Universidad de las Ciencias Informáticas

Facultad 1

Título: Capa de servicios web para el categorizador de texto del Motor de Clasificación Inteligente de Contenido

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores:

Patricia Tenorio Esclarazán

Amed Alcides Cortina Márquez

Tutores:

Ing. Paúl Rodríguez Leyva

Ing. Yurisleidy Hernández Moya

Junio de 2014
La Habana, Cuba

Declaración de autoría

Por este medio declaramos ser los únicos autores del presente trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo. Autorizamos a dicho centro para que haga el uso que estime pertinente con este trabajo.

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

Patricia Tenorio Esclarazán

_____
Firma del Autor

Amed Alcides Cortina Márquez

_____
Firma del Autor

Ing. Paul Rodríguez Leyva

_____
Firma del Tutor

Ing. Yurisleidy Hernández Moya

_____
Firma del Tutor

Agradecimientos

A todos los profesores que de una u otra forma contribuyeron a nuestra formación como ingenieros. A los tutores, por su paciencia y dedicación. A los miembros del tribunal y al oponente que influyeron en el perfeccionamiento de esta investigación.

Patricia

Quiero agradecer en primer lugar a mi familia por el apoyo y la comprensión que me han brindado a lo largo de esta carrera. Agradezco a mi abuelo por ser el mejor ejemplo que podría soñar tener, a mi mamá por siempre estar ahí para mí y por haber hecho el papel de padre y madre de la forma que lo ha hecho hasta hoy. A mi hermana por ser incondicional y sacrificarse tanto por mí. A mi novio por brindarme paz, seguridad y amor. A Raciel por contribuir en mi formación y ser ejemplo de perseverancia. A mi compañero de tesis Amed por haber sido el mejor de los compañeros. A mis compañeros de aula y amigos que siempre han estado presente en cada uno de mis éxitos y errores, de una forma muy especial agradecer a Eli y Elaimy, mis amigas de siempre que contribuyeron al resultado de mi trabajo. A toda mi familia, vecinos y amigos…Muchas gracias.

Amed

Agradezco a Dios por permitirme llegar hasta aquí y por estar siempre conmigo. A mis padres porque lo han dado todo por mí para poder lograr mis metas. A mis abuelos por ser especiales en mi formación como persona. A toda mi familia que siempre ha estado pendiente de mí y por su ayuda incondicional. A mi novia por ser mi ayuda idónea y por ser el complemento especial de mi vida. A mi compañera de tesis Patricia por ser imprescindible para el desarrollo de esta investigación. A todos mis compañeros de aula por formar un gran equipo para lograr el éxito en nuestra carrera, en especial a Julio César por su gran ayuda en esta tesis. A todos mis amigos de la universidad Luis Alberto, Richard, Ariel, Pedro, nunca los olvidaré. A todos mis más sinceros agradecimientos.

Dedicatoria

Patricia

Dedico el resultado de este trabajo a la memoria de mi papá Roberto Tenorio Aguilera quien sé que ha de estar muy orgulloso de mí…

Amed

A Jesús mi amigo fiel.

A mis padres queridos Dania y Alcides por ser la fuerza que mueve mi vida.

A mis familiares por su apoyo en todo momento.

A mi novia y futura esposa Dayliana por amarme como soy y ser la persona con la que voy a compartir el resto de mi vida.

A mis amigos por ayudarme durante estos cinco años a llegar a ser quien soy.

Resumen

El Centro de Ideoinformática (CIDI) de la Universidad de Ciencias Informáticas desarrolló una herramienta llamada Motor de Clasificación Inteligente de Contenido (MOCIC) con el objetivo de clasificar contenidos de documentos, páginas y sitios *web* a partir de sus imágenes, enlaces y textos. El proceso de categorización de texto en MOCIC se hace difícil al tener que conocer los comandos a utilizar para realizar las funcionalidades que brinda, además estas no se pueden utilizar desde aplicaciones desarrolladas en lenguajes de programación diferentes a Python. El objetivo general del trabajo es desarrollar una capa de servicios *web* para utilizar las funcionalidades del categorizador de texto de MOCIC desde otras aplicaciones. Entre las tecnologías y herramientas utilizadas para desarrollarla se encuentran: la metodología OpenUP, el lenguaje de modelado UML, lenguaje de programación Python, JSON como formato para el intercambio de información entre los servicios *web*, Django como *framework* y PostgreSQL como gestor de base de datos. El estudio del estado del arte permitió desarrollar una capa de servicios basada en las buenas prácticas empleadas por los sistemas estudiados y enfocada hacia la solución del problema identificado. Como resultado del trabajo realizado se logró el diseño e implementación de una capa de servicios *web* que permite la comunicación entre aplicaciones y el categorizador de texto de MOCIC, utilizando las funcionalidades del categorizador, y brindándolas como servicios *web*. La realización de pruebas permitió validar el correcto funcionamiento de la capa de servicios desarrollada simulando ambientes reales en diferentes entornos de ejecución.

Palabras Clave: capa de servicios, categorización, motor de clasificación, python.

Índice

Índice de figuras

Índice de figuras

Índice de tablas

Índice de tablas

Introducción

Desde el surgimiento de las computadoras, los programas informáticos han tenido un acelerado desarrollo, han sido utilizados para automatizar procesos de la actividad humana diaria. Aparejado a esto, surgió la infraestructura *Internet*, que trajo consigo una revolución en el mundo de la informática y las comunicaciones, esparciéndose por el mundo para crear la moderna red mundial de computadoras que hoy se conoce. A medida que la *World Wide Web (WWW)* creció como un medio potente de ofrecer y obtener información, se crearon los buscadores y los directorios *web* para localizar sitios *web* y permitir a las personas encontrar artículos de su preferencia. Los principales buscadores, que revolucionaron *Internet* fueron *Google* y *Yahoo*; que son empresas multinacionales especializadas en productos y servicios relacionados con *Internet*, *software*, dispositivos electrónicos y otras tecnologías de la información. Una vez creados los buscadores y directorios *web* era necesario viabilizar todos esos beneficios que traía la *web*, así como lograr una interoperabilidad entre ellos. Esta interoperabilidad se logró a través de los servicios *web*[1].

Cuba, a pesar de ser un país bloqueado económica, financiera y comercialmente por más de 50 años, ha tratado de reducir las consecuencias que provoca el uso limitado de *Internet* y la tecnología, para ello destina grandes recursos a la informatización de los sectores principales de la sociedad, como son la salud y la educación. Esto se evidencia con la creación de importantes proyectos, como son el surgimiento de varias universidades con el objetivo de formar profesionales en diversas áreas del conocimiento. Bajo este principio se creó la Universidad de las Ciencias Informáticas (UCI) con acceso a los últimos avances tecnológicos de la información, para especializar y formar profesionales en este campo.

El acceso a la tecnología y a la información de *Internet* en la UCI, se realiza mediante la navegación en la *web*, espacio que actualmente contiene un gran número de páginas con elevados contenidos pornográficos, subversivos y altamente violentos, considerados inadecuados por la sociedad[2]. Con el propósito de categorizar este tipo de contenidos se hizo necesaria la creación de un categorizador de texto que clasificara documentos y los contenidos de las páginas y sitios *web* a los que se acceden para que posteriormente los administradores de redes en la UCI tomen decisiones acerca de denegar o no estas páginas. En el Centro de Ideoinformática (CIDI), se desarrolló una herramienta llamada MOCIC (Motor de Clasificación Inteligente de Contenido) con el objetivo de categorizar documentos utilizando sus imágenes, enlaces y textos.

Esta herramienta brinda una solución a algunos de los problemas que ocurren cuando se busca en páginas *web* a las que nunca se ha accedido, como son, la referencia directa a sitios con contenidos

nocivos y la aparición de imágenes y videos pornográficos, El categorizador de texto de MOCIC permite categorizarlas, lo cual contribuye a saber si son inadecuadas o no. El categorizador de texto de MOCIC cuenta con varias funcionalidades desarrolladas en Python que permiten: extraer el texto plano de documentos en varios formatos, el preprocesamiento de texto plano, la detección de idioma y de codificación de caracteres, representar documentos en un modelo de espacio vectorial esparcido, el entrenamiento y la evaluación de categorizadores así como categorizar documentos.

Todas las operaciones que se realizan en el categorizador de texto se efectúan a través de códigos introducidos en una consola, esto trae consigo que los usuarios que no tengan un conocimiento elevado de los comandos a introducir para realizar los diferentes procesos del categorizador, tengan dificultades para interactuar con el subsistema, esto demanda la necesidad de tener una interfaz fácil de entender. La biblioteca de MOCIC *Textmining* , la cual permite la categorización de texto y contiene los métodos para categorizar los contenidos, está desarrollada en Python, por lo que sus funcionalidades no puede ser utilizadas desde aplicaciones desarrolladas en lenguajes de programación diferentes, esto hace necesario crear una solución para utilizar todas las funcionalidades y componentes del subsistema categorizador de texto ya implementadas, estas funcionalidades ayudan en la especialización de los componentes, así como en la distribución de las aplicaciones e interoperabilidad de las soluciones desarrolladas. A partir de lo antes expuesto se plantea el siguiente **problema de investigación**: ¿cómo garantizar el uso del subsistema de categorización de texto de MOCIC desde aplicaciones desarrolladas en lenguajes de programación diferentes a Python?

Para darle solución a este problema se define como **objeto de estudio**: los servicios *web*. El **campo de acción** está enmarcado en: los servicios *web* en el proceso de categorización de texto.

El **objetivo general** de la investigación es: desarrollar una capa de servicios *web* para utilizar las funcionalidades del categorizador de texto de MOCIC.

De donde se derivan los siguientes **objetivos específicos**:

- Construir el marco teórico y el estado del arte de la creación de servicios *web*.
- Diseñar los servicios *web* del categorizador de texto de MOCIC.
- Implementar los servicios *web* para el categorizador de texto de MOCIC.
- Implementar la interfaz de administración para la capa de servicios *web*.
- Validar la capa de servicios *web* del categorizador de texto de MOCIC.

Una vez planteado el objetivo general, así como los específicos, se identifica la siguiente **idea a defender:**

El desarrollo de una capa de servicios *web* posibilitará utilizar las funcionalidades del categorizador de texto de MOCIC, permitiendo su uso desde aplicaciones desarrolladas en lenguajes de programación diferentes a Python.

Para dar cumplimiento a los objetivos específicos se trazan las **tareas de investigación**:

- Análisis de sistemas que brinden servicios de categorización mediante servicios *web*.
- Definición de las tecnologías a utilizar en la implementación de los servicios *web*.
- Realización del análisis y diseño de la capa de servicios *web*.
- Implementación de la capa de servicios *web*.
- Implementación de la interfaz de administración para la capa de servicios *web*.
- Realización de pruebas funcionales a la capa de servicios *web*.
- Realización de pruebas funcionales a la interfaz de administración.

Para poder cumplir con las distintas tareas investigativas se emplearon los siguientes **métodos de investigación**:

Métodos Teóricos:

Histórico-Lógico: se utiliza en la fundamentación teórica, para conocer la evolución de los servicios *web* en la categorización de texto y las tendencias más relevantes en la actualidad así como en el estudio de las tecnologías[3].

Analítico-Sintético: se utiliza para el procesamiento de información y elaboración de conclusiones. Este método sirvió para analizar y comprender la teoría y documentación relacionada con el tema de investigación, permitiendo así extraer los elementos más importantes relacionados con el objeto de estudio[4].

Modelación: se emplea en la modelación de diagramas para representar el proceso de desarrollo mediante el lenguaje de modelado UML para reflejar la estructura, relaciones internas y características de la solución, de manera que se propiciará un mejor entendimiento de esta.

Métodos Empíricos:

Entrevista: ayuda a obtener información sobre el proceso de categorización de texto en servicios *web*, comprender y precisar el problema a resolver, así como a validar la propuesta que se presentará al cliente.[5]

Con la realización de la presente investigación se pretende obtener el siguiente **resultado**:

- Una capa de servicios que permita la utilización del categorizador de texto de MOCIC desde aplicaciones realizadas en diferentes lenguajes de programación.

Introducción

El presente trabajo, está estructurado en 3 capítulos, distribuidos de la siguiente forma:

Capítulo 1: Fundamentación teórica

Se abordan los conceptos fundamentales que permiten entender los servicios *web* en procesos de categorización de texto. Se caracterizan las aplicaciones existentes en el mundo que responden al problema a resolver y por qué no son factibles. Se definen las tecnologías a utilizar en el desarrollo de la investigación.

Capítulo 2: Diseño de la solución

Recoge la descripción y análisis de la solución propuesta, la caracterización de los servicios *web* a desarrollar, incluyendo los requerimientos. Se especifican los patrones usados, tanto de diseño como arquitectónicos.

Capítulo 3: Implementación y prueba

Se describen los estándares de código fundamentales usados en el desarrollo de la arquitectura de la capa de servicios *web*, se define y caracteriza el proceso de validación de los datos, el tratamiento de errores y la seguridad.

Capítulo 1: fundamentación teórica

1.1 Introducción

El desarrollo de las aplicaciones de *software* en la actualidad es de suma importancia para las organizaciones debido a que la mayoría de sus procesos y actividades dependen de sus sistemas de información. El constante cambio del negocio en las organizaciones aumenta la complejidad de sus procesos, imposibilitando su gestión por las aplicaciones existentes de forma individual, lo cual potencia la creación de tecnologías que satisfagan las nuevas necesidades. Una de estas son los servicios *web* que permiten la comunicación entre diversas aplicaciones y la utilización de sus funcionalidades.

En el presente capítulo se exponen los principales conceptos relacionados con los servicios *web* en procesos de categorización de textos. Se caracterizan servicios *web* existentes en el mundo que permiten categorizar textos. Además se describen la metodología de desarrollo de *software*, tecnologías y herramientas necesarias para la implementación de una capa de servicios *web*.

1.2 La *Web*

El medio donde los servicios *web* se despliegan se denomina la *web*, integrada por elementos como:

HTML *(HyperText Markup Language):* HTML es la lengua franca de la publicación de hipertexto en la *World Wide Web*. Es un formato no propietario basado en SGML (Estándar de Lenguaje de Marcado Generalizado)[6].

HTTP *(Hypertext Transfer Protocol):* como protocolo de comunicación entre los ordenadores de la *web*, encargado de la transferencia de las páginas *web* y demás recursos[7][8].

URL *(Uniform Resource Locator):* como medio de localización (direccionamiento) de los distintos recursos de la *web*[7][9].

La *web* se basa en dos puntales fundamentales: el protocolo HTTP y el lenguaje HTML. Uno permite una implementación simple y sencilla de un sistema de comunicaciones que permite enviar cualquier tipo de ficheros de una forma fácil, simplificando el funcionamiento del servidor y permitiendo que servidores poco potentes atiendan miles de peticiones y reduzcan los costes de despliegue. El otro proporciona un mecanismo de composición de páginas enlazadas simple y fácil, altamente eficiente y de uso muy simple[10].

1.2.1 Los servicios *web*

Son el conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la *web*. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer servicios. Estos proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan

entre sí para presentar información dinámica al usuario[11]. Para la comunicación entre los servicios *web* es necesario un lenguaje que permita el intercambio de la información, entre estos lenguajes se pueden mencionar XML *(Extensible Markup Language)* y JSON *(Java Script Object Notation)* como los más utilizados. Los servicios *web* pueden ser desarrollados en una gran variedad de lenguajes para ser implementados sobre muchos tipos de redes de computadoras. El éxito de la interoperabilidad se consigue gracias a la adopción de protocolos y estándares abiertos.

Existen un conjunto de servicios y protocolos usados para permitir la interacción entre los servicios *web*. Este conjunto está conformado esencialmente de cuatro subconjuntos[1]:

- Servicio de transporte
- Servicio de mensajería
- Descripción del servicio
- Descubrimiento de servicios

1.2.2 Servicio de transporte

Es el encargado del transporte de los mensajes entre aplicaciones sobre la red. Incluye varios protocolos del nivel de aplicación, algunos de estos son:

HTTP: define la sintaxis y la semántica utilizada para la arquitectura *web*. En el contexto de los servicios *web* es utilizado para la transferencia de las transacciones XML a través de la red utilizando los mismos principios del HTML *(Hyper Text Markup Languaje)*[12].

FTP *(File Transfer Protocol):* se encarga de los servicios de transmisión de archivos a través de redes soportadas sobre TCP. En el ámbito de los servicios *web* el FTP permite realizar modificaciones en equipos remotos evitando el uso de permisos sobre los archivos en la máquina cliente en sistemas operativos diferentes a *Windows*[13].

SMTP *(Simple Mail Transfer Protocol):* es un estándar de la capa de aplicación ampliamente utilizado para el envío de mensajes de correo electrónico a través de *Internet*[14].

1.2.3 Servicio de mensajería

Se encarga de la codificación de los mensajes a un lenguaje estándar, lo que posibilita ser interpretado en cualquiera de los nodos de la red. Los componentes más utilizados en este conjunto son los siguientes:

REST *(Representational State Transfer*): es un conjunto de principios para el diseño de redes, que es utilizado comúnmente para definir una interfaz de transmisión sobre HTTP sin una capa adicional como hace SOAP. Está basado en un conjunto de estándares tales como HTML, URL, XML, GIF *(Graphics*

Interchange Format), JPG *(Joint Photographic Experts Group)* y tipos MIME (*Multipurpose Internet Mail Extensions*)[1].

Los principios de REST son:

- Generalidad de Interfaces: gracias al protocolo HTTP, cualquier cliente puede interactuar con cualquier servidor HTTP sin ninguna configuración especial. Esto no es del todo cierto para otras alternativas, como SOAP para los servicios *web*.
- Puesta en funcionamiento independiente: garantiza que los servidores antiguos sean capaces de entenderse con clientes actuales y viceversa. Diseñar un protocolo que permita este tipo de características resulta muy complicado mientras que HTTP permite la extensibilidad mediante las cabeceras, haciendo uso de las URIs (*Uniform Resource Identifier*), a través de la habilidad para crear nuevos métodos y tipos de contenido.
- Compatibilidad con componentes intermedios: permite reducir la latencia de interacción, reforzar la seguridad y encapsular otros sistemas[15].

RPC *(Remote Procedure Calls): e*s una tecnología de *software* que permite ejecutar una rutina en un equipo o segmento de red de manera remota. Es un paradigma popular para la implementación de sistemas distribuidos bajo arquitecturas cliente servidor[16].

XML-RPC: es un protocolo de llamada remota que utiliza XML como lenguaje de codificación y HTTP como mecanismo de transporte. Es un protocolo sencillo ya que solo define algunos tipos de datos y comandos[17].

Las tecnologías basadas en modelos RPC son más adecuadas para entornos aislados, es decir, entornos que se conocen perfectamente. La evolución en este tipo de sistemas es sencilla, se modifica cada usuario para que cumpla con los nuevos requisitos pero cuando el número de usuarios es muy grande estos modelos no son adecuados ya que cambiar su interfaz resulta complicado[15].

SOAP *(Simple Object Access Protocol):* es un protocolo de la capa de aplicación para el intercambio de mensajes basados en XML sobre redes de computadoras. Permite utilizar lenguajes de alto nivel para llamar e implementar el servicio *web*. El principal beneficio de SOAP recae en ser fuertemente acoplado, lo que permite poder ser testado y depurado antes de poner en marcha la aplicación. Es un paradigma de una sola vía pero con la ayuda de las aplicaciones se puede llegar a crear patrones más complejos. Básicamente está constituido por:

- Un marco que describe el contenido del mensaje e instrucciones de proceso.
- Un conjunto de reglas para representar los tipos de datos definidos.

- Convenciones para representar llamadas a procedimientos remotos y respuestas[1][18].

A continuación se muestra una tabla con algunas características, ventajas y desventajas de los protocolos SOAP y REST[15].

	SOAP	REST
Características	Las operaciones se definen en los mensajes. Una dirección única para cada instancia del proceso. Cada objeto soporta las operaciones estándares definidas. Componentes débilmente acoplados.	Las operaciones son definidas como puertos WSDL (Lenguaje de descripción de servicios *web*). Dirección única para todas las operaciones. Múltiple instancias del proceso comparten la misma operación. Componentes fuertemente acoplados.
Ventajas declaradas	Bajo consumo de recursos. Las instancias del proceso son creadas explícitamente. El cliente no necesita información de enrutamiento a partir de la URI inicial. Los clientes pueden tener una interfaz “listener” (escuchadora) genérica para las notificaciones. Generalmente fácil de construir y adoptar.	Fácil (generalmente) de utilizar. La depuración es posible. Las operaciones complejas pueden ser escondidas detrás de una fachada. Envolver APIs existentes es sencillo. Incrementa la privacidad. Herramientas de desarrollo.
Desventajas	Gran número de objetos. Manejar el espacio de nombres (URIs) puede ser engorroso. La descripción sintáctica/semántica muy informal (orientada al usuario). Pocas herramientas de desarrollo.	Los clientes necesitan saber las operaciones y su semántica antes del uso. Los clientes necesitan puertos dedicados para diferentes tipos de notificaciones. Las instancias del proceso son creadas implícitamente.

Tabla 1 Comparación entre REST y SOAP

Luego de haber explicado las características y ventajas de los componentes anteriores del servicio de mensajería se decide utilizar REST, debido a que los sistemas basados en este estilo ofrecen potencial

escalabilidad, así como el acceso con escaso consumo de recursos a sus operaciones debido al limitado número de operaciones y el esquema de direccionamiento unificado.

1.2.4 Descripción del servicio

Los servicios *web* deben contar con una interfaz pública la cual es descrita por un formato WSDL (*Web Services Description Language*) que es un tipo de documento XML que describe lo que hace un servicio *web*. Este provee información importante para los desarrolladores, esto se debe a que describe el formato de los mensajes que utiliza y a cuales se puede responder. WSDL se puede utilizar para describir servicios *web* basados en REST. Un documento XML WSDL presenta los siguientes elementos:

- Tipos: tipos de datos usados por los mensajes.
- Mensaje: que datos son enviados desde un nodo a otro.
- Tipo de puerto: define las operaciones que pueden ser llamadas.
- Límite: es la descripción del protocolo que se está utilizando para transportar el mensaje que puede ser HTTP POST, HTTP GET, SOAP y MIME.
- Servicio: define una colección de puertos (nodos); el puerto especifica una dirección para el límite definiendo así la comunicación para un nodo específico[19].

1.2.5 Descubrimiento de Servicios

El descubrimiento de servicios consiste en localizar o descubrir uno o varios documentos relacionados que describen un servicio *web* determinado mediante un WSDL. A través del proceso de descubrimiento los clientes conocen la existencia de un servicio *web* y dónde encontrar el documento de descripción.

Para encontrar un servicio *web* XML en particular es necesario utilizar los directorios de servicios *web*, estos proporcionan ubicaciones centralizadas en las que los proveedores pueden publicar información acerca de los servicios de que disponen. Es necesario utilizar un directorio de servicios para localizar una organización que proporcione un servicio *web* con una función determinada o para especificar qué ofrece una organización concreta. UDDI *(Universal Description Discovery and Integration)* es un marco independiente de la plataforma para describir servicios, negocios e integrar servicios de negocios. La estructura de UDDI está basada sobre los servicios estándares de la *web*, lo que quiere decir que UDDI es accesible como un servicio *web*.

Su objetivo es ser accedido por los usuarios y clientes, y dar paso a documentos WSDL en los que se describen los requisitos del protocolo y los formatos del mensaje solicitado para interactuar con los servicios *web* del catálogo de registros[20].

1.3 Minería de datos y texto

La minería de datos es un proceso que consiste básicamente en la extracción de patrones en un conjunto de datos, mezclando procedimientos de inteligencia artificial y estadística, con la finalidad de poder clasificar o hacer predicciones respecto a los datos al enfrentar altos volúmenes de información.

Un área que representa un subconjunto de la minería de datos es la minería de texto, que corresponde a utilizar el concepto anterior, pero con la especificidad de hacerlo sobre texto en lugar de cifras o números; diferencia clave a la hora de manipular la información, ya que en lugar de procesar sólo atributos bien definidos, se utiliza generalmente el universo de palabras disponibles en un conjunto de textos, aumentando la dimensionalidad del problema en varios órdenes de magnitud. El objetivo de la minería de texto es analizar colecciones grandes de documentos sin estructura con el propósito de extraer patrones interesantes y no triviales[21]. Para tener efectividad, ambos mecanismos, la minería de datos y la minería de texto, requieren utilizar datos cuidadosamente seleccionados, acotados y muchas veces modificados para mejorar la calidad de la información, para esto los datos deben pasar por un proceso de preparación que se realiza mediante una numeración y una categorización. Las codificaciones numéricas tienen una definición en un libro de códigos y la categorización asigna a los documentos una o más categorías, etiquetas o clases, basadas en el contenido[22].

1.4 Categorización automática de textos

La categorización es una herramienta que permite almacenar artículos y otras páginas dentro de categorías, que reúnen a varios artículos o páginas de características similares. Las categorías tienen a su vez subcategorías (más específicas) y supercategorías (más generales), permitiendo navegar de lo general a lo concreto y viceversa, a través de una estructura de árbol. Ayudan a los lectores a conocer qué artículos existen sobre un determinado tema, incluso sin saber de antemano si ya existen o con qué nombre aparecen.

Consiste en descubrir variables que sean útiles para discriminar textos que pertenecen a categorías existentes. Por tanto, el objetivo de un categorizador (programa que ejecuta algoritmos de categorización) es indicar la categoría que debe asociarse a cada texto, partiendo de un esquema o modelo de categorización previamente creado[23]. La categorización automática de documentos tiene aplicaciones en diversos problemas, como por ejemplo: detección de e-mail spam, detección de contenido sexual explícito en sitios/páginas *web*, mantención de directorios o búsqueda en colecciones documentales enfocadas a temas específicos[24].

1.5 Servicios *web* de categorización de texto

Existen en el mundo empresas que brindan servicios *web* para categorizadores de texto. A continuación se analizan algunos de ellos:

1.5.1 Bitex

Bitext es una empresa de capital español, cuyo objetivo es proporcionar capacidades semánticas a cualquier aplicación de negocio. Bitext ha desarrollado una plataforma de análisis lingüístico, capaz de extraer información de negocio de cualquier tipo de texto y en diferentes idiomas. Con esta plataforma proporciona consultoría y soluciones para análisis semántico de redes sociales, analítica de textos, búsqueda semántica y asistentes virtuales.

La API (*Application Programming Interface*) de Bitext es un conjunto de servicios semánticos multilingües. Está diseñada específicamente para las necesidades de las empresas que ofrecen soluciones para procesar contenido generado por los usuarios. Actualmente Bitex dispone de cuatro servicios semánticos: extracción de entidades, extracción de conceptos, análisis de sentimiento y categorización textual. Estos servicios se han desarrollado mediante la tecnología *NaturalExtractor* y *NaturalOpinions*.

Extracción de entidades: este servicio reconoce y extrae nombres propios, entidades numéricas y entidades alfanuméricas y las clasifica según su tipo (personas, empresas, ciudades, calles). *NaturalExtractor* realiza el reconocimiento y clasificación de las entidades por medio de una combinación de análisis lingüístico, detección de patrones alfanuméricos y diccionarios.

Extracción de conceptos: este servicio detecta y extrae conceptos de cualquier tipo de texto, ya sea de alta calidad o lenguaje coloquial. *NaturalExtractor* aplica a los conceptos un proceso de normalización y los reduce a una forma estándar, para tratar consistentemente todas las apariciones de un mismo concepto. Una correcta normalización de los conceptos es esencial para el funcionamiento de servicios como categorización o para detección de tendencias.

Análisis de sentimiento: este servicio clasifica el texto según los sentimientos positivos o negativos que se expresan en él mediante opiniones y comentarios referidos a un tema dado o en general. *NaturalOpinions* emplea diferentes técnicas de análisis lingüístico para detectar las entidades y conceptos de los que el autor del texto está hablando. Posteriormente, se identifican estructuras lingüísticas específicas para detectar la actitud del autor sobre cada tema.

Categorización textual: este servicio permite clasificar texto en diferentes categorías, según un diccionario predefinido adaptado específicamente para cada sector. Mediante *NaturalExtractor* se analiza el texto lingüísticamente para asegurar que se localizan todas las apariciones de los términos del

diccionario, aunque estén expresadas de forma distinta a la forma canónica. Este proceso asegura un alto nivel de precisión en la categorización[25].

Los servicios antes mencionados realizan técnicas de análisis lingüístico como: detección de patrones, extracción de conceptos y entidades de cualquier tipo de texto y la detección de idiomas. La API de Bitext representa un nuevo canal de distribución de estas tecnologías, permite a las compañías ofrecer soluciones que analizan el contenido generado por sus usuarios y añadir valor a sus productos por medio de la integración de analítica textual. Los servicios que ofrece Bitext son privados y los precios por volumen parten de 500 euros al mes para 500.000 frases.

1.5.2 Soffik

Soffik es una empresa de servicios *outsourcing* (contratar una empresa externa para que realice un servicio), especializada en la gestión y creación de contenidos para sitios *web* corporativos, empresas *online* y todo tipo de proyectos en la *web*, además de la gestión de usuarios y otros servicios de contenidos tanto para el entorno *online* como *offline*. Trabaja con sitios *web* de diferentes tamaños, niveles y sectores. Además gestiona y crea contenido para páginas o sitios *web*, tanto del sector formativo/educativo, como del sector comercial, del área periodística/informativa, del ocio, o del tipo institucional en sus diferentes variantes. Dentro de sus principales servicios se encuentra la gestión y publicación de contenidos. Algunas de las tareas de gestión que realizan son: la edición, validación y categorización sin discriminar el tipo de plataforma o formato.

En el servicio de categorización Soffik trata el contenido de un sitio *web* como un ente variable, mutable y generalmente en crecimiento. Planifica la categorización de acuerdo a los parámetros SEO (*Search Engine Optimization*) inherentes a la estructura del sitio *web*, o a la proyección de la misma en caso de rediseño o reestructuración de categorías. Los servicios que ofrece Soffik son privados y la forma de pago para estos puede ser por el precio de un determinado servicio en un determinado tiempo de trabajo o por jornada para trabajos generalmente de mediano y largo alcance[26].

1.5.3 Categorizador textual para enrutadores de llamadas, asistentes virtuales y clasificación de correo electrónico.

Es un sistema que permite a las plataformas automáticas de atención al cliente (telefónicas, *web*, correo electrónico), reconocer e interpretar el lenguaje espontáneo de los usuarios y asignarlo a categorías temáticas. Está compuesto por varios módulos, entre ellos: una herramienta con interfaz gráfica para crear los espacios temáticos y un categorizador que emplea dichos espacios para procesar y categorizar unidades de texto. Combina técnicas de representación de texto de manera vectorial y simplificada, con

algoritmos inspirados en psicología cognitiva. Estos algoritmos extraen la idea principal de los textos mediante mecanismos basados en modelos formales de comprensión humana. Es un sistema muy flexible que permite tiempos cortos de respuesta al cambiar criterios en la línea de negocio. No requiere un reprocesamiento por cada cambio ni etiquetados masivos de muestras. El sistema procesa y categoriza peticiones textuales en la nube, actuando con arquitectura SaaS (*Software as a service*). Emplea protocolos estándar de *Internet* (http, XML, SOAP), por lo que se integra con cualquier arquitectura existente y el coste de desinstalación es mínimo[27].

Este es un sistema muy útil para categorizar unidades de texto derivadas del lenguaje espontáneo de los usuarios, pero utiliza SOAP como protocolo para el intercambio de mensajes y SaaS como arquitectura, además exige una licencia o pagar por el uso de cada uno de sus módulos.

1.5.4 Tanalyzer

Tanalyzer es una plataforma de clasificación de texto de la empresa Betazeta Networks S.A. dedicada a la publicación de información mediante una red de blogs de diversas temáticas. Tanalyzer permite manejar grandes volúmenes de información, visualizar, clasificar y hacer predicciones sobre las temáticas de nuevos documentos utilizando minería de texto. Esta plataforma admite escalar tanto en velocidad como en costos el proceso de categorizar blogs, con un tiempo de ejecución de 2.5 horas para 300.000 documentos.

Tanalyzer está implementada sobre el lenguaje Python utilizando como *framework* a Django, utiliza como arquitectura SOA, esto posibilita que cada aplicación tenga un servicio disponible mediante una url de esta forma cada aplicación se mantiene independiente. Solo es para categorizar textos en el idioma español y debido a que la plataforma tiene un tamaño grande e involucra complejas etapas de manipulación de texto se usa como mecanismo de generación de modelos predictivos LDA (*Latent Dirichlet Allocation*)[21].

