UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 1

Entorno colaborativo para la administración de documentación (SUNSET)

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autor: José Jairán Breijo Rodríguez

Tutores:

Ing. Adrián Hernández Barrios

Ing. Mayleidis López Fernández

Ing. Royli Hernández Delgado

La Habana, 15 de junio de 2016

Dedicatoria

A mi familia, que es lo más grande que tengo en esta vida.

A ñaña, a pipo y a Javier, que ya no están.

A mi hermano que está lejos.

A mi vieja.

Agradecimientos

A mis padres y a mi hermanita, por su apoyo constante, por tanto sacrificio y amor a lo largo de estos 23 años y por ser la razón de mi orgullo y felicidad.

A tita y a tío, por ser mis otros padres.

Al Baky, a Liván, a Ernesto Soto y al Wisho, por ser los hermanos que no tuve.

A Adrián Hernández (el sensei), por su interminable paciencia, amistad y sabiduría.

A Osay, Royli, Mayi y Machado por su constante preocupación durante este viaje.

A mi compadre Yasmani, al Levi, al Papo, a Campillo y Asney por su amistad incondicional.
A mi hermanazo Manoli, por enseñarme que siempre se puede hacer el bien.
A los cracks de Pinar del Río: Cesar, el Isra y Peter.

A todos aquellos que se convirtieron en mis amigos durante estos cinco años, pues sería muy injusto no mencionarlos.

A todos aquellos que me ayudaron a forjarme como profesional.

A toda la gente que quiero y me quiere.

Declaración jurada de autoría

Declaro que soy el autor de la presente investigación titulada *Entorno colaborativo para la administración de documentación (SUNSET)*, para optar por el título de Ingeniero en Ciencias Informáticas.

El presente trabajo fue desarrollado en el período 2015-2016.

Finalmente declaro que todo lo anteriormente expuesto se ajusta a la verdad y asumo la responsabilidad moral y jurídica que se derive de este juramento profesional. Autorizo como único autor, a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente declaración jurada de autoría en La Habana a los 15 días del mes de junio del año 2016.

_____

Firma del Autor

José Jairán Breijo Rodríguez

_____	_____	_____
Firma de Tutor	Firma del Tutor	Firma del Tutor
Ing. Adrián Hernández Barrios	Ing. Mayleidis López Fernández	Ing. Royli Hernández Delgado

Resumen

La gestión y acceso a la documentación por parte de una comunidad de usuarios representa una necesidad para muchas instituciones. La búsqueda exhaustiva sobre documentos constituye una tarea compleja para cualquier sistema de gestión bibliográfica que se disponga a automatizar dicho proceso. La misma resulta imprescindible cuando el volumen de información almacenada se incrementa considerablemente. La propuesta de solución que se describe en la presente investigación consiste en desarrollar un entorno colaborativo de administración de documentos digitales, que disminuye la redundancia y aumenta la disponibilidad de la bibliografía que poseen los usuarios de la Universidad de las Ciencias Informáticas (UCI). El entorno permite la búsqueda de documentos, la gestión en una biblioteca personal, la posibilidad de compartirlos con el resto de la comunidad, así como la retroalimentación por medio de comentarios. La indexación de documentos incorpora un proceso de filtrado que emplea una expresión regular para la tokenización que elimina los caracteres que no son relevantes en el contenido. Posteriormente se eliminan las palabras más comunes que no aportan significado para la búsqueda y las resultantes son traducidas al origen de su significado, mediante la aplicación del stemmer de Porter. El empleo de estas técnicas de procesamiento de lenguaje natural produjo una reducción del 45% del tamaño del espacio de búsqueda y trajo consigo la disminución de los tiempos de respuesta del servidor de Elasticsearch en un 18%, capaz de ofrecer resultados acertados a las consultas de los usuarios, en un tiempo promedio de 28 milisegundos.

Palabras clave: búsqueda, documentos digitales, gestión bibliográfica, procesamiento de lenguaje natural.

Índice general

Índice de tablas

Índice de figuras

Introducción

La comunicación es una herramienta insustituible en los procesos de coordinación, integración y participación; es imprescindible además en la toma de decisiones, por la riqueza y variedad de información que proporciona. La esencia de cada pueblo está en el legado que conserve. Para ello la comunicación ha sido un eslabón fundamental y desde tiempos inmemoriales, ante nuevos retos y condicionantes, al hombre le ha surgido la necesidad de tener una guía a la hora de realizar cualquier actividad. El éxito de la misma ha dependido en gran medida de la correcta toma de decisiones, utilizando como punto de apoyo la experiencia acumulada con el tiempo.

En la medida en que las personas fueron desarrollando su capacidad de relacionarse y cooperar entre sí, utilizaron nuevos procedimientos y mecanismos al servicio de la comunicación y al traspaso de dicha información. La necesidad de reconceptualizar los mismos para dar soluciones nuevas a las exigencias de la cotidianidad, unido al imperativo de conservar lo heredado de los antepasados ante los retos de un mundo cambiante, hicieron posible que surgieran lo que hoy llamamos TIC (Tecnologías de la Información y la Comunicación). Estas herramientas permiten agilizar y optimizar muchas de las tareas de la vida diaria con el uso intensivo de la tecnología.

Todo esto condujo a un cambio pues ahora el conocimiento no se preserva solamente con el paso de las experiencias vividas de generación a generación, en forma de historias o hazañas, sino que surgen diversas formas de almacenar, gestionar y salvaguardar dicho conocimiento. Una de las formas más generalizadas de realizar esta gestión son las llamadas bibliotecas digitales. Ellas tienen la tarea de poner al alcance de una comunidad definida de usuarios, una amplia gama de fuentes documentales, con la garantía de que se puede tener acceso a este recinto informático todos los días del año, a cualquier hora.

Lo anterior conlleva a que un inmenso cúmulo de personas utilicen estas fuentes en su vida diaria, como apoyo para realizar tareas, ya sean de investigación, recreación o producción, como es el caso del desarrollo de *software* [1]. Considerado un ejercicio que requiere de un alto nivel de conocimiento y disponibilidad de la información, transita por diferentes fases: planificación, implementación, pruebas, documentación, despliegue y mantenimiento. En cada una de ellas intervienen disímiles tecnologías que traen aparejado el estudio intensivo de sus peculiaridades; con el objetivo de obtener un producto final de alta calidad, por lo

que se precisa del acceso constante a la documentación [2].

En Cuba existe una incipiente infraestructura tecnológica, hay acceso limitado a la red y a ello se suman las políticas implementadas por los Estados Unidos para restringir la conectividad, a pesar de los esfuerzos que se han hecho para mejorarla [3]. Todo lo anterior ha impedido utilizar por completo las opciones existentes en Internet, para aumentar el soporte al desarrollo de *software* mundial, así como las investigaciones. Opciones como: *Google Books*[1], *Scribd*[2], *Issuu*[3] y otras, ofrecen documentación que, de algún modo, ya sea por la necesidad de pago o las limitadas opciones de conectividad y cuota de Internet, no son posible utilizar en toda su extensión.

En la Universidad de las Ciencias Informáticas (UCI) existe una primera alternativa, que posibilita encontrar localmente documentación de apoyo a la producción y demás usos. *Sunshine*[4] es una aplicación que permite a los usuarios subir documentos y realizar búsquedas sobre el contenido de los mismos, pero, desarrollada sobre una tecnología que actualmente carece de soporte, cuenta ya con varios años de explotación y no está siendo mantenida. Los problemas asociados a la inestabilidad de los servicios que brinda el sitio, así como la falta de opciones y usabilidad del mismo, traen como consecuencia la necesidad de ofrecer un servicio de calidad, que permita no solo tener acceso a la documentación mediante su búsqueda, sino la posibilidad de gestionarla de manera adecuada y poder socializarla.

Mediante la utilización de este servicio se podrán consultar y gestionar adecuadamente los ejemplares existentes en la plataforma, que aumentarán a medida que cada usuario ponga a disposición de los demás aquellos de su pertenencia. De esta forma se elimina la necesidad de que cada usuario tenga una copia local de su documentación. Esto provocará un incremento significativo en la disponibilidad del cúmulo de información contenida en la UCI, ya sea por parte de la universidad o sus usuarios y una disminución de la redundancia o duplicidad de dichos documentos. Una gestión eficiente de la documentación y el aprovechamiento del papel catalizador de la comunicación enfocada a los mecanismos para ponerla al alcance de todos los usuarios, debe permitir un mejor desempeño en las tareas de la universidad.

[1] Servicio de Google que busca texto completo sobre libros. Disponible en https://books.google.es/
[2] Sitio web para leer y compartir documentos. Disponible en https://es.scribd.com/
[3] Servicio en línea que permite la visualización de material digitalizado electrónicamente. Disponible en https://issuu.com/
[4] Sitio web para el almacenamiento y búsqueda de documentos. Disponible en http://sunshine.prod.uci.cu

De acuerdo con la **problemática** planteada anteriormente emerge el siguiente **problema de investigación**: ¿Cómo contribuir a la disponibilidad y utilización de la documentación que poseen los usuarios de la Universidad de las Ciencias Informáticas?

La presente investigación enmarca su **objeto de estudio** en el procesamiento y la gestión de documentación digital.

Se establece como **objetivo general** desarrollar un entorno colaborativo de administración de documentación, con modernas tecnologías, que disminuya la redundancia y aumente la disponibilidad de la bibliografía que poseen los usuarios de la Universidad de las Ciencias Informáticas.

Para alcanzar este objetivo general se definen los siguientes **objetivos específicos**:

1. Analizar los aspectos teóricos-prácticos referentes a sistemas homólogos, así como técnicas de minería de texto.
2. Identificar las metodologías, herramientas y tecnologías para el desarrollo de la propuesta de solución.
3. Establecer el diseño de la propuesta de solución.
4. Implementar las funcionalidades del entorno colaborativo de administración de documentación.
5. Examinar la propuesta de solución a través de las pruebas de software.

La presente investigación se rige por la siguiente **idea a defender**: si se desarrolla un entorno colaborativo de administración de documentación entonces se contribuye a la disminución de la redundancia y al aumento de la disponibilidad de la documentación existente en la Universidad de las Ciencias Informáticas.

Métodos teóricos formales

- Analítico: Se utilizó en el análisis de los procedimientos referentes a la gestión de documentación, para identificar estándares, herramientas y tecnologías idóneas para el desarrollo de la aplicación en cuestión.

- Modelación: Se utilizó en aras de lograr un mayor entendimiento de las características intrínsecas del sistema, a través de la realización de diagramas.

- Histórico - Lógico: Se empleó en el estudio de los antecedentes, la evolución y el desarrollo que han tenido los sistemas de gestión de documentación.

Métodos empíricos

- Experimental: Se utilizó para registrar los tiempos de respuesta del entorno colaborativo con y sin la utilización de una estrategia de pre-procesado del contenido de los documentos.

Estructura del documento

El presente documento consta de 3 capítulos:

Capítulo 1: Fundamentación teórica. Es este capítulo se describen los principales conceptos asociados al problema de investigación planteado. Se hace un estudio sobre los sistemas homólogos al propuesto, así como de técnicas de minería de texto. Finalmente, se seleccionan las tecnologías, herramientas y metodología adecuadas para el desarrollo del entorno colaborativo.

Capítulo 2: Propuesta de solución. Se hace énfasis en la evolución de dicha solución a través de las fases iniciales: Planificación y Diseño, de la metodología de desarrollo de *software Extreme Programming (XP)*, presentando los diferentes artefactos generados por la misma. Se selecciona la estrategia a utilizar para la optimización del espacio de búsqueda.

Capítulo 3: Implementación y prueba. En este capítulo se les da cumplimiento a los planes trazados mediante la implementación de la propuesta de solución y finalmente se examina la calidad del entorno colaborativo a través de pruebas de *software*.

Capítulo 1

Fundamentación teórica

En el presente capítulo se establecen varios conceptos que son fundamentales para la mejor comprensión de la investigación. Se realiza un análisis acerca de aplicaciones web que presentan características afines al entorno colaborativo de administración de documentación. Se hace énfasis en diferentes técnicas de procesamiento de lenguaje natural, con el fin de emplear esta rama de la minería de texto en la optimización del espacio de búsqueda. Finalmente se identifican las tecnologías, metodologías y herramientas a utilizar para el desarrollo del entorno colaborativo.

1.1. Conceptos asociados al dominio del problema

Documentación: Se refiere al documento o colección de documentos digitales almacenados en formato PDF[5]. Este es un formato de archivo utilizado para presentar e intercambiar documentos de forma fiable, independiente del *software*, el *hardware* o el sistema operativo. Creado por *Adobe*[6], es ahora un estándar abierto y oficial reconocido por la Organización Internacional para la Estandarización (*ISO 32000*). Estos archivos pueden contener vínculos, botones, campos de formulario, audio, vídeo y se pueden firmar digitalmente [4].

1.2. Estudio de sistemas homólogos

El análisis de las particularidades de los sistemas afines al propuesto, permite conocer características que, por su relevancia, pueden ser fundamentales para la propuesta a desarrollar, o servir como objeto de estudio

[5] https://acrobat.adobe.com/la/es/why-adobe/about-adobe-pdf.html
[6] Empresa de software estadounidense destaca por sus programas de edición de páginas web, vídeo e imagen digital. Sitio web: http://www.adobe.com/es

para realizar una selección acertada de estándares, herramientas y tecnologías a utilizar para su construcción.

1.2.1. Principales alternativas identificadas a nivel internacional

Google Books como primera alternativa, es un proyecto audaz que comenzó en 2004 con el propósito de utilizar el poder de las búsquedas de *Google*[7] en los libros impresos. Este incluía un programa para escanear millones de libros de las principales bibliotecas del mundo y aplicar indexación de texto completo, haciendo posible realizar dichas búsquedas sobre el contenido de estos documentos [5]. Actualmente más de 25 millones de libros han sido digitalizados, incluyendo textos en 400 lenguajes diferentes provenientes de más de 100 países [6].

El análisis y reconocimiento de texto para el proceso de búsqueda, es posible debido al empleo de modelos como *n-gramas*, un tipo de lenguaje de modelado probabilístico que permite la predicción del siguiente elemento en una secuencia de orden (n-1). Es ampliamente utilizado en probabilidad y lingüística computacional, como es el caso del procesamiento de lenguaje natural. Una muestra de su eficiencia fue su empleo en la clasificación de diferentes idiomas, en los que alcanzó una tasa de acierto del 99.8%, con una serie de artículos de *Usenet*[8] [7].

Google Books Ngram Corpus ha permitido el análisis cuantitativo de las tendencias lingüísticas y culturales que se reflejan en millones de libros escritos durante los últimos cinco siglos. Los *corpus*[9] constan de palabras y frases (es decir, n-gramas) y su frecuencia de uso en el tiempo. Los datos están disponibles para su descarga y también se pueden observar a través del visor interactivo de *Google Books*[10] [8].

Scribd comenzó como un sitio para compartir y poner en línea documentos. Ya en 2007, después de un largo tiempo trabajando con autores y editores, se evidencia la necesidad de más lectores, así como más ingresos. Esto da al traste con un modelo de suscripción a libros, que permitía ampliar la distribución y

[7] Compañía especializada en productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras tecnologías. Disponible en https://google.com
[8] Colección de grupos de noticias, donde los usuarios pueden publicar mensajes y estos mensajes publicados se distribuyen a través de los servidores de Usenet. Sitio web: http://www.usenet.org/
[9] Conjunto cerrado de textos o de datos destinado a la investigación científica
[10] Visor interactivo de N-gramas de Google Books, disponible en http://books.google.com/ngrams

monetización, siendo esta la opción más acertada para el producto y su público objetivo y el modelo adecuado para el futuro [9].

La aplicación fue desarrollada en *Ruby on Rails, Python, MySQL* y utiliza *Squid* como sistema de caché avanzado. Cuenta con soporte para varios formatos que pueden ser convertidos a *iPaper*[11], lo que permite a los usuarios incrustar documentos en una página web. Hoy *Scribd* cuenta con 80 millones de lectores mensuales en más de 190 países volviéndose el sitio web más popular de este tipo, con un enfoque parecido a *YouTube*[12] pero para texto [10].

ISSUU es una plataforma líder de publicación digital, que entrega experiencias excepcionales de lectura de revistas, catálogos y periódicos. Es la plataforma de publicación digital con el crecimiento más rápido del mundo, con alrededor de 60 millones de lectores mensuales y más de 10 millones de publicaciones disponibles. Constituye un sitio de destino muy popular, donde los usuarios están involucrados con las mejores publicaciones de la web y donde los editores construyen su audiencia. Para el análisis de las publicaciones redactadas en más de 180 lenguajes y poder recomendar lecturas similares a sus clientes, se utiliza *LDA*[13] con filtrado colaborativo para filtrar la información, revelar patrones y hacer predicciones automáticas acerca del interés de los usuarios, apoyado en sus preferencias [11].

1.2.2. Principales alternativas identificadas a nivel nacional

LUPA[14] es un buscador de contenidos en la REDUNIV[15] desarrollado en la Universidad de Pinar del Río, con el propósito de localizar materiales dispersos en sitios web y repositorios públicos de la red del Ministerio de la Educación Superior y en sus enlaces transversales de alta velocidad (MINED[16], INFOMED[17], Cultura, Joven Club, UCI). El empleo de esta herramienta tiene un impacto directo en el aprovechamiento de los contenidos dentro del país que disminuye el tráfico de los usuarios de la REDUNIV hacia Internet en busca de información, en los populares buscadores internacionales. Entre sus funcionalidades principales se

[11] Formato de documento rico similar al PDF construido con Adobe Flash
[12] Sitio web en el cual los usuarios pueden subir y compartir vídeos. Disponible en https://www.youtube.com/
[13] Análisis discriminante linear, técnica empleada comúnmente para la clasificación de datos
[14] Sitio web: http://lupa.upr.edu.cu/
[15] Red Nacional Universitaria. Sitio web: http://www.reduniv.mes.edu.cu/
[16] Ministerio de la Educación
[17] Portal de la red de salud de cuba, disponible en http://www.sld.cu/

encuentran la sugerencia de documentos similares, el filtrado de resultados por tipos de archivo y la posibilidad de ofrecer búsquedas relacionadas [12].

DIMA[18] (Directorio Integrado de Materiales de Acceso Abierto) es un proyecto investigativo en el área de la recuperación de información académica y científica, que surge con el objetivo de facilitar el acceso a parte importante de la producción académica y científica nacional. Permite concentrar parte de la producción científica nacional en un repositorio único que facilite a estudiantes, académicos e investigadores, la búsqueda y valoración de las investigaciones realizadas en el país en un determinado dominio. Está construido sobre la arquitectura *LAMP*, acrónimo usado para describir un sistema de infraestructura de Internet, que utiliza sistema operativo ***L****inux*, servidor web ***A****pache*, gestor de base de datos ***M****ySQL* y el lenguaje de programación ***P****HP* [13].

