UNIVERSIDADE
DE VIGO

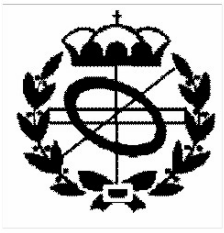
ESCOLA SUPERIOR DE ENXEÑERÍA INFORMÁTICA

Manual Técnico do Proxecto Fin de Carreira que presenta

Dna Sara Carrera Carrera

para a obtención do Título de **Enxeñeiro en Informática**

Adquisición semi-automática del conocimiento: una arquitectura preliminar



Xuño, 2007

Proxecto Fin de Carreira Nº: ENI 171

Director/a: Manuel Vilares Ferro
Área de coñecemento: Ciencias da Computación e Intelixencia Artificial
Departamento: Informática

"We must focus on seeding new trees, and hope that one day, when we will no longer be around, they will have mature, and bear new fruits..."

Gilles Khan.

AGRADECIMIENTOS

Ha llegado el momento de expresar mi gratitud a todas las personas que me han apoyado.

En primer lugar quiero dar las gracias a mi familia, especialmente a mis padres y a mi hermano. Siempre habéis estado cuando os he necesitado, me habéis proporcionado la mejor educación que cabe imaginar.

No quiero olvidarme de vosotras, Anita y Sabrina, por compartir conmigo tantos buenos momentos y por contar con vuestro apoyo cuando más lo he necesitado.

Existen unas personas que han sido las verdaderas protagonistas de mi vida a lo largo de estos años. Sois vosotras Isa, Mari, Rosa, Sandra y Tina. Gracias por compartir penas y alegrías.

A mis compañeros de fatigas, Miki, Santi, Iago, Sole, María. M., Alex y especialmente a ti María. B.

No me puedo olvidar de mis compañeros de laboratorio, Erica, Juan y Moli. Especialmente quiero dedicarle este proyecto a Mila, sin quien nada de esto hubiese sido posible. Gracias por la constante aportación de ideas y por el apoyo mostrado a lo largo de estos meses.

También dar las gracias a mi director Manuel Vilares por la oportunidad brindada al poder entrar en el laboratorio.

Finalmente estás tú, Celso. Tu apoyo incondicional es él que me da fuerzas cada día para seguir adelante. Estás ahí en los momentos flojos dándome tu apoyo y aportando una sonrisa siempre tan necesaria. En los bueno momentos sigues ahí, me acompañas. A ti quiero dedicar la culminación de mis años en la facultad.

A cada uno de vosotros gracias.

TABLA DE CONTENIDOS

1. IDENTIFICACIÓN DEL PROYECTO

titulo: Adquisición semi-automática del conocimiento: una arquitectura preliminar.

código: ENI - 171

alumna: Carrera Carrera, Sara

fecha: Junio 2007

director: Vilares Ferro, Manuel

área: Ciencias de la computación e inteligencia artificial

departamento: Informática

titulación: Ingeniería Informática

INTRODUCCIÓN

2. ORGANIZACIÓN DE LA DOCUMENTACIÓN

La documentación del proyecto se organiza en dos tomos, el Manual Técnico y la Memoria que incluye el Manual de Usuario.

En el Manual Técnico (documento actual) se abordan los siguientes temas:

- *Análisis del sistema*: Incluye la especificación de requisitos, el diagrama de clases del análisis junto a la descripción de las mismas, los detalles de los casos de uso así como los diagramas de secuencia del análisis.
- *Diseño del sistema*: Incluye los diagramas de clases de cada paquete así como la descripción de cada una de las clases, los patrones de diseño utilizados y el diseño de la base de datos.
- *Implementación del sistema*: En este apartado se incorporan todos aquellos detalles de implementación considerados relevantes.
- *Pruebas del sistema*: Finalmente se incorporan las pruebas llevadas a cabo para comprobar el correcto funcionamiento del sistema desarrollado.

En la Memoria se abordan los siguientes temas:

- *Introducción y situación del proyecto*: Se describe el problema abordado así como la solución desarrollada y los objetivos planteados.
- *Desarrollo del proyecto*: En esta sección se detalla la metodología de desarrollo utilizada así como aquellas tecnologías empleadas en la implementación.
- *Recursos de PLN*: En este apartado se hace una breve introducción al procesamiento del lenguaje natural (PLN) y se describen las características de las herramientas de PLN empleadas en el sistema.
- *Ontologías*: Incluye una descripción del término ontología y sus aspectos fundamentales.

- *Apéndice Técnico*: En el apéndice se describen dos tecnologías empleadas en el desarrollo que son *Lucene* y *OWL*.

En el Manual de Usuario se incorpora toda la información para la puesta en marcha de la aplicación, incorporando el manual de instalación, el Manual de Usuario para el manejo de la aplicación creada y el manual del usuario experto para poder llevar a cabo ciertas tareas que requieren unos conocimientos avanzados.

Además de la documentación aportada a través de los documentos descritos se incluye también un disco. Este disco tiene la documentación en formato digital así como el código fuente para instalar la aplicación. Se añaden también las herramientas necesarias para la puesta en marcha del sistema.

3. ESPECIFICACIÓN DE REQUISITOS

El objetivo de la especificación de requisitos es definir de manera clara y precisa todas las funcionalidades y restricciones del sistema construido texto

3.1.FUNCIONALIDADES DEL SISTEMA

Como ya se ha comentado en la Memoria de este proyecto el objetivo principal es el desarrollo de una arquitectura genérica que permita, a partir de un conjunto de fuentes textuales, la extracción de conocimiento. Éste se representará en forma de términos clave y en forma de dependencias entre palabras, ambos enfoques estructurados en forma de ontología.

El sistema debe cumplir una serie de requisitos que satisfagan las funcionalidades especificadas para los usuarios así como para los documentos almacenados en el mismo.

El sistema está enfocado hacia dos tipos de usuario, un usuario que cabe considerarlo como usuario normal y otro considerado como usuario experto. Cabe destacar, que el usuario experto está en condiciones de realizar todas aquellas operaciones que el usuario normal puede satisfacer. Se detalla más adelante las diferencias entre ambos. A continuación se detallan los requisitos en cuanto a la información almacenada y manejada

3.1.1.CONFIGURACIÓN Y USUARIOS

La configuración del sistema se encuentra definida en un fichero (ver apartado: *10.2 interpretación del fichero de configuración*). Es a través del mismo que la aplicación recupera todos los parámetros necesarios para efectuar las operaciones peticionadas por los usuarios.

El sistema permite además modificar la configuración a través de la propia interfaz:

- Se puede añadir un nuevo tema de corpus al sistema. Para ello el usuario indicará el nombre del tema que será único en el sistema así como los idiomas utilizados en ese tema. Por lo tanto los idiomas deberán estar definidos previamente. Añadir un nuevo tema implica crear una referencia a la base de datos asociada a cada idioma.
- Se puede borrar un tema del sistema.
- Se puede añadir un nuevo idioma al sistema. El usuario indicará el nombre del idioma y la abreviatura del mismo, ambos elementos serán únicos en el sistema. Además deberá decir si para ese idioma existe un fichero de *lista de parada* que será empleado en la búsqueda y recuperación de documento.
- Se puede borrar un idioma del sistema.
- Se puede añadir un nuevo extractor al sistema. El usuario indicará el nombre del extractor que será único, señalará el idioma asociado y deberá indicar el recurso que lleva a cabo esta tarea.
- Se puede borrar un extractor.
- Se puede añadir un nuevo analizador sintáctico al sistema. El usuario indicará el nombre del analizador sintáctico que será único, señalará el idioma asociado y deberá indicar el recurso que lleva a cabo esta tarea.
- Se puede borrar un analizador sintáctico.

En este punto se puede definir la diferencia principal entre los dos tipos de usuarios definidos en el sistema. Relacionado con la adición de un nuevo extractor o bien de un nuevo analizador sintáctico, si bien ambos usuarios pueden introducir los datos en el formulario, se considera que sólo el usuario experto dispondrá de los conocimientos necesarios para desarrollar el recurso necesario para llevar a cabo la tarea. El usuario normal sólo necesitará disponer de aquellos conocimientos que le permitan navegar por la interfaz definida para el sistema.

3.1.2.CORPUS

En el sistema pueden convivir distintos corpus. Los mismos vienen definidos en función del tema que describen los documentos base del corpus y en función de la lengua en la qué estén escritos esos documentos. Por lo tanto, para un determinado tema se puede tener más de un idioma y un idioma puede estar asociado a diversos temas.

El sistema está entonces enfocado y guiado en cada una de sus tareas en función del tema e idioma elegido en cada caso.

3.1.3.INFORMACIÓN

- En el sistema se almacenarán documentos. Estos documentos podrán ser introducidos por cualquiera de los usuarios.
    - Cada nueva colección introducida deberá ser asociada a un corpus concreto así como a un idioma. Es decir, un nuevo documento introducido pertenecerá a un determinado tema, previamente definido, y a un idioma asociado a ese tema.
- Una vez se introduce un nuevo conjunto de documentos estos son tratados por el sistema:
    - A partir de los mismos se construye un corpus. Por consiguiente, existirá un corpus por cada tema e idioma.
    - Existe un índice de búsqueda asociado a los documentos, que se actualiza cada vez que el sistema recibe nuevas fuentes textuales. Este índice tiene la peculiaridad de ser creado con los documentos lematizados. Es decir, para cada información nueva introducida en el índice, se realiza previamente un proceso de lematización. Se toma esta decisión observando el siguiente hecho: Las ontologías van a trabajar en todo momento con los lemas del corpus. Sabiendo que las ontologías se van a utilizar en la recuperación de documentos relevantes, es necesario, por lo tanto, que los documentos que conforman el índice también estén lematizados, de este modo la recuperación de una consulta será factible.
    - Es necesario, además para un subconjunto de las funcionalidades llevadas a cabo en la aplicación, que se realice un análisis sintáctico de cada nueva fuente introducida. Se mantiene de esta forma

actualizada la base de datos que contiene el grafo de dependencias de un determinado corpus. Existe por lo tanto una base de datos por cada tema e idioma.

- Dado el idioma elegido para introducir los nuevos documentos, el sistema presentará los analizadores sintácticos disponibles, el usuario elegirá uno de ellos.

❖ Asociados a un corpus y a un idioma es posible realizar un conjunto de actividades, que pueden ser ejecutadas por cualquiera de los usuarios:

- Extracción de términos clave dentro del dominio definido por los documentos de ese corpus e idioma. Según el idioma elegido el sistema muestra los extractores disponibles, el usuario podrá elegir más de uno para realizar la operación.
  - Clasificación de estos términos en clave en base a un algoritmo definido concretamente para el sistema.
  - A partir de esa clasificación es posible crear una ontología que almacene las clases creadas. De esta forma se puede recuperar esta ontología o utilizarla en otras tareas descritas más adelante.
- Jerarquización e interrelación de los elementos que forman el grafo de dependencias.
  - Al igual que con la clasificación de términos, es posible partir de esta jerarquización crear una ontología. Este es un proceso iterativo, en el sentido que el usuario cada vez que finaliza el proceso puede observar los resultados obtenidos y modificarlos según considere oportuno, (este hecho se detalla en el apartado 13 de implementación).

❖ Los usuarios (bien sean normales bien sean expertos) tendrán acceso a los documentos introducidos en la aplicación mediante la realización de consultas contra los mismos. Es decir, se define una búsqueda y recuperación de documentos relevantes a la consulta del usuario. La peculiaridad de esta tarea en el sistema es que cabe la posibilidad de ampliar las consultas del usuario con los conceptos declarados en las ontologías previamente generadas.

- Puesto que la expansión de la consulta del usuario se define en base a las ontologías creadas, la búsqueda de documentos se enfoca en dos vertientes, la búsqueda en base a términos y la búsqueda en base a dependencias.

- Las búsquedas incluyen un proceso previo de lematización de la consulta.
- La búsqueda por dependencias va a permitir recuperar documentos en varios idiomas. El usuario habrá proporcionado, además de la consulta, un tema e idioma, entonces el sistema traducirá dicha consulta al resto de idiomas del tema actual. Una vez se tengan las consultas traducidas es posible ampliarlas con las ontologías correspondientes.

ANÁLISIS

4. ESPECIFICACIÓN DE LAS CLASES

4.1.DIAGRAMA DE CLASES

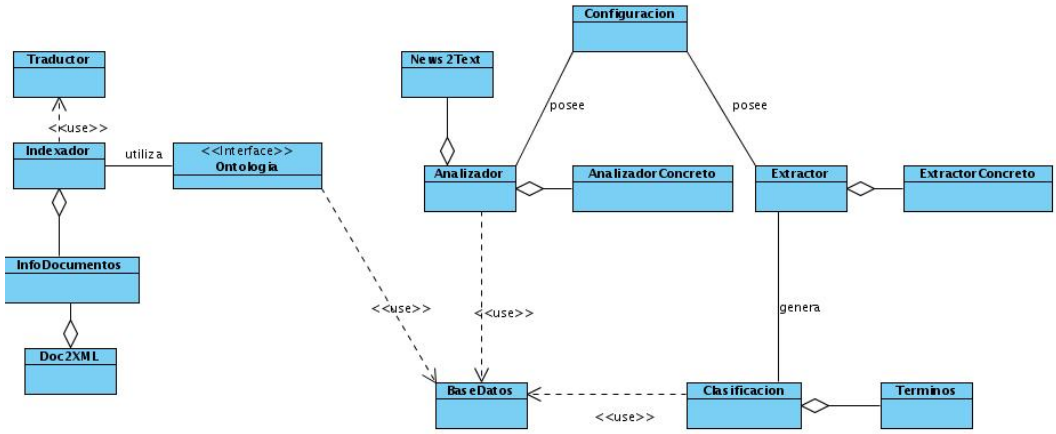


Figura 1 diagrama de clases del análisis

4.2.DICCIONARIO DE CLASES

CONFIGURACIÓN

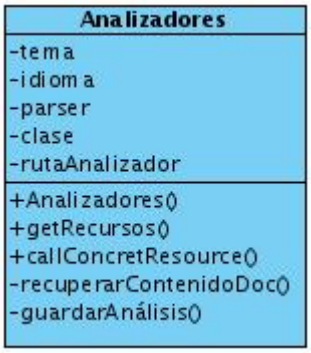
CONFIGURACIÓN	
Configuracion -abreviaturas -analizadores -extractores -idiomas -temas +set/getAnalizadores() +set/getExtractores() +set/getTemas() +set/getIdiomas() +getTemaVSabr() +getIdVSabr() +anadirTema() +borrarTema() +anadirIdioma() +borrarIdioma() +anadirExtractor() +borrarExtractor() +anadirAnalizador() +borrarAnalizador()	Descripción Esta clase gestiona la información relativa a las distintas herramientas disponibles en el sistema para realizar la extracción y el análisis sintáctico. Mantiene además la información relativa a los idiomas y temas disponibles.

Atributos	abreviaturas: Vector con las abreviaturas existentes en el sistema que se corresponden con un idioma. analizadores: Vector con los analizadores existentes en el sistema. extractores: Vector con los extractores existentes en el sistema. idiomas: Vector con los idiomas existentes en el sistema. temas: Vector con los temas existentes en el sistema.
Métodos	set/getAnalizadores: Estos métodos inicializan y devuelven los analizadores existentes en el sistema. set/getExtractores: Estos métodos inicializan y devuelven los extractores existentes en el sistema. set/getTemas: Estos métodos inicializan y devuelven los temas existentes en el sistema. set/getIdiomas: Estos métodos inicializan y devuelven los idiomas existentes en el sistema. anadirTema: Este método añade un tema al sistema. borrarTema: Este método borra un tema del sistema. anadirIdioma: Este método añade un idioma al sistema. borrarIdioma: Este método borra un idioma del sistema. anadirExtractor: Este método añade un extractor al sistema. borrarExtractores: Este método borra un extractor del sistema. anadirAnalizador: Este método añade un analizador al sistema. borrarAnalizador: Este método borra un analizador del sistema.

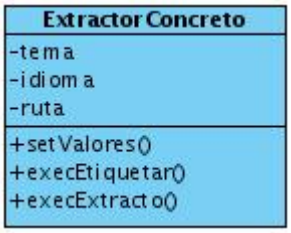
EXTRACTORES

EXTRACTORES	
	Descripción En esta clase se mantiene la información relativa a los extractores y se gestionan las llamadas a los mismos así como los resultados obtenidos.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. extractores: Vector con los extractores que se van a ejecutar en este instante. clase: Clase asociada al recurso a ejecutar. rutaExtractor: Almacena la ruta donde se almacena el extractor, en caso de que éste sea implementado por una herramienta independiente.
Métodos	Extractores: Inicializa los valores de tema, idioma y extractores. getRecursos: Localiza la clase asociada al recurso a ejecutar así como la ruta donde se ubica el extractor. También recupera el corpus. callConcretResource: Llama al extractor concreto que ha solicitado el usuario. guardaExtraccion: Guarda los resultados devueltos por la extracción. Se encarga además de eliminar elementos repetidos en los términos extraídos.

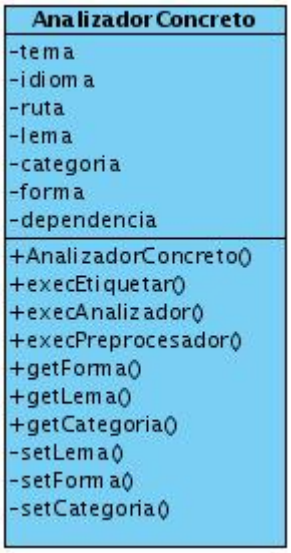
ANALIZADORES

ANALIZADORES	
	Descripción En esta clase se mantiene la información relativa a los analizadores sintácticos y se gestionan las llamadas a los mismos así como los resultados obtenidos.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. analizador: Almacena el analizador actual. clase: Clase asociada al recurso a ejecutar. rutaAnalizador: Almacena la ruta donde se almacena el analizador, en caso de que éste sea implementado por una herramienta independiente.
Métodos	Analizadores: Inicializa los valores de tema, idioma y analizador. getRecursos: Localiza la clase asociada al recurso a ejecutar así como la ruta donde se ubica el analizador. callConcretResource: Llama al analizador concreto que ha solicitado el usuario. recuperarContenidoDoc: Recupera del documento a analizar las partes relevantes, el título y el texto. guardarAnálisis: Almacena en el resultado del análisis sintáctico.

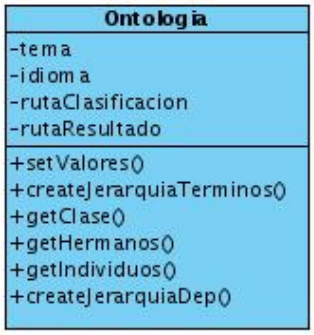
EXTRACTOR CONCRETO

EXTRACTOR CONCRETO	
	Descripción Esta clase manipula la información de los extractores introducidos en el sistema. Contiene aquellas operaciones necesarias para ejecutar estas herramientas.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. ruta: Almacena la ruta donde se ubica el recurso.
Métodos	setValores: Inicializa los atributos de la clase. execEtiquetar: Ejecuta la etiquetación. execExtractor: Ejecuta el extractor.

ANALIZADOR CONCRETO

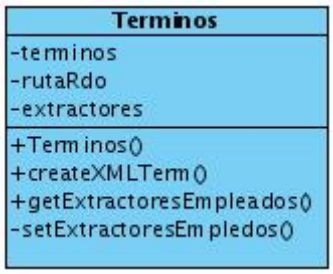
ANALIZADOR CONCRETO	
	Descripción Esta clase manipula la información de los analizadote sintácticos introducidos en el sistema. Contiene aquellas operaciones necesarias para ejecutar estas herramientas.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. ruta: Almacena la ruta donde se ubica el recurso. lema: Almacena el lema actual. categoria: Almacena la categoría actual. forma: Almacena la forma actual. dependencia: Almacena la dependencia actual.
Métodos	setValores: Constructor de la clase que inicializa los atributos. execEtiquetar: Ejecuta la etiquetación. execAnalizador: Ejecuta el analizador. execPreprocesador: Ejecuta el pre-porcesador. set/getLema: Estos métodos inicializan y devuelven el lema de la palabra actual. set/getCategoria: Estos métodos inicializan y devuelven la categoría de la palabra actual. set/getForma: Estos métodos inicializan y devuelven la forma de la palabra actual. set/getDependencia: Estos métodos inicializan y devuelven la dependencia actual.

ONTOLOGÍA

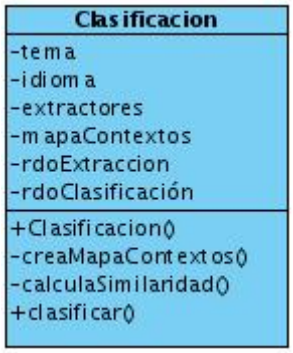
ONTOLOGÍA	
	Descripción Esta clase manipula la información necesaria para crear y explotar las ontologías generadas a partir de la extracción de términos.

Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. rutaClasificación: Almacena la ruta donde se guarda la clasificación actual. rutaResultados: Ruta para guardar la ontología generada.
Métodos	setValores: Inicializa los valores de los atributos. createJerarquiaTerm: Genera la jerarquía de clases e individuos resultantes de la clasificación de términos. createJerarquiaDep0: Genera la jerarquía de clases e individuos resultantes de la clasificación de dependencias. getClase: Devuelve una determinada clase. getIdividuos: Devuelve las instancias de una clase. getHermanos: Devuelve las clases hermanas de una determinada clase.

TÉRMINOS

TERMINOS	
	Descripción Con esta clase se crea la estructura necesaria para almacenar los resultados de la adquisición de terminología.
Atributos	terminos: Vector con los términos extraídos. extractores: Vector con los extractores implicados. rutaRdo: Ruta donde ubicar el resultado.
Métodos	Terminos: Constructor de la clase. getExtractoreEmpleados: Devuelve los extractores implicados en la extracción. setExtractoreEmpleados: Inicializa el vector de extractores implicados en la extracción. createXMLTerm: Genera el XML con los términos extraídos.

CLASIFICACIÓN

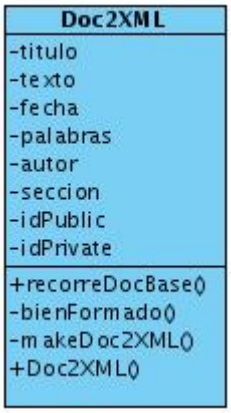
CLASIFICACIÓN	
	Descripción Esta clase manipula los datos resultantes de la extracción de términos para generar una clasificación de los mismos.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. extractores: Vector con los extractores implicados. mapaContextos: Tabla que almacena los contextos.

	rdoExtraccion: Ruta donde ubicar el resultado de la extracción. rdoClasificación: Ruta donde almacenar la clasificación.
Métodos	Clasificacion: Constructor de clase. creaMapaContextos: Recupera los términos y las palabras clave de los mismo. Además, genera una tabla con los contextos asociados a palabras clave. calculaSimilaridad: Calcula la similaridad entre los contextos de elementos distintos. clasificar: Lanza la clasificación.

BASE DATOS

BASE DATOS	
	Descripción Se recoge y trata la información que se almacena en la base de datos. Esta información es el resultado proporcionado por el análisis sintáctico.
Atributos	nbBaseDatos: Nombre de la base de datos. gestorBD: Nombre del sistema gestor de la base de datos. tema: Almacena el tema actual. idioma: Almacena el idioma actual.
Métodos	establecerConexion: Crear la conexión con la base de datos. crearBD: Crear la base de datos. BaseDatos: Constructor de clase. lematizarDocumento: Lematizar un documento, a partir de los resultados generados en el análisis sintáctico. set/getLema: Estos métodos inicializan y devuelven el lema de la palabra actual. set/getCategoria: Estos métodos inicializan y devuelven la categoría de la palabra actual. set/getForma: Estos métodos inicializan y devuelven la forma de la palabra actual. set/getDependencia: Estos métodos inicializan y devuelven la dependencia actual.

DOC2XML

DOC2XML	
	Descripción Con esta clase se tratan los documentos introducidos por el usuario para almacenarlos en el sistema en un formato estándar.
Atributos	titulo: Titulo del documento actual. texto: Texto del documento actual. fecha: Fecha indicada en el documento actual. palabras: Número de palabras en el documento actual. autor: Autor del documento actual. seccion: Sección o periódico que publica la noticia recogida en el documento actual. idPublic: Identificador público que viene dado por un valor único. idPrivate: Identificador privado que viene dado por un valor único.
Métodos	Doc2XML: Constructor de la clase. recorreDocBase: Recorre el documento base proporcionado por el usuario para recoger las distintas secciones y poder generar el XML correspondiente. bienFormado: Comprueba que el documento base proporcionado por el usuario se corresponda con la estructura esperada. makeDoc2XML: Una vez inicializados las distintas secciones, este método guarda cada una de ellas en una estructura XML.

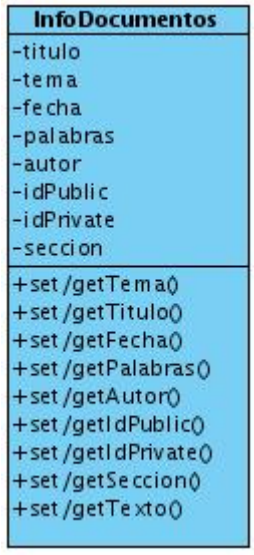
NEWS2TEXT

NEWS2TEXT	
	Descripción Esta clase permite extraer de los documentos a analizar sintácticamente aquella información pertinente.
Atributos	idioma: Almacena el idioma actual. rutaXML: Almacena el lugar donde se ubica el fichero XML que se está analizando. rutaNews: Almacena la ruta donde guardar el fichero temporal con el contenido relevante.
Métodos	News2Text: Consturctor de la clase. setContenidoNoticia: Recupera del documento XML las secciones consideradas con contenido, éstas son, el título y el texto. guardaContenido: Almacena el contenido relevante en un fichero temporal.

INDEXADOR

INDEXADOR	
	Descripción Esta clase permite gestionar toda la información relativa al índice relativo a un conjunto de documentos de un tema e idioma. Ofrece además la posibilidad de realizar búsquedas sobre los documentos.
Atributos	pathOnto: Ruta donde se ubica la ontología. pathIndexador: Ruta donde se ubica el índice. tema: Almacena el tema actual. idioma: Almacena el idioma actual.
Métodos	setValores: Inicializa los atributos. crearIndice: Crea el índice asociado a un número de documentos iniciales. ampliarConsulta: Amplia la consulta introducida por el usuario con los valores reflejados en la ontología. crearDocumento: A partir del documento XML extrae las partes pertinentes a introducir en el índice. A continuación indexa el documento con el índice ya creado. recuperarDocumento: A través de una consulta al índice se recuperan los documentos que cumplen con la consulta del usuario.

INFODOCUMENTOS

INFODOCUMENTOS	
	Descripción Esta clase recoge y devuelve todos aquellos valores almacenados en los documentos fuente del texto.
Atributos	titulo: Almacena el titulo actual. tema: Almacena el tema actual. fecha: Almacena la fecha actual. palabras: Almacena el número de palabras actual. autor: Almacena el autor actual. idPublic: Almacena el identificador público actual.

	idPrivate: Almacena el identificador privado actual. seccion: Almacena el periódico o sección actual.
Métodos	set/getTema: Estos métodos inicializan y devuelven el tema del documento actual. set/getTexto: Estos métodos inicializan y devuelven el texto del documento actual. set/getTitulo: Estos métodos inicializan y devuelven el titulo del documento actual. set/getFecha: Estos métodos inicializan y devuelven la fecha del documento actual. set/getPalabras: Estos métodos inicializan y devuelven el número de palabras del documento actual. set/getAutor: Estos métodos inicializan y devuelven el autor del documento actual. set/getIdPublic: Estos métodos inicializan y devuelven el identificador público del documento actual. set/getIdPrivate: Estos métodos inicializan y devuelven el identificador privado del documento actual. set/getSeccion: Estos métodos inicializan y devuelven el periódico o sección del documento actual.

TRADUCTOR	
	Descripción
	Esta clase llama a un traductor externo al sistema, para recuperar la traducción de la consulta en los idiomas correspondiente.
Atributos	
Métodos	traducirConsulta: Traduce la consulta que se le proporciona como parámetro.

5. CASOS DE USO

Se presentan a continuación el conjunto de casos de uso implicados en el sistema que aquí se presenta.

Cabe en este punto una aclaración. A pesar de considerar que el sistema está enfocado a dos tipos de usuario, en los diagramas de caso de uso realizados se ha representado únicamente el usuario normal. Se toma esta decisión pensando el objetivo que tiene la elaboración de los casos de uso. Un caso de uso representa una secuencia de acciones efectuadas por el sistema que producen un resultado observable para el usuario. Cualquiera de las tareas con las que el usuario quiera interaccionar con el sistema, implican solamente un conocimiento básico de navegación por un programa. De este modo, para la adición de nuevas herramientas, tales como extractores y analizadores, el usuario solamente deberá proporcionar una serie de datos a través de un formulario y proporcionar el recurso (es decir, la clase) asociado con la nueva herramienta. Ahora bien, es importante dejar claro que el recurso que se aporta al sistema necesita ser elaborado por una persona experta y con los conocimientos adecuados.

En primer lugar nos centramos en el caso de uso general de la arquitectura. En los siguientes apartados se desglosa el mismo y se aporta una explicación a cada uno de ellos.

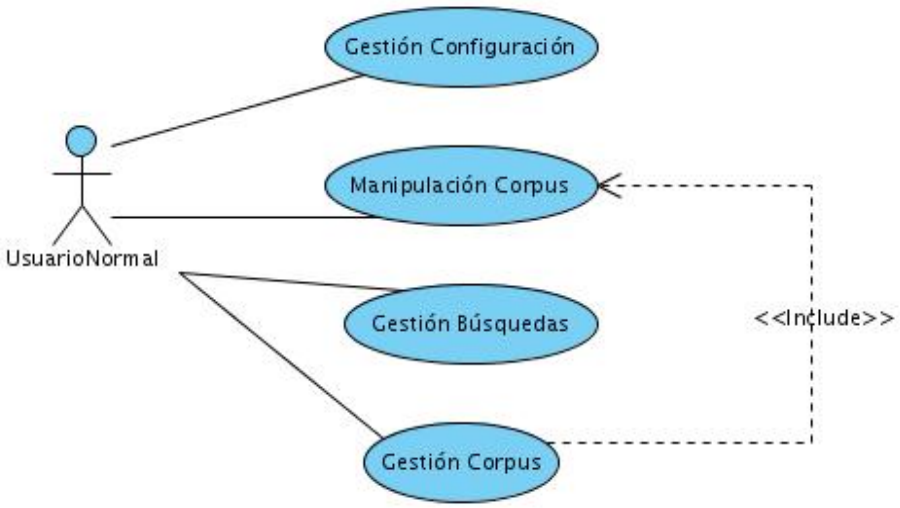


Figura 2 caso de uso general

5.1.CASO DE USO GESTIÓN CONFIGURACIÓN

Se describe en esta sección como se gestionan aquellos elementos relacionados directamente con el corpus. Por un lado se presenta la gestión de los temas y de los idiomas; y por otro la gestión de los recursos que explotan el corpus, los extractores y analizadores sintácticos. Se muestra aquí como el usuario a través de la interfaz puede modificar las distintas partes del sistema. También, es posible acceder manualmente al fichero de configuración y realizar los cambios que el usuario desee.

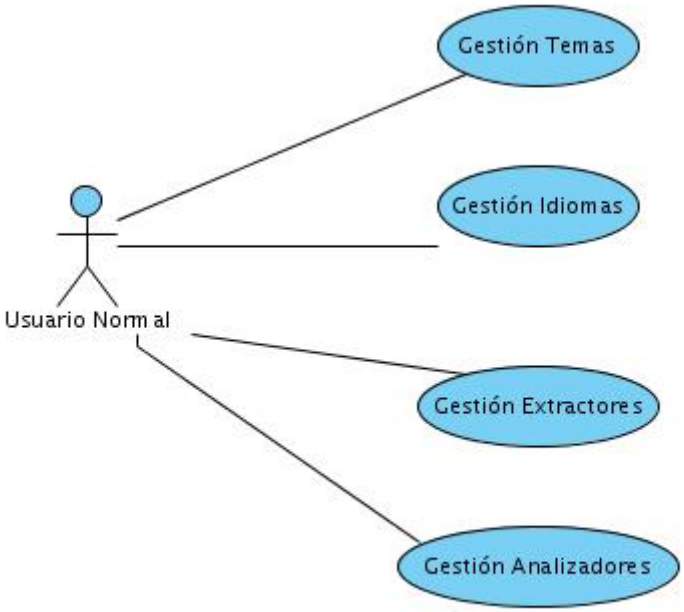


Figura 3 caso de uso gestión configuración

5.1.1.GESTIÓN TEMAS (ESCENARIO 1): AÑADIR TEMA

Gestión Temas (Escenario 1) : Añadir un tema	
DESCRIPCIÓN	Se describe en este escenario cómo se añade un nuevo tema al sistema.
PRECONDICIONES	Que exista el fichero de configuración. Que ya esté definido algún idioma.
FLUJO PRINCIPAL	El usuario accede a añadir un tema. El usuario introduce el nombre del tema [1]. El usuario elige el conjunto de idiomas que desea se asocien al tema. El sistema procede a efectuar los cambios en el fichero de configuración, añadiendo el nuevo tema con sus idiomas asociados. Crea además para cada uno de los idiomas la referencia a la base de datos asociada al idioma y tema.
FLUJO EXCEPCIONAL	[1] Si el nombre ya existe, el sistema informa de ello y el usuario debe introducir otro nombre.
POSTCONDICIONES	En el sistema existe un nuevo tema al cual los usuarios pueden acceder para realizar las distintas tareas que ofrece el sistema.

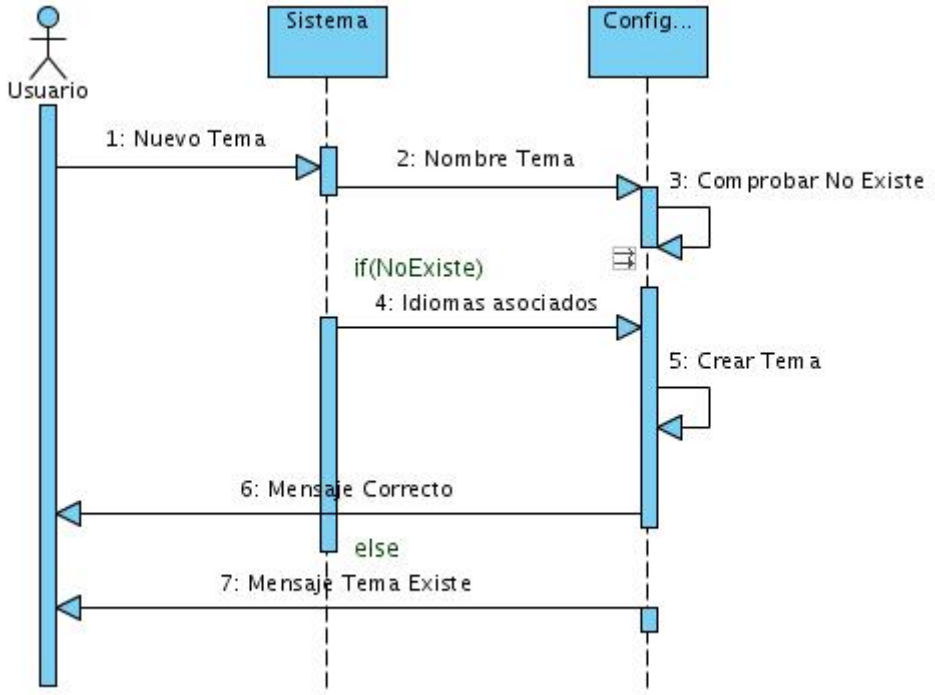


Figura 4análisis diag.sec añadir tema

5.1.2.GESTIÓN TEMAS (ESCENARIO 2): BORRAR TEMA

Gestión Temas (Escenario 2) : Borrar un tema	
DESCRIPCIÓN	En este escenario se describe cómo un tema puede ser eliminado del sistema.
PRECONDICIONES	Que exista el fichero de configuración. Que exista el tema que el usuario desea eliminar.
FLUJO PRINCIPAL	El usuario elige a través de la interfaz el tema que desea eliminar. El usuario elige eliminar el tema.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	En el sistema deja de existir el tema en cuestión.

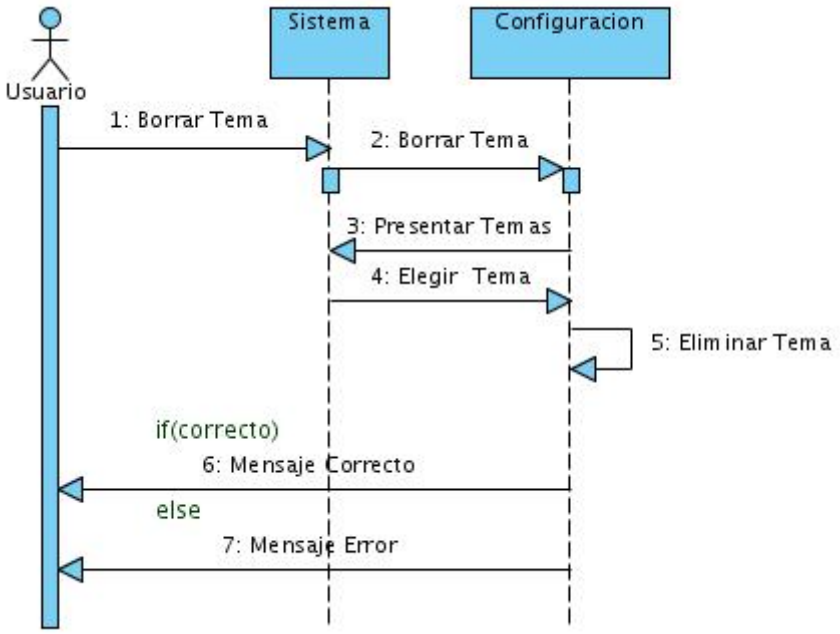


Figura 5 análisis diag. sec. borrar tema

5.1.3.GESTIÓN IDIOMAS (ESCENARIO 1): AÑADIR IDIOMA

Gestión Idiomas (Escenario 1) : Añadir un idioma	
DESCRIPCIÓN	Se describe en este escenario cómo se añade un nuevo idioma al sistema
PRECONDICIONES	Que exista el fichero de configuración
FLUJO PRINCIPAL	El usuario accede a añadir un idioma. El usuario introduce el nombre del idioma [1]. El usuario introduce la abreviatura asociada al idioma [2]. El usuario debe indicar si existe para ese idioma un fichero que contiene una lista de *palabras de parada* que se utilizará en el proceso de indexación. [3] El sistema procede a efectuar los cambios en el fichero de configuración, añadiendo el nuevo idioma con su abreviatura asociada. Se generan además nodos vacíos en la rama de extractores y en la rama de analizadores sintácticos para recoger los futuros recursos relacionados con el idioma.
FLUJO EXCEPCIONAL	[1] Si el nombre ya existe, el sistema informa de ello y el usuario debe introducir otro nombre. [2] Si la abreviatura ya existe el sistema informa de ello y el usuario debe introducir otro valor. [3] Si el fichero de *palabras de parada* no se encuentra en el directorio correspondiente el sistema informa de ello.
POSTCONDICIONES	En el sistema existe un nuevo idioma al cual los usuarios pueden acceder para realizar las distintas tareas que ofrece el sistema.

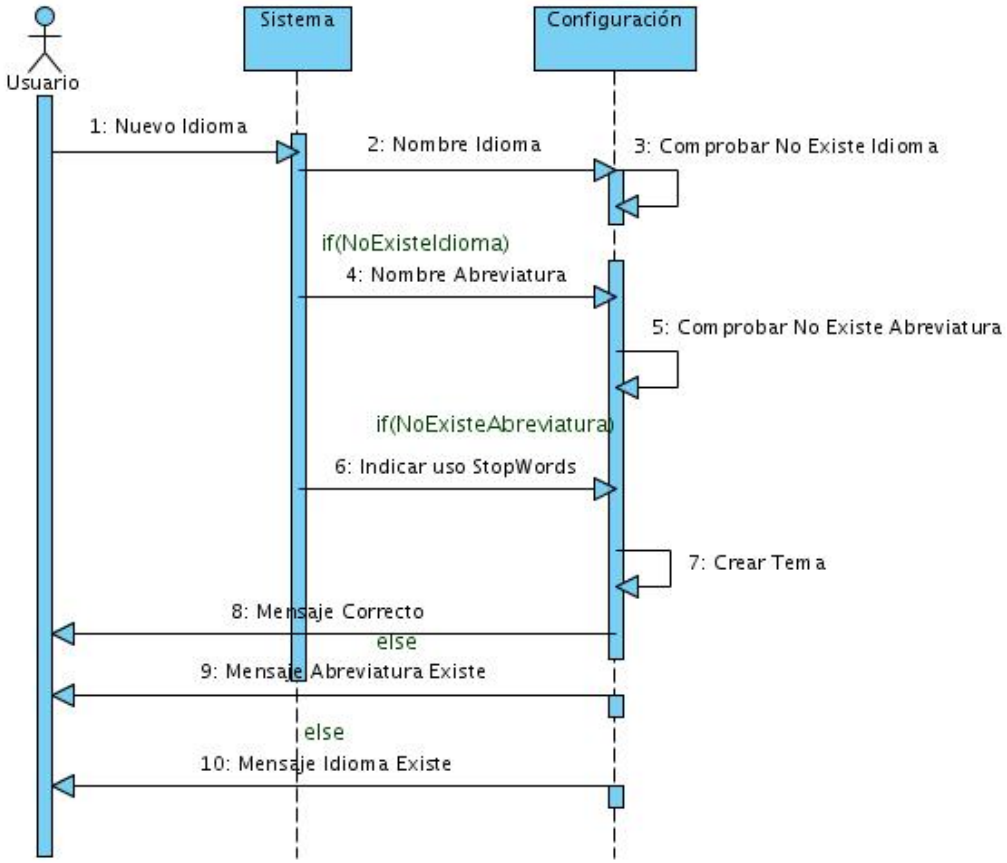


Figura 6 análisis diag. sec. añadir idioma

5.1.4.GESTIÓN IDIOMAS(ESCENARIO 2): BORRAR IDIOMA

Gestión Idiomas (Escenario 2) : Borrar un idioma	
DESCRIPCIÓN	En este escenario se describe cómo un idioma puede ser eliminado del sistema.
PRECONDICIONES	Que exista el fichero de configuración. Que exista el idioma que el usuario desea eliminar.
FLUJO PRINCIPAL	El usuario elige a través de la interfaz el idioma que desea eliminar. El usuario elige eliminar el idioma.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	En el sistema deja de existir el idioma en cuestión.

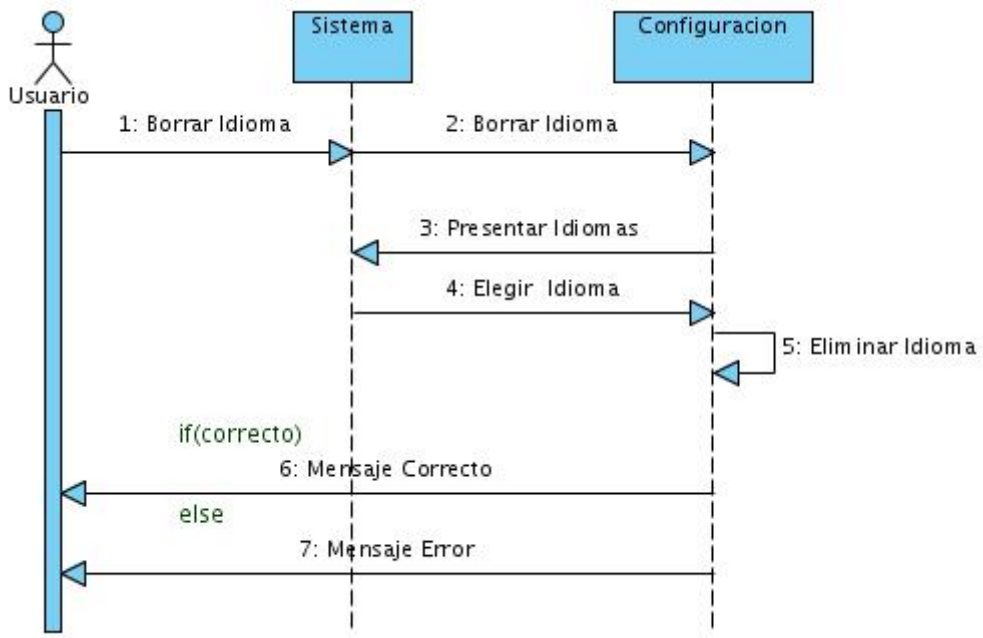


Figura 7 análisis diag. sec. borrar idioma

5.1.5.GESTIÓN EXTRACTORES (ESCENARIO 1): AÑADIR EXTRACTOR

Gestión Extractores (Escenario 2) : Añadir un extractor	
DESCRIPCIÓN	Se describe en este escenario cómo se añade un nuevo extractor al sistema
PRECONDICIONES	Que exista el fichero de configuración. Que exista el idioma al cual se quiere asociar el nuevo extractor.
FLUJO PRINCIPAL	El usuario accede a añadir un extractor. El usuario elige el idioma al que asocia el extractor. El usuario introduce el nombre del extractor [1] El usuario introduce el recurso asociado con el extractor, esto es, la clase que va a manejar el extractor.[2] El fichero de la clase se ubica en el directorio adecuado. El sistema procede a efectuar los cambios en el fichero de configuración, añadiendo el nuevo extractor con su idioma asociado a través de la abreviatura del mismo.
FLUJO EXCEPCIONAL	[1] Si el nombre ya existe, el sistema informa de ello y el usuario debe introducir otro nombre. [2] Si los datos de la clase no son correctos el sistema informa de ello.
POSTCONDICIONES	En el sistema existe un nuevo extractor al cual los usuarios pueden acceder para la lanzar la adquisición de terminología.

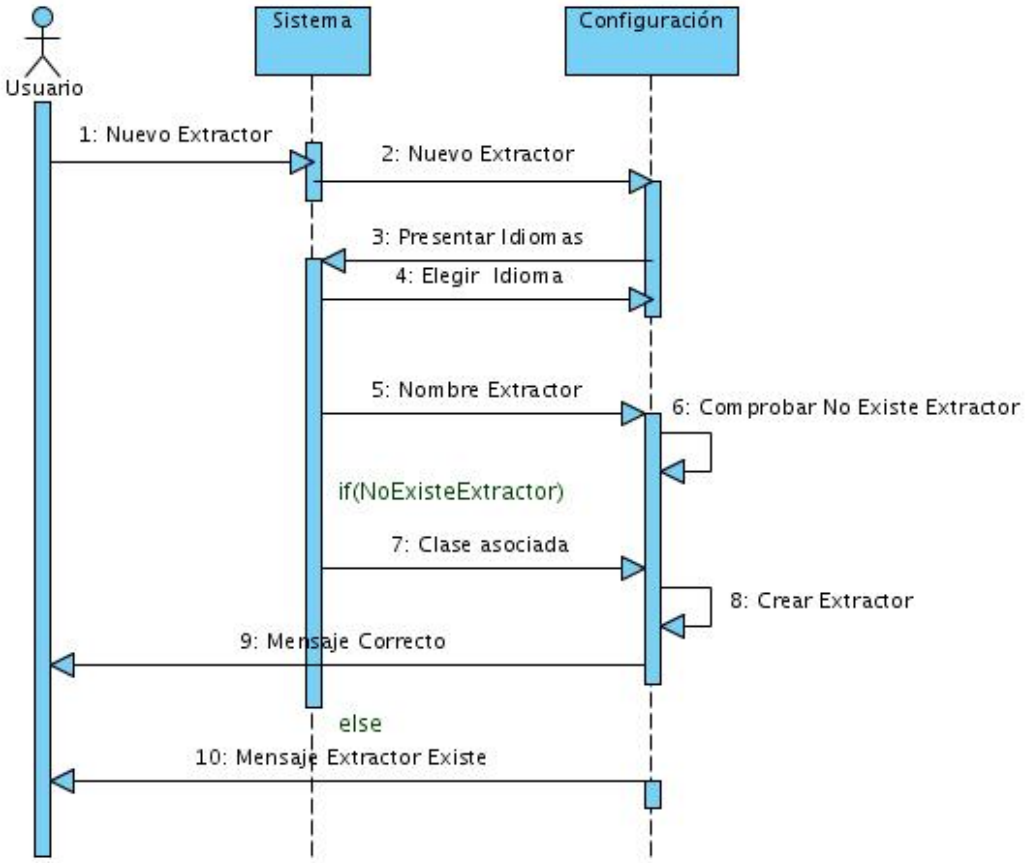


Figura 8 análisis diag.sec. añadir extractor

5.1.6.GESTIÓN EXTRACTORES (ESCENARIO 2): BORRAR EXTRACTOR

Gestión Extractores (Escenario 1) : Borrar un extractor	
DESCRIPCIÓN	En este escenario se describe cómo un extractor puede ser eliminado del sistema.
PRECONDICIONES	Que exista el fichero de configuración. Que exista el extractor que el usuario desea eliminar.
FLUJO PRINCIPAL	El usuario elige a través de la interfaz el extractor que desea eliminar. El usuario elige eliminar el extractor.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	En el sistema deja de existir el extractor en cuestión.

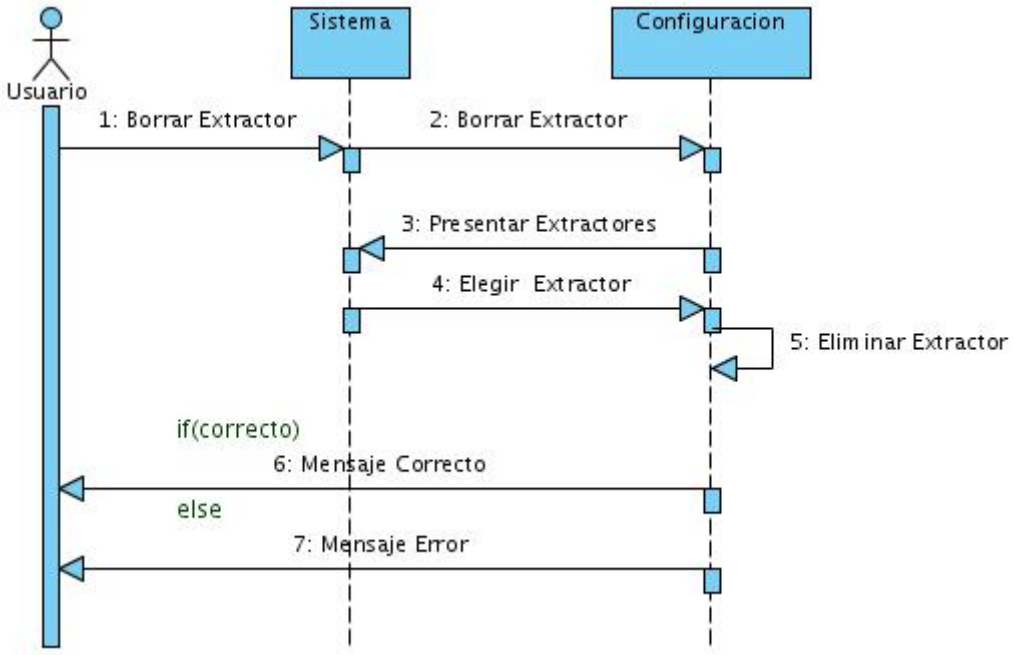


Figura 9 análisis diag.sec. borrar extractor

5.1.7.GESTIÓN ANALIZADOR (ESCENARIO 1): AÑADIR ANALIZADOR

Gestión Analizadores Sintácticos (Escenario 1) : Añadir un analizador sintáctico	
DESCRIPCIÓN	Se describe en este escenario cómo se añade un nuevo analizador sintáctico al sistema.
PRECONDICIONES	Que exista el fichero de configuración. Que exista el idioma al cual se quiere asociar el nuevo analizador sintáctico.
FLUJO PRINCIPAL	El usuario accede a añadir un analizador sintáctico. El usuario elige el idioma al que asocia el analizador sintáctico. El usuario introduce el nombre del analizador sintáctico [1] El usuario introduce el recurso asociado con el analizador sintáctico, esto es, la clase que va a manejar el analizador sintáctico.[2] El fichero de la clase se ubica en el directorio adecuado. El sistema procede a efectuar los cambios en el fichero de configuración, añadiendo el nuevo analizador sintáctico con su idioma asociado a través de la abreviatura del mismo.
FLUJO EXCEPCIONAL	[1] Si el nombre ya existe, el sistema informa de ello y el usuario debe introducir otro nombre. [2] Si los datos de la clase no son correctos el sistema informa de ello.
POSTCONDICIONES	En el sistema existe un nuevo analizador sintáctico al cual los usuarios pueden acceder para la lanzar el análisis sintáctico y generar el grafo de dependencias.