1.5.5 Xinetica

Xinetica (Plataforma de Servicios para la Minería de Textos) es una plataforma de servicios *web* de la empresa Datys que brinda soluciones a problemas complejos de Procesamiento de archivos, Minería de Textos y Procesamiento del Lenguaje Natural. Dentro de sus principales características están: la utilización simultánea por gran cantidad de usuarios y la integración de sus funcionalidades en otras aplicaciones de forma sencilla. Xinetica permite: reutilizar las funcionalidades y algoritmos implementados mediante el consumo de servicios *web*, reducir los costos en el desarrollo de aplicaciones industriales y científicas que requieran de los servicios que brinda la plataforma, detectar 73 tipos de formatos de archivos, extraer la información de texto contenida en un archivo, detectar el idioma en textos escritos en

español, inglés, francés, portugués, italiano, ruso y chino-mandarín, detectar y convertir la codificación de caracteres presente en un texto, detectar la función gramatical específica que desempeña cada palabra en un texto: sustantivo, adjetivo, preposición, verbo, adverbio, artículo, conjunción, interjección, detectar errores ortográficos y de concordancia en el texto, así como brindar listado de sugerencias, detectar y clasificar entidades como nombres de Persona, Lugar, Organización y Eventos, clasificar documentos de textos en temáticas definidas por los usuarios y confeccionar sumarios de un conjunto de documentos. Xinetica presenta algoritmos de categorización e investigación tecnológica propia y el uso de esta plataforma requiere de pagar una licencia[28].

1.5.6 Gema

Gema (Análisis Inteligente de Archivos) es una herramienta de la empresa Datys para el análisis de información contenida en grandes volúmenes de documentos de diferentes formatos e idiomas, con funcionalidades orientadas a la recuperación y clasificación de información. Es de gran utilidad en el descubrimiento y análisis de información en pequeños y grandes volúmenes de archivos, el análisis de la mensajería institucional para la detección de fugas de información sensible, en el monitoreo y filtrado de información temática, la detección de tópicos y entidades y la búsqueda de información en almacenes de documentos. Sus principales características son: la descarga de archivos desde directorios, sitios *web* y bases de datos, el procesamiento de documentos de 82 formatos escritos en 7 idiomas, la extracción del texto de imágenes utilizando técnicas de OCR (*Optical Character Recognition*) y brinda un API de servicios *web* para su integración con otros sistemas. Gema permite: recuperar información mediante un eficaz método de búsqueda por contenido, complementar los resultados de la búsqueda utilizando técnicas modernas de recuperación de información para obtener archivos similares, eliminar resultados no relevantes y agrupar documentos comunes. Obtener sumarios conformados por las oraciones con mayor importancia dentro del texto, brindando una idea general del contenido de los documentos, organizar de manera automática la información más relevante respecto a temáticas definidas por los usuarios y visualizar reportes estadísticos que faciliten el análisis de la información y el comportamiento del sistema[29]. Gema presenta gran cantidad de funcionalidades que abarcan áreas o aristas que son innecesarias en los servicios *web* del categorizador de texto de MOCIC y su utilización conlleva el pago de una licencia.

1.5.7 Lidt Noti

Lidt Noti (Lectura Inteligente de Noticias) es un sistema de la empresa Datys para la descarga y procesamiento de sitios digitales noticiosos, procesa grandes volúmenes de noticias y permite su

visualización en línea, por grupos, temáticas y categorías. Tiene gran utilidad en el análisis de información de noticias, en los departamentos de redacción de agencias noticiosas, en el desarrollo de aplicaciones de *software* de información noticiosa y como portal informativo de instituciones. Lidt Noti permite: visualizar nuevas noticias en línea, agrupar noticias de manera jerárquica y por temáticas de interés como Cultura, Deportes, Política, Salud y otros, construir resúmenes de noticias y grupos noticiosos y confeccionar boletines, buscar información noticiosa de manera inteligente a través de la búsqueda simple y avanzada por palabras claves y atributos. Además permite la búsqueda de similares y la retroalimentación por relevancia, detectar automáticamente entidades de tipo personas, lugares, organizaciones, eventos y ofrece gráficos de relaciones entre éstas, definir alertas por correo electrónico y RSS (*Really Simply Sindication*) de últimas noticias, categorías, alertas, búsquedas personalizadas que le permitirán obtener información desde su navegador o aplicación de correo favorita. Cuenta con un sistema de planificación de descarga acorde a las regulaciones del acceso a *Internet* de su organización y que optimiza el tiempo de trabajo de los usuarios, con un portal que le permitirá a instituciones estar informadas con las noticias más recientes y con múltiples servicios *web* que le permitirán insertar información noticiosa en portales temáticos[30]. Para realizar todas sus funcionalidades Lidt Noti presenta requerimientos de *software* y *hardware* que son incompatibles con los servicios *web* a desarrollar y para utilizar este sistema es necesario pagar una licencia.

1.6 Valoración de los servicios *web* estudiados

Los servicios *web* antes analizados aunque permiten la categorización de diferentes volúmenes de textos y poseen características que sirven de aporte a la investigación, como la utilización de técnicas de análisis lingüístico, la gestión de contenido y el uso de algoritmos de clasificación de textos, carecen de rasgos y funcionalidades que son importantes para lograr el objetivo de la presente investigación. Algunos de estos servicios no cumplen con el principio de soberanía tecnológica del país, la universidad y específicamente del CIDI. Soffik no satisface los parámetros de categorización utilizados por el categorizador de texto de MOCIC. El Categorizador textual para enrutadores de llamadas, asistentes virtuales y clasificación de correo electrónico difiere en la arquitectura y en el protocolo para el intercambio de mensajes, ya que utiliza SaaS y SOAP respectivamente. Tanalyzer está limitado a la categorización de texto en el idioma español y posee complejas etapas de manipulación de texto. Los sistemas presentados de la empresa Datys, poseen algoritmos de categorización e investigaciones tecnológicas propias y requieren de licencias para su utilización.

Por todo lo anteriormente expuesto los servicios *web* estudiados no satisfacen las condiciones necesarias para incluirse totalmente en la propuesta de solución.

1.7 Tecnologías seleccionadas

El desarrollo de un *software* es un proceso complejo en el cual se tienen muchos elementos que permitirán su construcción con calidad. Para la implementación de una aplicación, es necesario tener en cuenta las tecnologías sobre las cuales se va a desarrollar el sistema, pues de estas dependerá totalmente su funcionamiento y uso.

1.7.1 Lenguajes y Framework

Python 2.7

Fue creado a finales de la década de los 80 por el holandés Guido van Rossum el cual lanzó en 1991 el código versión 0.9.0. Este lenguaje se puede ejecutar sobre *Windows*, Linux/Unix, Mac OS X y ha sido portado a máquinas virtuales Java y .Net. Es libre de usar, incluso para productos comerciales, debido a su licencia OSI (Iniciativa de Fuente Abierta) de código abierto. Entre las principales ventajas que brinda se encuentra el ser un lenguaje multiparadigma lo que se puede interpretar como que el mismo permite optar por varios estilos ya sea programación orientada a objetos, estructurada o funcional. A continuación se presentan algunas características:

- Multiplataforma

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

- Interpretado

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

- Interactivo

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

- Orientada a Objetos

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

- Funciones y Librerías

Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen variadas librerías que se pueden importar en los programas para tratar temas específicos como la programación de ventanas o sistemas en red.

- Sintaxis Clara

Python tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. Para separar las porciones de código en Python se debe tabular hacia dentro, colocando un margen al código que iría dentro de una función o un bucle. Esto ayuda a que todos los programadores adopten las mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.
Se escoge Python como lenguaje de programación en primer lugar porque el categorizador de texto de MOCIC está desarrollado en este lenguaje. Trabaja con entornos de desarrollos con bajo consumo de máquina lo que aumenta el rendimiento de las estaciones de trabajo. Además su curva de aprendizaje no es elevada y se ajusta a todo tipo de programación (Programación Orientada a Objeto, Programación Estructurada o Programación Funcional)[31].

Django 1.5

Es un *framework* de alto nivel basado en Python que facilita el desarrollo de aplicaciones *web* dinámicas. Es un *framework* que abstrae a los programadores de los problemas comunes del desarrollo *web* y acelera las tareas más frecuentes en la programación. Proporciona un método de mapear las URLs ejecutando un código en especial para cada una. Permite mostrar y validar formularios de manera muy simple, manipulando el código del formulario y adaptándolo a las necesidades de la aplicación. El mismo convierte los datos enviados por los usuarios, en estructuras de datos que pueden ser manipuladas fácilmente. A través de plantillas ayuda a separar el contenido de la presentación evitando tener que manipular la lógica de negocio cuando se necesite realizar cambios de apariencia en la página. Permite lidiar con trescientas peticiones *web* por segundo[32]. Django incluye muchas aplicaciones comunes a todos los sitios *web*, como la autenticación de usuarios o la administración del contenido del sitio. Su arquitectura está inspirada en el patrón Modelo-Vista-Controlador (MVC), encargándose en gran parte de los controladores, y proveyendo herramientas para facilitar el desarrollo de las vistas y los modelos. Django posee además soporte nativo para PostgreSQL como gestor de bases de datos. Entre los sitios que usan este *framework* actualmente se encuentran, entre otros, Mozilla y OpenStack38[33]. Por todas

estas ventajas que ofrece se escoge Django como *framework* para el desarrollo de la capa de servicios para el categorizador de texto de MOCIC.

JSON 1.0

La Notación de Objetos de JavaScript es un formato ligero de intercambio de datos, a las máquinas resulta simple de interpretarlo y generarlo. Está basado en un subconjunto del lenguaje de programación JavaScript. Es un formato de texto completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C++, C#, Java, JavaScript, Perl, Python, y muchos otros. JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

Entre las ventajas se define que su procesamiento por parte de los ordenadores es rápido; se necesitan librerías muy pequeñas para trabajar con él (siendo posible incluso procesarlo sin librería)[34]. JSON será utilizado para el intercambio de información entre los servicios *web* de la capa de servicios del categorizador de texto de MOCIC.

UML 2.0

El Lenguaje Unificado de Modelado es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprenden el desarrollo de *software*. Construido sobre los conceptos fundamentales del paradigma Orientado a Objetos. Entrega una forma de modelar elementos conceptuales como son procesos de negocio y funciones de sistema, además de elementos concretos como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reusables[35]. Este lenguaje será utilizado en la realización de todos los diagramas necesarios para documentar el proceso de desarrollo de *software* de la capa de servicios *web* del categorizador de texto de MOCIC.

1.7.2 Herramientas

Eclipse 3.2

Es un entorno de desarrollo integrado (IDE), portable y multiplataforma. Diseñado por la empresa IBM y actualmente sujeto a la Fundación Eclipse que fomenta una comunidad de código abierto así como un conjunto de productos, capacidades y servicios. Brinda una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta, basada en *plugins* que posibilita integrar

múltiples lenguajes como Java, C++, PHP, Perl y Python. Además de integrar aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces y ayuda en línea para librerías. Entre sus principales características se encuentran: editor de texto, vistas, resaltado de sintaxis, refactorización, completamiento de código, compilación en tiempo real, pruebas unitarias, *frameworks* para el desarrollo de aplicaciones gráficas así como para aplicaciones *web*[36][37]. Esta herramienta se empleará para desarrollar la capa de servicios *web* del categorizador de texto de MOCIC que se implementarán en el lenguaje Python.

PyDev 3.3

Es un *plugin* para Eclipse que entrega a esta herramienta la potencialidad de un editor Python, Jython y IronPython de gran nivel. PyDev *(Python Development Environment)* se distribuye bajo la Eclipse Public License. Entre sus principales características están: incorpora un depurador gráfico de Python, resaltado de sintaxis, auto completado, estructura los proyectos en directorios y recursos lo que ayuda a organizarlos y establecer una jerarquía, integración con Django, navegador de tokens, consola interactiva y cobertura de código[38]. Este *plugin* se utilizará para poder desarrollar la capa de servicios *web* del categorizador de texto de MOCIC en la herramienta Eclipse sobre el lenguaje de programación Python.

Visual Paradigm para UML 8.0

Es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, realizar ingeniería inversa desde gestores de bases de datos, generación de bases de datos, código en lenguajes como Java o C++ desde diagramas y generar documentación. La herramienta también proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML[39]**.**

Visual Paradigm también ofrece: navegación intuitiva entre la escritura del código y su visualización, potente generador de informes en formato PDF/HTML, documentación automática Ad-hoc, ambiente visualmente superior de modelado, sofisticado diagramador automáticamente de layout y sincronización de código fuente en tiempo real u on demand[40]. Esta herramienta será utilizada para la realización y diseño de todos los diagramas necesarios para documentar el proceso de desarrollo de *software* de la capa de servicios *web* del categorizador de texto de MOCIC.

PostgreSQL 9.1

Es el gestor de bases de datos objeto-relacional de código abierto más avanzado actualmente, ofrece control de concurrencia multi-versión (permite a los accesos de sólo lectura continuar leyendo datos consistentes durante la actualización de registros), soporta casi toda la sintaxis SQL (incluyendo sub-consultas, transacciones, y tipos y funciones definidas por el usuario), cuenta con un amplio conjunto de enlaces con lenguajes de programación (incluyendo C, C++, Java, Perl y Python), además de proveer una arquitectura que ha ganado una fuerte reputación por su fiabilidad e integridad de sus datos. También permite bloqueos de tabla y filas para controlar el acceso concurrente a los datos en tablas. Algunos de estos modos de bloqueo los adquiere PostgreSQL automáticamente antes de la ejecución de una declaración, mientras que otros son proporcionados para ser usados por las aplicaciones. Es multiplataforma, estando disponible en casi cualquier Unix y altamente adaptable a las necesidades del cliente. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo paralelamente al sistema[41]. Esta herramienta será utilizada para gestionar todos los datos necesarios que son manejados por la capa de servicios *web* del categorizador de texto de MOCIC.

NGINX 1.4.4

Es un servidor *web* HTTP de código abierto que incluye servicios de correo electrónico con acceso al *Internet* Message Protocol (IMAP) y al servidor Post Office Protocol (POP). Presenta gran estabilidad y configuración simple. Otros usuarios, como Facebook y WordPress.com, lo utilizan porque la arquitectura asíncrona del servidor *web* deja una pequeña huella de memoria y bajo consumo de recursos, haciéndolo ideal para el manejo de páginas *web* múltiples y activas. NGINX se destaca por su rendimiento, escalabilidad y eficiencia de costes. Procesa decenas de miles de conexiones simultáneas en un proceso compacto y con varios núcleos de CPU[42]. Este servidor *web* se utilizará para montar los servicios *web* del categorizador de texto de MOCIC.

1.7.3 Metodología de desarrollo de software

OpenUP

Es una metodología de desarrollo de *software*, basada en RUP (Rational Unified Process), que contiene el conjunto mínimo de prácticas que ayudan a un equipo de desarrollo de *software* a realizar un producto de alta calidad, de una forma eficiente. Es un proceso unificado, iterativo e incremental que utiliza una

filosofía ágil enfocada en la naturaleza de colaboración en el desarrollo de *software*[43]. OpenUP se sustenta en los siguientes principios:

- Promover la colaboración para alinear los intereses comunes, difundir el conocimiento sobre el proyecto, y generar un entorno de trabajo saludable.
- Balancear las prioridades competitivas para maximizar el valor de los participantes del proyecto.
- Desarrollar una arquitectura, al inicio del proceso, con la finalidad de minimizar riesgos y planificar el desarrollo.
- Evolucionar para obtener retroalimentación continua en aras de una mejora constante.

OpenUP se organiza en dos dimensiones: contenido metodológico y contenido procedimental. El contenido metodológico es el que define elementos metodológicos tales como disciplinas, tareas, artefactos y procesos, independientemente de cómo se usen estos o se combinen. El contenido procedimental, por el contrario, es donde se aplican todos estos elementos metodológicos dentro de una dimensión temporal, pudiéndose crear multitud de ciclos de vida diferentes a partir del mismo subconjunto de elementos metodológicos[43].

Elementos metodológicos de OpenUP:

- Disciplinas: se centra en las siguientes disciplinas: gestión del proyecto, requisitos, diseño, implementación, prueba, configuración y gestión de cambios.
- Tareas: se define tareas como la unidad de trabajo que debe ser realizada por algún rol.
- Artefactos: un artefacto se considera a todo aquello que una tarea necesita para realizar su función, o bien la produce o modifica.
- Procesos: los procesos toman los elementos metodológicos y los relacionan entre si dentro de secuencias temporales que satisfacen las necesidades de distintos tipos de proyecto.

El ciclo de vida de un proyecto, según la metodología OpenUP, consta de cuatro fases:

Fase de inicio: en esta fase, las necesidades de cada participante del proyecto son tomadas en cuenta y plasmadas en objetivos del proyecto. Se definen para el proyecto: el ámbito, los límites, el criterio de aceptación, los casos de uso críticos, una estimación inicial del coste y un boceto de la planificación.

Fase de elaboración: en esta fase se realizan tareas de análisis del dominio y definición de la arquitectura del sistema. Se debe elaborar un plan de proyecto, estableciendo unos requisitos y una arquitectura estables. Por otro lado, el proceso de desarrollo, las herramientas, la infraestructura a utilizar y el entorno de desarrollo también se especifican en detalle en esta fase. Al final de la fase se debe tener

una definición clara y precisa de los casos de uso, los actores, la arquitectura del sistema y un prototipo ejecutable de la misma.

Fase de construcción: todos los componentes y funcionalidades del sistema que falten por implementar son realizados, probados e integrados en esta fase. Los resultados obtenidos en forma de incrementos ejecutables deben ser desarrollados de la forma más rápida posible sin dejar de lado la calidad de lo desarrollado.

Fase de transición: esta fase corresponde a la introducción del producto en la comunidad de usuarios, cuando el producto está lo suficientemente maduro. La fase de la transición consta de las subfases de pruebas de versiones beta, pilotaje y capacitación de los usuarios finales y de los encargados del mantenimiento del sistema. En función de la respuesta obtenida por los usuarios puede ser necesario realizar cambios en las entregas finales o implementar alguna funcionalidad más[43][44][45]. Para el desarrollo de la propuesta de solución, se escogió OpenUP como metodología para guiar el proceso de desarrollo de *software* porque:

- En la implementación de la capa de servicios participará un grupo pequeño de desarrolladores del cual el cliente forma parte.
- El proceso de desarrollo de software aunque tiene que asegurar la calidad debe ser ágil.
- Dentro de las políticas del centro se encuentra el empleo de la metodología OpenUP para el desarrollo de las aplicaciones con el objetivo de aprovechar características que esta posee como: facilidad en la colaboración para alinear los intereses y un entendimiento compartido, así como en la colaboración técnica, además de minimizar excesos y trabajo extra. Por otra parte esta metodología permite la continua evolución para reducir riesgos, demostrar y obtener resultados.

1.8 Conclusiones parciales

- El estudio de los servicios y protocolos usados para permitir la interacción entre los servicios *web* garantizó entender el funcionamiento para la transferencia de los mensajes entre aplicaciones sobre la red y la codificación de los mensajes a un lenguaje estándar.
- Con el estudio de los servicios *web* existentes en el mundo se adquirió una serie de conocimientos que son útiles para el desarrollo de la propuesta de solución de la presente investigación, aunque estos no reúnen los requisitos para ser incluidos completamente en la misma.
- El estudio de las tecnologías seleccionadas por los desarrolladores y el cliente, permitió un acercamiento a los elementos del marco de trabajo y sus características, para hacer más fácil su uso durante el desarrollo de la propuesta de solución.

Capítulo 2: Diseño de la solución

2.1 Introducción

Teniendo en cuenta las necesidades reales presentadas en la situación problemática de este trabajo de diploma, se decide desarrollar una capa de servicios *web* para el categorizador de texto del Motor de Clasificación Inteligente de Contenido. Esta capa de servicios debe utilizar las funcionalidades del categorizador de texto ya implementadas para que puedan ser accedidas por diferentes aplicaciones informáticas con necesidad de emplear las funcionalidades del categorizador de texto de MOCIC. Para el desarrollo de los diferentes servicios de la capa es de suma importancia tener en cuenta el funcionamiento de los servicios REST, así como la interoperabilidad entre estos servicios.

El presente capítulo contiene la propuesta de solución al problema científico de la investigación, en él se definen las clases del dominio así como su interrelación, los requisitos funcionales y no funcionales, y el modelado y descripción de los casos de uso.

2.2 Propuesta del sistema

La capa de servicios estará enfocada en permitirle a las aplicaciones utilizar las funcionalidades del categorizador de texto de MOCIC, a través de peticiones REST, brindando como resultado un fichero JSON con la respuesta de la petición.

Para que otras aplicaciones en diferentes entornos de desarrollo puedan utilizar las funcionalidades del categorizador de texto se desarrollará una interfaz de programación de aplicaciones (API), con la base tecnológica definida anteriormente en el capítulo 1, para que se conecte con la capa de servicios *web* del categorizador de texto, y así brindar una serie de funcionalidades, que harán más fácil el trabajo de los desarrolladores.

Los servicios que contendrá la capa de servicios para el categorizador de texto de MOCIC son:

- **Autenticar usuario**: permitirá autenticar un usuario entrando los parámetros de conexión usuario y contraseña, para los cuales se creará una sesión correspondiente a este usuario. Los roles pueden ser administrador y usuario. El administrador podrá realizar todas las peticiones de la capa de servicios, mientras que el usuario solo podrá realizar las peticiones a las cuales tenga permiso.
- **Extracción de texto plano de documentos en varios formatos**: permitirá la extracción de texto plano de archivos HTML, PDF, POSTSCRIPT, RFC822 (mensaje de correo electrónico) y algunos de los formatos de *Open Office* y *Microsoft Office*. Esta petición devuelve el texto plano del documento en formato JSON.

- **Detección de la codificación de caracteres en texto plano**: permitirá detectar la codificación de los caracteres de un documento en texto plano. Esta petición devuelve la codificación de caracteres determinada en formato JSON.
- **Detección de idioma**: permitirá la detección del idioma del documento, utilizando *stopwords* y trigramas. Esta petición devuelve el idioma del documento analizado en formato JSON.
- **Subir un categorizador**: consistirá en cargar un fichero .svn, este fichero es un tipo de categorizador que permitirá posteriormente darle una categoría a un documento que se le realice la petición de "Categorizar texto", esta petición muestra en formato JSON el archivo del categorizador subido.
- **Categorizar texto**: permitirá categorizar el texto del archivo que se adjunte, atendiendo a las categorías que tiene concebido el clasificador en su entrenamiento. Esta petición devuelve el resultado del servicio en formato JSON, con la categoría en la que ha sido clasificado el documento.
- **Crear entrenamiento**: permitirá crear un entrenamiento para uno o varios categorizadores de diferentes formas, subiendo una colección que tenga definido sus categorías o creando una nueva colección y adicionándole los nuevos documentos.
- **Crear Categorizador**: permitirá crear un categorizador con los parámetros que lo definen como son: nombre, tipo de categorizador, idioma, y características como si está entrenado o no, las categorías que tiene, la cantidad de términos y filtros, todas estas características y parámetros los categorizadores las obtienen de la colección con la que se le realiza el entrenamiento a cada uno. Esta petición muestra en formato JSON el nuevo categorizador.
- **Entrenar Categorizador**: permitirá entrenar un categorizador en las categorías que contenga la colección con la cual va a ser entrenado. Para esto es necesario especificar la colección y seleccionar el categorizador que se desea entrenar. Esta petición muestra en formato JSON el entrenamiento del categorizador.
- **Evaluar Categorizador**: permitirá evaluar un categorizador según sus categorías dando los valores generales de la evaluación de este como: la cantidad total de documentos, precisión, exhaustividad, documentos categorizados correctamente, documentos categorizados incorrectamente. Los valores de la evaluación por categorías son los mismos de la evaluación general, lo que específicos para cada categoría y una matriz de confusión que muestra la relación del total de textos evaluados en una categoría por la cantidad de textos evaluados en esa categoría correctamente. Para esto es necesario

seleccionar el categorizador que se va a evaluar y se adjunta los documentos con los cuales se realizará el servicio. Esta petición muestra en formato JSON la evaluación del categorizador.

Para ver el formato de intercambio (JSON) donde se muestran los resultados de los servicios que brindará la solución a desarrollar, remitirse al Anexo 3.

La imagen a continuación muestra gráficamente la idea del sistema a desarrollar en el presente trabajo de investigación.

Figura 1 Propuesta del sistema

2.3 Modelado del Dominio

El modelado del dominio representa cuáles son y cómo se relacionan los conceptos relevantes en el dominio de un problema. Explica los conceptos más significativos, teniendo como cualidad esencial representar elementos del mundo real.

2.3.1 Modelo de Dominio

Es un artefacto de la disciplina de análisis, construido con las reglas de UML, presentado como uno o más diagramas de clases y que contiene, no conceptos propios de un sistema de *software* sino de la propia realidad física. Los modelos de dominio pueden utilizarse para capturar y expresar el entendimiento

ganado en un área bajo análisis como paso previo al diseño de un sistema, ya sea de *software* o de otro tipo[46].

A continuación se muestra el modelo de dominio que representa la capa de servicios *web* para el categorizador de texto de MOCIC:

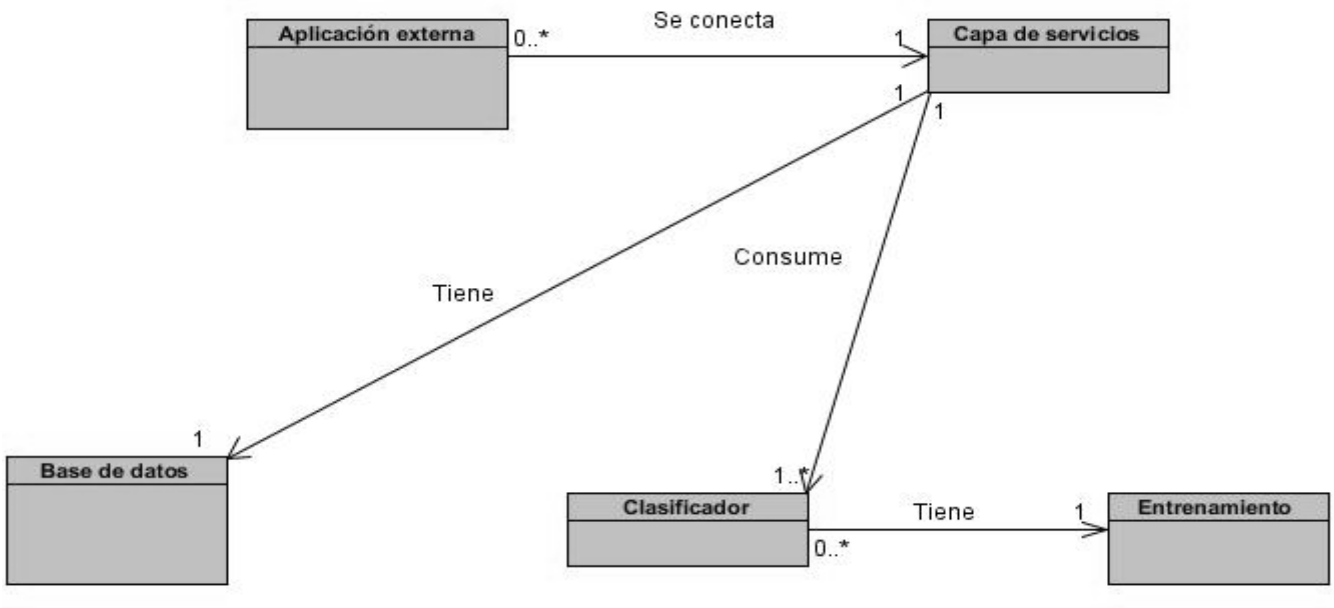

Figura 2 Modelo de dominio del sistema

2.3.2 Descripción de clases del Modelo de Dominio

Aplicación externa: aplicación que contiene la interfaz *web* que consume de la capa de servicios del categorizador de texto.

Base de datos: aplicación que almacena y registra de forma ordenada la información referente a los clasificadores o categorizadores y entrenamientos, además de los usuarios que consumen servicios del categorizador de texto.

Clasificador o Categorizador: archivo .svn que contiene un idioma, un grupo de categorías, filtros y transformaciones que obtiene del entrenamiento que recibe. Este tiene la propiedad de clasificar, detectar y extraer datos de diferentes documentos.

Entrenamiento: consiste en entrenar a un clasificador con una colección de documentos de la cual este toma sus atributos los cuales son: idioma, categorías, cantidad de términos y filtros.

Capa de servicios: sistema que brinda los servicios del categorizador de texto. Se conecta directamente con la base de datos y consume los métodos del categorizador de texto.

2.4 Técnicas de obtención de requisitos

La Ingeniería de Requisitos (IR) es el área del conocimiento que ayuda a realizar la obtención, análisis, especificación y validación de los requisitos del *software*. Tiene como principal objetivo proporcionar el mecanismo adecuado para entender lo que el cliente quiere, analizar las necesidades, evaluar la factibilidad, negociar una solución razonable y especificarla sin ambigüedades, así como evaluarla conforme esta se transforma en un sistema operacional[47].

La IR ofrece técnicas para identificar y extraer los requisitos, dentro de las que se encuentran: entrevistas, cuestionarios, tormentas de ideas, desarrollo conjunto de aplicaciones (JAD), escenarios, prototipos, reuniones de facilitación o taller de requisitos y observación directa (Etnografía).

De las técnicas antes mencionadas para la captura de requisitos se utilizan las siguientes:

Tormenta de ideas: es una técnica de reuniones en grupo que permite generar ideas originales en un ambiente libre de críticas. Ayudó a generar una gran variedad de vistas del problema y a formularlo de diferentes formas al inicio del proceso de captura, cuando los requisitos son todavía muy difusos[48].

Entrevistas: es una técnica en la que los desarrolladores se reúnen con el cliente para que este les exprese todas las necesidades y funcionalidades a satisfacer. Mediante esta técnica se obtuvieron la mayoría de los requisitos a cumplir por el sistema.

Observación directa: es una técnica de observación que se puede utilizar para entender los requerimientos sociales y organizacionales. Se centra en los siguientes aspectos: La forma en la que las personas trabajan y no en como el sistema los hace trabajar. Mediante esta técnica se tuvieron en cuenta las características de los usuarios que van a interactuar con el sistema para hacerlo más cómodo y fácil[49].

2.5 Definición de los requisitos funcionales y no funcionales

Se define un requisito como un atributo necesario en el sistema, una declaración que identifica una capacidad, característica o factor de calidad de un sistema con el fin de tener valor y utilidad para un cliente o usuario; con lo que se pretende cumplir determinadas necesidades o restricciones operativas y que contribuyen a solucionar un problema en un entorno real[47].

2.5.1 Requerimientos funcionales:

Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir. Las herramientas a desarrollar cuentan con los siguientes requisitos funcionales:

RF1 Implementar servicio Rest para autenticación.

RF2 Implementar servicio Rest para la extracción de texto plano de documentos en varios formatos.

RF3 Implementar servicio Rest para detección de la codificación de caracteres en texto plano.

RF4 Implementar servicio Rest para detección de idioma.

RF5 Implementar servicio Rest para subir un categorizador.

RF6 Implementar servicio Rest para categorizar texto.

RF7 Implementar servicio Rest para crear entrenamiento.

RF8 Subir una colección.

RF9 Crear una colección.

RF10 Adicionar documentos a la colección.

RF11 Implementar servicio Rest para crear categorizador.

RF12 Implementar servicio Rest para entrenar categorizador.

RF13 Implementar servicio Rest para evaluar categorizador.

RF14 Desarrollar interfaz *web* de administración.

RF15 Desarrollar interfaz *web* para probar las funcionalidades del categorizador.

2.5.2 Requerimientos no funcionales:

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener.

Requisitos de Software del Servidor:

- Sistema Operativo Ubuntu 12.04 o superior.
- Requiere para su funcionamiento tener instalado Python 2.7.
- Utilizar como servidor de Base de Datos a PostgreSQL 9.1.
- Utilizar como servidor *web* Nginx.

Hardware:

- Requiere como mínimo de RAM (*Random Access Memory*) 512 MB (*Mega Bytes*).
- El disco duro requiere como mínimo 5 GB (*Giga Bytes*) para almacenar la Base de Datos.

Seguridad:

Disponibilidad: la capa de servicios debe estar disponible para aquellas personas con acceso a la información.

Confidencialidad: los datos manejados por la capa de servicio no deben ser accedidos por personal no autorizado.

Integridad: los datos almacenados en la base de datos de la capa de servicio no pueden presentar incoherencias con respecto a la realidad.

Usabilidad:

La capa de servicio contará con una interfaz intuitiva que permita al administrador probar las funcionalidades del categorizador.

Portabilidad:

Independientemente de la plataforma de desarrollo en la que fue implementada la capa de servicios, las aplicaciones que la consuman pueden hacerlo sin tener en cuenta el lenguaje de programación en que fueron desarrolladas.

Rendimiento:

La capa de servicios debe ser capaz de responder una petición en el menor tiempo posible dependiendo del tipo de petición y los datos que se manejan en esta.

Legales:

La capa de servicios y toda la documentación generada pertenecen al Centro de Ideoinformática (CIDI) y la Universidad de las Ciencias Informáticas.