Dicho sistema cuenta con un indexador *OAI-PMH* (Iniciativa de Archivos Abiertos - Protocolo para la Recolección de Metadatos), iniciativa involucrada en el desarrollo de un marco tecnológico para mejorar el acceso a los archivos electrónicos, permitiendo así aumentar la disponibilidad de la comunicación científica. OAI se encuentra estrechamente relacionado con el movimiento de acceso abierto a publicaciones [14].

El análisis de los sistemas internacionales expuestos anteriormente, arrojó que la necesidad de pago, las limitantes en cuanto a conectividad desde el país y el consumo excesivo de la cuota de Internet en el caso de la universidad, son algunas de las razones que no permiten la correcta utilización de las alternativas de alcance internacional.

A pesar de que las alternativas identificadas a nivel nacional suponen una solución al problema de investigación planteado, mediante su utilización solamente se tiene acceso a la documentación que dichos servicios ofrecen. No existe la posibilidad de que los usuarios pongan a disposición de la comunidad la documentación que almacenan localmente, que realicen la gestión adecuada de aquellos materiales de su preferencia en una colección personal y que obtengan retroalimentación sobre los mismos, mediante su socialización y comentarios.

[18] Sitio web: http://dima.mes.edu.cu/

El estudio de las alternativas mencionadas anteriormente evidenció una tendencia a realizar la indexación del texto de los documentos con el fin de realizar búsquedas sobre los mismos. El empleo de esta práctica significa la utilización de una capacidad de almacenamiento considerable. Por consiguiente, aflora la necesidad de utilizar modernas técnicas de minería de texto con el objetivo de llevar a cabo una reducción del espacio de búsqueda a utilizar.

1.3. Minería de texto

El proceso de búsqueda representa la característica fundamental de la investigación, pues se deben garantizar los mejores resultados posibles. Para lograrlo, se debe hacer que el espacio sobre el que se realizan dichas búsquedas, dígase el contenido y metadatos de los documentos, presente características orientadas a favor de la obtención de un resultado más acertado y en el menor tiempo posible. Desafortunadamente, los métodos habituales de programación basados en la lógica tienen grandes dificultades para captar las relaciones difusas y a menudo ambiguas que presenta el contenido de los documentos; por ende, se debe hacer uso de la minería de texto.

La **minería de texto** es una variación de un campo llamado minería de datos[19]. Es también conocida como *análisis inteligente de texto* o *descubrimiento del conocimiento en el texto* (*KDT*, por sus siglas en inglés). Generalmente se refiere al proceso de extraer información interesante y no trivial y el conocimiento del texto no estructurado. Este es un campo interdisciplinario y joven que se basa en la *recuperación de información*, *minería de datos*, *aprendizaje automático*[20], *estadísticas* y *lingüística computacional*[21][15].

La investigación en el ámbito de la minería de texto aborda los problemas de representación de texto, clasificación, agrupación, la extracción de información o la búsqueda y el modelado de patrones ocultos. En este contexto, la selección de características y procedimientos específicos desempeña un papel importante, por lo que una adaptación de los conocidos algoritmos de minería de datos suele ser necesaria. Para lograrlo, con frecuencia se utiliza como base la experiencia y los resultados de la investigación sobre la recuperación

[19] Proceso de identificación de patrones en grandes conjuntos de datos. El objetivo es obtener conocimiento útil y previamente desconocido.
[20] Área de la inteligencia artificial se ocupa del desarrollo de técnicas que permitan a los ordenadores "aprender" mediante el análisis de conjuntos de datos.
[21] Campo interdisciplinario que trata con el modelado estadístico o basado en reglas del lenguaje natural desde el punto de vista computacional.

de información, procesamiento del lenguaje natural y la extracción de información [16]. En todas estas áreas también se aplican métodos de minería de datos y estadísticas para manejar tareas específicas como:

Recuperación de información: hallazgo de los documentos que contienen respuestas a preguntas y no el hallazgo de respuestas en sí [17]. Con el fin de lograr este objetivo se utilizan medidas y métodos estadísticos para el tratamiento automatizado de los datos de texto y la comparación con la pregunta en cuestión. Si bien la recuperación de información es un área de investigación relativamente antigua, donde los primeros intentos para la indexación automática se hicieron en 1975, la misma ganó una mayor atención con el auge de la *World Wide Web* y la necesidad de sofisticados motores de búsqueda.

Procesamiento de lenguaje natural: uno de los problemas más antiguos y más difíciles en el campo de la inteligencia artificial. Su objetivo es lograr que las computadoras puedan entender el lenguaje natural como lo hacen los humanos. Aunque este objetivo aún se encuentra un poco lejos, el procesamiento de lenguaje natural puede realizar algunos tipos de análisis con un alto grado de éxito. Analizadores de poca profundidad sólo identifican los principales elementos gramaticales en una oración, como frases nominales y verbales, mientras que los analizadores sintácticos profundos generan una representación completa de la estructura gramatical de una oración. Su papel en la minería de texto es proporcionar a los sistemas en la fase de extracción de información, de datos lingüísticos que estos necesitan para llevar a cabo su tarea [18].

Los fundamentos del procesamiento de lenguaje natural se encuentran en una serie de disciplinas, como la computación y ciencias de la información, la lingüística, matemáticas, ingeniería eléctrica, electrónica, inteligencia artificial, robótica y psicología. Sus aplicaciones se componen de un número de campos de estudios, tales como la traducción automática, el procesamiento de texto en lenguaje natural y su síntesis, recuperación de información multilingüe, reconocimiento de voz, sistemas de inteligencia artificial y de expertos [19].

La minería de texto hace énfasis en la clasificación de documentos en diferentes categorías, de acuerdo a su estructura y contenido. Para la presente investigación no es necesario realizar la clasificación de dichos textos, pues el usuario es quien describe sobre qué trata el documento mediante el uso de etiquetas, además de la información contenida como metadatos del fichero. La necesidad de emplear la minería de texto, recae en la utilización de la fase previa al proceso de clasificación, o sea, el **pre-procesamiento**.

Se ha demostrado que el tiempo dedicado al pre-procesamiento puede tardar desde el 50% hasta el 80% de todo el proceso de clasificación, lo que claramente avala su importancia [20]. Un pre-procesamiento efectivo de los documentos debe representar un aumento en la eficiencia de la aplicación, en términos de reducción de espacio (para indexar el contenido de los documentos) y de tiempo (para obtener los resultados según un criterio de búsqueda). En aras de lograr este aumento se utilizan técnicas de filtrado del texto como la **eliminación de stop-*words***, ***lemmatization*** y métodos de ***stemming***.

1.3.1. Eliminación de *Stop-Words*

Con el fin de obtener todas las palabras que se utilizan en un texto dado, se requiere de un proceso de tokenización; es decir, un documento de texto se divide en una corriente de palabras mediante la eliminación de todos los signos de puntuación y la sustitución de las tabulaciones y otros caracteres, por espacios individuales.

Las palabras vacías (***stop-words***) forman parte del lenguaje natural. El motivo de que deban ser removidas de un texto es que hacen que el mismo parezca más pesado y menos importante para su análisis. Las palabras más comunes en los documentos de texto son los artículos, preposiciones, conjunciones, pronombres, etc., que no añaden ningún significado y son tratadas como palabras vacías. Por ejemplo: el, en, un, una, con [21].

Existen diferentes métodos para llevar a cabo la eliminación de stop-words, a continuación, se ofrece un análisis de los más utilizados.

- **Método clásico**: El método clásico se basa en la eliminación de palabras vacías a partir de su comparación con listas pre-elaboradas.
- **Métodos basados en las leyes de *Zipf*. (Métodos Z)**: En adición a la lista clásica de stop-words, se eliminan también las palabras más frecuentes, las que ocurren una sola vez y las que tienen menor frecuencia inversa (IDF) [22].
    - IDF (t) = log(N/nt) donde N es el tamaño de la colección de documentos y nt es el número de documentos que contienen el término t.

- **Método de información mutua**: es un método supervisado que funciona mediante el cálculo de la información mutua entre un término dado y una clase de documento (por ejemplo: positivo, negativo), que proporciona una sugerencia de la cantidad de información que el término puede decir acerca de una clase determinada. Baja información mutua sugiere que el término tiene bajo poder discriminante y por consiguiente no es necesario [22].

1.3.2. *Lemmatization*

Los métodos de *lemmatization* tratan de establecer una relación entre las formas verbales con su infinitivo y de los sustantivos con su forma singular. Sin embargo, en orden de lograr este objetivo, se debe asignar a cada palabra su función como parte de discurso en el documento, o sea, comprender el contexto en que aparecen. Este proceso usualmente consume mucho tiempo y es propenso a errores, en la práctica se aplican frecuentemente métodos de ***stemming*** considerados tan efectivos como *lemmatization*, pero con un costo menor en rendimiento [16].

1.3.3. *Stemming*

Estos métodos tratan de construir las formas básicas de las palabras, es decir, quitar el plural de los sustantivos, el "ando" de los gerundios, u otros añadidos. Un ***stem*** es un grupo natural de palabras con el mismo o muy similar significado. Después del proceso de *stemming*, cada palabra está representada por su *stem*. Por ejemplo: “pescar”, “pescado”, “peces”, son derivados de “pez”, por ende, “pez” es su *stem*. Su propósito es reducir el número de palabras para obtener *stems* que coincidan con precisión y así ahorrar tiempo y espacio en memoria [16].

En este proceso, la traducción de una palabra a su *stem*, se hace asumiendo que están semánticamente relacionadas. Hay dos puntos a considerar cuando se usa un *stemmer*[22]:

1. Las palabras que no tienen el mismo significado deberán mantenerse separadas.

[22] Algoritmo o método encargado de llevar a cabo el proceso de stemming

2. Se asume que las formas morfológicas de una palabra tienen el mismo significado base y por tanto deberán ser relacionadas con un mismo *stem*.

Estas dos reglas son suficientes para llevar a cabo este proceso en la minería de texto o en aplicaciones de procesamiento de texto. Para idiomas con una morfología relativamente simple, el poder del stemming es menor que para aquellos que tienen una morfología más compleja. La mayor parte de los experimentos que se han hecho hasta ahora están en inglés y otros idiomas de Europa Occidental.

Por lo general, los algoritmos de stemming se pueden clasificar en tres grupos: métodos de truncado, métodos estadísticos y los métodos mixtos [22]. Cada uno de estos grupos tiene una forma típica de buscar el *stem*, de las variantes de una palabra. Para el propósito de esta investigación se hará referencia a un algoritmo de cada clasificación:

***Porter Stemmer* (Truncado):** El algoritmo de *Porter* es uno de los más populares algoritmos de *stemming*. Propuesto en 1980, se basa en la idea de que los sufijos en el idioma inglés (aproximadamente 1200) se componen de la agrupación de sufijos más pequeños y sencillos. Tiene cinco pasos y en cada uno se aplican reglas hasta que se cumpla alguna de ellas. Si se acepta una regla, el sufijo se elimina en consecuencia y se realiza el paso siguiente. El resultado será el *stem* obtenido al final de la quinta etapa. Este algoritmo tiene cerca de 60 reglas y es muy fácil de entender. *Porter* diseñó un marco de trabajo detallado de *stemming* que se conoce como "*bola de nieve*", cuyo propósito principal es permitir a los programadores desarrollar sus propios *stemmers* para otros conjuntos de caracteres o idiomas [22].

***N-Gram Stemmer* (Estadístico**): Es un *stemmer* independiente del lenguaje. Utiliza un enfoque de similitud de cadenas para convertir una palabra a su raíz. Un *n-grama* es una cadena de n caracteres, por lo general adyacentes, extraídos de una sección de texto continuo. La idea principal detrás de este enfoque evidencia que palabras similares tendrán una alta cantidad de *n-gramas* en común. Este *stemmer* tiene la ventaja de ser independiente del lenguaje y por lo tanto muy útil en muchas aplicaciones. La desventaja es que requiere gran memoria y almacenamiento, por lo tanto, no es un enfoque práctico [22].

***Krovetz Stemmer* (Mixto):** El *stemmer* de *Krovetz* fue presentado en 1993 por Robert Krovetz y es un stemmer de validación léxico lingüística. Puesto que se basa en la propiedad de inflexión de las palabras y

la sintaxis del lenguaje, presenta una naturaleza bastante complicada. Los sufijos se eliminan de forma efectiva y precisa en tres pasos:

1. Transformar el plural de las palabras a su forma singular.
2. Convertir el pasado de las palabras al presente.
3. Eliminar los sufijos restantes.

Este *stemmer* intenta aumentar la precisión y robustez mediante el tratamiento de los errores ortográficos y *stems* sin sentido. Si el tamaño del documento de entrada es grande, este método se debilita y no se ejecuta de manera muy eficaz. El principal defecto de métodos como estos basados en diccionarios, es la incapacidad para manejar palabras que no estén incluidas en dichos diccionarios [23].

A través de este análisis ha quedado en evidencia la importancia de la utilización y selección acertada de métodos de pre-procesado. Ello posibilitará la obtención de un espacio de búsqueda con características idóneas, lo cual debe producir un impacto positivo en el rendimiento del sistema, menores tiempos de respuesta y un aumento en la calidad de los resultados obtenidos.

1.4. Metodología de desarrollo de *software*

En la ingeniería de *software*, una metodología de desarrollo de *software* es la encargada de la separación de este proceso en distintas fases o etapas, que contienen actividades enfocadas a una mejor planificación y administración del mismo. Puede incluir la definición previa de una serie de artefactos que son creados y completados con el equipo que desarrolla o mantiene la aplicación [24].

Existen numerosas propuestas metodológicas que inciden en diferentes dimensiones del mismo. Hay propuestas más tradicionales que se centran especialmente en el control, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben generar, así como las herramientas a utilizar. Otras se enfocan en el factor humano o el producto de *software*. Esta filosofía de las metodologías ágiles da mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del *software* con iteraciones muy cortas [25].

Para el desarrollo de la solución propuesta se seleccionó ***Extreme Programming* (*XP*, por sus siglas en inglés)**, debido a que es una metodología ágil centrada en potenciar las relaciones interpersonales como clave del éxito. A través de su utilización se promueve el trabajo en equipo, predominando el aprendizaje de los desarrolladores y un buen clima de trabajo. Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios.

XP es especialmente adecuada para proyectos con requisitos muy cambiantes y donde existe un alto riesgo técnico. Su artefacto principal son las historias de usuario, tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, ya sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, pues en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarlas en unas semanas [26].

1.5. Herramientas y tecnologías

La selección acertada de herramientas y tecnologías para emplear en el proceso de desarrollo, apoyado en el dominio que se tenga de las mismas, es crucial para la obtención en tiempo de un producto de calidad. Con esa idea presente, se hizo uso de los recursos a continuación:

1.5.1. Modelado

El lenguaje de modelado ***UML*[23]** en su versión 2.5, permite comunicar la estructura de un sistema complejo, especificar el comportamiento deseado del sistema, comprender mejor lo que se está construyendo y a la vez descubrir oportunidades de simplificación y reutilización. Se basa en el hecho de que un modelo es la simplificación de la realidad [27].

[23] Lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad

Visual Paradigm en su versión 8.0, se selecciona como herramienta CASE[24] de modelado profesional, que utiliza UML para la completa representación de las etapas por las que transita un producto de *software*. Este permite la realización de una amplia gama de diagramas como: casos de uso, de actividades, de despliegue, entre otros, así como la generación de código fuente desde los mismos y la documentación asociada al proceso que esté siendo modelado.

1.5.2. Lenguajes de programación

Para el desarrollo de la aplicación se realizó un análisis teniendo presente su objetivo intrínseco: el trabajo con la documentación. Con ese fin se seleccionó *Python*, pues cuenta con una gran cantidad de librerías que permiten el trabajo con dichos documentos, así como herramientas para procesamiento de lenguaje natural, como *NLTK*[25](del inglés *Natural Languaje Toolkit*).

Python en su versión 3.4.3, es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Su elegante sintaxis y tipado dinámico junto con su naturaleza interpretada, hacen de este un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas [28].

Algunos casos de éxito en el uso de *Python* son *Google*, *Yahoo*, la *NASA*[26], Industrias *Light & Magic* y todas las distribuciones *Linux*, en las que cada vez representa un tanto por ciento mayor de los programas disponibles [29]. Es administrado por la *Python Software Foundation*, una organización sin fines de lucro orientada al avance de tecnologías de código abierto relacionadas con dicho lenguaje de programación. Posee una licencia de código abierto, denominada *Python Software Foundation License*[27].

Se seleccionó ***Django*** como framework de desarrollo web del lado del servidor, pues, construido sobre *Python*, fomenta el desarrollo ágil permitiendo la construcción de aplicaciones de alta calidad en un breve periodo de tiempo. Utiliza una modificación de la arquitectura Modelo-Vista-Controlador (*MVC*) llamada

[24] Ingeniería de Software Asistida por Computadora (del inglés *Computer Aided Software Engineering*). Aplicaciones destinadas a aumentar la productividad en el desarrollo de software reduciendo el costo del mismo en términos de tiempo y dinero.
[25] http://www.nltk.org/
[26] Agencia del gobierno estadounidense responsable del programa espacial civil, investigación aeronáutica y aeroespacial.
[27] https://www.python.org/psf/about/

Modelo-Plantilla-Vista (*MTV*, por sus siglas en inglés). Django impulsa el desarrollo de código limpio y promueve la utilización de buenas prácticas de desarrollo web como el principio *DRY*[28]. Entre algunos casos de éxito se encuentran: *Instagram*[29], *Pinterest*[30] y *The New York Times* [30].

Como framework de desarrollo web del lado del cliente se selecciona ***AngularJS*** en su versión 1.5.0. Este es un marco de trabajo que se utiliza principalmente para crear aplicaciones web modernas de una sola página, mediante el aumento del nivel de abstracción entre el desarrollador y las tareas comunes que conlleva dicho desarrollo [31]. Permite la construcción de aplicaciones web sencillas, así como de gran complejidad, pues toma cuidado de las características avanzadas a las que los programadores están acostumbrados en la actualidad, como son:

- Separación entre lógica de negocio, modelos de datos y vistas.
- Servicios *AJAX*[31].
- Inyección de dependencias.
- Historial del navegador.
- Pruebas

AngularJS promueve y usa patrones de diseño de *software*. En concreto, implementa lo que se llama *MVC*[32]. Básicamente este patrón nos marca la separación del código de los programas dependiendo de su responsabilidad. Eso permite repartir la lógica de la aplicación por capas, lo que resulta muy adecuado para aplicaciones de negocio y para las aplicaciones de una sola página (del inglés *Single Page Application*)[33]. Esta librería de código abierto es mantenida por *Google* bajo licencia *MIT*[34] [32].

