las llamadas codificaciones de caracteres, las cuales representan la simbología que utiliza la computadora para identificar todos los caracteres que existen en un archivo de cualquier tipo. El componente chardet brinda funcionalidades que permiten detectar la codificación que posee un archivo específico.
- **SaveDoc:** realiza el proceso de almacenamiento de la información en el módulo *Store*, tendrá tam-

bién al igual que el componente Extractor un comportamiento específico, que incluso se puede reimplementar. Si existe un nuevo tipo de copia se decidirá mediante el componente de configuración el tipo de copia que se realizará.

DataBase

Representa la utilización de los componentes para realizar las operaciones con la base de datos:

- **libpqxx:** representa las funcionalidades o operaciones que permiten la comunicación con el servidor de base de datos PostgreSQL, accediendo a través del protocolo TCP/IP independientemente de donde se encuentre este.
- **pymongo:** provee las herramientas para interactuar con el servidor de base de datos MongoDB desde el lenguaje de programación Python.

StartDeamon:

Permite la ejecución como servicio (Demonio) en GNU/Linux, las conexiones con los módulos que interactúan en la arquitectura de MOCIC y la ejecución en cualquier máquina que se decida realizar el pre-procesamiento.

3.4. Diagrama de despliegue

Para comprender como se ejecutará a nivel de hardware un sistema desarrollado y tener una visión clara de la estructura del sistema en ejecución y las relaciones entre los componentes que interactúan en el mismo, se realiza el diagrama de despliegue. Dicho diagrama tiene como objetivo reflejar lo anteriormente mencionado en una vista de despliegue, representando la disposición de las instancias de los componentes de ejecución en instancias de nodos conectados por enlaces de comunicación [28]. A continuación en la Figura 3.5 se muestra la vista de despliegue del módulo de pre-procesamiento para MOCIC.

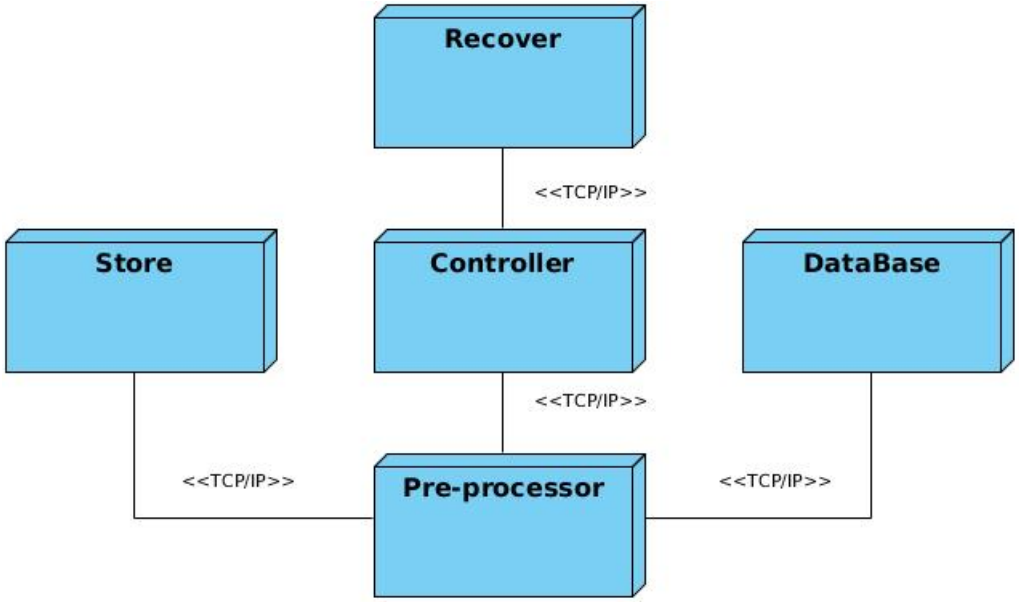

Figura 3.5: Diagrama de despliegue.

La estructura que sigue dicho diagrama esta conformada por cinco nodos los cuales se conectan mediante el protocolo TCP/IP. El proceso de comunicación entre los nodos se lleva acabo mediante el protocolo de comunicación definido por MOCIC, el cual posee mensajes parametrizados que guían el proceso de pre-procesamiento de un documento.