2.6 Modelo de Casos de Uso del sistema

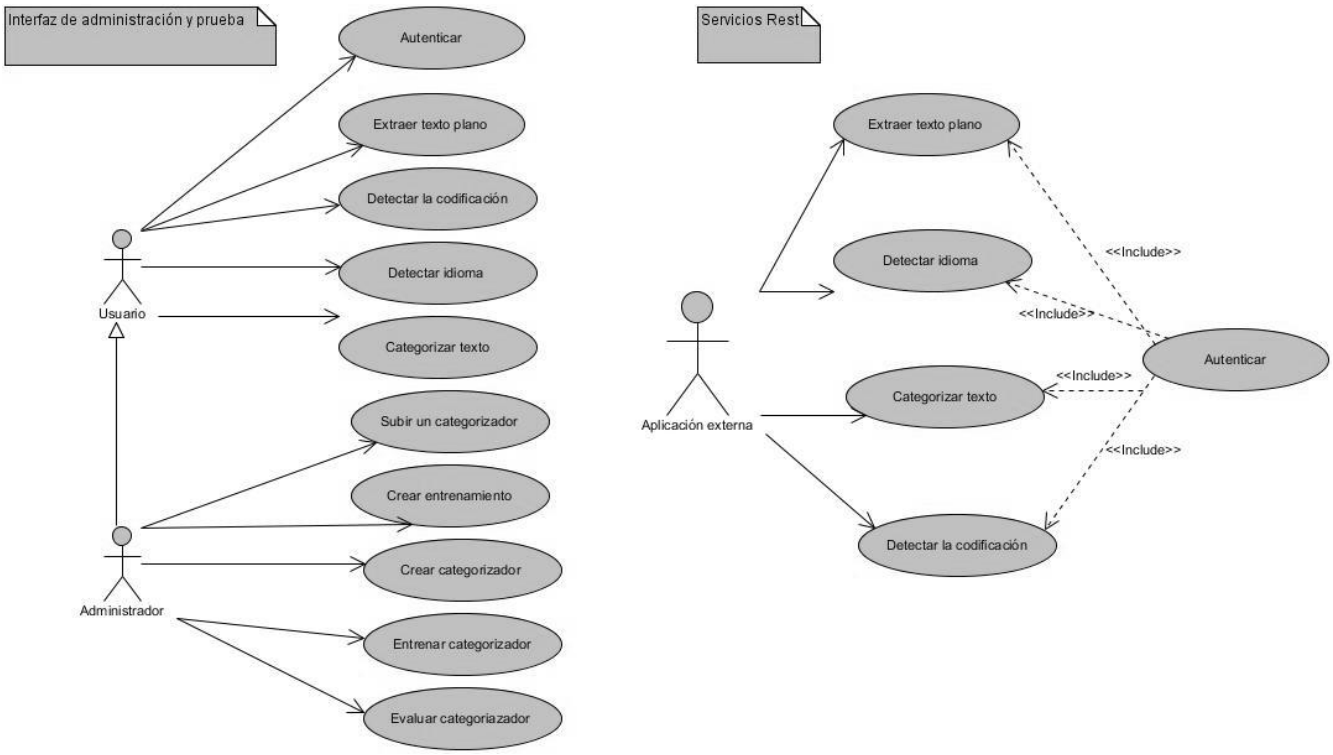

Figura 3 Diagrama de casos de uso del sistema

2.6.1 Definición de los casos de uso del sistema. Interfaz de administración y prueba.

A continuación se definen los casos de usos del sistema identificados para la capa de servicios del categorizador de texto de MOCIC:

CU_1	Autenticar Usuario
Actor	Usuario
Descripción	El caso de uso permite la autenticación para poder consumir los servicios del categorizador de texto a los cuales el usuario tenga permiso.
Referencia	RF1

Tabla 2 Definición del caso de uso Autenticar Usuario

CU_2	Extraer texto plano
Actor	Usuario
Descripción	Es el caso de uso que permite extraer de un documento en formato html, pdf, postscript, rfc822, el texto plano.
Referencia	RF2

Tabla 3 Definición del caso de uso Extracción del texto plano

CU_3	Detectar la codificación
Actor	Usuario
Descripción	Es el caso de uso que permite detectar y devolver la codificación del texto plano en un JSON.
Referencia	RF3

Tabla 4 Definición del caso de uso Detección de la codificación

CU_4	Detectar el idioma
Actor	Usuario
Descripción	Es el caso de uso que permite detectar y devolver el idioma del documento.
Referencia	RF4

Tabla 5 Definición del caso de uso Detección de idioma

CU_5	Categorizar texto
Actor	Usuario
Descripción	Es el caso de uso que permite categorizar un documento atendiendo al entrenamiento que se le da al categorizador.

Referencia	RF5

Tabla 6 Definición del caso de uso Categorizar texto

CU_6	Subir un categorizador
Actor	Administrador
Descripción	Es el caso de uso que permite subir un categorizador que no se encuentre en la base de datos para clasificar documentos.
Referencia	RF6

Tabla 7 Definición del caso de uso Subir un categorizador

CU_7	Crear entrenamiento
Actor	Administrador
Descripción	Es el caso de uso que permite crear un entrenamiento para un determinado clasificador.
Referencia	RF7

Tabla 8 Definición del caso de uso Crear entrenamiento

CU_8	Crear categorizador
Actor	Administrador
Descripción	Es el caso de uso que permite crear un categorizador.
Referencia	RF11

Tabla 9 Definición del caso de uso Crear categorizador

CU_9	Entrenar categorizador
Actor	Administrador
Descripción	Es el caso de uso que permite entrenar un categorizador agregándole colecciones de documentos o creando nuevas colecciones.
Referencia	RF12

Tabla 10 Definición del caso de uso Entrenar categorizador

CU_10	Evaluar categorizador
Actor	Administrador
Descripción	Es el caso de uso que permite evaluar un categorizador atendiendo a la correcta clasificación de

	los documentos.
Referencia	RF13

Tabla 11 Definición del caso de uso Evaluar categorizador

2.6.2 Especificación de Casos de Uso del sistema. Interfaz de prueba y administración.

A continuación se especifican los casos de usos del sistema identificados para la capa de servicios del categorizador de texto de MOCIC:

CU 1	Autenticar Usuario	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando un usuario introduce los parámetros de conexión usuario y contraseña para consumir los servicios de categorizador de texto.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF1	
Precondiciones	Debe estar autenticado	
Flujo de eventos		
Flujo básico <Autenticar usuario>		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Autenticar usuario” e introduce los parámetros “usuario y “contraseña”.	2. El sistema verifica si el usuario y la contraseña son correctos.
Flujos alternos		
	Actor	**Sistema**
		1. Si los parámetros de entrada “usuario” y “contraseña” no son correctos, el sistema envía un mensaje de error.
Postcondiciones	El usuario es autenticado.	

Tabla 12 Especificación del caso de uso Autenticar usuario

CU 5	Categorizar texto
Actores	Usuario

Resumen	El caso de uso se inicia cuando un usuario hace la petición al servicio Categorizar texto. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF5	
Precondiciones	Debe estar autenticado.	
Flujo de eventos		
Flujo básico <Categorizar texto>		
	Actor	Sistema
1.	El actor hace la petición al servicio “Categorizar texto”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento que se quiere categorizar, y un botón “Enviar” para realizar el servicio.
3.	El actor carga el documento que se quiere clasificar y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve la categoría del documento en un JSON.
Flujos alternos		
	Actor	Sistema
		1. Si el archivo no es cargado correctamente, el sistema envía un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 13 Especificación del caso de uso Categorizar texto

CU 6	Subir un categorizador
Actores	Administrador
Resumen	El caso de uso se inicia cuando un administrador hace la petición al servicio Subir un categorizador. El servicio es realizado y se brinda la respuesta correspondiente.
Complejidad	Media
Prioridad	Crítico
Referencia	RF6
Precondiciones	El usuario tiene que ser administrador del sistema.

Flujo de eventos		
Flujo básico < Subir un categorizador>		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Subir un categorizador”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el nuevo categorizador.
3.	El actor carga el categorizador y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema registra el nuevo categorizador en la base de datos.
Flujos alternos		
	Actor	**Sistema**
		1. El sistema verifica si el usuario autenticado es administrador del sistema antes de que realice la petición de Subir un categorizador. 2. Si el categorizador no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 14 Especificación del caso de uso Subir un categorizador

CU 2	Extraer texto plano
Actores	Usuario
Resumen	El caso de uso se inicia cuando un usuario hace la petición al servicio Extraer texto plano. El servicio es realizado y se brinda la respuesta correspondiente.
Complejidad	Media
Prioridad	Crítico
Referencia	RF2
Precondiciones	El usuario debe estar autenticado
Flujo de eventos	
Flujo básico < Extracción de texto plano >	
Actor	**Sistema**

1.	El actor hace la petición al servicio “Extracción de texto plano”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento al cual se le quiere extraer el texto plano.
3.	El actor carga el documento y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve el texto plano del documento.
Flujos alternos		
	Actor	**Sistema**
		1. Si el documento no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 15 Especificación del caso de uso Extracción de texto plano

CU 3	Detectar la codificación	
Actores	Usuario	
Resumen	El caso de uso se inicia cuando un usuario hace la petición al servicio Detección de la codificación. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF3	
Precondiciones		
Flujo de eventos		
Flujo básico < Detección de la codificación >		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Detección de la codificación”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento al cual se le quiere detectar la codificación.
3.	El actor carga el documento y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve la codificación del documento.
Flujos alternos		
	Actor	**Sistema**

		1. Si el documento no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 16 Especificación del caso de uso Detección de la codificación

Para ver la especificación de los casos de uso restantes remitirse al Anexo 4.

2.7 Descripción de la arquitectura y el diseño

La arquitectura de *software* es el diseño de más alto nivel de la estructura de un sistema, es también denominada arquitectura lógica, que consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco.

Una arquitectura de *software* se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como el mantenimiento, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información[50].

2.7.1 Arquitectura de la capa de servicios

La arquitectura de *software* es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones. Es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución[50].

Para el desarrollo de la capa de servicios *web* se decidió utilizar la arquitectura de *software* n-capas, específicamente tres capas; acceso, procesamiento y datos.

Esta arquitectura se basa en una distribución jerárquica de los roles y las responsabilidades para proporcionar una división efectiva del problema a resolver. Separa de forma clara la funcionalidad de cada capa. Tiene como principal ventaja que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que sea necesario realizar algún cambio, solo se corrige el nivel requerido sin tener que revisar todo el código.

El desarrollo de la capa de servicios está enfocado además en la arquitectura orientada a servicios SOA como sus siglas lo indican, ya que brinda una forma bien definida de exposición de servicios, permitiendo

así la creación de sistemas de información bien definidos y altamente escalables. Esta arquitectura brinda un concepto de cómo dar soporte a los requisitos y establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios[51].

A continuación se muestra una figura con la arquitectura de la capa de servicios:

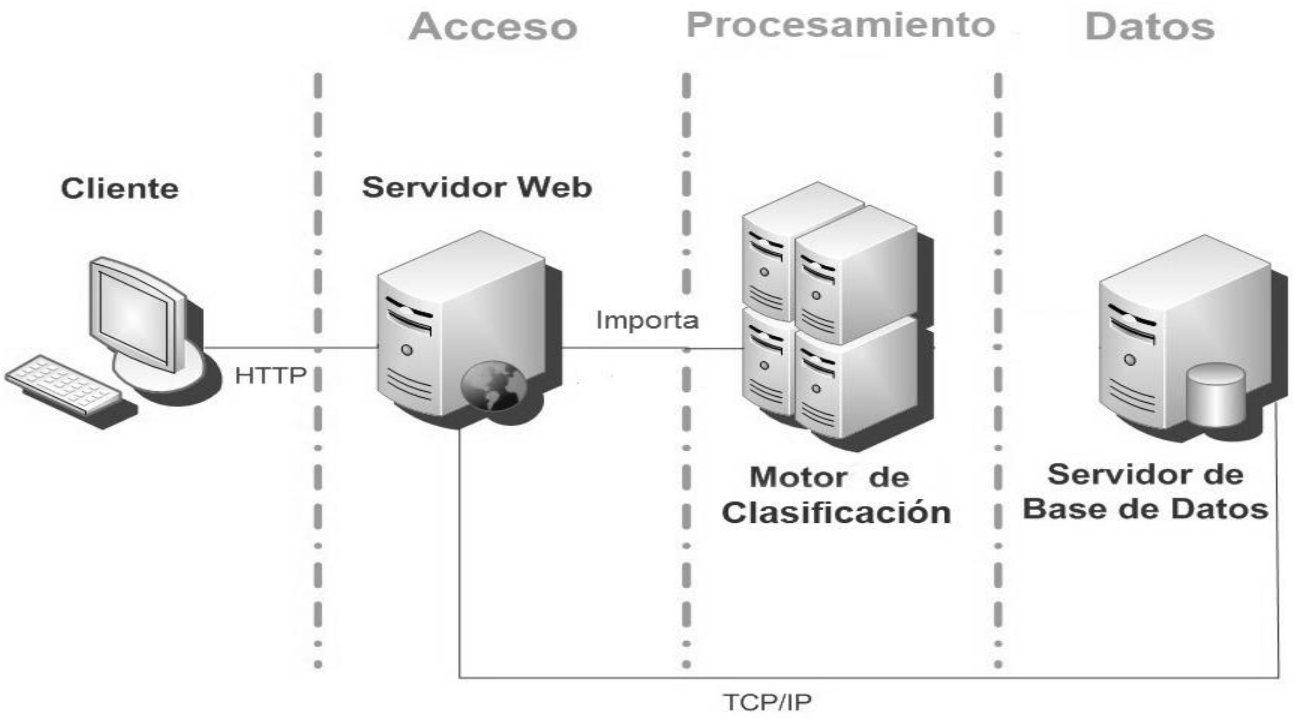

Figura 4 Arquitectura Tres Capas

2.7.2 Componentes de la capa de servicios

La capa de servicios está compuesta por tres componentes; el servidor *web*, el motor de clasificación y el servidor de base de datos.

Servidor *web*: se encarga de recibir todas las peticiones de los usuarios a los servicios. Contiene la interfaz *web* de acceso al sistema tanto para usuarios comunes como para administradores, se encarga de la parte lógica de la gestión de las entidades del sistema, es decir, tiene pleno acceso a la base de datos del mismo. Para la programación de este componente se cuenta con el *framework* para desarrollo *web* Django 1.5 y se tiene Nginx como servidor *web* HTTP.

Motor de clasificación: se encarga de realizar todas las funcionalidades del categorizador de texto (*Textmining*) cuando los usuarios hacen peticiones a estas. La implementación de este componente se realizó con el lenguaje de programación Python.

Servidor de base de datos: contiene la base de datos del sistema en la que se encuentran todos los datos de los clasificadores, los entrenamientos, las evaluaciones de estos, los usuarios normales y los administradores. Los datos del tipo de clasificador, los filtros que estos poseen, las transformaciones y el idioma de estos también permanecen en este servidor. El gestor de bases de datos usado es PostgreSQL 9.1.

2.8 Patrones de diseño

Los patrones de diseño se aplican a un elemento específico del diseño como un agregado de componentes para resolver algún problema de diseño, relaciones entre los componentes o los mecanismos para efectuar la comunicación de componente a componente[47].

2.8.1 Patrones para asignar responsabilidades (GRASP)

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Su propósito fundamental es encontrar un creador que se conecte con el objeto producido en cualquier evento. Se evidencia en la clase *Clasificar* que es la encargada de crear una instancia de la clase *Classifier*.

Alta Cohesión: asignar una responsabilidad de modo que la cohesión siga siendo alta. Caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Se evidencia en la clase *ClasificarSerializer* que cuenta únicamente con las funcionalidades necesarias para serializar y deserializar objetos[52].

2.8.2 Patrones GOF

Existen tres tipos de estos patrones: de creación, estructurales y de comportamiento.

Los patrones **de creación** abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas.

- **Singleton**: garantiza que únicamente se cree una instancia de una clase y provee un punto de acceso global a ella. Todos los objetos que utilizan una instancia de esa clase usan la misma instancia[53].

Los patrones **estructurales** describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

- **Iterador**: define una interfaz que declara métodos para acceder secuencialmente a los objetos de una colección. Una clase que accede a una colección solamente a través de una interfaz independiente de la clase que implementa la interfaz[54].

El uso del patrón se evidencia en el uso de la *GenericAPIView*, que provee los métodos necesarios para la creación y el trabajo con las vistas genéricas.

2.9 Diagrama de clases del diseño

Un diagrama de clases del diseño es un tipo de diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Estas clases se especifican utilizando la sintaxis del lenguaje de programación elegido[55].

2.9.1 Diagrama de clases del diseño con estereotipos *web*

Modela cualquier elemento estructural (Clases) del sistema, y sus relaciones. El estereotipo se utiliza para dar una particularidad a la clase, es un elemento de texto que al ser aplicado a otro elemento define su categoría. Es uno de los mecanismos de extensión del lenguaje que permite cambiar o complementar la semántica de cualquier elemento. A continuación se muestran los diagramas de clases del diseño y la especificación de cada una de sus clases. Para ver los diagramas de clases del diseño restantes remitirse al Anexo 6.

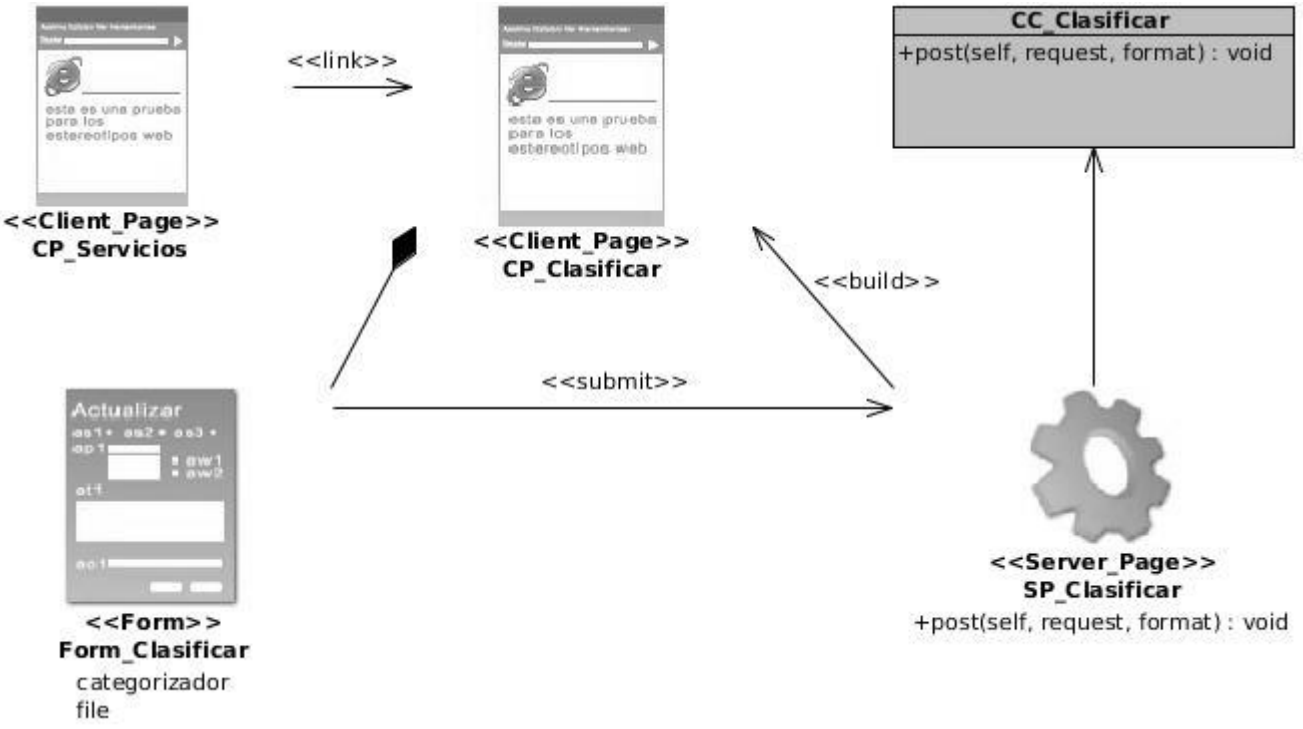

Figura 5 Diagrama de clases del diseño. Categorizar texto

Clase	Descripción
CP_Servicios	La página cliente CP_Servicios permite visualizar los servicios del categorizador de texto.
CP_Clasificar	La página cliente CP_Clasificar permite visualizar categorizadores que existen en la base de datos.
CC_Clasificar	La clase controladora CC_Clasificar permite la interacción entre las clases vistas y las clases modelos.
SP_Clasificar	La clase servidora SP_Clasificar permite realizar el servicio categorizar texto.
Form_Clasificar	El formulario Form_Clasificar permite entrar los datos necesarios para categorizar texto.

Tabla 17 Especificación de clases. Diagrama de clases del diseño. Categorizar texto

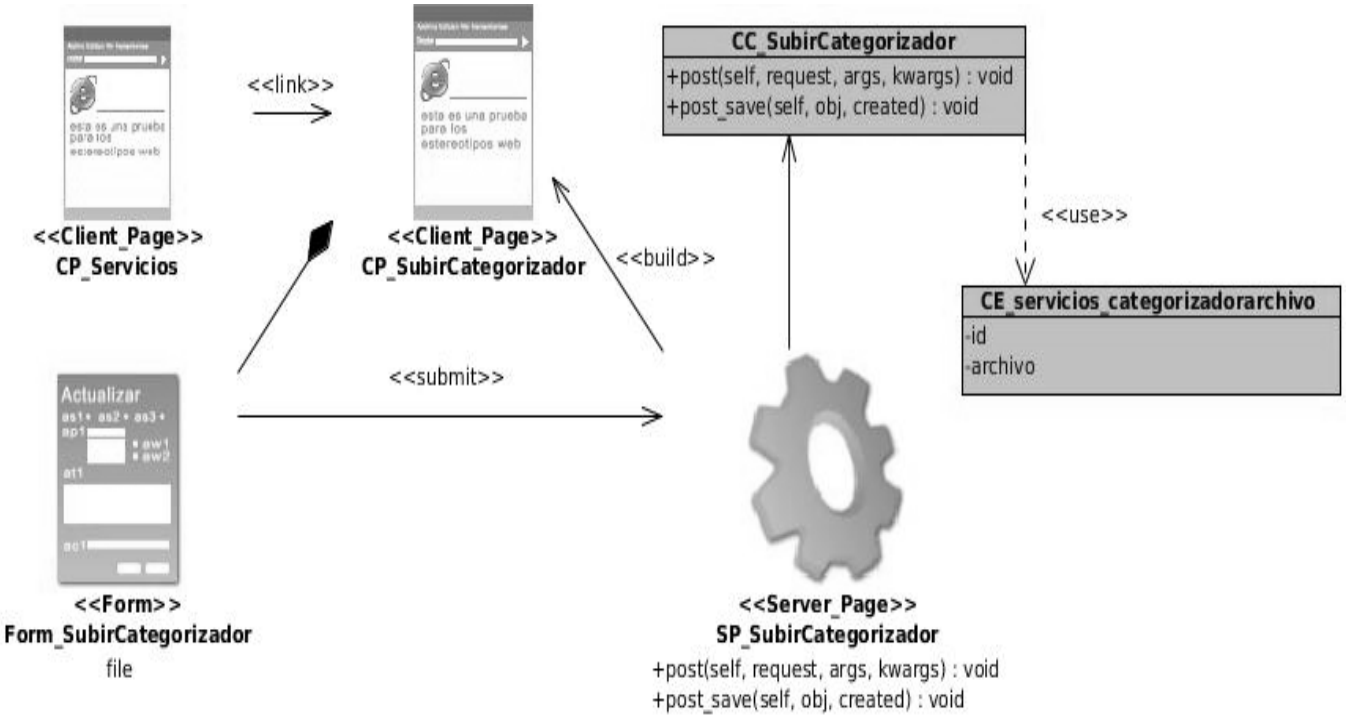

Figura 6 Diagrama de clases del diseño. Subir categorizador

Clase	Descripción
CP_Servicios	La página cliente CP_Servicios permite visualizar los servicios del categorizador de texto.
CP_SubirCategorizador	La página cliente CP_SubirCategorizador permite subir el categorizador de texto especificado.
CC_SubirCategorizador	La clase controladora CC_SubirCategorizador permite la interacción entre las clases vistas y las clases modelos.
SP_SubirCategorizador	La clase servidora SP_SubirCategorizador permite realizar el servicio subir categorizador.
Form_SubirCategorizador	El formulario Form_SubirCategorizador permite cargar el archivo .svn que es el categorizador.
CE_servicios_categorizadorarchivo	La clase entidad CE_servicios_categorizadorarchivo permite acceder a la base

	de datos para guardar el categorizador.

Tabla 18 Especificación de clases. Diagrama de clases del diseño. Subir categorizador

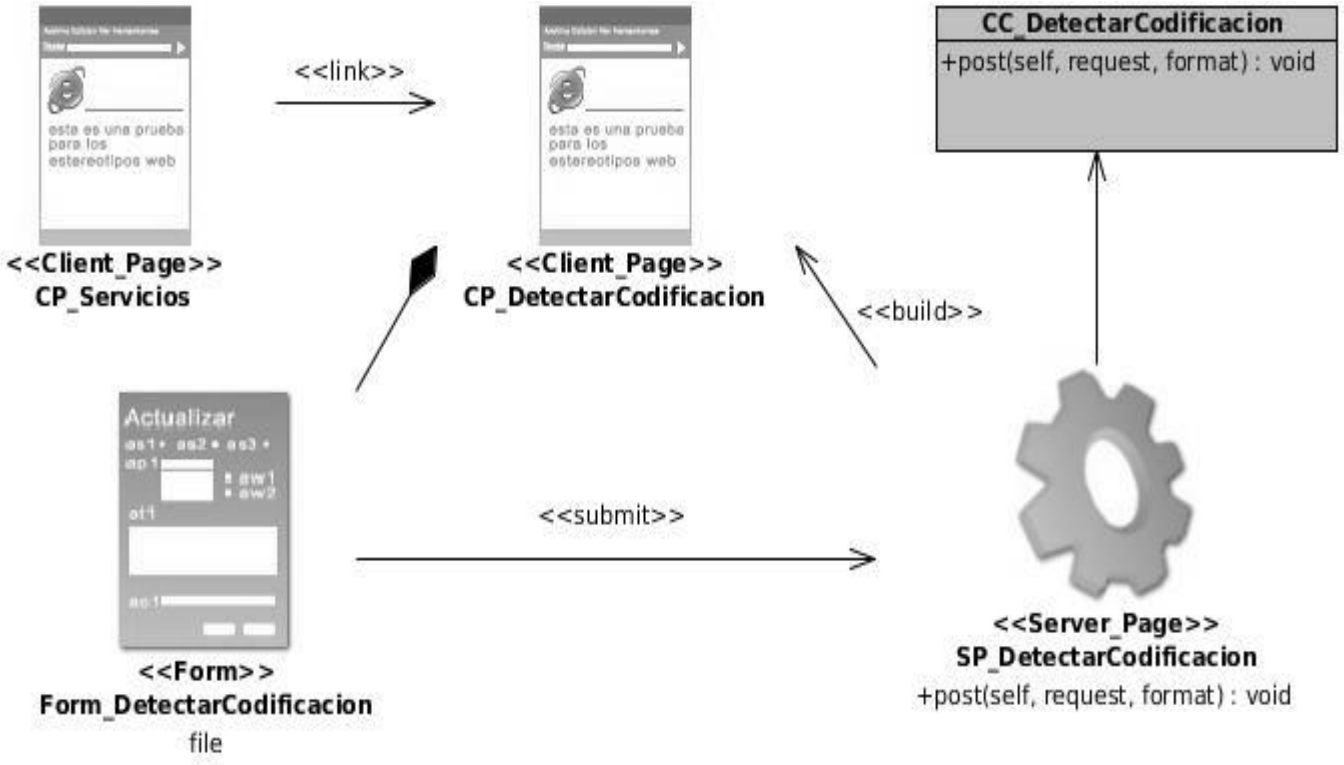

Figura 7 Diagrama de clases del diseño. Detectar codificación

Nombre de la clase	Descripción
CP_Servicios	La página cliente CP_Servicios permite visualizar los servicios del categorizador de texto.
CP_DetectarCodificacion	La página cliente CP_DetectarCodificacion permite detectar la codificación del documento.
CP_DetectarCodificacion	La clase controladora CP_DetectarCodificacion permite la interacción entre las clases vistas y las clases modelos.
CP_DetectarCodificacion	La clase servidora CP_DetectarCodificacion permite realizar el servicio detectar codificación.

Form_SubirCategorizador	El formulario Form_SubirCategorizador permite llenar los campos que sean necesarios para realizar el servicio.

Tabla 19 Especificación de clases. Diagrama de clases del diseño. Detectar codificación

2.9.2 Diagrama de interacción del diseño

Los diagramas de interacción se utilizan para modelar los aspectos dinámicos de un sistema. Están formados por un conjunto de objetos y sus relaciones, incluyendo los mensajes que se pueden enviar entre ellos. Un diagrama de secuencia muestra un conjunto de mensajes, dispuestos en una secuencia temporal. Cada rol en la secuencia se muestra como una línea de vida, es decir, una línea vertical, y un uso de los diagramas de secuencia es mostrar la secuencia del comportamiento de un caso de uso.

Muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se modela para cada caso de uso, y contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos.

Típicamente se examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si se dispone de la descripción de cada caso de uso como una secuencia de varios pasos, entonces se puede "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos. Muestra los objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales[56]. A continuación se muestran los diagramas de secuencia, para ver los restantes remitirse al Anexo 7.