[28] Principio orientado a reducir la repetición de información de todo tipo, especialmente útil en arquitecturas de múltiples niveles.
[29] Red social y aplicación para subir fotos y videos. Disponible en: https://www.instagram.com/
[30] Plataforma para compartir imágenes. Disponible en https://es.pinterest.com/
[31] Tecnología asíncrona. Los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.
[32] Patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones
[33] Sitios web donde los usuarios perciben una experiencia similar a la que se tiene con las aplicaciones de escritorio.
[34] http://opensource.org/licenses/mit-license.php

1.5.3. Estrategia de búsqueda

El proceso de búsqueda se reduce a un problema clásico en ciencias de la computación, llamado correspondencia de cadenas. Este propone métodos para encontrar el lugar donde una o varias cadenas son encontradas dentro de un texto, lo cual presenta múltiples aplicaciones como la detección de intrusiones en la red, su utilización en la bioinformática, detección de plagio, seguridad de la información, reconocimiento de patrones y minería de texto [33].

Entre los algoritmos más utilizados se encuentran:

- **Algoritmos de Fuerza Bruta**. No requieren de un pre-procesamiento del patrón a buscar (***p*** de ahora en adelante) o el texto (***t***) sobre el cual se busca, sino que ambos son comparados letra por letra. En caso de encontrar un desajuste[35] el patrón es corrido una posición a la derecha y la comparación es repetida hasta que se llegue al final del texto, obteniéndose una complejidad temporal de O[36](*n* * *m*), siendo ***n*** la longitud de ***p*** y ***m*** la longitud de ***t*** [34].

- ***Knut-Morris-Prat* (*KMP*)** observa que no es necesario correr ***p*** solo una letra hacia la derecha, sino que, mediante la utilización de la información ganada en comparaciones previas, evita comparar letras que ya han coincidido para determinar el corrimiento. Su tiempo de ejecución es de O (***n*** + ***m***) [35].

- ***Aho-Corasick*** se utiliza para búsqueda de múltiples patrones ***p*** en ***t***. Funciona como una comparación de diccionario[37] que localiza elementos de un conjunto finito de cadenas dentro del texto de entrada, a través de la confección de un autómata finito a partir de los patrones ***p***. Su atractivo se encuentra dado mayormente por el hecho de que varios patrones pueden ser buscados

[35] Dado una posición i y j, en p y t, respectivamente, p[i] es diferente de t[j]
[36] Notación utilizada para clasificar algoritmos por cómo responden, o sea, su tiempo de ejecución en dependencia de cambios es su entrada
[37] Colección no-ordenada de valores que son accedidos a través de una clave. En lugar de acceder a la información mediante el índice numérico, como es el caso de las listas, es posible acceder a los valores a través de sus claves, que pueden ser de diversos tipos.

simultáneamente con una complejidad lineal, pero mientras el tamaño del autómata aumenta, el algoritmo sufre una degradación en términos de tiempo y espacio [36].

El empleo de los algoritmos mencionados anteriormente, supone una solución eficiente a la hora de encontrar ocurrencias de patrones en cadenas de texto, pero el rendimiento se ve enormemente afectado cuando el volumen de los datos a analizar crece; se vuelve entonces intolerable su tiempo de ejecución.

Por esta razón, se debe tomar otro acercamiento a este tipo de problema, pues el entorno colaborativo debe ser capaz, dado un criterio de búsqueda provisto por el usuario, de encontrar los documentos que tengan relación con dicho criterio en el menor tiempo posible. El conjunto sobre el cual se hará este proceso será tan grande, como libros hayan sido agregados a la plataforma.

Para la obtención masiva de información, se utiliza un mecanismo llamado **índice invertido**. Para su explicación, se hace uso de tres documentos como ejemplo:

- D0 = "*a big apple*"
- D1 = "*a apple I love apple*"
- D2 = "*big pig eat apple*"

El índice invertido consta de una estructura de datos que mantiene la asociación entre el contenido (números o palabras) y su localización en una base de datos, o el documento o conjunto de documentos del cual forma parte. En un motor de búsqueda, el índice invertido es mayormente una asociación entre una palabra y el documento donde está presente.

Tabla 1: Descripción de la estructura de un índice invertido.

palabra	a	*big*	*apple*	I	*pig*	*love*
documento	D0, D1	D0, D2	D0, D1,D1, D2	D1	D2	D1

Para el criterio de búsqueda “a *apple*”, primero se busca la palabra “a” en la fila de las palabras y se obtiene su listado de documentos correspondiente R(“a”) = {D0, D1}. Luego, la búsqueda de “*apple*” resulta en R(“*apple*”) = {D0, D1, D2}. El resultado final sería la intersección de estos dos conjuntos:

$$R(\text{“a”}) \cap R(\text{“}apple\text{”}) = \{D0, D1\} \cap \{D0, D1, D1, D2\} = \{D0, D1\}$$

Normalmente se utiliza un mecanismo para dar una puntuación a los resultados. El más simple es en dependencia de la frecuencia de los términos, definido como la cantidad de veces que se encuentra una palabra en un documento. Para este ejemplo, la puntuación de D0 es 2 y D1 es 3. La complejidad temporal de este método depende principalmente del número de palabras en el criterio de búsqueda. Este método provee un sistema de búsqueda rápido. Solo se necesita solventar dos problemas para su utilización:

1. ¿Cómo segmentar el texto en “palabras”?
2. ¿Cómo asegurar que el conjunto de palabras o vocabulario no sea demasiado largo?

La primera condición se puede satisfacer haciendo uso de la segmentación en dependencia de espacios, o signos de puntuación. Si se está en presencia de un lenguaje que carece de ellos como el chino, se pueden utilizar *n-gramas* de tamaños predefinidos [37].

Para asegurar un tamaño idóneo del vocabulario o espacio de búsqueda, se dividen secuencias largas en cortas análogamente a la existencia de párrafos en un libro, aunque no existe un estándar establecido en cuanto a la longitud de dichas secuencias. Las ventajas asociadas a este proceso pueden ser explicadas fácilmente: para el criterio de búsqueda “*I love apple*”, “*I really love apple*” puede ser visto como una posible solución, pero “*I* + (1000 palabras) + *love* + (1000 palabras) *apple*” no figura como un resultado adecuado.

Una vez satisfechas estas condiciones se ordenan las palabras en el criterio de búsqueda por la *frecuencia de documento*, o sea, la cantidad de documentos que contiene a cada una de ellas. Luego se mezcla la lista de documentos arrojada por el método del índice invertido de acuerdo con esta lista ordenada, hasta que el número de candidatos resulte menor que un umbral [38].

Por ejemplo: para el siguiente índice invertido con un criterio de búsqueda {W1, W3, W4}; si se busca primero W1, R(“W1”) = {D1} y luego para R(“W3”) = {D0, D2, D4}. Su intersección resulta en {}, por tanto, se usaría solo D1 como resultado candidato.

Tabla 2: Índice invertido de muestra.

palabra	W1	W2	W3	W4
documento	D1	D0, D2,D3	D0, D2, D4	D0, D2

Si se hace un ordenamiento de acuerdo con la frecuencia de documento, el criterio quedaría como {W3, W4, W1}, entonces R(“W3”) = {D0, D2, D4}, R(“W4”) = {D0, D2} y su intersección sería {D0, D2}. A pesar de que no hay resultado para la intersección entre R(“W3”), R(“W4”) y R(“W1”), aún se puede utilizar el resultado anterior {D0, D2}, como resultado de la búsqueda, mucho mejor este que {D1} que solo concuerda con una palabra [39].

El empleo de la técnica anteriormente descrita por sí sola no representa una solución, sino que se necesita del poder de una tecnología que incorpore técnicas de correspondencia de cadenas, como las mencionadas anteriormente, así como el método del índice invertido. Se selecciona ***Elasticsearch***[38] en su versión 2.3.1, pues es un motor de búsqueda y análisis que funciona de forma distribuida en tiempo real empleado para la búsqueda de texto completo, estructurado y para la obtención de análisis sobre información. Permite explorar datos a una velocidad y escala que nunca antes había sido posible gracias a la utilización de *Apache Lucene*[39], la librería de alto rendimiento más avanzada que existe actualmente para motores de búsqueda [40].

Elasticsearch posibilita poderosas búsquedas de texto completo a través del empleo de índices invertidos, auto completado, sugerencia de semejanzas, soporte para técnicas de minería de texto, funcionalidades multilenguaje, así como consultas extensivas DSL (del inglés *Domain-Specific Language*), un lenguaje expresivo de búsqueda que se utiliza para exponer la mayoría del poder de *Lucene* a través de una simple interfaz *JSON*[40]. El contenido se almacena en estructuras llamadas ***documentos*** bajo particiones con el nombre de ***índices*** sobre las cuales Elasticsearch realiza las búsquedas. Esta arquitectura libre de esquemas proporciona a los desarrolladores flexibilidad y facilidad de configuración a la hora de utilizarla y

[38] https://www.elastic.co/products/elasticsearch.
[39] https://lucene.apache.org/core/
[40] Estructura que se utiliza para intercambio de información, presenta un formato {llave: valor}

permite que sea capaz de indexar y buscar contenido sin estructurar, haciéndola perfecta para proyectos pequeños , o grandes almacenes de datos incluso con petabytes[41] de datos sin estructurar [41].

Entre los casos de usos más destacados se encuentran:

- ***Wikipedia***[42]: utiliza *Elasticsearch* para proveer búsqueda de texto completo con resaltado de palabras claves, búsquedas del tipo '*mientras se escribe*', así como el ofrecimiento de sugerencias.

- ***Stack Overflow***[43]: combina búsquedas de texto completo con consultas de geolocalización[44], para a través de la funcionalidad '*más como esto*', encontrar preguntas y respuestas relacionadas a las enviadas por los usuarios.

- ***GitHub***[45]: utiliza *Elasticsearch* para realizar búsquedas sobre más de 130 billones de líneas de código.

1.5.4. Sistema gestor de bases de datos

PostgreSQL en su versión 9.3, es un Sistema Gestor de Bases de Datos (SGBD) objeto-relacional que se encuentra distribuido bajo la licencia de *software* libre *BSD*. Es mantenido por la organización *PostgreSQL Global Development Team* y cuenta con una amplia comunidad de usuarios y programadores que colaboran activamente. Se centra en que el *software* sea robusto, de calidad, fácil de mantener y se destaca por su estabilidad, potencia, robustez y facilidad de administración. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios, accediendo paralelamente al sistema [42].

Lo que diferencia a *PostgreSQL* de otras bases de datos es la facilidad con que se puede extender, por lo general, sin compilar ningún código. Este SGBD no sólo incluye características avanzadas como funciones de ventana SQL y replicación de streaming, raramente encontradas en otras bases de datos de código

[41] Unidad de almacenamiento de información, equivale a 10^{15} bytes
[42] Enciclopedia libre, políglota y editada colaborativamente. Disponible en https://es.wikipedia.org
[43] Sitio web utilizado por la comunidad de desarrolladores informáticos, en la cual otros desarrolladores pueden encontrar soluciones a problemas de programación en diferentes lenguajes. Sitio web: http://stackoverflow.com/.
[44] Capacidad para obtener la ubicación geográfica real de un objeto, como un radar, un teléfono móvil o un ordenador conectado a Internet
[45] Plataforma colaborativa para alojar proyectos utilizando el sistema de control de versiones Git

abierto, sino que también las ejecuta rápidamente y supera en muchas ocasiones a otras bases de datos incluso propietarias, como es el caso de Oracle, SQL Server y DB2 [43].

1.5.5. Entorno de desarrollo

PyCharm en su versión 4.5.4, es un entorno de desarrollo integrado multiplataforma utilizado para desarrollar en el lenguaje de programación *Python.* Provee funcionalidades que permiten una experiencia única y aumentan en gran medida la productividad:

- Completamiento de código de manera inteligente y señalamiento de errores con reparación de los mismos de forma automatizada.
- Integración con marcos de trabajo de desarrollo web modernos como *Django.*
- Depurador integrado y herramientas de pruebas.
- Soporte para bases de datos e integración con sistemas de control de versiones.

En adición a *Python*, *PyCharm* soporta *JavaScript*, *CoffeeScript*, *TypeScript*, *HTML*/*CSS*, *Cython*, lenguajes de plantilla, *AngularJS*, *Node.js* y más [44].

1.6. Conclusiones parciales

El estudio de las principales alternativas a la solución propuesta, evidenció la utilización del indexado del contenido de los documentos.

El análisis de las técnicas de procesamiento de lenguaje natural puso en evidencia la necesidad de una estrategia de pre-procesado con el objetivo de producir una optimización del espacio de búsqueda y un impacto significativo en el rendimiento del entorno colaborativo.

La selección de Python como lenguaje de programación para realizar el procesamiento de los documentos, Elasticsearch como motor de búsqueda y AngularJS como framework de desarrollo web, demostró el potencial de estas modernas tecnologías, en aras de lograr la creación de una aplicación de alta calidad, guiada por la metodología de desarrollo XP.

Capítulo 2

Propuesta de solución

En este capítulo se abordan las características de la propuesta de solución. Se desarrolla un modelo de dominio que reúne los principales conceptos a tratar. Posteriormente se hace énfasis en la evolución de dicha solución a través de las fases iniciales: Planificación y Diseño, de la metodología de desarrollo de *software Extreme Programming (XP)*, presentando los diferentes artefactos generados por la misma. Finalmente se selecciona la estrategia de pre-procesado orientada a optimizar el espacio de búsqueda.

2.1. Modelo de dominio

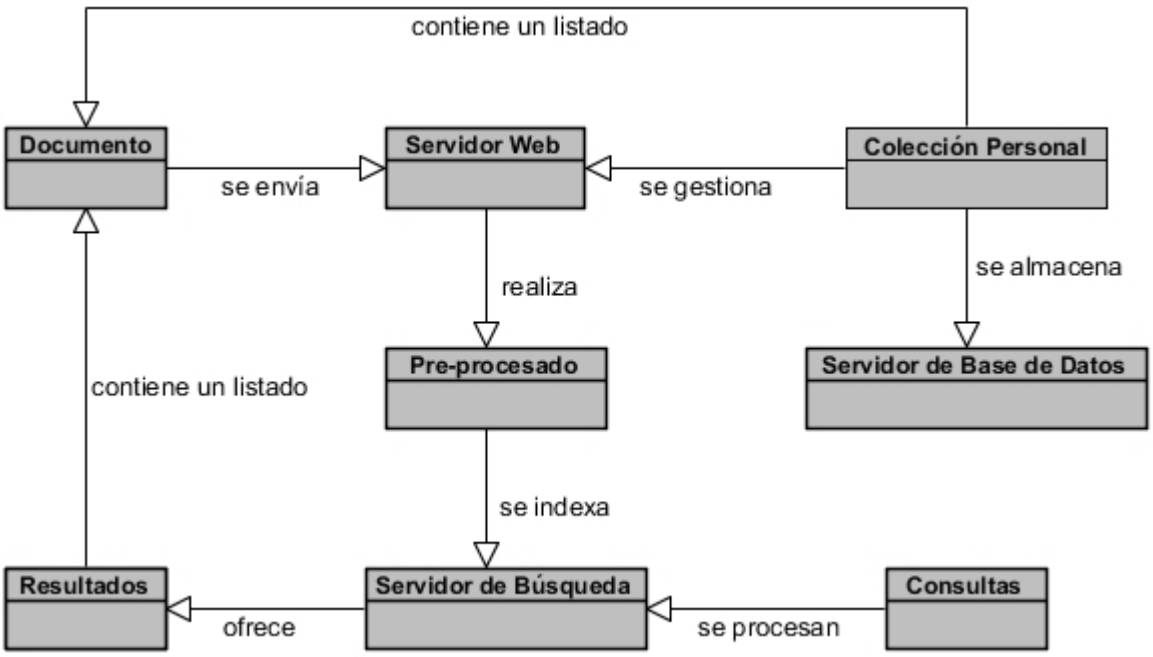

Figura 1: Modelo de dominio.

2.1.1. Glosario de conceptos del modelo de dominio

Documento: Libro electrónico en formato *PDF* añadido por el usuario a la plataforma.

Servidor Web: Es el encargado de procesar los documentos subidos por los usuarios y de la gestión de la colección personal. También posee la responsabilidad de almacenar los documentos en formato *PDF*.

Colección Personal: Relación de pertenencia que existe entre los usuarios y los documentos de su preferencia.

Servidor de Base de Datos: Encargado de almacenar la colección personal de cada usuario, así como las credenciales que intervienen en el proceso de autenticación.

Pre-procesado: Estrategia aplicada al contenido de los documentos, en orden de reducir su tamaño para su indexación en el servidor de búsqueda.

Consultas: Criterios de búsquedas provistos por el usuario desde la interfaz web.

Servidor de Búsqueda: Se encarga de realizar las búsquedas sobre el contenido de los libros, así como sus metadatos[46], de acuerdo a las consultas realizadas por los usuarios.

Resultados: Listado de documentos cuyo contenido o metadatos coinciden con las consultas realizadas al servidor de búsqueda.

2.2. Metáfora

El entorno colaborativo está compuesto por dos módulos con el objetivo de lograr la separación de los conceptos *vista* y *lógica de negocio*. El primer módulo es el encargado de la seguridad del sistema, así como de manejar y procesar para su búsqueda los documentos añadidos por los usuarios y gestionar la pertenencia de los mismos a su colección. Está construido en forma de Interfaz de Programación de Aplicaciones[47] (*API*, por sus siglas en inglés), lo que propicia que el entorno colaborativo pueda ser utilizado

[46] Se refiere a la información de la información, en este caso: autor del libro, ISBN, fecha de publicación, etc.
[47] Capa de abstracción que contiene procedimientos que pueden ser utilizados por un software.

por cualquier *software* que cumpla los mecanismos de seguridad establecidos, como un servicio de almacenamiento y búsqueda de documentos, característica esta que propicia un aumento en la disponibilidad de la documentación.