El nodo *Recover* envía un mensaje al nodo *Controller*, el cual posee los siguientes parámetros:

- -t: Contiene el valor del timestamp de cuando se comenzó a procesar el documento.
- -u: Dirección de donde se obtuvo el documento.
- -l: Identificador único (uid) del lote.
- -i: Identificador del documento (sha1).
- -c: Especifica el nombre de comunidad, si no se define este parámetro deben ser detectadas las comunidades para el lote que se esta procesando.

El nodo *Controller* envía el mensaje al nodo *Pre-processor* para realizarle el pre-procesamiento al documento, posteriormente el segundo mencionado salva en el nodo *DataBase* los valores de los parámetros

-t, -u, -i , [-c] y los metadatos que el mismo extrae del documento. El nodo *Pre-processor* almacena seguidamente en el nodo *Store* la carpeta con el documento pre-procesado. Específicamente crea una carpeta, si ya no existe, que tiene como nombre el valor del parámetro -l. Dentro de esta carpeta que representa el lote de documentos, crea entonces una carpeta para cada documento a pre-procesar y les pone por nombre el valor del parámetro -i. Dentro de estas carpetas copia la información del documento ya pre-procesado. Cuando el *Pre-processor* termina la copia, le envía un mensaje de respuesta al nodo *Recover* mediante el *Controller* con el parámetro -i.

El proceso de extracción de contenidos de un documento en el cual intervienen los nodos anteriormente mencionados se ejecutará de una manera distribuida. El nodo *Controller* en conjunto con el nodo *Pre-processor* poseerán una arquitectura cliente-servidor y la ejecución de estos nodos se comportará asincrónicamente debido a que existirá más de un nodo realizando el proceso a la vez.

3.5. Plan de liberación

Para estimar de forma ideal el tiempo que le tomará a los programadores la codificación de cada una de las historias de usuario es necesario una acertada planificación del proyecto. A partir de la prioridad de las historias se decide cuáles de ellas se implementarán en las primeras iteraciones, centrando la atención en las funcionalidades críticas del sistema.

Iteración	**Historia de Usuario a Implementar**	**Duración total (semanas)**
1	PRE-1,PRE-2, PRE-3, PRE-4, PRE-5, PRE-6	4
2	PRE-7, PRE-8	5
3	PRE-9, PRE-10	5

Conclusiones

Con la finalización de las pruebas realizadas y siguiendo las pruebas funcionales documentadas, se obtuvieron las siguientes conclusiones.

- Se desarrolló un componente que permite la extracción de forma automática de los elementos de un documento para el proceso de extracción de contenido en MOCIC.
- La aplicación de las pruebas de rendimiento evidenció la efectividad del proceso de extracción en archivos HTML.
- Mediante el desarrollo de las pruebas funcionales se comprobaron y validaron todas las funcionalidades requeridas por el sistema.
- Según las pruebas de rendimiento, el componente desarrollado es estable en cuanto el uso de recursos CPU.
- Las pruebas de resistencia mostraron el buen desempeño que posee el componente desarrollado ante el procesamiento de grandes colecciones de datos.

Conclusiones

Con la realización del presente trabajo de diploma se han cumplido satisfactoriamente los objetivos trazados y se han obtenido las siguientes conclusiones:

- El estudio y análisis de los principales conceptos teóricos sirvieron de base para la selección de la metodología, herramientas y tecnologías necesarias para el desarrollo de la aplicación.
- Se obtuvo una correcta definición de las características del sistema, mediante el levantamiento de los requisitos funcionales y no funcionales.
- Se realizó un diseño organizado y estructurado mediante la correcta selección de las buenas prácticas de diseño, así como la realización de las clases del modelo de datos.
- A partir del diseño y la implementación del módulo de pre-procesamiento para MOCIC se resuelven las limitantes planteadas en el problema. Se obtiene como resultado un componente eficiente y estructurado que favorecerá el mejoramiento del proceso de categorización en MOCIC.
- Mediante el desarrollo y aplicación de las pruebas funcionales se comprobó la completa y correcta realización de las funcionalidades requeridas por el sistema.

Recomendaciones

Luego de haber finalizado el presente trabajo de diploma se recomienda:

- Profundizar en las herramientas y tecnologías de extracción de contenidos, que con el continuo desarrollo de las mismas pueden surgir nuevas y mejores, las cuales podrían resolver de una mejor manera el problema.
- Agregar componentes extractores correspondientes a nuevos documentos que no se encuentren implementados en el módulo *Pre-processor*.