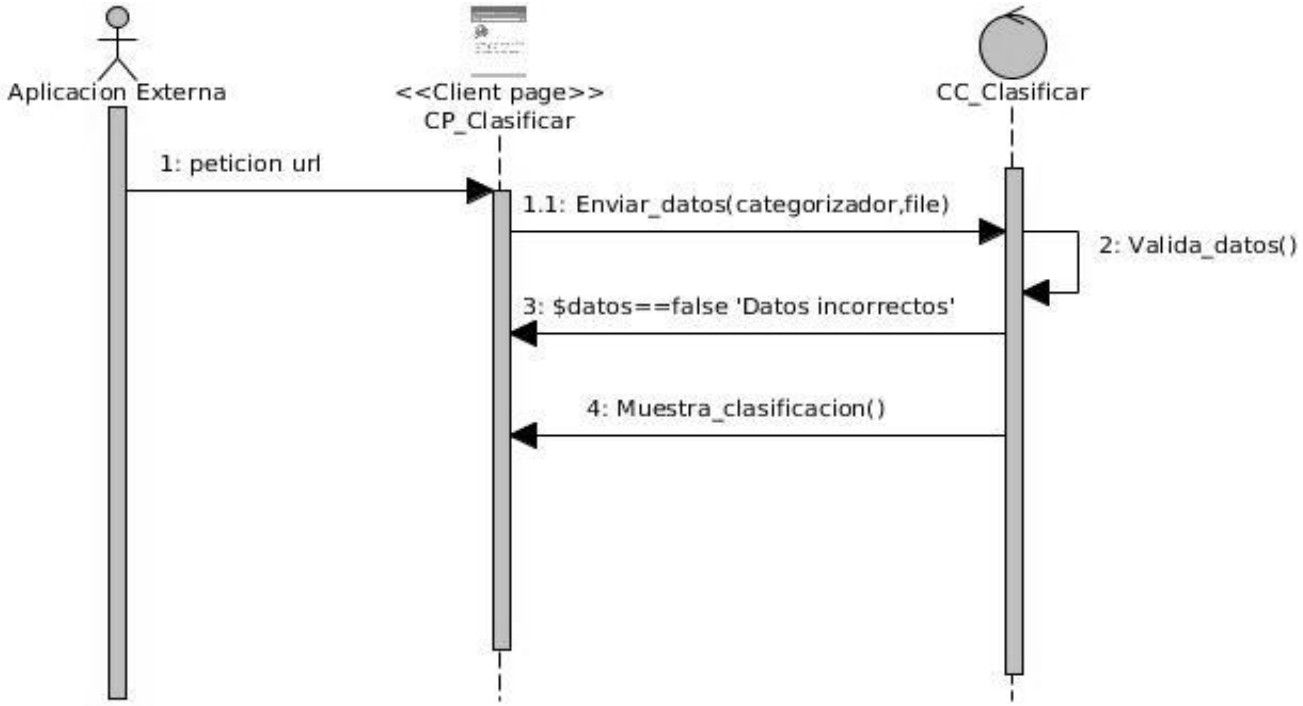

Figura 8 Diagrama de secuencia. Categorizar texto

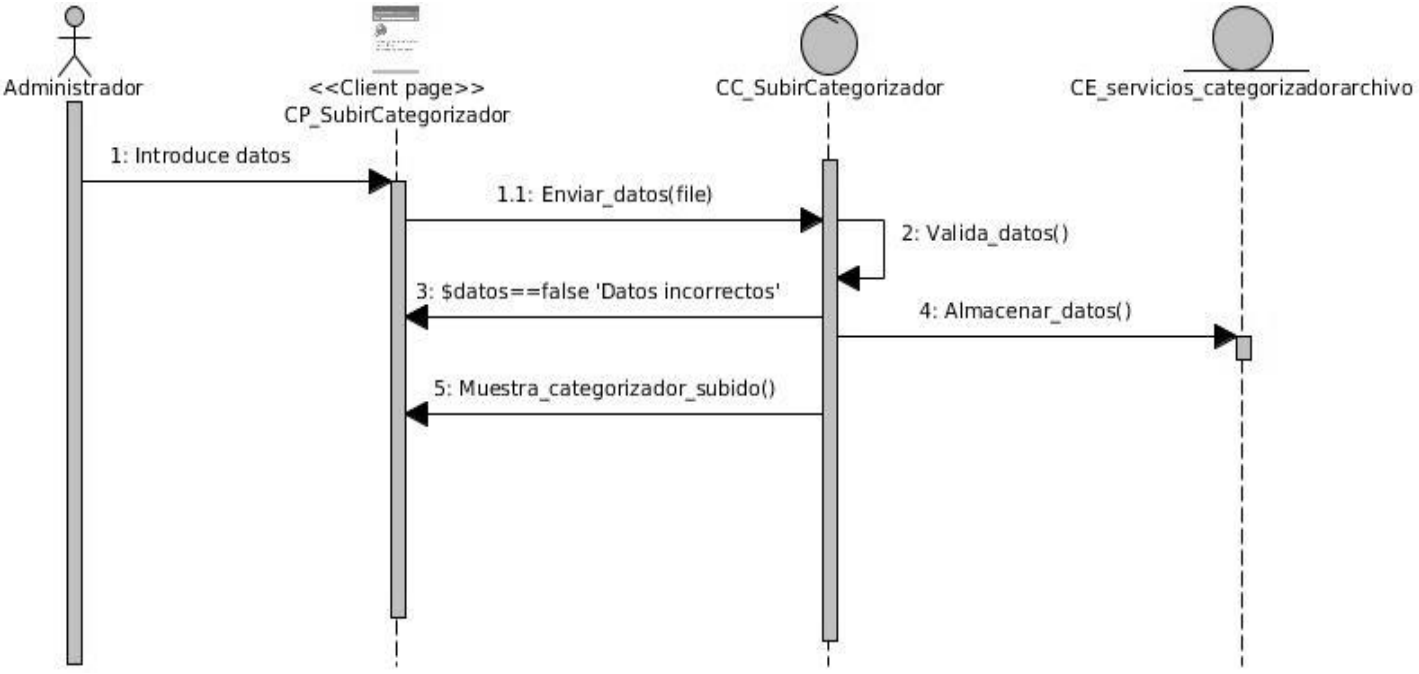

Figura 9 Diagrama de secuencia. Subir categorizador

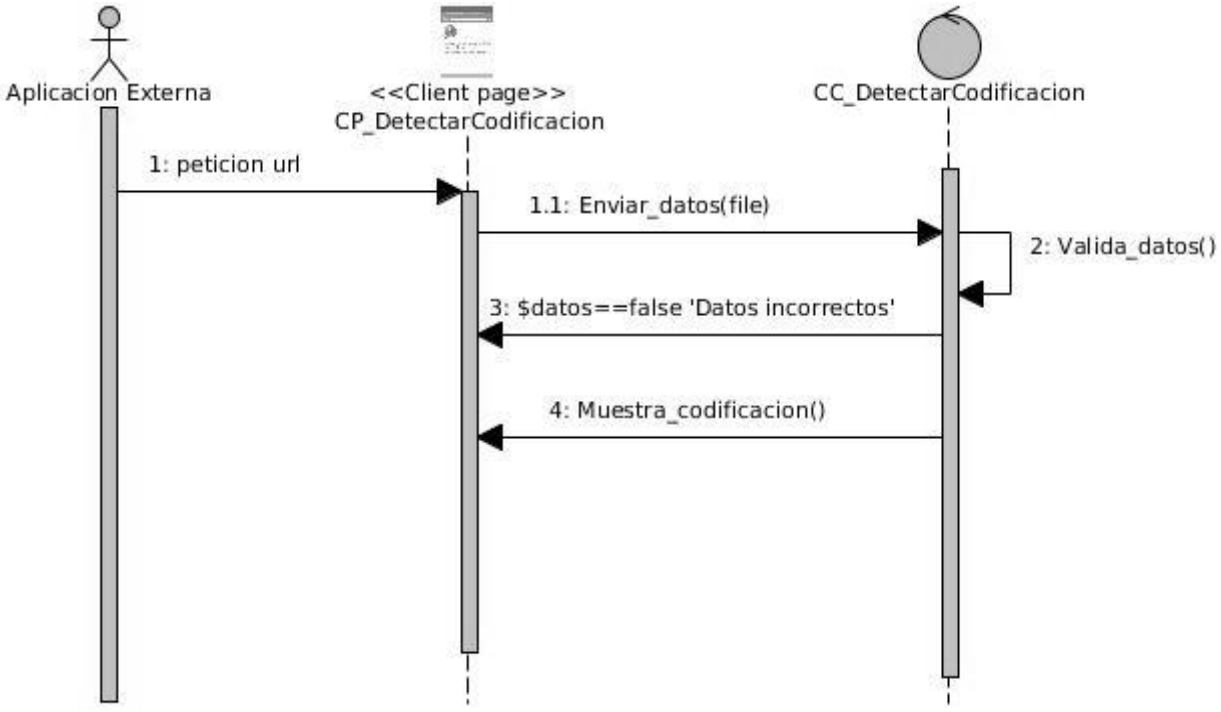

Figura 10 Diagrama de secuencia. Detectar codificación

2.10 Diseño de la base de datos

Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada, puesto que un diseño correcto es esencial para lograr los objetivos fijados para esta. Conjuntamente desempeñan un papel crucial en casi todas las áreas del desarrollo de *software*.

La capa de servicios *web* hace uso de una base de datos ya que la información a persistir será la obtenida del tráfico entre las aplicaciones externas y la capa de servicio *web* además todo el control de acceso a esta por los usuarios permitidos. A continuación se muestra el modelo físico de datos donde se determinaron las entidades (tablas), se especificaron los atributos (campos) y se relacionaron las tablas. Para ver la descripción de las tablas remitirse al Anexo 8.

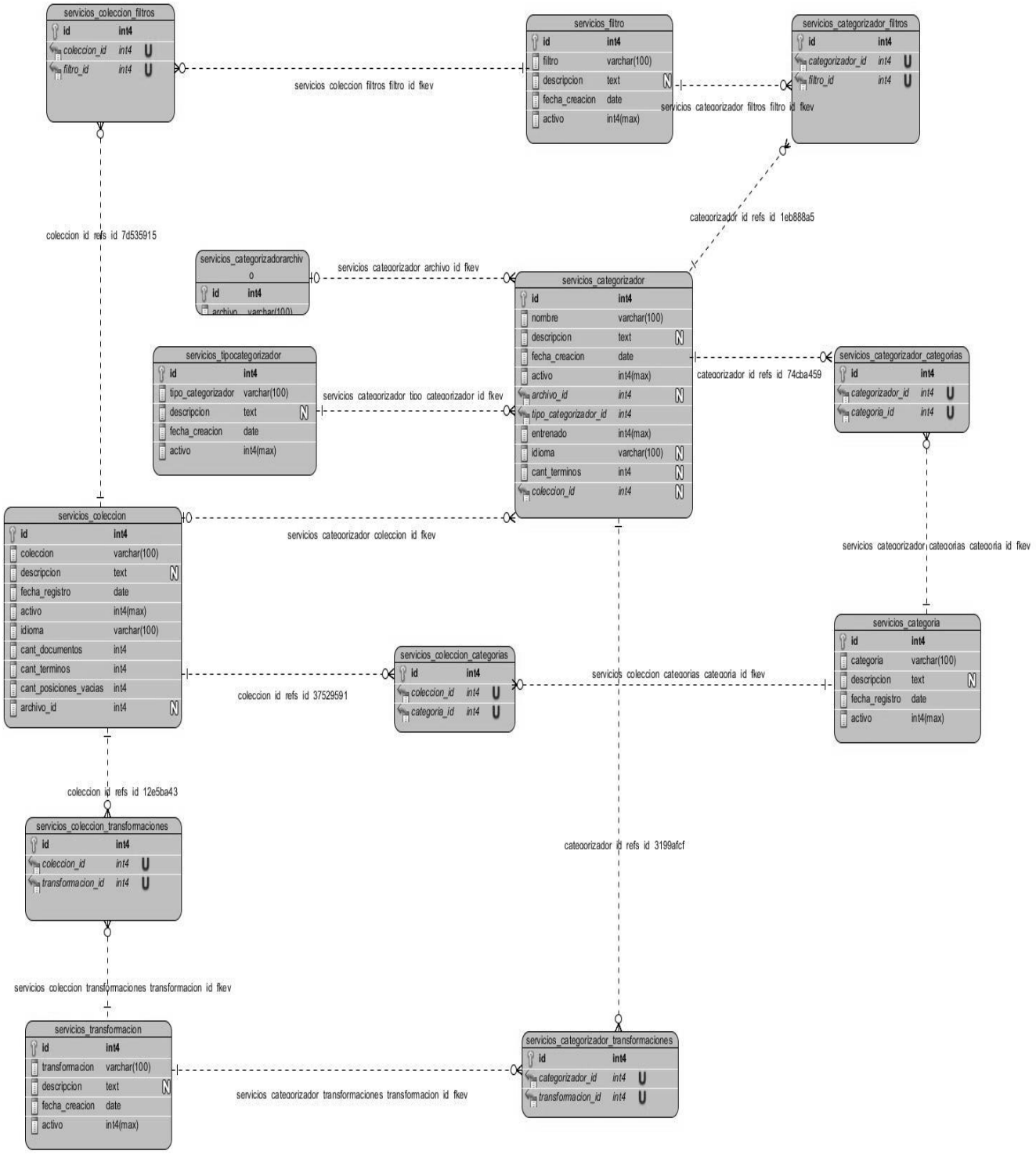

Figura 11 Modelo físico de datos (Parte 1)

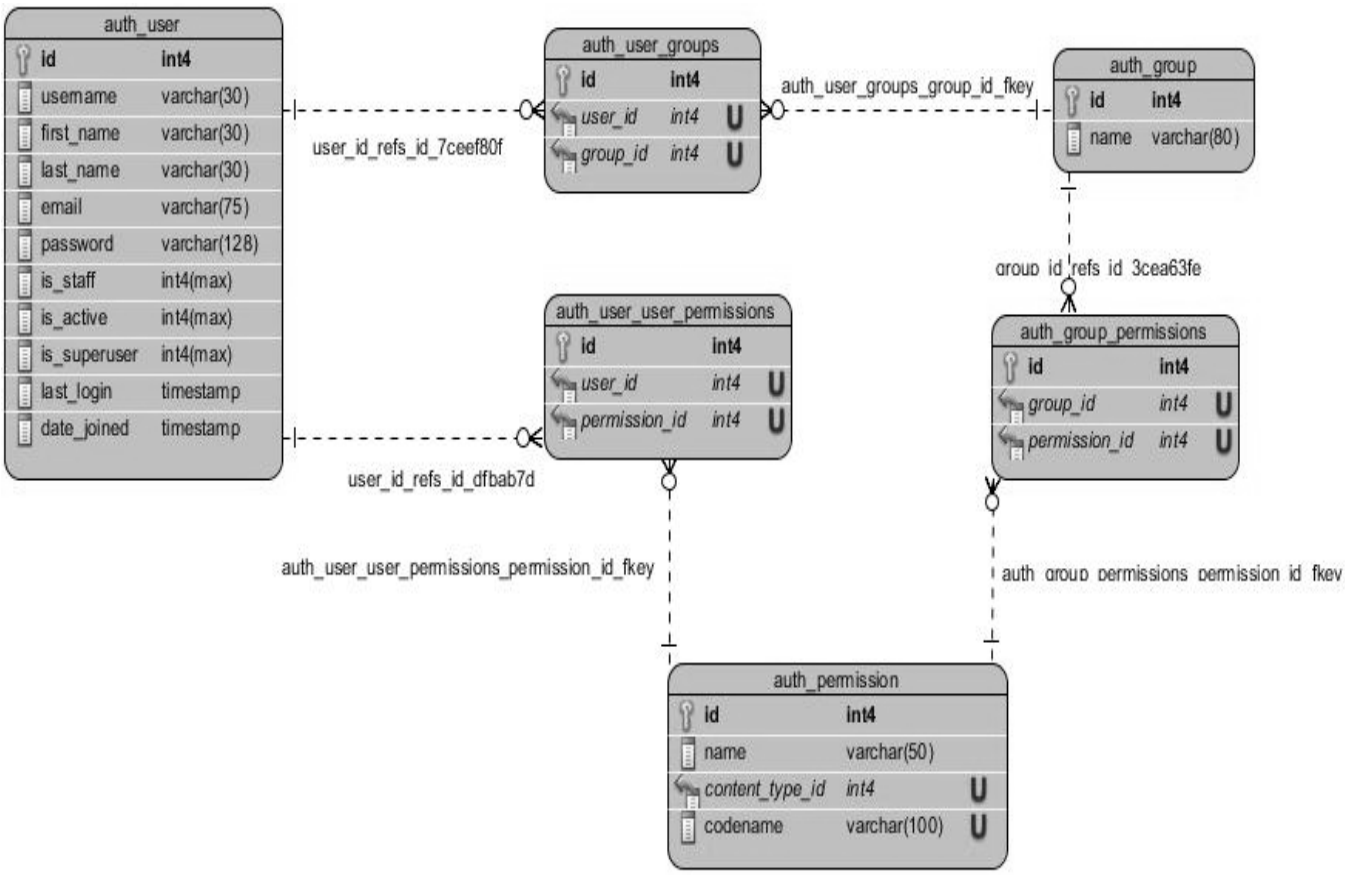

Figura 12 Modelo físico de datos (Parte 2)

2.11 Modelo de despliegue

Describe la distribución física del sistema entre los nodos de computación sobre los que se va a instalar el sistema. Se utiliza un nodo para identificar cualquier servidor, terminal de trabajo u otro *hardware* que se utiliza para desplegar componentes en el ambiente de producción[57].

El diagrama de despliegue de la capa de servicios *web* para el categorizador de texto de MOCIC que se muestra en la siguiente figura está estructurado de la siguiente forma: la aplicación externa se conecta al servidor *web* mediante el protocolo de comunicación HTTP y este a su vez se conecta al servidor de base de datos a través de la familia de protocolos TCP/IP (Protocolo de Control de Transmisión/Protocolo de Internet).

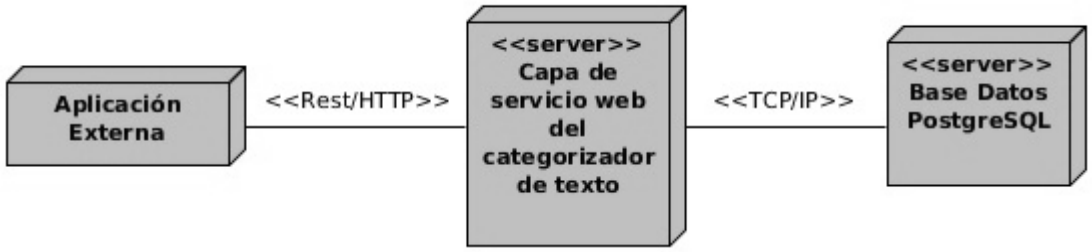

Figura 13 Diagrama de despliegue

2.12 Conclusiones Parciales

- La representación del modelo del dominio sirvió para comprender el funcionamiento entre las diferentes clases que integran el sistema.
- Las técnicas de captura de requisitos utilizadas facilitaron la definición y comprensión de las funcionalidades del sistema, lo que permitió realizar el levantamiento de los requisitos funcionales y no funcionales en su totalidad.
- Con la representación y la especificación de los casos de uso se documentó toda la información relativa al dominio de la capa de servicios.
- La descripción de los estándares de la arquitectura y el diseño permitió que se cuente con los elementos necesarios para la implementación de la capa de servicios *web*.

Capítulo 3: Implementación y prueba

3.1 Introducción

Partiendo de los artefactos resultantes del diseño de la capa de servicio, en el presente capítulo se describirá el diagrama de componentes que modela la capa de servicios *web* para el categorizador de texto de MOCIC según la definición de fases e iteraciones de la metodología OpenUP y se justificará el estándar de codificación utilizado.

Durante el período de implementación y pruebas se elaboran los elementos de código. En este proceso se generan todos los artefactos de esta etapa de desarrollo y se valida el correcto funcionamiento de la aplicación.

3.2 Diagrama de componentes

Un diagrama de componentes se representa como un grafo de componentes de *software* unidos por medio de relaciones de dependencia (compilación, ejecución). El diagrama de componentes representa los componentes de *software* (componentes de código fuente, componentes del código binario y componentes ejecutables, entre otros) que integran un sistema. Sus relaciones describen qué componentes utilizan los servicios ofrecidos por otros componentes, y se utilizan para modelar la vista estática de un sistema[58]. La siguiente figura muestra el diagrama de componentes de la capa de servicios *web* del categorizador de texto de MOCIC. Se puede observar la estructura de los paquetes y la distribución de los componentes dentro de estos. De forma general cada componente del paquete Servicio tendrá a la par un componente dentro del paquete Control que se relacionará con su respectivo componente dentro del paquete Modelo, todos ellos a su vez utilizarán los componentes del paquete Librería en dependencia de sus necesidades.

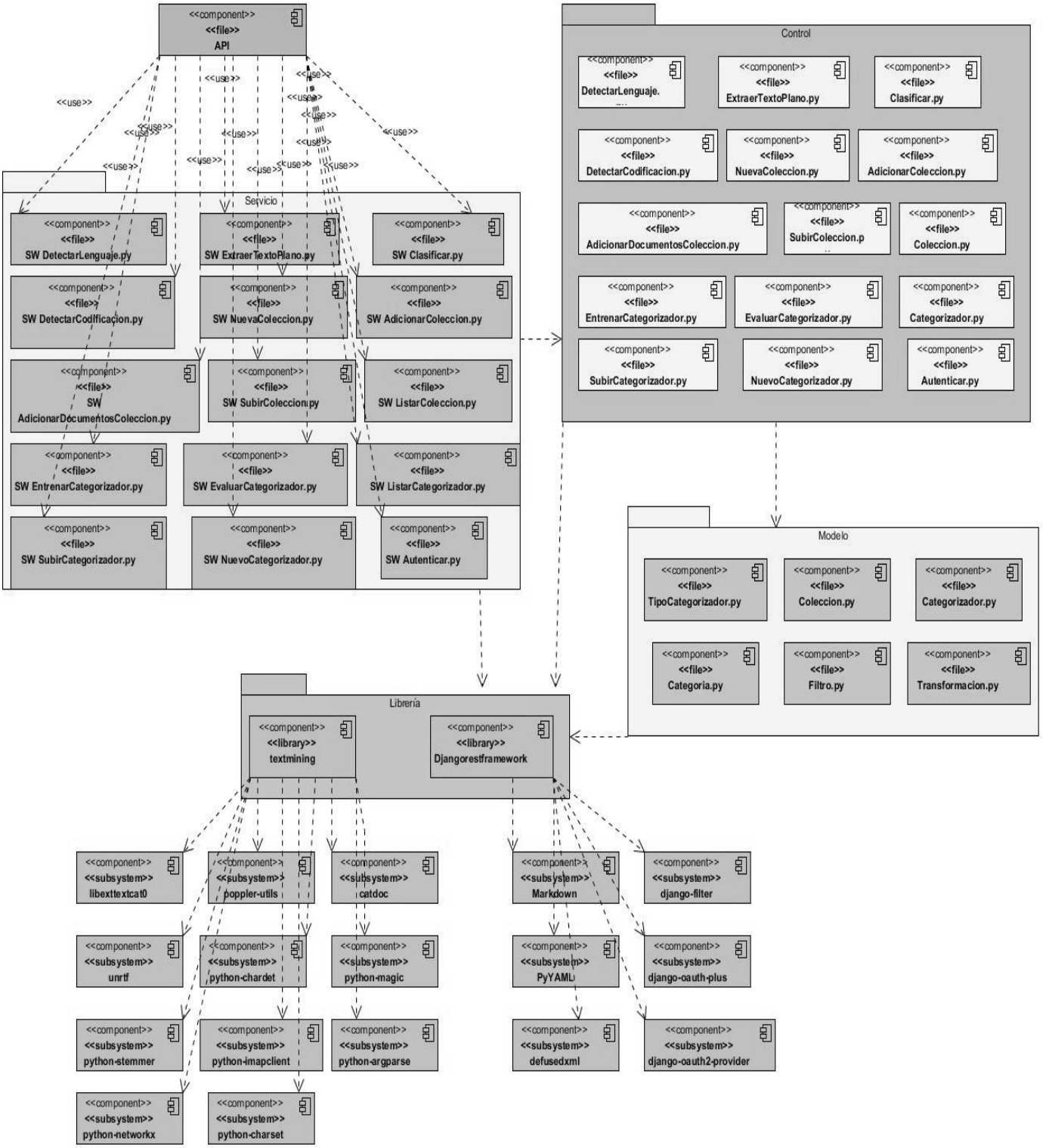

Figura 14 Diagrama de componentes

3.2.1 Descripción del diagrama de componentes

Componente API: interfaz o librería que contiene las funciones necesarias para realizar conexiones exitosas con la capa proveedora de servicios del categorizador de texto. Punto de enlace entre las aplicaciones externas y el categorizador de texto.

Paquete Servicio: dentro de este paquete se encuentra cada uno de los servicios de la capa del categorizador de texto. Estos servicios se muestran mediante las vistas que son las interfaces que revelan los formularios y elementos para introducir la información necesaria para realizar las peticiones.

Paquete Control: dentro de este paquete se encuentra cada una de las clases que controlan todo el flujo del proceso de categorización de texto encargándose de la lógica del negocio.

Paquete Modelo: dentro de este paquete se encuentra cada una de las clases que permiten la interacción con la base de datos creando una capa de abstracción de datos.

Paquete Librería: dentro de este paquete se encuentran las librerías *Textmining* y *DjangoRestFramework*. Dentro de la librería *Textmining* se encuentran los subsistemas que contienen todos los métodos necesarios para realizar las funcionalidades del categorizador de texto y dentro de la librería *DjangoRestFramework* se encuentran todos los subsistemas necesarios para la realización de la capa de servicios *web* para el categorizador de texto de MOCIC.

3.3 Estándares de código

Estándar de código o convención de código, es un estilo de programación que describe convenciones para escribir código fuente en cierto lenguaje de programación.

Para la implementación de la capa de servicios *web* se seleccionó el lenguaje de programación Python por lo que la propuesta de solución cumple con las convenciones de código definidas para el lenguaje Python, descritas en el documento “Pautas de Codificación”[59].

3.4 Tratamiento de errores

La presencia de errores durante el desarrollo de *software* es bastante común, pero lo importante no es equivocarse sino poder detectar los errores a tiempo, debido a que cuanto más tarde se descubran más costosos será corregirlos. El tratamiento de errores debe estar enfocado tanto a los errores que se producen a la hora de los usuarios introducir datos, como a los errores que puedan ser generados por el comportamiento incorrecto de los componentes internos. En la capa de servicios *web* el seguimiento que se le da a los datos introducidos por los usuarios es mediante validaciones que se les asignan a los campos de entrada de datos, garantizando que se muestre el tipo de error sobre el campo donde se introdujeron datos incorrectos. Para darle seguimiento a los errores internos de la capa de servicios hay que tener en cuenta el lenguaje de programación, y establecer los estándares de codificación correctos.

Una ventaja de darle seguimiento a los errores es poder señalar y separar el tratamiento de los errores mediante excepciones y poder cumplir una respuesta a errores particulares.

A continuación se muestra una tabla que evidencia el tratamiento de errores.

Error	Causa	Respuesta del sistema
Por favor introduzca un usuario y clave adecuado para una cuenta de personal. Fíjese que ambos campos son sensibles a mayúsculas.	El usuario introduce los parámetros usuario y contraseña incorrectos.	El sistema notifica al usuario que los datos introducidos son erróneos.
No tiene permiso para ejecutar esta acción.	El usuario hace una petición a un servicio que solo lo puede realizar el administrador.	El sistema notifica que el usuario no tiene permisos para ejecutar esa acción.
El archivo está defectuoso.	El usuario carga un archivo defectuoso para categorizar texto, o detectar idioma, o detectar codificación, o extraer texto plano.	El sistema notifica que el archivo está defectuoso por lo que no se puede realizar la operación.
Se deben seleccionar elementos para poder realizar acciones sobre estos. No se han modificado elementos.	El usuario con permiso de administrador realiza la acción de eliminar un categorizador y no selecciona ningún elemento.	El sistema notifica que no ha sido seleccionado ningún elemento.

Tabla 20 Respuesta del sistema a posibles errores

3.5 Seguridad

Con el objetivo de garantizar la seguridad de la capa de servicios *web* así como lograr la integridad, disponibilidad y confiabilidad de los datos, se trabajó con las sesiones de usuarios. Los administradores son los encargados de agregar a los usuarios que deseen utilizar los servicios del categorizador de texto de MOCIC y darles los permisos necesarios para acceder a las diferentes funcionalidades. El usuario para realizar una petición a un servicio, debe primero autenticarse. Cada usuario que se autentique en la capa de servicios tendrá una sesión y sus respectivos permisos, de esta forma se evita que estos puedan realizar operaciones a las que no tengan acceso.

Por otra parte los administradores pueden gestionar grupos de usuarios dándole los permisos necesarios a estos para acceder a los servicios que necesiten utilizar.

3.6 Pruebas de Software

Las pruebas de *software* son los procesos que permiten verificar y revelar la calidad de un sistema informático antes de su puesta en marcha. Permiten comprobar el grado de cumplimiento de las

especificaciones iniciales del sistema y no pueden asegurar la ausencia de defectos, sólo pueden demostrar que existen defectos en el *software*[60].

Pruebas del sistema: el objetivo en este nivel de pruebas es verificar el ingreso, procesamiento y recuperación apropiada de datos, así como la implementación adecuada de las reglas del negocio. Para la realización de las pruebas del sistema se usa el método de pruebas de caja negra, que son pruebas funcionales sin acceso al código fuente de las aplicaciones, donde se trabaja con entradas y salidas.

- **Pruebas funcionales:** son el proceso de probar una entrega del sistema que será distribuida a los clientes. El objetivo de estas pruebas es comprobar que el sistema satisface sus requerimientos y para esto deben demostrar la entrega de la funcionalidad especificada, rendimiento y confiabilidad y que no falla durante su uso normal. Para identificar los valores de los datos de entrada se utilizó la técnica de particiones o clases de equivalencias, esta técnica consiste en ejecutar el flujo básico de las funcionalidades utilizando datos válidos e inválidos para verificar que los resultados esperados ocurren cuando se utiliza un dato válido, que los mensajes de error o de advertencia aparecen en el momento adecuado cuando se utiliza un dato inválido. Las clases de equivalencia se identifican examinando cada condición de entrada y dividiéndola en dos o más grupos. Se definen dos tipos de clases de equivalencia: clases válidas, que son entradas válidas al sistema y clases no válidas, que son valores de entrada erróneos[60].

Las pruebas de caja negra pretenden encontrar diferentes errores como; funciones incorrectas o ausentes, errores en la interfaz, errores en estructuras de datos o en accesos a la bases de datos, errores de rendimiento, errores de inicialización y de terminación.

Pruebas de Rendimiento de software

La prueba de rendimiento de *software* se enfoca en la capacidad de recibir peticiones mediante la utilización de alguna herramienta, verificando cuantas peticiones pueda sostener el sistema sin que este se vea afectado, así como la velocidad de respuesta del mismo. Las pruebas de rendimiento para la capa de servicios *web* serán realizadas utilizando como técnica JMeter, que es una aplicación de escritorio y de código abierto, diseñada para realizar pruebas de carga, medir el comportamiento de las pruebas y medir el rendimiento de un sistema. JMeter también puede ser utilizado como una herramienta de prueba de carga para analizar y medir el desempeño de una variedad de servicios, con énfasis en aplicaciones *web*.

3.6.1 Modelo de pruebas

El modelo de pruebas, comprende una colección de casos de prueba, procedimientos y componentes que describen cómo se prueban los componentes ejecutables en el modelo de implementación, con pruebas de

sistema, ya que las pruebas son la actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos y los resultados son observados y registrados.

El desarrollo del *software* implica en sí, errores que pueden ocurrir desde el primer momento del proceso en el que los requerimientos pueden estar especificados de forma errónea, así como en los posteriores pasos del diseño e implementación. Es por esto, que se hace necesario contar con una actividad que garantice la calidad.

3.6.1.1 Pruebas Funcionales

A continuación se muestra el diseño de casos de prueba correspondiente a la funcionalidad “Categorizar Texto” y “Subir Categorizador”. Los restantes diseños de casos de pruebas pueden ser consultados en el Anexo 9 Tabla (32-39).

Caso de Uso	CU_5 Categorizar texto
Requisito	RF5
Condición	Debe estar autenticado.
Referencia	RF5

Tabla 21 Caso de Prueba. Categorizar texto

Descripción de variables

Parámetro	Valor	Descripción
Categorizador	V	Indica el categorizador que se utilizará para clasificar el documento.
Examinar	V	Indica que el documento que se clasificará debe ser cargado.

Tabla 22 Descripción de variables. Caso de prueba. Categorizar Texto.

Escenario	Descripción	V_Categorizador	V_Examinar	Respuesta del sistema	Flujo central
EC 1.1 Categorizar texto correctamente.	Mediante este escenario se carga el documento que se quiere categorizar.	V Nombre del categorizador	V Documento.doc	El sistema procesa la información del documento, lo categoriza y envía un JSON con el resultado de	-El usuario una vez autenticado hace una petición al servicio Categorizar Texto. -El sistema muestra una vista con un campo “Examinar” para cargar el documento. -El usuario carga el documento y acepta realizar la acción dándole clic al botón “Enviar”.

				la petición.	-El sistema envía un documento JSON con la categoría del documento.
EC 1.2 No existe documento para categorizar.	Mediante este escenario no se carga ningún documento para realizar la acción.	V	I	El sistema muestra el mensaje “No se ha seleccionado ningún archivo”.	-El usuario una vez autenticado hace una petición al servicio Categorizar Texto. - El sistema muestra una vista con un campo “Examinar” para cargar el documento. -El usuario no carga documentos y acepta realizar la acción dando clic al botón “Enviar”. -El sistema envía un mensaje “No se ha seleccionado ningún archivo”.
EC 1.3 Error en la categorización de texto.	Mediante este escenario se carga un documento defectuoso.	V	V	El sistema procesa la información, y al ser el documento defectuoso, informa que no puede categorizar el texto del documento a través del mensaje “El archivo está defectuoso”.	-El usuario una vez autenticado hace una petición al servicio Categorizar Texto. - El sistema muestra una vista con un campo “Examinar” para cargar el documento. -El usuario carga un documento defectuoso y acepta realizar la acción dando clic al botón “Enviar”. -El sistema envía un mensaje “El archivo está defectuoso.”
		Nombre del categorizador	Documento.doc		

EC 1.4 Error en la categorización de texto.	Mediante este escenario no se selecciona un categorizador.	I Nombre del categorizador	V Documento.doc	El sistema procesa la información, y al no haberse cargado el categorizador, informa que no puede categorizar el documento.	El usuario una vez autenticado hace una petición al servicio Categorizar Texto. - El sistema muestra una vista con un campo “Examinar” para cargar el documento. -El usuario carga el documento y no selecciona el categorizador, acepta realizar la acción dando clic al botón “Enviar”. -El sistema envía un mensaje “No se ha seleccionado el categorizador.”

Tabla 23 Diseño de casos de prueba. Categorizar Texto

Caso de Uso	CU_5 Categorizar texto
Requisito	RF5
Condición	Debe estar autenticado.
Referencia	RF5

Tabla 24 Caso de Prueba. Subir Categorizador

Descripción de Variables

Parámetro	Valor	Descripción
Examinar	V	Indica que el documento que se clasificará debe ser cargado.

Tabla 25 Descripción de variables. Caso de prueba. Subir Categorizador

Escenario	Descripción	V_Examinar	Respuesta del sistema	Flujo central
EC 1.1 Subir Categorizador correctamente.	Mediante este escenario se carga el categorizador.	V Categorizador.svn	El sistema procesa la información y agrega el categorizador a la base de datos.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Subir Categorizador. -El sistema muestra una vista con un campo “Examinar” para cargar el categorizador. -El usuario carga el categorizador y

				da clic en el botón “Enviar”. -El sistema guarda el categorizador en la base de datos correctamente.
EC 1.2 Error para subir un categorizador.	Mediante este escenario se carga el categorizador con una extensión errónea.	V Categorizador.txt	El sistema procesa la información, y no carga el categorizador si no tiene la extensión .svn.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Subir Categorizador. -El sistema muestra una vista con un campo “Examinar” para cargar el categorizador. -El usuario carga el categorizador con la extensión errónea y da clic en el botón “Enviar”. -El sistema muestra un mensaje de error.

Tabla 26 Diseño de casos de prueba. Subir Categorizador

3.6.1.2 Pruebas de rendimiento

Las pruebas de Rendimiento de *software* fueron realizadas para probar la escalabilidad del sistema y cuanto estrés puede soportar el mismo con el uso de la herramienta JMeter, la cual ofreció importantes datos de estrés que pudo soportar el sistema.

El significado de algunos campos se describe a continuación:

Label o etiqueta: el nombre de la petición HTTP.

 # Muestras: el número de muestras para cada petición.

Media: el tiempo medio transcurrido para un conjunto de resultados.

Mín: el mínimo tiempo transcurrido para las muestras de la petición dada.

Máx: el máximo tiempo transcurrido para las muestras de la petición dada.