El segundo módulo hace referencia a una aplicación de una sola página o *SPA*[48]*,* por sus siglas en inglés. Esta es capaz de proveer al usuario de la experiencia que se tiene al trabajar con aplicaciones de escritorio. Dicha rapidez y fluidez se obtiene por el hecho de que los recursos necesarios se cargan inicialmente en el navegador y se muestran al usuario en dependencia de sus acciones reduciendo así el tiempo de carga de las interfaces [45].

La comunicación con el servidor web y el servidor de búsqueda desde dicha aplicación se realiza utilizando la arquitectura *REST*[49] que plantea que los servidores que contienen los recursos no deben depender del conocimiento de las interfaces de usuario o su estado. De la misma forma los clientes que la utilicen no deben de preocuparse de qué manera se almacenan o procesan los datos. Como resultado se obtienen servidores con un mayor rendimiento y escalabilidad y clientes con una mejor portabilidad [46].

Para la obtención de los recursos necesarios bajo la demanda de los usuarios, se hacen una serie de llamadas *AJAX* enviando peticiones *HTTP*[50] como *GET*, *POST*, *PUT*, *DELETE*, donde cada uno de ellos contiene la información necesaria para ser procesados por los servidores y enviar la respuesta adecuada, produciendo además un aumento significativo en rendimiento.

2.3. Requisitos funcionales

Los requisitos funcionales explican en detalle las tareas que el sistema debe ser capaz de realizar. A continuación, se muestran los requisitos funcionales concernientes al sistema propuesto.

1. Autenticar usuario

[48] Del inglés Single Page Application, es un sitio web que cabe en una sola página con el propósito de dar una experiencia más fluida a los usuarios como una aplicación de escritorio.
[49] Estilo de arquitectura software para sistemas hipermedia (texto, audio, etc.) distribuidos como la World Wide Web.
[50] Protocolo de comunicación que permite las transferencias de información en la World Wide Web.

2. Subir documento
3. Realizar búsquedas
4. Adicionar documento a la colección
5. Eliminar documento de la colección
6. Compartir documentación
7. Comentar documentación

2.4. Requisitos no funcionales

Los requisitos no funcionales son aquellos que no describen información a guardar ni acciones a realizar, sino características de funcionamiento del sistema o requerimientos para su puesta en marcha.

- ✓ Interfaz

    1. El entorno colaborativo presentará una interfaz agradable y fácil de utilizar donde predomine el color azul.

- ✓ Usabilidad

    2. El entorno colaborativo tendrá un diseño adaptable para su correcta visualización en ordenadores y dispositivos móviles.

- ✓ Rendimiento

    3. El entorno colaborativo debe ser escalable, permitiendo incorporarle nuevas funcionalidades sin afectar las existentes.

    4. Debe responder a las búsquedas en un tiempo máximo de 2 segundos.

- ✓ Seguridad

    5. Los libros subidos no podrán ser eliminados por los usuarios, así como los datos indexados en el servidor de búsqueda.

    6. Los errores mostrarán información que no comprometan la seguridad e integridad del entorno colaborativo.

- ✓ Software

    7. El cliente necesitará *Google Chrome* en su versión 29 o superior, o *Mozilla Firefox* en su versión 45 o superior como navegador web.

    8. Intérprete de *Python* 3.4.3 en el servidor web.

    9. *Elasticsearch* en su versión 2.3.1 como servidor de búsqueda.

    10. *Django* 1.8 en el servidor web.

    11. *PostgreSQL* en su versión 9.3 en el servidor de base de datos.

- ✓ Hardware

    12. En el servidor web *CPU* 3.0 *GHz* o superior y 1 *Gb* de *RAM* o superior.

    13. En el servidor se búsqueda *CPU* 3.0 *GHz* o superior y 4 *Gb* de *RAM* o superior.

    14. *CPU* 2.3 *GHz* o superior y 1 *Gb* de *RAM* o superior en el servidor de base de datos.

    15. 120 *Gb* como mínimo para el almacenamiento de los documentos digitales en el servidor web.

- ✓ Operacionales

    16. Los documentos subidos a la plataforma estarán en idioma inglés o español y en formato *PDF*.

2.5. Historias de usuario

Las historias de usuario conforman la parte central de la metodología XP pues definen lo que se debe construir en el proyecto de *software*. Tienen una prioridad asociada, definida por el cliente con el objetivo de indicar cuáles son las más importantes para el resultado final. Se encuentran divididas en tareas y su tiempo será estimado por los desarrolladores. La estimación de puntos es de acuerdo al tiempo que toma la tarea en ser desarrollada, donde un punto equivale a una semana ideal de programación. Las historias de usuario generalmente valen de 1 a 3 puntos. A continuación, se describen las historias de usuario de mayor prioridad, las demás se encuentran descritas en el Anexo A. Historias de Usuario.

Tabla 3: HU1 Autenticar Usuario.

	Historia de Usuario
Número: HU1	**Usuario:** Todos
Nombre de historia: Autenticar usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Medio
Puntos estimados: 1.5	**Iteración asignada:** 1
Programador Responsable: José Jairán Breijo Rodríguez	
Descripción: El usuario inserta su identificador (Ejemplo: jjbreijo) y contraseña en el formulario de autenticación para acceder a las funcionalidades que esté autorizado.	
Observaciones: Si el sistema no identifica al usuario le mostrará una notificación sobre este problema.	

Tabla 4: HU2 Subir Documento.

	Historia de Usuario
Número: HU2	**Usuario:** Todos
Nombre de historia: Subir documento	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 1.5	**Iteración asignada:** 1

Programador Responsable: José Jairán Breijo Rodríguez
Descripción: El usuario activa el cuadro de diálogo para subir el documento, proporciona los datos correspondientes y selecciona el archivo en formato PDF.
Observaciones: El sistema debe notificar al usuario cuando la operación termine.

Tabla 5: HU3 Realizar Búsquedas.

	Historia de Usuario
Número: HU3	**Usuario:** Todos
Nombre de historia: Realizar búsquedas	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 2
Programador Responsable: José Jairán Breijo Rodríguez	
Descripción: El usuario escribe en el cuadro de búsqueda el criterio que desee y el sistema debe mostrar un listado con los libros que coincidan con el mismo.	
Observaciones: El sistema debe mostrar el número de resultados y el tiempo de búsqueda .	

2.6. Plan de entrega

Una vez concebidas las historias de usuario con la prioridad asignada por el cliente, se toman acuerdos sobre el contenido de las entregas y se determina un cronograma en conjunto con el mismo. Una entrega debería obtenerse en no más de tres meses. Para la presente investigación, se definieron tres iteraciones, en las que se codificarán las siete historias de usuario.

Tabla 6: Plan de entrega.

Entregable	**Iteración**	**Fin de iteración**
Autenticación y subida de documentos	1	febrero de 2016
Búsquedas	2	marzo de 2016
Gestión y socialización de documentos	3	mayo de 2016

2.7. Plan de iteraciones

Los proyectos que utilicen *XP* como metodología de desarrollo de *software* deben ser sometidos a iteraciones de aproximadamente 3 semanas. En este período de tiempo se lleva a cabo el desarrollo de un conjunto de funcionalidades que se corresponden a las definidas en las Historias de Usuario, con el objetivo de obtener un prototipo funcional de cada una de ellas. Para la determinación de la cantidad de iteraciones se tiene en consideración la velocidad de proyecto, pues este indicador es el que determina la cantidad de Historias de Usuario que pueden ser implementadas antes de una fecha determinada. Al final de la última iteración el sistema estará listo para entrar en producción. Para la propuesta de solución se define el siguiente plan de iteraciones:

Tabla 7: Plan de iteraciones.

Iteración	**No. HU**	**Historia de Usuario**	**Semanas estimadas**
1	1	Autenticar usuario	3
	2	Subir un documento	
2	3	Realizar búsquedas	3
3	4	Adicionar documento a la colección	4
	5	Eliminar documento de la colección	
	6	Compartir documentación	
	7	Comentar documentación	

En la primera iteración se le da prioridad a aquellas Historias de Usuario que son vitales para el funcionamiento del sistema, como es el caso de la autenticación y la posibilidad de adicionar nuevos documentos. En la siguiente se le permite al usuario realizar búsquedas para consultar los libros de su preferencia. Finalmente se desarrollan aquellas que tienen menor complejidad, como la posibilidad de adicionar y eliminar documentos de su colección personal, socializar la documentación y opinar respecto a la misma.

2.8. Tarjetas *CRC*

Para el diseño de las aplicaciones la metodología *XP* propone el uso de tarjetas *CRC* como alternativa a los diagramas *UML* de las clases, permitiendo al programador enfocarse en el desarrollo orientado a objetos. Las siglas *CRC* se refieren a Clases, Responsabilidades y Colaboradores. En ellas se plasman las responsabilidades que tienen cada uno de los objetos, la clase a la cual pertenece y con las que colaboran con cada responsabilidad. A continuación, se presentan las principales clases identificadas (consultar el resto en el Anexo B. Tarjetas CRC).

Tabla 8: Tarjeta CRC. Libro.

Clase Libro	
Responsabilidades	**Colaboradores**
Crear libro	Clase Usuario, Clase Carpeta
Modificar libro	Clase Carpeta
Indexar libro	

2.9. Arquitectura

Por décadas la arquitectura de *software* ha recibido una atención primordial en el campo de la ingeniería de *software*. Con el crecimiento de la industria, ha quedado en evidencia que un diseño previo y cuidadoso de la arquitectura de un producto puede reducir enormemente el número de fallas del mismo. Indicadores de calidad como la eficiencia, usabilidad, confiabilidad y seguridad pueden ser verificados y estimados con respecto al diseño arquitectural, mucho antes de que cualquier código sea escrito [47].

Teniendo en cuenta esto, se selecciona para el desarrollo del sistema propuesto la arquitectura **n-capas**, específicamente en 4 capas: Presentación, Procesamiento, Búsqueda y Datos. A continuación, se muestra la arquitectura del entorno colaborativo:

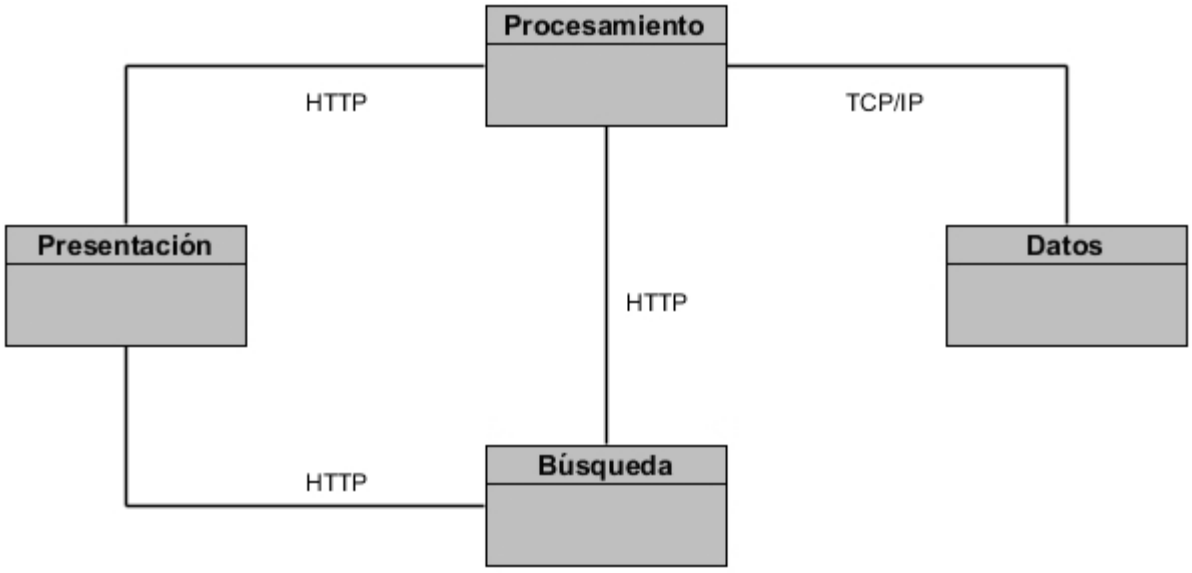

Figura 2: Arquitectura del entorno colaborativo.

Capa de Presentación: Esta capa se relaciona con las capas de Procesamiento y Búsqueda. Básicamente, en el navegador del usuario es descargada una aplicación de una sola página de *AngularJS* (*SPA*, por sus siglas en inglés), que emula el comportamiento modelo-vista-controlador, a través de la carga dinámica de las vistas que manejan los recursos que el usuario necesite. Estos pueden ser la colección de libros del usuario autenticado, o los resultados de la búsqueda a partir de un criterio provisto.

Capa de Procesamiento: Es la encargada del proceso de verificación de las credenciales del usuario para su autenticación. Una vez que el mismo tenga permisos podrá subir documentos. Debido a que múltiples usuarios pueden tener un mismo documento en su colección local, una vez que deseen añadirlos al entorno colaborativo, se debe garantizar que no existan archivos duplicados. Para ellos se realiza la comparación de los metadatos obtenidos de dichos archivos, con aquellos que ya fueron indexados para su búsqueda. En caso de existir una coincidencia se le notifica al usuario que ya hay un libro en existencia similar al suyo. De esta forma se reduce la redundancia al mantener en el servidor un solo archivo por cada documento añadido, el cual utilizarían todos los usuarios.

Esta capa se encarga de aplicar técnicas de procesamiento de lenguaje natural, con el objetivo de seleccionar solamente el contenido que puede ser de interés para los usuarios en el momento de realizar

las búsquedas. Básicamente es un proceso de limpieza y categorización del texto extraído para su posterior indexación. Por último, permite gestionar la colección personal de cada usuario, ofreciendo además la posibilidad de compartir dicha documentación y comentar respecto a ella.

Capa de Búsqueda: Su responsabilidad es la de mantener un índice con el contenido de los documentos y los metadatos asociados a los mismos. De esta forma, *Elasticsearch* se encarga de encontrar entre el total de libros añadidos a la plataforma aquellos cuyo contenido tenga la mayor semejanza con el criterio de búsqueda provisto por el usuario. Debido a que no es necesario estar autenticado para poder buscar, las consultas son elaboradas y enviadas directamente desde la capa de Presentación. La eliminación de la necesidad de hacer peticiones extras al servidor web produce un aumento significativo en el rendimiento del entorno colaborativo.

Capa de Datos: Es solamente accesible desde la capa de Procesamiento donde está contenida la lógica del sistema, aumentando de esta manera la seguridad. En ella se tiene una descripción de la colección personal de cada usuario autenticado. Es imprescindible tener en cuenta que los documentos no se almacenan físicamente en la base de datos sino en el servidor web, pues esto causaría una pérdida de rendimiento y escalabilidad considerable.

2.10. Patrones de diseño

Los patrones de diseño se utilizan en el desarrollo de *software* para ofrecer soluciones reutilizables y documentadas a problemas de diseño comunes. A pesar de que un patrón de diseño no es una solución finalizada que puede ser transformada directamente en código, es una descripción de cómo solucionar un problema que puede emerger en diferentes situaciones [48]. Para la propuesta de solución se hizo uso de varios *Patrones Generales de Software para Asignar Responsabilidad (GRASP)*.

El patrón **Experto** plantea que se debe asignar la responsabilidad a la entidad que posee la información necesaria para cumplir con la responsabilidad requerida. Este patrón es utilizado en el servicio[51] encargado

[51] Se refiere a una característica ofrecida por el framework de JavaScript: AngularJS. Es un conjunto de funcionalidades que se definen con una estructura específica y pueden ser requeridos a demanda mediante la inyección de dependencias.

de la autenticación en la capa de Presentación, pues este se encarga de encapsular la lógica referente al manejo de credenciales y la comunicación con el servidor web para ejecutar el mecanismo de autenticación.

Bajo Acoplamiento expone la necesidad de tener una alta reutilización de los componentes con una mínima dependencia entre ellos. Debido a la utilización de la arquitectura *REST* para la comunicación entre la capa de procesamiento, el servidor web y el servidor de búsqueda, se produce una clara separación de responsabilidades en cada una de ellas. Además, el procesamiento de datos queda encapsulado en cada una independientemente de su representación interna, haciendo posible su reutilización con una mínima dependencia. Ejemplo de ello es el procesamiento de un libro: una vez que el cliente lo añade al entorno colaborativo, la capa de procesamiento extrae el contenido y metadatos y los envía a la capa de búsqueda. Luego de su indexación, dichos datos pueden ser consumidos por el cliente a través de las búsquedas, sin este conocer cómo están representados internamente.

Por otra parte, la utilización de *AngularJS* trae consigo un mecanismo llamado *inyección de dependencias.* Este es el responsable de crear y distribuir los componentes de la aplicación, lo que posibilita así la carga dinámica y el manejo del ciclo de vida de cada uno de ellos. Ejemplo de lo anterior es la creación de servicios; funcionalidades que aíslan la lógica de negocio subyacente y que pueden ser requeridos en cualquier lugar, con la peculiaridad de que pueden ser probados de forma independiente [49].

En la capa de Presentación se hace uso intensivo de las búsquedas. Este proceso fue implementado en forma de servicio, haciendo posible su reutilización en cualquier funcionalidad que requiera de hacer búsquedas. La colección personal de cada usuario está construida de forma similar, siendo posible acceder a las funcionalidades que se encargan de su gestión desde cualquier parte de la propuesta de solución.

Además de los mencionados anteriormente, se utilizan algunos patrones de diseño de la *Banda de los Cuatro (GoF)*. Específicamente se utilizaron los patrones ***Singleton*** y **Fachada**.

El patrón ***Singleton*** es empleado cuando se necesita solamente una instancia de una clase. A través de la utilización de los módulos[52] de *Python* se está en presencia de este patrón, pues siempre que se requiera un mismo módulo en archivos diferentes, el intérprete del lenguaje se encargará de devolver la misma

[52] Conjunto de funcionalidades agrupadas en un mismo fichero, utilizables por otros módulos.

instancia. Este comportamiento también se observa en la capa de Presentación, mediante el uso de servicios de *AngularJS*. Siempre que se requiera el mismo servicio en componentes diferentes, el framework se encargará de retornar la misma instancia.

Fachada es utilizado cuando es necesario tener una interfaz única para acceder a un conjunto de funcionalidades de un subsistema. La capa de Procesamiento está construida en forma de *API* y para el acceso a los recursos que ella brinda, es necesario realizar una petición HTTP como las anteriormente mencionadas (*GET*, *POST*, *PUT*, *DELETE*). Por ejemplo: para adicionar un libro se debe hacer un *POST* a la URL [53] http://<host>/api/books y para eliminarlo, se debe hacer una petición *DELETE* a la URL http://<host>/api/books/<id>, siendo <host> la dirección donde se encuentra el servidor web e <id> el identificador del documento en la base de datos. En la capa de Presentación, el empleo del servicio **$resource** provisto por *AngularJS*, utilizado para administrar la colección de libros, abstrae al programador de este número de combinaciones posibles, brindándole una única interfaz con la cual puede realizar dicha gama de operaciones.