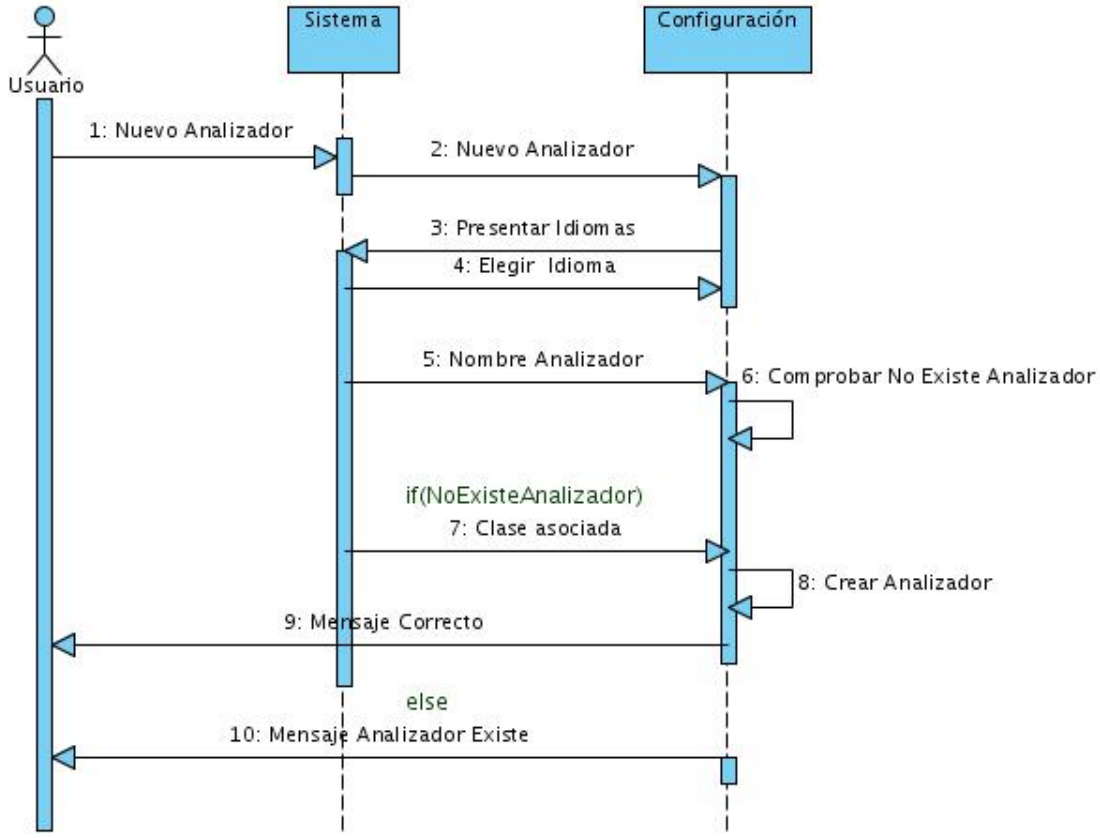


Figura 10 análisis diag.sec.añadir analizador

5.1.8.GESTIÓN ANALIZADOR (ESCENARIO 2): BORRAR ANALIZADOR

Gestión Analizadores Sintácticos (Escenario 2) : Borrar un analizador sintáctico	
DESCRIPCIÓN	En este escenario se describe cómo un analizador sintáctico puede ser eliminado del sistema.
PRECONDICIONES	Que exista el fichero de configuración. Que exista el analizador sintáctico que el usuario desea eliminar.
FLUJO PRINCIPAL	El usuario elige a través de la interfaz el analizador sintáctico que desea eliminar. El usuario elige eliminar el analizador sintáctico.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	En el sistema deja de existir el analizador sintáctico en cuestión

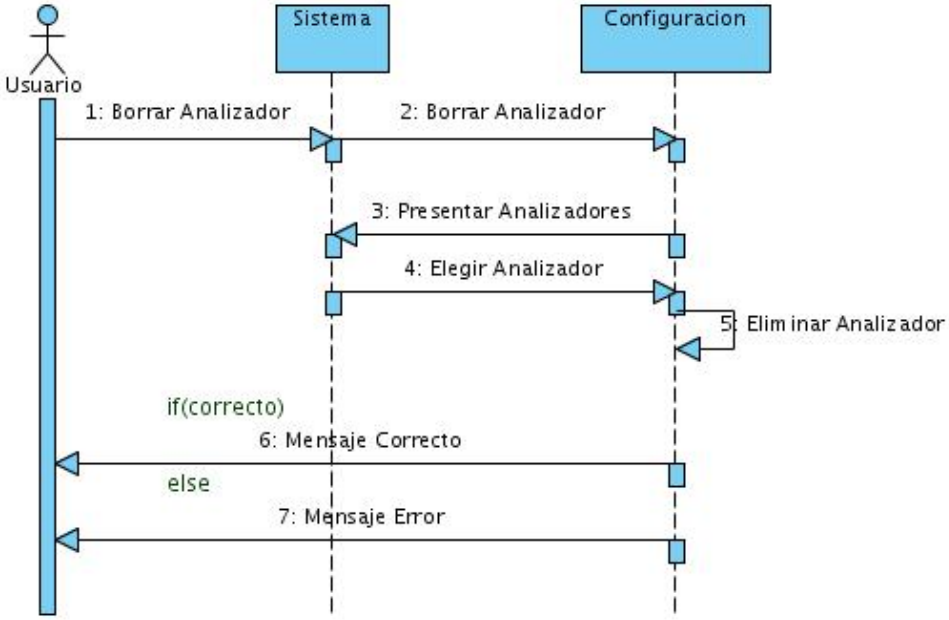


Figura 11análisis diag.sec.borrar analizador

5.2.CASO DE USO MANIPULACIÓN CORPUS

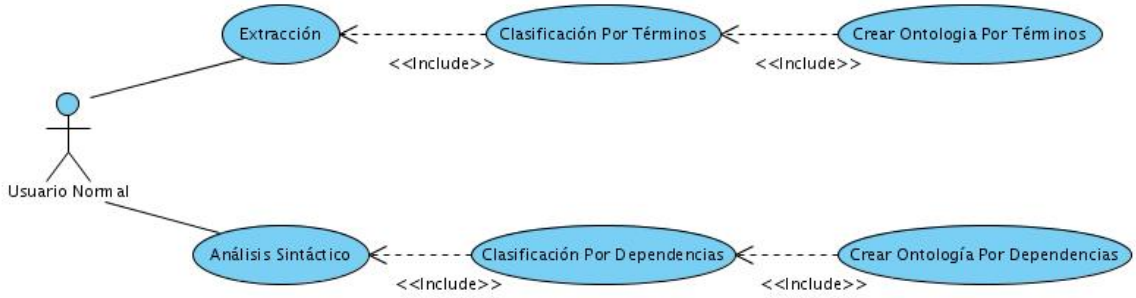


Figura 12 caso de uso manipulación corpus

5.2.1.CREAR ONTOLOGÍA POR TÉRMINOS

Crear ontología por términos	
DESCRIPCIÓN	El proceso consiste en crear un modelo de ontología adaptado a un estándar para su uso en diversos entornos. Dentro del sistema para llevar a cabo esta tarea; otras dos deben haber sido realizadas previamente, la extracción de términos y su clasificación. El caso de uso que aquí se describe consiste en recorrer la estructura creada por la clasificación y crear la ontología en base a un lenguaje estándar, definiendo clases e individuos.
PRECONDICIONES	Debe haberse lanzado la extracción y la clasificación de los términos. Debe por lo tanto conocerse el tema del corpus, así como el idioma tratado.
FLUJO PRINCIPAL	Se crea la cabecera de la ontología. Se recorre la estructura creada por la clasificación localizando los distintos elementos y se crean las distintas clases e individuos que conforman la ontología.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	Se tiene un fichero con la ontología creada en base a un modelo estándar y a partir de los términos extraídos en el dominio actual.

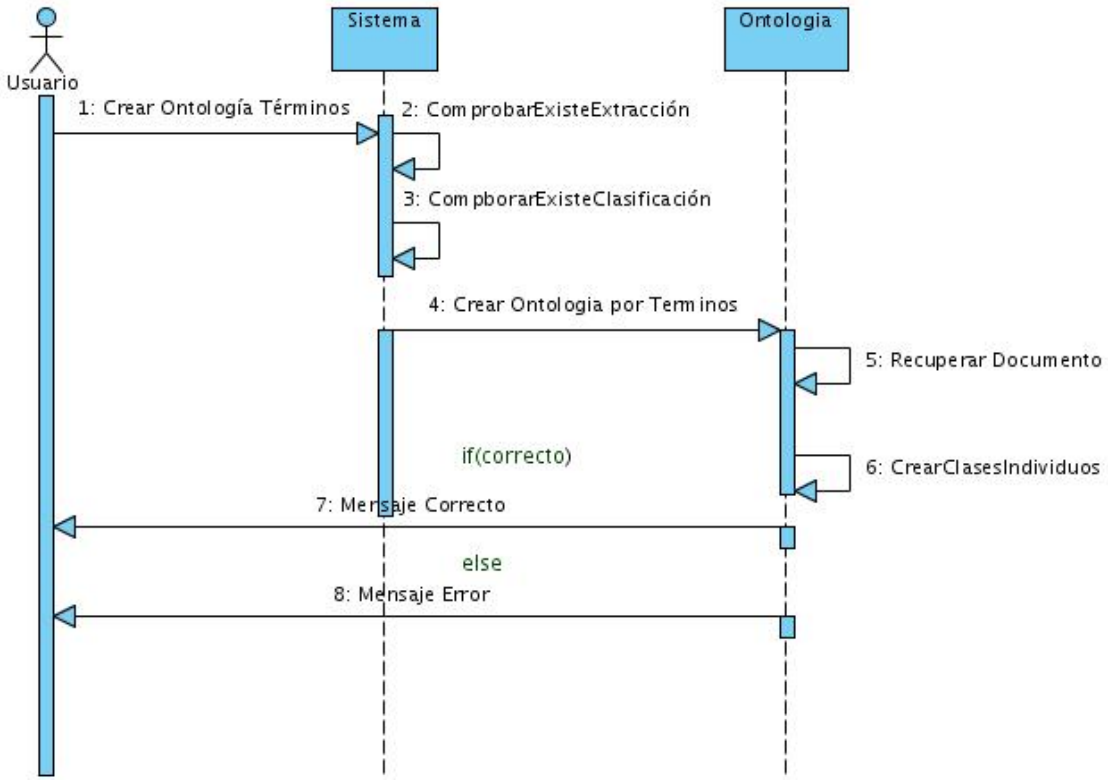


Figura 13 análisis diag.sec.crear ontología términos

5.2.2.CREAR ONTOLOGÍA POR DEPENDENCIAS

En este apartado se describe cómo se genera la ontología por dependencias. Es importante señalar que éste es un proceso iterativo, en el sentido que podrá lanzarse tantas veces como el usuario lo desee, efectuando en cada paso los cambios en los ficheros oportunos (detallados en el apartado 13.2 del presente documento). De este modo el usuario puede en cada iteración obtener unos resultados cada vez más acordes a sus necesidades.

Crear ontología por dependencias	
DESCRIPCIÓN	El proceso consiste en crear un modelo de ontología adaptado a un estándar para su uso en diversos entornos. Dentro del sistema para llevar a cabo esta tarea; otras dos deben haber sido realizadas previamente, el análisis sintáctico y la clasificación por dependencias. El caso de uso que aquí se describe consiste en recuperar de la base de datos el tipo de relación establecido entre las dependencias y, a partir de esa información, crear la ontología en base a un lenguaje estándar, definiendo clases e individuos.
PRECONDICIONES	Debe haberse lanzado el análisis sintáctico y la clasificación por dependencias. El usuario ya habrá introducido en un paso previo el idioma y el tema actual.
FLUJO PRINCIPAL	Se crea la cabecera de la ontología. Se recorre la base de datos y en base a las relaciones almacenadas en las mismas se construyen clases e individuos.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	Se tiene un fichero con la ontología creada en base a un modelo estándar y a partir de las dependencias relevantes deducidas con la clasificación de dependencias.

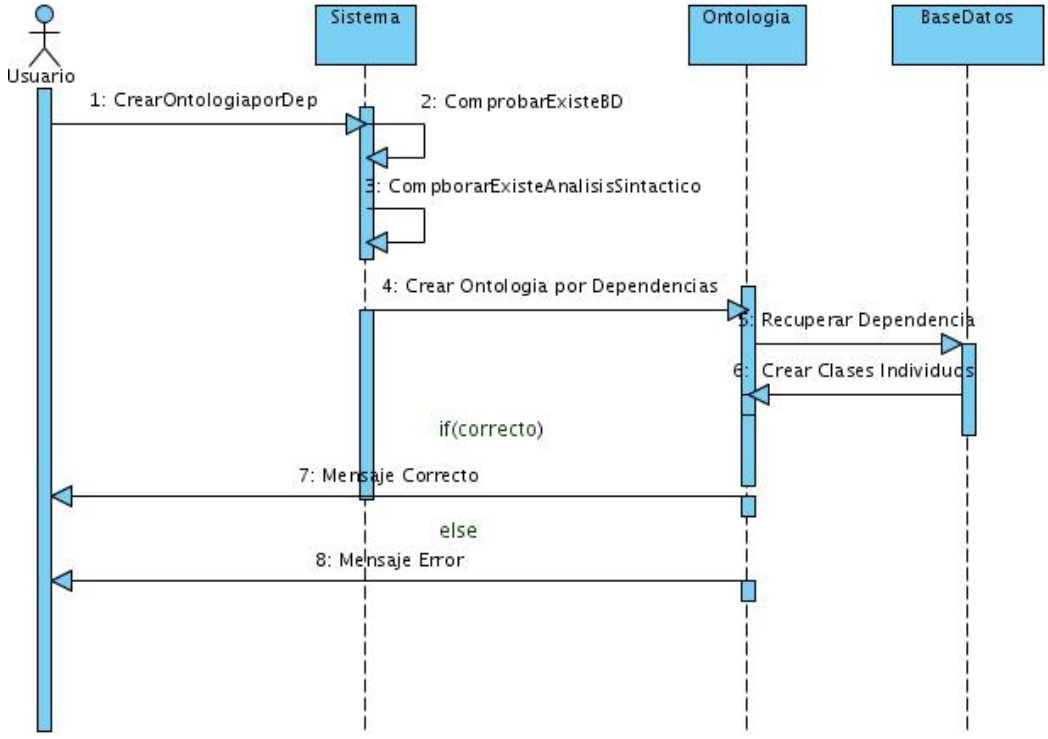


Figura 14 análisis diag.sec. crear ontología dependencias

5.2.3.CLASIFICACIÓN POR DEPENDENCIAS

Crear ontología por dependencias	
DESCRIPCIÓN	El caso de uso descrito, ante la petición del usuario, lanza una clasificación de las dependencias almacenadas en la base de datos. Este proceso implica una llamada a un recurso externo a la herramienta desarrollada y por lo tanto aquí sólo se indican las llamadas realizadas.
PRECONDICIONES	Debe haberse lanzado el análisis sintáctico y la clasificación por dependencias.
FLUJO PRINCIPAL	El usuario introduce el tema y el idioma El sistema hace una llamada externa solicitando la ejecución del algoritmo que clasifica las dependencias. [1]
FLUJO EXCEPCIONAL	[1] La llamada falla, el sistema informa de ello.
POSTCONDICIONES	La base de datos se actualiza con nuevos valores que indican el tipo de relación existente entre las dependencias consideradas relevantes..

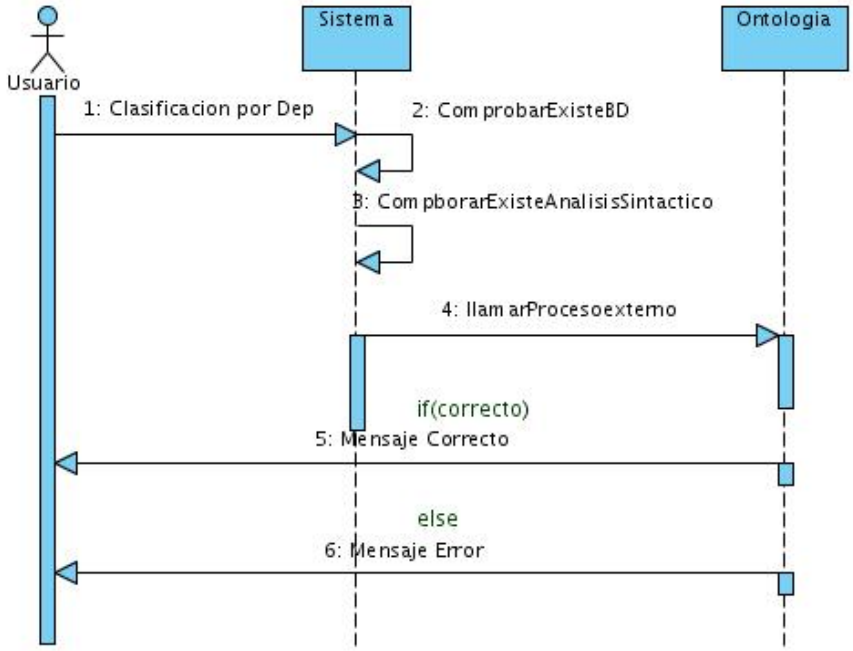


Figura 15 análisis diag.sec.clasificacion dependencias

5.2.4.CASO DE USO: EXTRACCIÓN

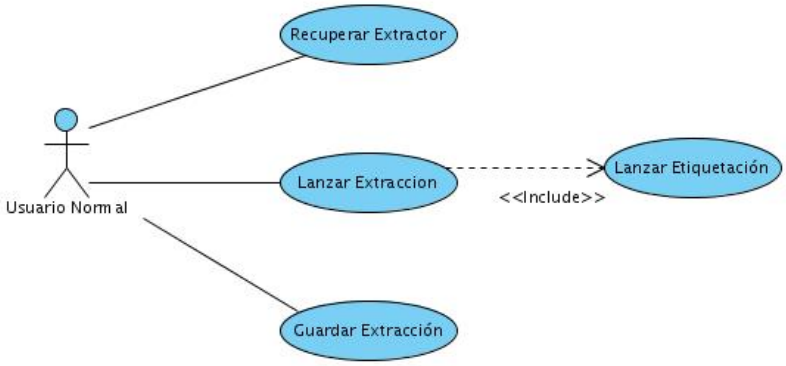


Figura 16 caso de uso extracción

(A) RECUPERAR EXTRACTOR

Recuperar Extractor	
DESCRIPCIÓN	Este caso de uso ocurre en el momento en él que el usuario solicita la extracción de términos. El sistema, a través del fichero de configuración, propone el corpus sobre él que realizar la extracción y los idiomas asociados a ese corpus. Una vez seleccionados esos elementos se le presentan los extractores a partir de los cuales es posible realizar la extracción[1]. Una vez elegidos todos los elementos se procede a recuperar los datos relativos al (o los) extractor(es) elegidos, esto es, las clases que los lanzan y si es el caso[Esto es, porque los extractores no tienen porque ser un recurso externo, como es el caso de los que se tienen a disposición en este momento, sino que podría ser una clase que implementa la extracción directamente], las rutas donde se ubican estos recursos. Así mismo, se recupera el directorio donde se ubica el corpus correspondiente
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma y entre el idioma y los extractores. Que exista el corpus conformado por los documentos.
FLUJO PRINCIPAL	El usuario elige un corpus de entre los existentes. Dado un corpus elige un idioma asociado al mismo.[1] Una vez elegido el idioma se presentan los extractores disponibles para ese idioma. El usuario podrá elegir realizar la extracción entre uno o varios extractores simultáneamente. [2] Se procede a recuperar la información pertinente: clase que ejecuta la extracción, rutas relativas a los directorios donde se ubican los extractores si es el caso y fichero de corpus.
FLUJO EXCEPCIONAL	[1] No existe el corpus, el sistema informa de ello. Este error se debe a que no existen documentos de asociados al tema e idioma elegidos. [2] No existe ningún extractor asociado al idioma el sistema informa de ello.
POSTCONDICIONES	Se recuperan los datos necesarios para lanzar la extracción, es decir, la clase pertinente, la ubicación del corpus y si es el caso la ruta donde se encuentra el extractor.

[1] Esto es porque los extractores no tienen porque ser un recurso externo, como es el caso de los que tienen a disposición en este momento sino que podría ser una clase que implementan la extracción directamente.

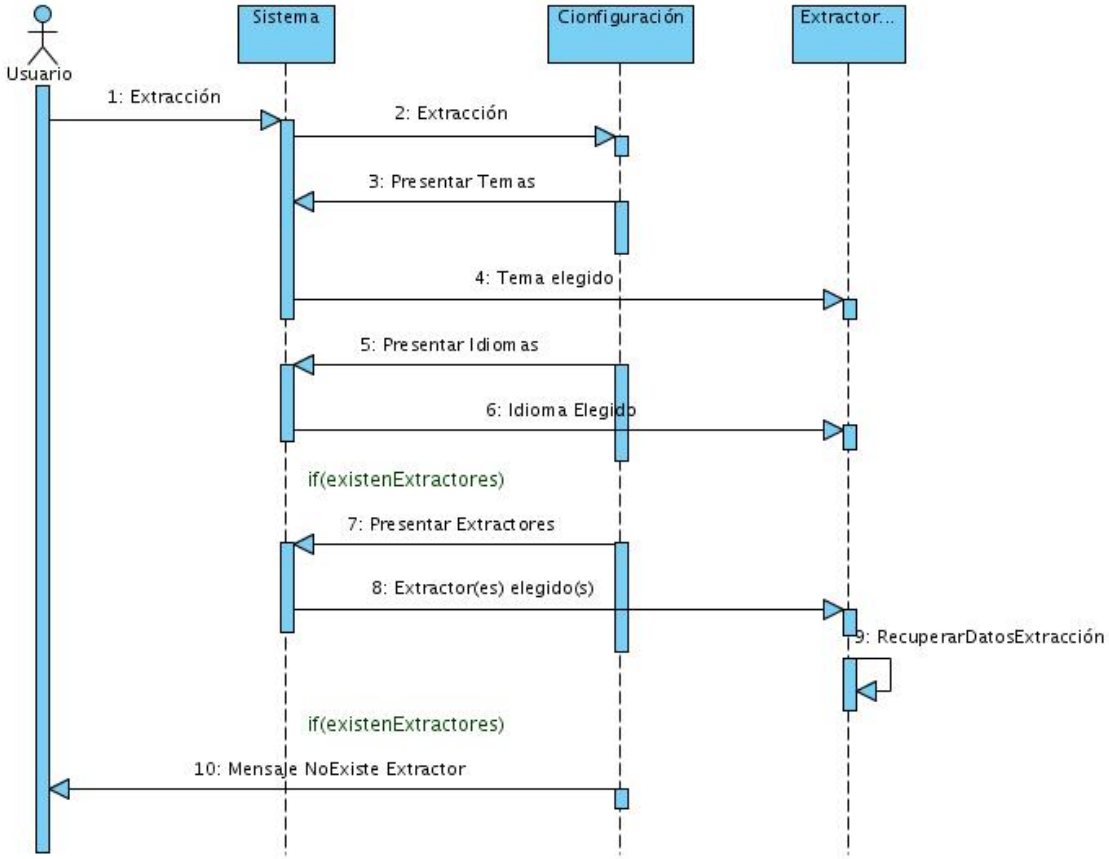


Figura 17 análisis diag.sec.recuperar extractor

(B) LANZAR EXTRACCIÓN

Lanzar Extracción	
DESCRIPCIÓN	En este caso de uso se refleja la llamada al extractor. Una vez recuperados los datos correspondientes a la extracción, comentados en el caso de uso anterior. Se procede a lanzar la extracción, en este proceso será necesario preprocesar el texto así como etiquetar el mismo.
PRECONDICIONES	El usuario previamente ha elegido el corpus y el idioma sobre los que realizar la operación, así como, el/los extractor/es implicados.
FLUJO PRINCIPAL	El sistema recupera el corpus asociado a la elección del usuario.[1] El texto es etiquetado, (ver el siguiente caso de uso). [2] El corpus etiquetado se introduce en el sistema de extracción que procederá a su manipulación para recuperar aquellos términos relevantes en el dominio. Los datos devueltos por la extracción de terminología deberán proporcionarse en un formato predefinido para poder ser posteriormente almacenados cumpliendo una estructura definida para ello.
FLUJO EXCEPCIONAL	[1] Se produce un error a la hora de recuperar el corpus, el sistema informa de ello. [2] Dependiendo de la implementación que se haga del extractor, pues esta es una de las partes dinámicas del sistema, esta tarea podría no incluirse.
POSTCONDICIONES	Se obtienen los términos relevantes a un determinado corpus, extraídos en función del algoritmo asociado al extractor.

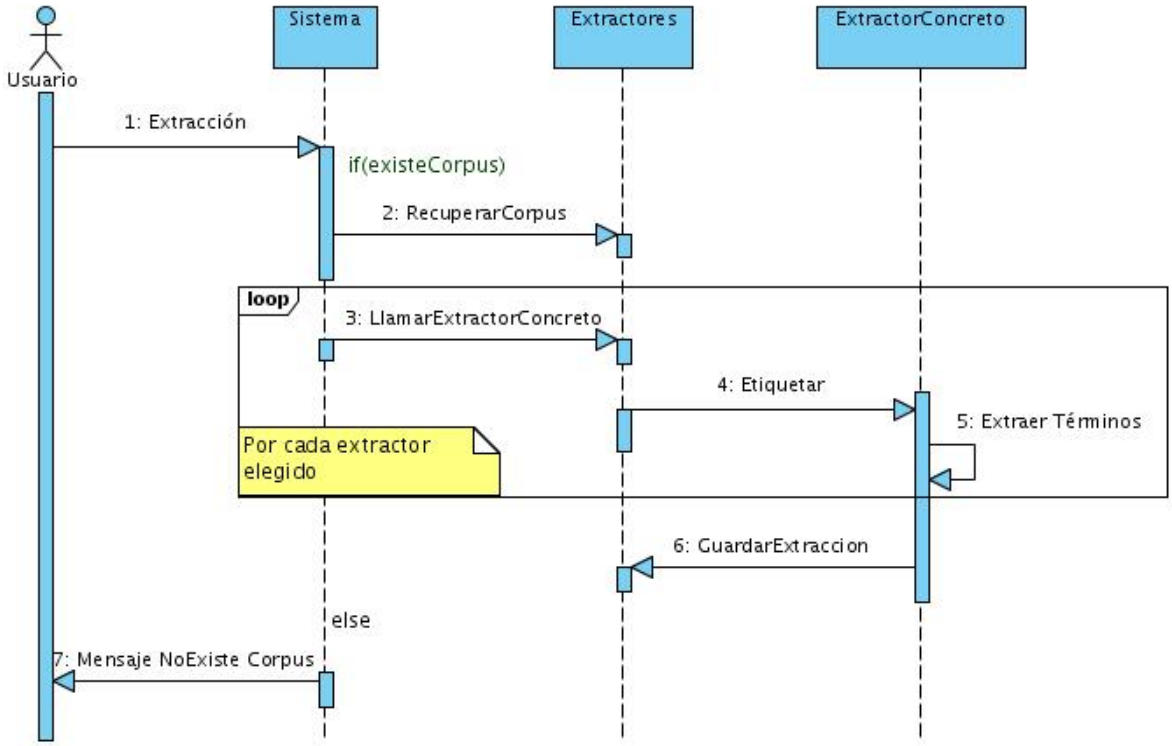


Figura 18 análisis diag.sec.lanzar extracción

(C) LANZAR ETIQUETACIÓN

Lanzar Etiquetación	
DESCRIPCIÓN	En este escenario se describe la llamada a un etiquetador, por norma general este tipo de llamadas supondrá una llamada externa al sistema. Además, como ya se ha indicado en el escenario anterior, este tipo de actividad no tiene porque ser requerida pues la implementación que haga cada usuario puede no implicar el uso de un etiquetador. Para proceder a su cometido, necesita conocer el corpus y el idioma.
PRECONDICIONES	Conocer el corpus y el idioma sobre él que se va a realizar la operación.
FLUJO PRINCIPAL	Localizar el fichero del corpus según el tema del corpus y el idioma elegido por el usuario. [1] Proceder al pre-procesamiento del corpus[2]. Realizar la etiquetación.
FLUJO EXCEPCIONAL	[1] Se produce un error a la hora de recuperar el corpus.
POSTCONDICIONES	El texto es almacenado en un fichero según el formato de salida asociado al etiquetador empleado. En ciertos casos dependiendo de la implementación concreta el formato puede necesitar un tratamiento para adecuarlo a la entrada del siguiente proceso. Es de notar en este punto, que el proceso de etiquetación, en el caso concreto implementado para este proyecto, es una actividad puente y necesaria para la realización de otras tareas.

[2] Como ocurre en el caso de la etiquetación, esta actividad depende del tipo de recurso proporcionado al sistema, siendo en algunos casos no necesaria esta tarea.

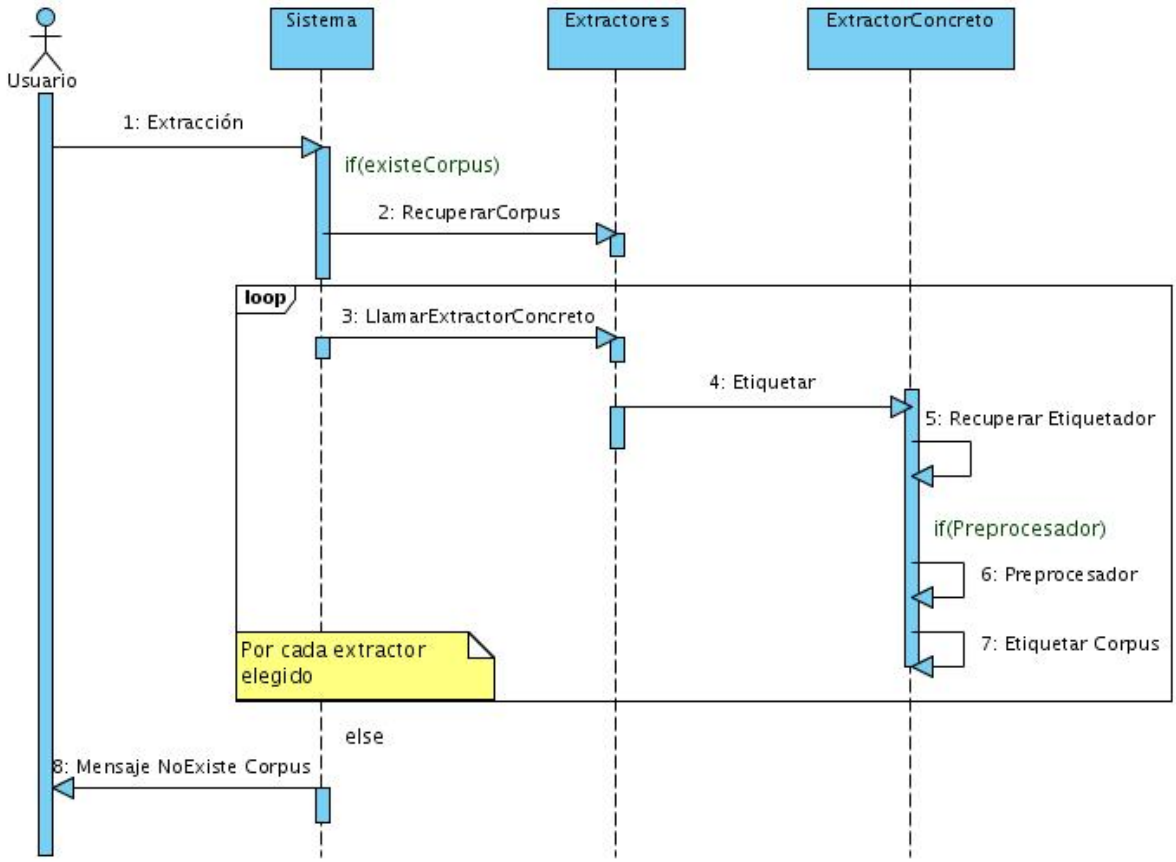


Figura 19 análisis diag.sec.lanzar etiquetación

(D) GUARDAR EXTRACCIÓN

Guardar Extracción	
DESCRIPCIÓN	Este escenario entra en juego una vez realizada la extracción. Cada uno de los extractores devolverá sus resultados en un formato distinto. Es preciso que dichos resultados se almacenen en el sistema de forma predefinida.
PRECONDICIONES	La extracción debe estar realizada. Conocer el tema del corpus, así como el idioma que ha originado la adquisición de terminología.
FLUJO PRINCIPAL	Se recogen los términos recuperados por el módulo de extracción. [1] Se procede a su almacenamiento en un documento estándar definido para la arquitectura. En caso de que el usuario elija más de un extractor los términos presentes en más de un extractor solamente se almacenan una vez. Se almacena así mismo el extractor que ha originado los términos. El documento se guarda en el sistema de archivos en función del tema del corpus y el idioma.
FLUJO EXCEPCIONAL	[1] La extracción no produce ningún término. El sistema informa de ello.
POSTCONDICIONES	En el directorio correspondiente se almacena el documento de formato estándar con el contenido de la adquisición de terminología.

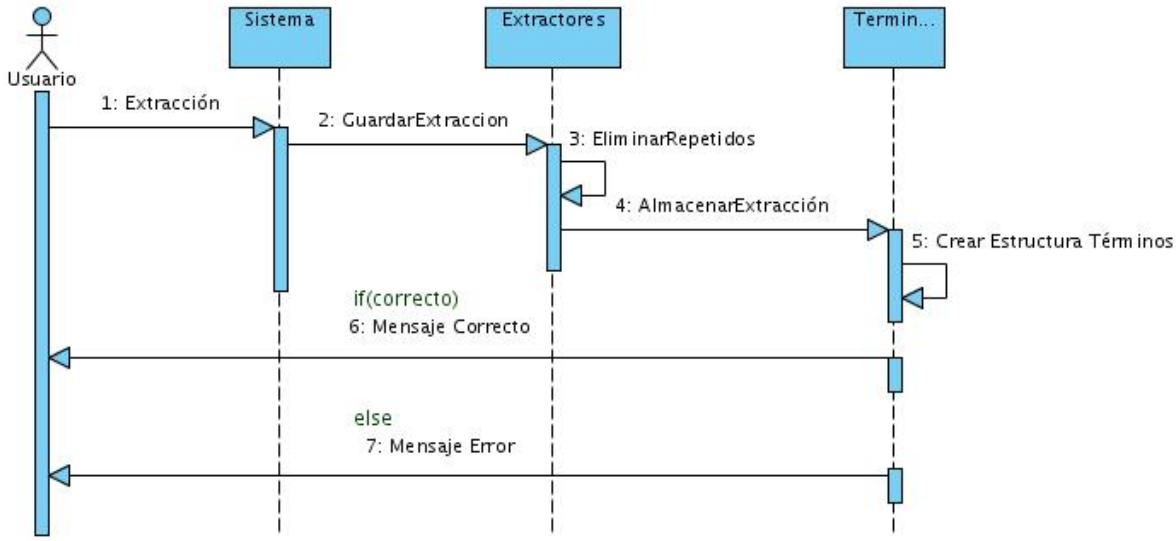


Figura 20 análisis diag.sec.guardar extracción

5.2.5.CASO DE USO: CLASIFICACIÓN POR TÉRMINOS

Este caso de uso depende de una extracción de términos previa. Es decir, incluye el caso de uso de extracción. Así, en función de los datos elegidos por el usuario para realizar la extracción se realiza la clasificación. Esto es así, porque la clasificación por términos sólo puede llevarse a cabo si previamente existe una extracción realizada.

Figura 21caso de uso clasificación términos

(A) RECUPERAR RESULTADOS EXTRACCIÓN

Recuperar resultados extracción	
DESCRIPCIÓN	Este caso de uso ocurre en el momento en él que el usuario solicita realizar la clasificación a partir de la extracción realizada. El sistema, a través del fichero de configuración, en un paso previo, habrá solicitado el corpus y el idioma sobre los que se efectuará la clasificación. Con los resultados de la extracción en mano, el sistema podrá recuperar el fichero que almacena los mismos.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma y entre el idioma y el extractor. Dado el tema, el idioma y el extractor exista una extracción realizada.

FLUJO PRINCIPAL	El sistema guarda información sobre el tema y el idioma implicados en la extracción. El sistema recupera el fichero de extracción para su procesamiento en la tarea de clasificación. [1]
FLUJO EXCEPCIONAL	[1] No existe el fichero asociado a la extracción, el sistema informa de ello.
POSTCONDICIONES	Se recuperan los datos necesarios para realizar la clasificación, éstos son, el tema del corpus, el idioma y el fichero con el contenido de la extracción.

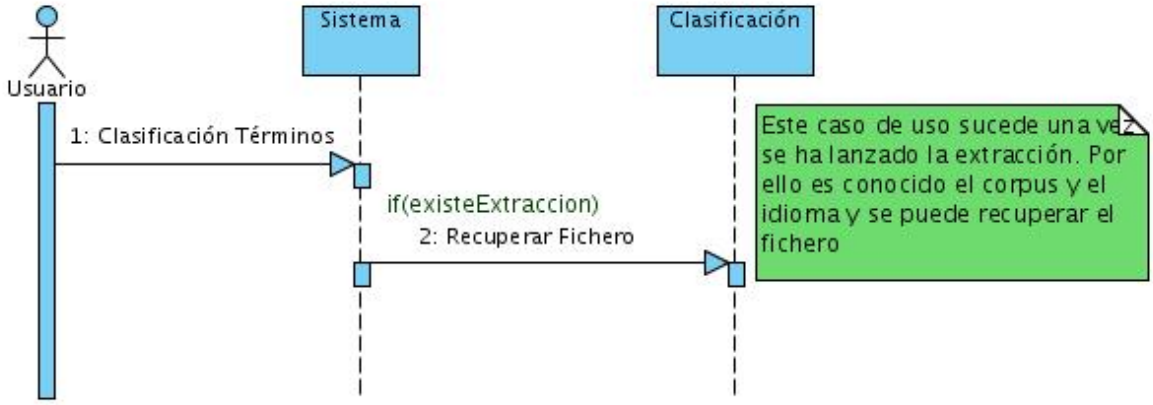


Figura 22 análisis diag.sec.recuperar resultados extracción

(B) LANZAR CLASIFICACIÓN

Lanzar clasificación	
DESCRIPCIÓN	Una vez localizado el fichero de extracción correspondiente. Se puede proceder a lanzar la clasificación de los términos en base a la similaridad de sus contextos y mediante la utilización de medidas estadísticas.
PRECONDICIONES	Que exista el archivo con el resultado de la extracción.
FLUJO PRINCIPAL	Se recupera para cada palabra clave de un término sus contextos (Este proceso se detalla en el caso de uso siguiente.) Creada la matriz de asociación entre las palabras claves y sus contextos, se aplican medidas para medir la similaridad entre esos contextos. La agrupación obtenida permite establecer una clasificación que se almacena en un fichero estándar definido para este cometido.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	Se obtiene una clasificación de los términos extraídos almacenados en un fichero definido para esta operación.

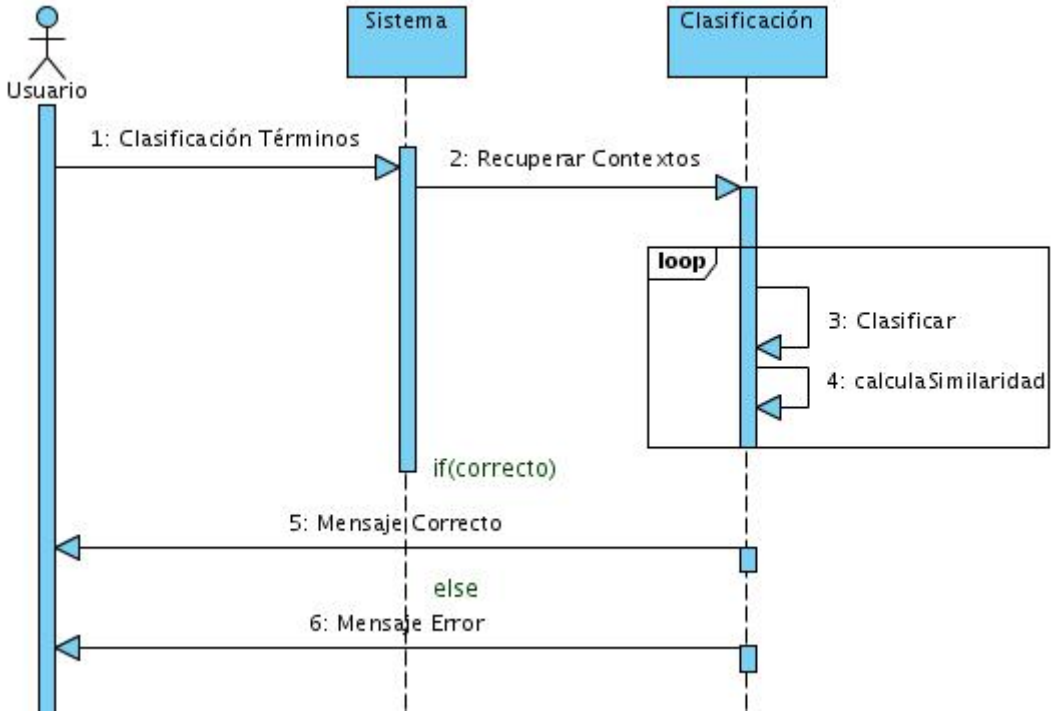


Figura 23 análisis diag.sec.lanzar clasificación

(C) RECUPERAR CONTEXTOS

Recuperar contextos	
DESCRIPCIÓN	En este caso de uso se procede a crear los contextos asociados a la palabra principal de los términos extraídos en el proceso de adquisición de terminología.
PRECONDICIONES	Que exista el archivo con el resultado de la extracción.
FLUJO PRINCIPAL	Para cada palabra que forma el término se recupera su categoría léxica, esto es posible gracias al análisis sintáctico. Para cada término se localiza su palabra principal, esto es, la primera palabra (o palabra clave) que lo forma, y se almacena junto con sus contextos. [1]
FLUJO EXCEPCIONAL	[1] Si la palabra clave no es un sustantivo el sistema no la almacena.
POSTCONDICIONES	Se obtiene un conjunto de palabras clave, asociadas con sus respectivos contextos.

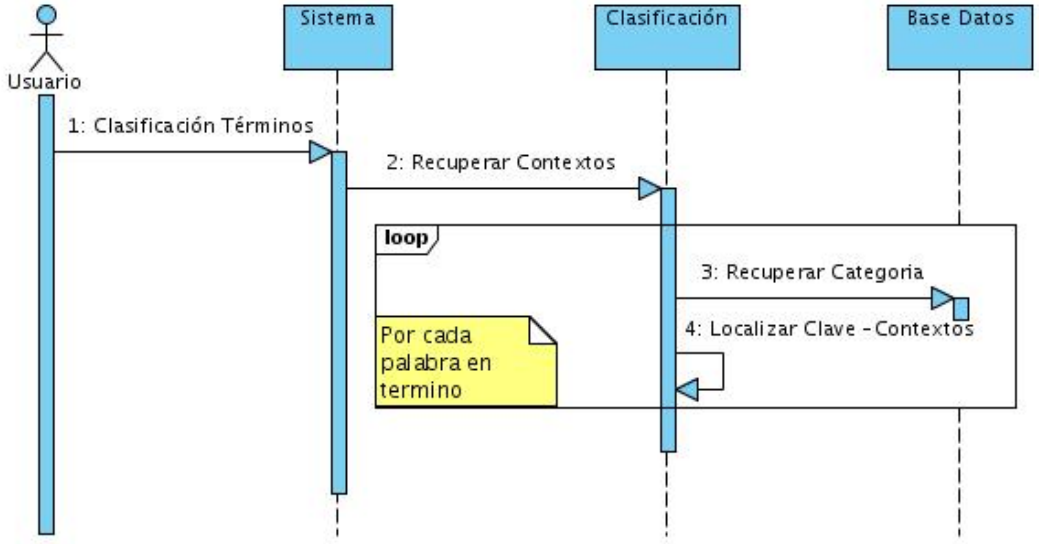


Figura 24 análisis diag.sec.recuperar contextos

5.2.6.CASO DE USO: ANÁLISIS SINTÁCTICO

El análisis sintáctico es una tarea que se lleva a cabo sobre los documentos que conforman el corpus, es decir no se trata el corpus de manera conjunta. Se analiza cada uno de los documentos de manera individual y para cada documento se analiza cada una de las frases que lo componen. Se tiene así, perfectamente organizado el análisis de forma a poder recuperar fácilmente en cada instante una determinada frase asociada a un documento.

Es destacable además que el análisis sintáctico es una tarea que se ejecuta cada vez que el usuario aporta nuevos documentos al sistema.

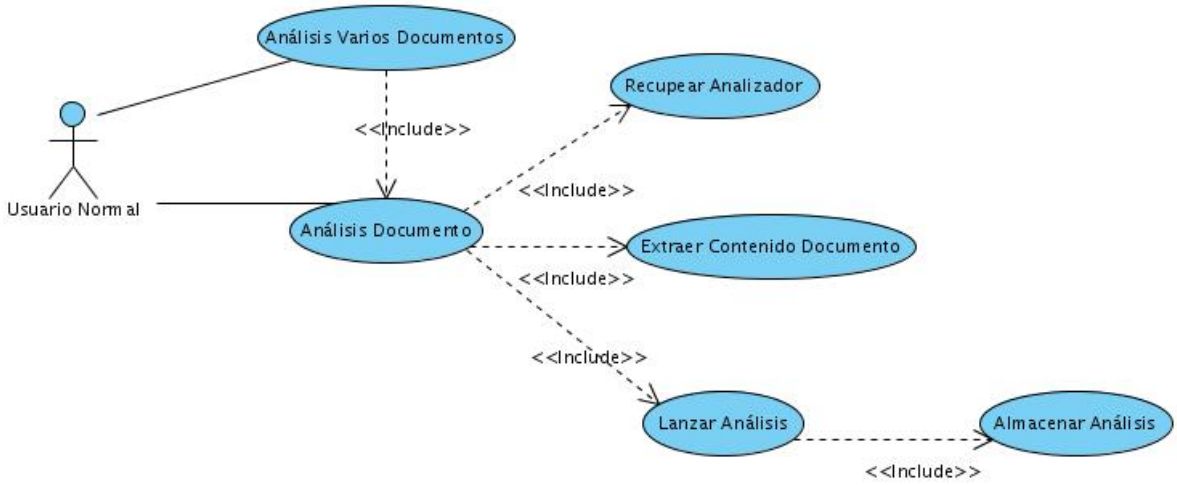


Figura 25 caso de uso análisis sintáctico

(A) ANÁLISIS VARIOS DOCUMENTOS

Este caso de uso no se describe en detalle, pues simplemente se señala que cabe la posibilidad de que el usuario añada más de un documento al sistema. Para cada uno de estos documentos se realizan las actividades descritas a continuación.

(B) ANÁLISIS UN DOCUMENTO

Análisis Documento	
DESCRIPCIÓN	En este caso de uso se describe como es realizado el análisis sintáctico de un documento. Al ser una tarea dependiente de la introducción de nuevos documentos en el sistema, el usuario previamente habrá indicado el corpus y el idioma correspondiente.
PRECONDICIONES	Debe existir el fichero de configuración Debe ser conocido el tema y el idioma asociados al documento y por lo tanto al análisis.
FLUJO PRINCIPAL	Se recupera el fichero sobre él que realizar el análisis sintáctico. [1] Se recuperan los datos necesarios para la realización de la tarea (caso de uso detallado a continuación). Se extrae del documento el contenido pertinente (caso de uso detallado a continuación). El sistema procede a ejecutar el analizador elegido (caso de uso detallado a continuación).
FLUJO EXCEPCIONAL	[1] El fichero no cumple con el formato esperado, el sistema informa de ello.
POSTCONDICIONES	En el grafo de dependencias asociado al corpus e idioma se han añadido nuevas dependencias resultado del análisis sintáctico de un documento.

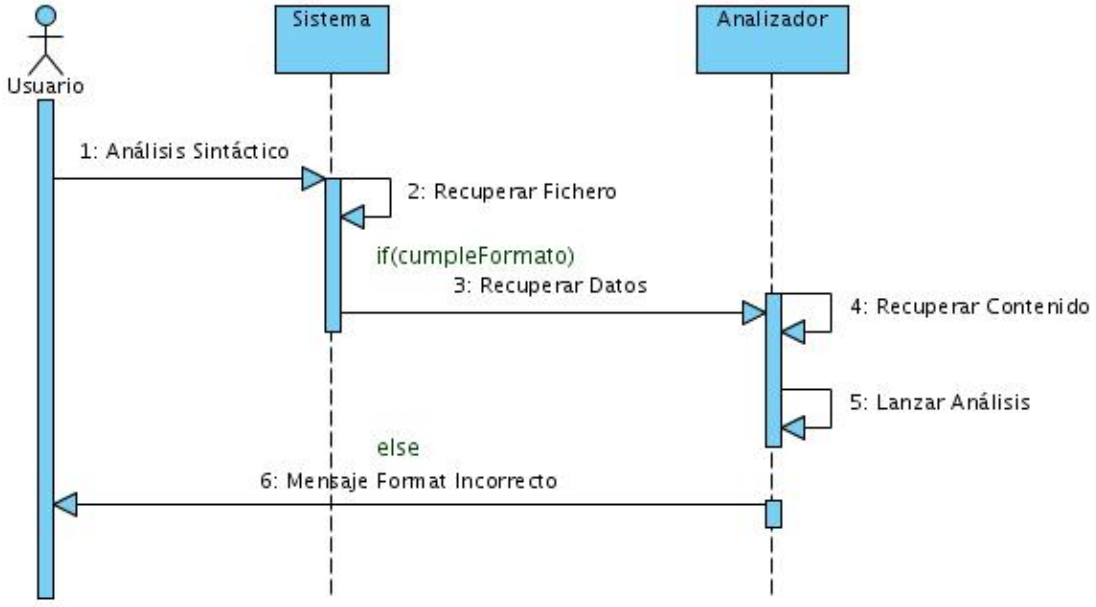


Figura 26 análisis diag.sec.análisis un documento

(C) RECUPERAR ANALIZADOR

Recuperar analizador	
DESCRIPCIÓN	Este caso de uso ocurre en el momento en él que el usuario solicita realizar el análisis sintáctico. El usuario previamente ha introducido la información requerida para añadir nuevos documentos al sistema. Por lo tanto es conocido el corpus y el idioma asociado. Con todos los elementos se procede a recuperar los datos relativos al extractor elegido, esto es, la clase que lo lanza y si es el caso las rutas[3] donde se ubican estos recursos.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma y entre el idioma y el analizador sintáctico. Que exista el corpus conformado por los documentos.
FLUJO PRINCIPAL	En un paso previo, el usuario habrá elegido el corpus y el idioma. Además, habrá indicado el fichero que desea analizar. Se presentan los analizadores sintácticos disponibles para ese idioma.[1] Se procede a recuperar la información pertinente: clase que ejecuta el análisis sintáctico y las rutas relativas a los directorios donde se ubican el analizador sintáctico.
FLUJO EXCEPCIONAL	[1] No existe ningún analizador sintáctico asociado al idioma. El sistema informa de ello.
POSTCONDICIONES	Se recuperan los datos necesarios para lanzar el análisis sintáctico, es decir, la clase pertinentes y si es el caso la ruta donde se encuentra el analizador sintáctico.

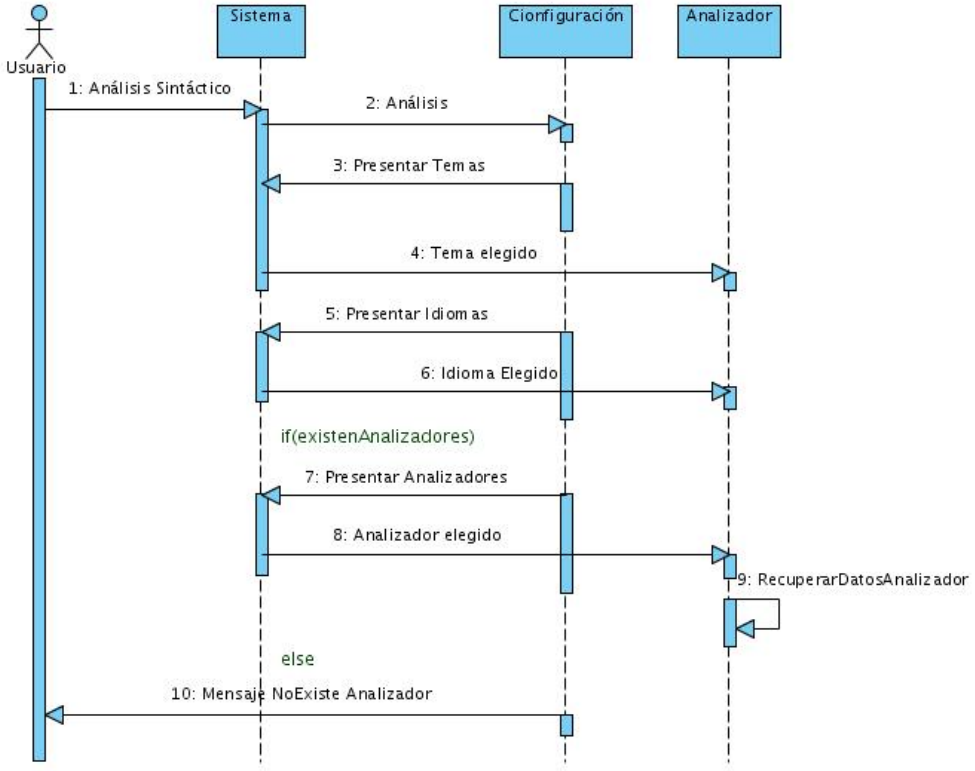


Figura 27 análisis diag.sec.recuperar analizador

[3] Los analizadores sintácticos no tienen porque ser un recurso externo, como es el caso de los que se tienen a disposición en este momento, sino que podrían ser una clase que implementa la extracción directamente.

(D) EXTRAER CONTENIDO DOCUMENTO

Extraer Contenido Documento	
DESCRIPCIÓN	El usuario puede introducir, bien un conjunto de documentos bien un único documento. En cualquier caso debe ser un documento estándar definido para el sistema. En este escenario se recuperan el contenido relevante del documento, esto es, el título y el texto.
PRECONDICIONES	Se debe conocer el corpus y el idioma asociado al análisis a realizar. El documento tiene que estar en el formato estándar definido para la arquitectura
FLUJO PRINCIPAL	Recuperar el documento. [1] Leer del documento extrayendo título y texto. [2] Almacenar el resultado en un fichero temporal para su posterior tratamiento.
FLUJO EXCEPCIONAL	[1] El documento no se ha podido recuperar o leer el sistema informa de ello. [2] El documento no cumple con el estándar establecido, el sistema informa de ello.
POSTCONDICIONES	El título y el texto de los documentos se encuentran en un fichero temporal para su inmediato tratamiento dentro del proceso de análisis sintáctico.