% Error: porcentaje de las peticiones con errores.

Rendimiento: rendimiento medido en peticiones por segundo/minuto/hora.

Kb/sec: rendimiento medido en kilobytes por segundo.

Label	# Muestras	Media	Mediana	Linea de 90%
Login Page	100	220	220	233
Servicios	100	168	166	214
Servicio Clasificar	100	223	222	239
TOTAL	300	204	217	233

Figura 15 Prueba de rendimiento (Parte 1)

Mín	Máx	% Error	Rendimiento	Kb/sec
162	272	0,00%	2,9/sec	10,6
110	236	0,00%	2,9/sec	10,7
126	276	0,00%	2,9/sec	10,7
110	276	0,00%	8,5/sec	31,7

Figura 16 Prueba de rendimiento (Parte 2)

3.6.2 Resultado de las pruebas

Para validar la capa de servicios se realizaron pruebas de caja negra en 3 iteraciones, obteniéndose los siguientes resultados en cada una de las iteraciones realizadas:

- En la primera iteración se obtuvieron 10 no conformidades. Dentro de estas se encuentra: la no visualización de los mensajes de error al cargar archivos defectuosos. Estas no conformidades fueron corregidas.
- En la segunda iteración se detectaron 6 no conformidades. Dentro de estas se encuentra: la no validación para la entrada de datos incorrectos en algunos campos de entrada de datos. Estas no conformidades también fueron rectificadas.
- En una tercera iteración la capa de servicios estaba libre de no conformidades, cumpliendo con los requisitos funcionales especificados. De esta forma el desarrollo de las pruebas funcionales de los casos de uso muestra una visión de la calidad del *software*.

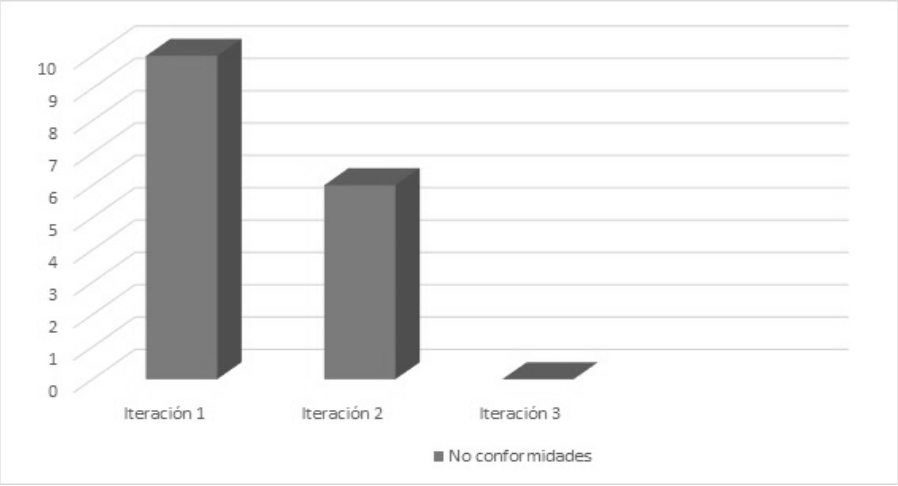

Figura 17 Resultado de las pruebas funcionales

En la figura 17 se observa un escenario simulado de 100 peticiones por cada etiqueta. Dichas peticiones fueron realizadas en un intervalo, entre una y otra, de 1 segundos. La prueba se realizó con un porciento de error de 0,00% y demostró que en esta situación la capa de servicios presenta un rendimiento de 8,5

peticiones por segundo; ambas conclusiones se interpretan de los campos %Error y Rendimiento respectivamente.

3.7 Conclusiones Parciales

- La representación del diagrama de componentes del sistema permitió mostrar las relaciones de dependencia entre los componentes del sistema, así como las librerías que utiliza, para una mejor comprensión del funcionamiento de la capa de servicios y las tecnologías que la componen.
- El estudio preliminar de las pruebas de *software* en general, permitió seleccionar las pruebas funcionales y de rendimiento para mostrar el correcto funcionamiento y la escalabilidad del sistema, arrojando resultados satisfactorios y demostrando el correcto desarrollo de la capa de servicios *web*.
- El sistema cumple con todos los requerimientos establecidos para su implantación y cumple con su objetivo de utilizar las funcionalidades del categorizador de texto de MOCIC.

Conclusiones

Una vez finalizado el proceso de desarrollo de *software* se demostró que la implementación de una capa de servicios que permite, además de las funcionalidades básicas del categorizador de texto de MOCIC, la gestión y configuración de la capa de servicios donde se logra la comunicación de aplicaciones externas con los servicios que ofrece el categorizador de texto. Esto es útil para aplicaciones que gestionen contenido ya que está capa da la posibilidad de realizar peticiones desde cualquier entorno de desarrollo. La solución desarrollada soluciona la problemática que lo originó mediante el cumplimiento de los objetivos trazados. De la investigación realizada se concluye lo siguiente:

- El estudio del estado del arte de los sistemas homólogos al Motor de Clasificación Inteligente de Contenido permitió conocer características comunes, servicios que brindan y funcionalidades relevantes para el desarrollo de la capa de servicios *web*.
- El análisis de las diferentes metodologías y herramientas posibilitó la selección adecuada de la base tecnológica para el desarrollo de la capa de servicios.
- El establecimiento de los requisitos funcionales y no funcionales permitió definir desde el principio las metas a alcanzar con la implementación de la capa de servicios, para que el subsistema categorizador de texto de MOCIC tuviese un mayor uso y fuese accesible a todos los usuarios.
- La definición de una arquitectura de tres capas: Acceso, Procesamiento y Datos; permitió la separación de las responsabilidades de cada componente de la capa de servicios lográndose una correcta interacción entre ellos.
- Definir los patrones de diseño para la codificación, permitió organizar la etapa de implementación ahorrando tiempo de desarrollo.
- La realización de las pruebas funcionales arrojó resultados positivos validando el correcto funcionamiento de la capa de servicios.
- Las pruebas de rendimiento permitieron demostrar que la capa de servicios funciona correctamente en escenarios de carga y estrés.

Debido a lo anteriormente planteado se puede decir que se cumplió el objetivo principal de la investigación, obteniéndose los resultados esperados.

Recomendaciones

Luego de concluida la investigación y de haberse obtenido finalmente la capa de servicios *web* para el categorizador de texto de MOCIC, se recomienda para futuras versiones:

- Agregar nuevas funcionalidades a la capa de servicios *web* para mejorar el proceso de categorización de textos.
- Agregar nuevas funcionalidades a la interfaz de administración para facilitar la configuración de la capa de servicios.

Referencias bibliográficas

[1] Carlos Andrés Morales Machuca, «Estado del Arte: Servicios Web», 2010.

[2] Luis Enrique Alvarado, «ESTADÍSTICAS ALARMANTES DE LA PORNOGRAFÍA», 2009. [En línea]. Disponible en: http://verdadyluzhoy.blogspot.mx/2009/03/estadisticas-alarmantes-de-la.html. [Accedido: 09-abr-2014].

[3] Dr. Rolando Alfredo Hernández León, Dra. Sayda Coello González, *EL PROCESO DE INVESTIGACIÓN CIENTÍFICA*. 2011.

[4] Dr. C. María Elena Guardo García, «LOS COMPONENTES DEL DISEÑO TEÓRICO DE LA INVESTIGACIÓN CIENTÍFICA. UNA REFLEXIÓN PRAXIOLÓGICA.», p. 26, 2009.

[5] Dr. Roberto Hernández Sampieri, Dr. Carlos Fernández Collado, Dra. Pilar Baptista Lucio, *Metodología de la investigación*, vol. Cuarta Edición. 2006.

[6] W3C: World Wide Web Consortium, «Guía breve de Servicios Web», 2006.

[7] Tomas A Powell, *Diseño de sitios web*. 2004.

[8] Luis Castro, «Guía básica sobre qué es HTTP y HTTPS». [En línea]. Disponible en: http://aprenderinternet.about.com/od/ConceptosBasico/a/Que-Es-Http.htm. [Accedido: 08-feb-2014].

[9] «Concepto de URL». [En línea]. Disponible en: http://www.aulapc.es/internet_lared_url.html. [Accedido: 08-feb-2014].

[10] Carles Mateu, *Desarrollo de aplicaciones web*. Fundación Universitaria Oberta de Catalunya, 2004.

[11] «Guía Breve de Servicios Web». [En línea]. Disponible en: http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb. [Accedido: 22-ene-2014].

[12] «HTTP - Hypertext Transfer Protocol Overview». [En línea]. Disponible en: http://www.w3.org/Protocols/. [Accedido: 22-ene-2014].

[13] M.Sc. José Yáñez Menéndez, *Redes, Comunicaciones y el Laboratorio de Informática*. 2002.

[14] José María Barceló Ordinas, Jordi Íñigo Griera, *Redes de computadores*. 2004.

[15] Rafael Navarro Marset, «REST vs Web Services». jul-2006.

[16] Terry Dawson, *Guía de Administración de Redes con Linux*. 2002.

[17] «XML-RPC». [En línea]. Disponible en: http://xmlrpc.scripting.com/default.html. [Accedido: 22-ene-2014].

[18] «SOAP (Simple Object Access Protocol)». [En línea]. Disponible en: http://www.desarrolloweb.com/articulos/1557.php. [Accedido: 22-ene-2014].

Referencias bibliográficas

[19] Lourdes Tajes Martínez, «WSDL (Web Services Description Language) Construcción de servicios *web*», 2008.

[20] «Directorios de servicios Web XML». [En línea]. Disponible en: http://msdn.microsoft.com/es-es/library/7e29kfs9(v=vs.80).aspx. [Accedido: 23-ene-2014].

[21] Camilo Alberto López Aravena, «DISEÑO Y CONSTRUCCIÓN DE UNA PLATAFORMA DE CLASIFICACIÓN DE TEXTO BASADA EN TEXTMINING APLICADA SOBRE UNA RED DE BLOGS PARA BETAZETA NETWORKS S.A.», 2012.

[22] Sholom M. Weiss, Nitin Indurkhya, Tong Fred J. DamerauZhang,, *Text Mining*. 2005.

[23] Rolando González Herryman, Raúl AmambayTarajanoPérez, «Sistema de Categorización Manual de URLs», 2010.

[24] Marcelo Mendoza, Ivette Ortiz, Víctor Rojas, «Revista signos - Categorización de texto en bases documentales a partir de modelos computacionales livianos». [En línea]. Disponible en: http://www.scielo.cl/scielo.php?pid=S0718-09342011000300004&script=sci_arttext. [Accedido: 23-ene-2014].

[25] «Página oficial de Bitext». [En línea]. Disponible en: http://www.bitext.com/es/. [Accedido: 23-ene-2014].

[26] «SOFFIK - Servicios Outsourcing de Contenidos y Gestión de Usuarios». [En línea]. Disponible en: http://www.soffik.com/. [Accedido: 23-ene-2014].

[27] Guillermo de Jorge Botana, Ricardo Olmos Albacete, Alejandro Barroso, «Categorizador textual para enrutadores de llamadas, asistentes virtuales y clasificación de correo-e.», .

[28] «Plataforma para el procesamiento de información». [En línea]. Disponible en: http://www.datys.cu/wpinfproducto.aspx?61. [Accedido: 23-ene-2014].

[29] «Análisis Inteligente de archivos». [En línea]. Disponible en: http://www.datys.cu/wpinfproducto.aspx?64. [Accedido: 23-ene-2014].

[30] «Sistema para la gestión inteligente de noticias». [En línea]. Disponible en: http://www.datys.cu/wpinfproducto.aspx?66. [Accedido: 23-ene-2014].

[31] «Python Programming Language – Official Website». [En línea]. Disponible en: http://www.python.org/. [Accedido: 23-ene-2014].

[32] Adrian Holovaty, Jacob Kaplan-Moss, *El libro de Django*. 2008.

Referencias bibliográficas

[33] «The Web framework for perfectionists with deadlines | Django». [En línea]. Disponible en: https://www.djangoproject.com/. [Accedido: 23-ene-2014].

[34] «JSON». [En línea]. Disponible en: http://www.json.org/json-es.html. [Accedido: 06-abr-2014].

[35] «IBM - Rational Unified Modeling Language», 19-feb-2014. [En línea]. Disponible en: http://www-01.ibm.com/software/rational/uml/. [Accedido: 03-jun-2014].

[36] «Eclipse - The Eclipse Foundation open source community website.» [En línea]. Disponible en: http://www.eclipse.org/. [Accedido: 08-feb-2014].

[37] «FAQ What is Eclipse? -Eclipsepedia». [En línea]. Disponible en: http://wiki.eclipse.org/FAQ_What_is_Eclipse%3F. [Accedido: 08-feb-2014].

[38] Raúl González Duque, *Python para todos*. 2008.

[39] «Visual Paradigm», *Visual Paradigm*. [En línea]. Disponible en: http://www.visual-paradigm.com/aboutus/. [Accedido: 03-jun-2014].

[40] «Visual Paradigm para UML - Programación - herramienta para desarrollo de aplicaciones utilizando modelado UML». [En línea]. Disponible en: http://www.software.com.ar/visual-paradigm-para-uml.html. [Accedido: 23-ene-2014].

[41] Regina Obe, Leo Hsu, *PostgreSQL: Up and Running*. 2012.

[42] «NginxEs - Nginx Community». [En línea]. Disponible en: http://wiki.nginx.org/NginxEs. [Accedido: 05-abr-2014].

[43] Santiago Ríos Salgado, Ing. Cecilia Hinojosa Raza, Ing. Ramiro Delgado Rodríguez, «APLICACIÓN DE LA METODOLOGIA OPENUP EN EL DESARROLLO DEL SISTEMA DE DIFUSIÓN DE GESTIÓN DEL CONOCIMIENTO DE LA ESPE», 2013.

[44] Oficina Asesora de Sistemas, «Guía Rápida Proceso de Desarrollo OPENUP/OAS». 2011.

[45] Michael Yang, «OpenUP». [En línea]. Disponible en: http://epf.eclipse.org/wikis/openup/. [Accedido: 06-feb-2014].

[46] Craig Larman, «MODELO DEL DOMINIO in UML y Patrones. 2ª Edición», p. 23, 2003.

[47] Roger S. Pressman, *Ingeniería de requisitos, in Ingeniería del Software, Un Enfoque Práctico*, 5ta edición. .

[48] Amador Durán Toro and B.B. Jiménez, *Metodología para la Elicitación de Requisitos de Sistemas Software*. Sevilla, 2000.

[49] Rojas, M.C.J.c.O, *Técnicas para la obtención de requerimientos*. Secretaría de Educación Pública, 2012.

[50] Erika Camacho, *ARQUITECTURAS DE SOFTWARE GUÍA DE ESTUDIO*. 2004.

[51] Microsoft Corporation, «La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real», p. 24, 2006.

[52] Informáticos, D.d.L.y.S, «Patrones de Asignación de Responsabilidades (GRASP)», p. 27.

[53] José David Parra, *Guía de Patrones, Prácticas y Arquitectura .NET*. 2008.

[54] Unidad Docente de Ingeniería del Software, *Patrones del «Gang of Four»*. Facultad de informática - Universidad Politécnica de Madrid, 2009.

[55] Manuel Antonio Novoa Proenza, Yasmani Joaquin Álvarez Gómez, «Capa de servicios *web* para el motor de búsqueda Orión», Universidad de las Ciencias Informáticas, 2012.

[56] Stephen A. White, Derek Miers, *Guía de Referencia y Modelado BPMN*. 2009.

[57] Carlos Alberto torres Ayazo, «Diagrama de Despliegue UML», *prezi.com*, 2013. [En línea]. Disponible en: http://prezi.com/svxjuygcrpse/diagrama-de-despliegue-uml/. [Accedido: 07-may-2014].

[58] Benet Campderrich Falgueras, *Ingeniería de software*, Primera edición. Editorial UOC, 2003.

[59] Yendry Machado García, «Pautas de codificación», p. 9, mar. 2013.

[60] CIBERTEC, *Pruebas de Software*. 2012.

Bibliografía consultada

1. Marset, Rafael Navarro. *REST vs Web Services.* 2007.
2. Comunicación, Instituto Nacional de Tecnologías de la. *GUÍA PRÁCTICA DE GESTIÓN DE REQUISITOS.* 2008.
3. Steven Bird, Ewan Klein, Eduard Loper. *Natural Language Processing with Python.* 2009.
4. Grant S. Ingersoll, Thomas S. Morton, Andrew L. Farris. *Taming Text.* 2013.
5. Montero, Sergio Infante. *Curso Django, El framework para perfeccionistas con deadlines.* s.l. : Eugenia Tobar, 2012.
6. Adrian Holovaty, Jacob Kaplan-Moss. *El libro de Django.* s.l. : Jeremy Dunck, 2008.
7. Lane, Kin. *The Building Blocks of a Successful API.* 2012 : s.n.
8. Daniel Jacobson, Greg Brail, Dan Woods. *APIs: A Strategy Guide.* 2011.
9. Allamaraju, Subbu. *RESTful Web Services Cookbook.* s.l. : Sumita Mukherji, 2010.
10. María Dolores Lozano, Isidro Ramos Salavert. Evolución y Perspectivas en el Desarrollo de Software: Nuevas Tendencias Orientadas a Objetos. *Evolución y Perspectivas en el Desarrollo de Software: Nuevas Tendencias Orientadas a Objetos.* Valencia : s.n, 2011.
11. Ward, Greg. *Distribución de módulos Python.* 2000.
12. *Estado del Arte: Servicios Web.* Machuca, Carlos Andrés Morales. 2008.
13. Zayas, Dr Cs. Carlos Alvarez de. *METODOLOGÍA DE LA INVESTIGACÍON CIENTÍFICA.* Santiago de Cuba : s.n, 1995.
14. Dr Roberto Hernández Sampieri, Dra Pilar Baptista Lucio. *Metodología de la Investigación Cuarta Edición.* México : s.n, 2006.

Glosario de términos

API: interfaz de programación de aplicaciones o API (del inglés *Application Programming Interface*) es el conjunto de funciones y procedimientos (o métodos, si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro *software* como una capa de abstracción.

Artefactos: un artefacto es un producto tangible resultante del proceso de desarrollo de *software*. Algunos artefactos como los casos de uso, diagrama de clases u otros modelos UML ayudan a la descripción de la función, la arquitectura o el diseño del *software*.

Biblioteca: en ciencias de la computación, una biblioteca (o librería) es un conjunto de subprogramas utilizados para desarrollar *software*. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos.

Paquete: un paquete de *software* es una serie de programas que se distribuyen conjuntamente. Algunas de las razones suelen ser que el funcionamiento de cada uno complementa a o requiere de otros, además de que sus objetivos están relacionados como estrategia de mercadotecnia.

Página *web*: documento situado en el *web* con información y diversos enlaces con otros documentos también situados en el *web*.

RF: requisito funcional.

Servicio *web*: sistemas de *software* que permiten el intercambio de datos y funcionalidades entre aplicaciones sobre una red soportado en diferentes estándares que garantizan la interoperabilidad de estos.

UML: siglas de Unified Modeling Language, lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de *software*.

URL: siglas de Uniform Resource Locator*,* es la dirección global de cualquier documento o recurso en la *web*, visto como localizador.

WWW: *World Wide Web* o Red Global Mundial es un sistema de documentos de hipertexto y/o hipermedios enlazados y accesibles a través de internet.

Anexos

Anexo 1: Índice pornográfico en Internet

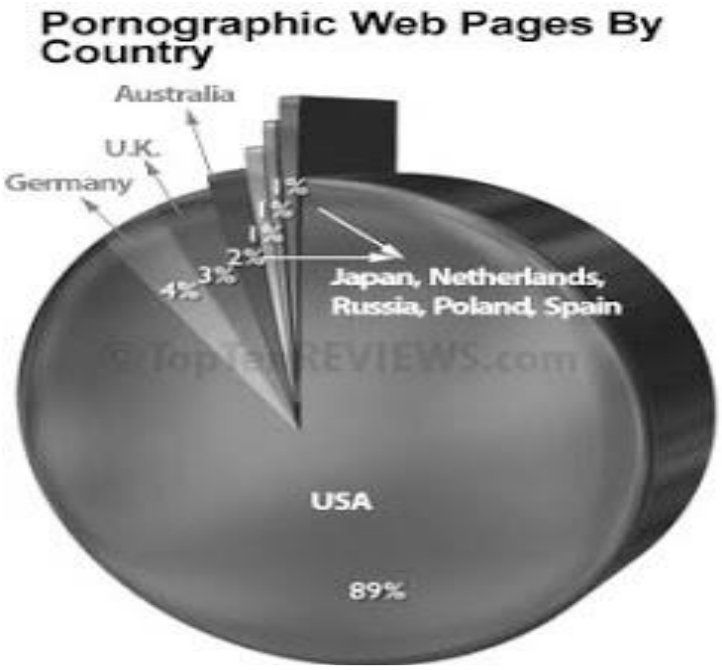

Figura 18 Índice pornográfico de Internet

Figura 19 Pornografía en el mundo: mapa con los países con más contenido online

Anexo 2: Modelo de entrevista

MODELO DE ENTREVISTA
Nombre(s):
Apellidos:
Cargo que Ocupa:
Departamento al que pertenece:
Preguntas
1. ¿En qué consiste la categorización de texto? 2. ¿Cuáles son las características fundamentales del Motor de Clasificación Inteligente de Contenido (MOCIC)? 3. ¿Cómo se realiza el proceso de categorización de texto en el MOCIC? 4. ¿Actualmente existe algún mecanismo para llevar a cabo el proceso de categorización de texto en el MOCIC? 5. ¿Qué limitaciones presenta el proceso de categorización de texto en el MOCIC actualmente? 6. ¿Qué herramientas son las más adecuadas para el desarrollo de una capa de servicios *web* para el MOCIC?

Tabla 27 Modelo de entrevista

Anexo 3: Respuesta de las peticiones a los servicios en formato JSON

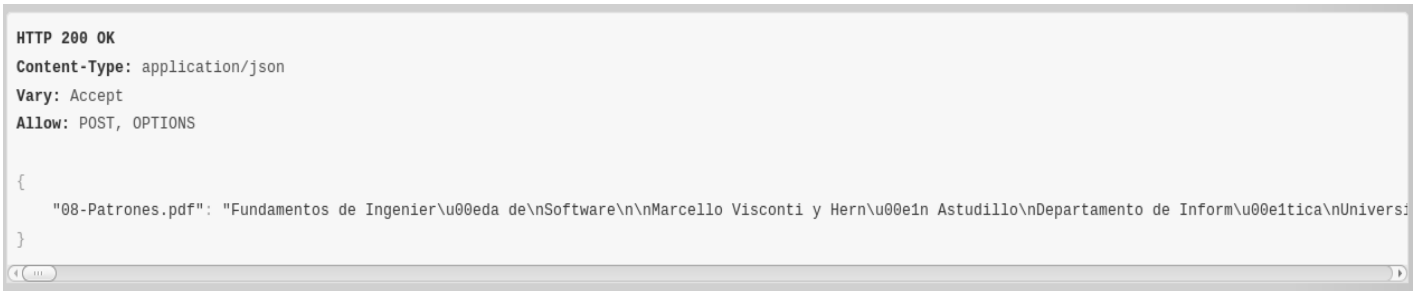

Figura 20 JSON del servicio Extraer texto plano

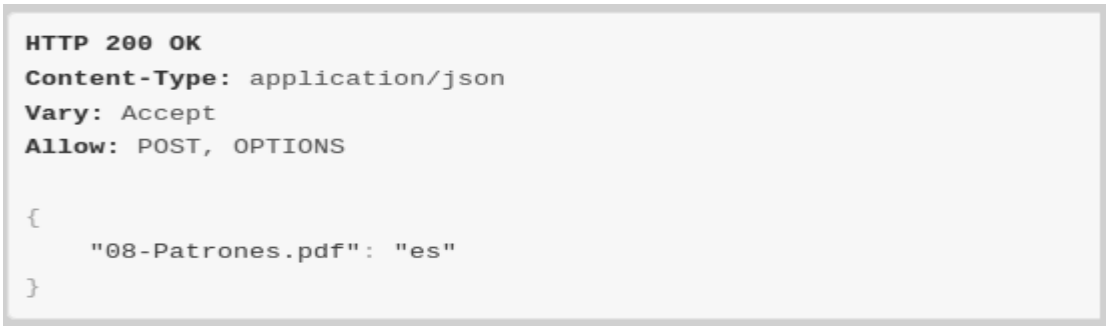

Figura 21 JSON del servicio Detectar lenguaje

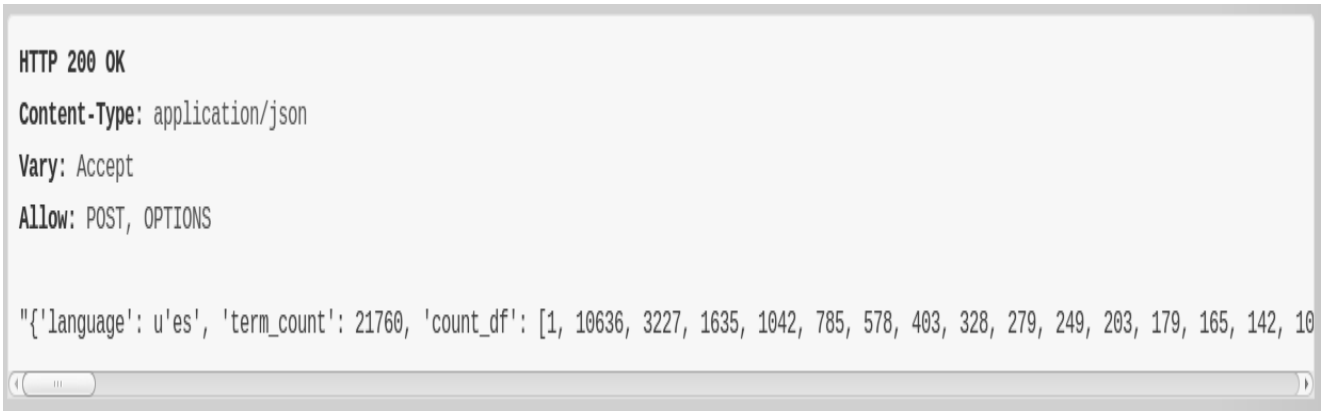

Figura 22 JSON del servicio Adicionar documentos a la colección

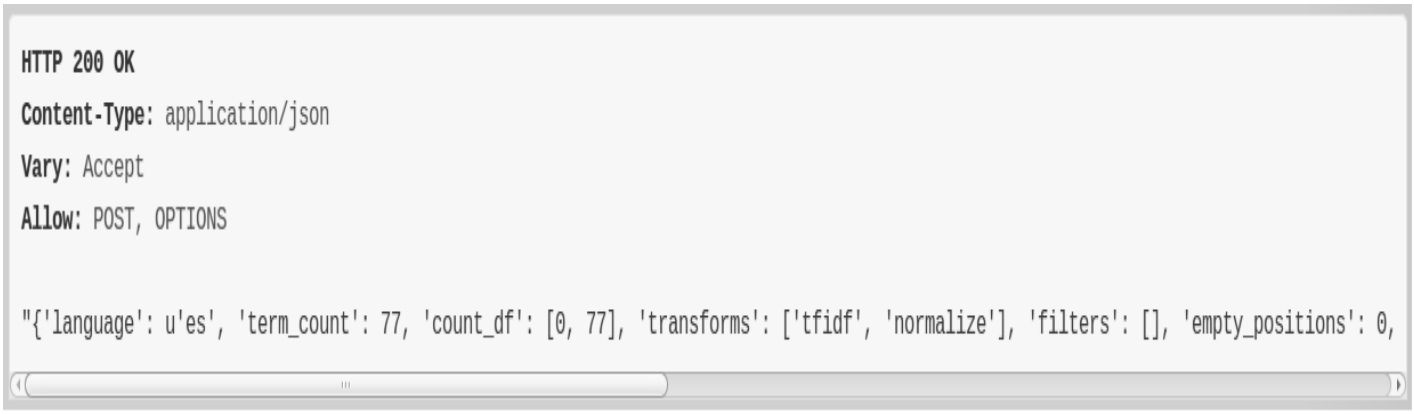

Figura 23 JSON del servicio Adicionar un documento a la colección

Figura 24 JSON del servicio Categorizar texto

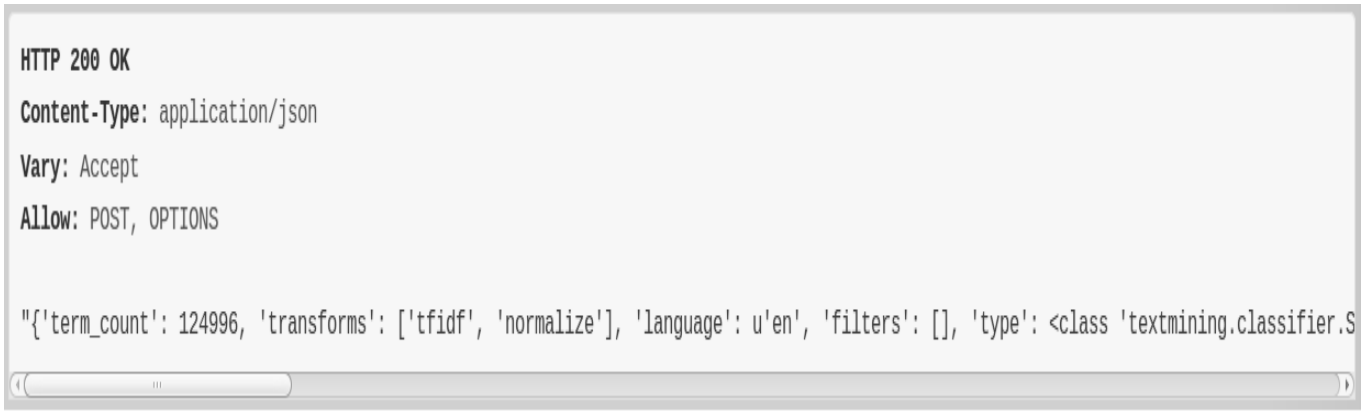

Figura 25 JSON del servicio Entrenar categorizador

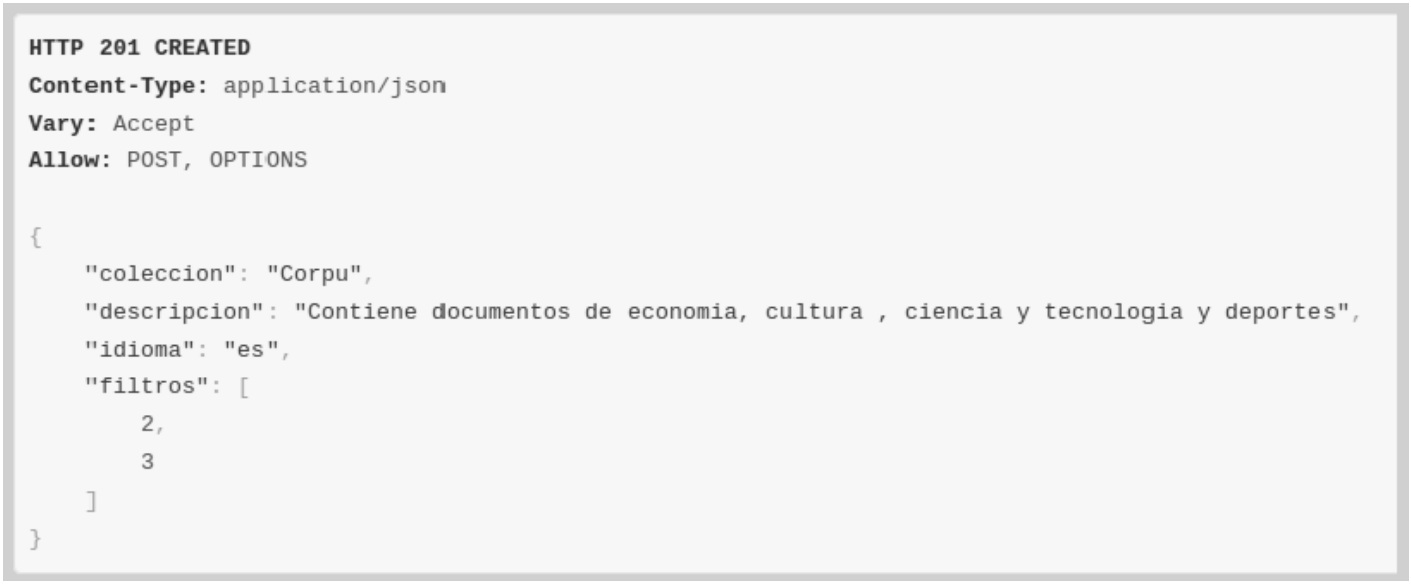

Figura 26 JSON del servicio Crear entrenamiento. Nueva colección

```
HTTP 201 CREATED
Content-Type: application/json
Vary: Accept
Allow: POST, OPTIONS

{
    "nombre": "Categorzador1",
    "descripcion": "Clasifica noticias",
    "tipo_categorizador": 1
}
```

Figura 27 JSON del servicio Nuevo categorizador

```
HTTP 201 CREATED
Content-Type: application/json
Vary: Accept
Allow: POST, OPTIONS

{
    "archivo": "categorizador/categorizador/upload/coleccion/c.collection"
}
```

Figura 28 JSON del servicio Crear entrenamiento. Subir colección

Anexo 4: Especificación de los casos de uso de las Interfaces de prueba y administración

CU 4	Detectar el idioma
Actores	Usuario
Resumen	El caso de uso se inicia cuando un usuario hace la petición al servicio Detección de idioma. El servicio es realizado y se brinda la respuesta correspondiente.
Complejidad	Media
Prioridad	Crítico
Referencia	RF4
Precondiciones	
Flujo de eventos	
Flujo básico < Detección de idioma >	

	Actor	Sistema
1.	El actor hace la petición al servicio “Detección de idioma”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento al cual se le quiere detectar el idioma.
3.	El actor carga el documento y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve el idioma del documento.
Flujos alternos		
	Actor	**Sistema**
		1. Si el documento no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 17 Especificación del caso de uso Detección de idioma

CU 7	Crear entrenamiento	
Actores	Administrador	
Resumen	El caso de uso se inicia cuando un administrador hace la petición al servicio Crear entrenamiento. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF7, RF8, RF9, RF10	
Precondiciones	El usuario tiene que ser administrador del sistema.	
Flujo de eventos		
Flujo básico < Crear entrenamiento >		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Crear entrenamiento”.	2. El sistema muestra una ventana en el navegador con un formulario que muestra dos posibles formas de crear entrenamiento; subiendo una colección que tenga definido sus categorías o creando una nueva colección y adicionándole los nuevos documentos.
3.	El actor escoge la forma de crear el	4. El sistema crea el entrenamiento para una categoría

	entrenamiento y da clic al botón “Enviar”.	específica.
Flujos alternos		
	Actor	**Sistema**
		1. Si la colección de documentos no es cargada correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 18 Especificación del caso de uso Crear entrenamiento

CU 8	Crear categorizador	
Actores	Administrador	
Resumen	El caso de uso se inicia cuando un administrador hace la petición al servicio Crear categorizador. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF11	
Precondiciones	El usuario tiene que ser administrador del sistema.	
Flujo de eventos		
Flujo básico < Crear categorizador>		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Crear categorizador”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el nuevo categorizador, y una serie de parámetros como son nombre, descripción y tipo de categorizador que son necesario llenar para realizar el servicio.
3.	El actor carga el categorizador, llena los parámetros y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema registra el nuevo categorizador en la base de datos.
Flujos alternos		
	Actor	**Sistema**
		1. El sistema verifica si el usuario autenticado es administrador del sistema antes de que realice la petición de Crear categorizador.