2.11. Estrategia de pre-procesado de documentos

Con el objetivo de disminuir el tamaño del espacio de búsqueda, el texto de los documentos pasa por un proceso de filtrado donde se aplican las técnicas de procesamiento de lenguaje natural mencionadas en el capítulo anterior. Para su ejecución se emplea *NLTK*, un conjunto de bibliotecas que se ha convertido en una de las mejores herramientas para la construcción de sistemas de procesamiento de lenguaje natural [50]. Estas proporcionan clases básicas para la representación de los datos relevantes a dicho procesamiento, interfaces estándar para realizar tareas tales como el etiquetado de la parte del discurso de cada palabra, el análisis sintáctico y clasificación de texto; además de implementaciones estándar para cada tarea que se puedan combinar para resolver problemas complejos, siendo respaldada por una extensa documentación [51].

[53] Ruta que se encuentra en la caja de texto ubicada en la barra de navegación del navegador, sirve para ubicar de manera precisa en un servidor, cualquier recurso: una imagen, un video o una página web.

Para el procesamiento del contenido de los documentos, se realiza en un primer momento un proceso de ***tokenización***. Este consiste en filtrar el contenido basado en una expresión regular[54] cuyo resultado sea solamente la selección de palabras, omitiendo números, signos de puntuación y otros caracteres, con el objetivo de eliminar todo lo que no represente información valiosa en relación con la temática del documento.

Posteriormente se utilizan los **métodos basados en las leyes de *Zipf*. (Métodos Z)** para eliminar las ***stop-words***. Esto significa un incremento en el rendimiento de *Elasticsearch*, pues esta herramienta calcula la relevancia de todos los documentos que, de alguna forma, coinciden con la consulta enviada. Mientras que la mayoría de las palabras ocurren en una proporción menor al 0.1% de todos los documentos, palabras como ***'el'*** pueden ser encontradas en la mayoría de ellos, lo que requería un procesamiento adicional para encontrar términos como este, que poseen un significado relativamente efímero.

El texto resultante es procesado mediante el empleo del ***stemmer* de *Porter*** para remplazar las palabras por su *stem* o raíz. Finalmente se indexan solamente aquellas que, por su significado puedan brindar alguna información sobre el contenido del documento. Se selecciona este algoritmo de truncado por el hecho de no presentar los problemas de almacenamiento y consumo de memoria que tiene el método estadístico, así como la no dependencia de un diccionario que presenta el método mixto mencionado. *NLTK* posee además un módulo que brinda las funcionalidades de dicho *stemmer* y su implementación como un marco de trabajo ('*Snowball*[55]') que ofrece soporte para múltiples lenguajes, entre ellos inglés y español, idiomas soportados por el entorno colaborativo.

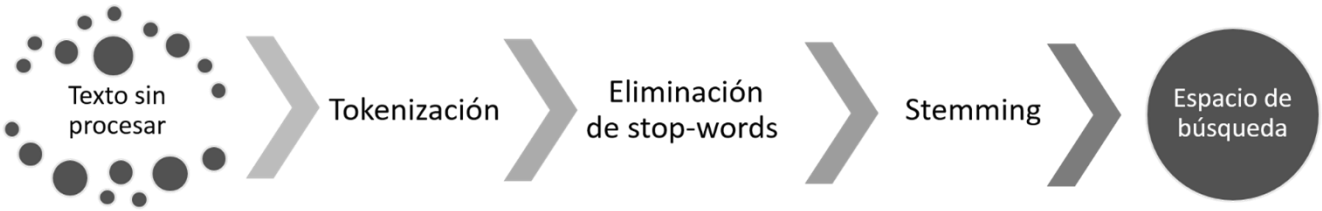

Figura 3: Estrategia de pre-procesado de documentos

[54] Secuencia de caracteres que forma un patrón de búsqueda, principalmente utilizada para la búsqueda de patrones de cadenas de caracteres
[55] http://www.nltk.org/api/nltk.stem.html

A continuación, se presenta un ejemplo de la aplicación de la estrategia de pre-procesado seleccionada al siguiente fragmento de texto:

"*La clave de los motores más exitosos fue transformar el código JavaScript en código máquina para lograr velocidades de ejecución similares a aquellas encontradas en aplicaciones de escritorio*" [52].

Resultado de la tokenización (colección de *tokens*):

['la', 'clave', 'de', 'los', 'motores', 'más', 'exitosos', 'fue', 'transformar', 'el', 'código', 'javascript', 'en', 'código', 'máquina', 'para', 'lograr', 'velocidades', 'de', 'ejecución', 'similares', 'a', 'aquellas', 'encontradas', 'en', 'aplicaciones', 'de', 'escritorio']

Resultado de la eliminación de *stop-words*:

['clave', 'motores', 'exitosos', 'transformar', 'código', 'javascript', 'código', 'máquina', 'lograr', 'velocidades', 'ejecución', 'similares', 'aquellas', 'encontradas', 'aplicaciones', 'escritorio']

Resultado de la aplicación del *stemmer* de *Porter* (colección de *stems*):

['clav', 'motor', 'exit', 'transform', 'codig', 'javascript', 'codig', 'maquin', 'logr', 'veloc', 'ejecu', 'similar', 'aquell', 'encontr', 'aplic', 'escritori']

A pesar de la disminución en la cantidad de palabras que se obtiene en relación al texto original, el resultado del proceso de stemming puede parecer una alteración del contenido inicial que poseen los documentos. *Elasticsearch* como motor de búsqueda solventa este problema presentando soporte para *stemming* y utilizando internamente **consultas difusas**. Su objetivo es ofrecer resultados que potencialmente coincidan con el criterio de búsqueda; por ejemplo: semejanzas a palabras mal escritas, o aquellas palabras cuya distancia de *Damerau–Levenshtein* sea menor que dos. Dicha distancia es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra, donde una operación puede ser una inserción, eliminación, sustitución o transposición de dos caracteres.

2.12. Conclusiones parciales

La creación del modelo de dominio y la metáfora asociada al mismo permitió esclarecer conceptos claves en cuanto a la estructura y funcionamiento de la aplicación, así como la obtención de los requisitos no funcionales y funcionales, descritos en las historias de usuario.

La selección de una arquitectura n-capas orientada a la extensibilidad permitió que el sistema pueda ser utilizado como un servicio de consulta y almacenamiento de documentos. Además, la utilización de patrones de diseño GRASP y GOF garantizaron la organización del código y el empleo de buenas prácticas en el proceso de codificación.

Se seleccionó como estrategia de pre-procesado del contenido de los libros un proceso que involucra la *tokenización*, eliminación de *stop-words* y aplicación de *stemming*, orientado a la reducción del espacio de búsqueda.

Capítulo 3

Implementación y prueba

Con el objetivo de materializar la solución propuesta y cumplir con los requisitos obtenidos al inicio de la investigación, se lleva a cabo la fase de implementación y prueba. En ella quedan recogidos los estándares de codificación que deben ser empleados en el desarrollo del entorno colaborativo, los diagramas de componentes y despliegue donde se observan las dependencias lógicas que existen entre los componentes del *software* y los nodos necesarios para la puesta en marcha del sistema. Finalmente se muestran las pruebas realizadas para validar el correcto funcionamiento de la propuesta de solución.

3.1 Tareas de ingeniería

Las tareas de ingeniería representan el trabajo realizado durante una iteración expresado en tareas de programación. Debido a que las historias de usuario no brindan el suficiente nivel de detalle para llevar a cabo la implementación, las tareas de ingeniería juegan un papel fundamental al indicar a los programadores las acciones a realizar por cada una de ellas, donde cada tarea es asignada a un programador directamente. A continuación, se muestran las tareas de ingeniería implicadas en la iteración 1. Consultar el resto en el Anexo C. Tareas de Ingeniería.

Tabla 9: Tareas de ingeniería. Iteración 1

Iteración 1	
Historia de Usuario	**Tareas**
Autenticar usuario	1. Definir interfaz de autenticación. 2. Validar los datos introducidos por el usuario (nombre de usuario y contraseña) 3. Notificar al usuario si los datos son incorrectos.

Subir documento	1. Definir interfaz para el formulario de subida de documento. 2. Validar los datos introducidos por el usuario (formato del documento, título, autor, etiquetas, idioma). 3. Extraer contenido, metadatos y portada del documento. 4. Verificar que no existan duplicados a través de una comparación con el índice de búsqueda 5. Procesar contenido del documento (aplicar técnicas de procesamiento de lenguaje natural). 6. Indexación del documento en el servidor de búsqueda. 7. Notificar al usuario al finalizar este proceso.

3.2 Estándares de codificación

Entre las buenas prácticas a aplicar durante el proceso de desarrollo de *software* que propone la metodología XP se encuentran la refactorización del código y la propiedad compartida del mismo, de forma que todo el personal pueda corregir y extender cualquier parte del producto. La existencia de un estándar de codificación es necesario pues debe existir un estilo consistente a lo largo de toda la aplicación y no el preferido por cada programador involucrado. Esto posibilita que se pueden añadir nuevas funcionalidades, modificar ya existentes o depurar errores con gran facilidad y obtener un código con mayor legibilidad. A continuación, se define el estándar a utilizar en la implementación del entorno colaborativo:

- La declaración de variables, funciones, constantes y nombres de clases se hará en inglés.
- Se empleará el estilo *lowerCamelCase*[56] para nombres de identificadores, donde todos deberán comenzar con una letra.
- Las constantes serán escritas en mayúsculas.

[56] Estilo de escritura que se aplica a frases o palabras compuestas. La primera letra de cada una de las palabras es mayúscula con la excepción de que la primera letra es minúscula

- Se utilizarán 4 espacios para la identación[57] del código.
- El tamaño de línea máximo será de 79 caracteres para evitar la necesidad de realizar un desplazamiento horizontal para visualizar el código fuente.
- Se hará uso de comentarios en aquellas funcionalidades de mayor complejidad.
- En el caso del lenguaje *Python*, se seguirá el estándar de codificación *PEP* 0008[58].

3.3 Diagrama de componentes

Los diagramas de componentes se utilizan con el propósito de modelar la vista estática de un *software*, o sea, cómo un sistema se encuentra dividido y qué relaciones de dependencias existen entre dichas partes que lo conforman. A continuación, se expone el diagrama definido para el entorno colaborativo:

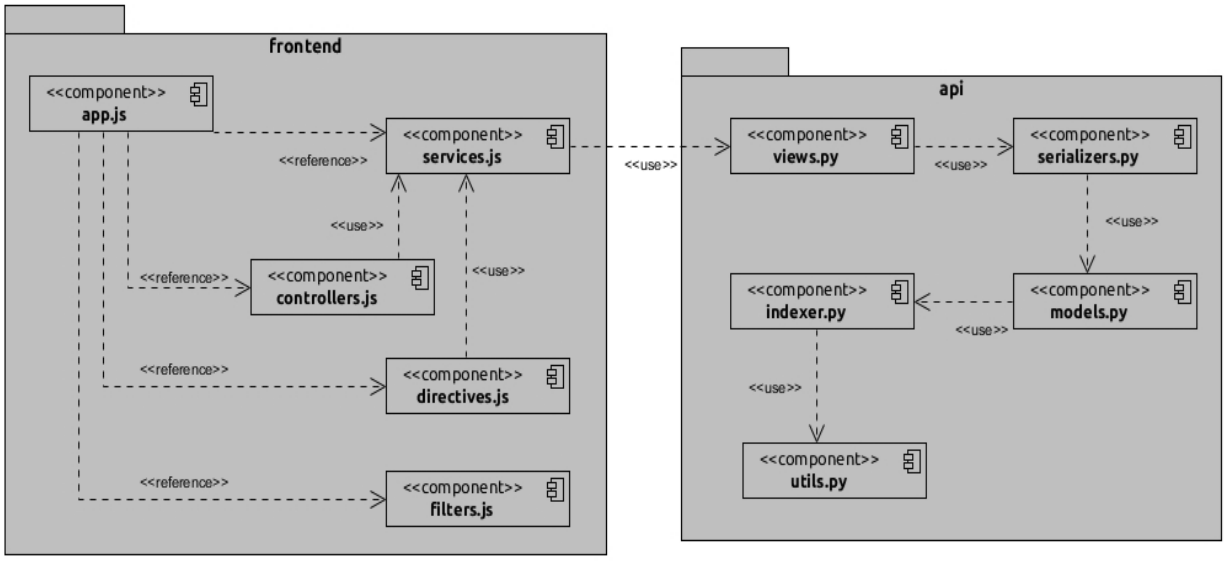

Figura 4: Diagrama de componentes.

[57] Significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para así separarlo del margen izquierdo y mejor distinguirlo del texto adyacente
[58] https://www.python.org/dev/peps/pep-0008/

Los componentes que intervienen en la aplicación quedan recogidos en dos grupos: **api** y **frontend**. El primero agrupa todas las funcionalidades que se ejecutan del lado del servidor y el segundo del lado del cliente, logrando de esta forma una clara separación de responsabilidades.

3.3.1 *API*

En el archivo ***models.py*** quedan definidos los modelos que sirven como abstracción para las tablas de la base de datos. El *ORM*[59] de *Django* es el encargado de asignar en la base de datos el tipo de dato subyacente adecuado para cada uno de las propiedades de dichos modelos. Luego de guardado un documento nuevo se hace uso de las funcionalidades contenidas en ***indexer.py***, el cual utiliza métodos auxiliares contenidos en ***utils.py*** para procesar el texto de dicho documento y enviar el resultado al servidor de búsqueda. En ***serializers.py*** se definen qué propiedades de cada modelo quedarán expuestas al usuario para su modificación o consulta. Finalmente, ***views.py*** expone los métodos necesarios para dicha modificación o consulta.

3.3.2 *FRONTEND*

Debido a la estructura de las aplicaciones que propone el framework *AngularJS*, el código que maneja la aplicación web del lado del cliente queda separado de la siguiente forma:

- ***app.js***: registra y configura las dependencias que tendrá la aplicación haciendo referencia a los controladores, directivas, servicios y filtros, así como librerías de terceros.
- ***filters.js***: se definen filtros encargados de cambiar la forma en que son representados los datos en la interfaz web de forma declarativa.
- ***controllers.js***: conjunto de funcionalidades que manejan los datos que se muestran en la interfaz web.

[59] Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional como motor de persistencia

- ***directives.js***: se definen nuevas etiquetas *HTML* de forma que su comportamiento es transparente al usuario tanto en funcionamiento como presentación.

- ***services.js***: contiene las funcionalidades encargadas de consumir los datos provenientes de la ***API*** necesarios para la aplicación. Dichas funcionalidades son reutilizadas en los controladores y directivas.

3.4 Diagrama de despliegue

El diagrama de despliegue es una herramienta muy útil que se emplea para exponer los elementos de configuración del despliegue de una aplicación y las conexiones entre estos elementos. Modelan la arquitectura de un sistema en tiempo de ejecución, permitiendo visualizar los diferentes nodos[60] que lo componen así como las relaciones físicas que existen entre ellos [53].

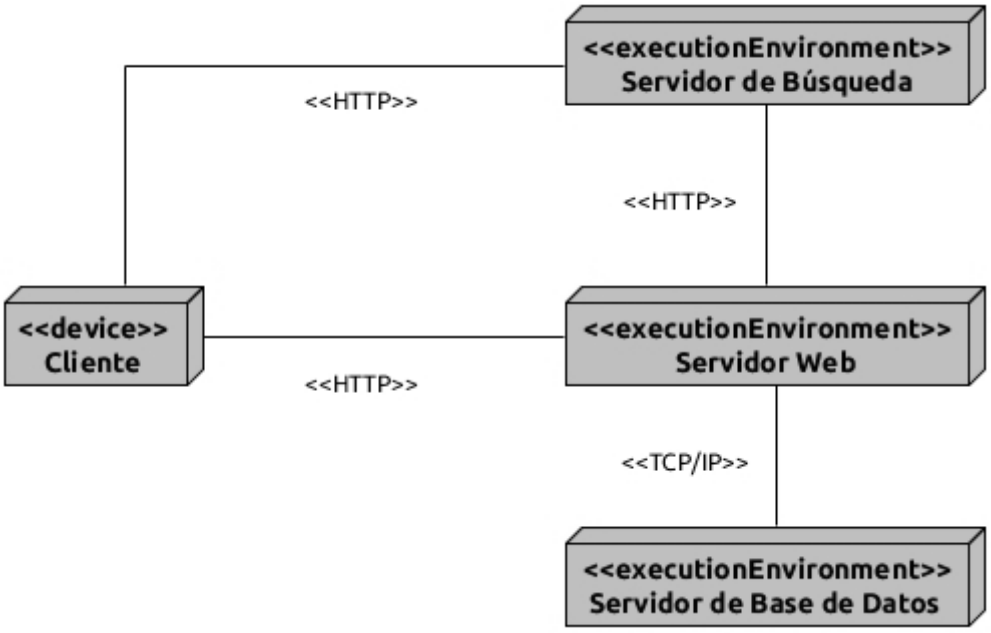

Figura 5: Diagrama de despliegue.

[60] Un nodo es un objeto físico en tiempo de ejecución, es decir, una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento.

La comunicación entre el cliente, el servidor web y el servidor de búsqueda, se realiza utilizando *HTTP* como protocolo para la transferencia de información. Ello permite hacer uso extensivo de la arquitectura *REST* en el sistema y lograr de esta manera una clara separación de responsabilidades. La separación física del servidor web y el servidor de búsqueda produce una mejora en cuanto a rendimiento, pues la carga que significa el procesamiento de nuevos documentos queda aislada del proceso de búsqueda. Por otra parte, la seguridad del sistema se incrementa al ser posible acceder al servidor de base de datos solamente desde el servidor web.

Cliente: Estación de trabajo que necesita un navegador web para conectarse al servidor web y al servidor de búsqueda utilizando el protocolo de comunicación *HTTP*.

Servidor Web: Estación de trabajo que hospeda el código fuente de la aplicación, así como los libros en formato PDF añadidos por los usuarios.

Servidor de Búsqueda: Instancia de *Elasticsearch* que almacena y realiza búsquedas sobre el contenido de los documentos y sus metadatos, procesados en el servidor web.

Servidor de Base de Datos: Servidor responsable de almacenar de los datos del sistema. Es solamente accesible desde el servidor web para garantizar su seguridad.