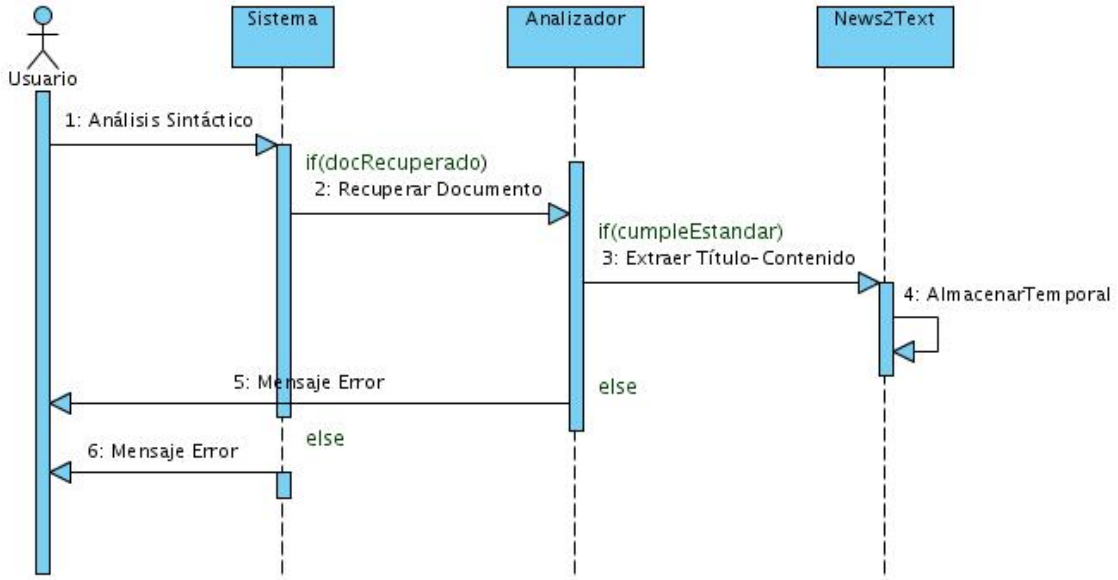


Figura 28 análisis diag.sec.extraer contenido documento

(E) LANZAR ANÁLISIS

Lanzar análisis	
DESCRIPCIÓN	Una vez obtenidos el tema, el idioma y el contenido a analizar se puede proceder a realizar el análisis sintáctico. Para ello, como se ha indicado previamente, el contenido debe procesarse frase a frase. El resultado del mismo será almacenado en un grafo de dependencias (en forma de base de datos) para su tratamiento en actividades posteriores
PRECONDICIONES	El usuario debe haber elegido un tema de corpus y un idioma. Se debe haber construido el contenido del documento para lanzar así el análisis sintáctico
FLUJO PRINCIPAL	Leer del fichero temporal de contenido y preprocesarlo. De esta manera se prepara el texto para la actividad del análisis. [1] Recoger las frases de una en una:[2] Realizar la llamada a la clase asociada al analizador sintáctico elegido, con la frase actual a procesar. Se etiqueta la frase. Este paso es necesario para tener información almacenada de todos las palabras que conforman el corpus, pues la herramienta o recursos elegido para etiquetar la frase deberá proporcionar para cada palabra su forma, su categoría y su lema.[4] Recuperar los resultados obtenidos para su almacenamiento en la base de datos. Proceso que se detalla en el siguiente escenario.
FLUJO EXCEPCIONAL	[1] No se puede leer del fichero temporal, el sistema informa de ello. [2] El documento no contiene frases, el sistema puede seguir su funcionamiento normal.
POSTCONDICIONES	Se obtiene el análisis de cada una de las frases del documento junto con la forma, categoría y lema de cada una de las palabras que conforman la frase.

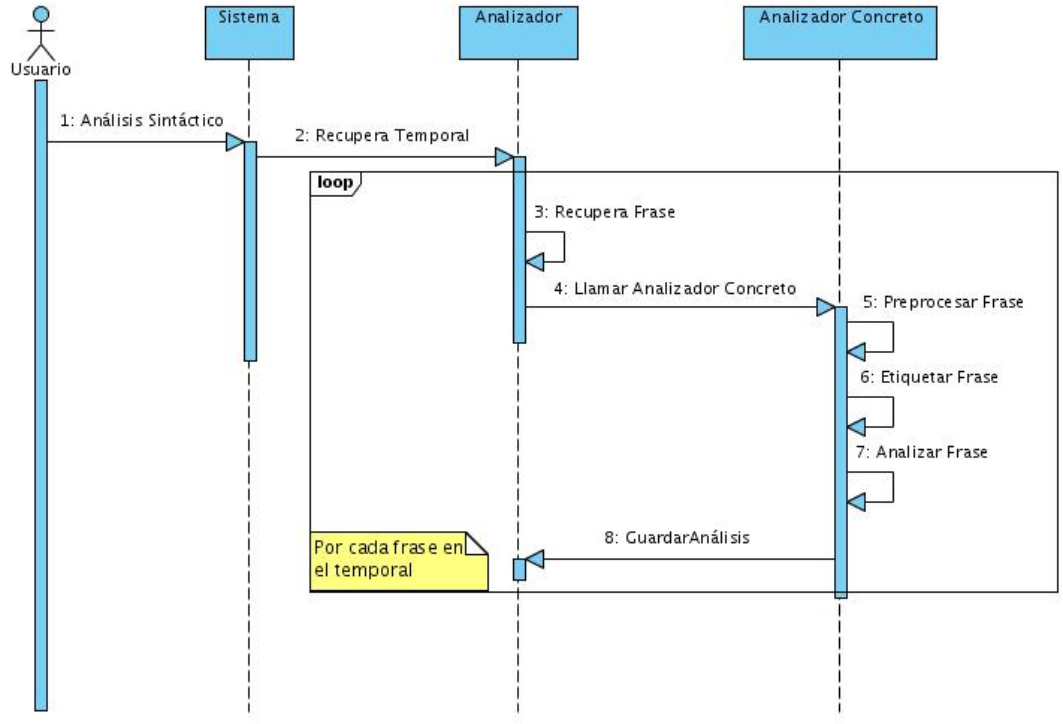


Figura 29 análisis diag.sec.lanzar análisis

[4] Al igual que sucede en la extracción, esta actividad no es obligatoria si la propia clase asociada al análisis ya efectúa esta tarea.

(F) ALMACENAR ANÁLISIS

Almacenar análisis	
DESCRIPCIÓN	Una vez que se ha lanzado el análisis sintáctico, para cada una de las frases, este proceso devuelve unos valores, indicados en el escenario anterior que necesitan ser almacenados. El uso que se hace posteriormente de los mismos implica que el almacenamiento se haga en una base de datos que se recuperará en función del corpus e idioma indicados por el usuario.
PRECONDICIONES	El sistema debe estar conectado a la base de datos. El análisis sintáctico de la frase a procesar debe estar hecho. El usuario debe haber elegido un tema de corpus y un idioma.
FLUJO PRINCIPAL	Se comprueba que la base de datos, definida por el corpus y el idioma, está creada. [1] En primer lugar se recuperan los datos que proporcionan la información sobre la forma, categoría y lema de cada una de las palabras de la frase: Estos datos son almacenados en la base de datos. En segundo lugar se recuperan las dependencias que proporciona el análisis sintáctico. De manera que se guarda en la base de datos la relación entre las palabras que forman una dependencia y la etiqueta asociada a esa dependencia. La información relativa a las palabras que forman una dependencia vendrá dada por el primer paso descrito en esta tabla, es decir, por las formas, categorías y lemas.[2]
FLUJO EXCEPCIONAL	[1] En caso de que la base de datos no esté creada se procede a su creación. [2] Para una determinada frase el analizador sintáctico puede no devolver ninguna dependencia. En este caso el sistema sólo almacena la información relativa a cada palabra (paso 1 del flujo principal).
POSTCONDICIONES	En la base de datos se encontrará la información sintáctica y morfosintáctica de la frase analizada.

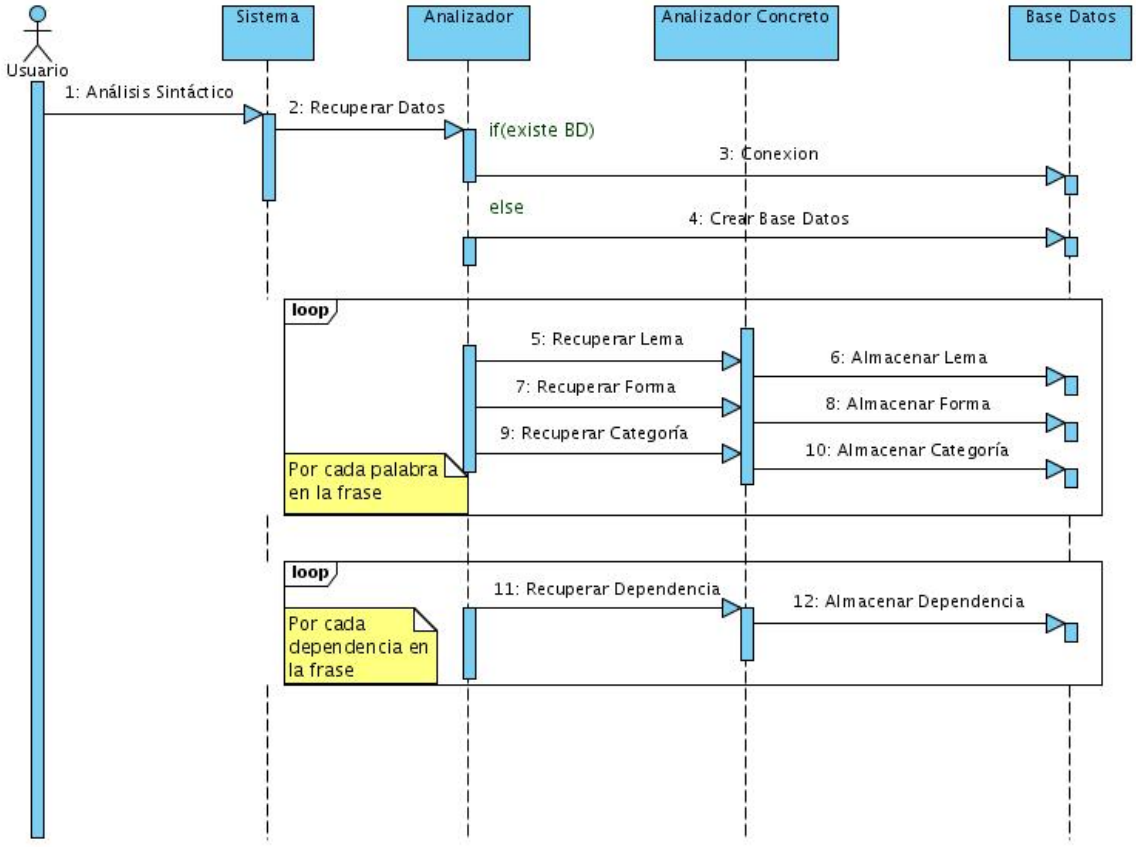


Figura 30 análisis diag.sec.almacenar análisis

5.3.CASO DE USO: GESTIÓN BÚSQUEDAS

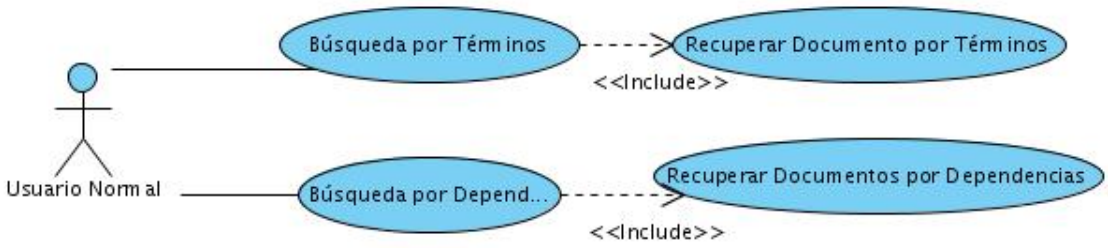


Figura 31 caso de uso gestión búsquedas

5.3.1.BÚSQUEDA POR TÉRMINOS

Búsqueda por términos	
DESCRIPCIÓN	A través del fichero de configuración el usuario introduce el corpus sobre él que desea realizar la búsqueda, así como el idioma. Una vez dado esos datos se solicita al usuario que introduzca sobre cuales de las ontologías creadas a partir de los términos desea realizar la búsqueda y recuperación de documentos.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus, el idioma y los ficheros contenedores de las distintas ontologías. Que exista el corpus conformado por los documentos.
FLUJO PRINCIPAL	El usuario introduce el tema del corpus. [1] El usuario introduce el idioma asociado en función del tema elegido. El usuario introduce el fichero sobre él que se va a realizar la búsqueda [2] El sistema recupera los datos oportunos para la operación de búsqueda.
FLUJO EXCEPCIONAL	[1] No existe ningún documento asociado con ese corpus, por consiguiente, tampoco existe ningún índice asociado al mismo. El sistema informa de ello. [2] No existe ninguna ontología creada para ese corpus y ese idioma. El sistema informa de ello.
POSTCONDICIONES	Las informaciones relativas al corpus, al idioma y al fichero de ontología conocida.

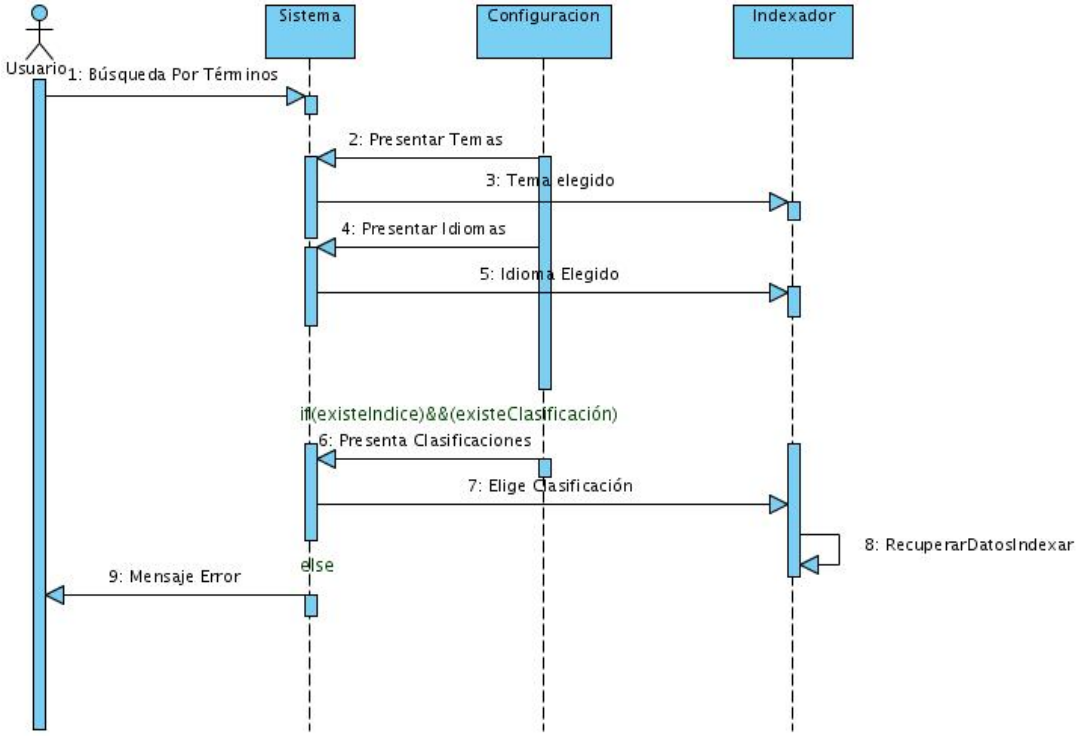


Figura 32 análisis diag.sec.búsqueda término

5.3.2.RECUPERAR DOCUMENTO POR TÉRMINOS (ESCENARIO 1): CON ONTOLOGÍA

Recuperar documentos por términos (Escenario 1): Recuperar por ontología	
DESCRIPCIÓN	En este escenario se detalla cómo se realiza la búsqueda de documentos en relación a la consulta del usuario, siendo posible expandir esta consulta a través de la ontología creada. Todo ello a partir de la clasificación previa realizada de los términos.
PRECONDICIONES	Es necesario haber recuperado todos los datos que permitan la localización de los documentos asociados al tema e idioma. El índice de los documentos asociados al corpus e idioma implicados debe estar creado Es necesario tener recuperada la ontología.
FLUJO PRINCIPAL	El usuario introduce una consulta. [1] La consulta introducida por el usuario tiene una asociación en la ontología, es decir, existe como clase de la ontología: Se recogen aquellos términos relacionados con la consulta y se amplía la misma con nuevos valores. Para la consulta, así como sus términos ampliados, se procede a la búsqueda. [2] 3. Se agrupan los resultados por consulta y términos ampliados.
FLUJO EXCEPCIONAL	[1] La consulta del usuario está vacía, el usuario escribe una nueva consulta. [2] La consulta y/o sus términos expandidos no tienen resultado en la búsqueda, el sistema para esos casos no devuelve nada, informando de que la consulta ha sido nula para esos casos.
POSTCONDICIONES	Se muestra para la consulta y para cada término expandido el conjunto de documentos que han sido indexados y recuperados.

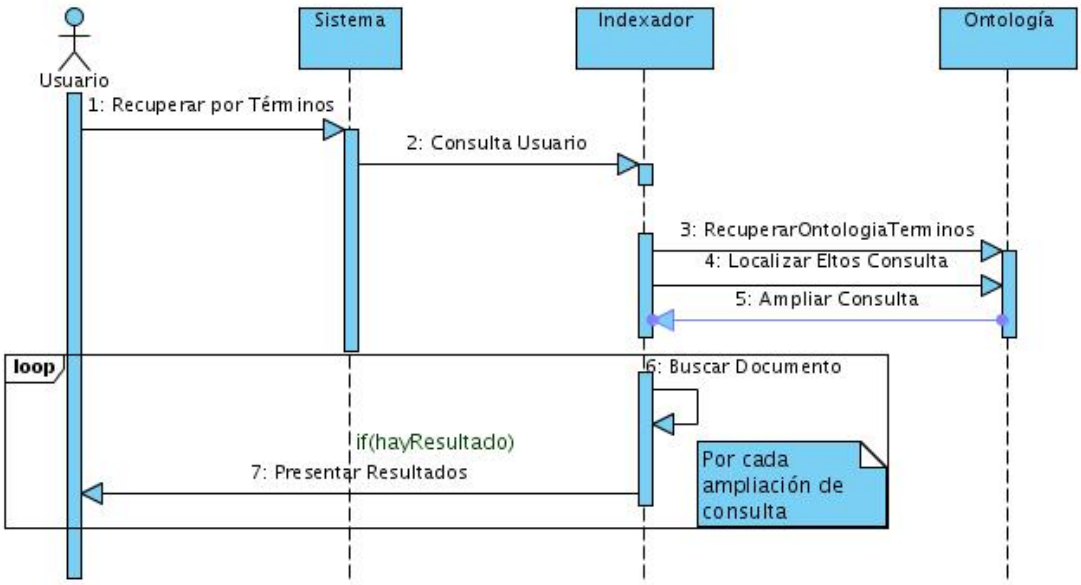


Figura 33 análisis diag.sec.recuperar documento término (escenario 1)

5.3.3. RECUPERAR DOCUMENTO POR TÉRMINOS (ESCENARIO 2): SIN ONTOLOGÍA

Recuperar documentos por términos (Escenario 2): Recuperar sin ontología	
DESCRIPCIÓN	En este escenario se detalla cómo se realiza la búsqueda de documentos en relación a la consulta del usuario, pero en este caso la consulta no puede ser ampliada, pues no existe ninguna clase asociada a la misma en la ontología.
PRECONDICIONES	Es necesario tener recuperada la ontología. El índice de los documentos asociados al corpus e idioma implicados debe estar creado.
FLUJO PRINCIPAL	El usuario introduce una consulta. [1] La consulta introducida por el usuario no tiene una asociación en la ontología. Se procede a la búsqueda únicamente con la consulta introducida. [2] Se agrupan los resultados.
FLUJO EXCEPCIONAL	[1] La consulta del usuario está vacía, el usuario escribe una nueva consulta. [2] La consulta no tiene resultado en la búsqueda, el sistema no devuelve nada, informando de que la consulta ha sido nula.
POSTCONDICIONES	Se muestra para la consulta el conjunto de documentos que ha sido indexado y recuperado.

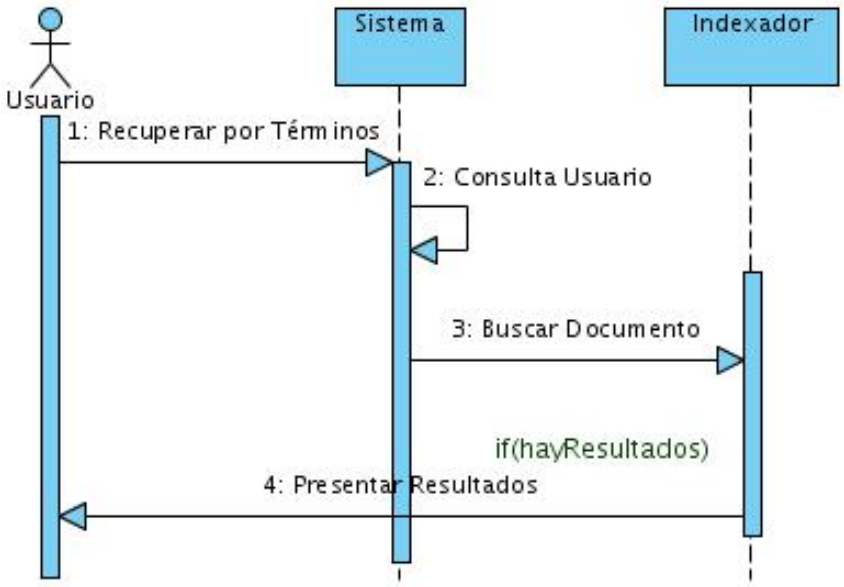


Figura 34análisis diag.sec.recuperar documento término (escenario 2)

5.3.4.BÚSQUEDA POR DEPENDENCIAS

Búsqueda por dependencias	
DESCRIPCIÓN	A través del fichero de configuración el usuario introduce el corpus sobre él que desea realizar la búsqueda, así como el idioma. El sistema recupera la ontología creada a partir de las dependencias para el corpus e idioma.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. Que exista la ontología generada a partir del grafo de dependencias.
FLUJO PRINCIPAL	El usuario introduce el tema del corpus. [1] El usuario introduce el idioma asociado en función del tema elegido. El sistema recupera el fichero de ontología, así como los demás datos oportunos.
FLUJO EXCEPCIONAL	[1] No existe ningún documento asociado con ese corpus, el sistema informa de ello. [2] No existe ninguna ontología creada para ese corpus y ese idioma. El sistema informa de ello.
POSTCONDICIONES	Las informaciones relativas al corpus, al idioma y al fichero de ontología son conocidas.

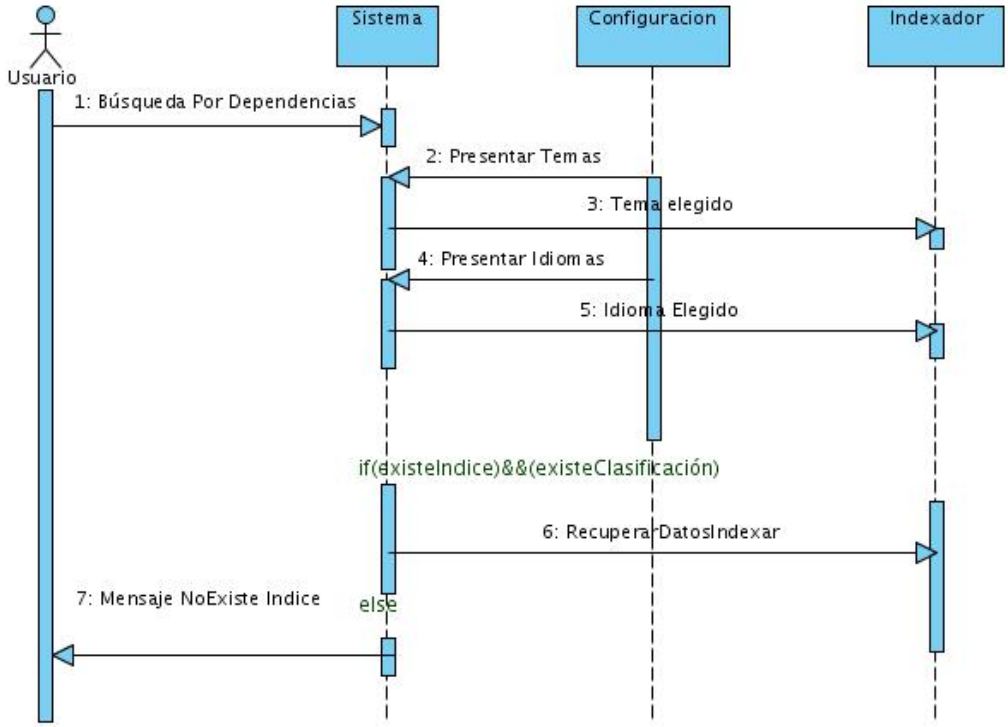


Figura 35análisis diag.sec.búsqueda dependencias

5.3.5.RECUPERAR DOCUMENTO POR DEPENDENCIAS (ESCENARIO 1): CON ONTOLOGÍA

Recuperar documentos por dependencias (Escenario 1): Recuperar por ontología	
DESCRIPCIÓN	En este escenario se detalla cómo se realiza la búsqueda de documentos en relación a la consulta del usuario, siendo posible expandir esta consulta a través de la ontología creada. Todo ello a partir del grafo de dependencias.
PRECONDICIONES	Es necesario haber recuperado todos los datos que permitan la localización de los documentos asociados al tema e idioma. El índice de los documentos asociados al corpus e idioma implicados debe estar creado. Es necesario tener recuperada la ontología.
FLUJO PRINCIPAL	El usuario introduce una consulta. [1] El sistema mira si la consulta puede ser traducida a los demás idiomas del tema actual. [2] La consulta introducida por el usuario tiene una asociación en la ontología, es decir, existe como clase de la ontología: Se recogen aquellos nodos relacionados con la consulta y se amplía la misma con nuevos valores. Para la consulta, así como sus términos ampliados, se procede a la búsqueda. [3] Se agrupan los resultados por consulta y términos ampliados.
FLUJO EXCEPCIONAL	[1] La consulta del usuario está vacía, el usuario escribe una nueva consulta. [2] Si la consulta no es traducida, se trabaja únicamente con la consulta inicial. [3] La consulta y/o sus términos expandidos no tienen resultado en la búsqueda, el sistema para esos casos no devuelve nada, informando de que la consulta ha sido nula para esos casos.
POSTCONDICIONES	Se muestra para la consulta y para cada término expandido el conjunto de documentos que han sido indexados y recuperados. Y esto es así para cada idioma traducido y para el idioma original

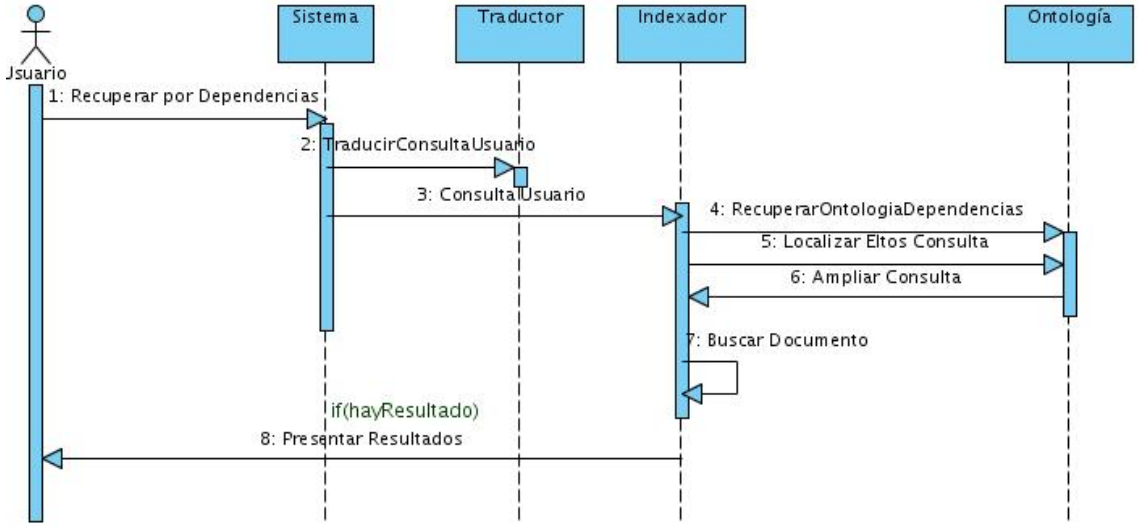


Figura 36 diag.sec.recuperar documento dependencias (escenario 1)

5.3.6.RECUPERAR DOCUMENTO POR DEPENDENCIAS (ESCENARIO 2): SIN ONTOLOGÍA

Recuperar documentos por dependencias (Escenario 2): Recuperar sin ontología	
DESCRIPCIÓN	En este escenario se detalla cómo se realiza la búsqueda de documentos en relación a la consulta del usuario, pero en este caso la consulta no puede ser ampliada, pues no existe ninguna clase asociada a la misma en la ontología.
PRECONDICIONES	Es necesario tener recuperada la ontología. El índice de los documentos asociados al corpus e idioma implicados debe estar creado.
FLUJO PRINCIPAL	El usuario introduce una consulta. [1] El sistema mira si la consulta puede ser traducida a los demás idiomas del tema actual.[2] La consulta introducida por el usuario no tiene una asociación en la ontología. Se procede a la búsqueda únicamente con la consulta introducida. [3] Se agrupan los resultados.
FLUJO EXCEPCIONAL	[1] La consulta del usuario está vacía, el usuario escribe una nueva consulta. [2] Si la consulta no se puede traducir se trabaja sólo con la consulta inicial. [3] La consulta no tiene resultado en la búsqueda, el sistema no devuelve nada, informando de que la consulta ha sido nula.
POSTCONDICIONES	Se muestra para la consulta el conjunto de documentos que ha sido indexado y recuperado. Y esto para cada consulta traducida y para la consulta original.

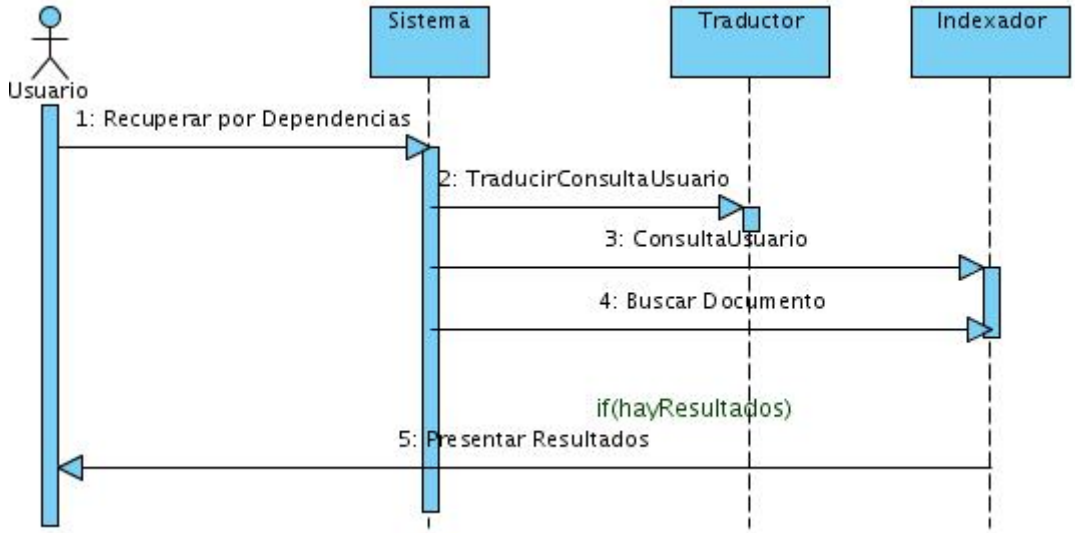


Figura 37 diag.sec.recuperar documento dependencias (escenario 2)

5.4.GESTIÓN CORPUS

En esta sección se trata de describir todos aquellos casos de uso que implican la modificación de los corpus presentes en el sistema.

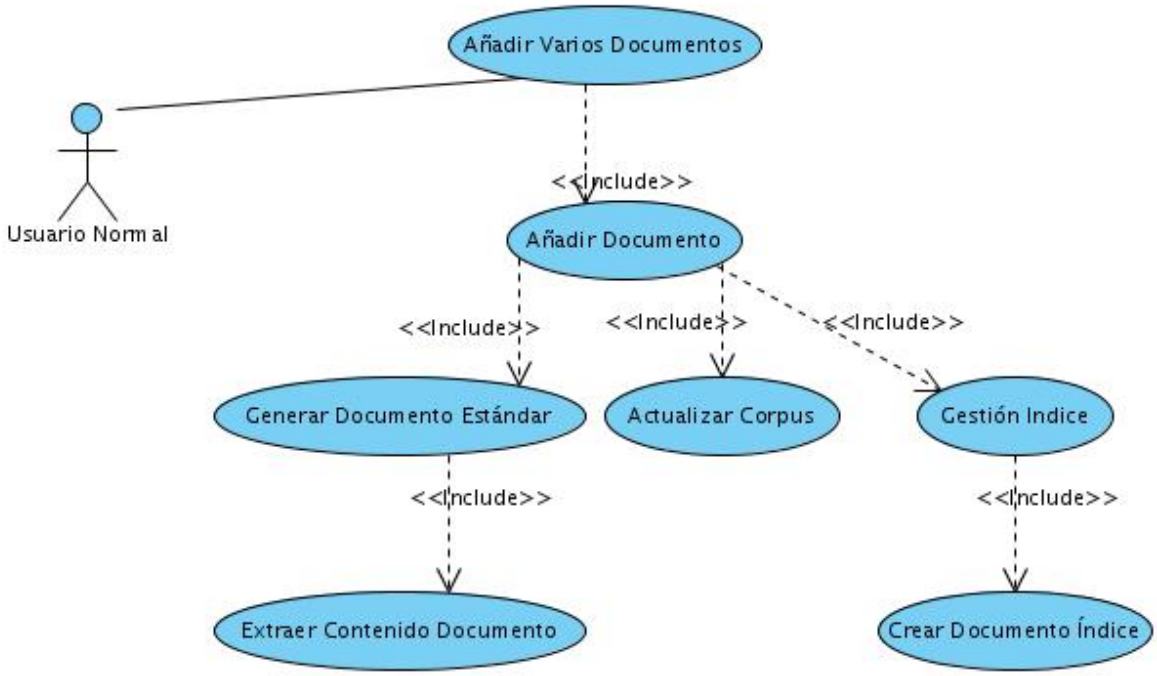


Figura 38 caso de uso gestión corpus

5.4.1.AÑADIR VARIOS DOCUMENTOS

Este caso de uso se muestra para indicar que el usuario puede introducir simultáneamente más de un documento al sistema, pero en cualquier caso el tratamiento del conjunto se reduce al tratamiento de cada documento de forma individual. Se presentan a continuación cada una de las tareas llevadas a cabo por cada documento nuevo.

5.4.2.AÑADIR UN DOCUMENTO

Añadir documento	
DESCRIPCIÓN	En este caso de uso se muestra como se añade un nuevo documento al conjunto de documentos existentes (NOTA: El documento a introducir en un momento dado, podría ser el primero de la colección.) en un corpus y/o idioma.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma.
FLUJO PRINCIPAL	El usuario indica el tema del corpus y el idioma. A partir de estos datos recupera la información necesaria para localizar donde guardar los documentos generados, los documentos proporcionados por el usuario y el corpus. Además recupera el tipo de extensión de archivo soportado para este corpus. Se recupera el documento introducido por el usuario. [1] Se examina el tipo de extensión del mismo. [2] Se procede a extraer el contenido pertinente del documento.(este caso de uso se detalla a continuación). Se genera el documento según un formato estándar definido para el sistema (este caso de uso se detalla a continuación). Se actualiza el corpus (este caso de uso se detalla a continuación). Se actualiza el análisis sintáctico (este caso de uso está explicado en los casos de uso relacionados con el análisis sintáctico). Se actualiza el índice. El fichero introducido por el usuario también se guarda.
FLUJO EXCEPCIONAL	[1] El usuario no introduce un documento válido, el sistema informa de ello. [2] Si la extensión no es válida se devuelve un mensaje de error y no se realiza la adición del documento.
POSTCONDICIONES	Los documentos se han añadido correctamente, tanto en su formato estándar como en su formato original. El corpus se ha actualizado, así como el análisis sintáctico.

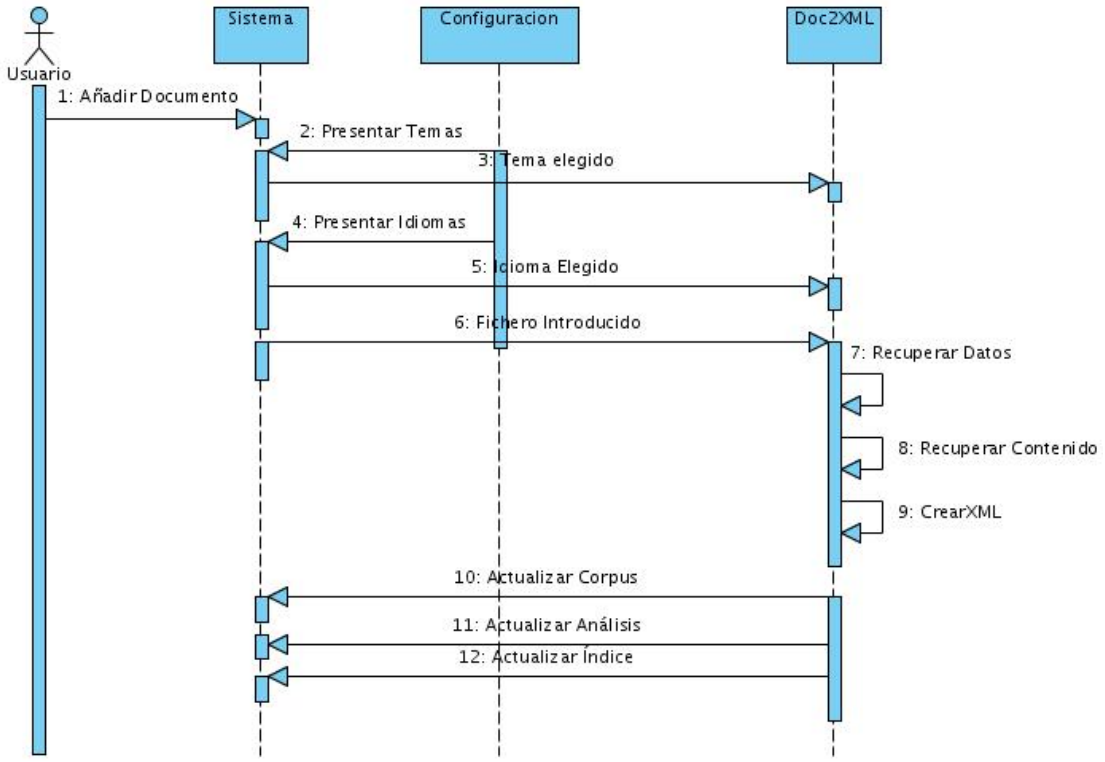


Figura 39 análisis diag.sec.añadir documento

5.4.3.EXTRAER CONTENIDO BASE

Extraer Contenido Base	
DESCRIPCIÓN	En este caso de uso se muestra como se recupera la información de un documento introducido por el usuario.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. Que exista el documento introducido por el usuario.
FLUJO PRINCIPAL	Se recupera la ruta donde se almacenan los documentos introducidos por el usuario. Se lee del fichero que indica las marcas de las noticias, es decir, los documentos introducidos por el usuario deben adaptarse a un formato específico. [1] Se recoge cada una de las secciones definidas en el documento del usuario[5]. Éstas son: título, fecha, número de palabras, sección del periódico, idioma original de la noticia, periódico que la publica, autor y finalmente el contenido propiamente dicho. Cada documento introducido por el usuario es almacenado. Para cada sección extraída se procede a su almacenamiento [2] (descrito en el caso de uso siguiente).
FLUJO EXCEPCIONAL	[1] El documento introducido por el usuario no cumple con el formato predefinido en el fichero de marcas, el sistema informa de ello y no introduce dicho documento. [2] Si alguna sección está vacía se mantiene así.
POSTCONDICIONES	Cada sección está ahora definida.

[5] Los documentos de los que se dispone actualmente son noticias de periódicos.

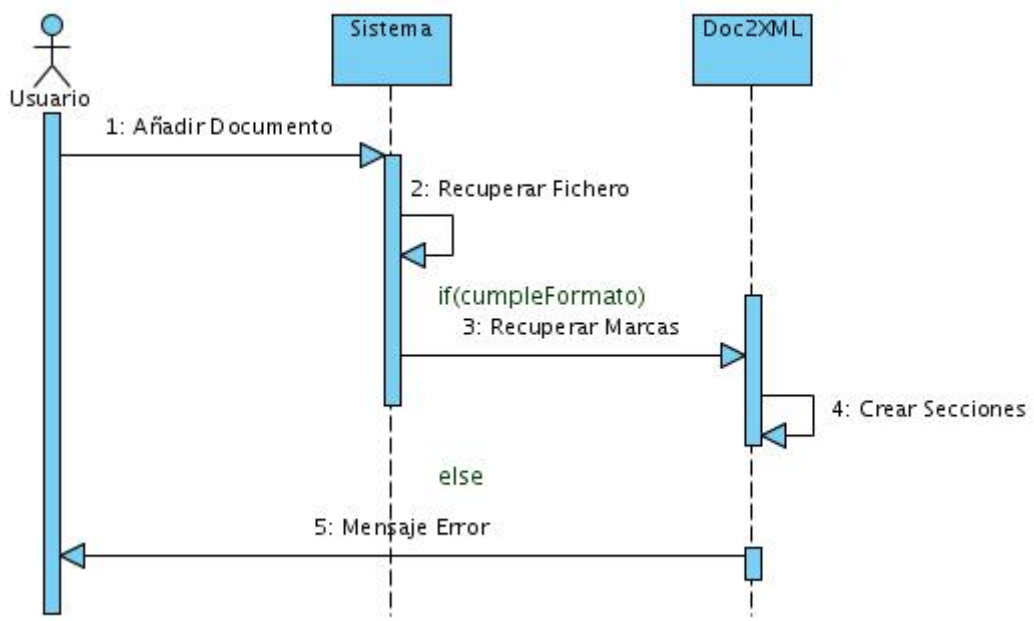


Figura 40 análisis diag.sec.extraer contenido base

5.4.4.GENERAR UN DOCUMENTO

Generar documento	
DESCRIPCIÓN	En este caso de uso se muestra cómo se almacena cada una de las secciones recuperadas de los documentos introducidos por el usuario.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. Que exista el documento introducido por el usuario. Que cada sección ya esté delimitada.
FLUJO PRINCIPAL	Se recupera la ruta donde se van a almacenar los ficheros generados. Cada fichero almacena un identificador privado que viene dado por un número único y el tipo de extensión del documento original. También guarda un identificador público que es un valor hash único para cada documento. El resto de las secciones se almacenan siguiendo el formato definido.[1] El fichero generado se almacena en la ruta recuperada en el primer paso.
FLUJO EXCEPCIONAL	[1] Es posible que alguna de las secciones permanezca vacía
POSTCONDICIONES	Para el documento introducido por el usuario, existe una versión en el formato definido para el sistema.

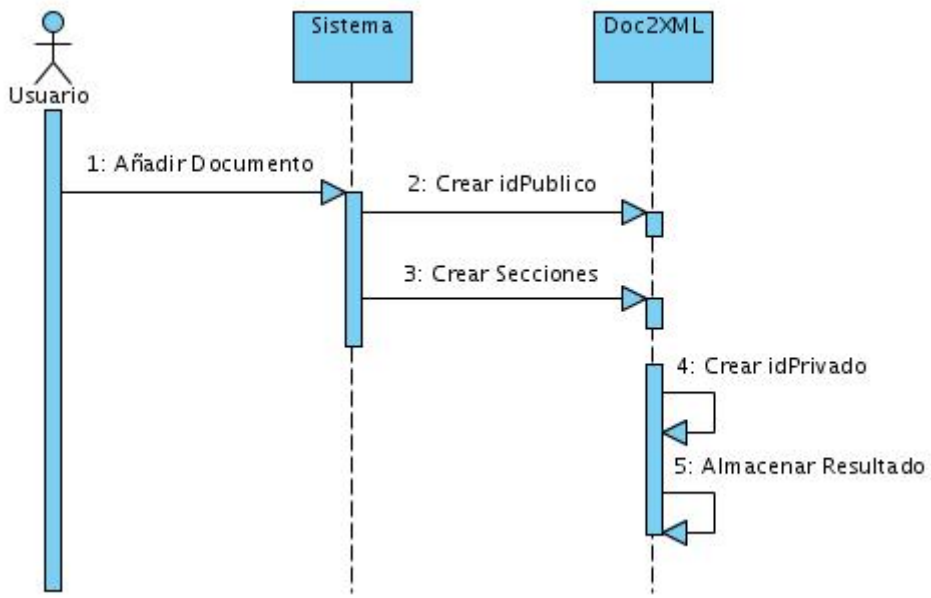


Figura 41 análisis diag.sec.generar documento

5.4.5.ACTUALIZAR CORPUS

Actualizar corpus	
DESCRIPCIÓN	En este caso de uso se describe cómo una vez introducido un nuevo documento en el sistema el corpus se actualiza. Para ello recupera el contenido del documento introducido, esto es el texto y el título y lo agrega al corpus ya creado[6].
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. Que exista el documento introducido por el usuario. El documento debe estar almacenado en el formato estándar definido para el sistema.
FLUJO PRINCIPAL	El sistema accede al último documento introducido en la arquitectura. Se lee del fichero y se recupera el título y el contenido del mismo, pues se consideran los elementos útiles para introducir en el corpus, no así, datos como la fecha o los autores. [1]
FLUJO EXCEPCIONAL	[1] En caso de que alguno de estos campos esté vacío, el sistema no guardaría ninguna información de corpus para ese documento.
POSTCONDICIONES	El corpus está actualizado y listo para su utilización en las tareas pertinentes.

[6] El documento actual podría ser el primer documento que conforma el corpus.

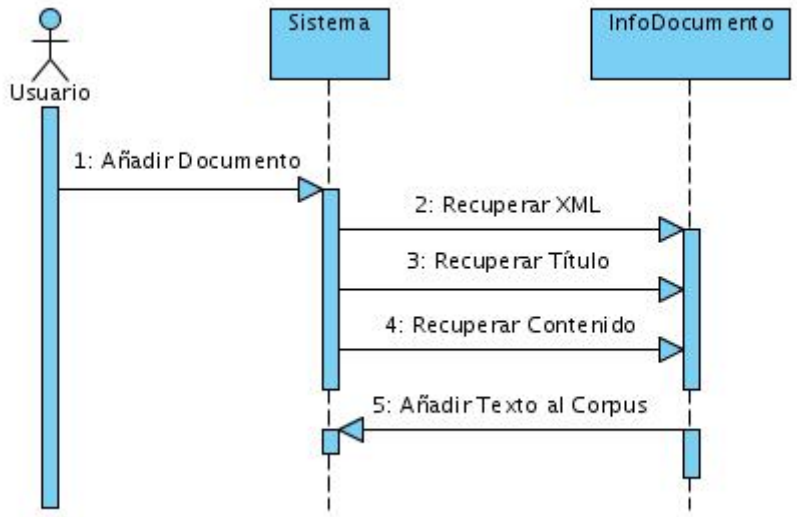


Figura 42 análisis diag.sec.actualizar corpus

5.4.6.GESTIÓN ÍNDICE (ESCENARIO 1): CREAR EL ÍNDICE

Gestión índice (Escenario 1): Crear el índice	
DESCRIPCIÓN	Se describe en este escenario, cómo a partir de una nueva colección de documentos asociados a un corpus y a un idioma se crea el índice de los mismos.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. El documento introducido por el usuario debe estar almacenado en el formato estándar definido para el sistema.
FLUJO PRINCIPAL	En un paso previo el usuario habrá introducido el tema del corpus así como el idioma. Esta información permitirá, a través del fichero de configuración, recuperar las rutas necesarias para almacenar los índices. [1] El sistema genera el índice y lo almacena.
FLUJO EXCEPCIONAL	[1] Si la ruta de conservación del índice no existe, el sistema se encarga de crearla.
POSTCONDICIONES	El índice se crea y se mantiene en su directorio correspondiente según el corpus e idioma elegidos.

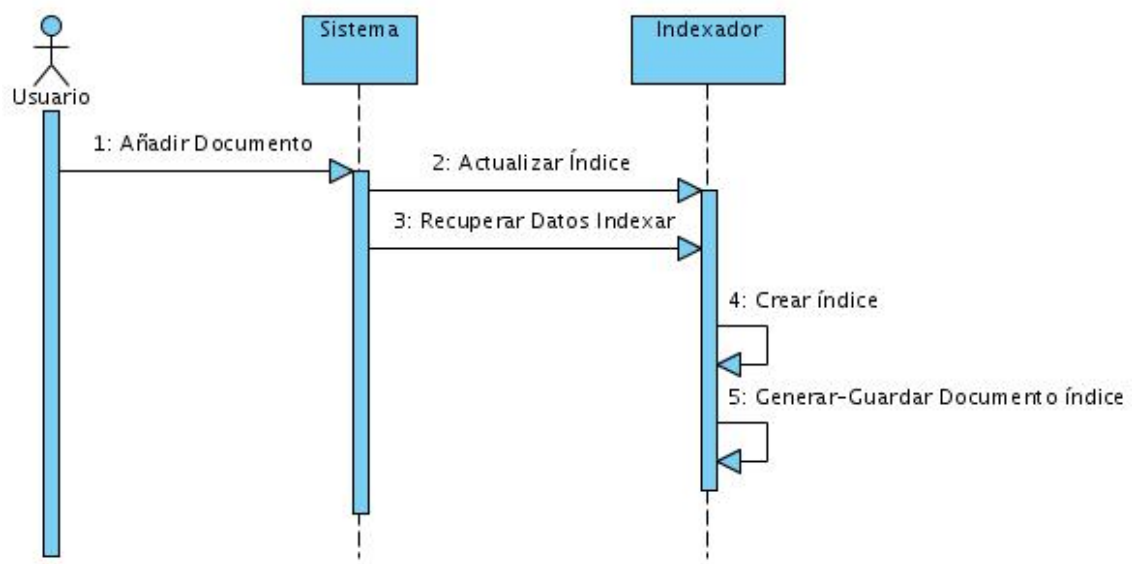


Figura 43 análisis diag.sec.crear indice

5.4.7. GESTIÓN ÍNDICE (ESCENARIO 1): ACTUALIZAR EL ÍNDICE

Gestión índice (Escenario 2): Actualizar el índice	
DESCRIPCIÓN	Se describe en este escenario, cómo cada nuevo documento añadido al sistema implica la actualización del índice previamente creado.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. El documento introducido por el usuario debe estar almacenado en el formato estándar definido para el sistema. El índice debe estar creado.
FLUJO PRINCIPAL	Se generan los documentos de índice (ver caso de uso siguiente) a partir de los documentos introducidos por el usuario almacenados de forma estándar. Cada uno de los documentos generado es añadido al índice.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	El índice está actualizado con los nuevos documentos añadidos.

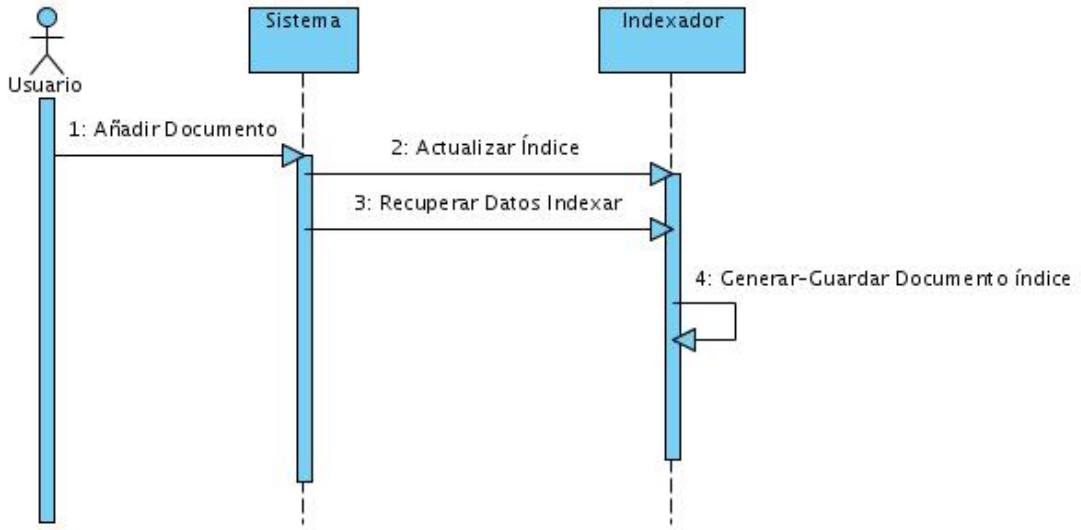


Figura 44 análisis diag.sec.actualizar índice

5.4.8.CREAR DOCUMENTO ÍNDICE

Crear documento índice	
DESCRIPCIÓN	Se describe en este caso de uso cómo se generan los documentos que son añadidos al índice.
PRECONDICIONES	Que exista el fichero de configuración. Que exista la asociación entre el tema reflejado en el corpus y el idioma. El documento introducido por el usuario debe estar almacenado en el formato estándar definido para el sistema. El índice debe estar creado.
FLUJO PRINCIPAL	Se recuperan los documentos introducidos por el usuario La información considerada pertinente para la indexación de documentos, son el título, el texto, el identificador privado, el tema y el idioma. Se recupera la información y se crea el documento de índice con los campos mencionados. El documento creado se asocia al índice.
FLUJO EXCEPCIONAL	
POSTCONDICIONES	El índice puede indexar sus búsquedas incluyendo los nuevos documentos creados

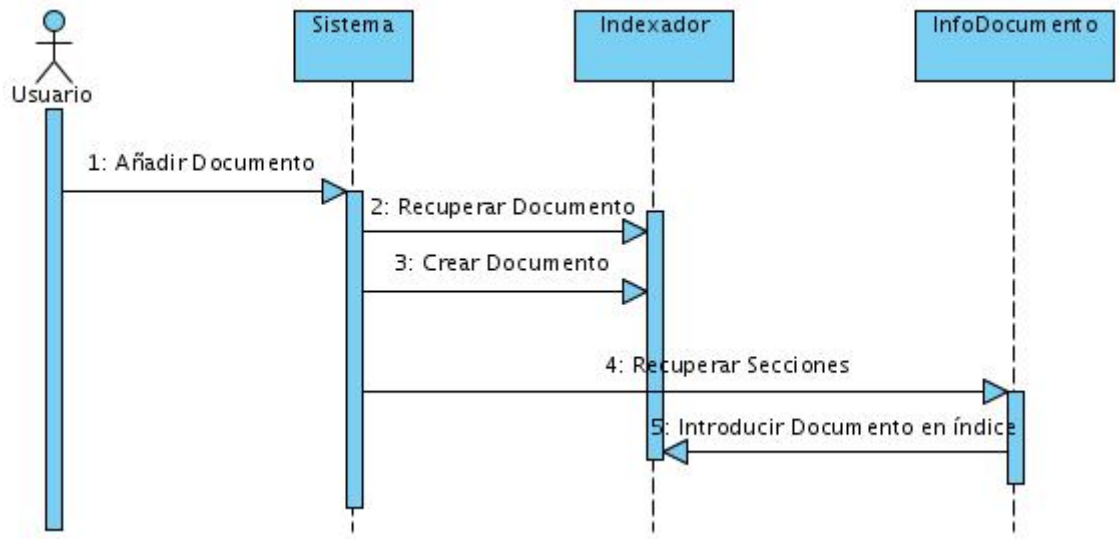


Figura 45 análisis diag.sec.crear documento índice

DISEÑO

6. ESPECIFICACIÓN DE LAS CLASES

En este apartado del Manual Técnico se pasa a detallar las clases generadas durante el proceso de diseño del sistema. Se ha optado por presentar las mismas encapsuladas en los paquetes que las contienen.