Lista de acrónimos

MOCIC Motor de Categorización Inteligente de Contenido:
Es un proyecto cuyo objetivo principal es categorizar, de forma automática, documentos de diversos formatos.

CERPAMID Centro de Estudios de Reconocimiento de Patrones y Minería de Datos

UCI Universidad de las Ciencias Informáticas

UML Lenguaje Unificado de Modelado :
Es un lenguaje usado para especificar, visualizar y documentar los componentes de un sistema en desarrollo orientado a objetos.

HTML *HyperText Mark Language*

CVS *Concurrent Versions System*

SQL *Structured Query Language*

PAM *Pluggable Authentication Modules*

DBMS *Database Management System*

EML *Email Markup Language*

GOF *Gan of Four*

PGDG *PostgreSQL Global Development Group*

MVCC *Multi Version Concurrency Control*

CPU *Central Processing Unit*

GNU *GNU is Not Unix*

GPL *General Public License*

PDF *Portable Document Format*

XML *Extensible Markup Languaje*

SGML *Standard Generalized Mark-up Language*

UTF *Unicode Transformation Format*

UCS *Universal Character Set*

ASCII *American Standard Code for Information Interchange*

LDAP *Lightweight Directory Access Protocol*

TCP *Transmission Control Protocolo*

XP *Extreme Pogramming*

PC *Personal Computer*

IP *Internet Protocol*

RAM *Random-access memory*

SXP *Scrum-Extreme Pogramming*

GRASP *General Responsibility Asignment Software Patterns*

PNG *Portable Network Graphics*

PPM *Portable Pixmap Format*

JPEG *Joint Photographic Experts Group*

PS PostScript :
PostScript es un potente lenguaje de programación basado en pila.

Referencias bibliográficas

[1] John F. Gantz, David Reinsel, Christopher Chute, Wolfgang Schlichting, John McArthur, Stephen Minton, Irida Xheneti, Anna Toncheva, and Alex Manfrediz. *The Expanding Digital UniverseAnna Toncheva*. Nature Publishing Group, March 2007.

[2] Linux App Finder. Popler [en línea]. 2008. Disponible en: http://linuxappfinder.com/package/poppler-utils [Accedida 10/12/2011].

[3] SourceForge. Podofo [en línea]. 2012. Disponible en: http://podofo.sourceforge.net/about.html [Accedida 10/12/2011].

[4] Python Software Foundation. SGMLOP [en línea]. 2011. Disponible en: http://pypi.python.org/pypi/sgmlop/1.1.1-20040207 [Accedida 09/12/2011].

[5] Python Software Foundation. EMAIL [en línea]. 2012. Disponible en: http://docs.python.org/library/email.html [Accedida 10/12/2011].

[6] Python Software Foundation. CHARDET [en línea]. 2011. Disponible en: http://pypi.python.org/pypi/chardet [Accedida 12/12/2011].

[7] W3C. Codificación de Caracteres [en línea]. 2011. Disponible en: http://www.w3.org/International/questions/qa-what-is-encoding.es.php [Accedida 1/01/2012].

[8] B. and Safari Tech Books Online Stroustrup. *The C++ programming language*, volume 3. Addison-Wesley Reading, MA, 1997.

[9] Python Software Foundation. Python [en línea]. 2012. Disponible en: http://www.python.org [Accedida 26/01/2012].

[10] Andrés Marzal and Isabel Gracia. *Introducción a la programación con Python*. Publicaciones de la Universitad Jaume I, 2003.

[11] Visual Paradigm. Visual Paradigm [en línea]. 2011. Disponible en: http://www.visual-paradigm.com/product/vpuml/editions/modeler.jsp [Accedida 13/02/2012].

[12] Ben Collins-Sussman, Brian W. Fitzpatrick, and C. Michael Pilato. *Version Control with Subversion* . TBA, 2006.

[13] Mauricio Rojas C and Ailin Orjuela Duarte. Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo [en línea]. June 2008. Disponible en: http://pisis.unalmed.edu.co/avances/archivos/ediciones/Edicion%20Avances%202008%202/21.pdf [Accedida 17/02/2012].