3.5 Interfaz gráfica

La interfaz de usuario es el medio con que el mismo puede comunicarse con una máquina, un equipo o una computadora y comprende todos los puntos de contacto entre la persona y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar. A continuación, se muestran las principales interfaces del entorno colaborativo de administración de documentación.

3.5.1 Subir documento

Cuando un usuario desea añadir un libro a la plataforma, activa el formulario donde quedan recogidos todos los datos del mismo: archivo en formato PDF, título, autores, etiquetas e idioma. Dichos archivos pueden proceder de disímiles fuentes y en ocasiones el nombre del mismo no coincide con el título del libro. Por

esta razón, el usuario debe proveer al menos el título y el idioma en que está escrito el documento, para poder aplicarle el pre-procesado correcto. Una vez enviado, se realiza una comparación entre sus metadatos y los demás libros indexados para garantizar que no existan duplicados y se almacena en el servidor web.

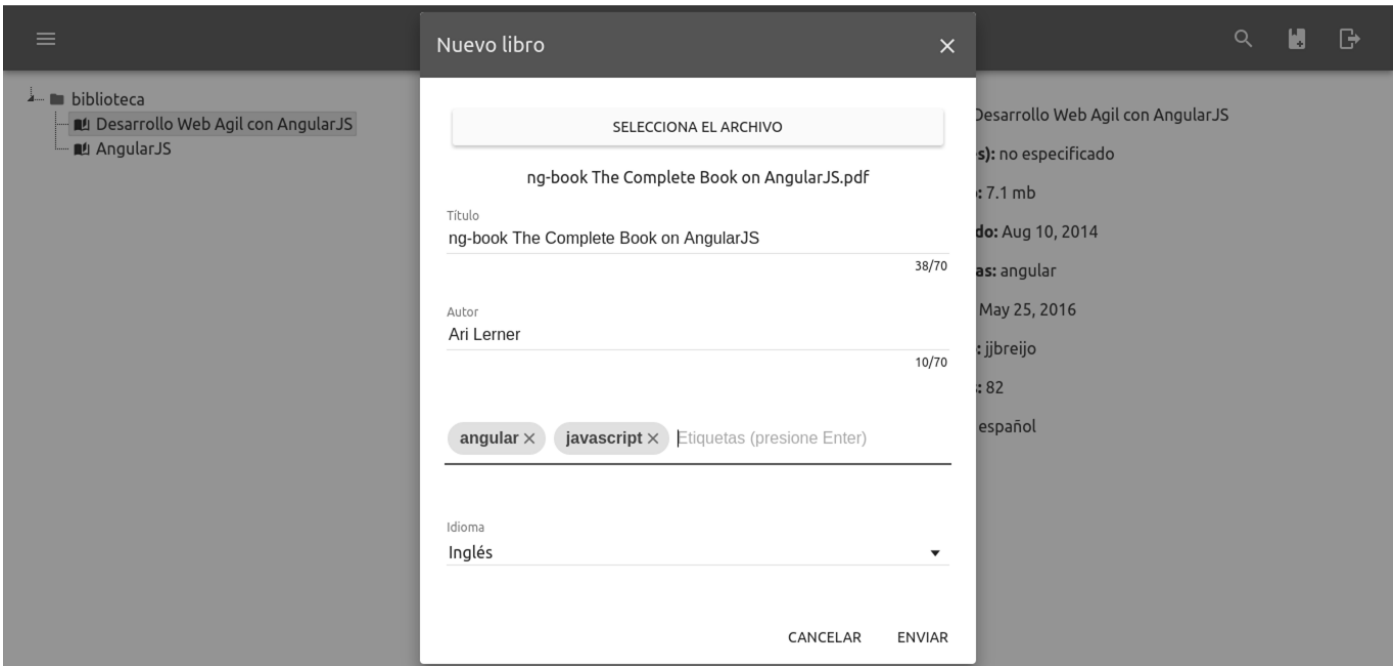

Figura 6: Añadir libro a la plataforma.

3.5.2 Colección personal

Una vez terminado el proceso de subida el documento es adicionado a la colección personal del usuario que lo añadió. Esta brinda la posibilidad de gestionar su organización mediante un componente con estructura de árbol, que simula una jerarquía de carpetas y ficheros análoga a los sistemas de ficheros de *Linux* o *Windows*. Dicho componente permite arrastrar y soltar los documentos en la ubicación que se desee y ofrece las operaciones correspondientes a cada elemento, a través de un menú contextual. Una vez seleccionado un libro, se muestra su información y se puede proceder a su lectura al activar la portada.

Figura 7: Colección personal.

3.5.3 Interfaz de búsqueda

A través de esta interfaz el usuario provee el criterio de búsqueda que desea y el sistema le ofrece en respuesta todos los documentos que de alguna forma estén relacionados con el mismo. Los resultados se ordenan de forma descendente, o sea, primero se listarán aquellos que mayor relación tengan con la consulta enviada. En caso de obtener múltiples resultados se elimina la necesidad de realizar una paginación[61] mediante el empleo de una técnica llamada *infinite scroll,* donde los resultados que no sean visibles en un primer momento irán apareciendo a medida que el usuario se desplace hacia abajo. Se brinda además la posibilidad de autocompletar las consultas con las palabras más relevantes de cada documento con el objetivo de mejorar la usabilidad.

[61] Separación de los resultados de la búsqueda en varias páginas para facilitar su representación.

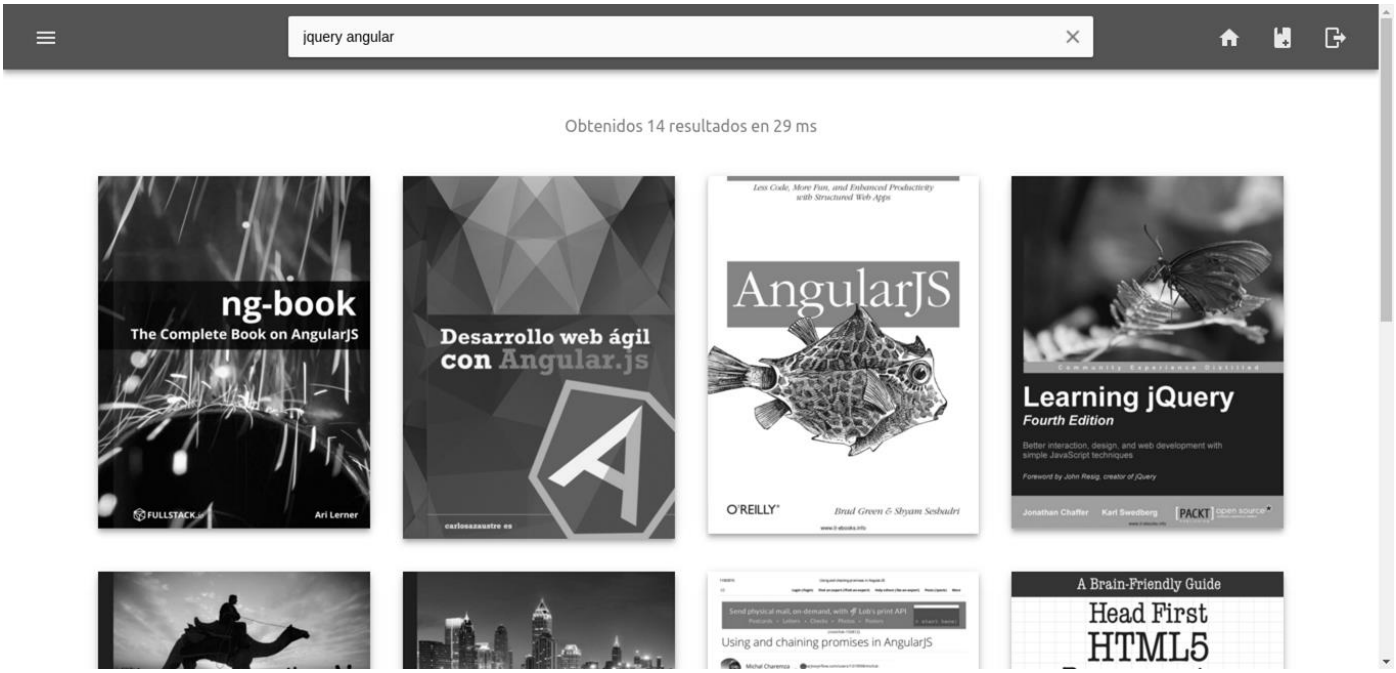

Figura 8: Resultados de búsqueda.

3.6 Pruebas

Las pruebas de *software* son la técnica más utilizada en la industria para garantizar la calidad del mismo y a menudo involucran más del cincuenta por ciento del esfuerzo total del ciclo de vida de desarrollo [54]. Al sistema se le aplicaron dos tipos de pruebas propuestas por la metodología *XP*: **pruebas de aceptación** y **pruebas unitarias**. Las primeras son diseñadas por los clientes para evaluar que al final de cada iteración se alcanzaron las funcionalidades requeridas; las segundas son diseñadas por los programadores para verificar que el código se ejecute correctamente. Posteriormente, fue sometido a **pruebas de rendimiento** para observar su comportamiento bajo demanda.

3.6.1 Pruebas de aceptación

Las pruebas de aceptación representan aquella fase del ciclo de vida de desarrollo de *software* en que el equipo de desarrollo y los usuarios tienen que garantizar que el sistema desarrollado se corresponda con los requerimientos definidos en la fase de análisis. Es fundamental que la calidad tanto del código como de

la documentación aportada al usuario sea alta y se corresponda con los parámetros adecuados para el *software* que concretamente se esté desarrollando [55].

Las pruebas de aceptación se realizan a partir de las Historias de Usuario, donde cada una se convierte en un caso de prueba en el cual el cliente especifica los puntos a probar. Además, una Historia de Usuario pudiera tener más de una prueba de aceptación, pues se deben realizar las necesarias para garantizar la calidad del producto final. A continuación, se muestra un caso de prueba de ejemplo para la funcionalidad Autenticar usuario. (Consultar el resto en el Anexo D. Pruebas de Aceptación)

Tabla 10: CP Autenticar usuario.

Caso de prueba de aceptación	
Código: HU1CP1	**Historia de usuario:** Autenticar usuario
Responsable de la prueba: José Jairán Breijo Rodríguez	
Descripción: Prueba para verificar el proceso de autenticación	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: 1. El usuario selecciona la funcionalidad Acceder. 2. El sistema muestra un formulario que espera los siguientes datos: • Nombre de usuario (cadena de texto no vacía, ejemplo: jjbreijo) • Contraseña (cadena de texto no vacía) 3. El usuario escribe los datos y los envía. 4. El sistema valida los datos. 5. Si los datos son válidos el sistema muestra la colección personal del usuario, de lo contrario muestra un mensaje con el texto: “Credenciales incorrectas”.	
Resultado esperado: El usuario accede a su colección personal.	
Evaluación de la prueba: Prueba satisfactoria.	

Una vez terminado este proceso el cliente quedó satisfecho, pues todos los casos de prueba se ejecutaron de forma satisfactoria.

3.6.2 Pruebas unitarias

Las pruebas unitarias son escritas por los programadores antes de comenzar la codificación. Su objetivo es aislar pequeñas e individuales porciones del código para verificar que no tengan errores. Con ello se logra mejorar la calidad, reducir el tiempo empleado en desarrollo y llevar el producto al mercado más rápido y sin errores [56].

Para garantizar la calidad del *software* en cuestión, el desarrollo de la aplicación se llevó a cabo utilizando una práctica llamada: Desarrollo guiado por pruebas o *TDD* (del inglés *Test-Driven Development*), que presenta el siguiente ciclo de vida:

1. Confeccionar una prueba para que el código falle y así detectar vulnerabilidades.
2. Escribir el código fuente para que pase dicha prueba.
3. Llevar a cabo la optimización y refactorización del mismo, mientras que la prueba no falle.
4. Repetir este proceso hasta que el proyecto esté completado.

Este flujo de trabajo permite un desarrollo incremental evitando tener que emplear tiempo en encontrar un error en particular, pues el desarrollador es capaz de ir directamente y cambiar la parte de la aplicación que esté presentando problemas. De esta forma es posible saber si la implementación de nuevas funcionalidades o cambios en las ya implementadas están afectando el comportamiento del sistema, solamente con ejecutar dichas pruebas [57].

La herramienta seleccionada para la ejecución de pruebas unitarias al sistema desarrollado es *unittest*[62], un marco de trabajo para pruebas unitarias inspirado en *JUnit*[63], que permite utilizar configuración compartida para cada caso de prueba, la agregación de pruebas en colecciones y la automatización de las mismas [58]. A continuación, se muestran las pruebas a las principales funcionalidades del sistema. (Consultar el resto en el Anexo E. Pruebas Unitarias).

[62] https://docs.python.org/3/library/unittest.html#module-unittest
[63] Conjunto de bibliotecas que son utilizadas para hacer pruebas unitarias de aplicaciones Java.

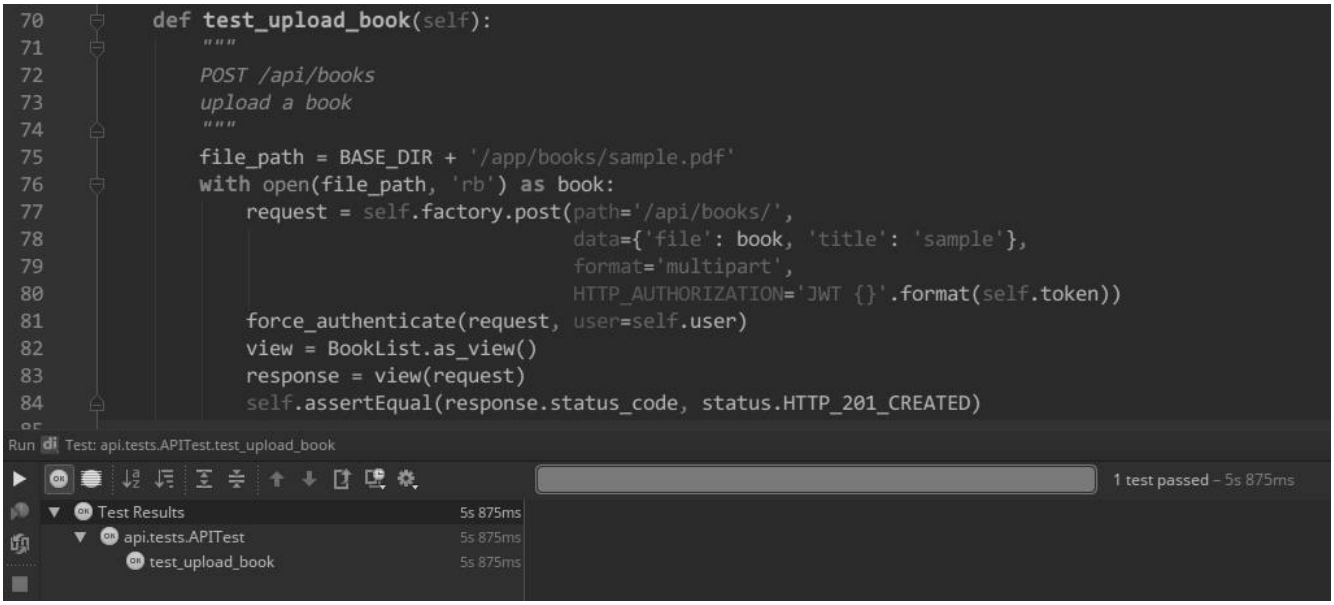

Figura 9: Prueba unitaria. Subir documento.

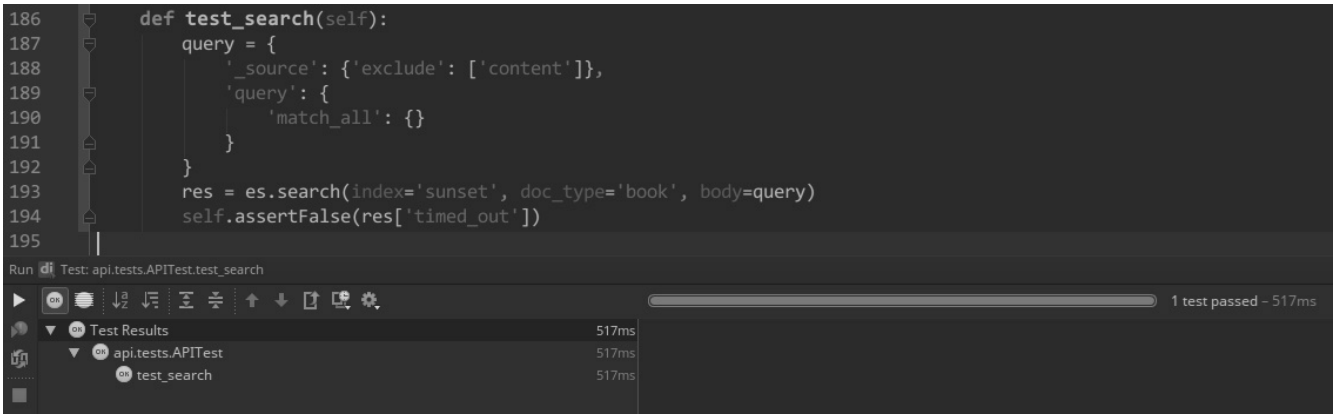

Figura 10: Prueba unitaria. Realizar búsqueda.

Las pruebas unitarias se realizaron utilizando la suite del Entorno de Desarrollo Integrado (*IDE,* por sus siglas en inglés) *PyCharm*, que internamente hace uso del módulo *unittest*. Esto permitió la automatización del proceso y una mejor retroalimentación visual del estado del mismo. Dichas pruebas estuvieron enfocadas en validar el correcto funcionamiento de las principales tareas del entorno colaborativo, dígase la subida de documentos y la realización de búsquedas; así como los procesos que involucraban la gestión

y el correcto funcionamiento del componente de árbol. Se aplicaron un total de 16 pruebas unitarias al entorno colaborativo de administración de documentación, obteniéndose resultados satisfactorios para cada una de ellas.

3.6.3 Pruebas de rendimiento

El entorno colaborativo desarrollado fue sometido a varias pruebas de rendimiento para medir su funcionamiento. Estas pruebas estuvieron enfocadas en el análisis del comportamiento de los tiempos de respuesta de las principales funcionalidades: la subida de un documento y la búsqueda de los mismos. Para ello se emplearon dos técnicas diferentes de indexación, para comprobar su impacto en dichos tiempos de respuesta:

- Indexación de **texto completo**: Se envía al servidor de búsqueda el texto extraído de los documentos sin ninguna modificación.