6.1.PAQUETE RECURSOS.PARSER

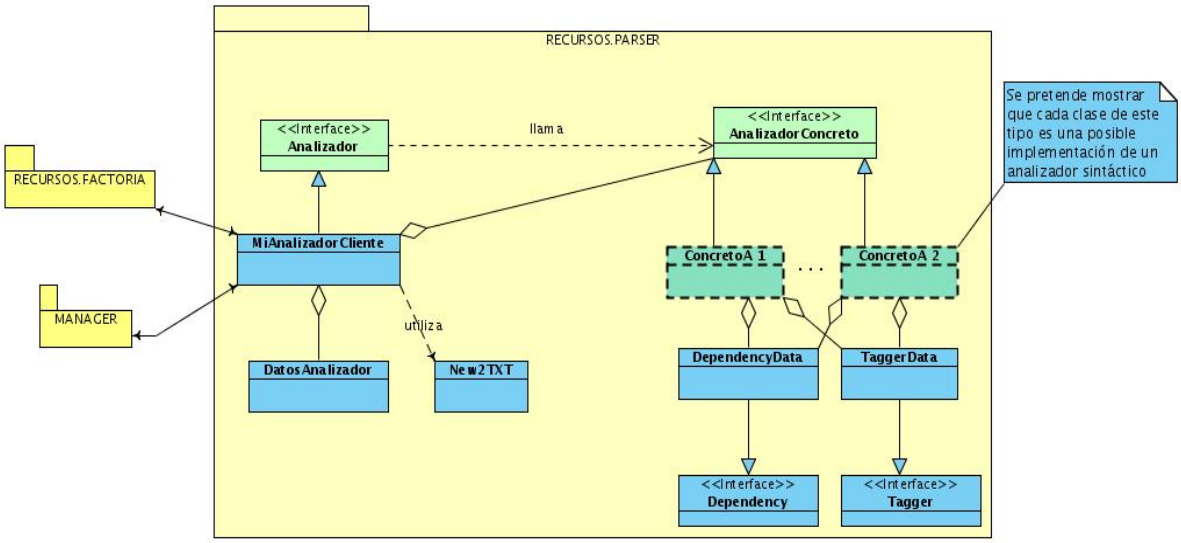


Figura 46 diseño paquete recursos.parser

6.1.1.DICCIONARIO DE CLASES

ANALIZADOR	
	Descripción
	Esta clase es una interfaz para manipular los analizadores. .
Atributos	No tiene atributos
Métodos	rtvResource: Recupera a partir del fichero de configuración la relación entre un idioma y su abreviatura callConcretResource: Llama al analizador concreto que ha solicitado el usuario setNews: Método encargado de llamar a la clase New2Text para obtener el contenido relevante de los documentos execParser: Método que lanza el análisis sintáctico y llama a los métodos que manipulan la base de datos

MI ANALIZADOR CLIENTE ANALIZDOR	
	Descripción
	Esta clase implementa la interfaz presentada anteriormente.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. parser: Almacena el analizador actual. clase: Clase asociada al recurso a ejecutar. rutaAnalizador: Almacena la ruta donde se almacena el analizador, en caso de que éste sea implementado por una herramienta independiente. abreviatura: Almacena la abreviatura del idioma actual. path: Almacena el directorio donde se encuentra instalada la aplicación. gestor: Es el gestor actual de la base de datos. nb_baseDatos: Es el nombre de la base de datos que va a guardar los resultados procedentes del análisis sintáctico. objectDoc: Es un atributo que referencia al documento de configuración. idIdioma: Almacena la relación entre los idiomas y las abreviaturas. idAnalizador: Almacena la relación entre los idiomas y los analizadores.
Métodos	Además de los métodos descritos en la interfaz, incorpora los siguientes: MiAnalizadorCliente: Constructor de la clase. Inicializa los valores de tema, idioma y analizador. recuperarAnalizadorConcreto:Localiza la clase asociada al recurso a ejecutar así como la ruta donde se ubica el analizador.

DATOS ANALIZADOR	
	Descripción
	Esta clase es una clase de apoyo para recopilar toda la información referente a un analizador concreto.
Atributos	nbAnalizador: Atributo que almacena el nombre del analizador. rutaAnalizador: Atributo que almacena la ruta donde se ubica el analizador, si éste es un recurso externo. clase: Atributo que almacena la clase que ejecuta la lógica del analizador
Métodos	DatosAnalizador: Constructor de la clase. set/getNbAnalizador: Estos métodos inicializan y devuelven el nombre del analizador actual. set/getRutaAnalizador: Estos métodos inicializan y devuelven la ruta del analizador actual. set/getAnalizador: Estos métodos inicializan y devuelven el nombre de la clase que manipula el analizador actual.

NEWS2TEXT	
	Descripción
	Esta clase es empleada para manejar el contenido de los documentos analizados y así poder recuperar el contenido de los mimos. Además se utiliza para crear los directorios donde se van a almacenar las frases analizadas.
Atributos	idioma: Almacena el idioma actual. rutaXML: Almacena el lugar donde se ubica el fichero XML que se está analizando. rutaNews: Almacena la ruta donde guardar el fichero temporal con el contenido relevante. directorioFrase: Cadena que almacena la ruta del directorio donde se van a guardar las frases extraídas
Métodos	News2Text: Constructor de la clase. setTextNoticia: Recupera del documento XML las secciones consideradas con contenido, éstas son, el título y el texto. getDirectorioFrases: Permite recuperar el directorio donde se almacenan las frases.

ANALIZDOR CONCRETO	
	Descripción
	Es una interfaz que describe el comportamiento que deben tener los analizadores que se introducen en el sistema
Atributos	
Métodos	setValores: Inicializa los valores de los atributos. startParser: Ejecuta el analizador. getVectorDependency: Método que devuelve el vector con las dependencias resultado del análisis sintáctico. getVectorTagger: Método que devuelve el vector con los resultados del etiquetador, esto es, el lema, forma y categoría de cada palabra. getSentence: Método que devuelve la frase actual analizada.

Las dos clases que se aprecian en la Figura 46 con líneas discontinuas representan las posibles implementaciones que pueden existir en el sistema de un determinado analizador sintáctico. En el caso concreto que se ha desarrollado, ésta ha sido la implementación.

ANALIZDOR_A 1	
	Descripción
	Implementa la interfaz AnalizadorConcreto
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. ruta: Almacena la ruta donde se ubica el recurso. vectorEtiquetador: Vector que guarda elementos de tipo TaggerData donde se almacena la información sobre la etiquetación. vectordependencias:Vector que guarda elementos de tipo DatosAnalizador donde se almacena la información sobre la etiquetación. fraseAnalizada: Atributo que almacena la frase actual.
Métodos	Además de los métodos descritos en la interfaz, esta clase desarrolla los siguientes: execEtiquetar: Ejecuta la etiquetación. salvarAnalisis: Método que salva en una estructura de datos los resultados procedentes del analizador sintáctico. salvarTagger: Método que salva en una estructura de datos los resultados procedentes del etiquetador.

DEPENDENCY	
	Descripción
	Esta es una interfaz de apoyo para que las clases que la implementen almacenen los resultados que devuelve el analizador sintáctico.
Atributos	
Métodos	DependencyData: Constructor de clase que inicializa los valores de los atributos. getLabel: Devuelve la etiqueta que recibe la dependencia. getLema1: Devuelve el primer lema de la dependencia. getLema2: Devuelve el segundo lema de la dependencia. getPos1: Devuelve la posición del primer lema de la dependencia. getPos2: Devuelve la posición del segundo lema de la dependencia.

TAGGER	
	Descripción
	Esta interfaz describe una implementación para recuperar la información morfológica y léxica de cada palabra analizada.
Atributos	
Métodos	getLabel: Devuelve la categoría léxica asignada a una palabra. getLema: Devuelve el lema de una palabra. getForma: Devuelve la forma de una palabra.

En la Figura 46 se puede ver además las clases *DependencyData* y *TaggerData*, ambas son implementaciones de las dos interfaces descritas anteriormente. En este caso ambas son implementaciones concretas realizadas para los analizadores empleados en el sistema (ErialES y ErialFR). Por lo que ya ha quedado explicado a través de sus interfaces el cometido de cada uno.

6.2.PAQUETE RECURSOS.FACTORIA

Este paquete sigue la estructura planteada por el patrón DAO en combinación con una fábrica abstracta.

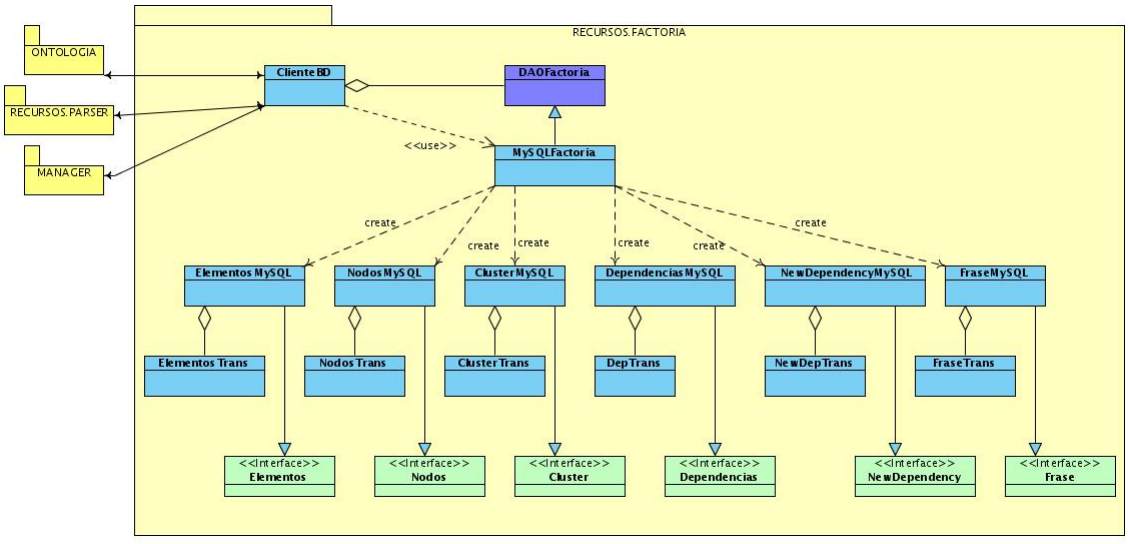


Figura 47 diseño paquete recursos.factoría

6.2.1.DICCIONARIO DE CLASES

CLIENTEBD	
	Descripción
	Esta clase permite al resto del sistema comunicarse con la base de datos.
Atributos	miBD: Atributo que guarda el nombre de la base de datos actual. login: Atributo que guarda el login de la base de datos. password: Atributo que guarda la contraseña de la base de datos. url: Atributo que guarda la dirección del host que ubica la base de datos. driver: Nombre del controlador necesario para el manejo del sistema gestor de base de datos actual. tema: Nombre del tema actual.
Métodos	ClienteBD: Constructor de la clase. recuperarLematizaciónDocumento: Método que permite recuperar un documento lematizado. salvarRelaciones: Método que permite almacenar en la base de datos todos los datos procedentes del analizador sintáctico actual. Este método llama a los métodos de las clases pertinentes

DAOFACTORIA	
	Descripción
	Esta es una clase abstracta que va a facilitar la incorporación de nuevos formatos de almacenamiento del grafo de dependencias resultado del análisis sintáctico.
Atributos	
Métodos	A continuación se presentan el conjunto de métodos abstractos que define esta clase. Estos métodos serán implementados por la clase que herede de DAOFactoria (MySQLFactoria), de modo que creara una instancias de las clases relacionadas con cada método. getElementosDAO: tiene que devolver una instancia de Elementos. getNodosDAO: tiene que devolver una instancia de Nodos. getClusterDAO: tiene que devolver una instancia de Cluster. getNewDependencyDAO: tiene que devolver una instancia de NewDependency. getDependenciaDAO: tiene que devolver una instancia de Dependencia. Además también están incluidos los siguiente métodos: getLogin: Devuelve el login de la base de datos. getPassword: Devuelve la contraseña de la base de datos. getURL: Devuelve la ruta donde se ubica la base de datos. getDriver: Devuelve el controlador que maneja la base de datos.

MYSQLFACTORIA	
	Descripción
	Esta es clase implementa la interfaz descrita en el punto anterior.
Atributos	RUTA_BD: Atributo que guarda la ruta hacia donde se ubica la base de datos. login: Atributo que guarda el login de la base de datos. password: Atributo que guarda la contraseña de la base de datos. url: Atributo que guarda la dirección del host que ubica la base de datos. driver: Nombre del controlador necesario para el manejo del sistema gestor de base de datos actual.
Métodos	comprobarBD: Método que comprueba si la base de datos ya existe. createBD: Método que crea la base de datos. createTAblas: Método que crea la estructura de tablas de las base de datos. getLogin: Método que devuelve el login actual de la base de datos. getPassword: Método que devuelve la contraseña actual de la base de datos. getUrl: Método que devuelve la ruta hacia donde se ubica la base de datos. getDriver: Método que devuelve el nombre del controlador necesario para el manejo del sistema gestor de base de datos actual. getElementosDAO: Devuelve un objeto de tipo ElementosDAO. getNodosDAO: Devuelve un objeto de tipo NodosDAO getDependenciasDAO: Devuelve un objeto de tipo DependenciasDAO getClusterDAO: Devuelve un objeto de tipo ClusterDAO getNewDependency: Devuelve un objeto de tipo NewDependencyDAO

A continuación se presenta la descripción de cada una de las clases que referencian a una tabla de la base de datos. Cada una de estas clases implementa una interfaz. Para no repetir información redundante se presenta aquí solamente la descripción de las clases, que ya describen el comportamiento encerrado en la interfaz.

ELEMENTOSMYSQL	
	Descripción
	Gestiona y maneja la tabla elementos.
Atributos	
Métodos	inserta: Método que inserta un nuevo elemento en la base de datos. Este elemento puede ser bien un lema, bien una forma o bien una categoría. consultaId: Método que devuelve el identificador de fila de un determinado elemento.

NODOSMYSQL	
	Descripción
	Gestiona y maneja la tabla nodos.
Atributos	
Métodos	inserta: Método que inserta un nuevo nodo en la base de datos. consultaId: Método que devuelve el identificador de fila de un determinado nodo. nodosINTOcluster: Método que devuelve el identificador del cluster incluido en el nodo. getFraseLematizada: Método que recoge la frase actual, a partir de sus lemas.

CLUSTERMYSQL	
	Descripción
	Gestiona y maneja la tabla cluster.
Atributos	
Métodos	inserta: Método que inserta un nuevo cluster en la base de datos. consultaId: Método que devuelve el identificador de fila de un determinado cluster.

DEPENDENCIASMYSQL	
	Descripción
	Gestiona y maneja la tabla dependencias.
Atributos	
Métodos	inserta: Método que inserta un nueva dependencia en la base de datos. Los métodos explicados a continuación se llevan a cabo para introducir la información en la tabla NewDependency (ver apartado 9) . sacaDepSUJOD: Método que extraer las dependencias entre un sujeto y un objeto directo, pues las palabras implicadas en tales dependencias se relacionan a través de un verbo y es interesante mantener la relación entre ellas y no con el verbo. crearDepSUJODIndependientes: Método que extrae las dependencias bien de un sujeto bien de un objeto directo, en las cuales no existe relación con un objeto directo o un sujeto, respectivamente. sacaRestoDep: Método que extrae el resto de las dependencias en las que no se incluye una relación con sujetos y objetos directos.

NEWDEPENDENCYMYSQL	
	Descripción
	Gestiona y maneja la tabla newdependency.
Atributos	
Métodos	inserta: Inserta las nuevas dependencias en la tabla correspondiente. recuperaDepRelevantes: Una vez ejecutado el algoritmo de clasificación por dependencias, en este método se recuperan aquellas dependencias relevantes junto a la relación que surge entre los elementos de esas dependencias.

FRASEMYSQL	
	Descripción
	Gestiona y maneja la tabla frase.
Atributos	
Métodos	inserta: Inserta la nueva frase en la tabla correspondiente.

Cada una de las clases presentadas arriba, tiene asociado una clase transporte. Este tipo de clases almacenan mediante atributos los valores de los campos de las tablas de la base de datos (ver diagrama de entidad relación en apartado: *9.1*). De forma que proporcionan métodos de establecimiento y de recuperación de cada uno de estos valores. A modo de ejemplo, para la clase *ClusterMySQL* existe un nodo transporte (*ClusterTransoprte*) con los siguiente atributos y métodos:

CLUSTERMYSQL	
	Descripción
	Gestiona y maneja la tabla frase.
Atributos	identificador: Identificador de la tabla cluster. codForma: Identificador de la tabla forma. codFrase: Identificador de la tabla frase. Posición: Posición en la frase actual.
Métodos	set/getIdentificador: Estos métodos inicializan y devuelven el identificador de cluster. set/getForma: Estos métodos inicializan y devuelven el identificador de forma. set/getFrase: Estos métodos inicializan y devuelven el identificador de frase. set/getPosición: Estos métodos inicializan y devuelven el la posición.

De esta forma cada una de las demás clases llevan asociada una clase transporte.

6.3.PAQUETE RECURSOS.EXTRACTOR

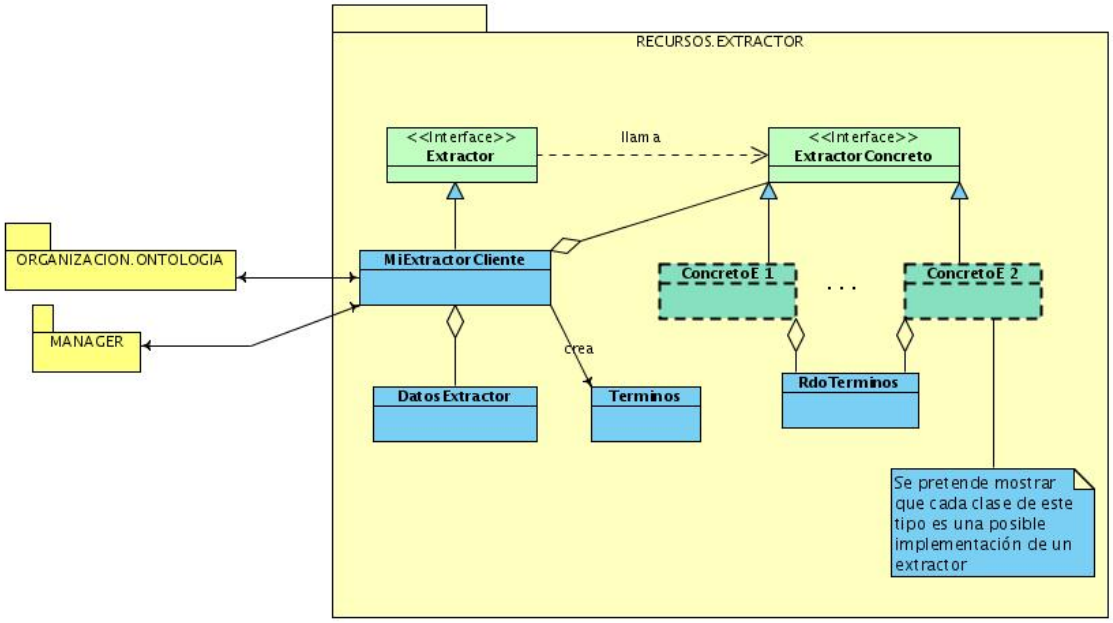


Figura 48 diseño paquete recursos.extractor

6.3.1.DICCIONARIO DE CLASES

EXTRACTOR	
	Descripción
	Esta interfaz describe el comportamiento de las clases que lanzan la extracción de términos.
Atributos	
Métodos	rtvResource: Recupera a partir del fichero de configuración la relación entre un idioma y su abreviatura. callConcretResource: Llama al extractor concreto que ha solicitado el usuario salvaExtraccion: Guarda los resultados devueltos por la extracción. Se encarga además de eliminar elementos repetidos en los términos extraídos.

MIEXTRACTORCLIENTE	
	Descripción
	Esta clase implementa la interfaz Extractor. En ella se llevan a cabo aquellas operaciones que permiten lanzar la extracción de términos.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. extractores: Vector con los extractores que se van a ejecutar en este instante. clase: Clase asociada al recurso a ejecutar. rutaExtractor: Almacena la ruta donde se almacena el extractor, en caso de que éste sea implementado por una herramienta independiente. path: Almacena el directorio actual donde está situada la aplicación. archivoRdoXML: Fichero donde se va a almacenar el resultado de la extracción. documento: Almacena el documento que se carga en memoria con la configuración. idIdioma: Almacena la relación entre los idiomas y las abreviaturas idExtractor: Almacena la relación entre los idiomas y los extractores. objectDoc: Es un atributo que referencia al documento de configuración.
Métodos	Además de implementar los métodos expuestos en su interfaz, también incluye los siguientes métodos: MiExtractorCliente: Inicializa los valores de tema, idioma y extractores. recuperaExtractorConcreto: Localiza la clase asociada al recurso a ejecutar, así como la ruta donde se ubica el extractor. También recupera el corpus. quitarRepetidos: En este método se comprueba si alguno de los términos extraído aparece dos veces.

DATOSEXTRACTOR	
	Descripción
	Permite recuperar los datos de ejecución de los extractores.
Atributos	idioma: Guarda el idioma asociado al extractor. nbExtractor: Nombre del extractor. rutaExtractor: Ruta donde se almacena el extractor, si éste es un recurso externo. clase: Almacena nombre de la clase que implementa la extracción.
Métodos	DatosExtractor: Constructor de la calse. set/getNbExtractor: Estos métodos inicializan y devuelven el nombre del extractor actual. set/getRutaExtractor: Estos métodos inicializan y devuelven la ruta del extractor actual. set/getClase: Estos métodos inicializan y devuelven el nombre de la clase que manipula el extractor actual. set/getIdioma: Estos métodos inicializan y devuelven el nombre del idioma del extractor actual

TERMINOS	
	Descripción
	Esta clase se encarga de la gestión del fichero que almacena los datos procedentes de la extracción.
Atributos	terminos: Vector con los términos extraídos. extractores: Vector con los extractores implicados. rutaRdo: Ruta donde ubicar el resultado.
Métodos	Terminos: Constructor de la clase. getExtractoreEmpleados: Devuelve los extractores implicados en la extracción. createXMLTerm: Genera el XML con los términos extraídos. createCabeceraXML: Genera la cabecera del fichero XML que contendrá los resultados de la extracción. setDocTerminos: Método que permite cargar en memoria un fichero que contiene resultados de una extracción.

RDOTERMINOS	
	Descripción
	Es una clase de apoyo que almacena los datos de cada uno de los términos extraídos.
Atributos	frecuencia: Si el extractor devuelve la frecuencia de aparición, se puede recuperar con este atributo. término: Almacena el término actual. extractor: Es un vector que almacena los extractores que han devuelto ese término.
Métodos	RdoTerminos: Constructor de clase que inicializa los atributos. getTermino: Devuelve el término. getExtractor: Devuelve el conjunto de extractores. getFrecuencia: Devuelve la frecuencia.

EXTRACTORCONCRETO	
	Descripción
	Es una interfaz que describe el comportamiento que deben tener los analizadores que se introducen en el sistema.
Atributos	
Métodos	setValores: Constructor de la clase. etiquetador: Ejecuta la etiquetación. execExtract: Ejecuta el extractor. getTerminos: Devuelve un vector en el cual se almacenan los términos extraídos.

Las dos clases que se aprecian en la Figura 48 con líneas discontinuas representan las posibles implementaciones que pueden existir en el sistema de un determinado extractor. En el caso concreto que se ha desarrollado ésta ha sido la implementación:

EXTRACTOR_E 1	
	Descripción
	Implementa la interfaz ExtractorConcreto
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. ruta: Almacena la ruta donde se ubica el recurso. path:Almacena el directorio donde se encuentra instalada la aplicación. vTerminos: Vector que almacena los términos extraídos.
Métodos	Los métodos ya están descritos en su interfaz.

6.4.PAQUETE MANAGER

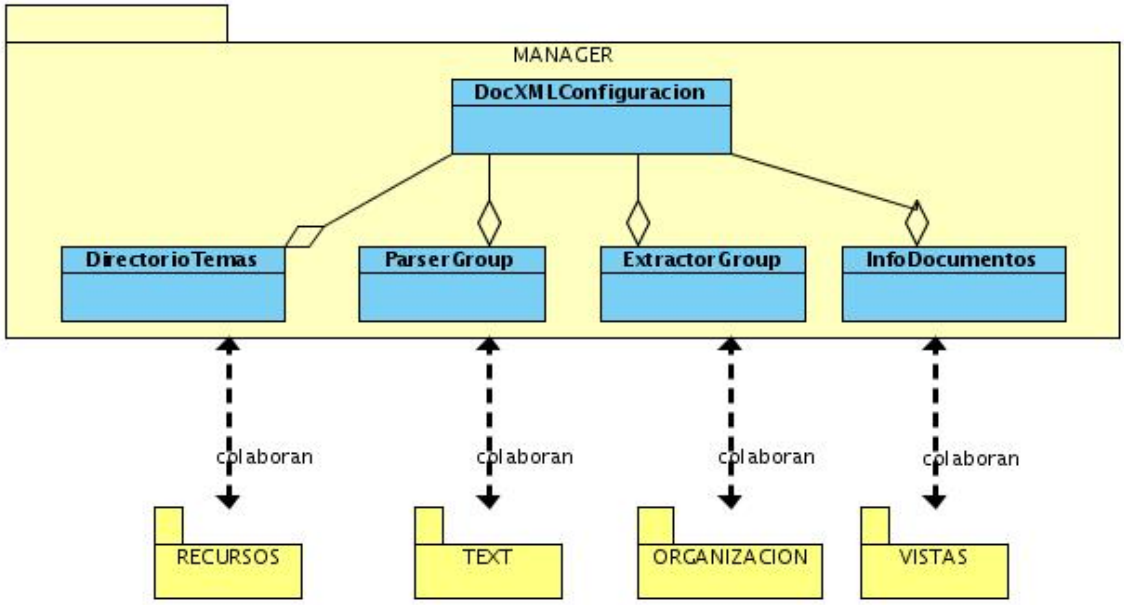


Figura 49 diseño paquete manager

6.4.1.DICCIONARIO DE CLASES

DOCXMLCONFIGURACION	
	Descripción
	Clase que manipula el fichero de configuración.
Atributos	abreviaturas: Vector con las abreviaturas existentes en el sistema que se corresponden con un idioma. analizadores: Vector con los analizadores existentes en el sistema. extractores: Vector con los extractores existentes en el sistema. idiomas: Vector con los idiomas existentes en el sistema. temas: Vector con los temas existentes en el sistema. abrIdioma: Vector que almacena las abreviaturas con sus idiomas abrStopWords: Vector que almacena si dado un idioma tiene asociado un fichero con palabras de parada. bdIdioma: Vector que almacena la relación entre un idioma y las bases de datos relacionadas con él. temaVSabr: Vector que almacena la relación entre la abreviatura de los idiomas y el tema al que pertenecen.
Métodos	set/getAnalizadores: Estos métodos inicializan y devuelven los analizadores existentes en el sistema. set/getExtractores: Estos métodos inicializan y devuelven los extractores existentes en el sistema. set/getTemas: Estos métodos inicializan y devuelven los temas existentes en el sistema. set/getIdiomas: Estos métodos inicializan y devuelven los idiomas existentes en el sistema. anadirTema: Este método añade un tema al sistema. borrarTema: Este método borra un tema del sistema. anadirIdioma: Este método añade un idioma al sistema.

	borrarIdioma: Este método borra un idioma del sistema. anadirExtractor: Este método añade un extractor al sistema. borrarExtractores: Este método borra un extractor del sistema. anadirAnalizador: Este método añade un analizador al sistema. borrarAnalizador: Este método borra un analizador del sistema. getTemaVSabr: Devuelve la relación entre los temas y las abreviaturas de los idiomas que se relacionan con ese tema. getIdVSabr: Devuelve la relación entre los idiomas y sus abreviaturas. setDocXMLConfiguracion: Carga el fichero de configuración en memoria. verificaInfoDocXML: Método que llama a otros métodos para comprobar la coherencia del fichero de configuración. tema2idioma: Recoge los temas con sus idiomas relacionados. abr2idioma: Recoge las abreviaturas con sus idiomas relacionados. abr2stopW: Recoge si un idioma determinado tiene fichero asignado de palabras de parada. bd2idioma: Recoge las bases de datos relacionadas con un determinado idioma. verificaDatosBD: Verifica que los nombres de las base de datos no se repiten en el sistema. viericaDatosTemas: Verifica que los nombres de los temas no se repiten en el sistema y que las rutas asignadas al tema son correctas o existen, si no existen las crea. verificaDatosExtractor: Verifica que los nombres de los extractores no se repiten en el sistema y que las rutas asignadas al extractor son correctas o existen, si no existen las crea. verificaDatosAnalizador: Verifica que los nombres de los analizadores no se repiten en el sistema y que las rutas asignadas al analizador son correctas o existen, si no existen las crea.

DIRECTORIOTEMAS	
	Descripción
	Clase que permite recuperar la información almacenada en el fichero de configuración, sobre un tema dado.
Atributos	clase: Almacena el nombre del paquete y la clase que maneja un determinado formato. idiomaAbreviatura: Almacena el idioma actual, a través de su abreviatura. rutaXML: Atributo que guarda la ruta donde se ubican los archivos XML de un idioma y un tema dado. corpus: Atributo que almacena la ruta donde se ubica el corpus de un idioma y un tema dado. documento: Representa el documento en memoria. formato: Vector que guarda el conjunto de formatos asociados a un idioma.
Métodos	DirectorioTemas: Constructor de la clase, que guarda el idioma, el tema y carga el documento de configuración en memoria. setRutas: Método que recupera todas las rutas relacionadas con un tema, ruta con los XML , paquete donde se ubica la clase que gestiona un determinado formato, ruta donde se ubica el corpus. Recupera además los formatos asociados a un tema. getRutaXML: Devuelve las rutas donde se ubican los archivos XML de los idiomas asociados al tema. getClase: Devuelve la clase que trata a un formato dado. getCorpus: Devuelve la ruta del corpus de un idioma dado. getFormato: Devuelve los formatos asociados al tema.

PARSERGROUP	
	Descripción
	Clase que permite recuperar del fichero de configuración el conjunto de analizadores asociados a un idioma.
Atributos	documento: Almacena el documento que se carga en memoria con la configuración. parser: Vector que almacena el conjunto de analizadores asociados al idioma actual. idioma: Nombre del idioma actual. abreviatura: Nombre de la abreviatura asociada al idioma actual.
Métodos	ParserGroup: Constructor de la clase, que carga en memoria el fichero de configuración. setIdioma: Método que establece el idioma actual. rtvResource: Método que devuelve los analizadores relacionados con el idioma actual. getResource: Método que permite recuperar una estructura que almacena los analizadores asociados al idioma actual.

EXTRACTORGROUP	
	Descripción
	Clase que permite recuperar del fichero de configuración el conjunto de extractores asociados a un idioma.
Atributos	documento: Almacena el documento que se carga en memoria con la configuración. extractor: Tabla que almacena el conjunto de extractores.
Métodos	ExtractorGroup: Constructor de la clase, que carga en memoria el fichero de configuración. rtvExtractors: Método que devuelve los extractores presentes en el sistema junto al idioma que tengan asociado. getExtractors: Método que permite recuperar la estructura que almacena los extractores.

INFODOCUMENTOS	
	Descripción
	Clase que permite recuperar la información almacenada en los documentos que conforman las fuentes textuales del sistema.
Atributos	titulo: Almacena el titulo actual. tema: Almacena el tema actual. fecha: Almacena la fecha actual. palabras: Almacena el número de palabras actual. autor: Almacena el autor actual. idPublic: Almacena el identificador público actual. idPrivate: Almacena el identificador privado actual. seccion: Almacena el periódico o sección actual.
Métodos	set/getTema: Estos métodos inicializan y devuelven el tema del documento actual. set/getTexto: Estos métodos inicializan y devuelven el texto del documento actual. set/getTitulo: Estos métodos inicializan y devuelven el titulo del documento actual. set/getFecha: Estos métodos inicializan y devuelven la fecha del documento actual. set/getPalabras: Estos métodos inicializan y devuelven el número de palabras del documento actual. set/getAutor: Estos métodos inicializan y devuelven el autor del documento actual. set/getIdPublic: Estos métodos inicializan y devuelven el identificador público del documento actual. set/getIdPrivate: Estos métodos inicializan y devuelven el identificador privado del documento actual. set/getSeccion: Estos métodos inicializan y devuelven el periódico o sección del documento actual. InfoDocumentos: Constructor de la clase. setDocumentoNoticia: Permite cargar en memoria el documento que contiene el documento actual.

6.5.PAQUETE ONTOLOGÍA

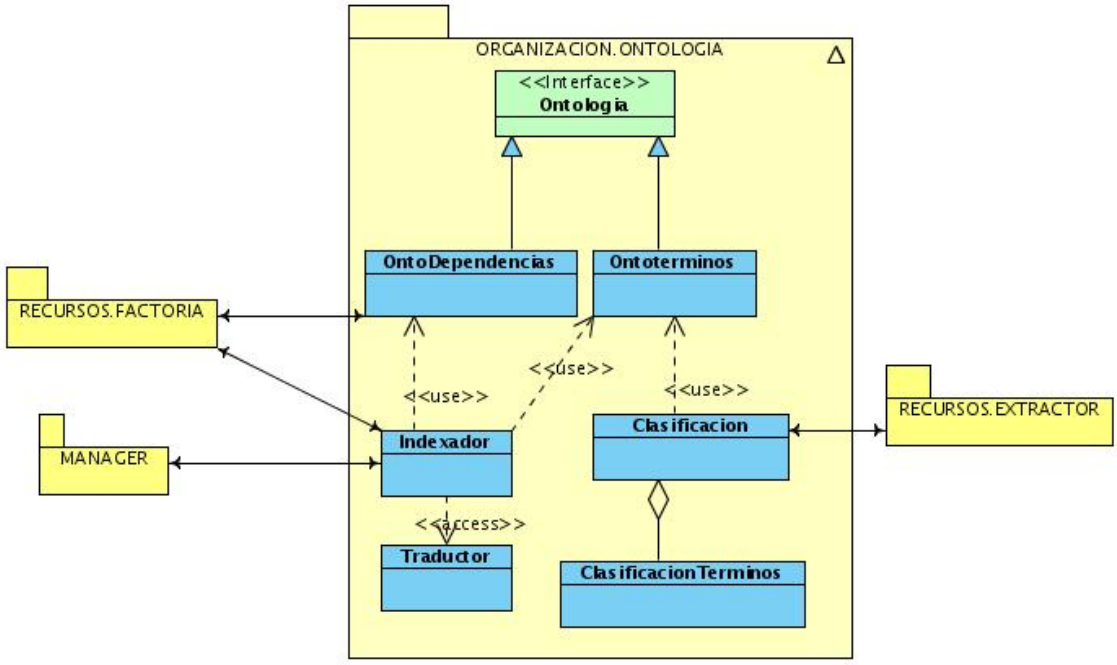


Figura 50 diseño paquete ontología

6.5.1.DICCIONARIO CLASES

ONTOLOGÍA	
	Descripción
	Esta interfaz encapsula el comportamiento que deben poseer las clases que implementen las ontologías en el sistema.
Atributos	
Métodos	setValores: Inicializa los valores de ciertos atributos. getSuperClase: Devuelve una determinada superclase de una clase dada. getBrothers: Devuelve las clases hermanas de una determinada clase. getIdividuals: Devuelve las instancias de una clase. createHierarchy: Genera la jerarquía de clases e individuos. asignaModeloLectura: Método que carga en memoria la ontología e inicializa un objeto que referencia y permite manejar la ontología. existeEnOntologia: Método que comprueba si dado un elemento, éste está presente como una clase en la ontología.

ONTOTERMINOS	
	Descripción
	Clase que maneja y gestiona las ontologías creadas a través de los términos.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. pathConceptos: Almacena la ruta donde se guarda la clasificación actual. pathOnto: Ruta para guardar la ontología generada. ONTnameSpace: Espacio de nombres asociado a la ontología actual. abreviatura: Abreviatura asociada al idioma actual. modelLectura: Objeto que permite referenciar y manipular la ontología cargada en memoria.
Métodos	Además de los métodos descritos en la interfaz Ontología, esta clase implementa las siguientes operaciones: getSubClase: Devuelve una determinada subclase de una clase dada. createRoot: Crea la cabecera de la ontología, con todas las declaraciones necesarias sobre los espacios de nombre. createClass: Método que se encarga de crear todas las clases a partir de las clases que se extraen del fichero que almacena la clasificación. createProperties: Método que crea las propiedades de los individuos de la ontología. createRestriction: Método que crea las restricciones de los individuos de la ontología. createIndividual: Método encargado de generar todas las instancias de las clases presentes en la ontología.

ONTODEPENDENCIAS	
	Descripción
	Clase que maneja y gestiona las ontologías creadas a través de las dependencias.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. pathConceptos: Almacena la ruta donde se guarda la clasificación actual. pathOnto: Ruta para guardar la ontología generada. Conceptos: Vector que almacena el conjunto de conceptos (o clases) que se recuperar del fichero de configuración de la clasificación por dependencias. Propiedades: Vector que almacena el conjunto de propiedades de tipo dato que se recuperar del fichero de configuración de la clasificación por dependencias. ONTnameSpace: Espacio de nombres asociado a la ontología actual. abreviatura: Abreviatura asociada al idioma actual. modelLectura: Objeto que permite referenciar y manipular la ontología cargada en memoria.
Métodos	Además de los métodos descritos en la interfaz Ontología, esta clase implementa las siguientes operaciones: getValoresFichero: Recupera los valores almacenados en el fichero de configuración de la clasificación por dependencias. createRoot: Crea la cabecera de la ontología, con todas las declaraciones necesarias sobre los espacios de nombre. createClass: Método que se encarga de crear todas las clases a partir de las clases que se extraen del fichero que almacena la clasificación. createObjectProperties: Método que crea las propiedades de tipo objeto de los individuos de la ontología. createDataProperties: Método que crea las propiedades de tipo dato de los individuos de la ontología. createIndividual: Método encargado de generar todas las instancias de las clases presentes en la ontología. createRestriction: Método que crea las restricciones de los individuos de la ontología. getObjectProperties: Método que devuelve las propiedades de tipo objeto de una clase. getClassDep: Método que devuelve el conjunto de clases en la ontología. getClasesDelIndividuo: Método que devuelve las clases a las que pertenece un individuo.

TRADUCTOR	
	Descripción
	Esta clase efectúa una llamada al traductor de Google para recuperar la traducción de la consulta introducida por el usuario.
Atributos	origen: idioma de origen de la traducción, en formato de abreviatura válida para Google. destino: idioma de destino de la traducción, en formato de abreviatura válida para Google.
Métodos	Traductor: Constructor que inicializa las variables origen y destino. TecnicaTraduccion: Método que efectúa la llamada al traductor y recoge el resultado.

INDEXADOR	
	Descripción
	Clase que maneja el índice, creándolo, actualizando. Se encarga, asimismo, de recuperar los documentos relevantes a la consulta del usuario.
Atributos	pathOnto: Ruta donde se ubica la ontología. pathIndexador: Ruta donde se ubica el índice. tema: Almacena el tema actual. idioma: Almacena el idioma actual. abreviatura: Guarda la abreviatura del idioma actual. fichStopW: Guarda la ruta donde se ubica el fichero de palabras de parada. objectDoc: Representa el documento de configuración en memoria.
Métodos	setValores: Inicializa los atributos. crearIndice: Crea el índice asociado a un número de documentos iniciales. existeStopW: Comprueba si para el idioma actual existe el fichero de palabras de parada. crearDocumento: A partir del documento XML extrae las partes pertinentes a introducir en el índice. A continuación indexa el documento con el índice ya creado. recuperarDocumento: A través de una consulta al índice se recuperan los documentos que cumplen con la consulta del usuario. indexarUnDocumento: Método que permite indexar un documento introducido en el sistema. recuperarDocPorProx: Método que permite recuperar los documentos relacionados con una consulta del usuario, en la que están implicadas más de una palabra. recuperarDocPorTerm: Método que permite recuperar los documentos relacionados con una consulta del usuario, en la que sólo está implicada una palabra.

CLASIFICACIÓN	
	Descripción
	Clase que lleva a cabo la clasificación de los términos extraídos.
Atributos	tema: Almacena el tema actual. idioma: Almacena el idioma actual. extractores: Vector con los extractores implicados. mapaContextos: Tabla que almacena los contextos de los términos. rdoExtraccion: Ruta donde ubicar el resultado de la extracción. rdoClasificación: Ruta donde almacenar la clasificación. bdElegida: Almacena el nombre de la base de datos actual, necesaria para la recuperación de las categorías. mapaTerminos: Tabla que almacena los términos junto a las categorías de cada una de las palabras que los conforman. vTerminos: Vector que guarda los términos extraídos. vPalabras: Vector que guarda las palabras presentes en los términos.
Métodos	Clasificacion: Constructor de clase. calculaContextosCompartidos: Este método analiza, dados dos términos, el número de contextos que éstos comparten. lanzaClasificacion: Método llamado fuera de la clase para lanzar todo el proceso de

	clasificación. llamarFicheroRdo: Método que permite cargar en memoria el fichero con los resultados de la extracción y así poder generar la clasificación. creaVectores: Este método genera los vectores vTerminos y vPalabras. creaHash: Este método genera las tablas mapaContextos y mapaTerminos. calculaNuevosContextos: Este método se llama una vez localizados aquellos dos términos más similares en una iteración. Esos dos términos se fusionan, uniendo también sus contextos de aparición. creaCatIdioma: Permite recuperar la categoría de cada palabra implicada en los términos extraídos. calculaCoeficienteDice: Este método calcula el coeficiente DICE asociado a dos términos.

CLASIFICACIÓNTERMINOS	
	Descripción
	Clase de apoyo a clasificación. Permite manejar el fichero que almacena los resultados de la clasificación.
Atributos	almacen: Ruta donde se ubica el fichero de clasificación. documento: Representa el documento en memoria.
Métodos	ClasificacionTerminos: Constructor de la clase. setFicheroClasi: Método que carga en memoria el fichero de clasificación. getExtractoresEmpleados: Método que devuelve los extractores implicados en un proceso de clasificación. creaFicheroClaseXML: Método que genera el fichero que almacena los términos clasificados.

6.6.PAQUETE TEXT

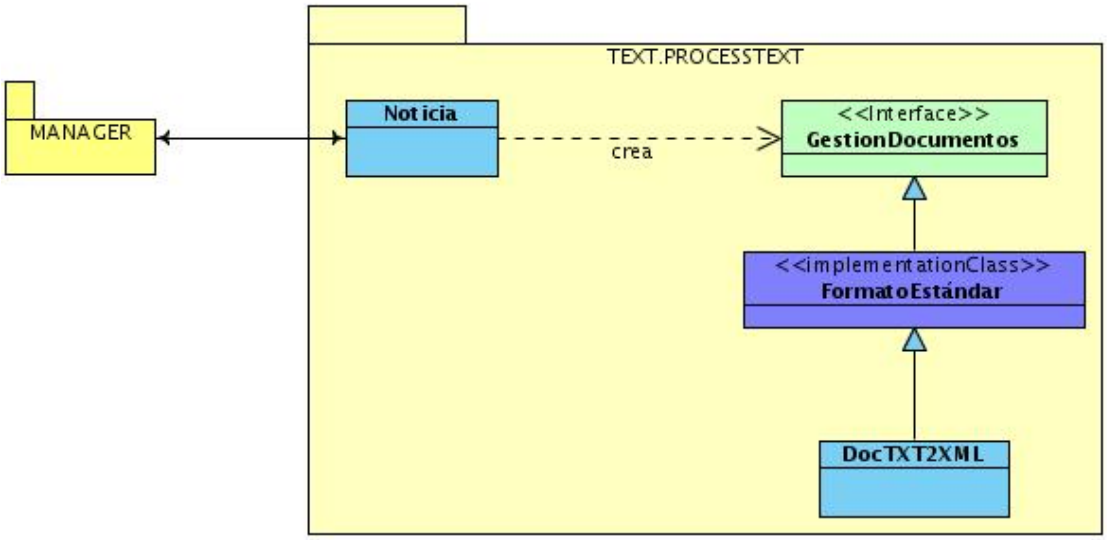


Figura 51 diseño paquete text

6.6.1.DICCIONARIO CLASES

NOTICIA	
	Descripción
	Clase que permite el acceso al manejo de los documentos introducidos por el usuario.
Atributos	tipoDOC: Este atributo almacena el tipo de extensión del documento a añadir al sistema, para de ese modo poder llamar a la clase adecuada.
Métodos	Noticias: Es el constructor de la clase. creaNoticia: Este método llama de forma dinámica a la clase que gestiona la extracción de la información pertinente de los nuevos textos introducidos en el sistema. Puesto que se contempla la posibilidad de que los textos puedan estar almacenados con diversas extensiones.

GESTIONDOCUMENTOS	
	Descripción
	Interfaz que define el comportamiento de aquellas clases que tengan que recuperar el contenido de las fuentes textuales introducidas por el usuario.
Atributos	
Métodos	recorreDocBase: Recorre el documento base proporcionado por el usuario para recoger las distintas secciones y poder generar el XML correspondiente. makeDoc2XML: Una vez inicializados las distintas secciones, este método guarda cada una de ellas en una estructura XML.

FORMATOESTANDAR	
	Descripción
	Clase abstracta para el manejo de los ficheros proporcionados al sistema por el usuario.
Atributos	titulo: Titulo del documento actual. texto: Texto del documento actual. fecha: Fecha indicada en el documento actual. palabras: Número de palabras en el documento actual. autor: Autor del documento actual. seccion: Sección o periódico que publica la noticia recogida en el documento actual. idPublic: Identificador público que viene dado por un valor único. idPrivate: Identificador privado que viene dado por un valor único.
Métodos	FormatoEstandar: Es el constructor de la clase. Se encarga de recuperar del fichero de marcas el conjunto de etiquetas que delimitan cada sección del texto. Los métodos heredados de la interfaz GestionDocumentos los implementa la clase siguiente.

DOCTXT2XML	
	Descripción
	Esta clase implementa el caso concreto de extraer el contenido de un documento introducido en el sistema con extensión ***txt***.
Atributos	
Métodos	Además de implementar los métodos descritos en la interfaz GestionDocumento, también incluye los siguientes: bienFormado: Comprueba que el documento base proporcionado por el usuario se corresponda con la estructura esperada. existeDoc: Comprueba si un determinado documento ya ha sido incorporado al tema e idioma actual DocTXT2XML: Constructor de la clase. setVbles: Método que asigna valores a cada una de las variables que representan las secciones de los documentos introducidos en el sistema

6.7.PAQUETE VISTAS

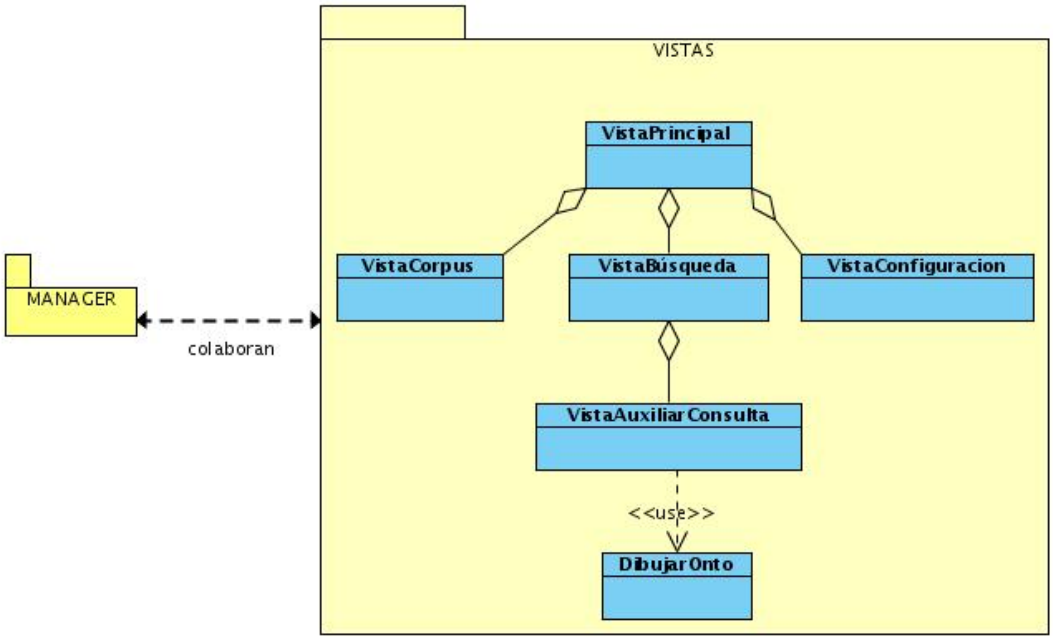


Figura 52 diseño paquete vistas

6.7.1.DICCIONARIO DE CLASES

VISTAPRINCIPAL	
	Descripción
	Clase que contiene la vista principal de la interfaz gráfica. Esto es, los elementos que se muestran al arrancar la aplicación, así como todos aquellos ítems que se mantienen en todas las vistas.
Atributos	documentoXMLConf: Objeto de tipo DocXMLConfiguración para el acceso a los métodos necesarios para la manipulación del fichero de configuración. ficheroConfiguracion: Ruta donde se encuentra el fichero de configuración. PATH: Ruta donde se guarda la aplicación. docConfig: Objeto que mantiene una referencia al documento de configuración que está en memoria.
Métodos	VistaPrincipal: Constructor de la clase. initComponentes: Inicializa la interfaz de entrada al sistema.

VISTABÚSQUEDA	
	Descripción
	Clase que permite la realización de las operaciones de búsqueda.
Atributos	temas: Vector con los temas presentes en la aplicación. temaVSidioma: Tabla que asocia los temas con sus idiomas. idiomaVSabr: Tabla que asocia cada idioma con su abreviatura. docConfig: Objeto que mantiene una referencia al documento de configuración que está en memoria. PATH: Ruta donde se guarda la aplicación.
Métodos	VistaBusqueda: Constructor de la clase. crearFondo: Método que crea el fondo de la interfaz. tabTerminos: Método que genera y carga la pestaña correspondiente a la parte de búsqueda por términos. tabDependencias: Método que genera y carga la pestaña correspondiente a la parte de búsqueda por dependencias.

VISTACORPUS	
	Descripción
	Clase que permite la realización de operaciones sobre el corpus. Es decir, añadir nuevos documentos, clasificar a partir de los términos y clasificar a partir de las dependencias.
Atributos	temas: Vector con los temas presentes en la aplicación. temaVSidioma: Tabla que asocia los temas con sus idiomas. idiomaVSabr: Tabla que asocia cada idioma con su abreviatura. docConfig: Objeto que mantiene una referencia al documento de configuración que está en memoria. PATH: Ruta donde se guarda la aplicación. extractores:Vector con los temas presentes en la aplicación. idiomaVSparser: Tabla que asocia los idiomas con sus analizadores sintácticos. parsers:Vector con los analizadores sintácticos presentes en la aplicación.
Métodos	VistaCorpus: Constructor de la clase. reinicializarValores: Método que permite actualizar el fichero de configuración. crearFondo: Método que crea el fondo de la interfaz. tabGeneracion: Método que genera y carga la pestaña correspondiente a la parte de generación del corpus. tabClasificacion: Método que genera y carga la pestaña correspondiente a la parte de clasificación del corpus.

VISTACONFIGURACIÓN	
	Descripción
	Clase que permite el manejo del fichero de configuración a través del conjunto de ventanas.
Atributos	temas: Vector con los temas presentes en la aplicación. idiomas: Vector con los idiomas presentes en la aplicación. extractores: Vector con los extractores presentes en la aplicación. temaVSidioma: Tabla que asocia los temas con sus idiomas. idiomaVSabr: Tabla que asocia cada idioma con su abreviatura. docConfig: Objeto que mantiene una referencia al documento de configuración que está en memoria. parsers: Vector con los analizadores sintácticos presentes en la aplicación. docXMLConfiguracion: Objeto de tipo DocXMLConfiguración para el acceso a los métodos necesarios para la manipulación del fichero de configuración
Métodos	VistaConfiguración: Constructor de la clase. reinicializarValores: Método que permite actualizar el fichero de configuración. crearFondo: Método que crea el fondo de la interfaz. tabDeclaraCorpus: Método que genera y carga la pestaña correspondiente a la parte de modificación del fichero de configuración respecto a los temas e idiomas. tabHerramientas: Método que genera y carga la pestaña correspondiente a la parte de modificación del fichero de configuración respecto a los extractores y analizadores sintácticos.

VISTAAUXILIARCONSULTA	
	Descripción
	Clase que implementa la vista que muestra el resultado de una consulta planteada por el usuario.
Atributos	
Métodos	abrirVentana: Método que crea la ventana en la que se muestran los documentos recuperados para la consulta del usuario y carga un gráfico con la parte correspondiente de la ontología. crearVentanaMuestraDoc: Método que crea la ventana en la que se muestra el contenido de los documentos extraídos relevantes a la consulta del usuario.

DIBUJARONTOLOGIA	
	Descripción
	Clase que permite dibujar en un gráfico la parte de la ontología que se corresponde con la consulta del usuario.
Atributos	tema: En este atributo se almacena el tema actual. grupo: En este atributo se guarda el valor del grupo al que pertenece un determinado término (Esto se utiliza al generar el diagrama para la ontología por términos). hijos_clase: Vector que almacena las clases de la ontología relevantes en la consulta planteada por el usuario. hijos_individuos: Vector que almacena las instancias de clase de la ontología que son relevantes en la consulta planteada por el usuario.
Métodos	setValores: Inicializa los valores de los atributos de la clase. dibujarTerminos: Método que genera un gráfico con la relación de los términos extraídos para la consulta introducida por el usuario. dibujarDependencias: Método que genera un gráfico con la relación de las dependencias extraídos para la consulta introducida por el usuario.