[14] proyectosagiles. SCRUM [en línea]. Disponible en: http://www.proyectosagiles.org/requisitos-de-scrum [Accedida 26/01/2012].

[15] proyectosagiles. SCRUM Ventajas [en línea]. Disponible en: http://www.proyectosagiles.org/beneficios-de-scrum [Accedida 26/01/2012].

[16] SOFTENG. Ventajas SCRUM [en línea]. 2010. Disponible en: http://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html [Accedida 26/01/2012].

[17] Itzcoalt Alvarez M. Joiz.Net. Desarrollo Ágil con SCRUM.

[18] Ph.D. R.S. Pressman. *Software Engineering A Practitioner's Approach*. McGraw Hill Publication, 2010.

[19] Maite Corbea and Ordóñez Maylin Pérez. Rodríguez. LA METODOLOGÍA XP APLICABLE AL DESARROLLO DEL SOFTWARE EDUCATIVO EN CUBA, 2007.

[20] Peñalver Ing. Gladys. METODOLOGÍA ÁGIL PARA EL DESARROLLO DE SOFTWARE [en línea]. 2008. Disponible en: http://usbvirtual.usbcali.edu.co/ijpm/images/stories/documentos/v1n2/009.pdf [Accedida 17/02/2012].

[21] Yurien Ramos Garcia. Propuesta de diseño de una aplicación Web para el Control de Trabajos de Diploma, June 2008.

[22] Yaslin Carballo Villar. Interfaz de Administración Web para Motor de Categorización Inteligente de Contenido, June 2011.

[23] Carnegie Mellon University. Arquitectura de Software [en línea]. 2012. Disponible en: http://www.sei.cmu.edu/architecture/start/glossary/community.cfm.

[24] SmallSquid S.L. PostgresSQL [en línea]. Disponible en: http://www.aplicacionesempresariales.com/ventajas-y-desventajas-de-postgresql.html [Accedida 12/03/2012].

[25] 10gen, Inc. Mongodb [en línea]. 2012. Disponible en: http://www.mongodb.org.

[26] Scribd Inc. Arquitectura basada en Componentes [en línea]. 2012. Disponible en: http://www.scribd.com/doc/14704374/Arquitectura-Basada-en-Componentes [Accedida 20/02/2012].

[27] Cavero J.M. y Marcos E. Sánchez Fúquene, D.M. *The concepts of model in information systems engineering: a proposal for an ontology of models.* Knowledge Eng., 2009.

[28] Michael Huallpara Hugo and Susana Limachi Nancy. Diagrama de Despliegue [en línea]. Disponible en: http://virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc [Accedida 20/03/2012].

Bibliografía

- David M. Beazley. *Python essential reference, fourth edition*. Pearson Education, Inc, 2009.
- ason Brittain and Ian F.M. Jones. *Python for unix and linux system administration*. O'Reilly Media, Inc., 2008.
- Doug Hellmann. *The python standard library by example*. Pearson Education, Inc., 2011.
- Jeff McNeil. *Python.2.6.Text.Processing.Beginners.Guide*. Packt Publishing, 2010.
- Roger S. Pressman. *Software Engineering*. RaghothamanSrinivasan, 2010.
- Brandon Rhodes and John Goerzen. *Foundations of python network programming 2nd edition*. Paul Manning, 2010.
- Gladys Marsi Peñalver Romero. *Metodología ágil para proyectos de software libre.* UCI, 2008.
- Ian Sommerville. *Ingeniería del Software*. Pearson Education Limited, United Kingdom, 2005.

Anexo A

Tamaño promedio de un archivo

Con el objetivo de conocer el tamaño promedio de un archivo se tomó como muestra una carpeta común de 50 ejemplares, se sumó el tamaño de cada uno de ellos obteniendo un resultado de 120MB como tamaño total. Seguidamente se dividió este resultado entre la cantidad de archivos de muestra (50) y se obtuvo un promedio de 2.4MB por cada uno, este seria el tamaño promedio de un archivo.

Anexo B

Tiempo de categorización manual de un documento

Se tomó una muestra de una carpeta la cual contenía 20 archivos, seguidamente se categorizó manualmente cada uno de estos para saber cuánto tarda una persona en categorizarlo. Cada tiempo se fue sumando hasta terminar con la carpeta de muestra y se obtuvo como resultado 984 segundo, luego se dividió 984 entre 20 que es la cantidad de archivos que contiene la carpeta y como resultado se obtuvo 49 segundos, este sería el tiempo promedio que demora una persona para categorizar un documento.