		2. Si los campos que se muestran para crear un categorizador no son completados correctamente, el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 19 Especificación del caso de uso Crear categorizador

CU 9	Entrenar categorizador	
Actores	Administrador	
Resumen	El caso de uso se inicia cuando un administrador hace la petición al servicio Entrenar categorizador. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF12	
Precondiciones	El usuario tiene que ser administrador del sistema.	
Flujo de eventos		
Flujo básico < Entrenar categorizador >		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Entrenar categorizador”.	2. El sistema muestra una ventana en el navegador con un “select” para seleccionar la colección de documentos con que se quiere entrenar el categorizador, y otro “select” para seleccionar el categorizador que se desea entrenar.
3.	El actor selecciona el categorizador y la colección de documentos con la que lo va a entrenar y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema entrena categorizador con la colección de documentos seleccionados.
Flujos alternos		
	Actor	**Sistema**
		1. El sistema verifica si el usuario autenticado es administrador del sistema antes de que realice la petición de Entrenar categorizador.

		2. El sistema verifica que sean seleccionados los campos requeridos para realizar el servicio.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 20 Especificación del caso de uso Entrenar categorizador

CU 10	Evaluar categorizador	
Actores	Administrador	
Resumen	El caso de uso se inicia cuando un administrador hace la petición al servicio Evaluar categorizador. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF13	
Precondiciones	El usuario tiene que ser administrador del sistema.	
Flujo de eventos		
Flujo básico < Evaluar categorizador >		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Evaluar categorizador”.	2. El sistema muestra una ventana en el navegador con un “select” para seleccionar el categorizador que se desea evaluar y un “Examinar” para cargarlo en caso de que se desee evaluar un categorizador que no esté en la base de datos.
3.	El actor carga el categorizador o lo selecciona y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema evalúa el categorizador y devuelve la evaluación del mismo.
Flujos alternos		
	Actor	**Sistema**
		1. El sistema verifica si el usuario autenticado es administrador del sistema antes de que realice la petición de Evaluar categorizador.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 21 Especificación del caso de uso Evaluar categorizado

Especificación de los casos de uso correspondientes a los Servicios REST

CU 5	Categorizar texto	
Actores	Aplicación Externa	
Resumen	El caso de uso se inicia cuando una aplicación externa hace la petición al servicio Categorizar texto. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF5	
Precondiciones	Debe estar autenticado.	
Flujo de eventos		
Flujo básico <Categorizar texto>		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Categorizar texto”.	5. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento que se quiere categorizar, y un botón “Enviar” para realizar el servicio.
6.	El actor carga el documento que se quiere clasificar y acepta realizar la petición dando clic en el botón “Enviar”.	7. El sistema devuelve la categoría del documento en un JSON.
Flujos alternos		
	Actor	**Sistema**
		8. Si el archivo no es cargado correctamente, el sistema envía un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 28 Especificación del caso de uso. Servicio REST. Categorizar texto.

CU 2	Extraer texto plano
Actores	Aplicación Externa
Resumen	El caso de uso se inicia cuando una aplicación externa hace la petición al servicio Extracción

	de texto plano. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF2	
Precondiciones	El usuario debe estar autenticado	
Flujo de eventos		
Flujo básico < Extracción de texto plano >		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Extracción de texto plano”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento al cual se le quiere extraer el texto plano.
1.	El actor carga el documento y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve el texto plano del documento.
Flujos alternos		
	Actor	**Sistema**
		1. Si el documento no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 29 Especificación del caso de uso. Servicio REST. Extraer texto plano

CU 4	Detectar el idioma
Actores	Aplicación Externa
Resumen	El caso de uso se inicia cuando una aplicación externa hace la petición al servicio Detección de idioma. El servicio es realizado y se brinda la respuesta correspondiente.
Complejidad	Media
Prioridad	Crítico
Referencia	RF4
Precondiciones	
Flujo de eventos	
Flujo básico < Detección de idioma >	

	Actor	Sistema
1.	El actor hace la petición al servicio “Detección de idioma”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento al cual se le quiere detectar el idioma.
3.	El actor carga el documento y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve el idioma del documento.
Flujos alternos		
	Actor	**Sistema**
		1. Si el documento no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 30 Especificación del caso de uso. Servicio REST. Detectar el idioma

CU 3	Detectar la codificación	
Actores	Aplicación Externa	
Resumen	El caso de uso se inicia cuando una aplicación externa hace la petición al servicio Detección de la codificación. El servicio es realizado y se brinda la respuesta correspondiente.	
Complejidad	Media	
Prioridad	Crítico	
Referencia	RF3	
Precondiciones		
Flujo de eventos		
Flujo básico < Detección de la codificación >		
	Actor	**Sistema**
1.	El actor hace la petición al servicio “Detección de la codificación”.	2. El sistema muestra una ventana en el navegador con un “Examinar” para cargar el documento al cual se le quiere detectar la codificación.
3.	El actor carga el documento y acepta realizar la petición dando clic en el botón “Enviar”.	4. El sistema devuelve la codificación del documento.

Flujos alternos		
	Actor	**Sistema**
		1. Si el documento no es cargado correctamente el sistema muestra un mensaje de error.
Postcondiciones	Se obtiene la respuesta del sistema.	

Tabla 31 Especificación del caso de uso. Servicio REST. Detectar la codificación.