7. DIAGRAMAS DE SECUENCIA DEL DISEÑO

En esta sección se presentan los diagramas de secuencia detallados que se corresponden a los casos de uso y diagramas de secuencia desarrollados durante el análisis.

7.1.CASO DE USO GESTIÓN CONFIGURACIÓN

7.1.1.GESTIÓN TEMAS (ESCENARIO 1): AÑADIR TEMA

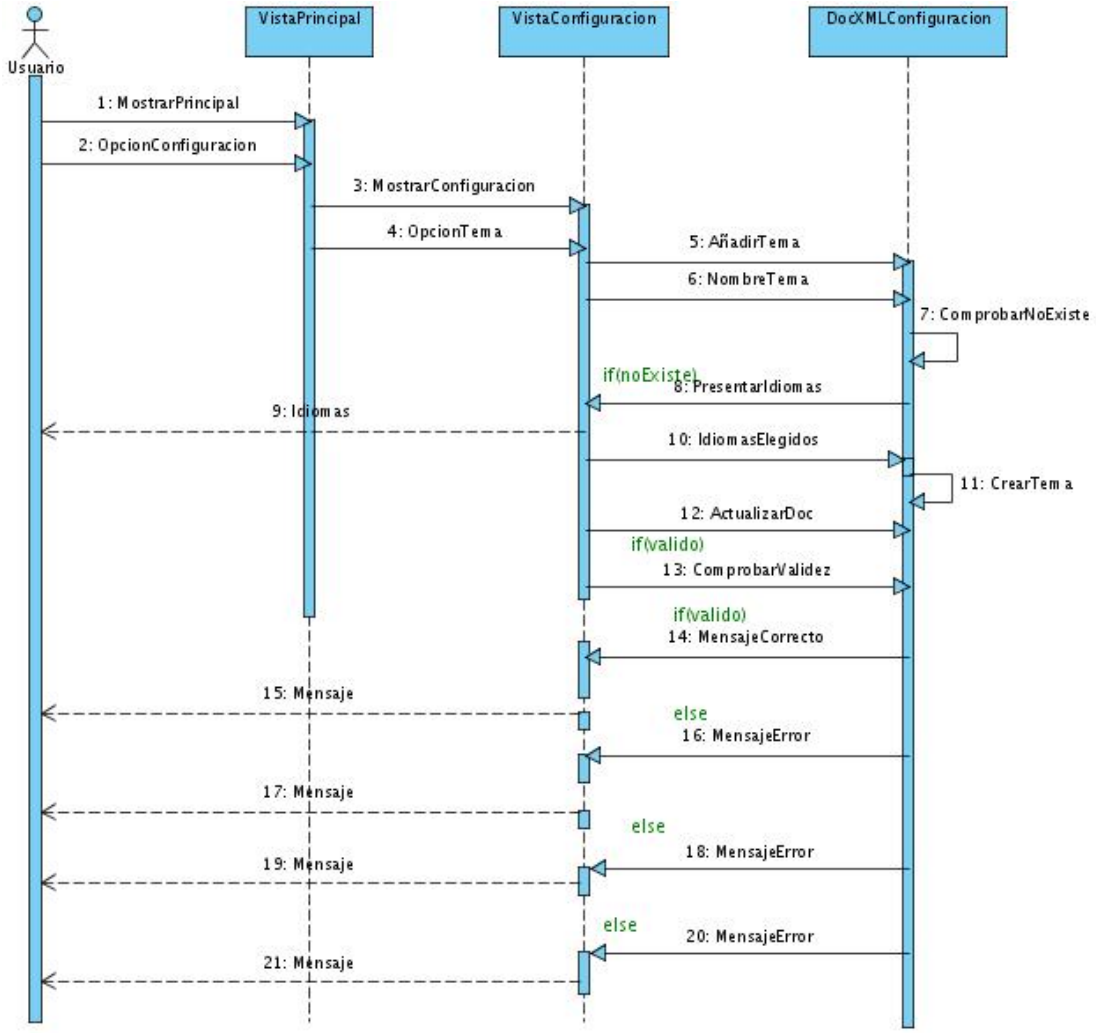


Figura 53 diseño diag.sec.añadir tema

7.1.2.GESTIÓN TEMAS (ESCENARIO 2): BORRAR TEMA

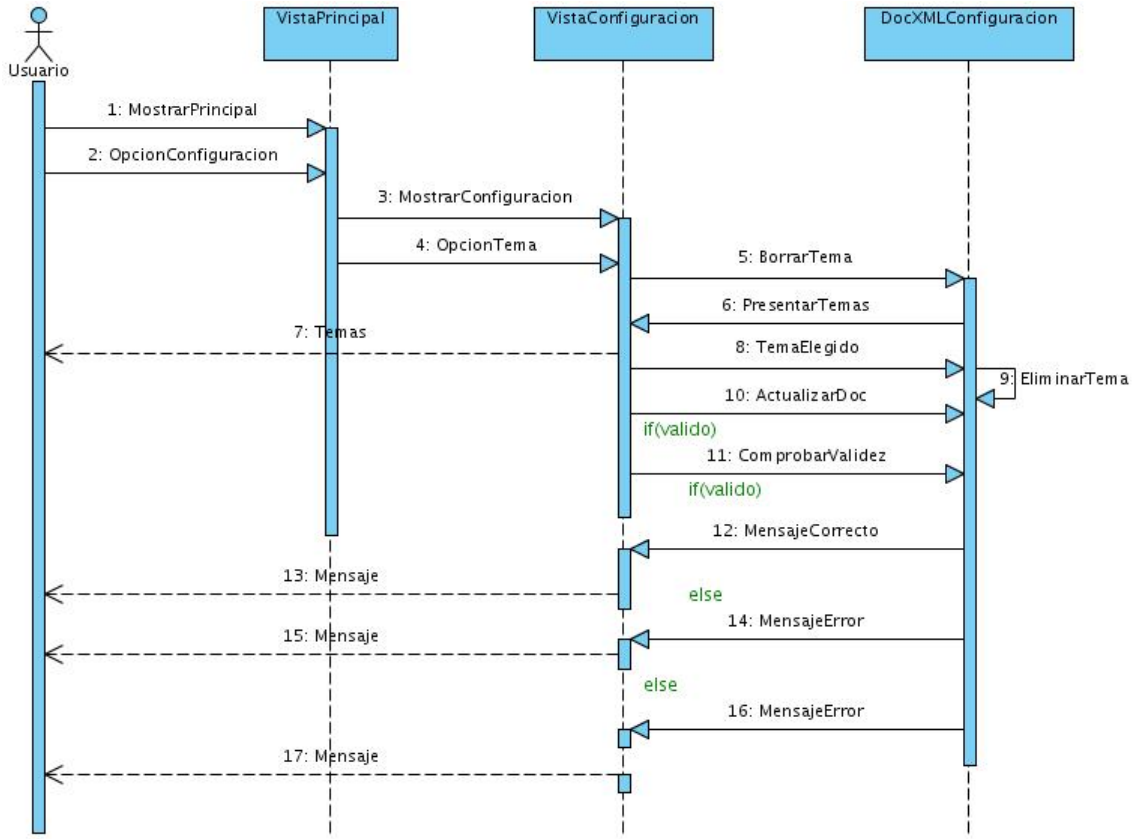


Figura 54 diseño diag.sec.borrar tema

7.1.3.GESTIÓN IDIOMAS (ESCENARIO 1): AÑADIR IDIOMA

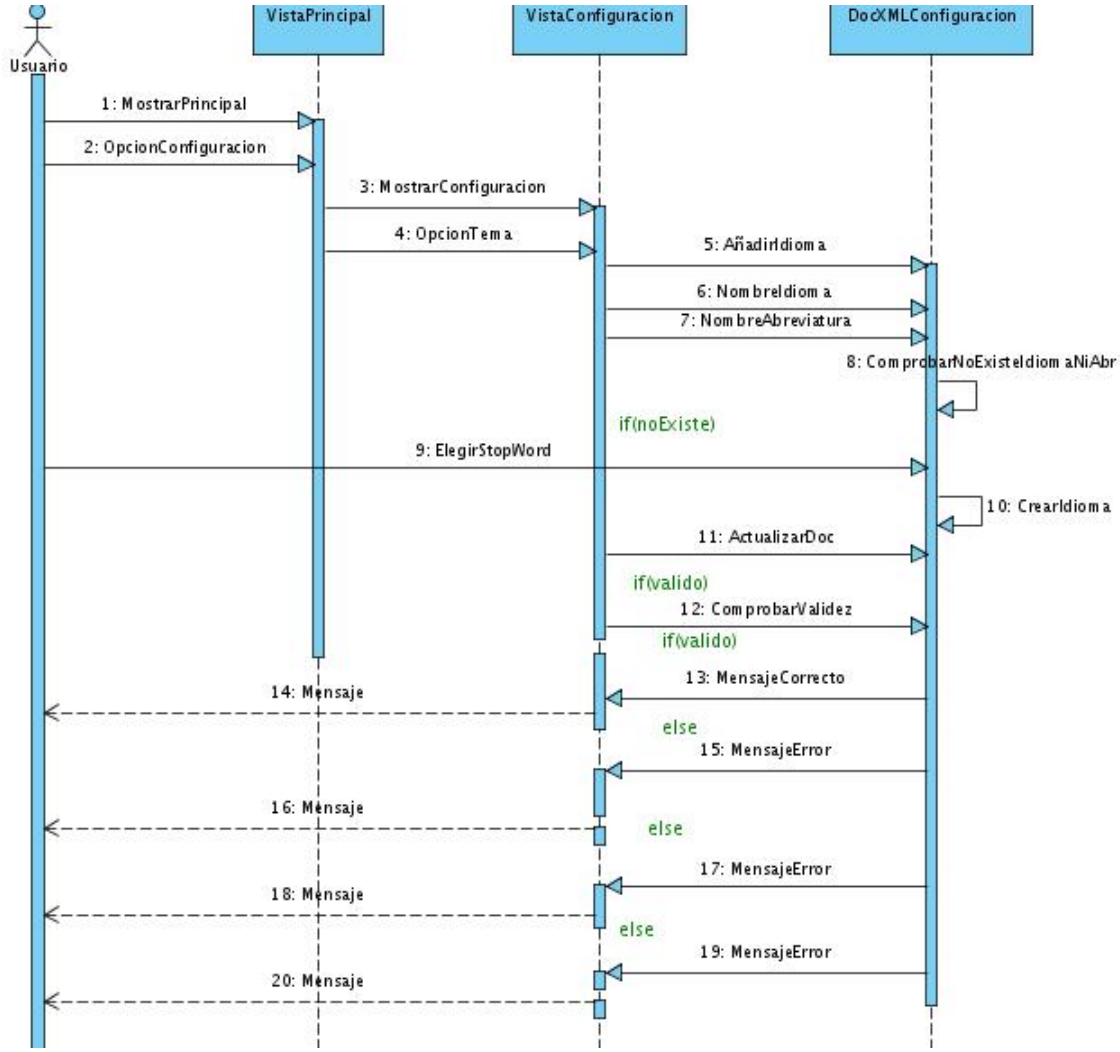


Figura 55 diseño diag.sec.añadir idioma

7.1.4.GESTIÓN IDIOMAS(ESCENARIO 2): BORRAR IDIOMA

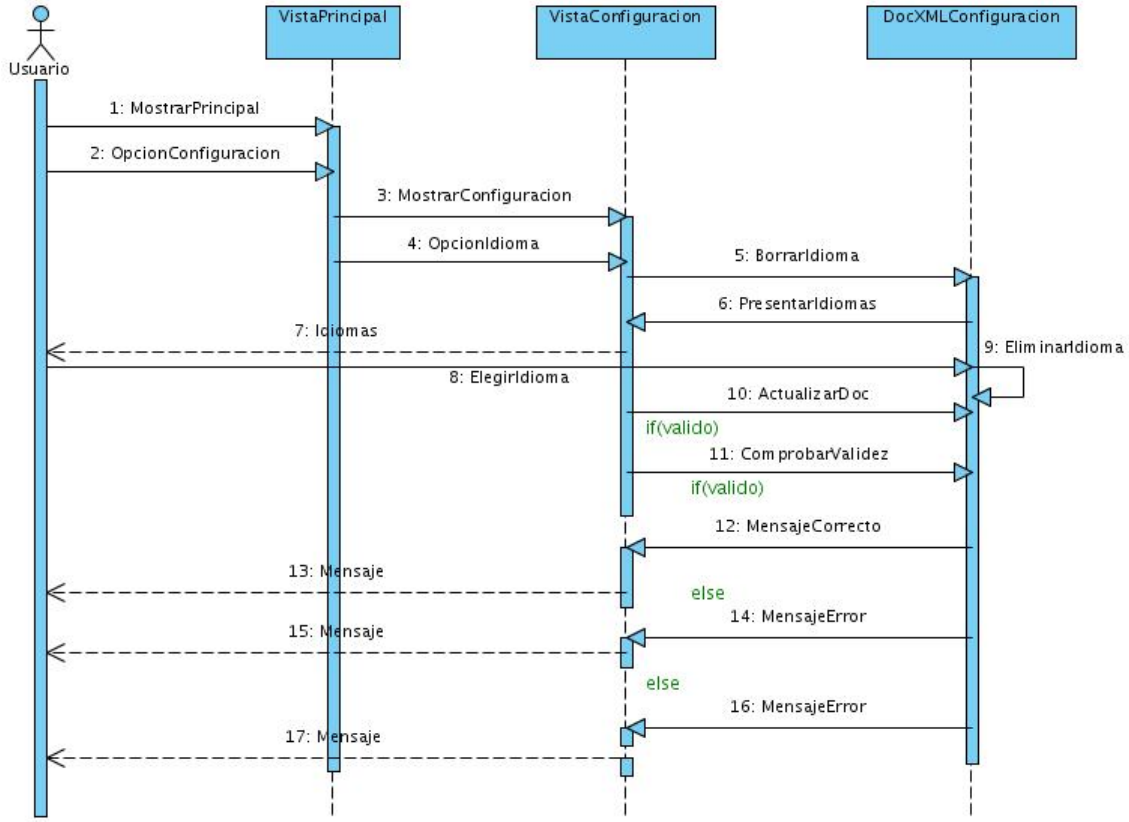


Figura 56 diseño diag.sec.borrar idioma

7.1.5.GESTIÓN EXTRACTORES (ESCENARIO 1): AÑADIR EXTRACTOR

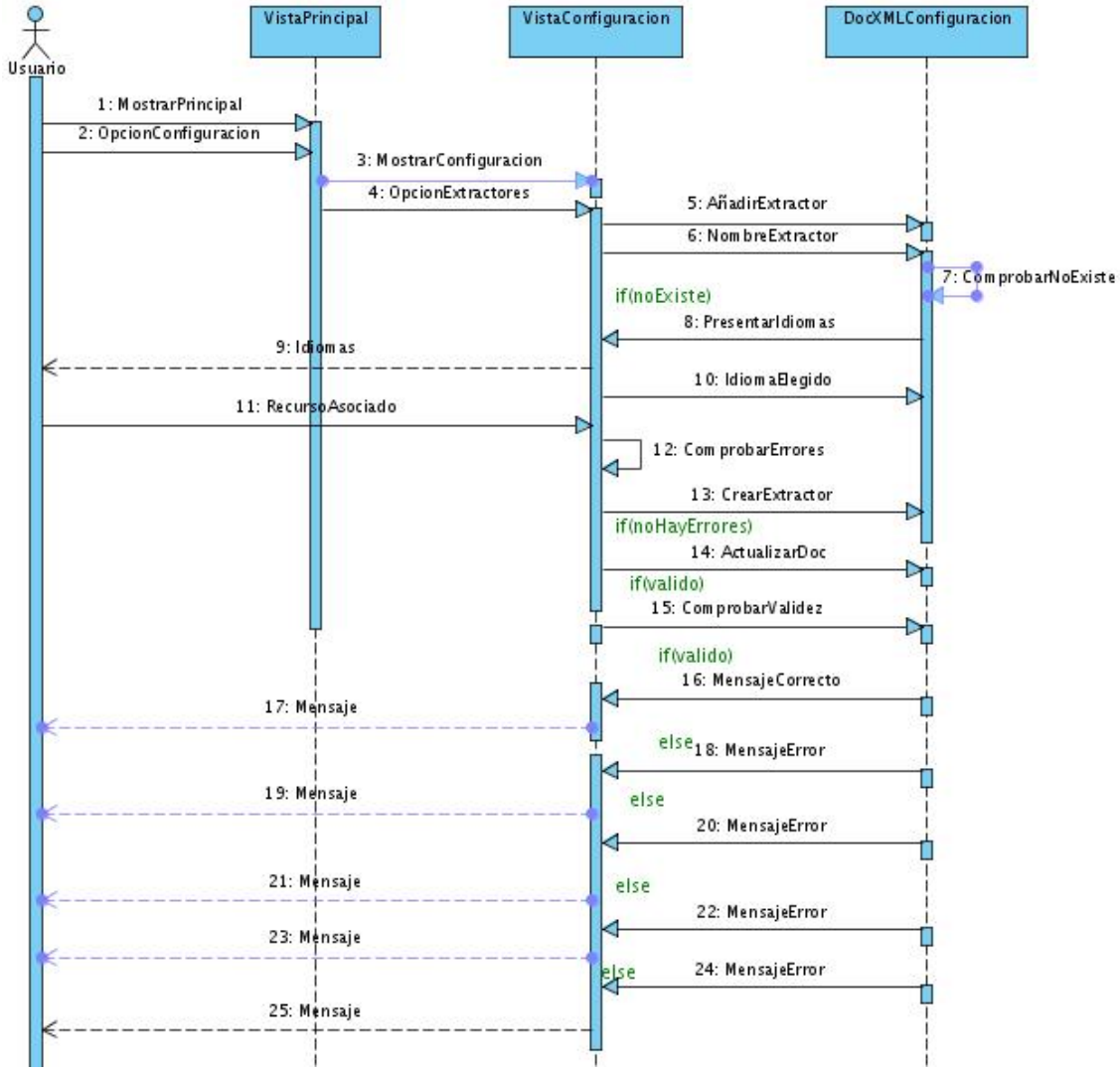


Figura 57 diseño diag.sec.añadir extractor

7.1.6.GESTIÓN EXTRACTORES (ESCENARIO 2): BORRAR EXTRACTOR

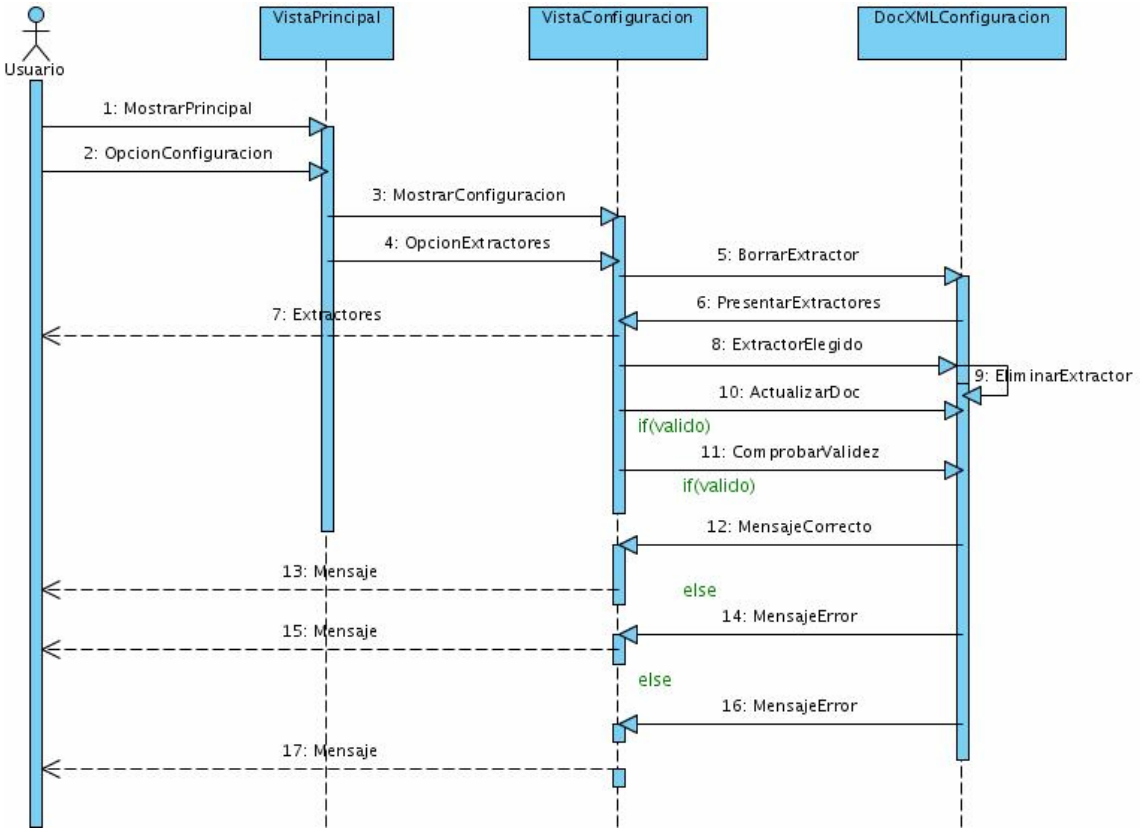


Figura 58 diseño diag.sec.borrar extractor

7.1.7.GESTIÓN ANALIZADOR (ESCENARIO 1): AÑADIR ANALIZADOR

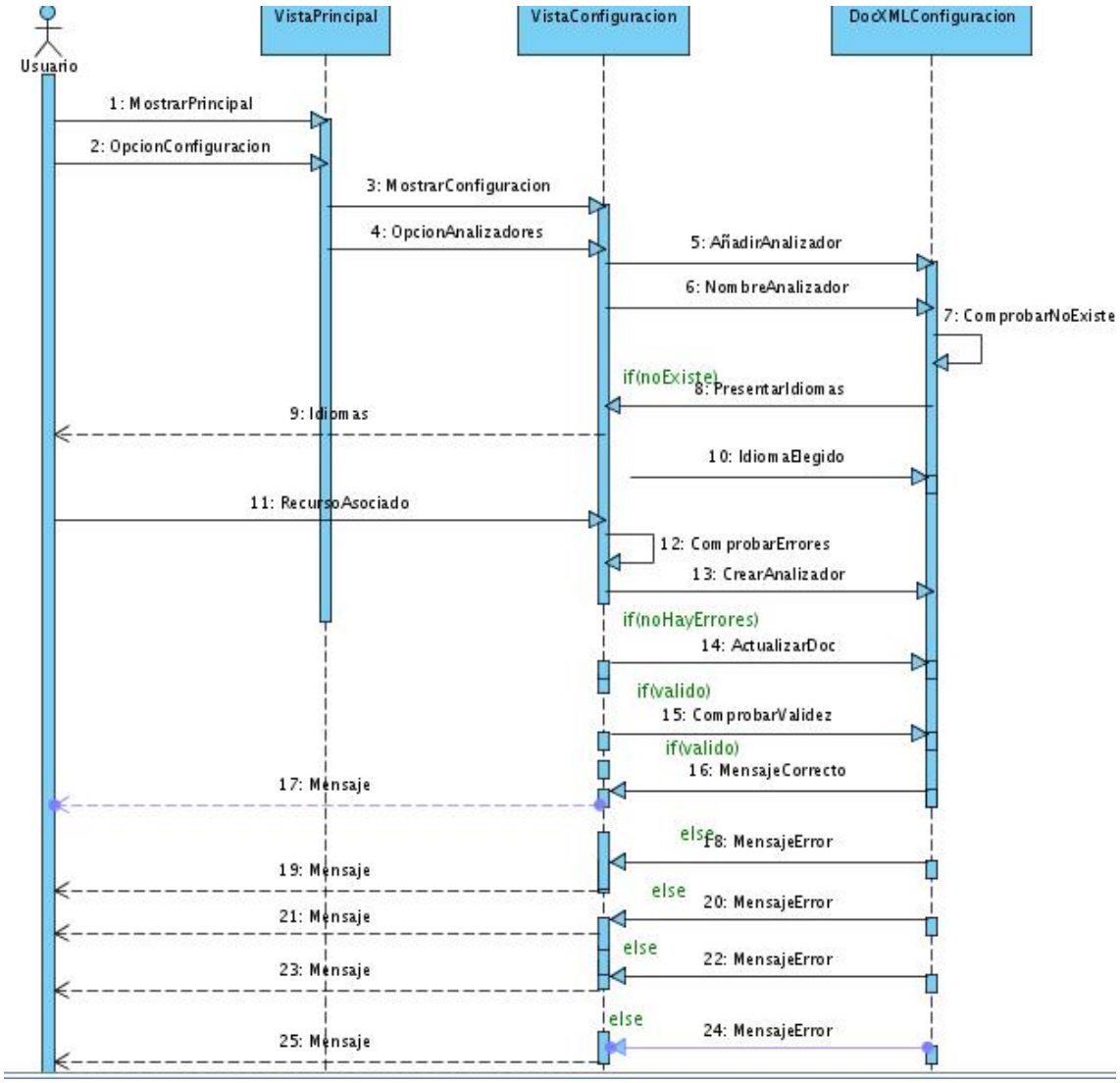


Figura 59 diseño diag.sec.añadir analizador

7.1.8.GESTIÓN ANALIZADOR (ESCENARIO 2): BORRAR ANALIZADOR

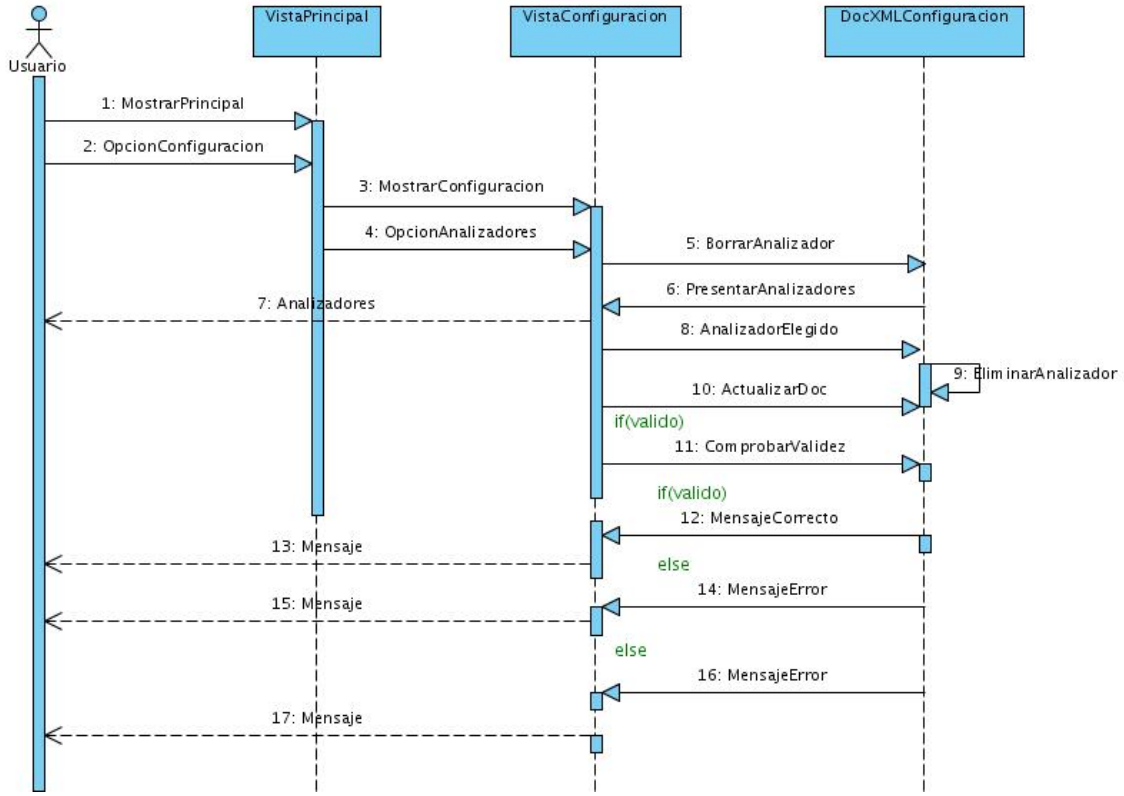


Figura 60 diseño diag.sec.borrar analizador

7.2.CASO DE USO MANIPULACIÓN CORPUS

7.2.1.CREAR ONTOLOGÍA POR TÉRMINOS

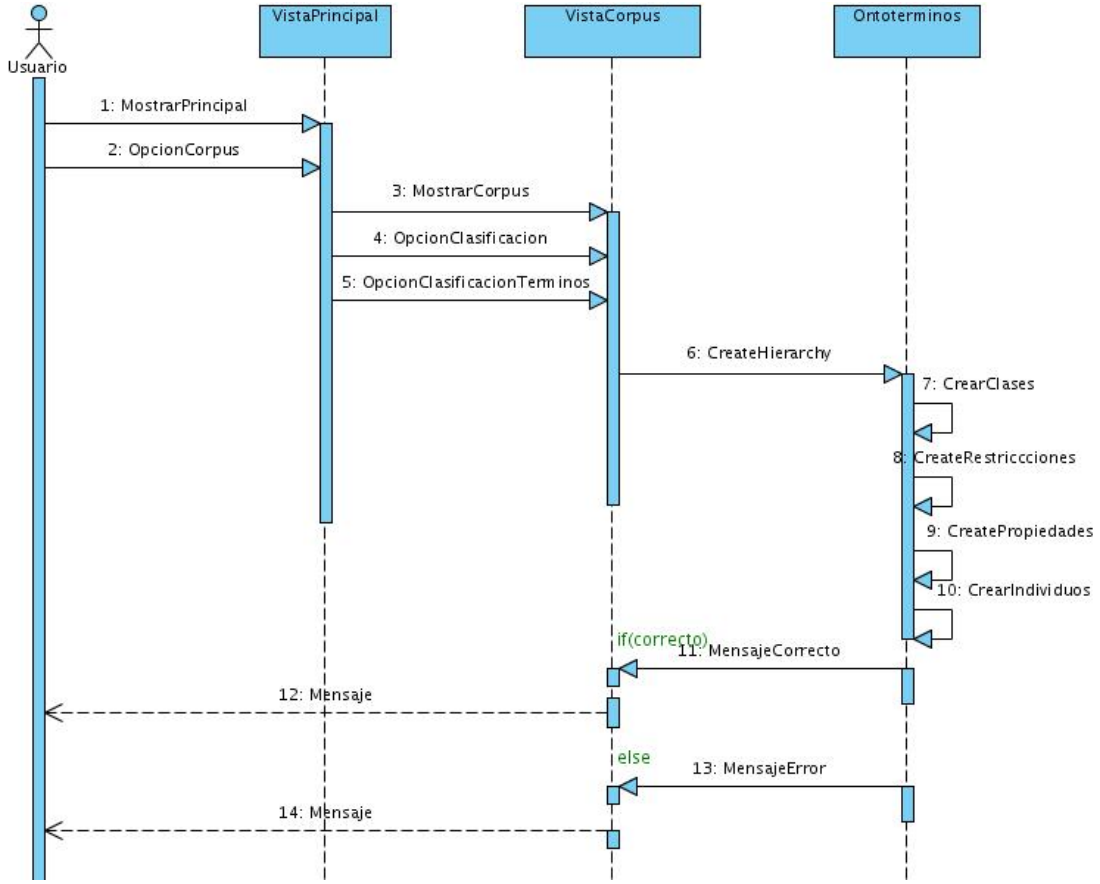


Figura 61 diseño diag.sec.crear ontología términos

7.2.2.CREAR ONTOLOGÍA POR DEPENDENCIAS

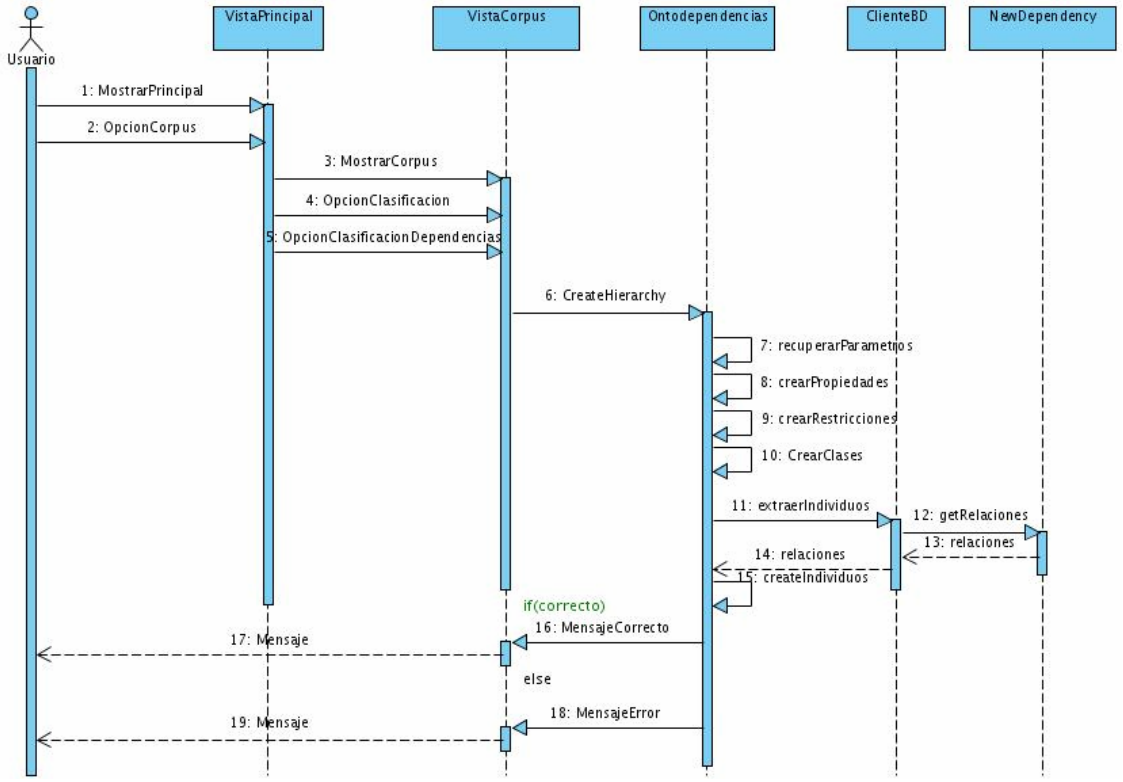


Figura 62 diseño diag.sec.crear ontología dependencias

7.2.3.CASO DE USO: EXTRACCIÓN

(A) RECUPERAR EXTRACTOR

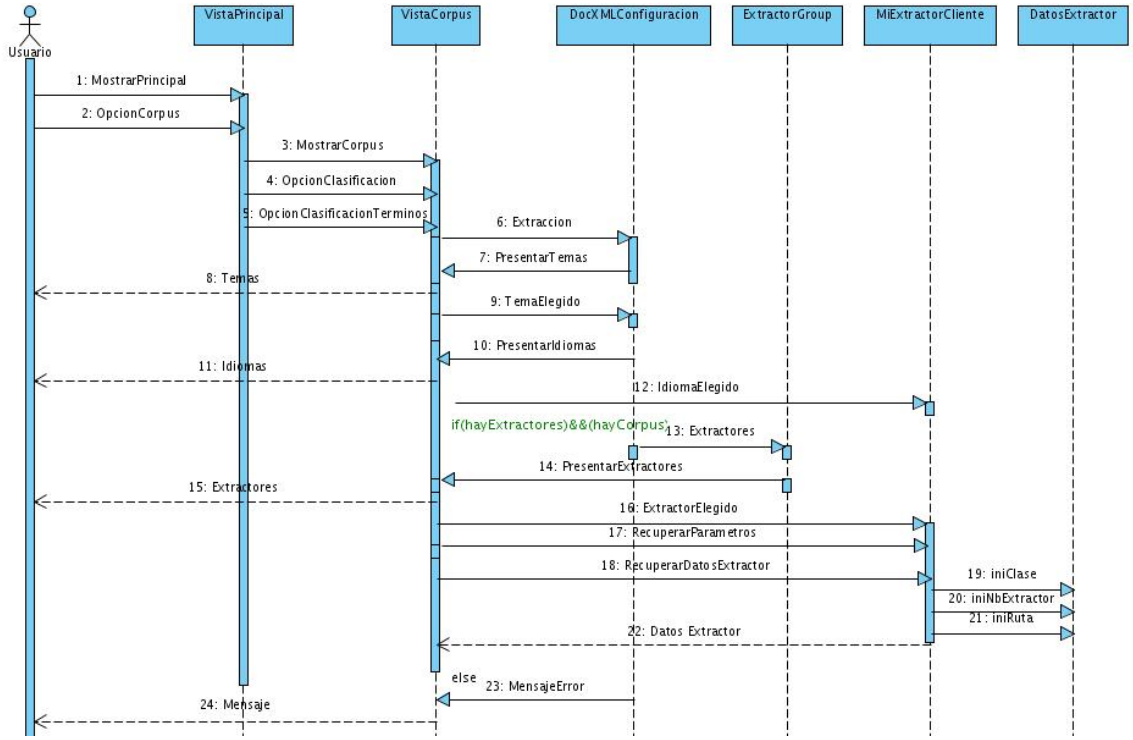


Figura 63 diseño diag.sec.recuperar extractor

(B) LANZAR EXTRACCIÓN

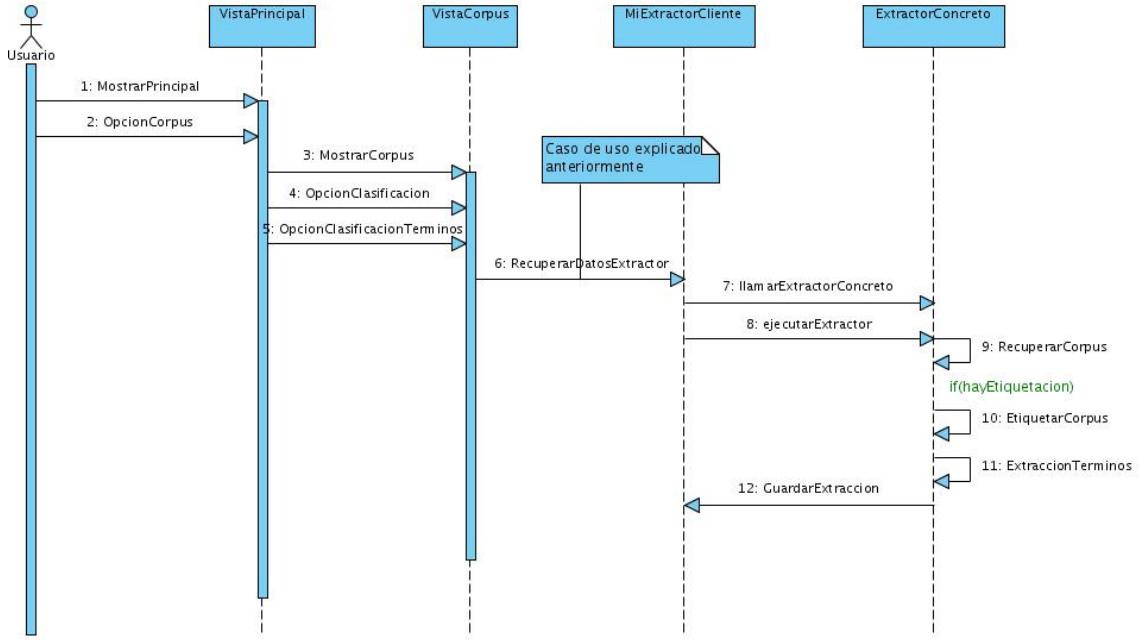


Figura 64 diseño diag.sec.lanzar extracción

(C) LANZAR ETIQUETACIÓN

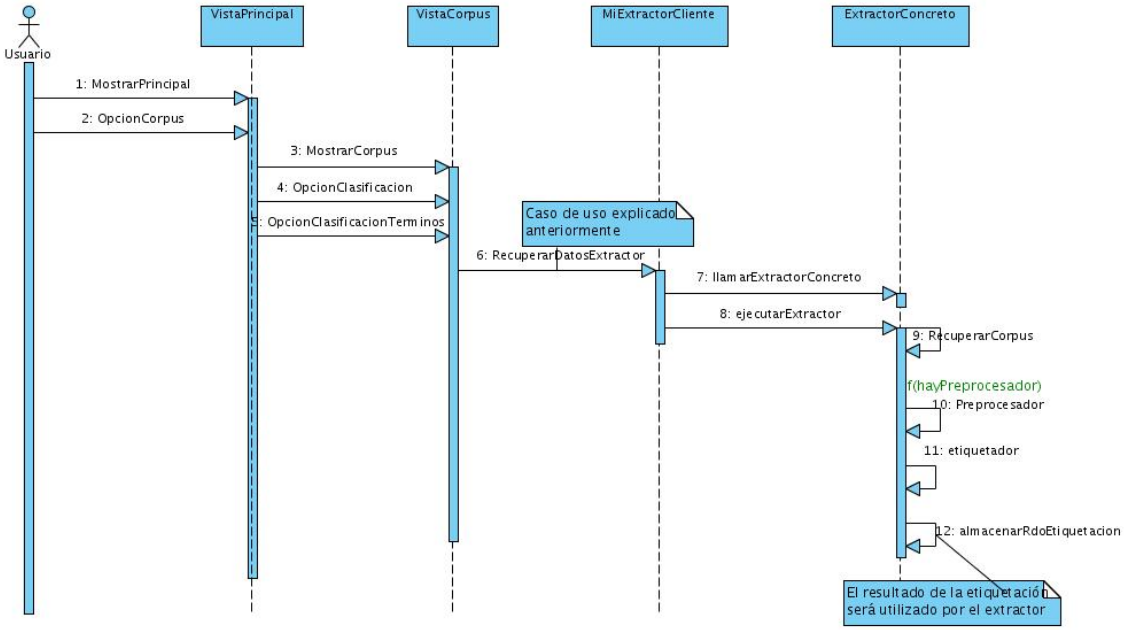


Figura 65 diseño diag.sec.lanzar etiquetación

(D) GUARDAR EXTRACCIÓN

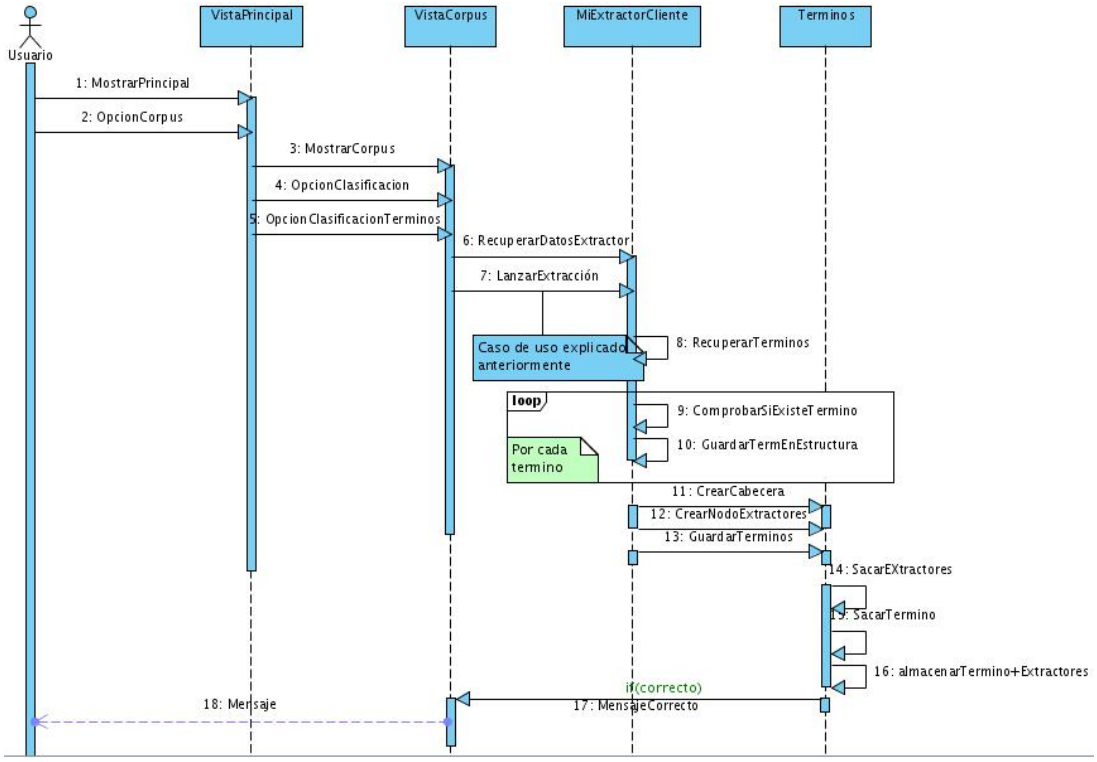


Figura 66 diseño diag.sec.guardar extracción

7.2.4.CASO DE USO: CLASIFICACIÓN POR TÉRMINOS

(A) RECUPERAR RESULTADOS EXTRACCIÓN

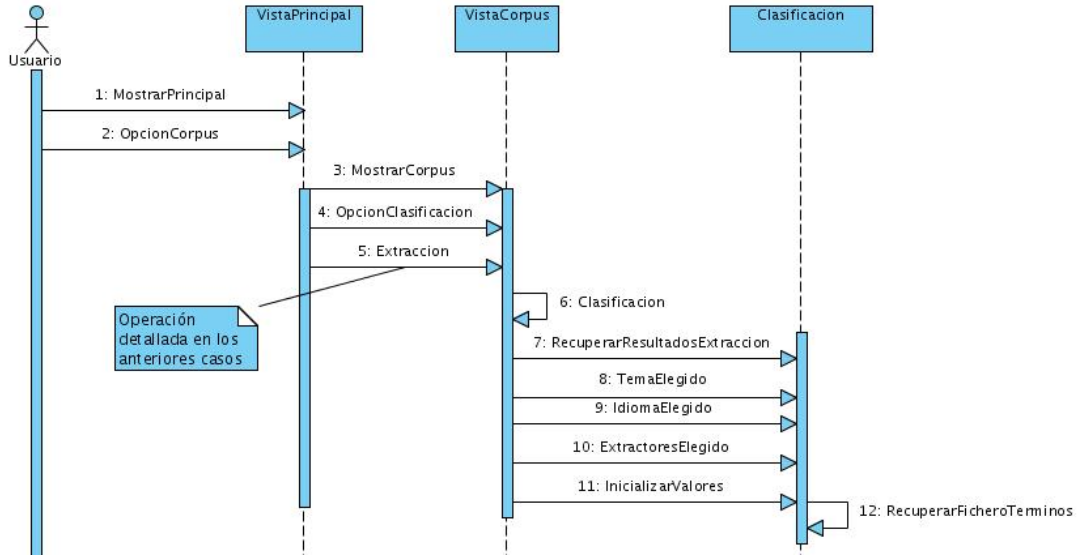


Figura 67 diseño diag.sec.recuperar resultados extracción

(B) LANZAR CLASIFICACIÓN

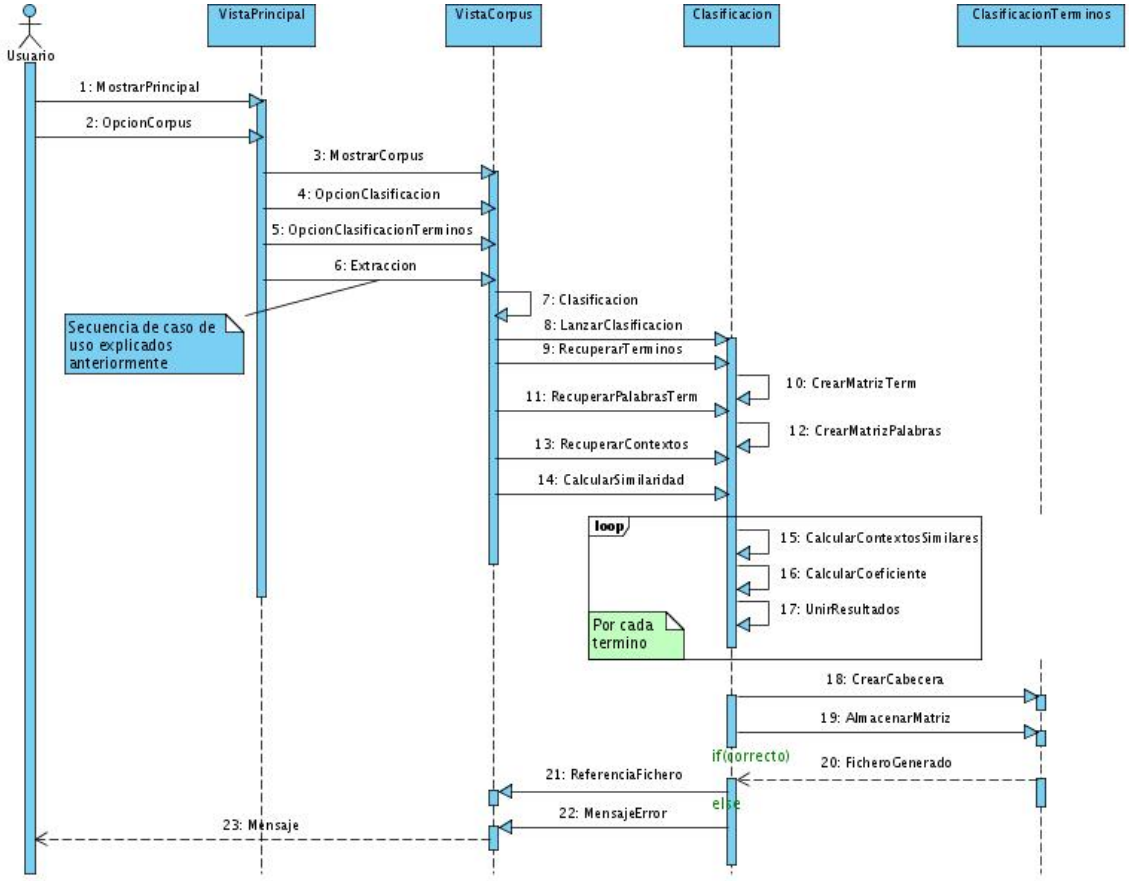


Figura 68 diseño diag.sec.lanzar clasificación

(C) RECUPERAR CONTEXTOS

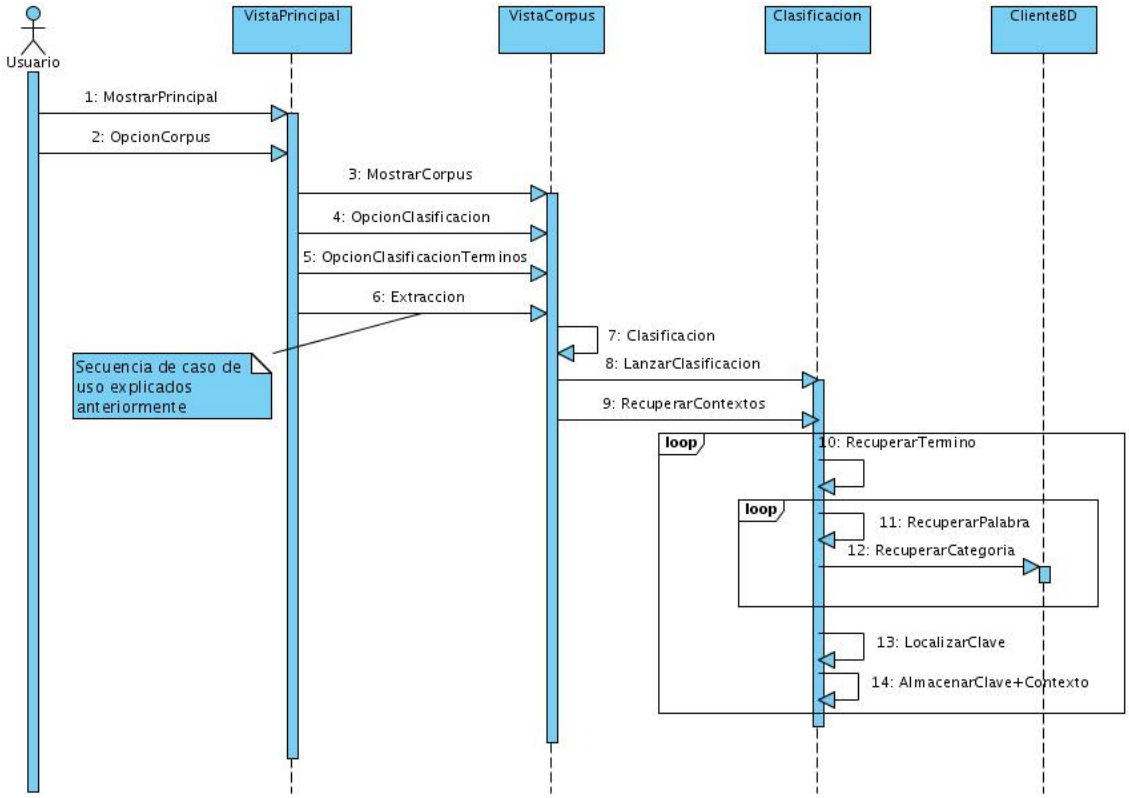


Figura 69 diseño diag.sec.recuperar contextos

7.2.5.CASO DE USO: ANÁLISIS SINTÁCTICO

(A) ANÁLISIS DE VARIOS DOCUMENTOS

Este caso de uso, al igual que en el análisis, no tiene diagrama asociado, pues analizar sintácticamente un conjunto de documentos equivale a analizar cada uno de ellos por separado. Este hecho se presenta a continuación en los casos de uso sucesivos.