Anexo C

Codificaciones de caracter popular

- ISO 646
- ASCII
- EBCDIC
- ISO 8859:
  - ISO 8859-1, ISO 8859-2, ISO 8859-3, ISO 8859-4, ISO 8859-5, ISO 8859-6, ISO 8859-7, ISO
  - 8859-8, ISO 8859-9, ISO 8859-10, ISO 8859-11, ISO 8859-13, ISO 8859-14, ISO 8859-15, ISO 8859-16
  - CP437, CP737, CP850, CP852, CP855, CP857, CP858, CP860, CP861, CP863, CP865, CP866, CP869
- Juegos de caracteres de MS-Windows:
  - Windows-1250 para idiomas de europa central que utilizan el alfabeto latino, (polaco, checo,
  - eslovaco, húngaro, eslovenio, croata, rumano y albanés)
  - Windows-1251 para alfabetos cirílicos
  - Windows-1252 para idiomas occidentales
  - Windows-1253 para griego
  - Windows-1254 para turco
  - Windows-1255 para hebreo
  - Windows-1256 para árabe
  - Windows-1257 para idomas bálticos
  - Windows-1258 para vietnamita
- Mac OS Romano

- KOI8-R, KOI8-U, KOI7
- MIK
- Cork o T1
- ISCII
- VISCII
- Big5 (variante de Microsoft Code página 950)
    - HKSCS
- Guobiao
    - GB2312
    - GBK (Microsoft Code página 936)
    - GB18030
- Shift JIS para japonés (Microsoft Code página 932)
- EUC-KR para coreano (Microsoft Code página 949)
- ISO-2022 y EUC para juegos de caracteres CJK
- Unicode (incluyendo los subjuegos 16-bit)
- ANSEL o ISO/IEC 6937

Anexo D

Historias de usuarios

	Historia de Usuario
Número: PRE-4	**Nombre Historia Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo PPT(Presentación de PowerPoint).
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodríguez Reyes	**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Dada la entrada del mensaje del protocolo de comunicación, se obtiene la dirección del archivo PPT(Presentación de PowerPoint), seguidamente se le extrae los contenidos, ya sea texto, imagen o enlace.	
Observaciones:	

Cuadro D.1: Historia de Usuario 4.

	Historia de Usuario
Número: PRE-5	**Nombre Historia Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo EXCEL.
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodríguez Reyes	**Iteración Asignada:** Sprint 1
Continúa en la próxima página	

Prioridad en Negocio: Muy Alta	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alto	**Puntos Reales:** 1
Descripción: Dada la entrada del mensaje del protocolo de comunicación, se obtiene la dirección del archivo EXCEL, seguidamente se le extrae los contenidos, ya sea texto, imagen o enlace.	
Observaciones:	

Cuadro D.2: Historia de Usuario 5.

		Historia de Usuario
Número: PRE-6	**Nombre Historia Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo comprimido.	
Modificación de historia de Usuario Número: Ninguna		
Usuario: Alexis Rodríguez Reyes		**Iteración Asignada:** Sprint 1
Prioridad en Negocio: Muy Alta		**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Alto		**Puntos Reales:** 1
Descripción: Dada la entrada del mensaje del protocolo de comunicación, se obtiene la dirección del archivo comprimido, seguidamente se le extrae los contenidos, ya sea texto, imagen o enlace.		
Observaciones:		

Cuadro D.3: Historia de Usuario 6.

Historia de Usuario	
Número: PRE-10	**Nombre Historia Usuario:** Realizar trazas del módulo en un fichero de texto.
Modificación de historia de Usuario Número: Ninguna	
Usuario: Alexis Rodríguez Reyes	**Iteración Asignada:** Sprint 3
Prioridad en Negocio: Media	**Puntos Estimados:** 1 Semana
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Existirá un fichero en el cual se guardarán las trazas del módulo Pre-procesador, para saber el historial del funcionamiento y los errores que han ocurrido en la ejecución del servicio(demonio) a lo largo del tiempo, según especifique el administrador del sistema que utilice dicho servicio.	
Observaciones:	

Cuadro D.4: Historia de Usuario 10.