Anexo 5: Interfaz de administración e interfaz de prueba de los servicios *web*

Figura 29 Capa de servicios

Figura 30 Interfaz de prueba. Adicionar documentos a la colección

Figura 31 Interfaz de prueba. Adicionar un documento a la colección

Figura 32 Interfaz de prueba. Categorizar

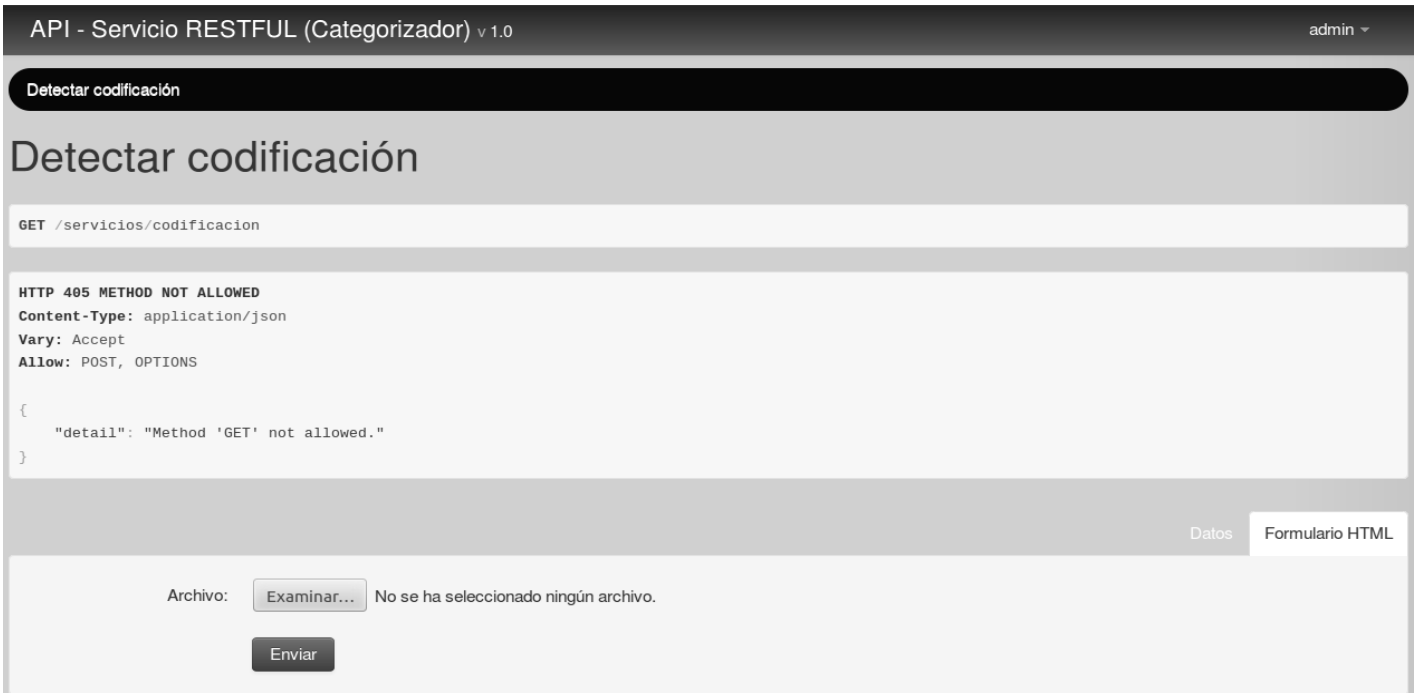

Figura 33 Interfaz de prueba. Detectar codificación

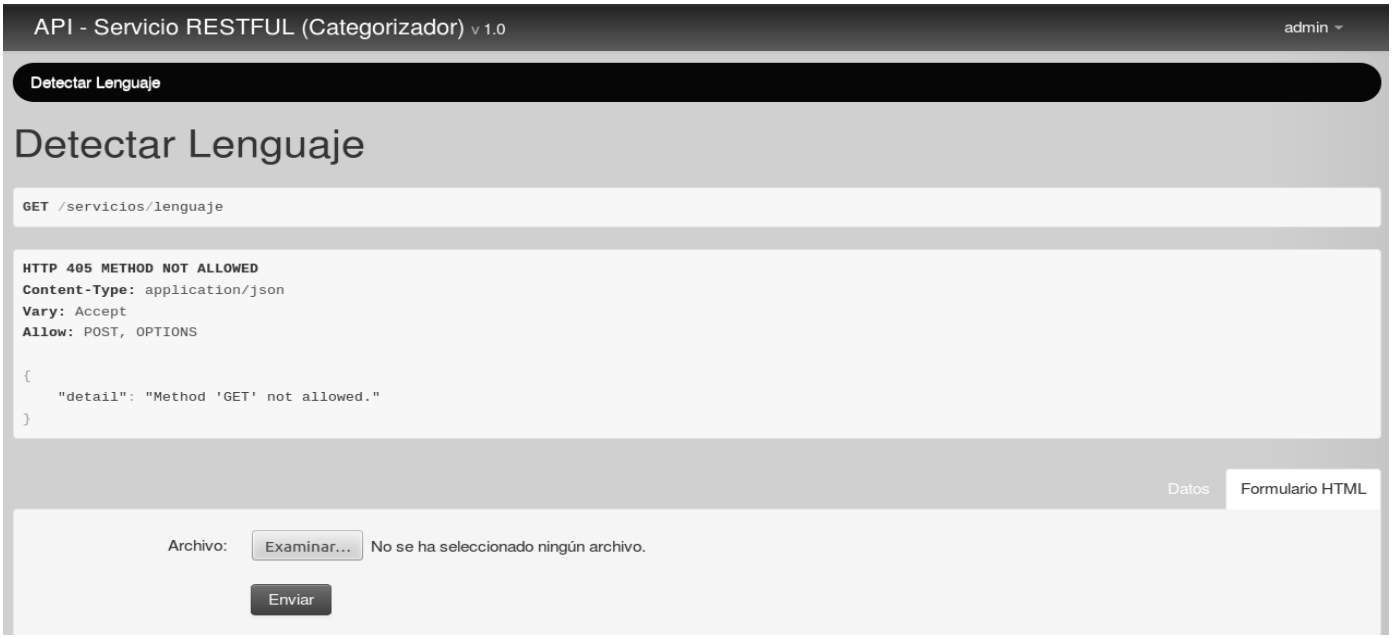

Figura 34 Interfaz de prueba. Detectar lenguaje

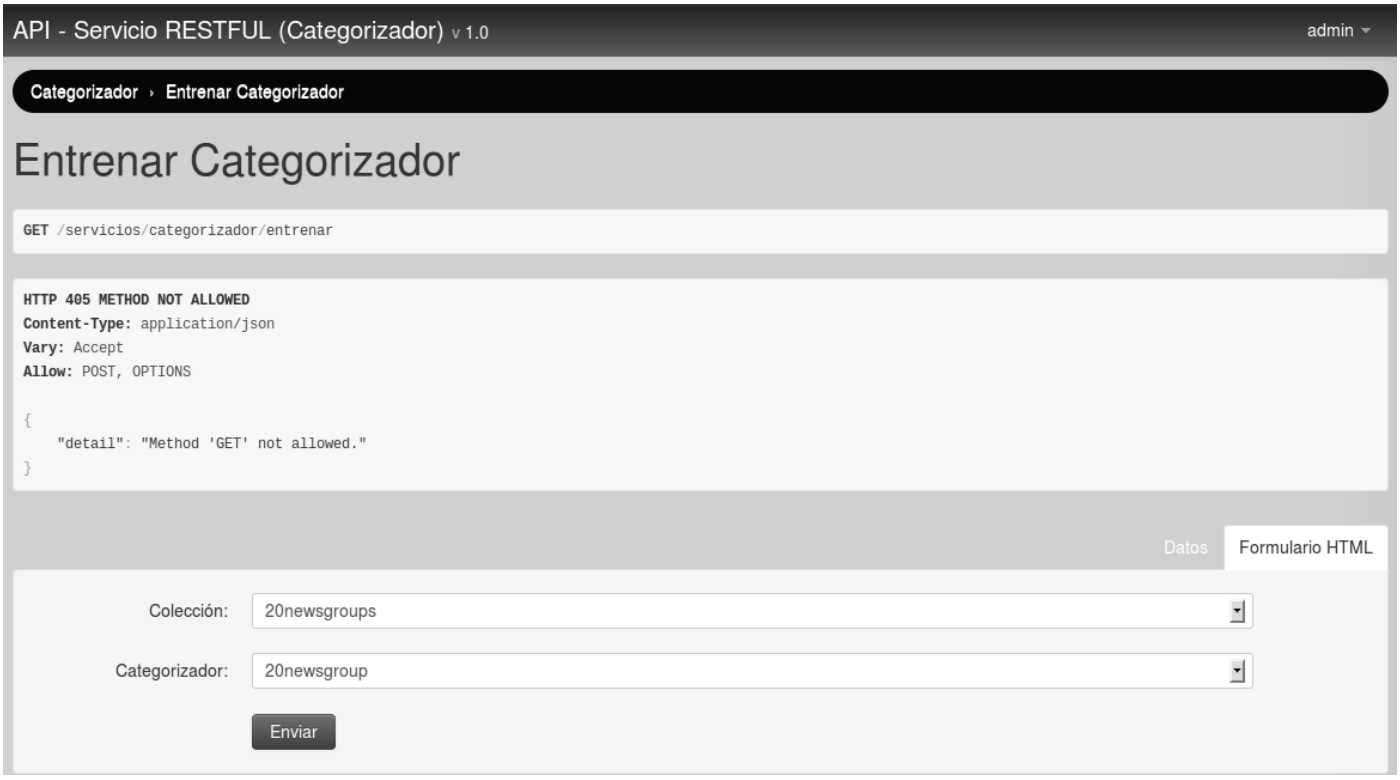

Figura 35 Interfaz de prueba. Entrenar categorizador

Figura 36 Interfaz de prueba. Evaluar categorizador

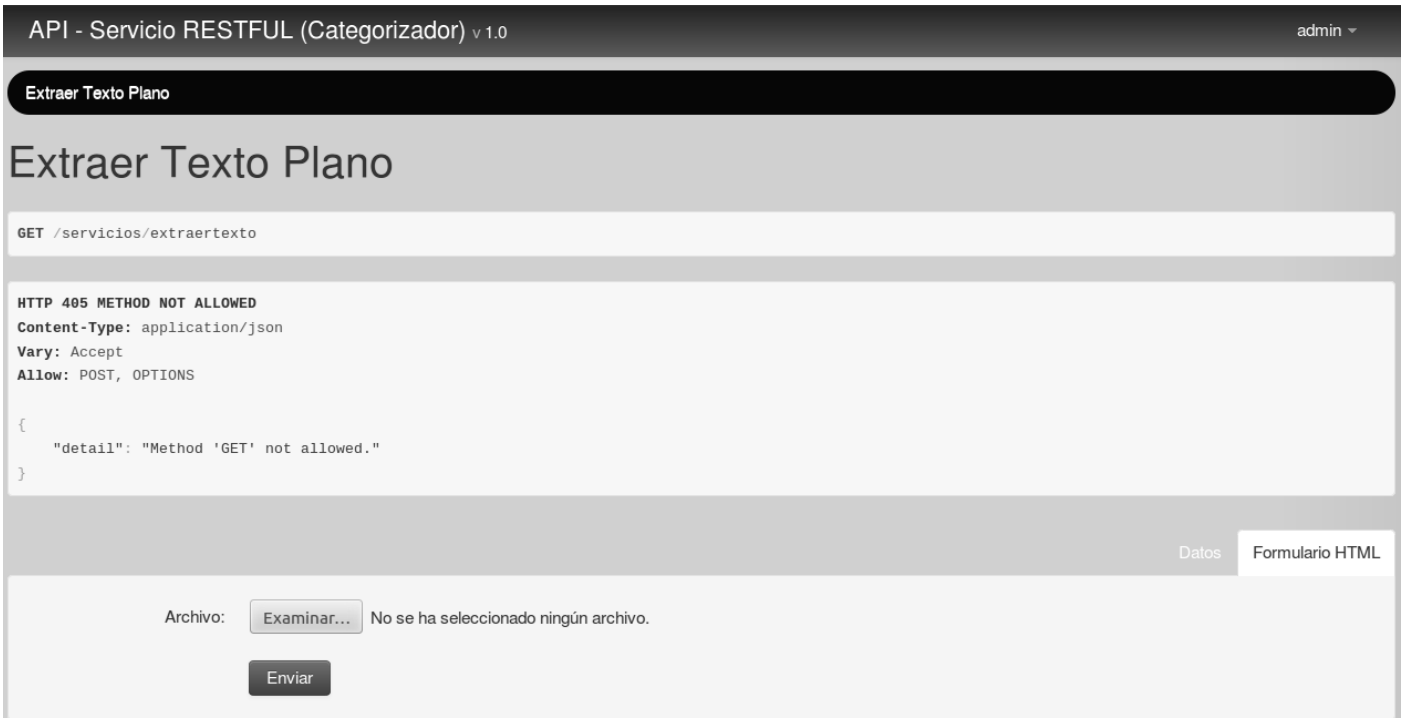

Figura 37 Interfaz de prueba. Extraer texto plano

Anexo 6: Diagramas de clases del diseño con estereotipos *web*

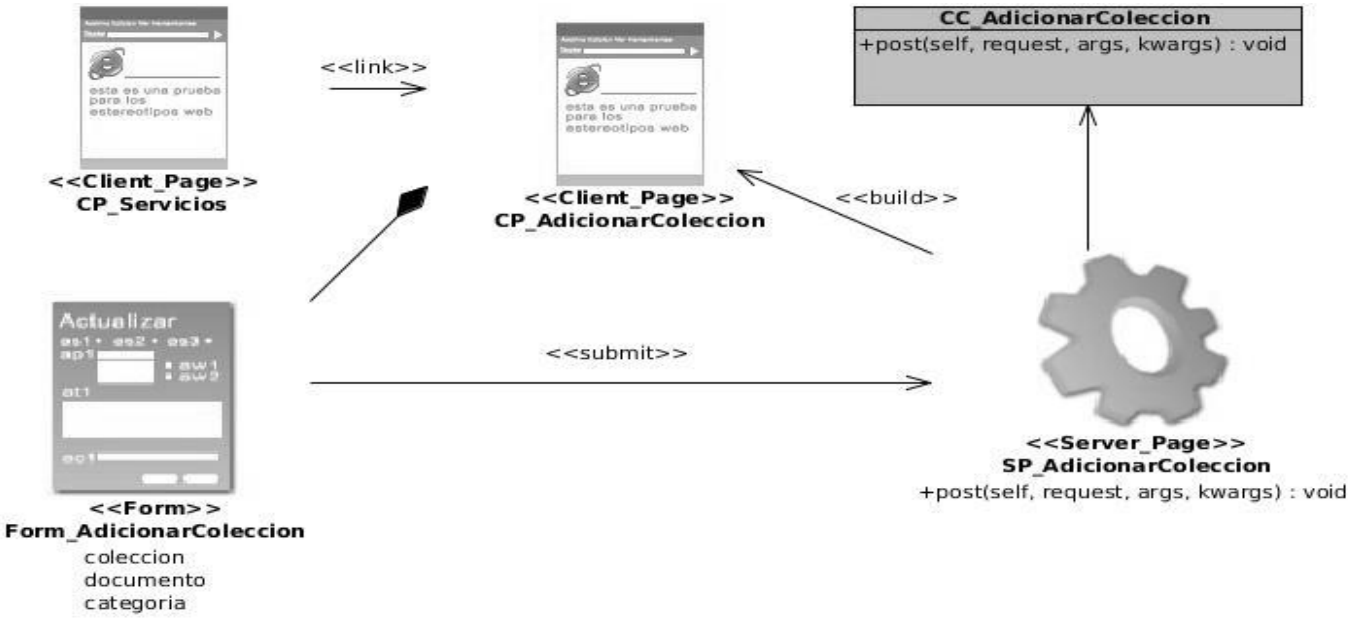

Figura 38 Diagrama de clases del diseño. Adicionar colección

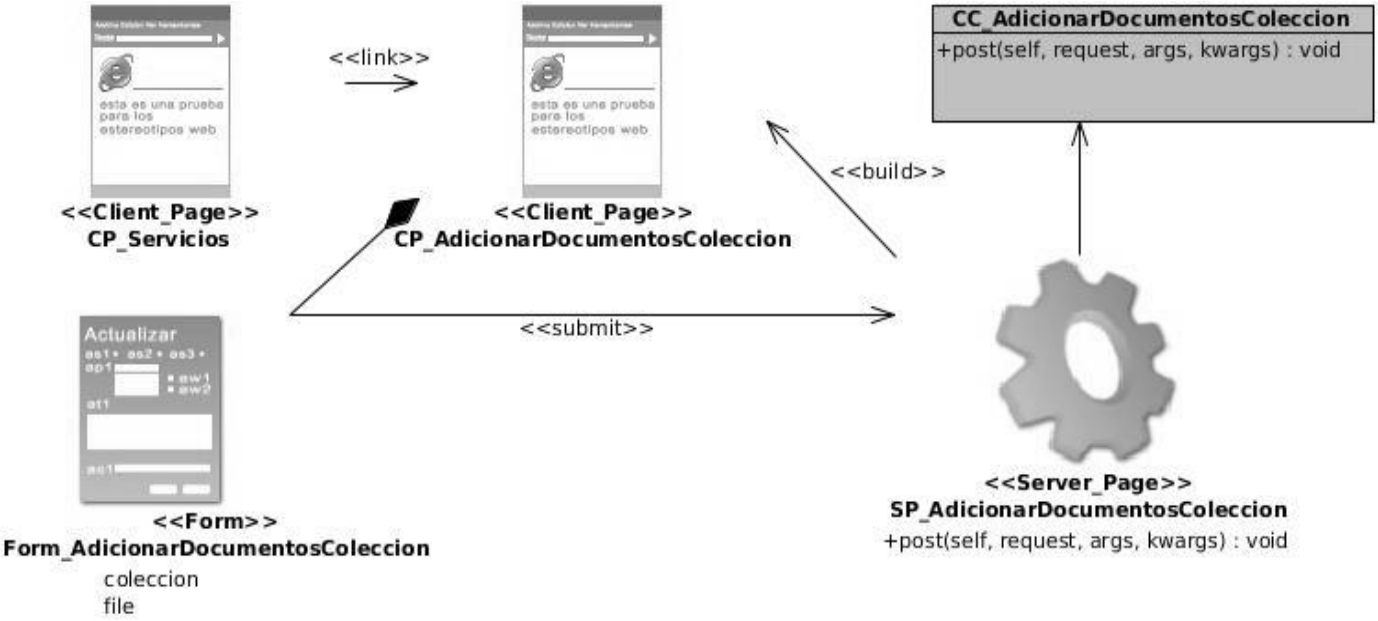

Figura 39 Diagrama de clases del diseño. Adicionar documentos colección

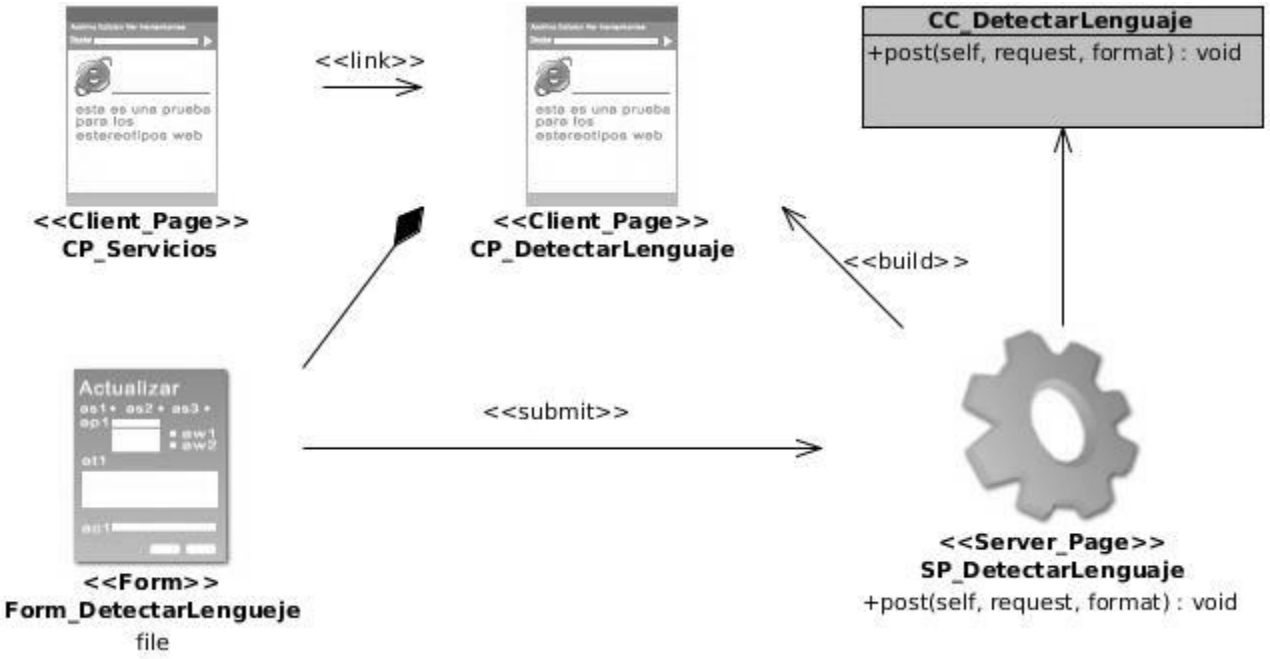

Figura 40 Diagrama de clases del diseño. Detectar lenguaje

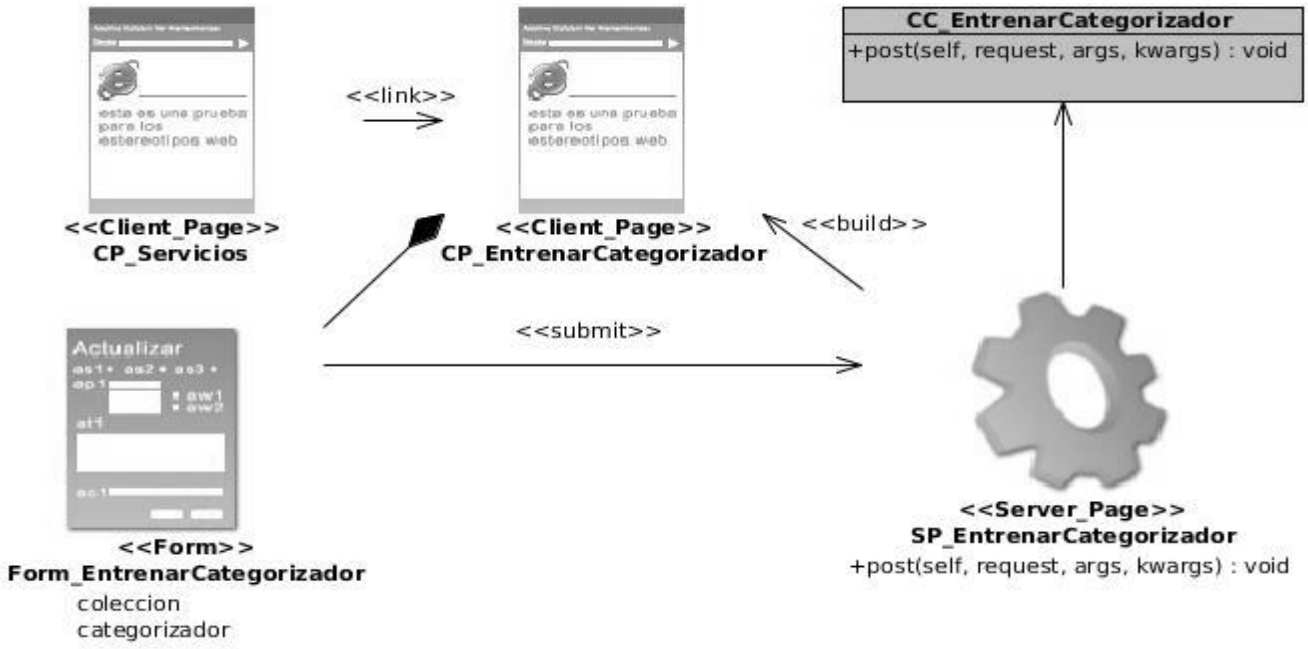

Figura 41 Diagrama de clases del diseño. Entrenar categorizador

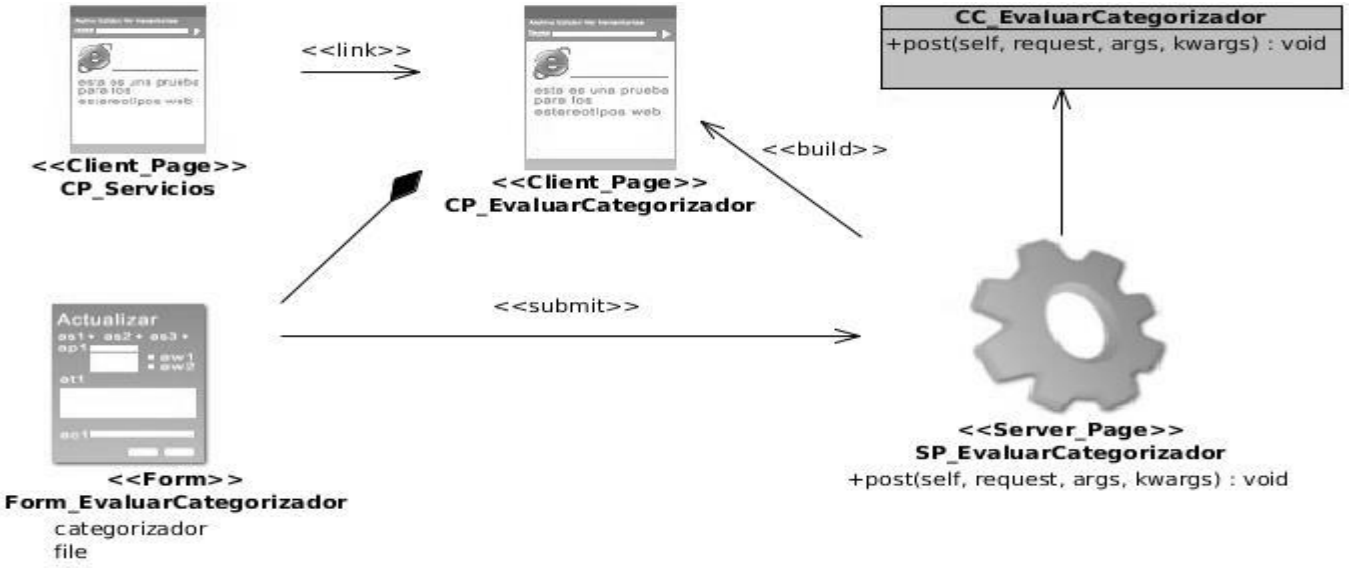

Figura 42 Diagrama de clases del diseño. Evaluar categorizador

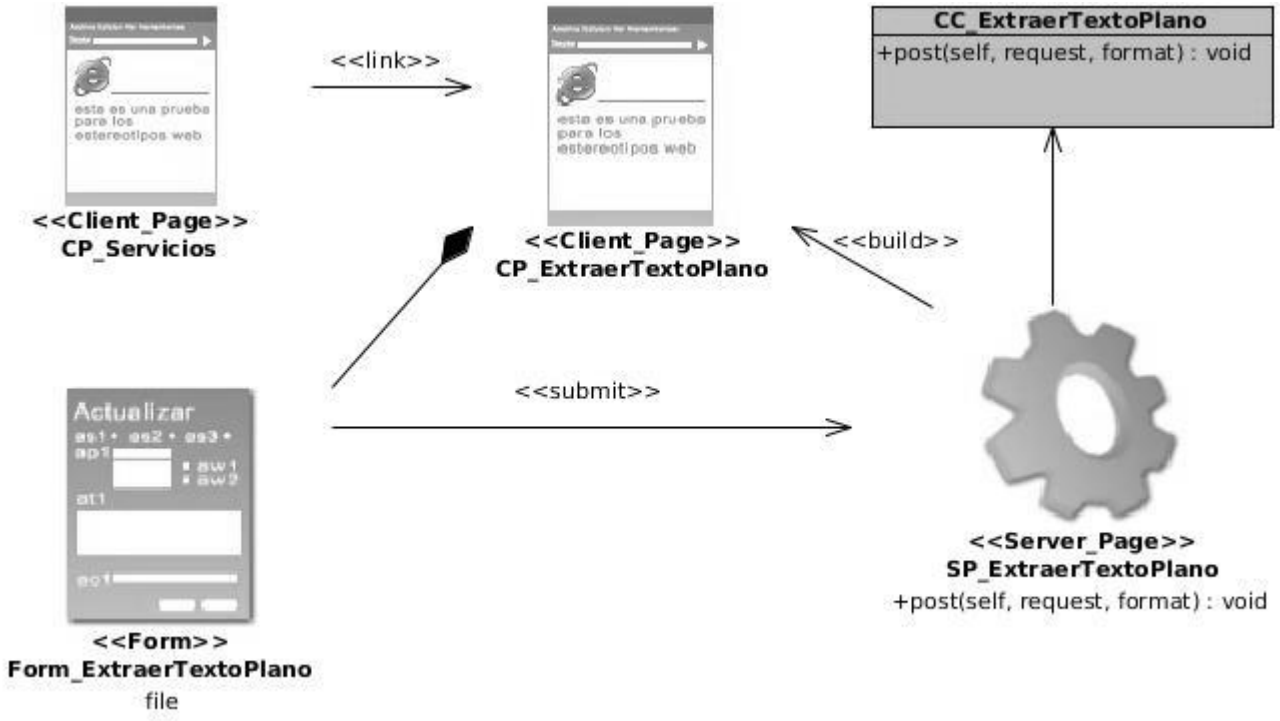

Figura 43 Diagrama de clases del diseño. Extraer texto plano

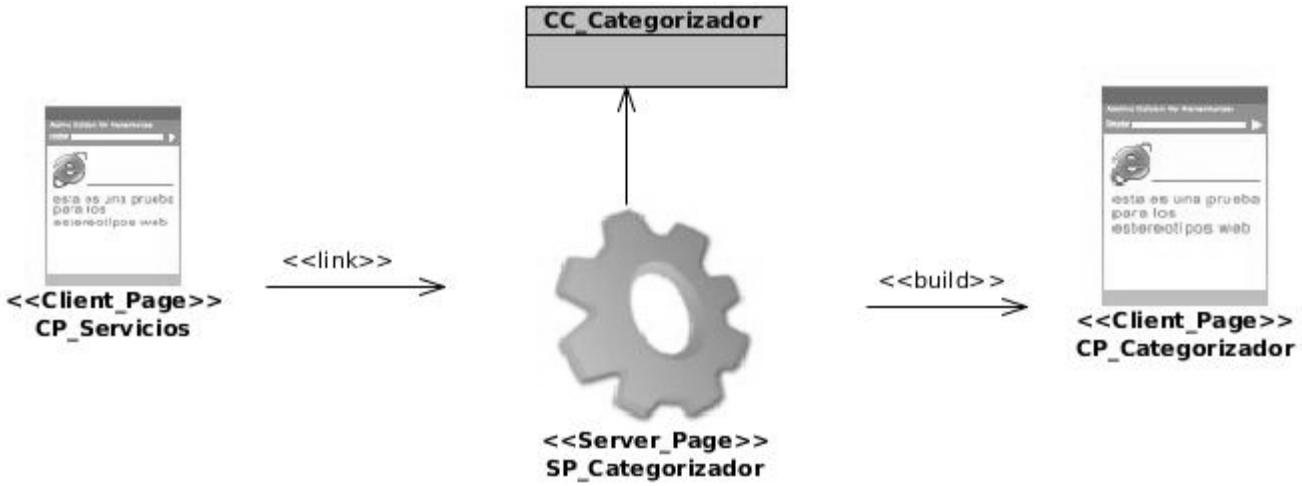

Figura 44 Diagrama de clases del diseño. Listar categorizador

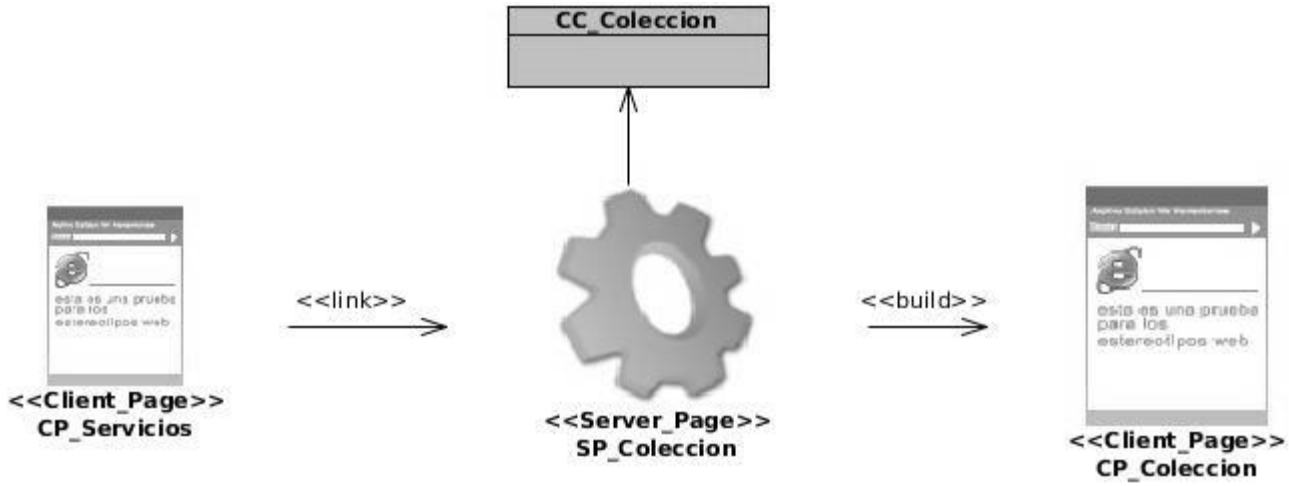

Figura 45 Diagrama de clases del diseño. Listar colección

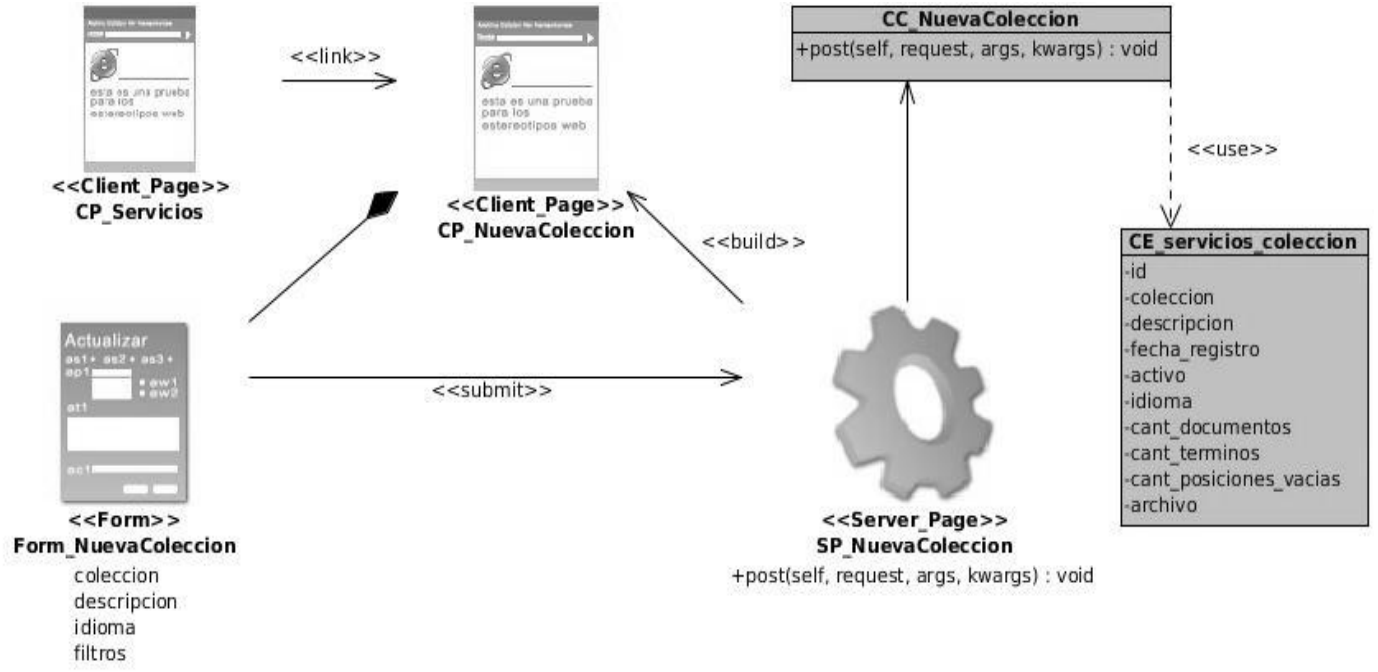

Figura 46 Diagrama de clases del diseño. Nueva colección

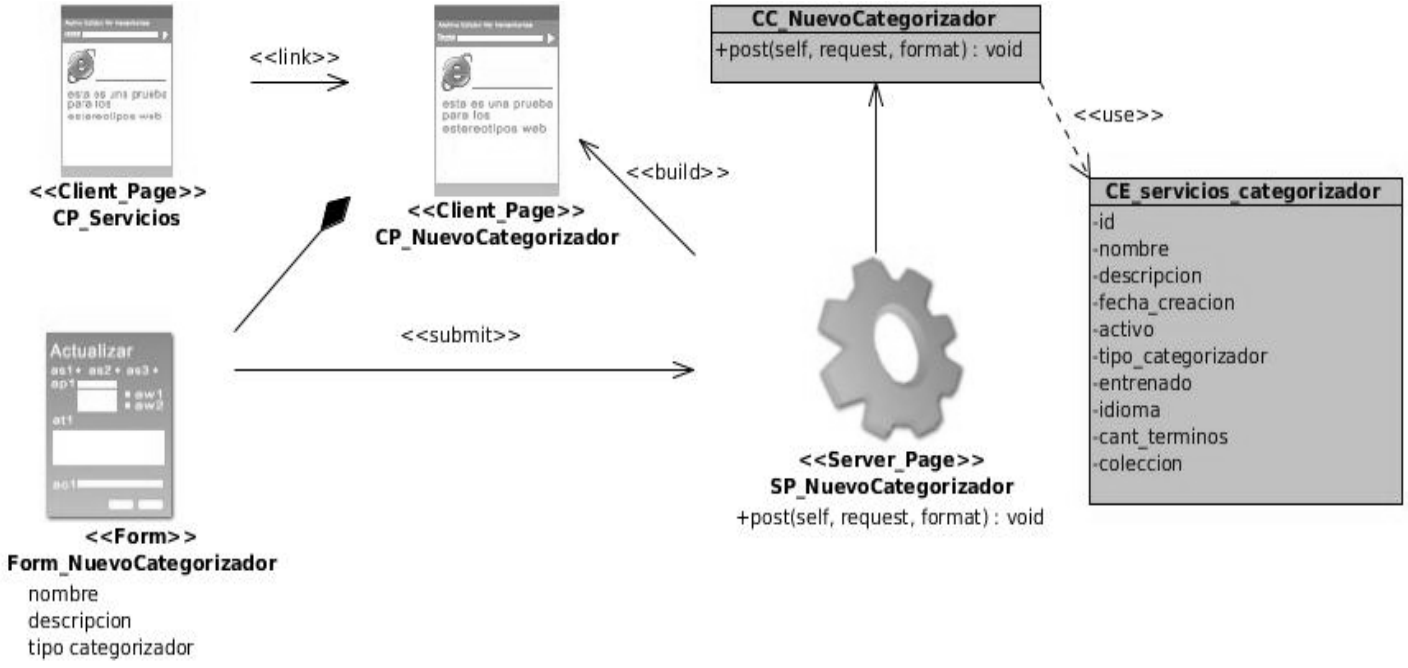

Figura 47 Diagrama de clases del diseño. Nuevo categorizador

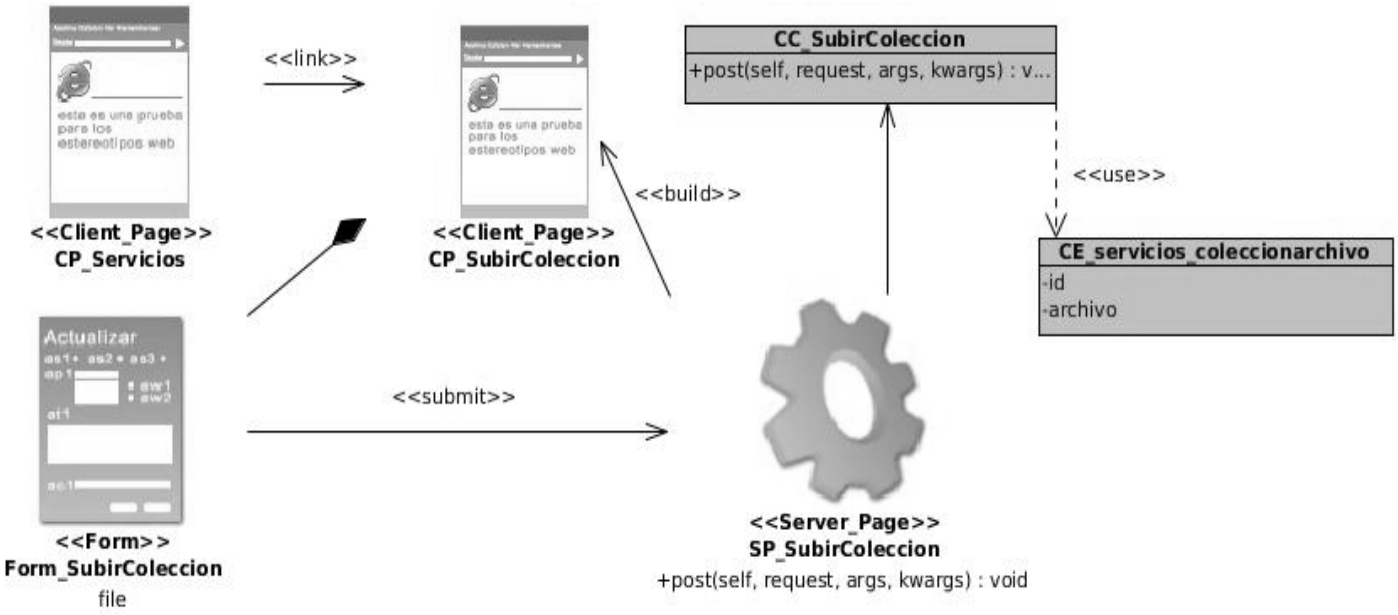

Figura 48 Diagrama de clases del diseño. Subir colección

Anexo 7: Diagramas de secuencia

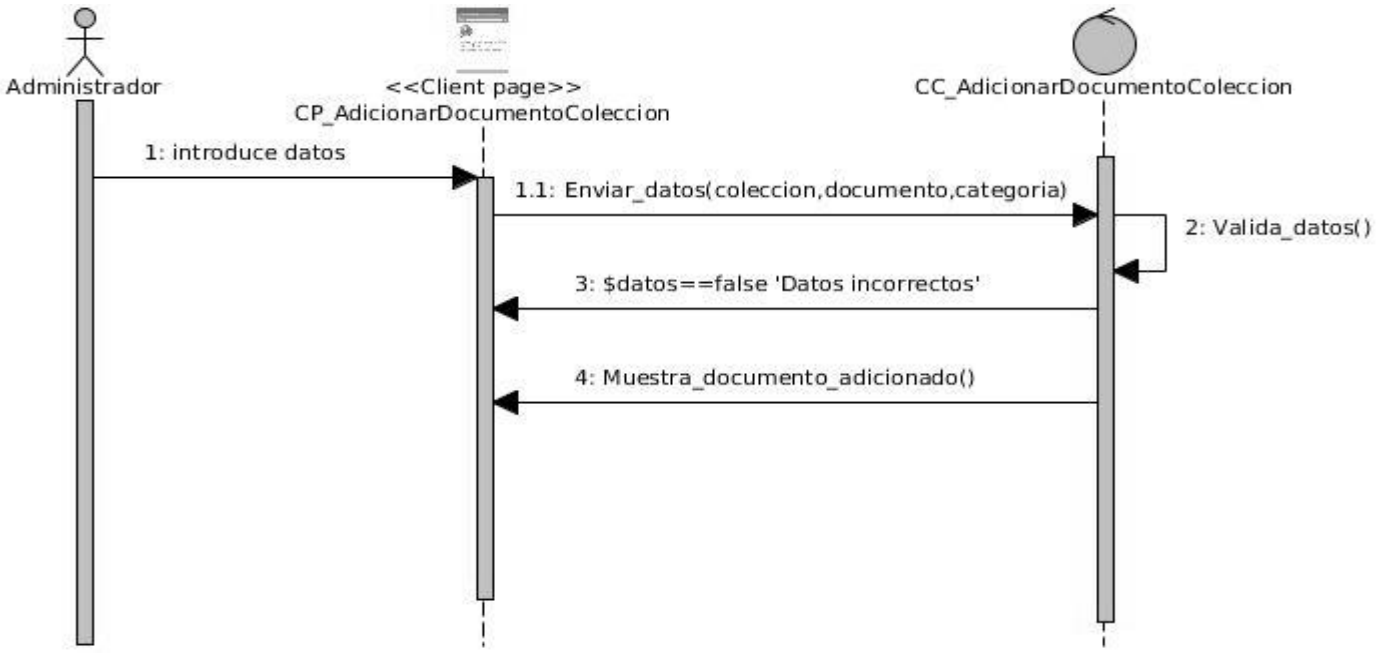

Figura 49 Diagrama de secuencia. Adicionar documento a la colección

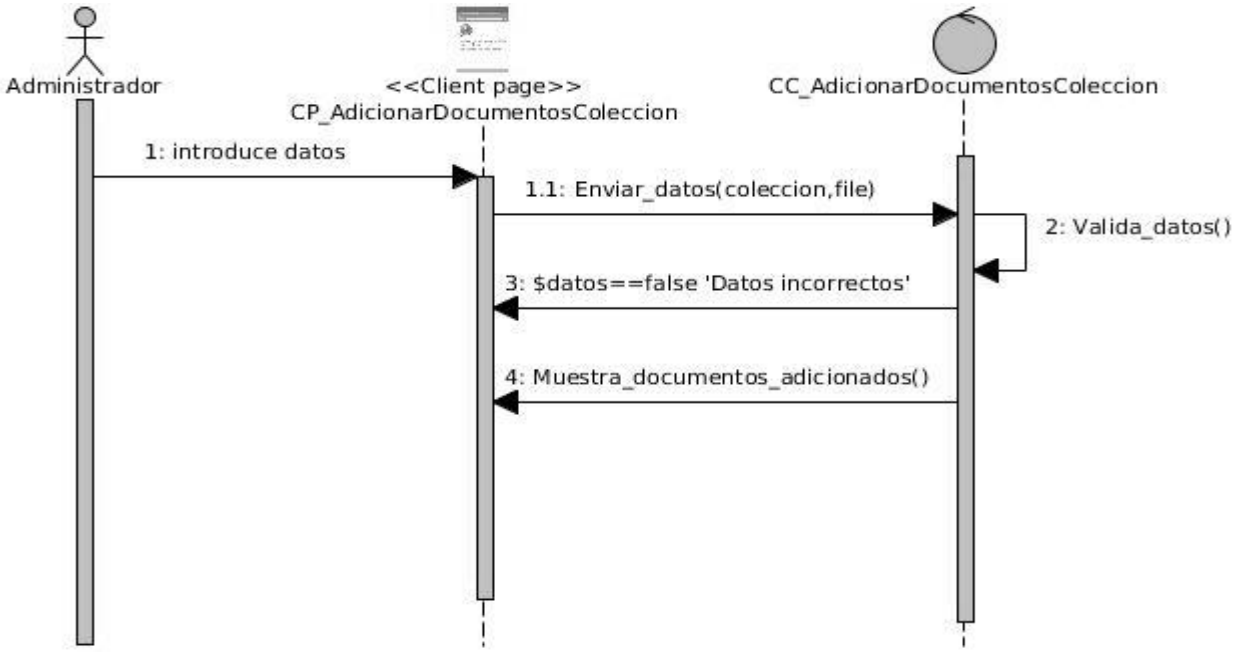

Figura 50 Diagrama de secuencia. Adicionar documentos a la colección

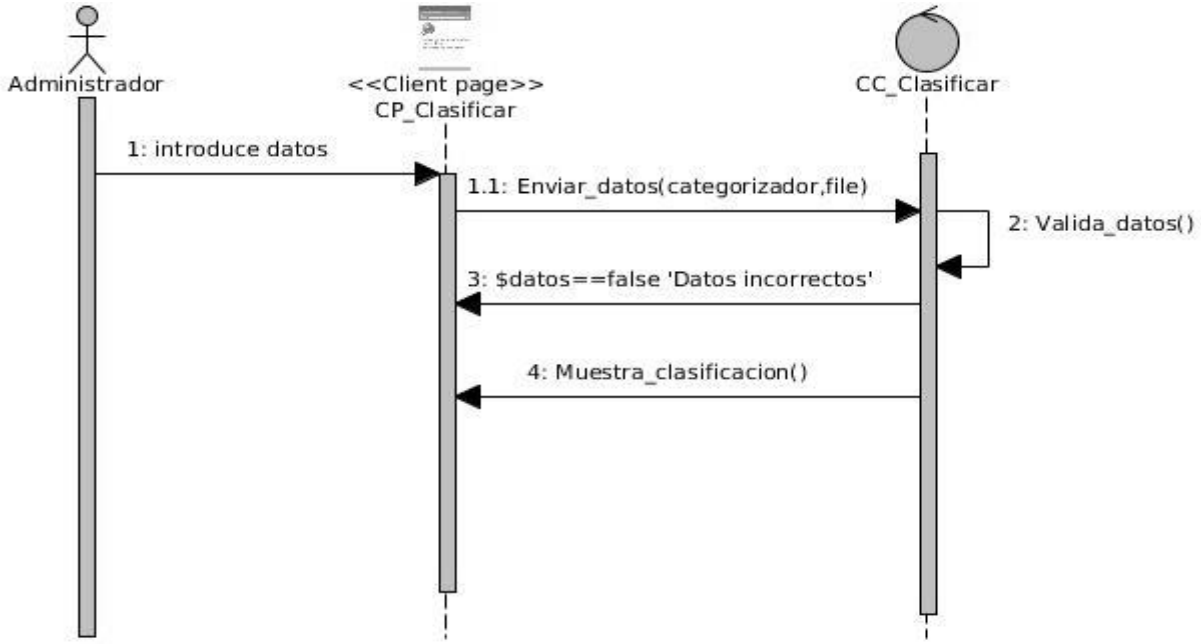

Figura 51 Diagrama de secuencia. Clasificar administrador

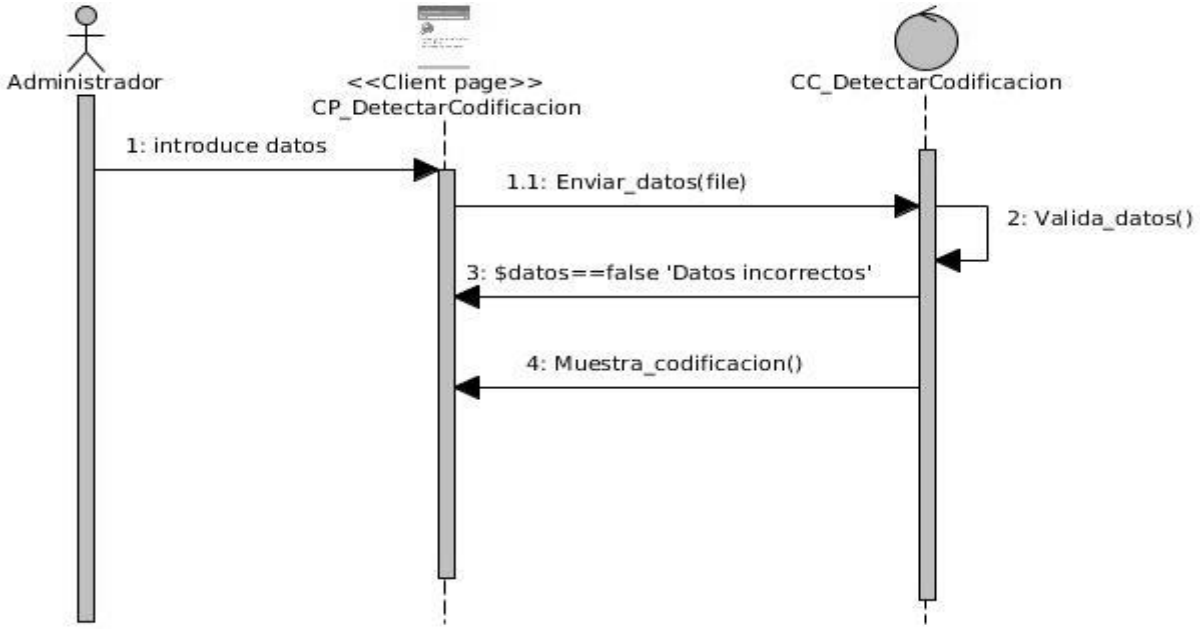

Figura 52 Diagrama de secuencia. Detectar codificación administrador

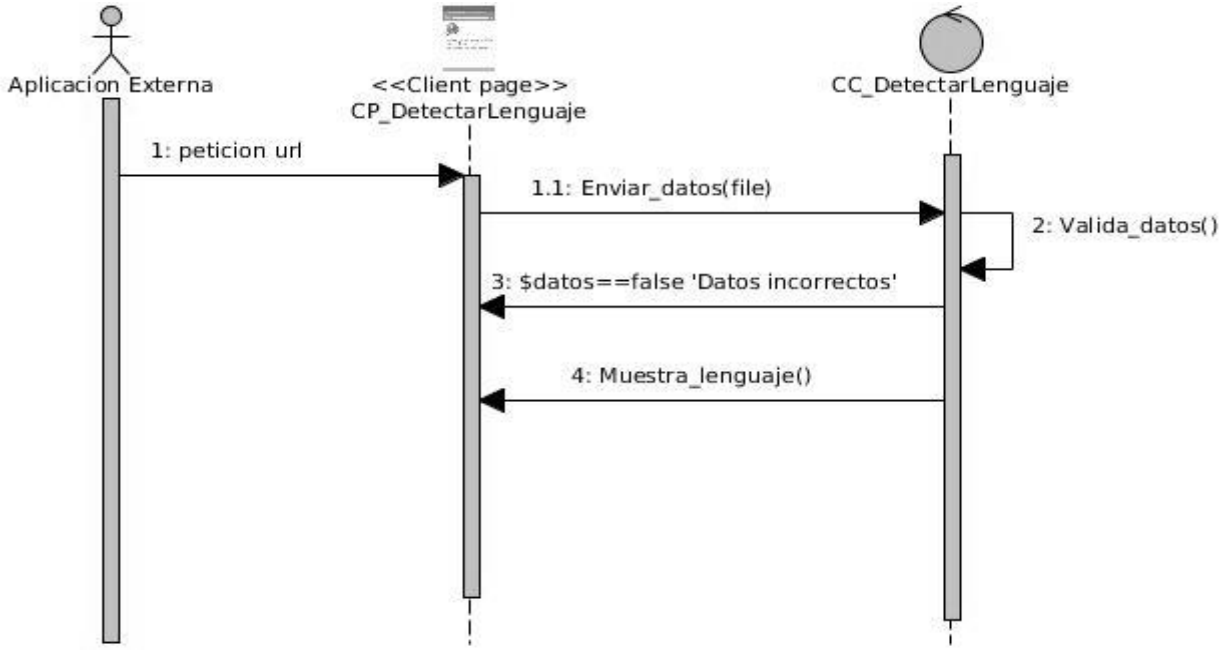

Figura 53 Diagrama de secuencia. Detectar lenguaje

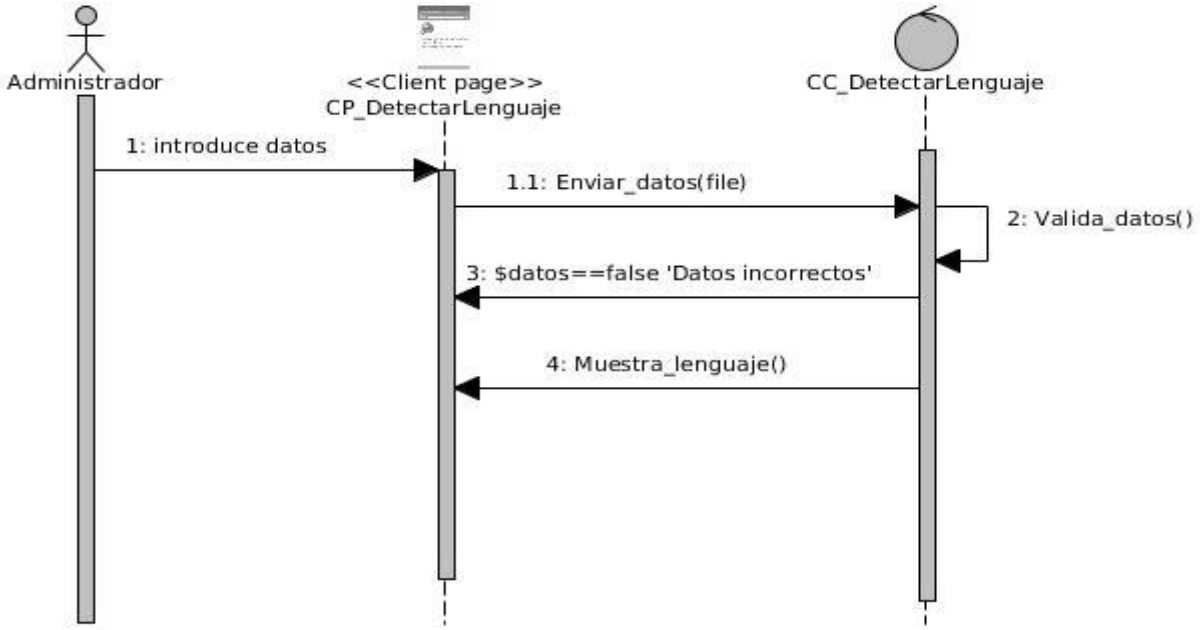

Figura 54 Diagrama de secuencia. Detectar lenguaje administrador

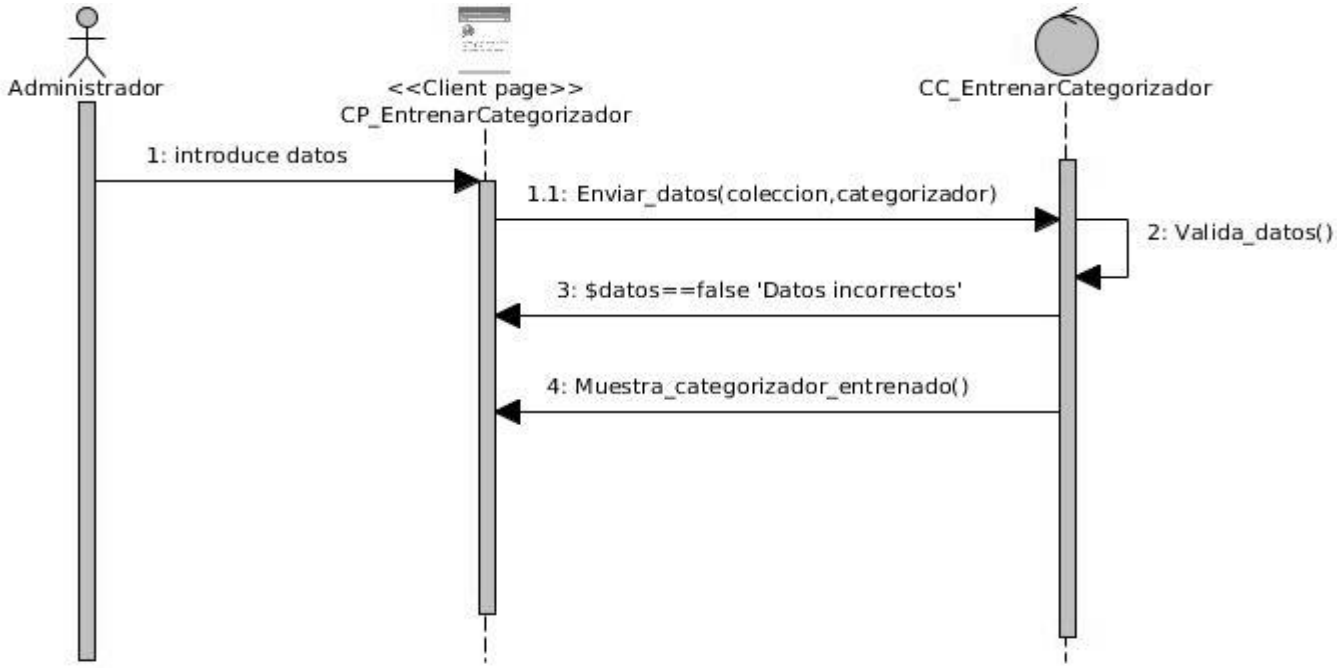

Figura 55 Diagrama de secuencia. Entrenar administrador

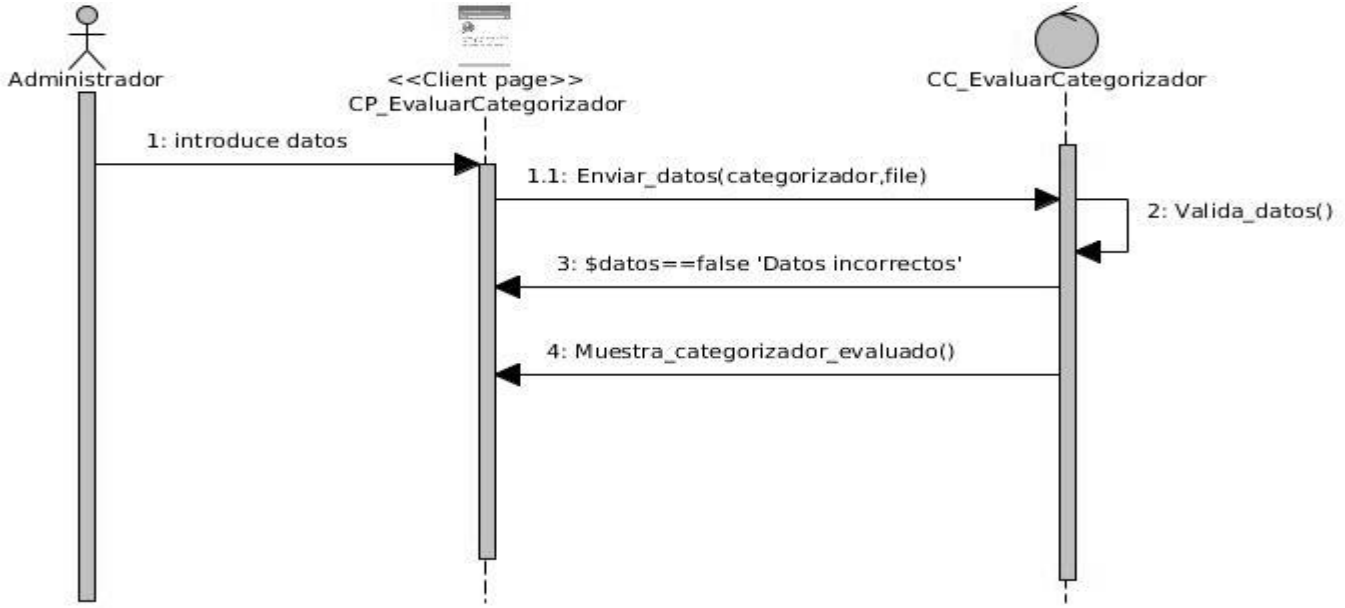

Figura 56 Diagrama de secuencia. Evaluar categorizador

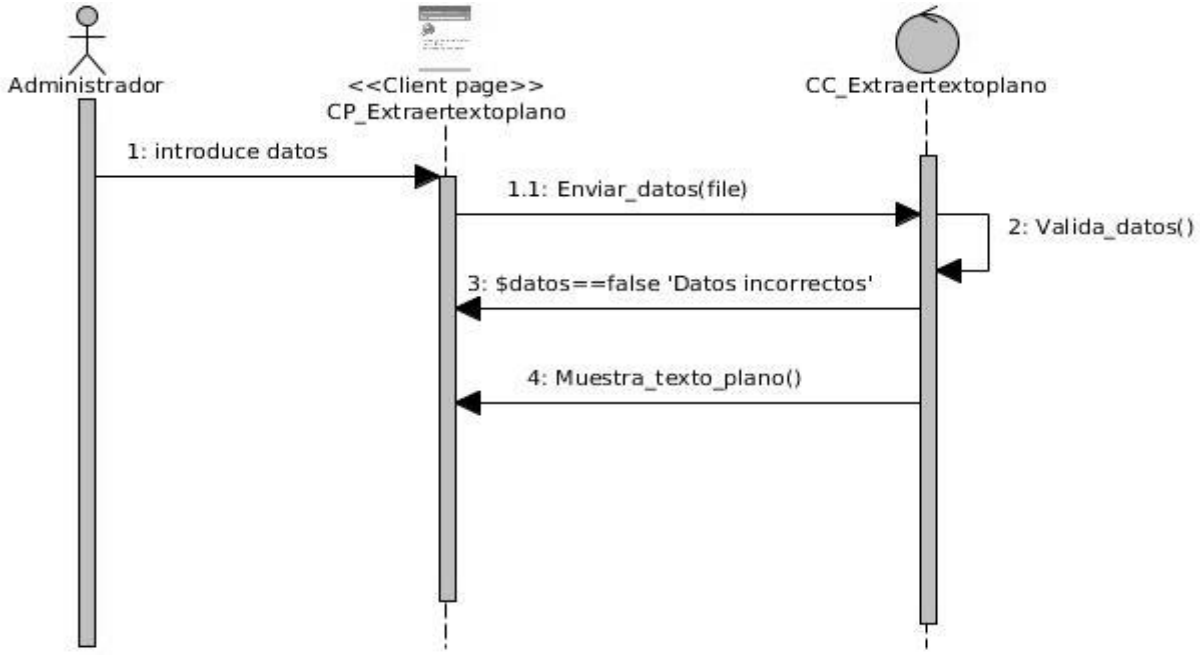

Figura 57 Diagrama de secuencia. Extraer texto plano administrador

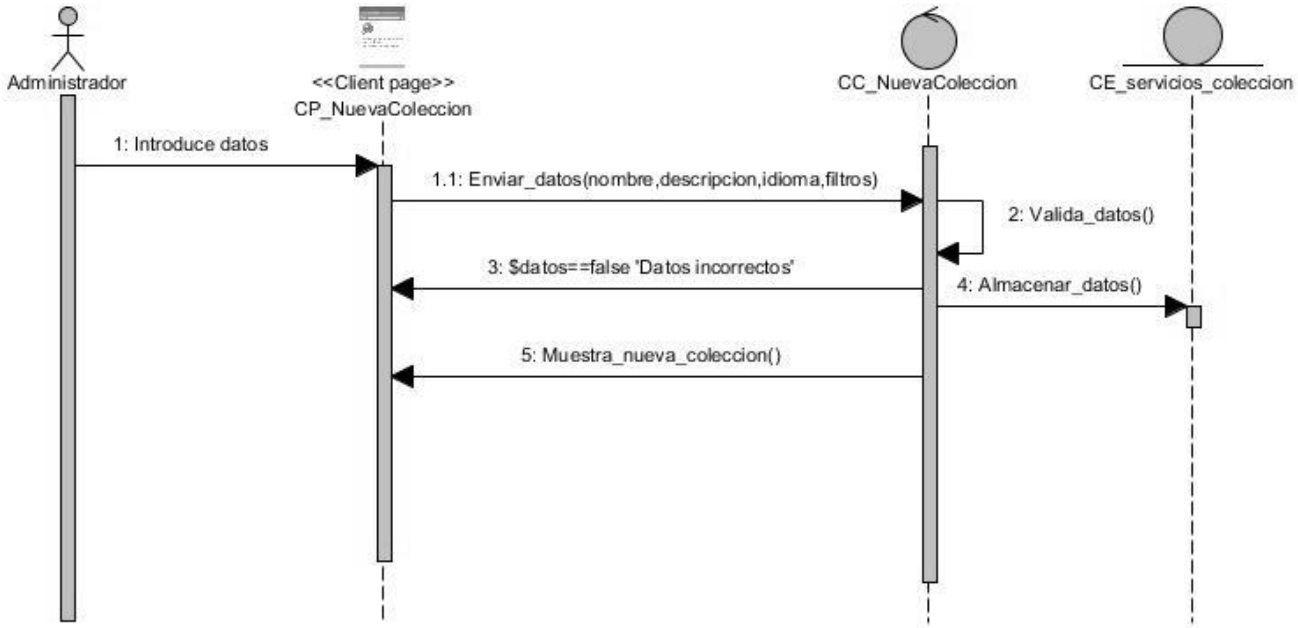

Figura 58 Diagrama de secuencia. Nueva colección

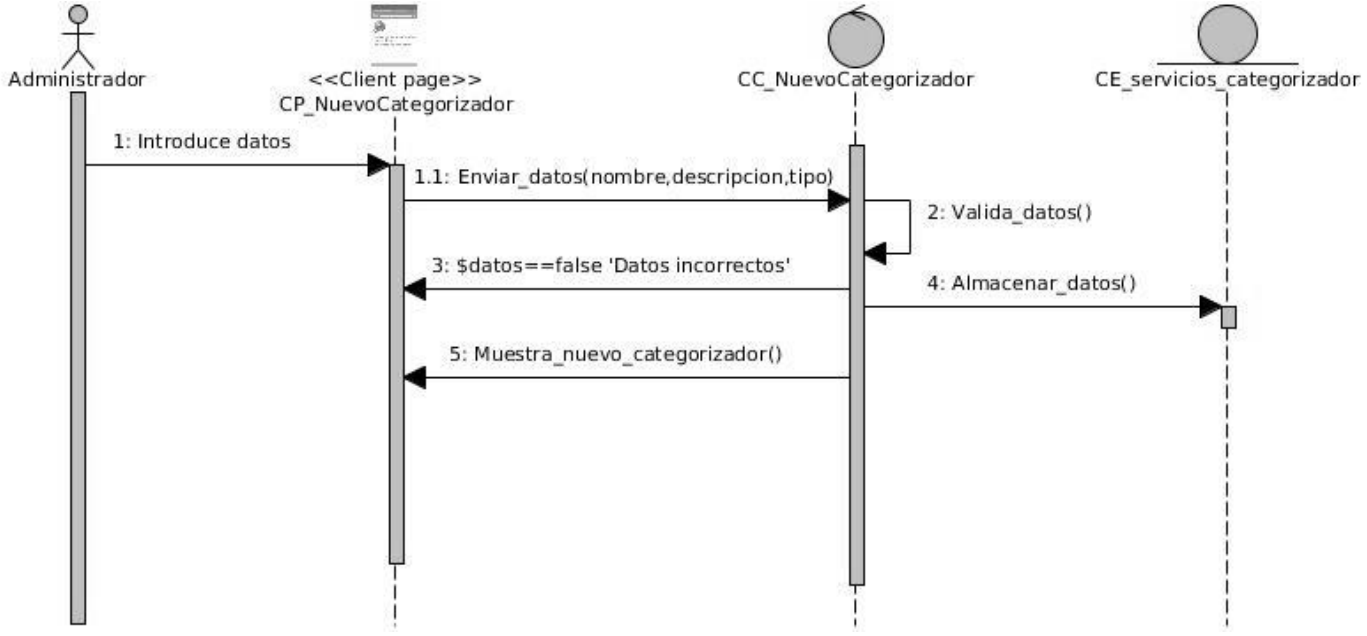

Figura 59 Diagrama de secuencia. Nuevo categorizador

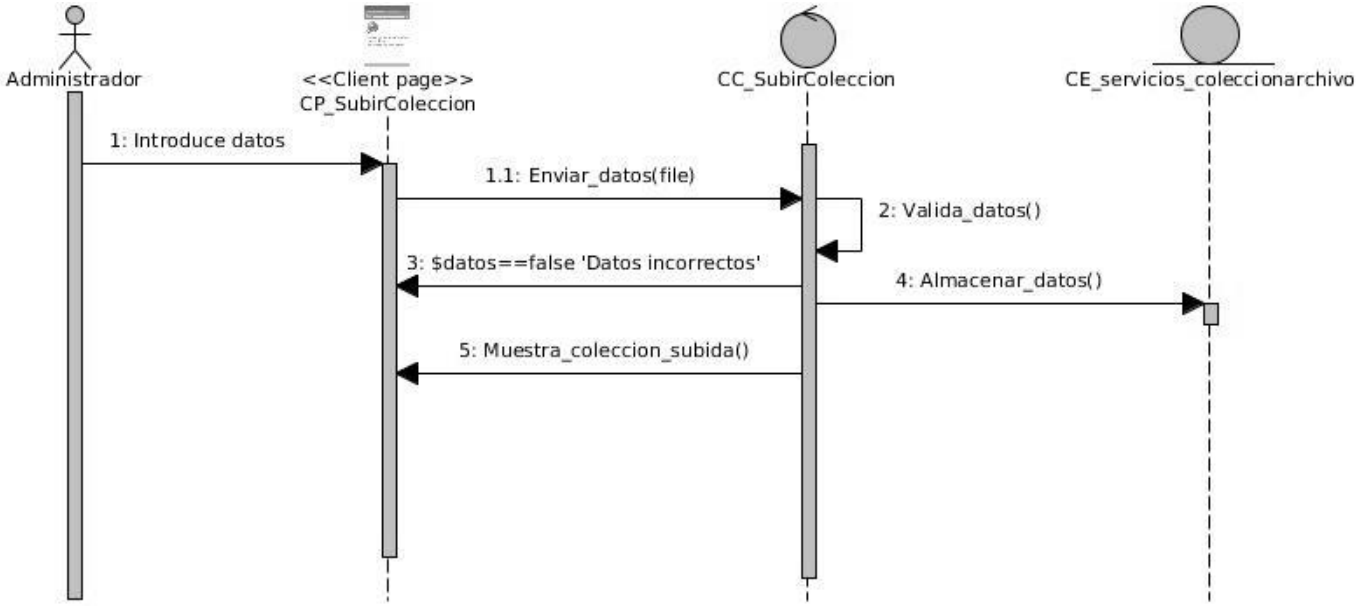

Figura 60 Diagrama de secuencia. Subir colección

Anexo 8: Descripción de las tablas de la Base de datos

Tabla servicios_categorizador: en esta tabla se guardan todos los categorizadores que tiene la capa de servicios *web* del categorizador de texto. Cuenta con los siguientes atributos: id, nombre del categorizador, descripción, fecha de creación, activo, entrenado e idioma. Tiene relación de uno a muchos con las tablas: servicios_categorizador_filtros, servicios_categorizador_categorías, servicios_categorizador_transformaciones.

Tabla servicios_filtros: en esta tabla se guardan todos los filtros que se le pueden aplicar a las colecciones y categorizadores existentes en la capa de servicios. Cuenta con los siguientes atributos: id, nombre de filtro, descripción, fecha de creación y activo. Tiene relación de uno a muchos con la tabla servicios_categorizador_filtros y servicios_colección_filtros.

Tabla servicios_categoría: en esta tabla se guardan todas las categorías que tienen los documentos existentes en cada una de las colecciones con las que se entrenan los categorizadores de la capa de servicios, las cuales utilizan los categorizadores para clasificar textos. Cuenta con los siguientes atributos: id, nombre de categoría, descripción, fecha de creación y activo. Tiene relación de uno a muchos con la tabla servicios_categorizador_ categorías y servicios_colección_categorías.

Tabla servicios_categorizador_filtros: en esta tabla se guardan los filtros que posee un categorizador entrenado con una determinada colección. Cuenta con los siguientes atributos: id, id_categorizador e id_filtro.

Tabla servicios_categorizador_categorías: en esta tabla se guardan todas las categorías pertenecientes a cada uno de los categorizadores que existe en la capa de servicios *web*. Cuenta con los siguientes atributos: id, id_categoría e id_categorizador.

Tabla servicios_categorizador_transformaciones: en esta tabla se guardan todas las transformaciones pertenecientes a cada uno de los categorizadores que existe en la capa de servicios *web*. Cuenta con los siguientes atributos: id, id_ transformación e id_categorizador.

Tabla servicios_categorizador_archivo: en esta tabla se guardan los archivos físicos de los categorizadores. Cuenta con los siguientes atributos: id, archivo. Tiene relación de uno a muchos con la tabla servicios_categorizador.

Tabla servicios_tipo_categorizador: en esta tabla se guardan los tipos de categorizadores que pueden existir en la capa de servicios. Cuenta con los siguientes atributos: id, tipo de categorizador, descripción, fecha de creación y activo. Tiene relación de uno a muchos con la tabla servicios_categorizador.

Tabla servicios_colección: en esta tabla se guardan todas las colecciones que tiene la capa de servicios *web* del categorizador de texto. Cuenta con los siguientes atributos: id, nombre, descripción, fecha de creación, activo, idioma, cantidad de documentos, cantidad de términos, archivo. Tiene relación de uno a muchos con las tablas: servicios_colección_filtros, servicios_categorizador, servicios_colección_categorías y servicios_colección_transformaciones.

Tabla servicios_colección_filtros: en esta tabla se guardan todos los filtros que posee una colección. Cuenta con los siguientes atributos: id, id_colección, id_filtro. Tiene relación de uno a muchos con la tabla servicios_filtros.

Tabla servicios_colección_categorías: en esta tabla se guardan todas las categorías pertenecientes a cada uno de los documentos que existe en las colecciones de la capa de servicios *web*. Cuenta con los siguientes atributos: id, id_categoría e id_colección.

Tabla servicios_transformación: en esta tabla se guardan todas las transformaciones que se le pueden aplicar a las colecciones y categorizadores existentes en la capa de servicios. Cuenta con los siguientes atributos: id, nombre de transformación, descripción, fecha de creación y activo. Tiene relación de uno a muchos con las tablas: servicios_categorizador_transformaciones y servicios_colección_tranformaciones.

Tabla servicios_colección_tranformaciones: en esta tabla se guardan todas las transformaciones que se le pueden aplicar a las colecciones. Cuenta con los siguientes atributos: id, id_transformación e id_colección.

Anexo 9: Diseño de casos de prueba

Escenario	Descripción	V_Examinar	Respuesta del sistema	Flujo central
EC 1.1 Subir una colección correctamente.	Mediante este escenario se carga una colección de documentos, en formato documentos.collection	V Colección.collection	El sistema procesa la información y sube la colección correctamente.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Subir Colección. -El sistema muestra una vista con un campo “Examinar” para cargar la colección de documentos. -El usuario carga la colección y da clic en el botón “Enviar”. -El sistema guarda la colección de documentos en la base de datos correctamente.
EC 1.2 Error al subir una colección.	Mediante este escenario se carga una colección de documentos, con la extensión incorrecta.	V Colección.svn	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Subir Colección. -El sistema muestra una vista con un campo “Examinar” para cargar la colección de documentos. -El usuario carga la colección con la extensión incorrecta y da clic en el botón “Enviar”. -El sistema envía un mensaje de error al subir una colección.

Tabla 32 Diseño de Casos de Prueba. Subir una Colección

Escenario	Descripción	V_Colección	V_Descripción	Respuesta del sistema	Flujo central
EC 1.1 Crear colección correctamente.	Mediante este escenario se crea una colección de documentos.	V Colection1	V Colección de documentos matemáticos.	El sistema procesa la información y crea la nueva colección correctamente.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Crear nueva colección. -El sistema muestra una vista con un campo “Colección” para especificar el nombre de la colección de documentos, y un campo “Descripción” para describir la nueva colección. -El usuario introduce los datos correctamente y da clic en el botón “Enviar”. -El sistema crea la colección de documentos en la base de datos correctamente.
EC 1.2 Error al crear colección.	Mediante este escenario no se especifica el nombre de la colección que se va a crear.	I	V Colección de documentos matemáticos.	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Crear nueva colección. -El sistema muestra una vista con un campo “Colección” para

					especificar el nombre de la colección de documentos, y un campo “Descripción” para describir la nueva colección. -El usuario introduce la descripción de la nueva colección y da clic en el botón “Enviar”. -El sistema muestra un mensaje de error al no haber introducido el usuario todos los datos para crear una nueva colección.
EC 1.3 Error al crear colección.	Mediante este escenario no se especifica la descripción de la colección que se va a crear.	V	I	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Crear nueva colección. -El sistema muestra una vista con un campo “Colección” para especificar el nombre de la colección de documentos, y un campo “Descripción” para describir la nueva colección. -El usuario introduce el nombre de la nueva colección y da clic en el botón “Enviar”.
		DocsMat			

					-El sistema muestra un mensaje de error al no haber introducido el usuario todos los datos para crear una nueva colección.

Tabla 33 Diseño de Casos de Prueba. Adicionar Colección

Escenario	Descripción	V_Examinar	V_Colección	Respuesta del sistema	Flujo central
EC 1.1 Adicionar documentos a la colección correctamente.	Mediante este escenario el usuario adiciona documentos a una colección.	V	V	El sistema procesa la información y adiciona los documentos a la colección correctamente.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Adicionar documentos Colección. -El sistema muestra una vista con un campo “Examinar” para cargar los documentos y un campo Colección para seleccionar una colección ya existente en la base de datos a la que desee adicionar documentos. -El usuario carga los documentos, selecciona la colección y da clic en el botón “Enviar”. -El sistema adiciona los documentos cargados a
		documentos.(txt, pdf, doc, docx)	Nombre de la colección		

					la colección seleccionada correctamente.