(B) ANÁLISIS UN DOCUMENTO

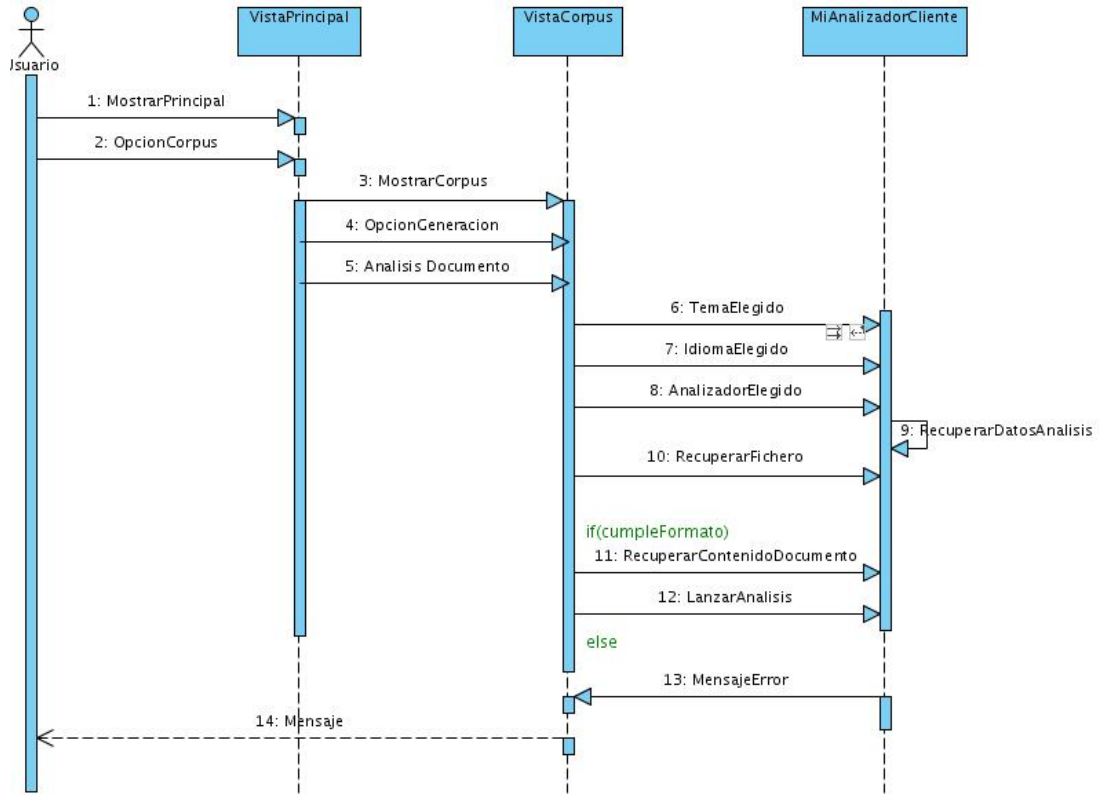


Figura 70 diseño diag.sec.análisis documento

(C) RECUPERAR ANALIZADOR

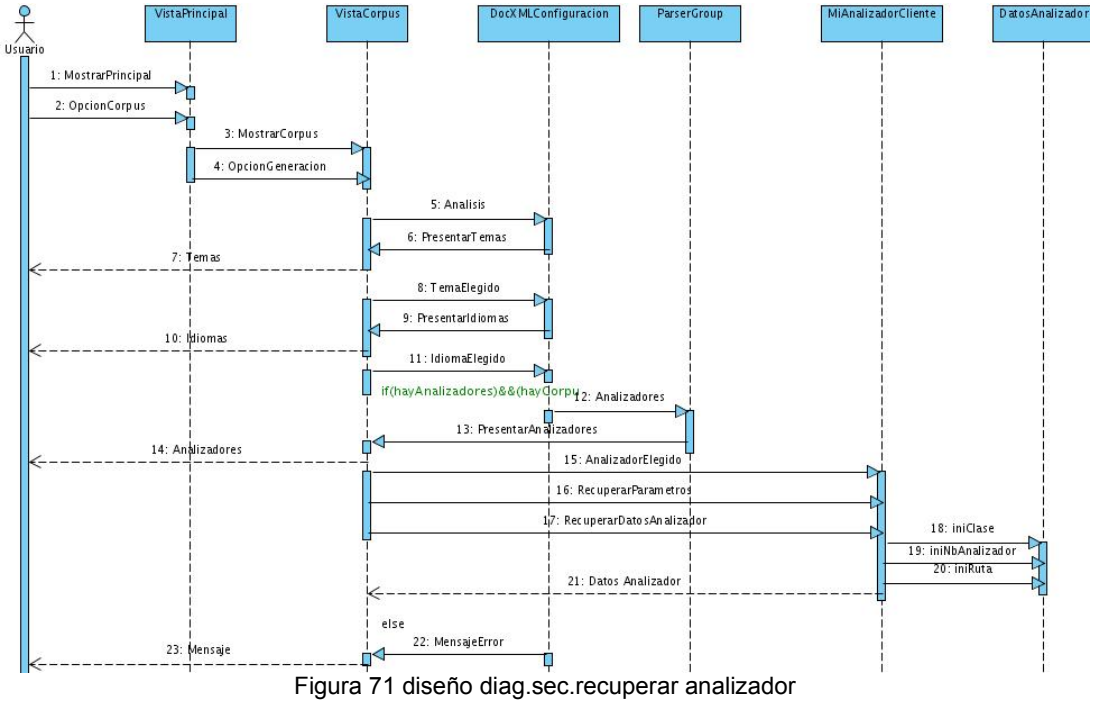


Figura 71 diseño diag.sec.recuperar analizador

(D) EXTRAER CONTENIDO DOCUMENTO

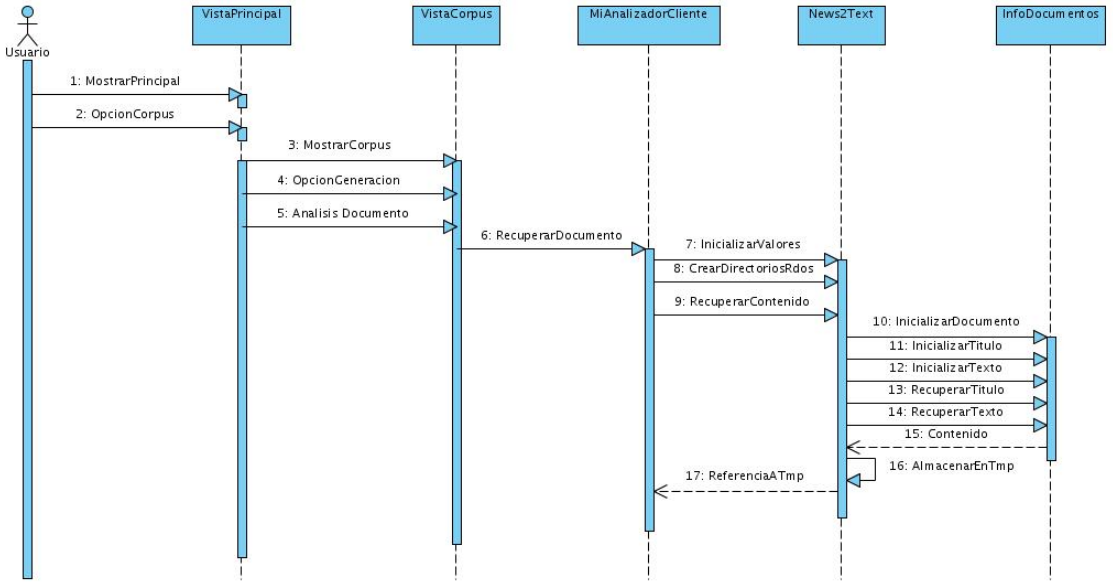


Figura 72 diseño diag.sec.extraer contenido documento

(E) LANZAR ANÁLISIS

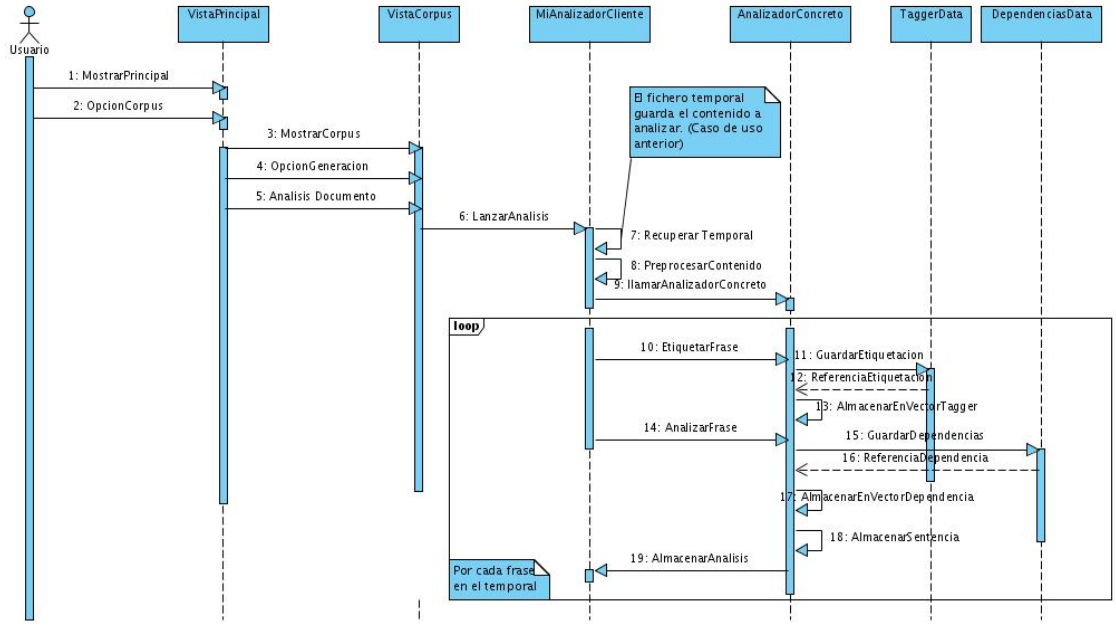


Figura 73 diseño diag.sec.lanzar análisis

(F) ALMACENAR ANÁLISIS

Este caso de uso es muy amplio, pues involucra a todas las clases implicadas en la base de datos. Para su mejor comprensión se divide el diagrama en dos partes.

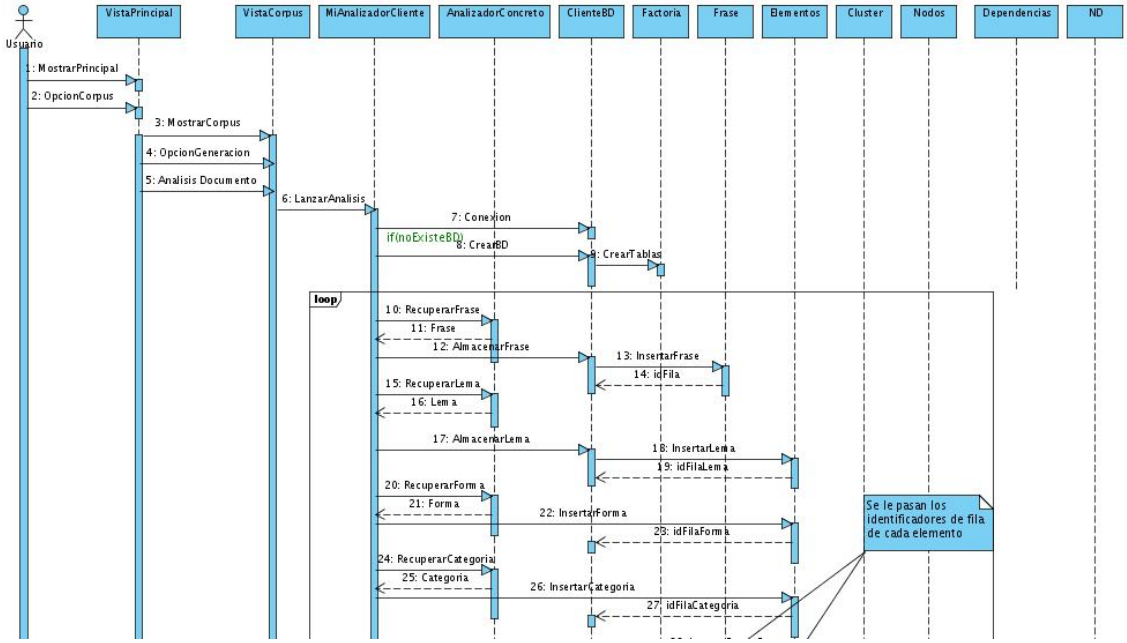


Figura 74 diseño diag.sec.almacenar análisis 1

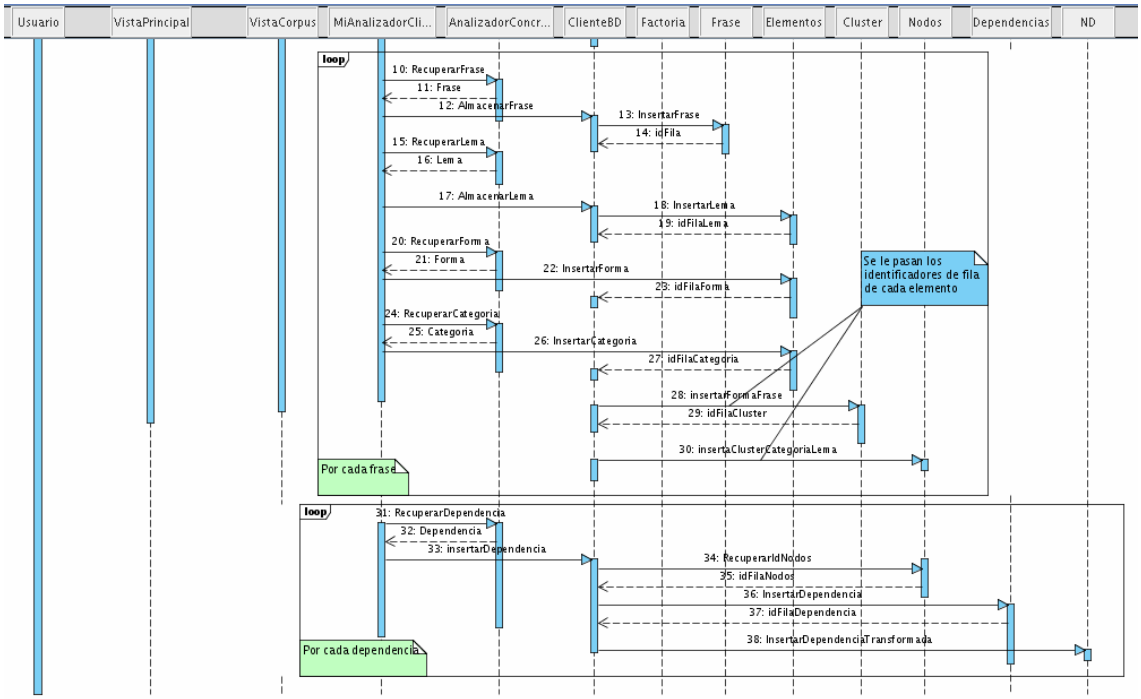


Figura 75 diseño diag.sec.almacenar análisis 2

7.3.CASO DE USO: GESTIÓN BÚSQUEDAS

7.3.1.BÚSQUEDA POR TÉRMINOS

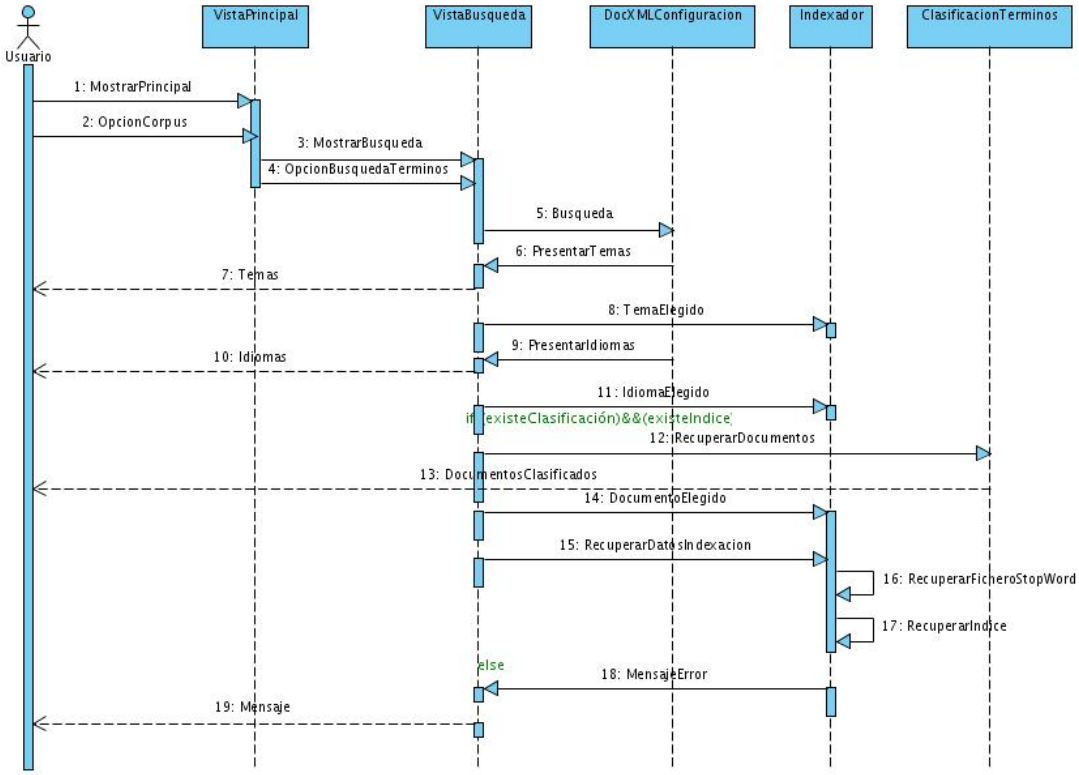


Figura 76 diseño diag.sec.búsqueda términos

7.3.2.RECUPERAR DOCUMENTO POR TÉRMINOS (ESCENARIO 1): CON ONTOLOGÍA

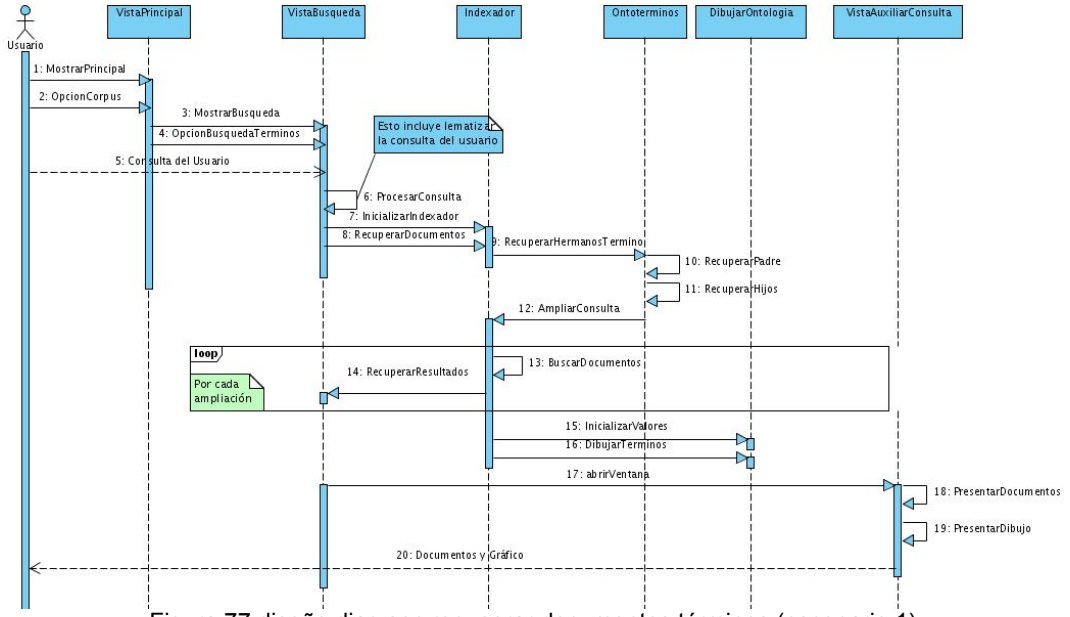


Figura 77 diseño diag.sec.recuperar documentos términos (escenario 1)

7.3.3.RECUPERAR DOCUMENTO POR TÉRMINOS (ESCENARIO 2): SIN ONTOLOGÍA

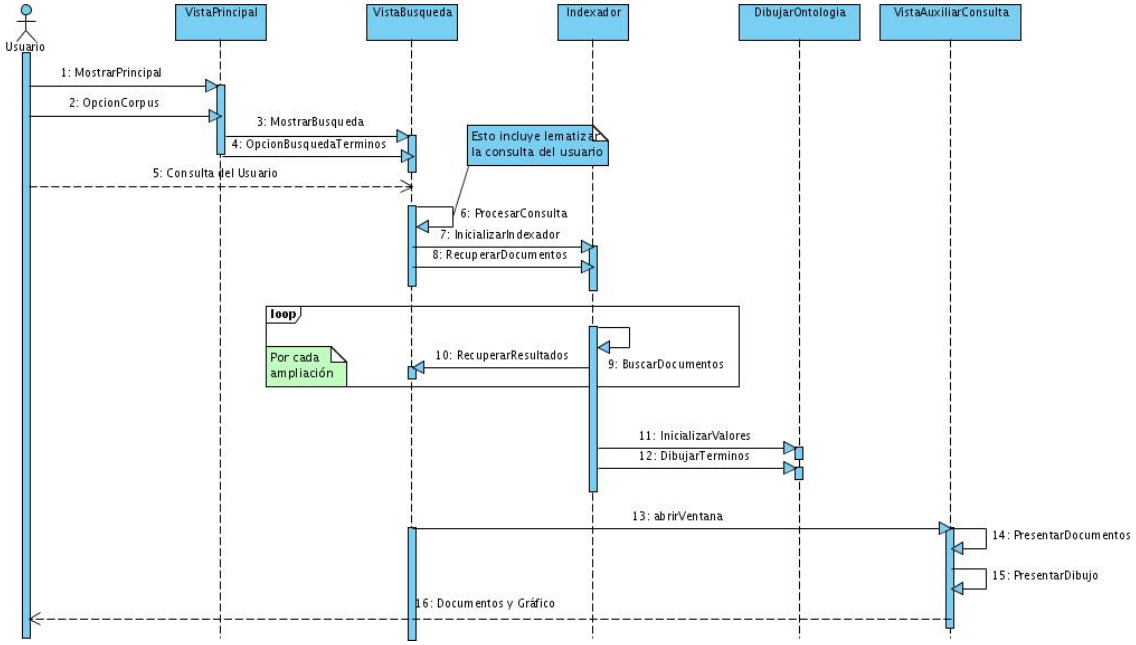


Figura 78 diseño diag.sec.recuperar documentos términos (escenario 2)

7.3.4.BÚSQUEDA POR DEPENDENCIAS

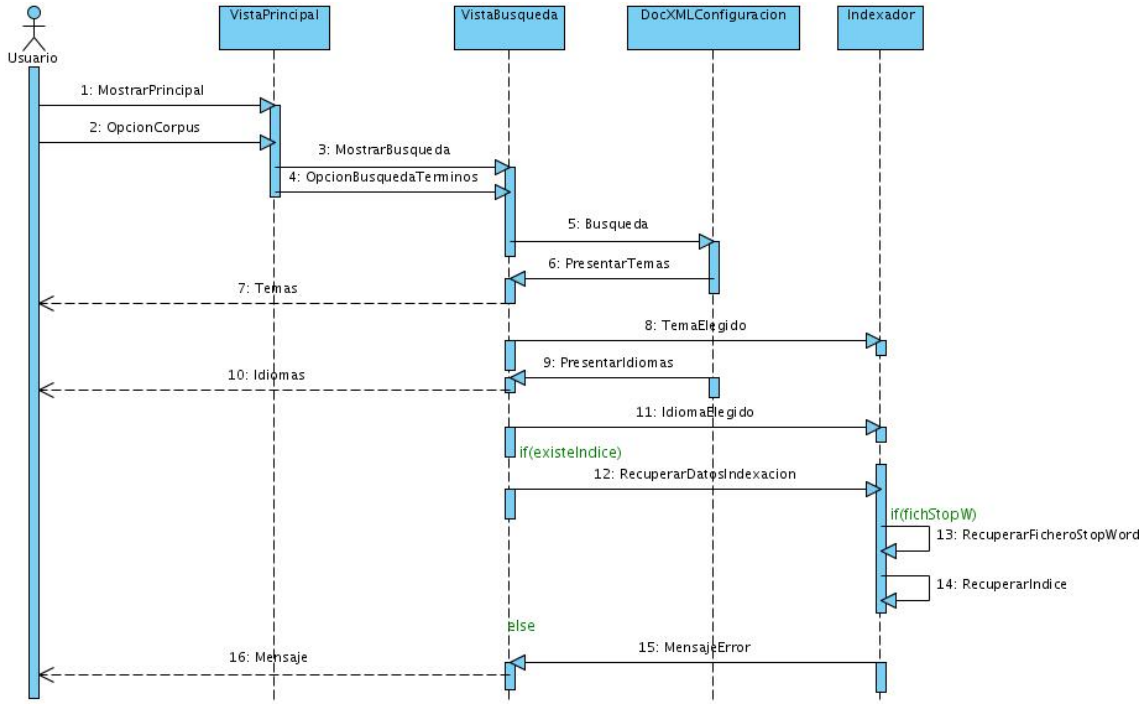


Figura 79 diseño diag.sec.recuperar documentos dependencias (escenario 1)

7.3.5.RECUPERAR DOCUMENTO POR DEPENDENCIAS (ESCENARIO 1): CON ONTOLOGÍA

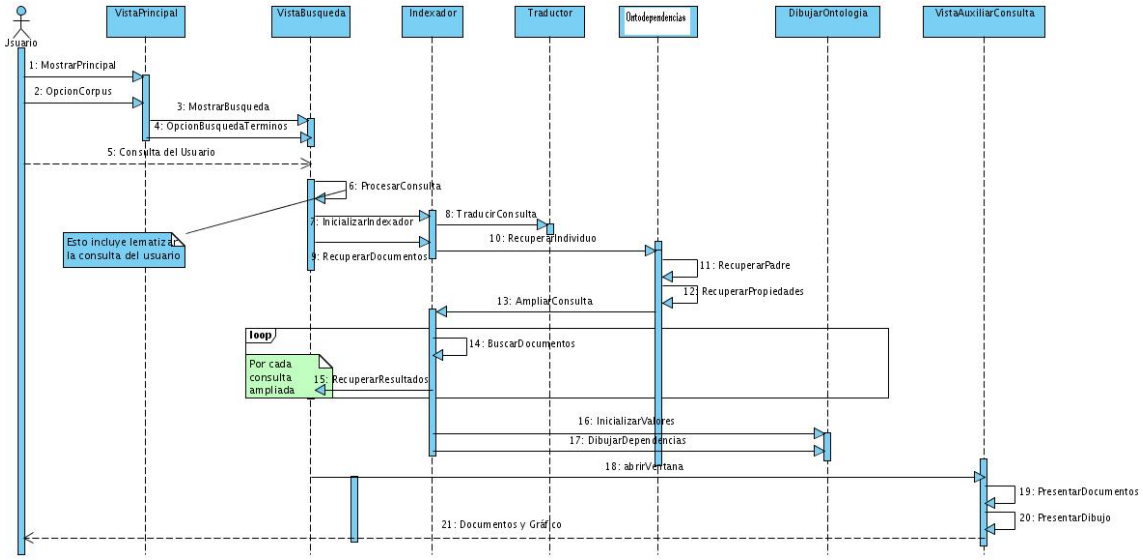


Figura 80 diseño diag.sec.recuperar documentos dependencias (escenario 1)

7.3.6.RECUPERAR DOCUMENTO POR DEPENDENCIAS (ESCENARIO 2): SIN ONTOLOGÍA

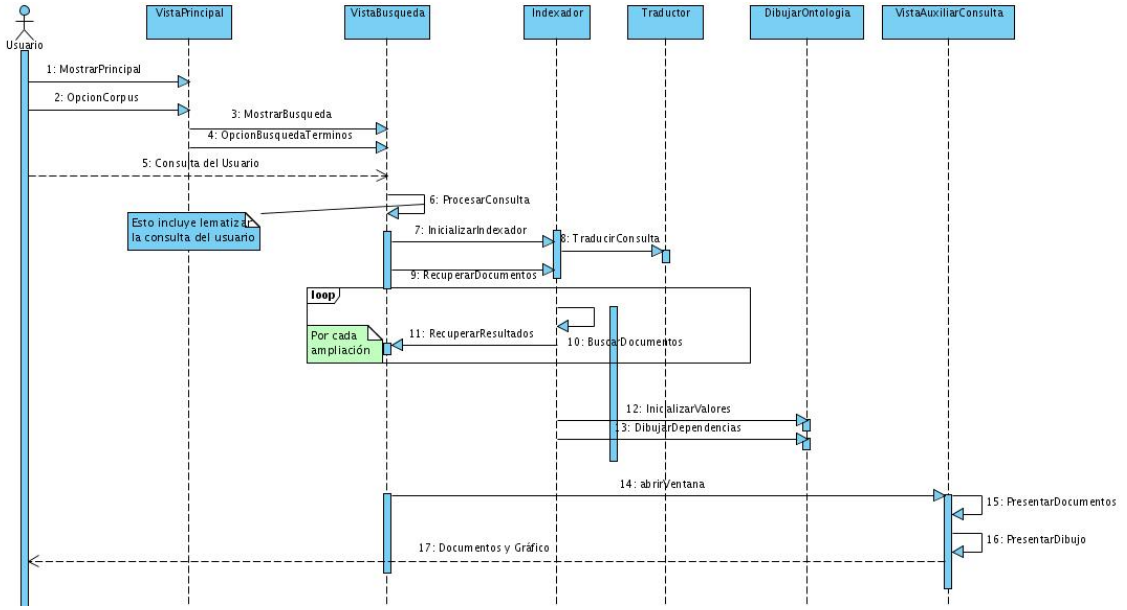


Figura 81 diseño diag.sec.recuperar documentos dependencias (escenario 2)

7.4.GESTIÓN CORPUS

7.4.1.AÑADIR VARIOS DOCUMENTOS

Al igual que en el análisis, añadir varios documentos al sistema es equivalente a añadirlos de uno en uno. Es por esta razón, por la que este caso de uso incluye todos los que se describen a continuación.

7.4.2.AÑADIR UN DOCUMENTO

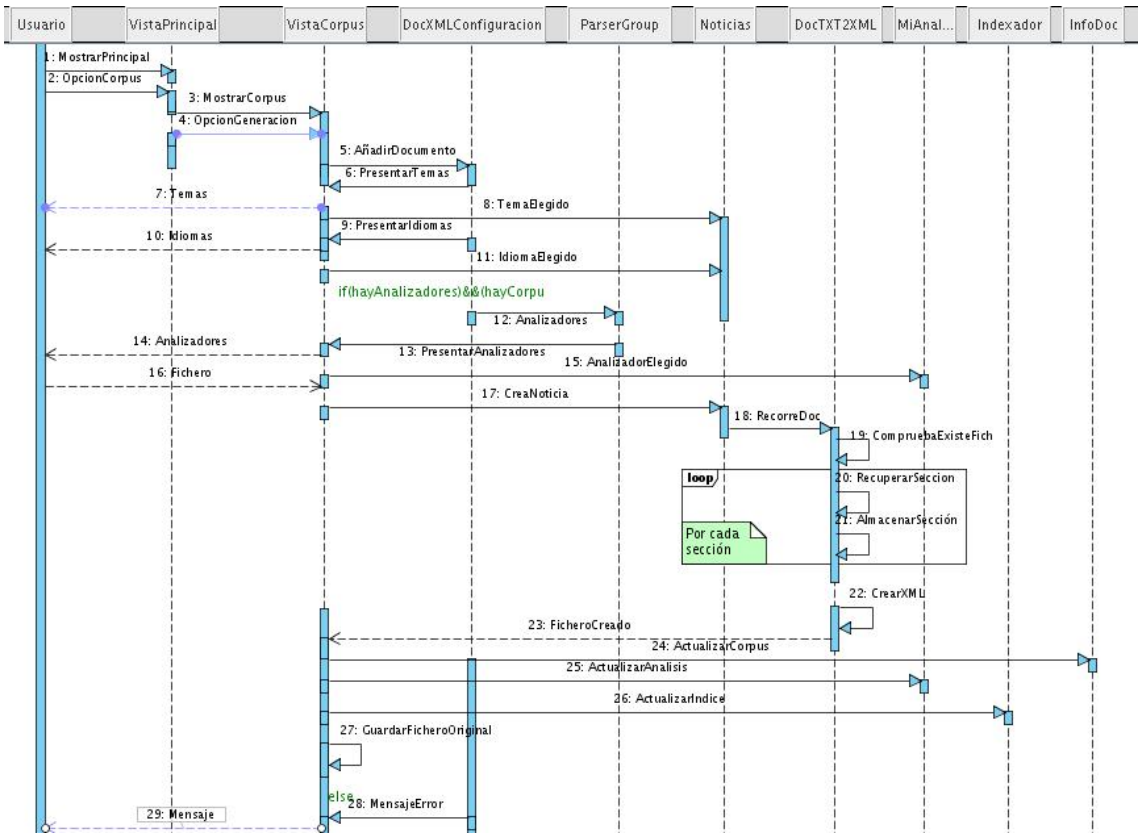


Figura 82 diseño diag.sec.añadir documento

7.4.3.EXTRAER CONTENIDO BASE

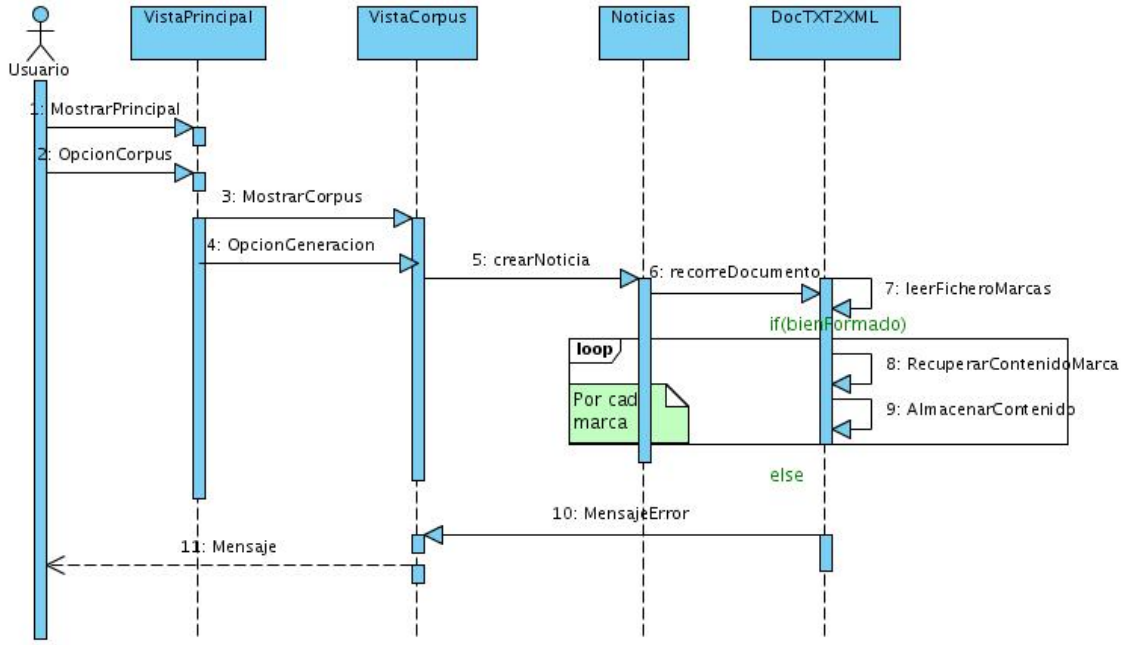


Figura 83 diseño diag.sec.extraer contenido base

7.4.4.GENERAR DOCUMENTO

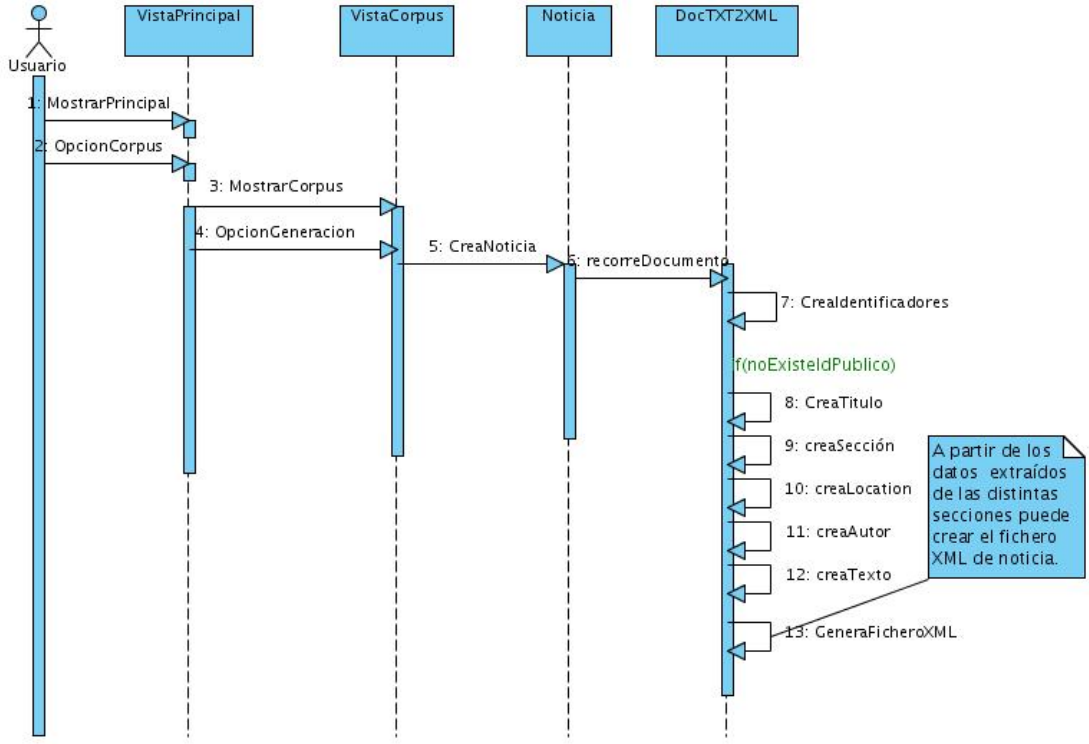


Figura 84 diseño diag.sec.generar documento

7.4.5.ACTUALIZAR CORPUS

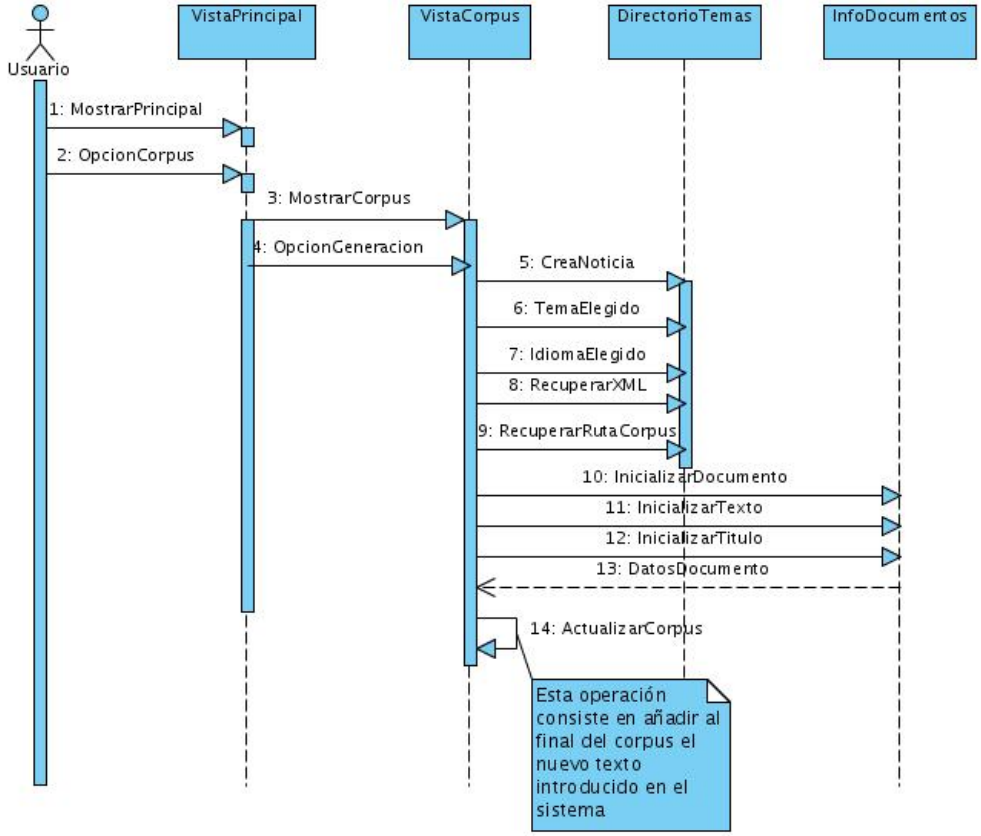


Figura 85 diseño diag.sec.actualizar corpus

7.4.6.GESTIÓN ÍNDICE (ESCENARIO 1): CREAR EL ÍNDICE

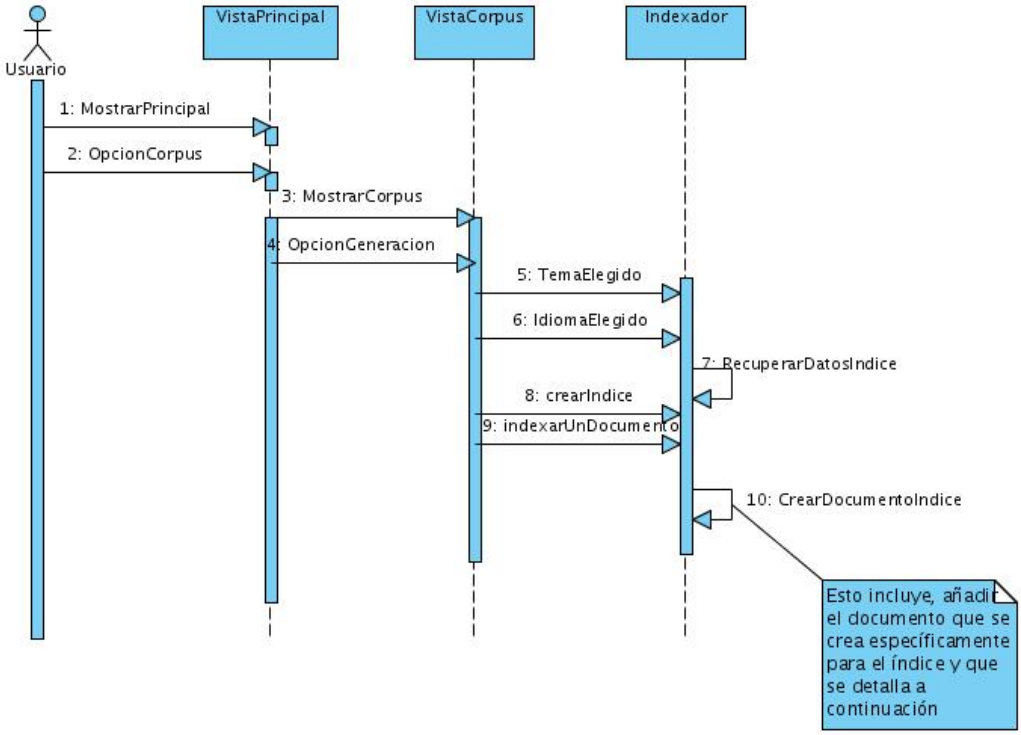


Figura 86 diseño diag.sec.crear índice

7.4.7.GESTIÓN ÍNDICE (ESCENARIO 1): ACTUALIZAR EL ÍNDICE

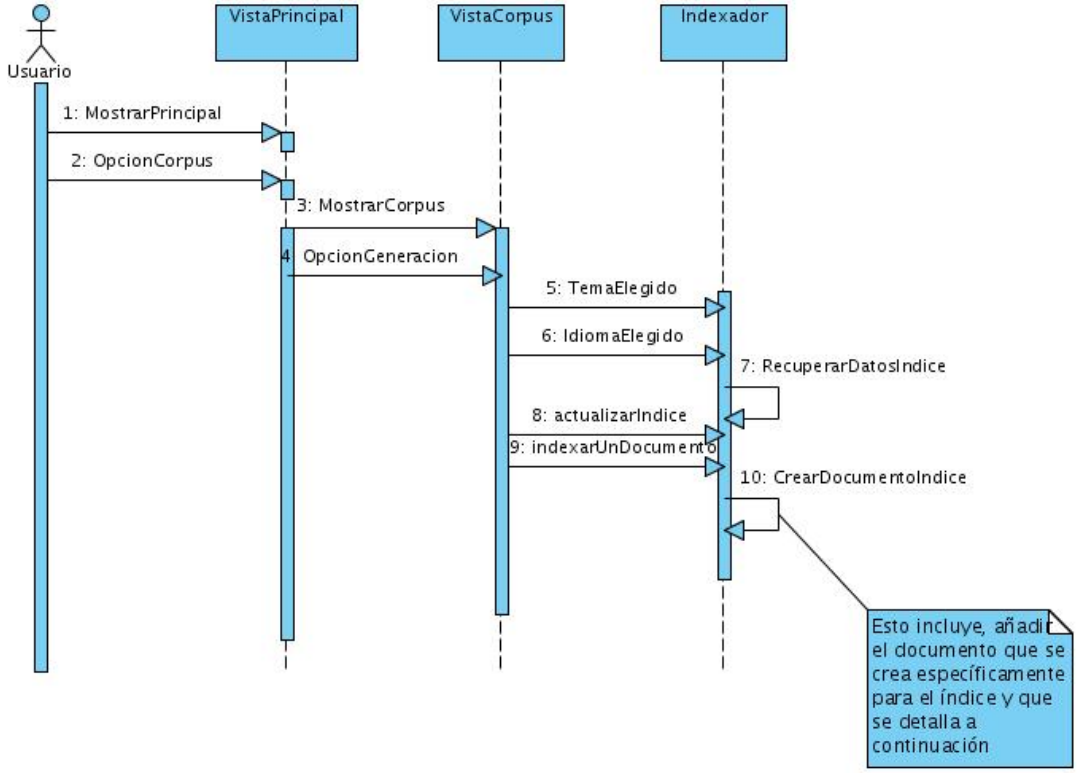


Figura 87 diseño diag.sec.actualizar índice

7.4.8.CREAR DOCUMENTO ÍNDICE

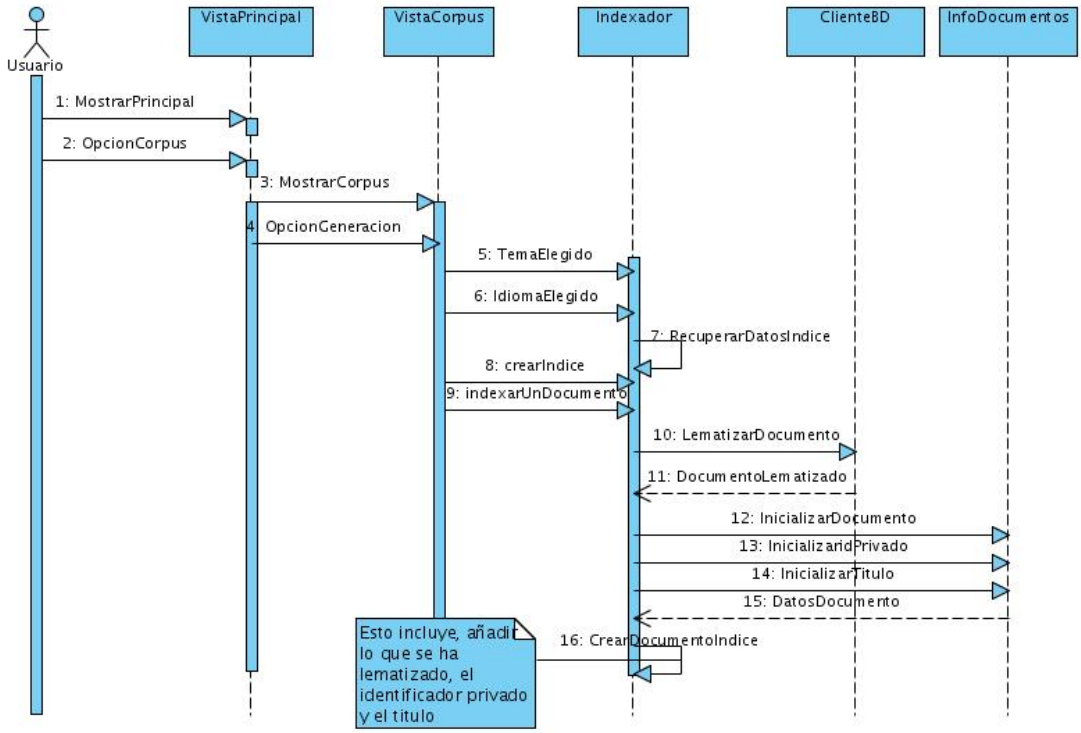


Figura 88 diseño diag.sec.crear documento índice

8. PATRONES DE DISEÑO

La idea principal de los patrones es el concepto de estandarización de la información sobre un problema común y su solución.

Los patrones de diseño representan una evolución importante en la abstracción y reutilización del software. Estos dos conceptos son básicos en la programación. La abstracción representa la forma que tienen los desarrolladores para resolver problemas complejos dividiéndolos en otros más simples. La reutilización es igual de vital para el desarrollo de software.

8.1. PATRÓN ESTRATEGIA

El patrón estrategia es un patrón de comportamiento. Este tipo de patrones está relacionado con el flujo de control en un sistema. Ciertas formas de organizar el control en una aplicación pueden derivar en grandes beneficios para la eficiencia y el mantenimiento del sistema.

El propósito de este patrón es definir un grupo de clases que representan un conjunto de posibles comportamientos. Estos comportamientos pueden ser fácilmente intercambiados en una aplicación, modificando la funcionalidad en cualquier instante.

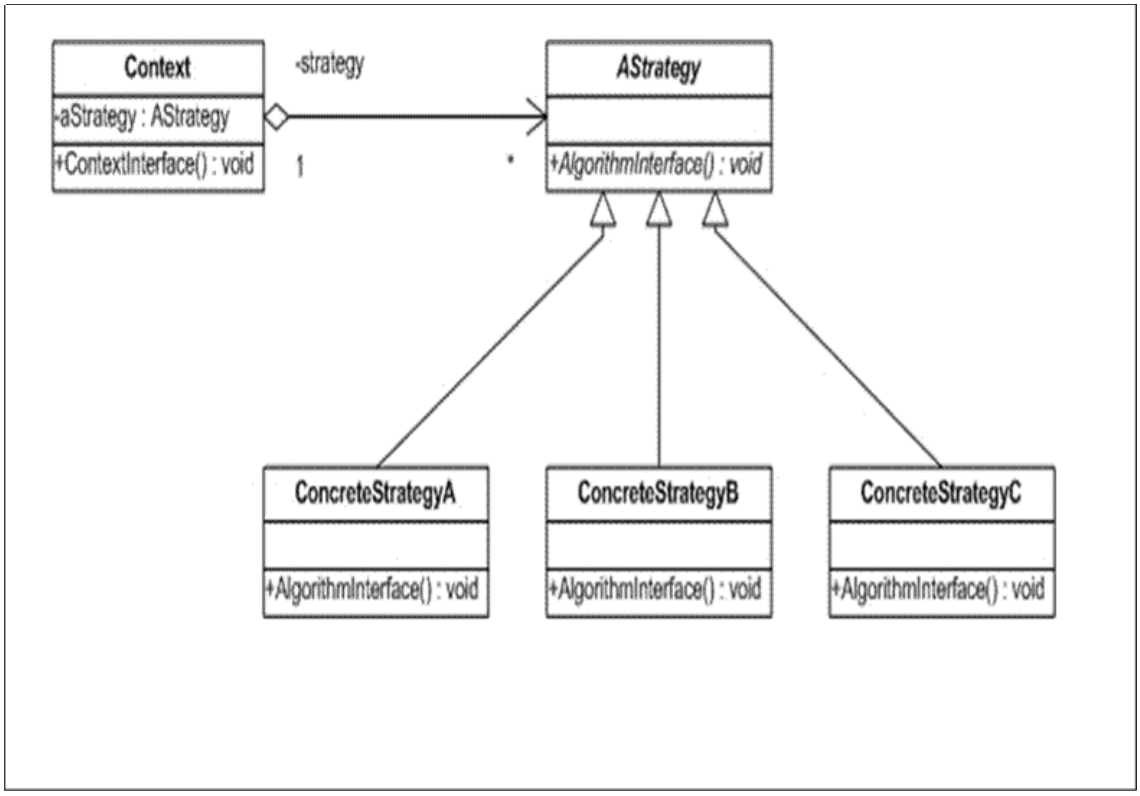


Figura 89 patrón estrategia

Este patrón se ha empleado tanto en la implementación de la extracción como la implementación del análisis sintáctico. En ambos casos las estrategias para llevar a cabo las funcionalidades dependen del algoritmo empleado en cada uno de los extractores y/o de los analizadores. La clase cliente solicita la ejecución de alguna de las clases que pertenezcan a la interfaz, bien de los extractores bien de los analizadores sintácticos, siendo para ella transparente la ejecución de cualquiera de esas tareas.

Este patrón también se ha utilizado para extraer la información de los documentos. Pues la aplicación está enfocada en permitir que los documentos del sistema sean proporcionados en distintos formatos. En estos momentos el enfoque se centra en archivos *txt*, pero la arquitectura implementada permite la fácil adición de otros formatos, por ejemplo documentos con una extensión *pdf*. Además, como se describe en la sección de implementación, los textos proporcionados al sistema deben estar estructurados de un determinado modo. Con el patrón estrategia se consigue por un lado que los documentos tengan asociadas varias extensiones, y por otro lado que la estructura de los mismos cumpla varios formatos definidos para el sistema implementado.

8.2.PATRÓN DAO Y PATRÓN DE FÁBRICA ABSTRACTA

También se ha utilizado a lo largo de la implementación estos dos patrones de manera combinada.

El acceso al almacenamiento persistente, como una base de datos, varía en gran medida dependiendo de la estrategia usada para guardar los datos. El patrón DAO surge como respuesta al problema de la poca homogeneidad del almacenamiento persistente, el cual produce distintas implementaciones para lograr la accesibilidad a los datos. Un cambio en el sistema de almacenamiento puede conducir a una reimplantación de los componentes de datos y componentes vinculados que requieren esos datos.

El objetivo perseguido con el uso de este patrón es desacoplar la lógica de negocio de la lógica de acceso a datos, de manera que se pueda cambiar la fuente de datos fácilmente. Consiste en utilizar un objeto de acceso a datos para abstraer y encapsular todos los accesos a la fuente de datos.

Se pueden resumir las ventajas del uso de este patrón en la independencia que aporta respecto al tipo de recurso utilizado (base de datos relacional, base de datos orientada a objetos, ficheros XML, ficheros planos, etc.) y en la reducción de la complejidad de la implementación de la lógica de negocio.