- Indexación de **texto procesado**: Se hace un pre-procesamiento del contenido de los libros aplicando la estrategia explicada en el capítulo anterior, donde el texto pasa por un proceso de filtrado que comprende los siguientes pasos: *tokenización*, eliminación de *stop-words* y *stemming***.**

Las pruebas se llevaron a cabo en un ordenador con un microprocesador *Intel Core* i3-2120 a 3.*3GHz* y 4 *Gb* de memoria *RAM*. Para probar el tiempo que el sistema empleaba en procesar un libro desde que el usuario lo añade a la plataforma, hasta que se encuentra listo para su gestión y consulta, se realizó la carga de 100 ejemplares en formato *PDF* de disímiles tamaños. A continuación, se muestra una comparativa del tiempo de carga, aplicándole al contenido de los mismos las técnicas de indexación mencionadas anteriormente:

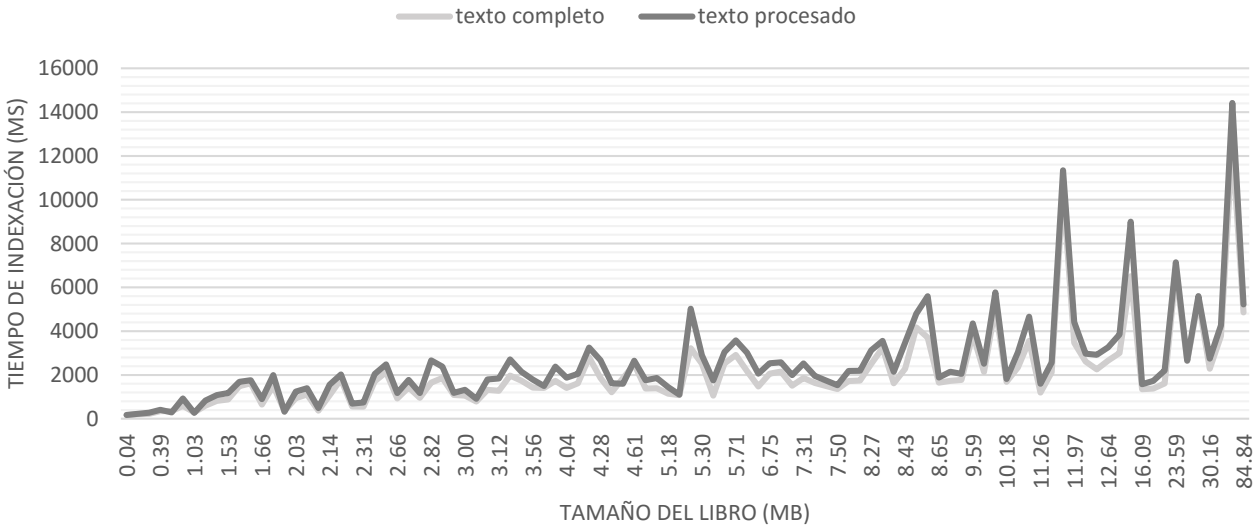

Figura 11: Comparación de tiempo de subida con diferentes estrategias de pre-procesado.

Los resultados de esta comparación arrojaron que la aplicación de técnicas de procesamiento de lenguaje natural al contenido de los libros, representó un aumento del tiempo empleado en el proceso de subida, con un tiempo promedio de 2089.03 milisegundos(ms) para la indexación de texto completo y 2546.30 ms utilizando la otra alternativa. El máximo tiempo empleado fue de 14418,43 ms al procesar un libro con un tamaño de aproximadamente 85 MB. Una vez terminada la indexación de los documentos en el servidor de búsqueda, el tamaño del índice destinado a guardar los documentos procesados con cada estrategia (espacio de búsqueda), quedó de la siguiente forma:

Tabla 11: Estadísticas del espacio de búsqueda

Estrategia	**texto completo**	**texto procesado**
Tamaño del índice[64] (Mb)	58	32
Tiempo destinado a la indexación[65] (ms)	6016	3781

[64] Se refiere a la estructura que almacena el servidor de búsqueda
[65] Tiempo total destinado a la creación de los documentos que forman parte del espacio de búsqueda.

A pesar de que la aplicación de la estrategia de pre-procesado produjo un aumento del tiempo total empleado en la subida de documentos, también trajo consigo una disminución de aproximadamente el 45% del tamaño del espacio de búsqueda. Además, se produjo una reducción de aproximadamente el 37% del tiempo total destinado a la indexación del texto que conforma dicho espacio de búsqueda.

Debido a que el entorno colaborativo es capaz de realizar búsquedas, estas deben estar relacionadas estrechamente con los criterios de rapidez y efectividad. La rapidez está dada por el tiempo que se empleó en obtener los resultados de una búsqueda y la efectividad por la semejanza que debe existir entre el criterio de provisto y dichos resultados. Estos dependerán de la relevancia que *Elasticsearch* le dé a cada documento con relación a la consulta enviada desde la capa de Presentación. Este proceso se desarrolla realizando una comparación de todos los datos que fueron indexados con la consulta enviada, utilizando un algoritmo de similitud conocido como *frecuencia del término / frecuencia inversa del documento (TF/IDF,* por sus siglas en inglés*)*, que toma los siguientes factores en consideración [45]:

- **Frecuencia del término (TF)**: Describe la frecuencia con que aparece un término en un documento. Mientras más común sea el término, mayor es su frecuencia. Por ejemplo: un documento que contenga cinco menciones del mismo término, es más relevante que uno que contenga solo una mención.

- **Frecuencia inversa de documento (IDF)**: Describe la frecuencia con que aparece cada término en el índice de *Elasticsearch*. Mientras más común sea, menos relevante es el término, o sea, los términos que aparecen en muchos documentos tienen un menor peso para las búsquedas que términos menos comunes.

Siempre que se ejecute una búsqueda *Elasticsearch* será el responsable de aplicar este algoritmo y comparar el criterio provisto por el usuario con los metadatos y el contenido procesado de todos los libros añadidos a la plataforma. Se determina así una puntuación para cada resultado, que será la medida de cuán relevante es un documento o que semejanza presenta con las consultas elaboradas por los usuarios. De esta forma *Elasticsearch* siempre ofrece resultados acertados y garantiza que las búsquedas sean efectivas.

Con el objetivo de verificar si la reducción del espacio de búsqueda que significó aplicar técnicas de procesamiento de lenguaje natural, produjo una disminución de los tiempos de respuesta del sistema, se

realizó la ejecución de 300 consultas con más de 8 términos en cada una sobre ambos índices, donde uno contiene el texto de los 100 libros sin procesar y el otro el texto resultante de emplear la estrategia mencionada anteriormente a todos los libros. A continuación, se muestran dos gráficas comparativas con los resultados de las búsquedas sobre ambos índices.

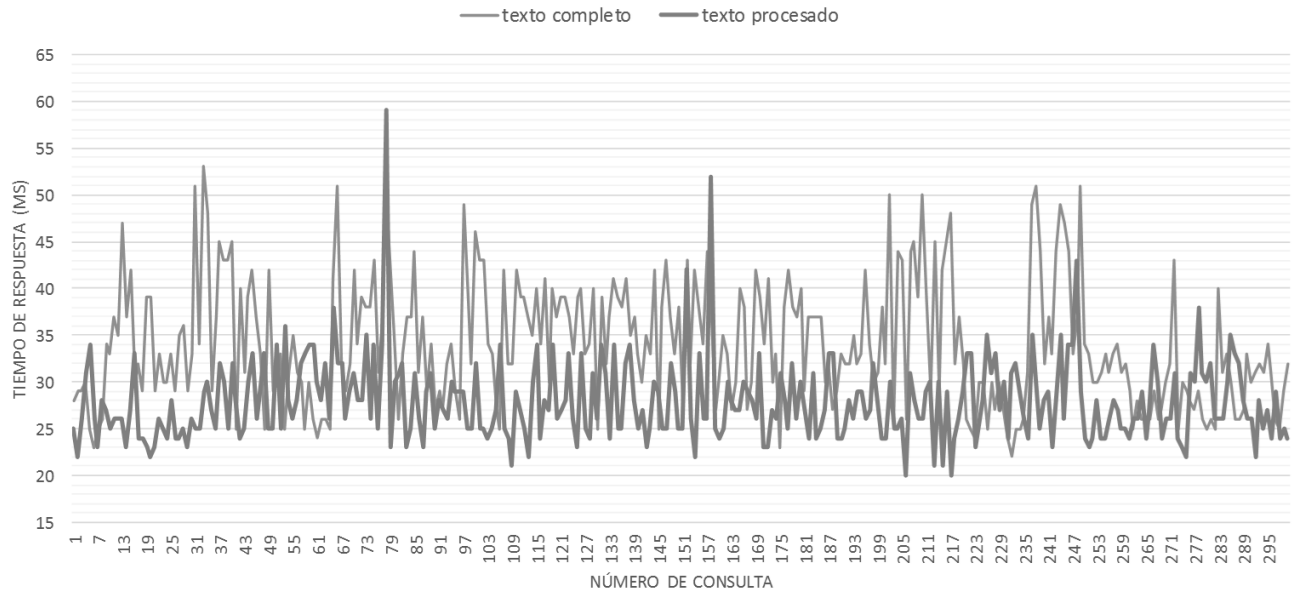

Figura 12: Comparativa de los tiempos de búsqueda sobre diferentes índices.

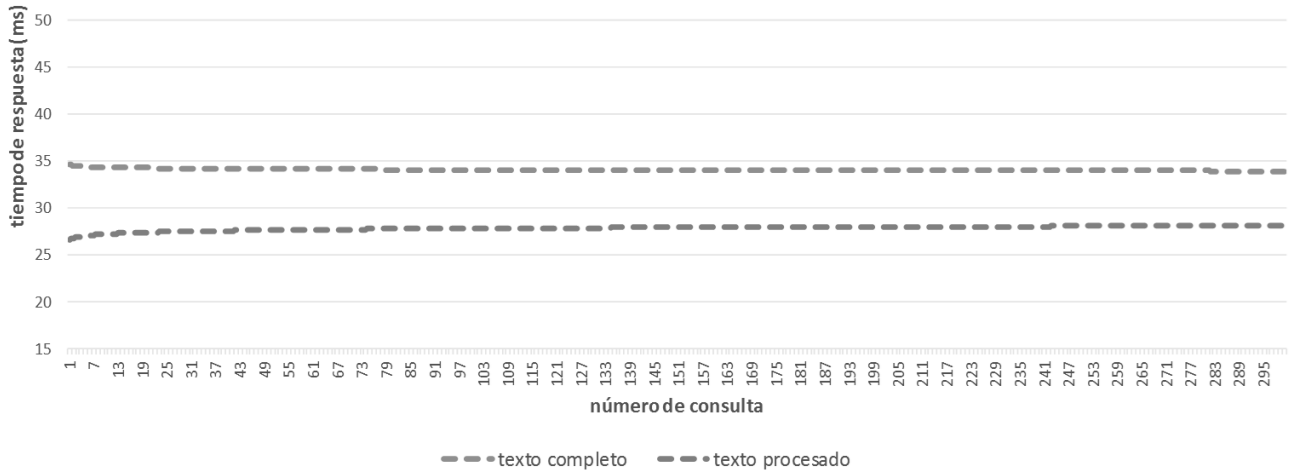

Figura 13: Línea de tendencia obtenida de la comparativa de los tiempos de búsqueda.

El análisis de esta comparativa arrojó que la aplicación de la estrategia de pre-procesado descrita en el capítulo anterior tuvo un impacto satisfactorio en la reducción de los tiempos de búsqueda, pues se logró una reducción en el tiempo empleado por consulta de aproximadamente el 18%. Ello trajo consigo un tiempo promedio de respuesta del sistema de **28 milisegundos**, en comparación con los resultados de buscar sobre el texto sin procesar, que produjo una media de 34 milisegundos. De esta forma queda demostrada la importancia de aplicar técnicas de procesamiento del lenguaje natural para la optimización del espacio de búsqueda.

3.7 Conclusiones parciales

A través del desglose de las historias de usuario en tareas de ingeniería, se mejoró el flujo de trabajo al indicar al programador específicamente las funcionalidades a desarrollar.

La creación de los diagramas de componentes y despliegue permitió conocer la separación lógica del código fuente, las relaciones existentes entre dichos componentes y la distribución física del sistema sobre una arquitectura de hardware.

El desarrollo de la aplicación guiado por pruebas permitió la obtención de las funcionalidades descritas en la fase de análisis del ciclo de vida del proyecto, lo que condujo a la satisfacción del cliente.

Los resultados de las pruebas de rendimiento, evidenciaron el impacto positivo de la aplicación de técnicas de procesamiento de lenguaje natural en la reducción de los tiempos de búsqueda de la aplicación.

Conclusiones

A través de la utilización del entorno colaborativo de administración de documentación digital desarrollado, se eliminó la necesidad de los usuarios de la universidad de almacenar su bibliografía de forma local, permitiendo su consumo desde cualquier dispositivo.

La utilización de *AngularJS* como marco de trabajo para el desarrollo de aplicaciones web del lado del cliente significó una reducción en los tiempos de carga de las interfaces de usuario; brindando la rapidez y experiencia de usuario de una aplicación de escritorio.

El empleo de un componente con estructura de árbol para la gestión de la colección personal produjo una significativa mejora de la experiencia de usuario, al poder simular la jerarquía de carpetas y archivos que implementan los sistemas operativos.

La combinación de una arquitectura n-capas y con la utilización de la arquitectura *REST* para el transporte de los datos, produjo un aumento en la flexibilidad del sistema haciéndolo altamente extensible y permitiendo su utilización como un servicio de almacenamiento y búsquedas de documentos.

El empleo por parte del servidor de búsqueda de las consultas difusas y de la técnica de similitud llamada *frecuencia del término / frecuencia inversa del documento,* para la búsqueda de semejanzas entre el total de documentos y las consultas realizadas, permitió la obtención de resultados acertados en el orden de los milisegundos.

La aplicación de la estrategia de pre-procesado que incluye la *tokenización*, la eliminación de *stop-words* y la aplicación de *stemming* a las palabras que conforman el contenido de los documentos, posibilitó la reducción del espacio de búsqueda en un 45%, significando esto un aumento en la escalabilidad del servidor con esta responsabilidad. Dicha reducción trajo consigo una disminución del 18% del tiempo de respuesta promedio por consulta, lográndose como tiempo promedio 28 milisegundos.

Recomendaciones

Luego de cumplido el objetivo general y analizados los resultados obtenidos se recomienda:

- Ejecutar de forma asincrónica[66] la subida de documentos para reducir el tiempo de espera en la interfaz web y así obtener una mejora en la experiencia de usuario.

- Implementar un componente *drag and drop*[67] que permita la subida a la plataforma de múltiples documentos de forma simultánea.

- Emplear técnicas de minería de texto para ofrecer sugerencias de libros que puedan ser del agrado de los usuarios, en relación con sus preferencias.

[66] Se refiere a la ejecución por separado de una tarea, sin bloquear el procesamiento por parte del servidor.
[67] Acción de seleccionar un elemento con el mouse y soltarlo en una localización diferente

Referencias bibliográficas

1. ZERMEÑO, M. G. L. *Bibliotecas digitales: recursos bibliográficos electrónicos en educación básica*. Comunicar: Revista Científica de Comunicación y Educación, vol. 20, no 39, p. 119-128. 2012.

2. RUPARELIA, N. B. *Software development lifecycle models*. ACM SIGSOFT Software Engineering Notes, vol. 35, no 3, p. 8-13. 2010.

3. GRANT, W. *BBC Mundo*. [En línea] 13 de octubre de 2014. Disponible en: http://www.bbc.com/mundo/noticias/2014/10/141013_tecnologia_cuba_internet_falta_wifi_lv

4. ADOBE. *Adobe Acrobat DC*. [En línea] 2015. Disponible en: https://acrobat.adobe.com/la/es/products/about-adobe-pdf.html.

5. UNIVERSITY, S. *Stanford University Libraries*. [En línea] 2015. Disponible en: https://library.stanford.edu/projects/google-books

6. HEYMAN, S. *Google Books: A Complex and Controversial Experiment.* [En línea] 28 de octubre de 2015. Disponible en: http://www.nytimes.com/2015/10/29/arts/international/google-books-a-complex-and-controversial-experiment.html?_r=2

7. CAVNAR, W. B., *et al. N-gram-based text categorization*. Ann Arbor MI, vol. 48113, no 2, p. 161-175. 1994.

8. LIN, Y., *et al. Syntactic annotations for the google books ngram corpus. En Proceedings of the ACL 2012 system demonstrations.* Association for Computational Linguistics. p. 169-174. 2012.

9. ORIN, A. *Lifehacker*. [En línea] 6 de noviembre de 2014. Disponible en: http://lifehacker.com/behind-the-app-the-story-of-scribd-1589303246

10. STONE, B. *The New York Times*. [En línea] 18 de mayo de 2009. Disponible en: http://www.nytimes.com/2009/05/18/technology/start-ups/18download.html?_r=0

11. LESOV, D. *Incorporating social media for personalization of ISSUU content*. 2013.

12. UPREDES. *LUPA*. [En línea] 2013. Disponible en: http://lupa.upr.edu.cu/acerca-de.html

13. REDUNIV. *Directorio de Artículos de Acceso Abierto*. [En línea] 2014. Disponible en: http://dima.mes.edu.cu/?q=hoja-de-ruta

14. KAPIDAKIS, S. *et al. Flexible metadata mapping using OAI-PMH. En Proceedings of the 8th ACM International Conference on Pervasive Technologies Related to Assistive Environments*. ACM, 2015. p. 86.

15. GUPTA, V.; LEHAL, G. S. *A survey of text mining techniques and applications*. Journal of emerging technologies in web intelligence, 2009, vol. 1, no 1, p. 60-76.

16. HOTHO, A.; NÜRNBERGER, A.; PAAß, G. *A Brief Survey of Text Mining*. En Ldv Forum. 2005. p. 19-62.

17. HEARST, M. A. *Untangling text data mining. En Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*. Association for Computational Linguistics, 1999. p. 3-10.

18. GHOSH, S.; ROY, S; BANDYOPADHYAY, S. *A tutorial review on Text Mining Algorithms*. International Journal of Advanced Research in Computer and Communication Engineering, 2012, vol. 1, no 4, p. 7.

19. JUSOH, S; AL-FAWAREH, H. M. *Natural language interface for online sales systems*. En Intelligent and Advanced Systems, 2007. ICIAS 2007. International Conference on. IEEE, 2007. p. 224-228.