EC 1.2 Error al adicionar documentos a la colección.	En este escenario el usuario no carga documentos para adicionar a la colección.	I	V	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Adicionar documentos Colección. -El sistema muestra una vista con un campo “Examinar” para cargar los documentos y un campo Colección para seleccionar una colección ya existente en la base de datos a la que desee adicionar documentos. -El usuario selecciona la colección y da clic en el botón “Enviar”. -El sistema muestra un mensaje de error.
			Nombre de la colección		
EC 1.3 Adicionar colección correctamente.	Mediante este escenario se adiciona una colección de documentos.	V	I	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Adicionar Colección. -El sistema muestra una vista con un campo “Examinar” para cargar la colección de documentos, un campo
		documentos.(txt, pdf, doc, docx)			

					para especificar la categoría y el campo Colección para seleccionar una colección ya existente en la base de datos y que desee adicionar. -El usuario carga los documentos y da clic en el botón “Enviar”. -El sistema muestra un mensaje de error.

Tabla 34 Diseño de Casos de Prueba. Adicionar documentos a la colección

Escenario	Descripción	V_Usuario	V_Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Autenticar correctamente.	Mediante este escenario el usuario introduce los parámetros de conexión usuario y contraseña para realizar las peticiones a los servicios de la capa.	V Ariel	V Ariel	El sistema procesa la información crea la sesión para ese usuario.	-El sistema muestra la interfaz de autenticación. -El usuario una vez en la interfaz de autenticación introduce los parámetros de conexión. -El sistema verifica la validez de los datos introducidos y carga la página de los servicios. -El usuario una vez autenticado puede ejecutar los servicios que aparecen en dicha interfaz.
EC 1.2 Error de autenticación.	Mediante este escenario el usuario introduce los parámetros de	V Ariel	I ariel	El sistema procesa la información, y no encuentra usuario registrado con esos datos.	-El sistema muestra la interfaz de autenticación. -El usuario una vez en la interfaz de autenticación introduce los parámetros de conexión.

	conexión usuario y contraseña para realizar las peticiones a los servicios de la capa.			Muestra un mensaje “Por favor introduce un usuario y clave adecuado para una cuenta de personal. Fíjate que ambos campos son sensibles a mayúsculas.”	-El sistema verifica la validez de los datos introducidos y muestra un mensaje de error.

Tabla 35 Diseño de Casos de Prueba. Autenticar

Escenario	**Descripción**	**V_Examinar**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Extraer texto plano correctamente.	Mediante este escenario el usuario carga un documento para realizar la extracción del texto plano.	V documentos.(txt, pdf, doc, docx)	El sistema procesa la información y extrae el texto plano del documento.	-El usuario una vez autenticado, hace una petición al servicio Extraer texto plano. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. -El sistema extrae el texto plano del documento correctamente.
EC 1.2 Error al extraer texto plano.	Mediante este escenario el usuario carga un documento defectuoso para realizar la extracción del	Documento.pdf	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Extraer texto plano. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción.

	texto plano.			-El usuario carga el documento, y da clic en el botón “Enviar”. - El sistema procesa la información y muestra un mensaje de error especificando “El archivo está defectuoso”.
EC 1.3 Error al extraer texto plano.	Mediante este escenario el usuario carga un documento para realizar la extracción del texto plano con la extensión errónea.	Documento.html	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Extraer texto plano. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. - El sistema procesa la información y muestra un mensaje de error especificando que la extensión del documento cargado es errónea.
EC 1.4 Error al extraer texto plano.	Mediante este escenario el usuario no carga documento para realizar la extracción del texto plano.	I	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Extraer texto plano. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario da clic en el botón “Enviar”. -El sistema muestra un mensaje de error “Se deben seleccionar elementos para poder realizar acciones sobre estos. No se han modificado elementos”.

Tabla 36 Diseño de Casos de Prueba. Extraer texto plano.

Escenario	Descripción	V_Examinar	Respuesta del sistema	Flujo central
EC 1.1 Detectar Codificación correctamente.	Mediante este escenario el usuario carga un documento para realizar la detección de la codificación.	V documentos.(txt, pdf, doc, docx)	El sistema procesa la información y detecta la codificación del documento.	-El usuario una vez autenticado, hace una petición al servicio Detectar Codificación. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. -El sistema detecta la codificación del documento correctamente.
EC 1.2 Error al detectar la codificación.	Mediante este escenario el usuario carga un documento defectuoso para detectarla codificación.	Documento.pdf	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Detectar Codificación. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. - El sistema procesa la información y muestra un mensaje de error especificando “El archivo está defectuoso”.
EC 1.3 Error al detectar la codificación.	Mediante este escenario el usuario carga un documento para realizar la detección de la codificación con la extensión errónea.	Documento.html	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Detección de la codificación. -El sistema muestra una vista con un campo “Examinar” para cargar

				el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. - El sistema procesa la información y muestra un mensaje de error especificando que la extensión del documento cargado es errónea.
EC 1.4 Error al detectar la codificación.	Mediante este escenario el usuario no carga documento para realizar la detección de la codificación.	I	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Detectar codificación. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario da clic en el botón “Enviar”. -El sistema muestra un mensaje de error “Se deben seleccionar elementos para poder realizar acciones sobre estos. No se han modificado elementos”.

Tabla 37 Diseño de Casos de Prueba. Detectar codificación.

Escenario	**Descripción**	**V_Examinar**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Detectar idioma	Mediante este escenario el usuario carga un	V documentos.(txt,	El sistema procesa la información y detecta el idioma del	-El usuario una vez autenticado, hace una petición al servicio Detectar Idioma. -El sistema muestra una vista con un

correctamente.	documento para realizar la detección del idioma.	pdf, doc, docx)	documento.	campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. -El sistema detecta el idioma del documento correctamente.
EC 1.2 Error al detectar idioma.	Mediante este escenario el usuario carga un documento defectuoso para detectar el idioma.	Documento.pdf	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Detectar Idioma. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. - El sistema procesa la información y muestra un mensaje de error especificando “El archivo está defectuoso”.
EC 1.3 Error al detectar idioma.	Mediante este escenario el usuario carga un documento para realizar la detección del idioma con la extensión errónea.	Documento.html	El sistema procesa la información y muestra un mensaje de error.	-El usuario una vez autenticado, hace una petición al servicio Detección Idioma. -El sistema muestra una vista con un campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario carga el documento, y da clic en el botón “Enviar”. - El sistema procesa la información y muestra un mensaje de error especificando que la extensión del documento cargado es errónea.
EC 1.4 Error al detectar	Mediante este escenario el usuario no carga	I	El sistema procesa la información y muestra un mensaje	-El usuario una vez autenticado, hace una petición al servicio Detectar Idioma. -El sistema muestra una vista con un

Idioma.	documento para realizar la detección del idioma.		de error.	campo “Examinar” para cargar el documento y un botón “Enviar” para realizar la acción. -El usuario da clic en el botón “Enviar”. -El sistema muestra un mensaje de error “Se deben seleccionar elementos para poder realizar acciones sobre estos. No se han modificado elementos”.

Tabla 38 Diseño de Casos de Prueba. Detectar idioma

Escenario	**Descripción**	**V_nombre**	**V_descripcion**	**V_Tipo_Categorizador**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Crear categorizador correctamente.	Mediante este escenario se crea un nuevo categorizador, especificando los datos de entrada correctamente.	V	V	V	El sistema procesa la información y crea el categorizador correctamente.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Crear Categorizador. -El sistema muestra una vista con un campo “nombre” para especificar el nombre del categorizador que se va a crear, un campo “descripción” donde se describe el nuevo categorizador y un campo “Tipo Categorizador” para seleccionar el tipo de categorizador que se desea crear. -El usuario introduce correctamente los valores
		Nuevo_Categorizador	Categorizador para contenidos pornográficos.	SVM-1		

						y da clic en el botón “Enviar”. -El sistema procesa la información y crea el nuevo categorizador.
EC 1.2 Error al crear un nuevo categorizador.	Mediante este escenario no se introducen todos los datos para crear un nuevo categorizador.	I	NA	V SVM-1	El sistema procesa la información y envía un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Crear Categorizador. -El sistema muestra una vista con un campo “nombre” para especificar el nombre del categorizador que se va a crear, un campo “descripción” donde se describe el nuevo categorizador y un campo “Tipo Categorizador” para seleccionar el tipo de categorizador que se desea crear. -El usuario selecciona el tipo de categorizador que desea crear y da clic en el botón “Enviar”. -El sistema procesa la información y envía un mensaje de error al no introducir todos los datos necesarios.

EC 1.3 Error al crear un nuevo categorizador.	Mediante este escenario no se introducen todos los datos para crear un nuevo categorizador.	V	V	I	El sistema procesa la información y envía un mensaje de error.	-El usuario una vez en la interfaz de administración, hace una petición al servicio Crear Categorizador. -El sistema muestra una vista con un campo “nombre” para especificar el nombre del categorizador que se va a crear, un campo “descripción” donde se describe el nuevo categorizador y un campo “Tipo Categorizador” para seleccionar el tipo de categorizador que se desea crear. -El usuario introduce el nombre y la descripción correctamente y da clic en el botón “Enviar”. -El sistema procesa la información y envía un mensaje de error al no seleccionar el tipo de categorizador que se desea crear.
		Nuevo_Categorizador	Categorizador para contenidos pornográficos.			

Tabla 39 Diseño de Casos de Prueba. Crear categorizador

Anexo 10: Avales y Reconocimientos

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 1
2013-2014

RECONOCIMIENTO

Otorgado a: Capa de Servicios Web para el categorizador de texto del Valor de Clasificación Inteligente de Contenidos (VCIC).

Por haber obtenido DESTACADO

en El Trabajo en Comisiones.

12ma jornada científica estudiantil

Claudia Durán Rodríguez
Presidente de la FEU

UCI FACULTAD 1

Damián Cervantes Rondón
Decano de la Facultad