El patrón DAO se puede flexibilizar adoptando el patrón fábrica abstracta. Cuando el almacenamiento está sujeto a cambios de una implementación a otra, la combinación de ambos patrones es adecuada. Esta estrategia proporciona un objeto factoría abstracta de DAO que puede construir varios tipos de factorías concretas DAO, cada factoría soporta un tipo diferente de implementación del almacenamiento persistente. Una vez que se obtiene la factoría concreta de DAO para una implementación específica, se usa para producir los DAO soportados e implementados para tal caso.

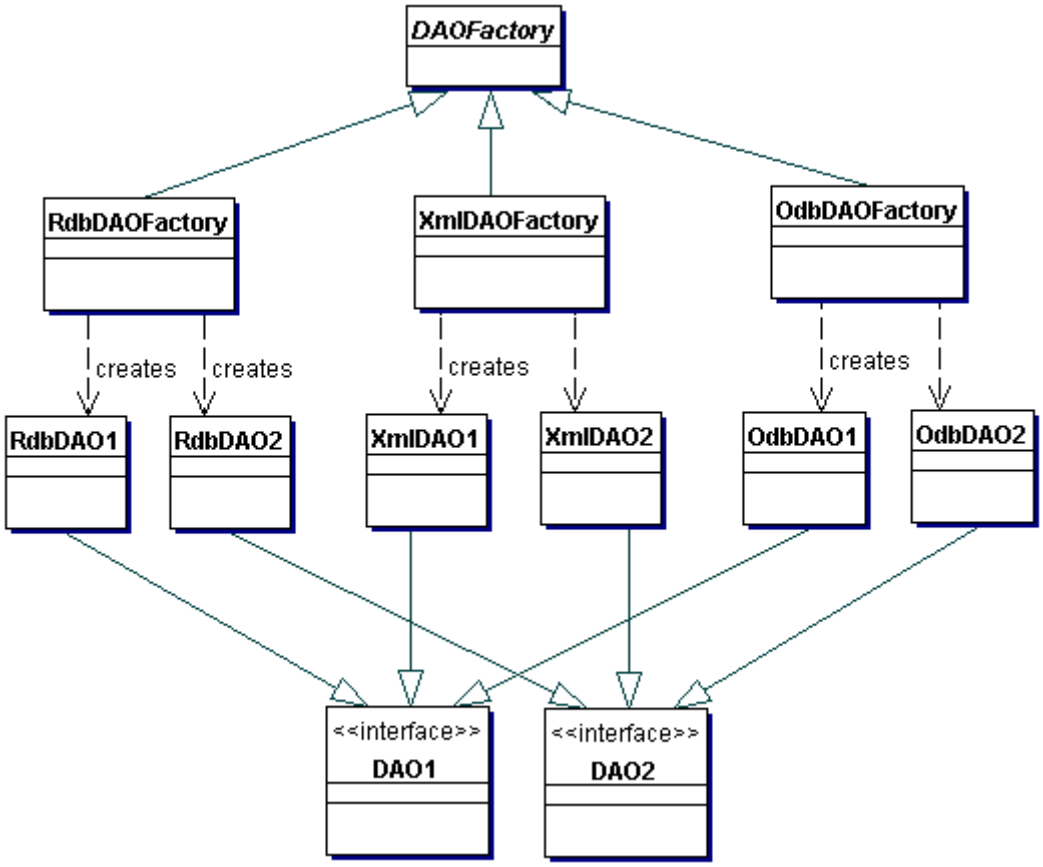


Figura 90 patrón DAO y fábrica abstracta

Estos patrones se han utilizado en la implementación de la base de datos.

9. DISEÑO DE LA BASE DE DATOS

En esta sección se hace una descripción detallada de las tablas tenidas en cuenta para el almacenamiento del grafo de dependencias en la base de datos. Asimismo, también se muestra el diagrama de entidad/relación. Para mejorar la legibilidad del diagrama se ha optado por no incluir los atributos de las entidades. Estos atributos se describen a continuación en las tablas.

9.1. DIAGRAMA DE ENTIDAD/RELACIÓN

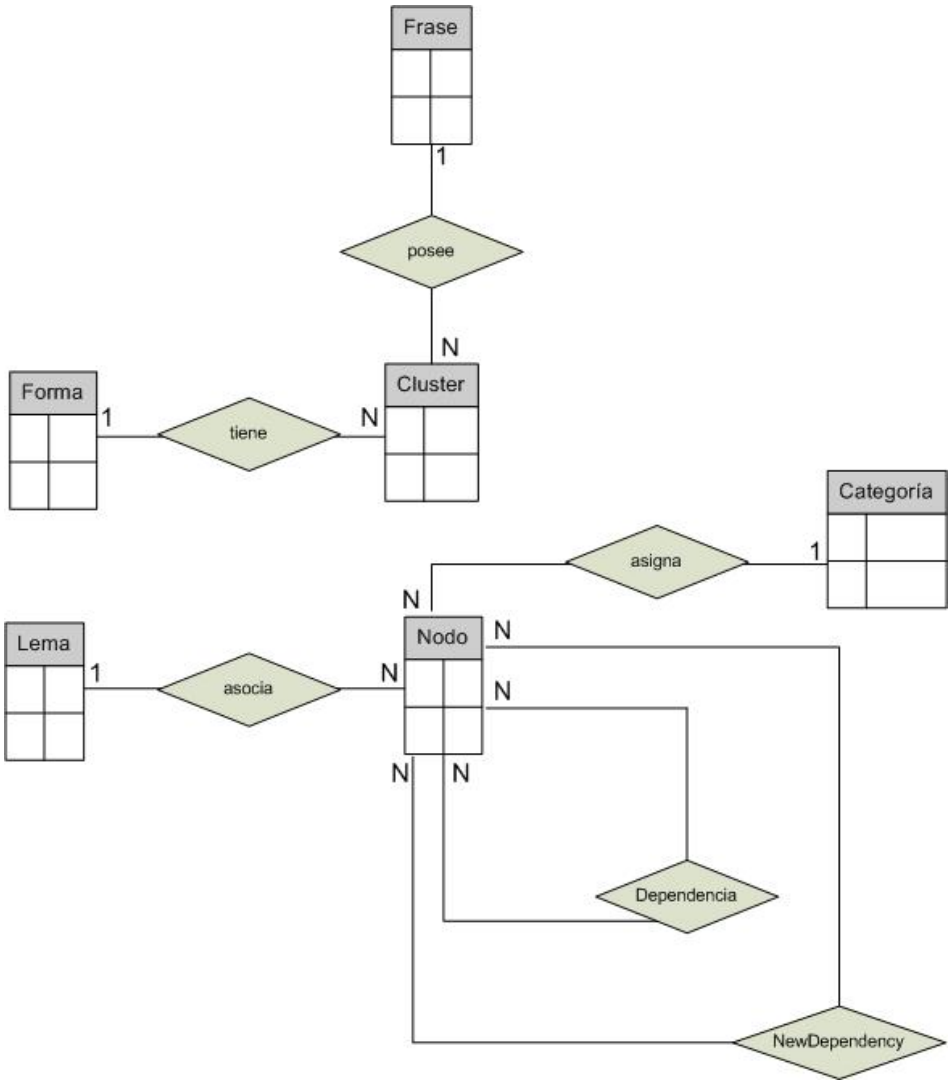


Figura 91 diagrama de entidad/relación

9.2.PASO A TABLAS DEL MODELO ENTIDAD/RELACIÓN

Se aplican un conjunto de reglas a la hora de pasar a tablas las entidades y atributos del diagrama de entidad/relación:

- Los conjuntos de entidades regulares se transforman en tablas, eligiendo una de las claves candidatas como primarias.
- Las relaciones binarias 1:N se resuelven introduciendo una clave foránea en la entidad del lado N que hace referencia a la entidad del lado 1.
- Las relaciones N:M generan una tabla adicional cuya clave es la combinación de las claves primarias de las entidades participantes. En el caso de que la relación tuviese atributos, éstos serán incorporados como atributos de la tabla también.

CATEGORÍA		
Nombre	Tipo	Descripción
id_categoria	int	Es la clave de la tabla Categoría. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
categoria	varchar	Valor de la categoría.

LEMA		
Nombre	Tipo	Descripción
id_lema	int	Es la clave de la tabla Lema. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
lema	varchar	Valor del lema.

FORMA		
Nombre	Tipo	Descripción
id_forma	int	Es la clave de la tabla Forma. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
forma	varchar	Valor de la forma.

CLUSTER		
Nombre	Tipo	Descripción
id_cluster	int	Es la clave de la tabla Categoría. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
cod_forma	int	Clave foránea que referencia a la forma asociada a ese cluster.
cod_frase	int	Clave foránea que referencia a la frase asociada a ese cluster.
pos	int	Posición que ocupa la palabra actual en la frase actual.

DEPENDENCIA		
Nombre	Tipo	Descripción
id_node_source	int	Clave foránea que referencia al nodo origen asociado con esa dependencia.
id_node_fin	int	Clave foránea que referencia al nodo destino asociado con esa dependencia.
label	varchar	Nombre asociado a la dependencia.
taux	float	Probabilidad de que la dependencia sea tenida en cuenta.

FRASE		
Nombre	Tipo	Descripción
id_frase	int	Es la clave de la tabla Frase. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
nb_archivo	varchar	Nombre del archivo que almacena la frase, será un valor único en la base de datos.
sentencia	varchar	Valor de la frase.

NEWDEPENDENCY		
Nombre	Tipo	Descripción
id_ndep	int	Es la clave de la tabla NewDependency. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
id_source	int	Clave foránea que referencia al nodo origen asociado con esa dependencia.
id_fin	int	foránea que referencia al nodo destino asociado con esa dependencia.
label	varchar	Nombre asociado a la dependencia.
taux_conf	float	Probabilidad de que la dependencia sea tenida en cuenta.
nom	varchar	Nombre asociado con la clasificación creada a partir de las dependencias.
poid	int	Establece si la dependencia actual va a ser tomada en cuenta.
del	float	Peso asignado a la dependencia actual.

En esta tabla se almacena las dependencias existentes en la tabla Dependencia salvo ciertos casos, que se describen a continuación:

- Las dependencias generadas entre un sujeto y un verbo y entre ese mismo verbo y un objeto directo, se transforman en una única dependencia entre el sujeto y el objeto directo en cuya etiqueta se añade el verbo implicado.
- Lo mismo sucede en las frases pasivas entre el sujeto y el complemento agente.
- Las dependencias que sólo tiene sujeto o sólo tiene objeto directo, se mantienen como estaban.
- Las demás dependencias también se mantienen con sus valore originales.

Esta tabla se crea para recuperar los datos necesarios en la clasificación por dependencias y poder almacenar en el campo *nom* el tipo de relación planteado entre los elementos de la dependencia. Se podría pensar en realizar directamente las operaciones sobre la tabla *Dependencia* sin embargo la clasificación elimina aquellas dependencias no relevantes y por lo tanto se perdería un subconjunto de las dependencias iniciales.

NODE		
Nombre	Tipo	Descripción
id_node	int	Es la clave de la tabla Node. Es un valor auto-incrementable, por lo tanto nunca existirán valores repetidos.
cod_categoria	int	Clave foránea que referencia a la categoría asociada con ese nodo.
cod_lema	varchar	Clave foránea que referencia al lema asociado con ese nodo.
cod_cluster	int	Clave foránea que referencia al cluster asociado con ese nodo.
peso_categoria	float	Probabilidad de que la categoría sea correcta.

IMPLEMENTACIÓN

10.IMPLEMENTACIÓN GENERAL

En la fase de implementación se obtiene el código fuente del sistema. En nuestro caso serán clases Java principalmente para llevar a cabo toda la lógica de la aplicación.

Este apartado resulta de interés para comprender, desde una óptica de bajo nivel, el funcionamiento interno del sistema desarrollado. Con esta finalidad se hará una descripción de los siguientes aspectos:

- Diagrama de componentes.
- Estructura de los distintos ficheros de apoyo y almacenamiento.
- Detalles de ejecución.
- Implementación de las tareas de PLN.
- Clasificación de las dependencias y de los términos.
- Construcción de la ontología.

A continuación, se intentará detallar lo máximo posible estos puntos.

10.1.DIAGRAMA DE COMPONENTES

El diagrama de componentes se utiliza para modelar la estructura del software, incluyendo las dependencias entre sus componentes, los componentes de código binario, y los ejecutables.

En el diagrama de componentes se modelan los correspondientes al sistema, en ocasiones agrupados por paquetes, y las dependencias que existen entre dichos componentes.

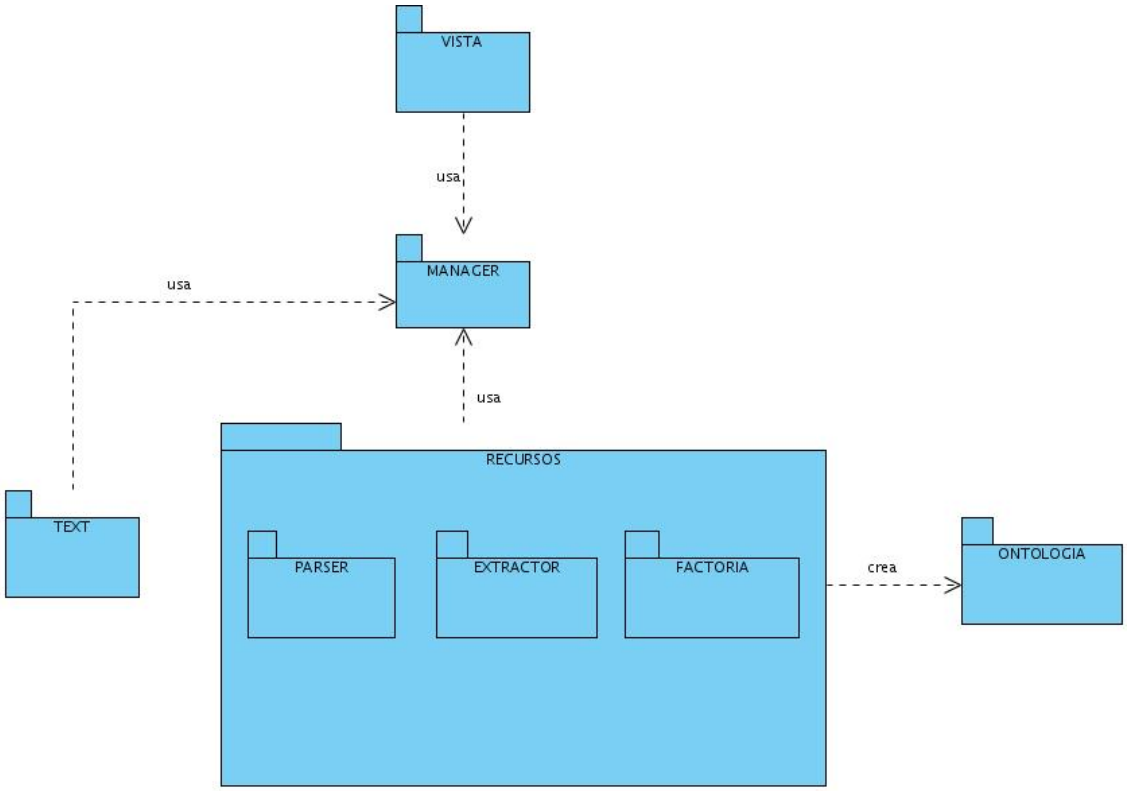


Figura 92 diagrama de componentes

- **vista**: Este paquete incorpora las clases que implementan la interfaz gráfica del sistema.

- **manager** : Este paquete es el encargado de recoger las peticiones solicitadas por el usuario, así como del manejo del fichero de configuración

recursos: Este paquete agrupa el conjunto de funcionalidades principales del sistema.

  - extractor: Maneja toda la información relativa a la tarea de extracción de terminología.

  - parser: Agrupa las clases relacionadas con el análisis sintáctico.

  - factoria: Gestiona el manejo de la base de datos, utilizada para el análisis sintáctico y para la extracción del conocimiento.

- **text**: Encargado de manipular los recursos textuales. Es decir, genera el corpus, estructura los documentos en los ficheros XML, recupera la información de los documentos, etc.

- **ontología**: Lanza el algoritmo de extracción de la ontología, ordena los resultados y organiza los términos obtenidos en el proceso de extracción terminológica. También incorpora la gestión del módulo de recuperación de documentos

10.2. INTERPRETACIÓN DEL FICHERO DE CONFIGURACIÓN

Esta sección se centra en detallar el fichero de configuración, parte esencial del sistema. Para una mejor comprensión del mismo se ha optado por describirlo a través del propio fichero XML y no a través de su DTD. Se presenta la lógica asociada a cada una de las partes del archivo:

```
<idiomas>
	<idioma abr="es" nombre="espanol">
		<basedatos gestor="mysql" tema="economia">economiaES</basedatos>
		<ontologia stopword="y"/>
	</idioma>
	<idioma abr="fr" nombre="frances">
		<basedatos gestor="mysql" tema="economia">economiaFR</basedatos>
		<ontologia stopword="y"/>
	</idioma>
</idiomas>
```

Dentro de la rama *idiomas* se encuentran todos los detalles relativos a una lengua. Cada idioma se identifica por su nombre y por su abreviatura. Estos dos atributos tienen un valor que es único en todo el fichero, es decir sólo existe un idioma llamado *espanol* y sólo existe un abreviatura que sea *es*.

En el interior de la marca *idioma* existe la posibilidad de definir una etiqueta que referencia a las bases de datos que se pueden utilizar para ese lenguaje. Concretamente habrá una base de datos por tema e idioma. Estas bases de datos se describen en *basedatos,* indicando en el atributo *gestor* el gestor de base de datos correspondiente y en el atributo *tema* se referencia el tema en él que está incluido el idioma. El campo de texto almacena el nombre asignado a la base de datos que también deberá ser único en todo el fichero, pues será una combinación del nombre del tema y de la abreviatura del tema, ambos valores únicos en el fichero.

```
<extractores>

<ext lang="it"/><ext lang="es">
      <extractor name="fastr_es">
            <rutaExtract>Extract/espanol/FastrEs/</rutaExtract>
            <clase>universidad.proyecto.recursos.extractor.FastrES</clase>
      </extractor>
      <extractor name="gamallo">
            <rutaExtract>Extract/espanol/Gamallo/</rutaExtract>
            <clase>universidad.proyecto.recursos.extractor.Gamallo</clase>
      </extractor>
</ext>
<ext lang="fr">
      <extractor name="fastr_fr">
            <rutaExtract>Extract/frances/FastrFr/</rutaExtract>
            <clase>universidad.proyecto.recursos.extractor.FastrFR</clase>
      </extractor>
      <extractor name="acabit">
            <rutaExtract>Extract/frances/Acabit/</rutaExtract>
            <clase>universidad.proyecto.recursos.extractor.Acabit</clase>
      </extractor>
</ext>
</extractores>
```

Las marcas *extractores* albergan la información relativa a estas herramientas. Para cada idioma existe un conjunto de recursos de extracción que se engloban bajo las etiquetas *ext*.

Como puede apreciarse el idioma se referencia mediante su abreviatura a través del atributo *lang*. Las características de cada herramienta de extracción se sitúan en el subárbol *extractor,* cuyo atributo *name* contiene el nombre de cada extractor y que será único en todo el fichero.

El subárbol *extractor* contiene los siguientes nodos hoja:

rutaExtract → Camino relativo donde se ubica el extractor

clase → Nombre de la clase[7] que implementa la extracción

Gracias a esta última información se podrán realizar las llamadas dinámicas en función del extractor peticionado.

```
<parsers>
      <parser lang="it"/><parser lang="es">
            <analyzer name="erialES">
                  <rutaParser>Analyzer/espanol/erialES</rutaParser>
      <claseParser>universidad.proyecto.recursos.parser.ErialES</claseParser>
            </analyzer>
      </parser>
```

[7] Es necesario indicar también el paquete donde se incluye la clase.

```
      <parser lang="fr">
            <analyzer name="erialFR">
                  <rutaParser>Analyzer/frances/erialFR</rutaParser>
      <claseParser>universidad.proyecto.recursos.parser.ErialFR</claseParser>
            </analyzer>
      </parser>
</parsers>
```

En la tabla anterior se muestran los datos relativos a los analizadores sintácticos. La interpretación es similar al caso de los extractores. Cada idioma, referenciado por su abreviatura, tiene asignado un conjunto de analizadores (en el caso del ejemplo sólo un analizador por idioma). Para cada uno de los analizadores, cuyo nombre es único en la configuración, se guarda información sobre las rutas relativas necesarias para la ejecución, así como la clase que implementa la estrategia de análisis.

```
<directory tema="economia">
<formato clase="universidad.proyecto.text.processText.DocTXT2XML" tipo="txt">
      <rfich lang="es">textos/economia/textosTXT/espanol/</rfich>
      <rfich lang="fr">textos/economia/textosTXT/frances/</rfich>
</formato>
<rxml lang="es">textos/economia/textosXML/espanol/</rxml>
<rxml lang="fr">textos/economia/textosXML/frances/</rxml>
<corpus>textos/economia/corpus/</corpus>
</directory>
```

Por último, se presenta la información asociada a los tema activos. Bajo el árbol *directory* se guarda información sobre los temas que se pueden aplicar en el sistema así como las rutas relativas para acceder a los documentos. La marca *formato* incluye los datos relacionados a las fuentes textuales.

Se contempla la posibilidad de tener los ficheros en diversos formatos. En el caso concreto, nuestras fuentes textuales están en ***txt***[8]***,*** pero podrían estar en *pdf*, en ***xml***, etc. El hecho de manejar distintos formatos implica la utilización de distintas clases en cada caso, por lo que dicha información se guarda en la variable *clase* de la etiqueta *formato*.

La etiqueta *formato* tiene un subárbol *rfich*, concretamente habrá tantos subárboles de este tipo como idiomas aplicados al tema actual. En ella se guarda la ruta donde se almacenan las fuentes textuales originales, proporcionadas al sistema.

[8] La implementación realizada sólo abarca el manejo de ficheros ***txt***, pero el diseño de la arquitectura es tal, que permite una fácil adaptación a futuros formatos.

Al igual que el caso de *rfich*, existirán tantas etiquetas *rxml* como idiomas se haya asignado al tema correspondiente. En el ejemplo, para el tema de economía se manipulan dos idiomas. Se ha comentado anteriormente que el empleo de un idioma en un determinado tema depende de la tenencia de fuentes textuales, de ahí que cada tema sólo tenga asignado un conjunto de idiomas. Así, en el interior del campo *rxml* se guarda la ruta donde se van a almacenar los ficheros XML generados para cada documento o fuente textual.

Por último y de manera intuitiva la etiqueta *corpus* guarda la ruta donde almacenar los ficheros que contienen los corpus de cada idioma.

10.3.FORMATO DE LOS DOCUMENTOS

Antes de explicar cual es la estructura de los documentos, es importante señalar que ha sido necesario incluir un identificador de fichero que fuese único para cada archivo. Esto se consiguió mediante el uso de la librería fast-MD5_2.6, disponible en la dirección *http://www.twmacinta.com/myajva/fast_md5.php*. Esta librería implementa un algoritmo de *hash* MD5 totalmente escrito en java. MD5 es un acrónimo que significa "Message Digest Algorithm 5". Este algoritmo hace una reducción criptográfica de 128 bits ampliamente usado. De este modo incluyendo un pequeño código antes de generar el archivo, se obtiene un código hash del archivo que no podrá coincidir con este mismo código de otro archivo. De este modo ficheros con el mismo contenido sólo se guardarán una vez.

Originalmente los documentos pueden venir dadas en distintos tipos de formatos. Sin embargo se ha diseñado el sistema de tal manera que el contenido de los mismos siga un determinado patrón mostrado en la siguiente tabla

Como se explicara en el apartado *10.6 implementación del patrón estrategia* aunque en estos momentos los documentos son noticias de periódicos (y es el ejemplo que aquí se presenta) se podrían incorporar documentos diferentes de forma sencilla.

Mediante la utilización de marcas, cada porción del texto es destacada para optimizar su almacenamiento en el fichero XML. Para una mejor comprensión se presenta un ejemplo:

```
#headline>¿Es necesario intervenir en los mercados de divisas?<headline#
#words>628 palabras.<words#
#date>16 de diciembre de 2003.<date#
#section>Le Monde Economie.<section#
#numero>2.<numero#
#original>francés.<original#
#media>(c) Le Monde, 2003.<media#
#text>«¡El banco! ¡El banco!»
   En la sala de mercado de una gran financiera parisina, el grito del
cambista encargado de fijar la paridad franco-marco provoca el pánico. El
Banco de Francia acaba de ponerse en contacto con él para comprarle 100
millones de francos. En cuestión de segundos, la información da la vuelta al
mundo, y aparece en miles y miles de pantallas de ordenadores. Inmediatamente,
el franco se recupera. Esta escena tuvo lugar en 1995, cuando el franco era
víctima de ataques especulativos. Desde que el euro sustituyó al franco, al
Banco Central Europeo, el Banco de Francia, y la mayoría de los agentes de
cambio emigraron de París hacia Londres. Pero nada cambió en las modalidades
de intervención de los bancos centrales y en la efervescencia que
desencadenan.<text#
#author>P.-A. d.<author#
```

En la tabla anterior se puede ver una noticia en su formato base, antes de ser procesada y almacenada en la estructura XML.

Las marcas destacadas en negrita delimitan cada una de las partes del texto. Dichas marcas están definidas en un fichero XML. Es decir, las marcas pueden ser variadas a gusto del usuario. El único requisito es mantener una etiqueta de inicio y otra de fin para cada componente.

En el fichero XML, correspondiente a las noticias, se ha almacenado la información proporcionada por las fuentes originales pero con ciertas modificaciones. Se muestra a continuación la estructura seguida:

```
<news>
 Etiqueta del documento.
    <idnews>
    Etiqueta que recoge datos identificativos de la noticia
        <date literal="16 de diciembre de 2003">20031216</date>
Etiqueta cuyo contenido recoge en el atributo literal la fecha de publicación
y en el campo de texto un valor calculado a partir de la fecha.
        <idprivate>03_txt</idprivate>
       Identificador único de la noticia que indica el tipo de formato del que
procede.
        <idpublic>16122003978936962810 3</idpublic>
       Identificador único de la noticia. Es un valor calculado y único en el
conjunto de ficheros (se ha descrito al principio de este apartado). Permite
que no se almacenen documentos repetidos el corpus actual.
    </idnews>
    <managmentnews>
    Bajo esta etiqueta se recoge información de gestión de la noticia.
        <media section="Le Monde Economie.">(c) Le Monde, 2003.</media>
        Sección del periódico en la cual la noticia es publicada.
        <lang>es</lang>
       Idioma en el que se encuentra la noticia.
        <original>francés.</original>
       Idioma de cual procede la noticia.
        <words value="628"/>
       Etiqueta que indica el número de palabras que contiene la noticia.
    </managmentnews>
    <contentnews>
    Etiqueta que recoge el contenido de la noticia.
```

```
        <author name="P.-A. d."/>
        Etiqueta que indica el autor o autores de la noticia. En caso de que
hubiese más de un autor, esta etiqueta se repite.
        <location/>
        Etiqueta indicadora del lugar en el cual se publica o escribe la
noticia.
        <headline>¿Es necesario intervenir en los mercados de
divisas?</headline>
        Etiqueta que recoge el título asignado al artículo.
        <text>«¡El banco! ¡El banco!» En la sala de mercado de una gran
financiera parisina, el grito del cambista encargado de fijar la paridad
franco-marco provoca el pánico. El Banco de Francia acaba de ponerse en
contacto con él para comprarle 100 millones de francos. En cuestión de
segundos, la información da la vuelta al mundo, y aparece en miles y miles de
pantallas de ordenadores. Inmediatamente, el franco se recupera. Esta escena
tuvo lugar en 1995, cuando el franco era víctima de ataques especulativos.
Desde que el euro sustituyó al franco, al Banco Central Europeo, el Banco de
Francia, y la mayoría de los agentes de cambio emigraron de París hacia
Londres. Pero nada cambió en las modalidades de intervención de los bancos
centrales y en la efervescencia que desencadenan.
Etiqueta que recoge el texto propio de la noticia.
         </text>
        <plusinfo/>
        Etiqueta almacena información a mayores sobre el texto o noticia.
    </contentnews>
</news>
```

Como se ha mencionado al principio de esta apartado, el sistema está diseñado para facilitar la incorporación de noticias con formatos distintos a la extensión *txt*. Un ejemplo sería la inclusión de noticias en *pdf* y la utilización del API *PDFBox* para la manipulación de las mismas.

10.4. CREACIÓN DE LA ONTOLOGÍA CON OWL Y JENA

A la hora de tomar la decisión de qué estrategia emplear para construir la ontología, se optó por un lenguaje estandarizado y ampliamente usado, OWL. Pero además de elegir un lenguaje se investigó cuál era la manera más óptima de explotar el estándar OWL, se comprobó que lo más utilizado era la API Jena.

OWL es un lenguaje principalmente orientado al desarrollo de aplicaciones para la Web Semántica. Aunque el proyecto aquí presentado poco tiene que ver con la Web Semántica la decisión de utilizar OWL se basa en sus características (descritas en la Memoria en el apartado 20.). OWL es además un estándar basado en XML que aporta flexibilidad y potencialidad. Una vez planteadas las necesidades concretas de almacenamiento y manejo de las ontologías, se consideró que OWL era una buena solución.

Jena es un marco de trabajo desarrollado por *HP Labs* para manipular meta-datos desde una aplicación Java. Aunque existen dos versiones, para el sistema se ha utilizado Jena 2 puesto que es la versión que incluye la API para el manejo de Ontologías y además soporta el lenguaje OWL (que es precisamente lo que se necesitaba).

Se presenta a continuación a grandes rasgos, los métodos empleados tanto para crear como para manipular la ontología. La ontología creada depende del método seleccionado por el usuario, que podrá ser bien a través de los términos extraídos, bien a través de las dependencias.

En primer lugar es necesario crear la cabecera de la ontología, que contendrá tanto la declaración de los espacios de nombres requeridos como el espacio de nombre de la ontología creada. Además se incluye información sobre la propia ontología.

```
OntModel modelo= ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);

Ontology ontologia=modelo.createOntology(ONTnameSpace);
String value="Ontologia sobre "+tema+" a partir de: "+extractoresImplicados;
ontologia.addComment(value,"");

modelo.setNsPrefix(tema, ONTnameSpace);

objEscritura.setProperty("showXmlDeclaration", true);
objEscritura.setProperty("showDoctypeDeclaration", true);
objEscritura.write(modelo, outFichRdo, ONTnameSpace);
```

A través de un objeto *OntModel* se crea una nueva ontología. A través del método *createOntologyModel* se carga una nueva ontología, se le indica como parámetro el lenguaje empleado. Se inicializa un objeto *Ontology* con el espacio de nombres creado para la ontología, este espacio de nombres variará en función del corpus e idioma elegidos por el usuario. Se añade a la ontología un comentario.

Un objeto de tipo *RDFWriter* (objEscritura) se encargará de almacenar los datos en el fichero OWL, representado por una clase *OutputStream* (outFichRdo).

Las clases son los elementos constructivos básicos. Para su representación se utiliza la interfaz *OntClass*. Para crear una clase con Jena se procede de la siguiente manera:

```
OntModel modelo=ModelFactory.createOntologyModel(OntModelSpec.OWL_DL_MEM);
modelo.read(new FileInputStream(pathOnto), ONTnameSpace,"RDF/XML-ABBREV");
…
OntClass clase=modelo.createClass(ONTnameSpace +nombreClase);
…
objEscritura.write(modelo, new FileOutputStream(pathOnto), ONTnameSpace);
```

Se recupera el modelo que contiene la ontología. Se genera una nueva clase a través del método *createClass*. A continuación se escribe en el fichero.

Además de clases, en la ontología también se incluyen propiedades y restricciones. En el caso de la ontología por términos las propiedades creadas con propiedades de tipo dato.

```
DatatypeProperty
propiedadDato=modelo.createDatatypeProperty(ONTnameSpace+"propiedad");
propiedadDato.addDomain(c);
propiedadDato.addRange(XSD.xstring);

MinCardinalityRestriction mcr=m.createMinCardinalityRestriction(null,
propiedadDato, 1);
mcr.setSubClass(c);
```

A través del método *createDatatypeProperty* de la clase *DatatypeProperty* se indica el nombre de la propiedad que se va añadir a la ontología. Los métodos *addDomain* y *addRange* añaden el dominio e imagen de la propiedad respectivamente.

La clase *MinCardinalityRestriction* define una cardinaliad mínima a la cual se le indica que la propiedad, creada previamente, para una clase *c* va a tener un valor de 1.

En el caso de la ontología por dependencias se crean dos tipos de propiedades. Un tipo de propiedad tipo dato que ya se ha descrito para los términos y además un tipo de propiedad tipo objeto. Estas últimas, relacionan instancias de clases. La forma de generarlas es la siguiente:

```
String nbProp=”nb_propiedad”;
ObjectProperty
propiedadObjeto1=modelo.createObjectProperty(ONTnameSpace+nbProp);
OntClass claseDominio=modelo.getOntClass(ONTnameSpace+”claseDominio”);
propiedadObjeto1.addDomain(claseDominio);
OntClass claseRango=modelo.getOntClass(ONTnameSpace+”claseRango”);
propiedadObjeto1.addRange(claseRango);

MinCardinalityRestriction mcr1=
modelo.createMinCardinalityRestriction(null, propiedadObjeto1, 1);
mcr1.setSubClass(claseDominio);
```

En cuanto a la interpretación, se define una propiedad de tipo *ObjectProperty*. Se recuperan las clases que van a ser respectivamente el dominio y el rango de la propiedad y se añaden a la misma como tales. Finalmente a través de la clase *MinCardinalityRestriction* se indica la cardinalidad mínima que tendrá la propiedad y se guarda la clase asignada como dominio.

El último tipo de elemento introducido en la ontología son los individuos.

```
Individual individuo=m.createIndividual(ONTnameSpace+nbClase,nb_individuo);
individuo.addProperty(propiedadDato,nbInd);
…
objEscritura.write(modelo, new FileOutputStream(pathOnto), ONTnameSpace);
```

Una instancia de una clase se crea a través de la clase *Individual*. A través del método *createIndividual* de la clase *OntModel* se indica el espacio de nombre de la clase del individuo así como el nombre del individuo. Se inicializa en el individuo creado las propiedades que tiene asignadas. Y finalmente, se escribe el individuo en el fichero de la ontología.

OntResource, es una clase proporcionada por Jena que permite que todos objetos de una ontología tengan esta super-clase en común. De este modo pueden compartir funcionalidades y proporciona un manejo adecuado de los valores devueltos por métodos generales.

Recuperar los individuos de una clase:

```
ObjectProperty propObjeto=modelLectura.getObjectProperty(ONTnameSpace+pObj);
DatatypeProperty
propData=modelLectura.getDatatypeProperty(ONTnameSpace+pData);
OntClass padre=modelo.getOntClass(ONTnameSpace+termino);
ExtendedIterator stmtI = padre.listInstances();
while (stmtI.hasNext())
{
   OntResource c=(OntResource)stmtI.next();
   if (c.isIndividual())
   {
      Individual r = c.asIndividual();
      salida.add(r.getLocalName());
      NodeIterator nodeI=r.listPropertyValues(propData);
      while(nodeI.hasNext())
      {
               String valor=nodeI.next().toString();
               salida.add(valor);
           }

      NodeIterator nodeI=r.listPropertyValues(propObjeto);
      while(nodeI.hasNext())
      {
               String valor=nodeI.next().toString();
               salida.add(valor);
           }

   }
}
```

Se obtiene la ontología del fichero y se localizan las instancias de la clase implicada a través del método *listInstances*. De cada uno de los individuos se recupera su nombre y el valor de sus propiedades. En este caso, primero se recuperan las propiedades de tipo dato referenciadas por el objeto *propData* de la

clase *DatatypeProperty*. Después se recuperan las propiedades de tipo objeto referenciadas por el objeto *propObjeto* de la calse *ObjectProperty*.

10.5.CREACIÓN Y MANIPULACIÓN DEL ÍNDICE CON LUCENE

10.5.1.CREACIÓN DEL ÍNDICE

El punto de partida consiste en la creación de un índice. Una vez que exista el mismo, se podrán añadir los documentos susceptibles de ser indexados.

```
Directory dir=FSDirectory.getDirectory(pathIndexador);
IndexWriter writer=null;
writer= new IndexWriter(dir, null,true);
writer.close();
```

La clase *IndexWriter* se usa tanto para la creación como para el mantenimiento de un índice. Cuando se crea un objeto de este tipo, al constructor se le pasan tres parámetros. Como en este caso estamos creando el índice, sólo son relevantes el primer y tercer parámetro: el primero representa el directorio donde se almacenará el índice una vez creado; el tercero indica que lo que estamos haciendo es justamente crear un índice y no abriéndolo para su mantenimiento. El método *close()* se llama para liberar todos los recursos asociados a la creación del índice.

10.5.2.AÑADIR UN DOCUMENTO AL ÍNDICE

Una vez que se ha creado un índice es posible añadir documentos al mismo. En primer lugar es necesario crear los documentos para introducir en el índice, esto es, indicarle al índice los campos del documento que se va a indexar y que son relevantes.

```
Document d=new Document();
…
d.add(new Field("idPrivate", doc.getIdPrivate(), Store.YES, Index.NO));
d.add(new Field("tema", doc.getTema(), Store.YES, Index.NO));
d.add(new Field("lang", doc.getIdioma(), Store.YES, Index.NO));
d.add(new Field("contenido", contenido, Store.YES, Index.TOKENIZED));
```

En primer lugar se crea un objeto de tipo *Document*. A través del método add del objeto Document se añaden los campos relevantes para ese documento. En este caso, son cuatro los campos retenidos, *idPrivate, tema, lang* y *contenido*. Como primer parámetro del constructor de la clase *Field* se indica el nombre asociado al campo. En segundo lugar se indica el valor de dicho campo. El tercer parámetro indica si se almacena el campo. Por último, en el cuarto parámetro, se almacena si se desea indexar el valor del campo.

Una vez creado el documento, éste puede ser añadido al índice de la siguiente manera:

```
Directory dir=FSDirectory.getDirectory(pathIndexador);
File stopWordsFile=new File(fichStopW);
Analyzer analyzer=new StopAnalyzer(stopWordsFile);
…

IndexWriter writer=new IndexWriter(dir,analyzer,false);
Document d=this.crearDocumento(doc);
writer.addDocument(d);
writer.optimize();
writer.close();
```

En primer lugar se indica el la ruta donde se ubica el índice. En este caso se utiliza un fichero con palabras que forman la *lista de parada*, por lo tanto el analizador utilizado es un *StopAnalyzer*, al cual se le pasa el archivo con la lista de parada.

A continuación se crea el *IndexWriter* usando el analizador creado previamente. Se indica además el directorio que ubica el índice. El valor booleano al final indica que estamos actualizando el índice, añadiendo un nuevo documento.

Seguidamente se crea el documento, como ya se ha mencionado arriba. Una vez el documento disponible se procede a añadirlo al índice a través del método *add* de la clase *IndexWriter*. Los analizadores preprocesan el texto de entrada convirtiéndolo en una secuencia de segmentos. Estos segmentos se utilizan para añadir un documento a un índice así como en los procesos de búsqueda. Esto es lógico, pues el criterio de búsqueda debe de ser procesado de la misma manera que el contenido de los campos del documento.

10.5.3.BÚSQUEDA DE DOCUMENTOS

Una vez añadidos los documentos al índice, la búsqueda de documentos constituye la funcionalidad principal de Lucene.

```
Directory dir=FSDirectory.getDirectory(pathIndexador);
IndexSearcher searcher=new IndexSearcher(dir);

File stopWordsFile=new File(fichStopW);
Analyzer analyzer=new StopAnalyzer(stopWordsFile);


QueryParser parser=new QueryParser("contenido",analyzer);
Query query=parser.parse(consulta);
Hits hits=is.serach(query, ,Sort.RELEVANCE);
for(int i=0;i<hits.length();i++){
Document doc=hits.doc(i);
```

En primer lugar se localiza el analizador. Éste se pasa como parámetro en una clase *QueryParser* que además necesita el campo a través del cual buscar. A continuación se analizará la consulta devolviendo un objeto de tipo *Query*. A partir de aquí se puede realizar la petición de búsqueda sobre el objeto *IndexSearcher*. Esto devolverá un objeto de tipo *Hits* que será una colección de documentos con el criterio de la búsqueda.

10.6.IMPLEMENTACIÓN DEL PATRÓN ESTRATEGIA

Se detalla a continuación la estructura empleada para el patrón estrategia en la implementación de los documentos añadidos al sistema. Los patrones proporcionados al sistema actualmente son noticias de periódicos y se ha seguido una determinada nomenclatura para recoger las mismas. Como se ha visto en la sección de los patrones de diseño empleados (sección *8)* el uso del patrón *estrategia* permite añadir fácilmente nuevas nomenclaturas.

La estructura de los documentos viene dada mediante el uso de un conjunto de etiquetas que cierran las distintas secciones del documento. Estas secciones pueden ser fecha de publicación, medio de publicación, título, contenido, etc.

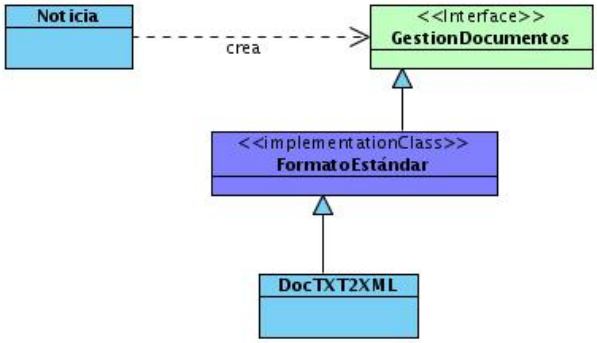


Figura 93 implementación patrón estrategia

Por lo tanto, con el empleo de esta jerarquía de clases se consigue, por un lado que el sistema sea independiente del formato que tenga el documento y por otro lado que sea independiente del tipo de fichero introducido (esto es, de la extensión del archivo). La interfaz *GestionDocumentos* define el comportamiento común que deben implementar sus clases hijas. En la clase abstracta *FormatoEstandar* definimos la nomenclatura actual que tiene los documentos. Finalmente la clase *DocTXT2XML* implementa la extracción del contenido del documento dado con extensión *.txt*.

11.IMPLEMENTACIÓN DE LOS RECURSOS DE PLN

A continuación se muestran las actividades llevadas a cabo para la puesta en funcionamiento de las herramientas de PLN dentro del sistema desarrollado. En este apartado se estudian tanto la estructura de clases que deben mantener las herramientas que se incluyen en la aplicación, como las modificaciones y adaptaciones realizadas para el caso concreto considerado[9].

11.1.IMPLEMENTACIÓN DE LA EXTRACCIÓN

En este apartado se pretende proporcionar al lector una visión general de la tarea de extracción terminológica en el sistema. Se concretizan, además, las adaptaciones sufridas por dos de las herramientas ya descritas en la Memoria de este proyecto.

11.1.1.COMUNICACIÓN CON LOS EXTRACTORES

La adquisición de términos es una tarea que se realiza en el módulo correspondiente de extracción del subsistema de procesamiento. Para la implementación de este módulo se han utilizado dos abstracciones representadas mediante interfaces

[9] Se refiere esto a la aplicación de textos económicos sobre el idioma francés y el idioma español.

(A) INTERFAZ EXTRACTORCLIENTE

La interfaz *ExtractorCliente* proporciona un acceso a los datos generales de los extractores incluidos en el sistema, esto es un acceso al fichero de configuración para recuperar la información necesaria para ejecutar un extractor determinado. Las clases que implementen esta interfaz deben incorporar los siguientes métodos:

rtvResource → En este método se debe recuperar la correspondencia entre una abreviatura y su idioma. Como se ha visto en el fichero de configuración (Sección: *10.2 interpretación del fichero de configuración*) cada idioma se identifica a lo largo del archivo mediante su abreviatura. Es decir, los distintos recursos asignados a una actividad de PLN se recuperan a través de la abreviatura del idioma.

callConcretResource → En este método se aplica una llamada dinámica (en tiempo de ejecución) al extractor requerido por el usuario. Mediante una llamada a la clase que maneja el fichero de configuración se recuperan todos los datos relativos al extractor concreto que se va a lanzar.

Es preciso indicar en este punto que la necesidad de una llamada dinámica es sumamente importante. Esto se debe a que en un principio no se sabe cuantos son los extractores disponibles y, por consiguiente, sólo a través de una ejecución dinámica se podrá abarcar las distintas posibilidades que ofrece el sistema. Este hecho conlleva la necesidad de indicar en el archivo de configuración el paquete y la clase que contiene la lógica del extractor.

DETALLES DE UNA LLAMADA DINÁMICA

```
Class claseDinamicaIdioma = Class.forName(dExtractor.getClase());
Class [ ] paramClase_1={(new String()).getClass(),(new
String()).getClass(),(new String()).getClass(),(new
String()).getClass(),(new String()).getClass()};
Object [ ] paramMetodo_1 =
{dExtractor.getRutaExtractor(),ResultadoTema,path,tema,ficheroConfiguracion};
Constructor constructorDinamico =
claseDinamicaIdioma.getConstructor(paramClase_1);
ExtractorConcreto
ex=(ExtractorConcreto)constructorDinamico.newInstance(paramMetodo_1);
```

Ésta es la llamada dinámica realizada para lanzar un determinado extractor. Se indica brevemente el significado de los métodos implicados.

Class.forName(dExtractor.getClase()) → Recibe como parámetro el nombre completo de la clase, incluido el paquete.

paramClase_1 → Es un array en el cual se deben indicar los tipos de datos de los parámetros que recibe el método que se va a invocar, en este caso el constructor.

paramMetodo_1 → Es un array en el cual se incluyen los parámetros que recibe el método que se va a invocar, en este caso el constructor.

claseDinamicaIdioma.getConstructor(paramClase_1) → Devuelve un objeto de tipo Constructor que referencia al constructor publico de la clase representada por el objeto Class.

constructorDinamico.newInstance(paramMetodo_1) → Crea e inicializa una nueva instancia del constructor de la clase, con los parámetros de inicialización especificados.

Una vez se tiene la instancia de *ExtractorConcreto* inicializada ya se puede acceder a cada uno de los métodos definidos en la interfaz y aplicar los pasos necesarios para la extracción.

(B) INTERFAZ EXTRACTORCONCRETO

La interfaz *ExtractorConcreto* proporciona un acceso a las clases que implementan los extractores que conforman el sistema. Las clases que implementen esta interfaz deben incorporar los siguientes métodos:

setValores → En este método se deben definir los atributos necesarios para la extracción, como por ejemplo el tema tratado.

etiquetador → En este método debe implementarse un procedimiento de etiquetación de un texto, para posteriormente poder utilizarlo con el extractor.

execExtract → Este método debe proporcionar una implementación de una técnica de extracción de términos a partir de un texto dado.

getTerminos → Por último se debe poder recuperar los términos extraídos. Éstos deben guardar un formato concreto para que puedan almacenarse y recuperarse, y así efectuar posteriormente su procesamiento. Este formato consiste en guardar en la misma estructura el término junto al extractor que lo ha generado.

11.1.2.LLAMADA A LOS EXTRACTORES CONCRETOS

Como se ha comentado en la Memoria, en el caso concreto desarrollado se han utilizado un total de 4 extractores; dos para el francés y otros dos para el español. Como estos extractores están implementados en un lenguaje de programación distinto a Java, ha sido necesaria la llamada a un proceso externo. Para realizar esta tarea existe en el JDK 1.5 una serie de clases que permiten el manejo de tales invocaciones.

En la tabla se presenta la llamada realizada al extractor *Fastr_Fr*.

```
ProcessBuilder pb = new
ProcessBuilder("perl","FastrFr_use.pl",rutaFastr,rutaGuardaResultado);
java.lang.Process p=null;
Map<String, String> env = pb.environment();
env.put("DIR_FASTRFR",rutaScript);
pb.directory(new File(rutaScript));
try  {p = pb.start();}
BufferedReader in = new BufferedReader (new InputStreamReader (
p.getInputStream (  ), Charset.forName("ISO-8859-15") )  ) ;
String currentLine = null;
while   (( currentLine = in.readLine (  )  )  != null)
{
   if(!currentLine.matches(""))
  {
       String [] elementos=currentLine.split("\t");
            if(elementos.length==2)
           {
               if((new Integer(elementos[1])).intValue()>2)
               {
                   RdoTerminos rdoTerm=new
RdoTerminos(elementos[0],elementos[1],"fastr_fr");
                   vTerminos.add(rdoTerm);;
               }
            }
    }
}
int valorRetorno=p.waitFor();
```

CLASE PROCESSBUILDER

La clase *ProcessBuilder* es utilizada para crear procesos del sistema operativo. Cada una de las instancias maneja una colección de atributos de proceso. Los atributos utilizados en este caso son el directorio de trabajo y una variable de entorno.

El constructor de esta clase recibe como parámetros todos aquellos valores que un usuario introduciría en una línea de comandos para ejecutar el proceso en cuestión:

```
pb = new
ProcessBuilder("perl","FastrFr_use.pl",rutaFastr,rutaGuardaResultado);
```

Esto se traduciría en la línea de comandos a:

```
sarinha@sarinhapc:~/MultilingualKnowledge$ perl FastrFr_use.pl Extract/frances/Fastr/
TermResult/economia/frances
```

El método *directory* se utiliza para asignar el directorio de trabajo desde donde se lanza el proceso externo.

El método *environment* devuelve una tabla *Map* con el entorno de construcción del proceso. En este caso se introduce en la tabla el nombre de la variable de entorno (DIR_FASTRFR) a la cual se asigna la ruta donde se encuentra el proceso a lanzar.

Finalmente, el método *start* crea un proceso nativo y devuelve una instancia de una subclase de *Process* que puede ser usado para controlar el proceso y obtener información sobre los mismos.

CLASE PROCESS

La clase *Process* provee métodos que dan la entrada de datos al proceso externo, así como facilidades para recoger la salida hacia la clase Java de llamada. Otra de sus funcionalidades consiste en un análisis continuo del estado del proceso para saber cuando éste se ha completado.

El método *waitFor* permite que el hilo lanzado espere hasta que el objeto Process haya finalizado. El método retorna inmediatamente una vez el proceso haya acabado. Mientras no finalice, el hilo permanece esperando.

En este ejemplo puede verse como se ha realizado la recuperación de los datos. Se ha procurado recuperar directamente los resultados de los procesos sin tener que pasar por un estado intermedio de almacenamiento temporal en un fichero.

Esto es así, porque inicialmente los resultados de la extracción se almacenaron en un fichero que posteriormente era procesado para la recuperación de los términos. El empleo de la actual estrategia conlleva una mejora considerable del tiempo de ejecución al no tener que lidiar con aperturas y cierres de archivo.

Cabe señalar que el extractor Acabit, almacena por defecto los términos en un fichero XML, por lo tanto para esta aplicación la recuperación de los términos se hizo mediante el procesamiento del archivo resultado.

11.1.3.ALMACENAMIENTO

Una vez que los términos son extraídos se almacenan en un fichero XML que se ha definido para tal cometido. La extracción puede realizarse sobre distintos extractores a la vez, pero sobre un único idioma simultáneo. De esta manera los resultados muestran los distintos términos junto al extractor (o extractores) que los ha obtenido.

La estructura de ese fichero es la siguiente:

```
<extraccion>
      <idioma lang="frances">
            <termino>
                  <extractor>fastr_fr</extractor>
                  <term>croissance américain</term>
            </termino>
            <termino>
                  <extractor>fastr_fr</extractor>
                  <term>volume de transaction</term>
            </termino>
      </idioma>
</extraccion>
```

El ejemplo es muy intuitivo. En la etiqueta *idioma* se incluye un atributo que indica el idioma sobre él que se ha realizado la extracción. En las etiquetas *termino* se incorpora información sobre el extractor y el término extraído. El caso de un término devuelto por más de un extractor, se reflejará añadiendo tantas marcas *extractor* como fuesen necesarias.

11.2.MODIFICACIONES DE LA EXTRACCIÓN

En este apartado se describen las modificaciones llevadas a cabo para adaptar los extractores empleados a las necesidades concretas. No se detallan las llamadas realizadas, pero ambas herramientas incluyen una llamada previa a su etiquetador correspondiente.

11.2.1.MODIFICACIONES PARA EL EXTRACTOR ACABIT

El extractor Acabit se detalla en la Memoria del proyecto, en el apartado 13. Aquí se exponen los cambios que se tuvieron que realizar en el mismo para poder ser utilizado en la arquitectura.

(A) ENTRADA Y ADAPTACIONES

El extractor en funcionamiento normal funciona con una entrada segmentada y etiquetada por el etiquetador Brill (Eric Brill, Universidad de Pensilvania) y el lematizador Flemm. Concretamente para el francés existe una versión de Brill, WinBrill distribuido por ATILF (*Analyse et traitement informatique de la langue française*). En este caso no fue posible conseguir dicho etiquetador.

Debido a esto y dado que Flemm trabaja también con el etiquetador TreeTagger, se optó por la adaptación de las salidas de Flemm al formato de entrada de Acabit.

El fichero de entrada del extractor debe estar marcado y formateado de la siguiente manera:

```
<record>
<AN>…</AN>
<TI>…</TI>
<AB>
       <ph_nb=xxx>frase</ph>
       …
       <ph_nb=xxx>frase</ph>
</AB>
</record>
```

Donde:

<*record*>: marca los diferentes textos.
<*AN*>: marca información del texto.
<*TI*>: marca el título del texto.
<*AB*>: marca el cuerpo del texto.
<*ph_nb*=***xxx***>: marca las distintas frases del texto.

En el desarrollo del proyecto se ha prescindido de la información referente a las marcas <*AN*> y <*TI*>.

Como se comenta en la Memoria, apartado 11, la ejecución de Flemm con el etiquetador TreeTagger devuelve un fichero XML. Se ha generado un conversor en PERL para analizar este fichero y obtener la información morfológica que da entrada al extractor. Cada una de las frases del corpus se descompone en palabras a las que se asocia información sobre su forma, lema y otros rasgos morfológicos. Dependiendo de la categoría léxica de la palabra, los datos morfológicos retenidos varían.

Así, para los **adjetivos**, **artículos** y **sustantivos** se almacena información morfológica sobre la categoría, el género y número. Par los **verbos** se guarda la categoría, la persona, el número y el modo. Para el resto de categorías, se guarda la etiqueta morfológica. A continuación se muestra un ejemplo para cada caso.

		INFORMACIÓN MORFOLÓGICA							
Forma	Tipo	Cat.	gº	nº	Modo	Pers	nº_v	Lema	Salida
récente	adjetivo	ADJ	f	s	------	-----	-----	récent	récente/ADJ:f:s/récent
est	verbo	VER :pres	--	--	i	3	s	être	est/VER :pres:i:3:s/être
de	preposición	PRP	--	--	------	-----	-----	de	de/PRP/de

Los elementos que se agruparan entre las etiquetas <*ph_nb=xxx*> serán cada una de las palabras que forman una frase, pero con el formato mostrado en el campo *salida* de la tabla anterior.