Anexo E

Pruebas funcionales

	Prueba funcional
Código de prueba funcional: PRE-4-1	**Nombre de Historia de Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo PPT(Presentación de PowerPoint).
Nombre de la persona que realiza la prueba funcional: Alexis Rodríguez Reyes	
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la extracción de todos los contenidos de un archivo PPT(Presentación de PowerPoint).	
Condiciones de ejecución: ■ El módulo *Pre-processor* debe conocer los parámetros del mensaje del protocolo de comunicación. ■ El documento a realizar la prueba debe ser del tipo de los formatos permitidos.	
Continúa en la próxima página	

Entradas / **Pasos de ejecución:**

- Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros:
  - -t: Valor del timestamp de cuando se empezó a procesar el documento.
  - -u: Dirección del documento PPT(Presentación de PowerPoint) a procesar.
  - -l: Identificador (uid) del lote.
  - -i: Identificador del documento(sha1).
  - -c: Comunidad(opcional).

.

Resultado esperado: Se guarda en una carpeta los contenidos (textos, enlaces, imágenes) del archivo PPT(Presentación de PowerPoint).

Evaluación: Prueba satisfactoria

	Prueba funcional
Código de prueba funcional: PRE-5-1	**Nombre de Historia de Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo EXCEL.
Nombre de la persona que realiza la prueba funcional: Alexis Rodríguez Reyes	
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la extracción de todos los contenidos de un archivo EXCEL.	
Condiciones de ejecución: - El módulo *Pre-processor* debe conocer los parámetros del mensaje del protocolo de comunicación. - El documento a realizar la prueba debe ser del tipo de los formatos permitidos.	

Continúa en la próxima página

Entradas / **Pasos de ejecución:**

- Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros:
  - -t: Valor del timestamp de cuando se empezó a procesar el documento.
  - -u: Dirección del documento EXCEL a procesar.
  - -l: Identificador (uid) del lote.
  - -i: Identificador del documento(sha1).
  - -c: Comunidad(opcional).

.

Resultado esperado: Se guarda en una carpeta los contenidos (textos, enlaces, imágenes) del archivo EXCEL.

Evaluación: Prueba satisfactoria

	Prueba funcional
Código de prueba funcional: PRE-6-1	**Nombre de Historia de Usuario:** Realizar la extracción de texto, imágenes, y enlaces de un archivo comprimido.
Nombre de la persona que realiza la prueba funcional: Alexis Rodríguez Reyes	
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se realice la extracción de todos los contenidos de un archivo comprimido.	
Condiciones de ejecución: - El módulo *Pre-processor* debe conocer los parámetros del mensaje del protocolo de comunicación. - El documento a realizar la prueba debe ser del tipo de los formatos permitidos.	

Continúa en la próxima página

Entradas / **Pasos de ejecución:**

- Se recibe por consola un prototipo de mensaje del protocolo de comunicación con los siguientes parámetros:
  - -t: Valor del timestamp de cuando se empezó a procesar el documento.
  - -u: Dirección del archivo comprimido a procesar.
  - -l: Identificador (uid) del lote.
  - -i: Identificador del documento(sha1).
  - -c: Comunidad(opcional).

.

Resultado esperado: Se guarda en una carpeta los contenidos (textos, enlaces, imágenes) del archivo comprimido.

Evaluación: Prueba satisfactoria

	Prueba funcional
Código de Prueba funcional: PRE-10-1	**Nombre de Historia de Usuario:** Realizar trazas del módulo en un fichero de texto.
Nombre de la persona que realiza la prueba funcional: Alexis Rodríguez Reyes	
Descripción de la Prueba: Se ejecuta la prueba y se verifica que se guarden las trazas del módulo *Pre-processor* en un documento.	
Condiciones de ejecución: - El módulo *Pre-processor* debe conocer la ruta del fichero donde se guardarán las trazas. - El archivo texto para guardar las trazas debe tener los permisos de escritura y lectura.	

Continúa en la próxima página

Entradas / **Pasos de ejecución:** ▪ Se ejecuta el proceso mediante un ejecutable y se guarda en un fichero de texto la información(errores, excepciones, rutas no validas).
Resultado esperado: Se guarda en el fichero de texto toda la información necesaria de la ejecución del servicio.
Evaluación: Prueba satisfactoria