20. MORIK, K.; SCHOLZ, M. *The miningmart approach to knowledge discovery in databases*. *En Intelligent technologies for information analysis*. Springer Berlin Heidelberg, 2004. p. 47-65.

21. PORTER, M. F. *An algorithm for suffix stripping*. Program, 1980, vol. 14, no 3, p. 130-137.

22. SHARMA, D. *Stemming algorithms: a comparative study and their analysis*. International Journal of Applied Information Systems, 2012, vol. 4, no 3, p. 7-12.

23. KROVETZ, R. *Viewing morphology as an inference process*. Artificial intelligence, 2000, vol. 118, no 1, p. 277-294.

24. CMS. *Selecting a development approach*. [En línea] marzo 27, 2008. Disponible en: http://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

25. CANÓS, J.; LETELIER, P.; PENADÉS, M. C. *Metodologías Ágiles en el desarrollo de Software.* Universidad Politécnica de Valencia, Valencia. 2003.

26. JEFFRIES, R., ANDERSON, A., HENDRICKSON, C. 2001. *Extreme Programming Installed*. Addison-Wesley. 2001.

27. BOOCH, G., *et al*. *El lenguaje unificado de modelado*. Addison-Wesley. 1999.

28. ROSSUM, G. *El tutorial de Python.* [ed.] Jr. Fred L. Drake. 2009.

29. GONZÁLEZ, R. *Python para todos.* [En línea] 2010. Disponible en: http://edge.launchpad.net/improve-python-spanish-doc/0.4/0.4.0/+download/Python%20para%20todos.pdf.

30. CONDORI, J. L. *Python-Django. Framework de desarrollo web para perfeccionistas basado en el Modelo MTV*. Revista de Información, Tecnología y Sociedad, 2012, p. 36.

31. LERNER, A. *The Complete Book on AngularJS.* 2013.

32. BASALO, A; ÁLVAREZ, M. *Desarrollo Web*. [En línea] 28 de agosto de 2014. Disponible en: http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html.

33. SAIKRISHNA, V; RASOOL, A.; KHARE, N. S*tring Matching and its Applications in Diversified Fields*. International Journal of Computer Science Issues, vol. 9, no 1, p. 219-226. 2012.

34. MOHAMMAD, A.; SALEH, O.; ABDEEN, R. A. *Occurrences algorithm for string searching based on brute-force algorithm*. Journal of Computer Science, vol. 2, no 1, p. 82-85. 2006.

35. MANDUMULA, K. K. *Knuth-Morris-Pratt*. 2011.

36. HASIB, S; MOTWANI, M.; SAXENA, A. *Importance of Aho-Corasick String Matching Algorithm in Real World Applications*. *International journal of computer science and information technologies*, vol. 4, no 3, p. 467-469. 2013.

37. YEH, J-F., *et al. Chinese word spelling correction based on n-gram ranked inverted index list.* En Sixth International Joint Conference on Natural Language Processing. 2013. p. 43.

38. CAMBAZOGLU, B. B., *et al. A term-based inverted index partitioning model for efficient distributed query processing.* ACM Transactions on the Web (TWEB), 2013, vol. 7, no 3, p. 15.

39. LIANG, W.; BO, F. *How to build a DNA search engine like Google?* ArXiv preprint arXiv: 1006.4114. 2010.

40. GORMLEY, C.; TONG, Z.. *Elasticsearch: The Definitive Guide.* O'Reilly Media, Inc., 2015.

41. MOHAN, V. *Elasticsearch Blueprints.* Birmingham B3 2PB, UK : Packt Publishing Ltd, 2015.

42. MARTÍNEZ, R. *Sobre PostgreSQL. www.postgresql.org.es.* [En línea] 2010. Disponible en: http://www.postgresql.org.es/sobre_postgresql.

43. OBE, R. O.; HSU, L. S. *PostgreSQL: Up and Running: A Practical Introduction to the Advanced Open Source Database.* O'Reilly Media, Inc., 2014.

44. JETBRAINS. *JetBrains*. [En línea] 2015. Disponible en: https://www.jetbrains.com/pycharm/.

45. MIKOWSKI, M. S.; JOSH C. P. *Single Page Web Applications.* B and W. 2013.

46. CHEN, S. *The concept of representational state transfer (rest)*. 2013.

47. QIAN, K. *Software architecture and design illuminated*. Jones & Bartlett Learning, 2010.

48. AMPATZOGLOU, A.; SOFIA C.; IOANNIS S. *Research state of the art on GoF design patterns: A mapping study*. Journal of Systems and Software 86.7. 2013.

49. BRANAS, R. *AngularJS Essentials*. Packt Publishing Ltd, 2014.

50. PERKINS, J. *Python 3 Text Processing with NLTK 3 Cookbook*. Packt Publishing Ltd, 2014.

51. BIRD, S.; KLEIN, E.; LOPER, E.. *Natural Language Processing with Python.* " O'Reilly Media, Inc.", 2009.

52. GAUCHAT, J. D. *El gran libro de HTML5, CSS3 y Javascript.* 2012.

53. JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *El proceso unificado de desarrollo de software*. Reading: Addison Wesley, 2000.

54. ESCALONA, M. J.; VOS, T.; GUTIÉRREZ, J. J. *Pruebas de software en la enseñanza universitaria de la informática: un título propio. Jornadas de Enseñanza de la Informática (18es: 2012: Ciudad Real)*, 2012.

55. GONZÁLEZ, J. F. P. *et al. Pruebas de aceptación orientadas al usuario: contexto ágil para un proyecto de gestión documental*. Ibersid: revista de sistemas de información y documentación, 2014, vol. 8, p. 73-80.

56. CLIFTON, M. *Unit Testing Succinctly.* s.l. Syncfusion Inc, 2013.

57. HARVEY, K. *Test-driven development with Django.* 2015.

58. PERCIVAL, H. *Test-Driven Development with Python*. O'Reilly Media, Inc., 2014.

Anexos

Anexo A. Historias de Usuario

Tabla 12: HU4 Adicionar documento a la colección.

	Historia de Usuario
Número: HU4	**Usuario:** Todos
Nombre de historia: Adicionar documento a la colección	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: José Jairán Breijo Rodríguez	
Descripción: Luego de una búsqueda el usuario selecciona un libro y activa la funcionalidad de adicionar a la colección. El documento se añadirá a un componente que permite categorizarlos con una estructura de árbol.	
Observaciones: El sistema debe notificar luego de añadir el documento a la colección	

Tabla 13: HU5 Eliminar documento de la colección.

	Historia de Usuario
Número: HU5	**Usuario:** Todos
Nombre de historia: Eliminar documento de la colección	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: José Jairán Breijo Rodríguez	
Descripción: El usuario selecciona el libro en su colección personal y a través del menú activa la funcionalidad de eliminar.	
Observaciones:	

Tabla 14: HU6 Compartir documentación.

	Historia de Usuario
Número: HU6	**Usuario:** Todos
Nombre de historia: Compartir documentación	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: José Jairán Breijo Rodríguez	
Descripción: Una vez buscado un documento el usuario activa el diálogo que muestra los detalles del mismo. A continuación selecciona la funcionalidad compartir y escribe el correo de la persona a la cual desea enviarle el documento.	
Observaciones: El sistema debe notificar al usuario si se envió correctamente.	

Tabla 15: HU7 Comentar documentación

	Historia de Usuario
Número: HU7	**Usuario:** Todos
Nombre de historia: Comentar documentación	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 3
Programador Responsable: José Jairán Breijo Rodríguez	
Descripción: Una vez buscado un documento el usuario activa el diálogo que muestra los detalles del mismo. A continuación, selecciona el apartado de los comentarios y escribe su opinión sobre el documento seleccionado.	
Observaciones:	

Anexo B. Tarjetas CRC

Tabla 16: Tarjeta CRC. Usuario.

Clase Usuario	
Responsabilidades	**Colaboradores**
Crear usuario	
Modificar usuario	
Eliminar usuario	
Obtener datos del usuario	

Tabla 17: Tarjeta CRC. Comentario.

Clase Comentario	
Responsabilidades	**Colaboradores**
Crear comentario	Clase Usuario, Clase Libro
Modificar comentario	
Eliminar comentario	
Obtener datos del comentario	

Tabla 18: Tarjeta CRC. Etiqueta.

Clase Etiqueta	
Responsabilidades	**Colaboradores**
Crear etiqueta	
Indexar etiqueta	

Anexo C. Tareas de Ingeniería

Tabla 19: Tareas de ingeniería. Iteración 2.

Iteración 2	
Historia de Usuario	**Tareas**
Realizar búsquedas	1. Definir interfaz del buscador. 2. Definir interfaz para mostrar los resultados de la búsqueda. 3. Mostrar los documentos más recientes si no se ha enviado aún una consulta. 4. Elaborar consulta a enviar al servidor de búsqueda con el criterio provisto por el usuario. 5. Mostrar tiempo transcurrido y resultados de la búsqueda.

Tabla 20: Tareas de ingeniería. Iteración 3.

Iteración 3	
Historia de Usuario	**Tareas**
Adicionar documento a la colección	1. Obtener y enviar identificador del documento a la capa de Procesamiento. 2. Buscar documento en la base de datos 3. Añadir documento a la colección del usuario. 4. Modificar componente de árbol adicionando el nuevo documento. 5. Notificar al usuario una vez termine este proceso.

Eliminar documento de la colección	1. Obtener y enviar identificador del documento a la capa de Procesamiento. 2. Buscar documento en la base de datos 3. Eliminar documento de la colección del usuario. 4. Eliminar del componente de árbol el documento seleccionado.
Compartir documentación	1. Definir interfaz para compartir un documento. 2. Validar correo de la persona a quien se envía dicho documento. 3. Enviar un correo al destinatario con el título del documento y donde lo puede consultar. 4. Notificar al usuario una vez termine este proceso
Comentar documentación	1. Definir interfaz para añadir un comentario. 2. Enviar comentario a la capa de Procesamiento. 3. Guardar el comentario en la base de datos 4. Actualizar el listado de comentarios del libro seleccionado

Anexo D. Pruebas de Aceptación

Tabla 21: CP Subir documento.

Caso de prueba de aceptación	
Código: HU2CP1	**Historia de usuario:** Subir documento
Responsable de la prueba: José Jairán Breijo Rodríguez	
Descripción: Prueba para verificar el proceso de subida de documentos	
Condiciones de ejecución: El usuario tiene que estar autenticado	
Entrada/Pasos de ejecución: 1. El usuario selecciona la funcionalidad Subir documento 2. El sistema muestra un formulario que espera los siguientes datos: • Archivo (documento en formato *PDF*) • Título del documento (cadena de texto no vacía) • Autores • Etiquetas • Idioma 3. El usuario escribe los datos y los envía 4. El sistema valida los datos, procesa el documento, lo indexa para su búsqueda y notifica al usuario.	
Resultado esperado: El documento es añadido a la colección personal del usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Tabla 22: CP Realizar búsquedas.

Caso de prueba de aceptación	
Código: HU3CP1	**Historia de usuario:** Realizar búsquedas

Responsable de la prueba: José Jairán Breijo Rodríguez
Descripción: Prueba para verificar el proceso búsqueda
Condiciones de ejecución:
Entrada/Pasos de ejecución: 1. El usuario selecciona la funcionalidad Buscar 2. El sistema muestra un formulario que espera los siguientes datos: • Criterio de búsqueda (cadena de texto no vacía) 3. El usuario escribe la consulta y la envía 4. El sistema busca documentos que concuerden con dicho criterio
Resultado esperado: Se obtiene un listado con los documentos semejantes al criterio de búsqueda ordenados de forma descendente por su relevancia
Evaluación de la prueba: Prueba satisfactoria

Tabla 23: CP Adicionar documento a la colección.

Caso de prueba de aceptación	
Código: HU4CP1	**Historia de usuario:** Adicionar documento a la colección
Responsable de la prueba: José Jairán Breijo Rodríguez	
Descripción: Prueba para verificar la adición de un documento a la colección personal	
Condiciones de ejecución: El usuario tiene que estar autenticado	
Entrada/Pasos de ejecución: 1. El usuario busca el documento de su preferencia 2. Activa la vista de detalle del documento 3. Activa la funcionalidad de Añadir a la colección	
Resultado esperado: El documento es añadido a la colección personal del usuario.	
Evaluación de la prueba: Prueba satisfactoria	

Anexo E. Pruebas Unitarias

```python
def test_add_to_collection(self):
    """
    POST /api/add-to-collection/
    add to the collection a book uploaded by the current user
    """
    file_path = BASE_DIR + '/app/books/sample.pdf'
    with open(file_path, 'rb') as book:
        request = self.factory.post(path='/api/books/',
                                    data={'file': book, 'title': 'sample'},
                                    format='multipart',
                                    HTTP_AUTHORIZATION='JWT {}'.format(self.token))
        force_authenticate(request, user=self.user)
        view = BookList.as_view()
        response = view(request)

    book_id = response.data['id']

    request = self.factory.post(path='/api/add-to-collection/',
                                data={'id': book_id, 'fid': None, 'tags': None},
                                HTTP_AUTHORIZATION='JWT {}'.format(self.token))

    force_authenticate(request, user=self.user)
    view = AddToCollection.as_view()
    response = view(request)
    self.assertEqual(response.status_code, status.HTTP_200_OK)
```

Run Test: api.tests.JSTreeTestCase.test_add_to_collection

1 test passed - 4s 349ms

Test Results 4s 349ms

api.tests.JSTreeTestCase 4s 349ms

test_add_to_collection 4s 349ms

Figura 14: Prueba unitaria. Adicionar libro a la colección.

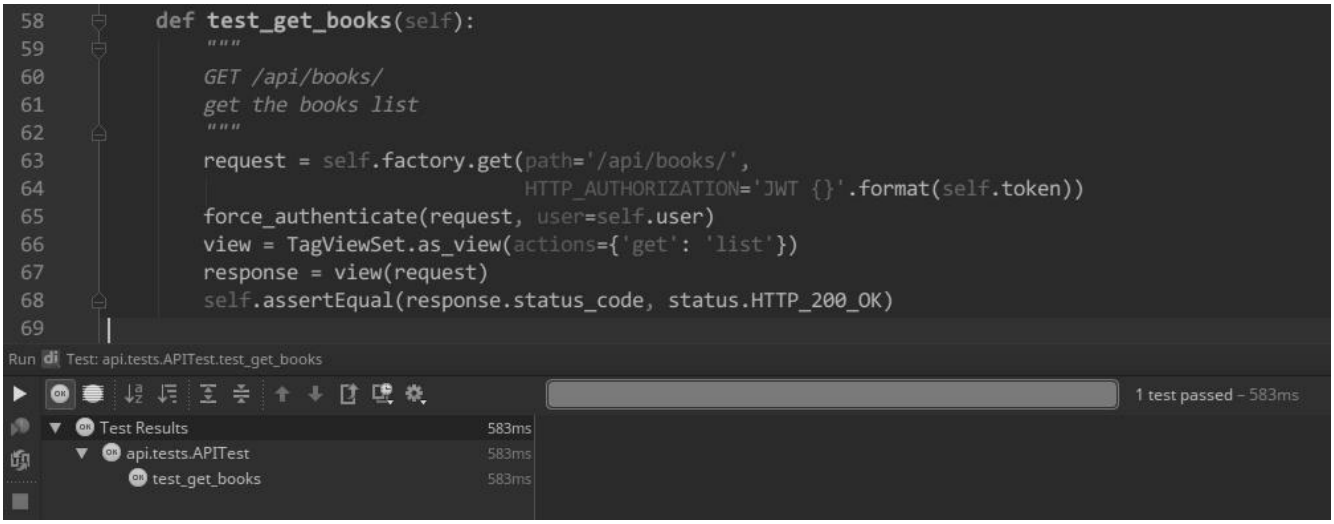

Figura 15: Prueba unitaria. Obtener listado de libros.

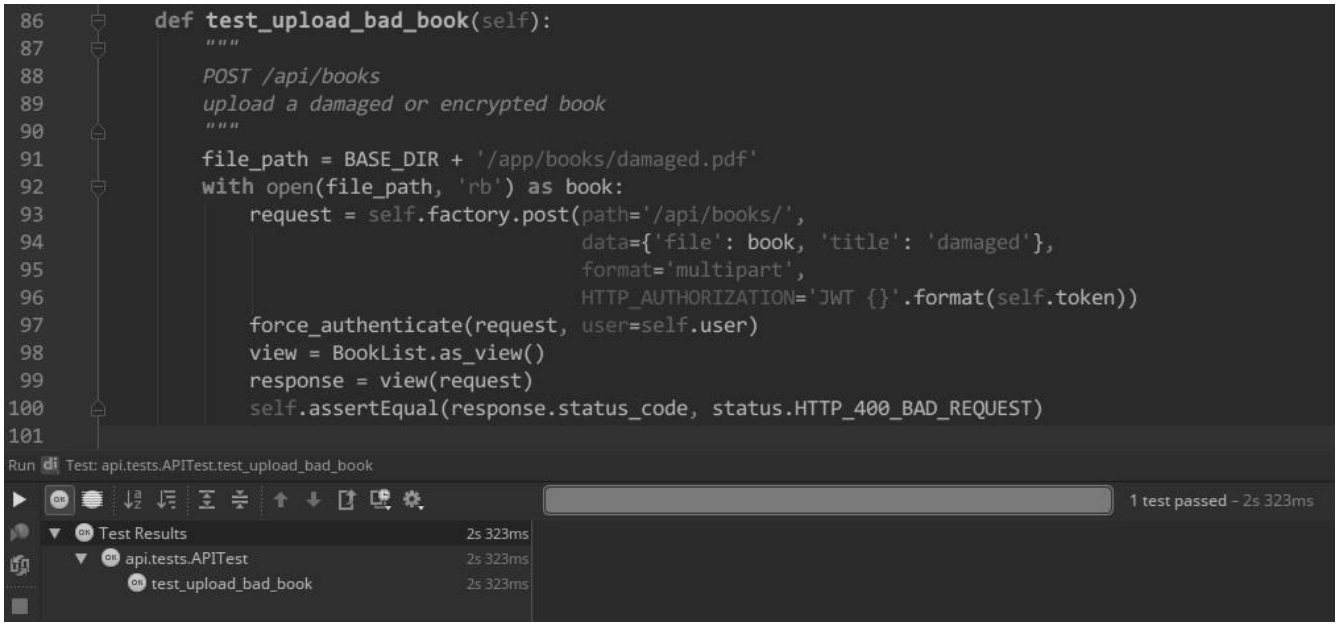

Figura 16: Prueba unitaria. Subir documento dañado.