Además de implementar el conversor, ha sido necesario retocar el propio código fuente para adaptar la gramática del extractor a las etiquetas de TreeTagger

(B) ESQUEMA DE FUNCIONAMIENTO

El gráfico siguiente muestra el flujo de datos de Acabit una vez realizados los cambios explicados

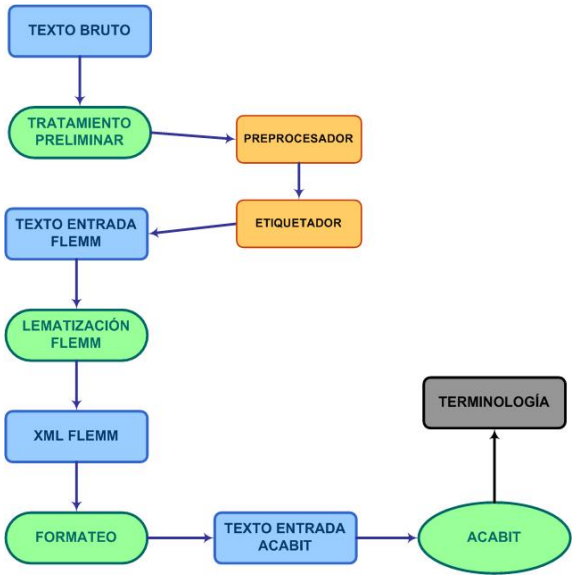


Figura 94 esquema funcionamiento acabit

11.2.2.MODIFICACIONES PARA EL EXTRACTOR FASTER

(A) ENTRADA Y ADAPTACIONES

En el caso del francés la única modificación ha consistido en pasarle de entrada el texto ya etiquetado con TreeTagger[10].

En el caso del español las modificaciones van un poco más allá. En los propios ficheros fuente no se ha realizado ningún cambio, pero si en el formato de entrada de los mismos. Así, las etiquetas proporcionadas por TreeTagger son diferentes para cada uno de los idiomas tratados. Mediante el uso de un fichero DMB se ha sustituido la etiqueta del español por su correspondiente etiqueta del francés. Se ha procedido de tal manera, considerando que las estructuras gramaticales del francés y del español son similares y, por lo tanto, se pueden aplicar las mismas reglas para la detección de los términos

(B) SALIDA PROPORCIONADA

La salida del extractor se ha almacenado en un fichero *txt* que posteriormente se ha recorrido y analizado para recoger solamente la información referente a los términos. La herramienta analiza los términos frase a frase, de ahí que la salida proporciona para cada frase los términos extraídos y sus posibles variantes. Las frases sin término se imprimen sin otro tipo de procesado. Así en primer lugar aparece el término y a continuación su variante junto con el tipo de meta-regla aplicada. En caso de que el término no proceda de ninguna variante, en el lugar donde aparece la variante se muestra el propio término y en el tipo de meta-regla se sitúa un 0.

```
Amorcé en 1945  avec la création du système monétaire international  conforté
en 1971  avec la mise à mort de ce même système  le " privilège exorbitant "
que se sont arrogé les Etats-Unis de pouvoir financer la croissance de leur
économie aux frais du reste du monde n a jamais été en danger jusqu à présent.
000127 système monétaire    système monétaire
000127 système international      système monétaire international XX,4,Modif

Depuis le début de l' année 2003  l' euro a gagné 17% par rapport au dollar .
000048 dollar par rapport   rapport au dollar  XXX,27,NtoN

Inefficaces  voire contre-productives à très court terme  les interventions
```

[10] En funcionamiento normal, Fastr llama directamente el etiquetador.

```
seraient  en revanche  susceptibles d inverser à moyen et long termes les
tendances  à condition de respecter ces trois critères.
000123 moyen terme   moyen et long termes     XX,23,Coor
```

En el momento de analizar el fichero de salida, se recogen únicamente los términos y no las variantes.

11.3. IMPLEMENTACIÓN DEL ANÁLISIS SINTÁCTICO

En este apartado se pretende proporcionar al lector una visión de cómo se lleva a cabo la tarea de análisis sintáctico en el sistema. Se detallan asimismo las adaptaciones sufridas por la herramienta descrita en la Memoria de este proyecto, en el apartado 9, para poder aplicarla en los dos idiomas utilizados en el caso concreto.

11.3.1. COMUNICACIÓN CON LOS ANALIZADORES

El análisis sintáctico es una tarea que se realiza en el módulo correspondiente de análisis en el subsistema de procesamiento. Para la implementación de este módulo se han utilizado dos abstracciones representadas mediante interfaces.

(A) INTERFAZ ANALIZADORCLIENTE

La interfaz *AnalizadorCliente* proporciona un acceso a los datos generales de los analizadores sintácticos incluidos en el sistema, esto es, un acceso al fichero de configuración para recuperar la información necesaria para ejecutar un analizado determinado. Las clases que implementen esta interfaz deben incorporar los siguientes métodos:

rtvResource → Con este método, al igual que en el caso de la extracción, se recupera la correspondencia entre un idioma y su abreviatura. De esta manera se recorren las secciones del fichero de configuración accediendo a los recursos e información necesaria para la puesta en marcha del análisis.

callConcretResource → A través de una llamada dinámica este método realiza las operaciones necesarias para llamar a la clase del analizador que se va a ejecutar en ese momento. Como ocurre en la extracción de términos, la invocación debe ser dinámica, pues se desconocen a priori los analizadores disponibles.

setNews → A través de este método se recuperan los documentos con las cuales se van a realizar el análisis sintáctico. Examinando el XML de cada documento se toma para el análisis el título del mismo, así como el texto.

execParser → Uno de los requisitos necesarios para el análisis sintáctico es situar cada una de las frases de los textos de los documentos en un único fichero. Este método se encarga de tal tarea. En el apartado siguiente se detalla el proceso de almacenamiento de los resultados del análisis sintáctico.

(B) INTERFAZ ANALIZADORCONCRETO

La interfaz *AnalizadorConcreto* proporciona el punto de entrada a cada una de las clases definidas en el sistema para la ejecución del análisis sintáctico. Los métodos definidos en esta interfaz son los que debe implementar cualquier nuevo analizador añadido al sistema:

setValores → En este método se deben definir los atributos necesarios para el análisis sintáctico, como por ejemplo el tema tratado.

startParser → Con este método se lanza el análisis sintáctico. Los métodos detallados a continuación permiten el almacenamiento de la información proporcionada por el análisis.

getVectorDependency → En primer lugar hay que señalar que el análisis sintáctico debe devolver los resultados en forma de pares. Existen algunas herramientas de PLN ya implementadas que, por ejemplo, generan el análisis a través de un grafo de dependencias. Será entonces tarea del usuario extraer las dependencias que considere oportunas en forma de pares. Una vez los resultados en forma de pares, éstos deben devolverse como un objeto que contiene atributos para almacenar la etiqueta asignada a esa dependencia, así como el lema y posición de cada palabra en la dependencia.

getVectorTagger → Este método es necesario para recoger los otros datos relevantes del análisis: la categoría, la forma y el lema. Para todas las palabras que aparecen en el corpus se debe guardar su correspondiente información morfológica.

getSentence → Por último, el sistema necesita recoger la frase que se está analizando en cada instante.

Al igual que para la extracción se lanzó un proceso externo a través de las facilidades que proporciona el propio JDK. Se señala además que también en este caso se ha prescindido de un paso intermedio a un fichero temporal, enviando los resultados del análisis directamente al proceso Java. Como ya se ha especificado cómo se realizan estas llamadas para el caso de la extracción de términos, no se entra aquí en más detalles.

11.3.2. ALMACENAMIENTO DEL ANÁLISIS SINTÁCTICO

El proceso de análisis sintáctico se realiza sobre un idioma y sobre un analizador a la vez. La técnica utilizada para la extracción de conocimiento necesita que los resultados del análisis se almacenen de una forma muy concreta.

Actualmente esta técnica se apoya en una base de datos para realizar todas sus tareas. Por consiguiente el análisis sintáctico se ha almacenado en una estructura de este tipo.

Para facilitar el mantenimiento y depuración de errores, el análisis debe realizarse sobre una frase a la vez. De manera que cada resultado devuelto por el análisis está totalmente localizado en el conjunto de documentos, pues se utiliza una estructura de directorios bien definida para guardar las frases y los documentos:

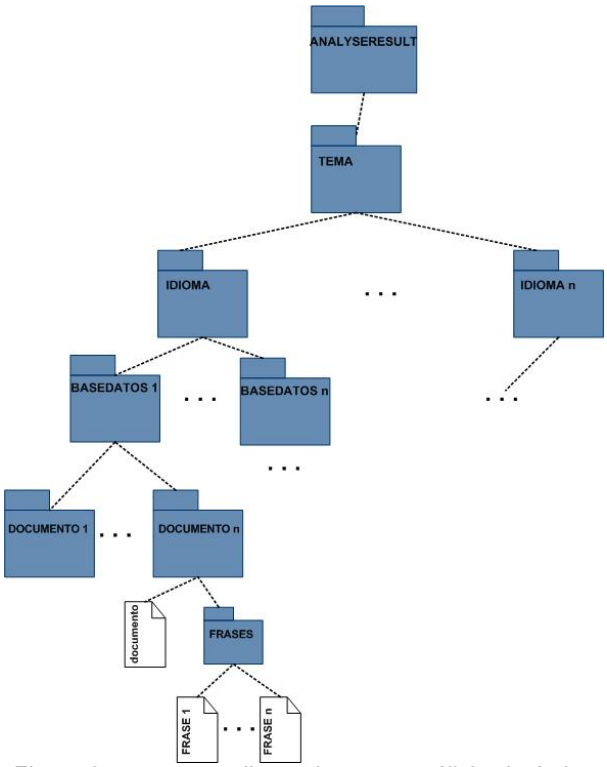


Figura 95 estructura directorios para análisis sintáctico

Para cada tema se guardan los idiomas sobre los que se realiza el análisis. Además, una vez dentro de un tema, el análisis podría ser lanzado para distintos gestores de bases de datos[11]. En este caso como se ha descrito en el apartado del fichero de configuración (Sección: *10.2 interpretación del fichero de configuración*), cada base de datos debe tener un nombre distinto.

Como se puede ver en la Figura 95, una vez situados dentro de un tema, un idioma y una base de datos, cada documento se sitúa en un directorio que a su vez contiene un fichero por cada una de las frases del documento. Estas frases tienen un identificador único que también se almacena en la base de datos.

El resto de información que debe guardarse es la que proporcionan los métodos *getVectorDependency getVectorTagger getSentence* . No se detalla más en este punto ya que la descripción de la base de datos se ha realizado en el apartado 9, donde se especifica el diagrama de Entidad/Relación.

11.4.MODIFICACIONES DEL ANÁLISIS SINTÁCTICO

11.4.1.MODIFICACIONES EN EL ANALIZADOR ERIAL

Como ya se ha introducido, el analizador sintáctico se ha utilizado para los dos idiomas manejados en la arquitectura de este proyecto. Este hecho se debe principalmente a la dificultad asociada a la disponibilidad de un analizador sintáctico libremente disponible para el francés. En la mayoría de los casos existen herramientas disponibles, pero a las cuales es necesario proporcionar una gramática del francés.

La decisión de usar Erial para el francés se toma previo análisis de la implementación del mismo. Se parte de la base de que ambos idiomas tienen una estructura gramatical muy similar, así las construcciones manejadas en el analizador sintáctico pueden ser adaptadas a este idioma.

[11] En estos momentos la técnica de extracción de conocimiento se aplica sobre un gestor MySQL.

(A) MODIFICACIONES EN LA ENTRADA

En el apartado 9 de la Memoria, se ha descrito el funcionamiento del analizador sintáctico en un modo normal. Para poder utilizar el mismo y beneficiarse de sus capacidades se han tenido que llevar a cabo una serie de modificaciones.

En primer lugar el análisis se llevo a cabo frase a frase, facilitándole la tarea al analizador a la hora de delimitar las oraciones. Cada frase del corpus se almacenó en un fichero y se extrajo para su análisis por parte de Erial. El pre-procesador facilitó esta tarea al situar cada frase del corpus en una línea. Esto se hizo así para tener localizada en cada momento la procedencia de una determinada dependencia, y facilitar el manejo de la base de datos (lugar donde se almacenan los resultados del análisis sintáctico).

Cada frase se proporciona etiquetada y lematizada con TreeTagger. Se comenta en la Memoria que la entrada estándar a Erial era MrTagoo. Se podía haber optado por el uso de este etiquetador-lematizador, lo que habría evitado la adaptación de la salida de TreeTagger a la entrada de Erial. Sin embargo, al tratar con un dominio concreto como es la economía muchas palabras del corpus iban a ser desconocidas para MrTaggo. Este hecho ya estaba superado en las modificaciones que se habían realizado en TreeTagger previamente.

Las modificaciones se han llevado a cabo en dos sentidos, por un lado para el español y por otro lado para el francés.

Modificaciones para el español

Sustitución de las cifras: en TreeTagger las cifras dadas de la siguiente manera:

[0-9] card [0-9] ò valor card @card@

se transforman en la terna*: Forma Cifra Forma*

Por ejemplo: 9 card 9 → 9 cifra 9 ó nueve card @card@ → nueve cifra nueve

Para el resto de las cifras que funcionan de determinante, TreeTagger detecta correctamente la categoría, y la transición hacia Erial implica un cambio en la etiqueta.

Tratamiento de las contracciones: en TreeTagger las contracciones se consideran como una única entidad mientras que en Erial es necesario separar sus componentes para realizar el análisis. En este paso pues, se creó un fichero con las correspondientes contracciones del castellano y un programa en PERL que toma de entrada una terna formada por una contracción; y devuelve dos ternas una con la información correspondiente a la preposición y otra con la información correspondiente al determinante.

Paso de las etiquetas de TreeTagger a Erial: A través de un programa PERL que accede a un fichero DBM que almacena una tabla de indexación. Esta tabla de indexación contiene las correspondientes etiquetas del TreeTagger como elementos clave y las etiquetas de Erial como elementos valor.

Tratamiento de los porcentajes: Por defecto la versión de TreeTagger para el español trata los número seguidos de porcentaje como dos elementos distintos. Sin embargo, Erial trata esos elementos como una unidad, por lo tanto se ha procedido a juntar en un único elemento este tipo de ocurrencias.

Modificaciones para el francés

Las modificaciones hechas para el francés son las mismas que para el español salvo en el tratamiento de porcentajes, puesto que la salida de TreeTagger en modo francés ya devuelve el número y su porcentaje, como un único elemento

(B) MODIFICACIONES EN EL ANALIZADOR

En primer lugar y dada la estructura de la base de datos es necesario almacenar para cada lema de las dependencias su posición dentro de la frase. Por lo tanto, en el pre-procesamiento que incorpora Erial realizado en la etapa previa al análisis, se añade a cada lema su posición con la marca "~" seguida del número de posición. En consecuencia, ha sido necesario modificar la mayoría de las expresiones regulares que modelan las reglas de reconocimiento de funciones sintácticas.

En segundo lugar, cada una de las expresiones regulares ha tenido que ser adaptada al francés previa consulta de una gramática del idioma. Como es lógico se han utilizado ficheros distintos para el francés y el español. Esto fue posible partiendo de la base de que las expresiones regulares originales tienen una estructura muy similar en ambas lenguas.

Por último, se ha añadido a las dependencias del tipo *Complemento Nominal* y *Complemento Circunstancial* (*CN* y *CC*, respectivamente) la preposición que las precede. Esto se hizo por ser importante el tipo de preposición a la hora de extraer automáticamente una ontología, pues el tipo de relación entre dos sustantivos varía en función de la preposición que los une.

A continuación se muestra un ejemplo del análisis devuelto para una frase:

```
"La caída del dólar se ha acelerado durante los últimos días y la moneda
norteamericana cayó el día 12 de diciembre

SA      (día~11 NCMS , último~10 AQMS)
SA      (moneda~14 NCMS , norteamericano~15 AQMS)
CNde    (caída~1 NCMS , dólar~4 NCMS)
SUJA    (acelerar~7 V, caída~1 NCMS)
OD      (acelerar~7 V , moneda~14 NCMS)
CCdurante       (acelerar~7 V1SRI , día~11 NCMS)
OD      (caer~16 V1SRI , día~18 NCMS)
```

11.5. IMPLEMENTACIÓN DEL ETIQUETADOR

En esta sección se detallan las adaptaciones llevadas a cabo para adecuar el etiquetador al caso concreto.

En la arquitectura diseñada no se ha incluido el hecho de poder añadir un etiquetador, pues se ha considerado que cada nueva herramienta incorporada al sistema debe incluir toda aquella información que precise, así como aquellos recursos necesarios. Es el caso por ejemplo de Acabit, como se puede ver en la Memoria, utiliza el lematizador Flemm para ejecutar su tarea.

Sin embargo, se ha optado por describir el etiquetador empleado. Esto se debe a que en el caso concreto implementado se ha utilizado el mismo etiquetador para todas las herramientas. Es decir, Acabit, Fastr y Erial se han basado en el etiquetador TreeTagger.

11.5.1.MODIFICACIONES EN EL ETIQUETADOR

Para el sistema desarrollado, el uso del etiquetador ha sido fundamental. A la hora de buscar herramientas que proporcionen las funcionalidades requeridas para este proyecto es importante analizar todos los recursos que las propias herramientas necesitan.

Aunque podría ser interesante empezar buscando un etiquetador y a continuación un extractor, esa no es la lógica correcta. Puesto que un etiquetador eventualmente se puede utilizar en más de un sistema de extracción, un extractor probablemente sólo funcionará con un etiquetador (a lo sumo con dos). La aproximación en este caso sugiere un enfoque descendente, empezando por encontrar un extractor (situado en un nivel más alto de abstracción) y a continuación el extractor pertinente.

Todos los recursos concretos empleados para el proyecto funcionan con TreeTagger. Como se detallará más adelante, algunos ya incorporaban el uso del etiquetador y otros han tenido que ser adaptados para utilizarlo. Además, visto el domino concreto tratado en el desarrollo ha sido necesario modificar el propio TreeTagger para adecuarlo al entorno.

Los idiomas tratados, como ya se ha comentado son el francés y el español. TreeTagger proporciona dos ficheros distintos para cada uno de los idiomas; *tree-tagger-french* y *tree-tagger-spanish*. Estos ficheros son los que se han modificado. En funcionamiento normal cada uno de ellos hace una llamada al programa *tree-tagger* pasándole este conjunto de parámetros:

- *token* : para imprimir la forma de la palabra analizada.

- *lemma*: para imprimir el lema de la palabra analizada.

- *smgl*: para omitir el análisis de etiquetas sgml que pudiesen estar presentes en el texto.

A éstos se han añadido dos parámetros, - *no-unknown* y - *pt-with-lemma*. El primero de ellos asigna como lema de una palabra desconocida la forma. El segundo indica que existe un pre-procesamiento de las palabras a etiquetar. Este pre-procesamiento consiste en ejecutar dos programas.

El primero une los grupos de palabras que designan en conjunto una categoría, es decir cumplen la misma función sintáctica que alguna palabra y por lo tanto podría sustituirse el grupo por dicha palabra. Se puede decir que los componentes del conjunto han perdido su autonomía semántica. Un ejemplo de este hecho sería "*a mediados de"* que tiene significado en conjunto, concretamente es una locución adverbial. Es importante tratar este tipo de ocurrencias como un todo. Las ocurrencias que el programa debe mantener unidas están almacenadas en un fichero, llamado *idioma-mwls.txt*[12]

El segundo programa, permite asignar a una palabra desconocida sus posibles etiquetas junto con sus lemas. En este caso también la información asociada a estas palabras desconocidas está almacenada en un fichero con una estructura bien definida, llamado *idioma-lexicon.txt*

Este hecho ha sido crucial, pues el lexicón no está orientado a un dominio específico, sin embargo el corpus tratado en el sistema es un corpus económico. Consecuentemente el número de palabras que el etiquetador desconocía era importante. Debido a este hecho y ante la carencia de recursos para poder entrenar el etiquetador se optó por introducir todos esos ítems desconocidos a mano, así como los grupos de palabras con significado conjunto.

En una actividad previa a la etiquetación, se aplica el pre-procesador (detallado en la sección 8 y 10 de la Memoria)al corpus. De esta manera las palabras y espacios están correctamente segmentados y se evitan errores de este tipo introducidos por TreeTagger. El etiquetador pues, analiza en un primer momento y a través de un programa PERL el fichero *idioma-mwls.txt* para unir en un único item aquella serie de palabras con significado global. A continuación repasa el fichero *idioma-lexicon.txt* para asignar la etiqueta adecuada a las formas desconocidas. En cualquier caso si la palabra (o conjunto de palabras) ya existe en el lexicón propio no se podrá introducir nuevamente en estos ficheros. Se nombra esto aquí debido a ciertas inconsistencias de las categorías devueltas por la herramienta que no han podido solucionarse

[12] Hay que sustituir *idioma* bien por *french* bien por *spanish,* según el idioma tratado.

12.IMPLEMENTACIÓN DE LA ONTOLOGIA POR TÉRMINOS

En este apartado se describe el proceso de generación de la ontología por términos. En un primer momento se describe la tarea de clasificación de los términos y en un segundo lugar se explica cómo a partir de esa clasificación se crea la ontología.

12.1.CLASIFICACIÓN POR TÉRMINOS

Esta sección describe la puesta en marcha de un prototipo rápido de clasificación de terminología. La extracción de términos proporciona una lista de términos candidatos sin ningún tipo de agrupación entre ellos. Es, por lo tanto, necesaria una ordenación de los mismos.

Se trata de explotar técnicas lingüísticas más que estadísticas. La elección de una organización u otra depende de un factor primordial, la aplicación meta en la que se va a aplicar la estructura. Por lo tanto, se trata de desarrollar una técnica lo suficientemente genérica para poderse aplicar en varios dominios.

La estrategia presentada aquí enlaza, por lo tanto, una primera etapa de extracción de términos para a continuación aplicar una metodología que aproxime éstos y los agrupe según su similaridad.

12.1.1.MÉTODO

El método se asienta en el hecho de que los términos situados en los mismos contextos son términos similares o incluidos dentro de la misma familia de términos. La idea parte de utilizar los contextos en los cuales se usan los términos, y poner en marcha un método que permita compararlos con el fin de poder aproximar aquellos términos cuyo contexto es similar.

Llegados a este punto es necesario definir dos relaciones utilizadas a lo largo del proceso, y que son la relación de cabeza y la de expansión. La *cabeza* de un término se define cómo el núcleo del mismo y la *expansión* como el resto del término que no es la cabeza, a modo de aclaración:

" sociedad de gestión "

sociedad → cabeza

de gestión → expansión

La clasificación es un proceso de edificación de una jerarquía de clases. El agrupamiento de objetos (en este caso los términos) en clases, permite detectar ciertas propiedades comunes (los contextos). Se pretende, por lo tanto, proporcionar conjuntos de términos que se refieran al mismo dominio.

12.1.2.PROCESO

La aproximación seguida es una clasificación no supervisada, pues no se supone la existencia de ejemplos. Trabaja comparando los objetos a organizar procurando unirlos por similaridad.

El método más empleado para clasificar se conoce con el nombre de *clasificación automática*. En cada iteración se trata de unir dos grupos. El proceso finaliza cuando ya no existe ninguna similaridad entre los grupos.

Inicialmente se toma la lista de todos los términos extraídos. Los cálculos se realizan partiendo de todos aquellas palabras que son cabeza de un término y además sustantivo, nombre propio o acrónimo. Se toma esta decisión considerando que los sustantivos, nombres propios y acrónimos son dentro de los términos extraídos los elementos que mayor información semántica aportan al proceso [15]. Se llamarán a partir de este momento *elementos principales*.

Para cada uno de los elementos principales se guarda sus contextos, tanto aquellos en los que aparece como *cabeza*, como en los que aparece como *expansión*.

Se lanza el procedimiento iterativo que aplicará una medida de similaridad entre todos los elementos principales uniendo en cada iteración sólo dos elementos principales. La medida de similaridad se calcula sobre los contextos. Se finaliza cuando ya ninguno es similar a otro.

12.1.3.MEDIDAS DE SIMILARIDAD

La similaridad también se conoce con el nombre de distancia entre dos objetos [16]. Se define por el número de propiedades que tienen en común las entidades implicadas.

Existen distintas medidas de similaridad que se pueden aplicar. Algunos autores utilizan El *coeficiente prox*, el *coeficiente j* o el *coeficiente Dice*. En ciertos casos aplican una combinación de los mismos. En este caso se ha empleado el *coeficiente Dice*:

$$Dice(c1,c2)=|c1 \Omega c2| \ / \ [(|c1|+|c2|)/2]$$

El *coeficiente Dice* mide el número de contextos comunes entre dos grupos de términos normalizando el resultado por el tamaño medio de los contextos. Para calcular el número de contextos comunes se aplica el *coeficiente a*. Éste coeficiente realiza la intersección entre los contextos de dos términos quedándose únicamente con el número de contextos iguales.

12.1.4.ALMACENAMIENTO DEL RESULTADO

Una vez finalizado el proceso se almacenan los resultados en un fichero XML. Bajo la etiqueta *grupo* guardamos aquellos elementos principales que se han recuperado como similares. Estos elementos principales se localizan en el subárbol delimitado por la etiqueta *clase*. Dentro de dicha etiqueta se sitúan los términos organizados según establezcan una relación de cabeza o de expansión con el elemento principal. Se guarda también información morfológica de cada uno de los términos a través del atributo *tipo*.

```
<jerarquia>
<grupo>
        <clase>
            <eltoPpal>société</eltoPpal>
            <cabeza>
                <term tipo="NA">société américain</term>
                <term tipo="NA">société général</term>
                <term tipo="NPN">société de gestion</term>
                <term tipo="NPN">société de logistique</term>
            </cabeza>
            <expansion>
                <term tipo="NPN">résultat du société</term>
                <term tipo="NPN">résultat de société</term>
                <term tipo="NPN">action de société</term>
            </expansion>
        </clase>
        <clase>
            <eltoPpal>entreprise</eltoPpal>
            <cabeza>
                <term tipo="NA">entreprise américain</term>
            </cabeza>
            <expansion>
                <term tipo="NPN">défaut de entreprise</term>
                <term tipo="NPN">résultat du entreprise</term>
                <term tipo="NPN">résultat de entreprise</term>
            </expansion>
        </clase>
    </grupo>
</jerarquia>
```

12.2. CREACIÓN DE LA ONTOLOGÍA POR TÉRMINOS

Una vez expuestos cómo se han generados los distintos componentes que conforman la ontología, se detalla en este apartado y en el apartado siguiente la estrategia utilizada para generar las distintas ontologías según el tipo de clasificación elegida (por términos o por dependencias).

A la hora de crear la ontología por términos se analiza el fichero de clasificación previamente creado. La estrategia empleada para clasificar los términos así como la estructura del fichero se detallan en el apartado anterior. Concretamente en el punto *12.1.4* se explica cómo se almacena la clasificación.

La ontología creada para los términos se ideó de la siguiente manera:

Los términos están agrupados según su palabra clave (o cabecera), así todos los términos que coincidan en su palabra clave estarán juntos. A modo de ejemplo:

recuperación norteamericana

recuperación económica

recuperación de actividad

Estos tres términos estarán agrupados bajo el mismo nodo en la clasificación, ese nodo se denota con el nombre de *clase* y cuyo nombre será *resultados*.

Además, cada una de estas clases está a su vez incluida en un nodo situado en otro nodo cuyo nombre es *grupo* y puede, por lo tanto, considerarse el padre de la clase.

Teniendo en cuenta esto, la ontología se ha creado generando en primer lugar todos los conceptos llamados *grupo*. Y bajo cada uno de estos grupos se crearon las subclases correspondientes a aquellos términos incluidos en los nodos *clase*. Estas subclases recibieron el nombre de la cabecera de los términos a los que se añadió un prefijo *_class*. Para todas estas clases creadas se definió una propiedad común llamada *termino*.

Para almacenar los términos en primer lugar se crearon instancias de las clases. Se nombraron las instancias con el mismo nombre que las clases (esta vez sin el prefijo *_class*). Y cada una de estas instancias fue completada añadiendo tantas *termino* como términos hubiese para esa clase.

<recuperacion_class rdf_ID="recuperación">
<termino>recuperación norteamericano</termino>
<termino>recuperación económico</termino>
<termino>recuperación de actividad</termino>
</ recuperacion_class>

A la hora de recuperar elementos y ampliar la consulta se procede de la siguiente manera:

Se examina si la consulta introducida por el usuario existe en la ontología en forma de *clase*.

Si existe en forma de clase en primer lugar se recupera el padre de esa clase, es decir el *grupo* al que pertenece. Una vez se tiene el *grupo* ya se pueden recuperar todas las subclases pertenecientes al mismo. Se obtienen así los hermanos de la clase detectada con la consulta del usuario. Las instancias de esos hermanos junto con las instancias de la clase se devuelven como consulta ampliada.

13.IMPLEMENTACIÓN DE LA ONTOLOGÍA POR DEPENDENCIAS

Se describe en este apartado como se aplica en el proyecto un algoritmo que es capaz de extraer el conocimiento a partir de un conjunto de textos. El algoritmo actúa sobre la base de datos, actualizando la misma. Posteriormente se toman los datos generados en la base de datos para estructurar esa información en una ontología.

13.1.APLICACIÓN DE UN ALGORITMO DE ERROR-MINING DE EXTRACCIÓN DE CONOCIMIENTO

La cantidad de información textual disponible en formato digital lleva asociada una problemática, ¿cuál es la mejor forma de explotar esa información?, ¿de qué manera es posible extraer el conocimiento encerrado en todas estas fuentes documentales? En este apartado se presenta una técnica que pretende aportar una solución a esta disyuntiva.

El estado del arte actual distingue dos corrientes a la hora de suplir el problema planteado[17]:

El primer enfoque extrae conocimiento mediante la aplicación de técnicas estadísticas. A partir de un conjunto de categorías léxicas asignadas a las palabras del texto a tratar, se calcula la frecuencia de aparición de las mismas. Con esos resultados se genera un grafo de términos relevantes en ese dominio.

El segundo enfoque explota el texto mediante técnicas de análisis sintáctico que intentan detectar relaciones semánticas. Así, se combinan técnicas de extracción de términos con técnicas de agrupamiento de los mismos. Como resultado se obtiene un grafo que refleja las relaciones entre los términos.

En la técnica presentada aquí, se analizan los términos[18] y dependencias existentes entre ellos, presentes en los textos, aplicándoles un conjunto de medidas estadísticas. Las dependencias se generan con la herramienta de análisis sintáctico previamente presentada, *Erial*. El agrupamiento de los términos se lleva a cabo con la aplicación de un algoritmo detallado a continuación

13.1.1.PROBLEMÁTICA SUBYACENTE

La estrategia que se presenta parte de la base de que dado un dominio concreto, las palabras suelen tener un único significado en ese dominio. Siguiendo al hipótesis de Harris [19] se busca trabajar en un entorno en el cual las palabras tengan un significado claramente diferenciado. Por ejemplo, en un dominio económico la palabra *banco* está perfectamente delimitada, siendo su significado más probable él de entidad *financiera*.

En la fase previa a la aplicación del algoritmo es necesaria la extracción de las dependencias de palabras, relevantes en el dominio. Como se ha comentado anteriormente esta tarea se lleva a cabo con la utilización de un analizador sintáctico que devuelve pares de palabras que se relacionan en las frases del texto.

Los analizadores sintácticos trabajan con información léxica. En la herramienta empleada para el análisis sintáctico, se emplea un etiquetador-lematizador (en la Memoria: *6. Etiquetación:TreeTagger*)para aportar la información sobre las categorías léxicas y los lemas de las palabras.

Sin embargo, tanto a nivel léxico como a nivel sintáctico pueden surgir una serie de ambigüedades que es necesario solventar:

Ambigüedades léxicas. Se pueden dividir en dos tipos:

Ambigüedad a nivel de categoría: El etiquetador podría devolver en ciertos casos más de una categoría para una determinada palabra.

Ambigüedades a nivel de lema: El lematizador empleado podría devolver más de un lema para una determinada forma, por ejemplo la palabra *deudas* tiene asociado los lemas *deudo* y *deuda*.

Ambigüedades sintácticas: Se podrían dar casos en los que no está claro de que palabra depende otra. El ejemplo típico, en estos casos, es la frase:

> Juan vio a un hombre con un telescopio
> ¿de quién depende un telescopio? ¿de Juan? o ¿de un hombre?

El algoritmo empleado intenta eliminar este tipo de ambigüedades aplicando la siguiente restricción: En una determinada frase cualquier palabra sólo puede tener una palabra *gobernante*, sin embargo una palabra *gobernante* puede tener más de un *gobernado*. A partir de un corpus, y dado que estamos hablando de un dominio muy concreto, se supone que va ser posible la desambiguación. Esto se consigue examinando el total de corpus para detectar en cuantas ocasiones una palabra es gobernada por uno u otro elemento.

Los analizadores sintácticos sobre lenguajes naturales abordan el problema de las ambigüedades léxicas mediante la predicción de la correcta categoría a través de la gramática base, bien relegando la fase de decisión al etiquetador usualmente siguiendo un modelo estadístico basado en Modelos Ocultos de Markov (Hidden Markov Models). En cualquier caso, el acercamiento más usual consiste en asumir en la etapa de análisis léxico o sintáctico, la preeminencia de una interpretación sobre otra.

La consecuencia inmediata en la aplicación de esta estrategia es que dado que sólo el usuario puede estar en disposición de conocer exactamente la intención de su mensaje, los errores en la hipótesis sobre las que se eliminan ambigüedades léxicas no sólo son habituales, sino que poseen un efecto multiplicado a nivel sintáctico y semántico; llegando a imposibilitar el proceso de análisis.

Otra clase de analizadores, como es el caso de DyALog [20], optan por aplicar técnicas de programación dinámica que no sólo permiten trabajar en paralelo con todas las interpretaciones sino que compactan los cálculos de manera que es posible en la práctica llegar a la fase de análisis semántico sin descartar ninguna posibilidad a dicha interpretación. Ello permite generar una estructura semántica general, sobre la que aplicar sin restricciones previas procesos de adquisición del conocimiento que, grosso modo, suponen un filtrado de interpretaciones inviables o inadecuadas.

13.1.2.ADQUISICIÓN DE CONOCIMIENTO

A continuación se describe el proceso que lleva a cabo la adquisición del conocimiento. La base del procedimiento se encuentra en examinar los valores a un nivel local, es decir a nivel de frase, así como a un nivel global, es decir a nivel de corpus. Combinando ambos enfoques se obtiene de forma bastante certera una clasificación del conocimiento [17].

(A) DESAMBIGUACIÓN LÉXICA

Se calcula la probabilidad de que una determinada palabra tenga asociada bien un determinado lema, bien una determinada categoría. Se examinan los valores obtenidos a nivel de corpus, así como a nivel de frase actual. Se ilustra esto con un ejemplo:

Los Quince **creen** que habrá una recuperación.

En esta frase el analizador devuelve dos posibles lemas para la palabra *creen*, *crear* y *creer*. En esta etapa se va, pues, a intentar obtener cual de los lemas es el correcto.

El proceso se basa, como no puede ser de otra forma, en el conocimiento contenido en el propio documento. Se trata, por tanto, de delimitar el significado de un texto a partir de su propia estructura. Ello conlleva la consideración de un proceso iterativo de punto fijo en el que cada paso de la cadena secuencial reinyecta en la siguiente iteración la información previa obtenida en forma de peso probabilística asignado a una determinada categoría léxica susceptible de ser asociada a un *token*.

NOTA: El corpus que se ha utilizado en el caso concreto desarrollado, es un corpus muy pequeño a partir del cual es muy difícil obtener unos valores globales razonables, pues en ciertos casos no existe ninguna frase en la que la ambigüedad pueda decidirse.

(B) DESAMBIGUACIÓN SINTÁCTICA

En esta etapa se consigue que las palabras de una frase que tengan un gobernante, éste sea único. El proceso igual que el anterior, pero en este caso las fórmulas aplicadas son diferentes.

(C) APRENDIZAJE DEL CONOCIMIENTO

Esta etapa basa su funcionamiento en el hecho de que palabras que compartan contextos de aparición son palabras con una proximidad semántica muy fuerte, y por consiguiente, es posible poder asignarlas a una misma clase o categoría semántica.

El algoritmo parte aquí de un conjunto de clases iniciales que es necesario que el usuario proporcione al sistema.

$DEPENDENCIAS=\{dep_1(elto_{11},elto_{12});\ dep_2(elto_{21},elto_{22});\ldots;\ dep_m(elto_{m1},elto_{m2})\}$
$CLASES=\{c_1;c_2;\ldots;c_n\}$

Desde i=1 ata i=|DEPENDENCIAS|
  Desde j=1 ata j=|CLASES|
    probabilidad($elto_{i1}$:cat_$elto_{i1}$:c_j) sabiendo
    Desde k=1 ata k=|CLASES|
      probabilidad($elto_{i2}$:cat_$elto_{i2}$:c_k)

Es decir para cada dependencia se calculan todas las posibles probabilidades que puedan tener sus elementos perteneciendo a una determinada clase y siempre teniendo en cuenta los valores de ambos elementos de la dependencia. El valor *cat_elto* es la categoría que tiene asignada la palabra en esa frase.

El proceso global se realiza aplicando en un primer momento la etapa 1. Finalizada la etapa 1, las etapas 2 y 3 se realizan simultáneamente en la misma iteración, donde los valores de la etapa 3 dependen de lo obtenido en la etapa 2. En un caso ideal el proceso debería ejecutarse realizando las tres etapas en la misma iteración, pero debido al coste computacional y temporal que esto supone, se ha optado por realizar ambas partes de forma separada.

13.2. INTERACCIÓN DEL USUARIO EN EL PROCESO DE GENERACIÓN DE LA ONTOLOGÍA

Como se ha comentado anteriormente, el proceso de crear una ontología por dependencias implica, si el usuario lo desea, una interacción del mismo para mejorar los resultados que él considere oportunos. Hay que tener en cuenta, que la arquitectura descrita en este proyecto es una estructuración semi-automática del conocimiento, esto implica que el usuario puede participar en la misma para obtener resultados óptimos. Por otro lado, el algoritmo planteado necesita partir de una base para poder aprender cuales son las dependencias relevantes en el dominio.

Así, a partir de un conjunto de ficheros, que se detallan a continuación, el usuario puede mejorar la especificidad de la ontología generada, realizando cambios oportunos en dichos archivos.

FICHERO: ELEM.TXT

Este fichero contiene una jerarquía de cuales son los conceptos y propiedades definidos para la ontología:

```
MERCADOS
        ENTIDADES
        INDICES
        DINEROS
        PAISES
        PERSONAS
PROPIEDADES
        LUGARES
        TIPOS
        TAMAÑO
        PERÍODO
        CATEGORÍA
```

La explicación de la tabla anterior es la siguiente:

- Los elementos situados bajo MERCADOS son los conceptos de la ontología, es decir, definirán las clases de la misma, MERCADOS, será una super-clase de todas ellas.

- Los elementos situados bajo PROPIEDADES son las propiedades definidas para cada una de las clases anteriores.

El usuario podrá modificar estos nombres por los que él considere oportunos.

FICHERO: MARCADORES.TXT

```
bolsa     MERCADOS        PAISES/INDICES   [2]
bolsa     PROPRIEDADES    LUGARES/TIPOS    [2]
mercadoPROPIEDADES        LUGARES/TIPOS    [2]
mercadoMERCADOS           INDICES/DINEROS [2]
tamaño    PROPRIEDADES    TAMAÑO [2]
gestión   MERCADOS        ENTITES/DINEROS          [1,2]
gestión   PROPIEDADES     TIPO/TAMAÑO/LUGARES      [2]
inicio    PROPRIEDADES    PERÍODO[2]
```

En este fichero se definen los marcadores del corpus. Los marcadores son palabras clave en los textos que marcan un tipo de relación entre las palabras que involucra. A modo de ejemplo, en la tabla anterior, en primer lugar se tiene

bolsa MERCADOS PAISES/INDICES [2]

Esto indica que la palabra bolsa puede relacionarse con palabras que definan conceptos (esto está indicado con MERCADOS) y concretamente, entre los posibles conceptos, se relaciona bien con PAÍSES bien con ÍNDICES. El número entre corchetes, indica la posición del concepto dentro de la relación, por ejemplo el término bolsa de Japón genera la dependencia (bolsa, Japón), donde Japón marca el concepto y se sitúa en la segunda posición de la dependencia.

El usuario puede añadir marcadores o modificar los actuales según los resultados que desee obtener.

FICHERO: PROPIEDADES.TXT

Este fichero guarda un conjunto de propiedades que el algoritmo va generando. Este fichero en cada iteración es recuperado por el proceso para poder aprender nuevas propiedades según los valores que encuentre en el mismo. Su estructura es la siguiente:

```
europeo LUGARES
baja    TAMAÑO
caída   TAMAÑO
krach   TAMAÑO
junio PERÍODO
año     PERÍODO
día     PERÍODO
abril   PERÍODO
mitad   TAMAÑO
americano       LUGARES
parisino LUGARES
```

```
alto        TAMAÑO
internacional       LUGARES
francés  LUGARES
general TIPO
único       TIPO
```

En la primera columna del fichero se puede observar los valores recuperados del corpus y en la columna de la derecha el tipo de propiedad que se le ha asignado. De este modo, el usuario podría recuperar el documento y bien añadir nuevos valores o bien modificar los existentes.

FICHERO: MERCADOS.TXT

En este fichero se recuperan un conjunto de conceptos que el algoritmo va generando. Al igual que con el fichero anterior, el proceso recupera el fichero en cada ejecución, ayudándose de su contenido para buscar nuevos conceptos y añadiendo al mismo esos nuevos conceptos encontrados. Su estructura es similar al anterior:

```
ministro PERSONAS
fluctuación         DINEROS
banco    ENTIDADES
divisa   DINEROS
grupo    ENTIDADES
inversor PERSONNES
Dow&Jones           INDICES
Nikkei   INDICES
```

En la primera columna del fichero se puede observar los valores recuperados del corpus y en la columna de la derecha el tipo de concepto que se le ha asignado. De este modo, el usuario podría recuperar el documento y bien añadir nuevos valores o bien modificar los existentes.

13.3.CREACIÓN DE LA ONTOLOGÍA POR DEPENDENCIAS

Para generar la ontología por dependencias es necesario previamente haber lanzado el correspondiente algoritmo. El algoritmo descrito anteriormente, actúa sobre la base de datos modificando la tabla *NewDependency* (ver diagrama de entidad/relación en el apartado 9.1). Es analizando esta tabla que se obtienen los datos necesarios para crear y almacenar la ontología.

Además, es necesario examinar el fichero ELEM.txt, comentado en el apartado anterior, para la generación de conceptos en la ontología. En este fichero se detallan cuales son los conceptos bases considerados y cuales son las propiedades básicas. A partir de esos datos el algoritmo agrupará bajo un concepto o propiedad un determinado valor. Los conceptos tenidos en cuenta son:
ENTIDADES
INDICES
DINEROS
PAISES
PERSONAS

En cuanto a las propiedades se tiene:

LUGAR
TIPO
TAMAÑO
TIEMPO
CARGO

De este modo, se definirán, por un lado, relaciones entre conceptos, por ejemplo una relación entre una entidad y un índice; y por otro lado, relaciones que impliquen un concepto y una propiedad, por ejemplo una relación entre una persona y su cargo.

Con esta información el prime paso efectuado es la creación de las clases que serán los conceptos definidos en el fichero nombrado anteriormente. En un segundo caso se definirán las propiedades. A la hora de definir las propiedades se ha de tener en cuenta que se tienen propiedades de tipo *dato*, es decir, propiedades que

implican que una clase se relacione con un valor; y se tienen propiedades de tipo objeto, es decir, propiedades que implican que una clase se relacione con otra. Por consiguiente, se crean tantas propiedades de tipo dato como propiedades aparezcan en el fichero mencionado y se crean tantas propiedades de tipo objeto como relaciones posibles entre las clases (esto incluye también que una clase se pueda relacionar consigo misma [13])

Una vez la jerarquía creada se puede pasar a analizar los datos procedentes de las base de datos y que almacenarán relaciones entre palabras marcando el tipo de relación existente. Las relaciones se almacenarán en la ontología en forma de individuos. Así, una clase de tipo PERSONAS podrá tener varios individuos asociados como por ejemplo GEORGE BUSH o LAURENT FABIUS. Cada individuo tendrá asociado una serie de propiedades que serán de alguno de los tipos que ya se haya definido.

A la hora de ampliar las consultas del usuario, el proceso es el siguiente:

Como en el caso anterior se examina si la consulta es sobre un elemento que pertenece a la ontología, pero en este caso se comprueba si el elemento existe como instancia.

En segundo lugar hay que recuperar los posibles padres de ese elemento, pues en la organización planteada una determinada instancia podría pertenecer a más de una clase, puesto que en ciertos casos será considerada de un tipo y en otros casos de otro.

Se recuperan a continuación todos los individuos correspondientes a la consulta del usuario y ésta se amplia con el valor de las propiedades del individuo

[13] Es necesario destacar que la relación no es propiamente entre dos clases, sino que entre dos instancias de clase.

PRUEBAS

14.PRUEBAS DEL SISTEMA

En esta sección se detallan las pruebas realizadas para comprobar el correcto funcionamiento del sistema.

A lo largo del desarrollo de un proyecto software pueden aparecer errores en los sitios más comunes como en aquellas líneas de código donde no cabía esperar la presencia de errores. Por este motivo el proceso de pruebas es una parte fundamental de un desarrollo y a la cual el esfuerzo dedicado debe repartirse a lo largo de la implementación y en cada etapa del proceso.

Se puede definir una prueba como el proceso de ejercitar un programa con la intención específica de encontrar errores previos a la entrega al usuario final. El objetivo de las pruebas es entonces:

- Encontrar defectos en el software.

El éxito de una prueba depende de si encuentra o no un error. Así una prueba es óptima si descubre un error que no existía hasta el momento y que hacía suponer que el sistema funcionaba de manera esperada.

- Encontrar aquellos errores que no se han descubierto hasta el momento.

Por consiguiente el motivo detrás del diseño de las pruebas es encontrar aquellas que tengan la más alta probabilidad de detectar el mayor número de errores.

14.1.PRUEBAS DE UNIDAD

Estas pruebas se centran en cada unidad de software o módulo individualmente, realizando pruebas de caja blanca y de caja negra. Se tiene que probar que la información que fluye entro los módulos del sistema se hace de forma adecuada hacia y desde la unidad de programa que está siendo probada. Es necesario, por lo tanto, probar cada componente de forma individual.

14.1.1.PRUEBAS DE CAJA BLANCA

Las pruebas de caja blanca se centran en el examen de detalles de bajo nivel (módulos). Se encargan de validar la lógica de cada uno de esos módulos y deben garantizar:

- El recorrido de todos los caminos independientes de cada módulo.
- El recorrido de todos los bucles.
  Es, por ello, que durante la implementación del sistema, se han ido validando cada uno de los métodos, probando todos los caminos de control importantes así como los caminos excepcionales. Se trata de averiguar que se cumplan los requisitos justo arriba presentados.

De este modo, se han examinado las estructuras de datos locales para asegurarse que los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de la ejecución. Se verifica además que el flujo de datos es adecuado entre los métodos de un mismo módulo.

14.1.2.PRUEBAS DE CAJA NEGRA

Estas evaluaciones se centran en probar que se verifican los requisitos funcionales del software, es decir, que las funcionas para las cuales se ha diseñado el sistema se cumplen sin errores. Es en esta parte en la que también se incluyen las pruebas que tratan de encontrar fallos que no se atienen a su especificación, como la interfaz con el usuario, la apariencia de los menús, el control de teclas, etc.

Para probar que el funcionamiento del sistema era el deseado se probaron aspectos como:

- Se proporcionan una serie de clases que implementan concretamente la extracción (referencia: en el caso del proyecto aquellos extractores incluidos, Acabit, Fastr, etc). Definiendo esas clases con los métodos definidos en la interfaz que implementan, se ha comprobado que las llamadas y el proceso lanzado por cada clase era satisfactorio. De este modo se comprobó que se producía una extracción correcta y que la estructura devuelta por un extractor era la esperada para generar el fichero que almacena la adquisición de terminología.

- Del mismo modo se procedió con el análisis sintáctico. Definiendo dos clases concretas que efectúan el análisis sintáctico, éstas son: ErialES y ErialFR. Implementándolas con la estructura definida en la interfaz que implementan, se obtuvieron resultados satisfactorios en cuanto a las llamadas efectuadas y en cuanto al resultado obtenido.

- Se hicieron pruebas con la carga de los documentos en el sistema, comprobando que los ficheros proporcionados al mismo cumplían con la estructura predefinida en el fichero de marcas. En este paso se averiguo, además, si los ficheros creados se validaban de acuerdo a su DTD.

- Además se han realizado pruebas sobre todas las sentencias SQL utilizadas en el sistema para comprobar que los datos extraídos de la base de datos eran los que se solicitaban. Para ello se ha utilizado la herramienta MySQL Query Browser, plataforma gráfica para el desarrollo y consulta en bases de datos MySQL.

- Respecto a la configuración se tuvieron que hacer pruebas de que no se repetían elementos en el fichero de configuración. Así, el fichero de configuración es validado en dos sentidos, contra su DTD, comprobando su correcta estructura y verificando que no existen duplicados de idiomas, temas, abreviaturas, extractores y analizadores sintácticos. Además del fichero de configuración existen otros ficheros también con base XML sobre los cuales también se realiza una comprobación sobre sus correspondientes DTDs.; éstos son los ficheros con los textos del sistema, los ficheros con los resultado de la extracción y los ficheros con la clasificación realizada.

- Por otro lado, se analizó si ciertas operaciones necesarias en etapas previas a otros procesos ya habían sido llevadas a cabo. Por ejemplo, se hace una comprobación de si existe el corpus asignado a un determinado tema e idioma, antes de lanzar un proceso de extracción de términos sobre ese corpus. Del mismo modo se procedió con la búsqueda de documentos para la cual es imprescindible que las ontologías bien de términos bien de dependencias estén creadas.

- Otro de los aspectos fundamentales tenido en cuenta en las pruebas fue la correcta validación de las rutas hacia directorios, siendo necesario en ciertos casos que el sistema cree la correspondiente jerarquía.

- Por último, se ha comprobado que las clases que un usuario puede introducir como nuevos elementos en el sistema, se refiere esto a la posibilidad de integrar nuevos extractores así como nuevos analizadores sintácticos, sean correctas, para ello se procede a realizar una compilación de cada nueva clase proporcionada al sistema. A continuación se debe probar que la nueva clase está cargada en el sistema y se puede empezar a utilizar, este hecho fue comprobado obteniendo resultados satisfactorios.

PRUEBAS SOBRE LA INTERFAZ

Fichero de configuración con extensión errónea	
Descripción	Se introduce un fichero con extensión incorrecta (distinta de .xml).
Resultado	El sistema no activa el menú principal.
Evaluación	Correcta.

Fichero de configuración con estructura errónea	
Descripción	El fichero de configuración proporcionado no cumple con el formato especificado en la DTD.
Resultado	El sistema informa de ello a través de una caja de diálogo.
Evaluación	Correcta.

Fichero de configuración con elementos repetidos	
Descripción	El fichero de configuración proporcionado contiene elementos repetidos.
Resultado	El sistema informa de ello a través de una caja de diálogo.
Evaluación	Correcta.

Elegir un tema e idioma sin corpus	
Descripción	El usuario elige un tema de corpus y a continuación el idioma correspondiente.
Resultado	No existe un corpus creado para el tema e idioma elegido, esto es debido a que el sistema no ha recibido ningún documento todavía, se muestra un mensaje de error a través de una caja de diálogo.
Evaluación	Correcta.

Elegir un tema e idioma sin ontología por dependencias o por términos creada	
Descripción	El usuario elige un tema de corpus y a continuación el idioma correspondiente.
Resultado	No existe la ontología por dependencias necesaria para el proceso siguiente, búsqueda de documentos,(bien es cierto que la consulta introducida por el usuario podría no tener una correspondencia en la ontología, pero es necesario que ésta esté creada para realizar dicha comprobación), esto es debido a que todavía no se ha lanzado la clasificación por dependencias, el sistema informa de ello a través de una caja de diálogo.
Evaluación	Correcta.

Elegir un idioma sin ningún analizador sintáctico asignado	
Descripción	El usuario elige un idioma para efectuar una operación que requiera el uso de un analizador sintáctico.
Resultado	No existe ningún analizador sintáctico para ese idioma, el sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Elegir un idioma sin ningún extractor asignado	
Descripción	El usuario elige un idioma para efectuar una operación que requiera el uso de un extractor.
Resultado	No existe ningún extractor asignado a ese idioma, el sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Activar botón de extracción	
Descripción	El usuario elige los parámetros necesarios para proceder a la extracción.
Resultado	El sistema detecta si se han introducido todos los datos: tema del corpus, idioma y extractor/es. Si es el caso se activa el botón de la extracción.
Evaluación	Correcta.

Activar botón de clasificación por términos	
Descripción	El sistema ha finalizado correctamente la extracción de términos.
Resultado	El sistema activa el botón de clasificación.
Evaluación	Correcta.

Activar botón de creación de ontología por términos	
Descripción	El sistema ha finalizado correctamente la clasificación de términos.
Resultado	El sistema activa el botón de creación de la ontología.
Evaluación	Correcta.

Activar botón de la clasificación por dependencias	
Descripción	usuario elige los parámetros necesarios para proceder a la clasificación.
Resultado	El sistema detecta si se han introducido todos los datos: tema del corpus e idioma. Si es el caso se activa el botón de la clasificación por dependencias.
Evaluación	Correcta.

Activar botón de creación de ontología por dependencias	
Descripción	El sistema ha finalizado correctamente la clasificación por dependencias.
Resultado	El sistema activa el botón de creación de la ontología
Evaluación	Correcta.

Lanzar la extracción	
Descripción	El usuario una vez elegidos los elementos necesarios procede a lanzar la extracción.
Resultado	El sistema muestra una caja de confirmación que informa de la tarea que se va a llevar a cabo con los parámetros introducidos por el usuario, si el usuario está de acuerdo deberá aceptar dicho diálogo.
Evaluación	Correcta.

Lanzar la clasificación por dependencias	
Descripción	El usuario una vez elegidos los elementos necesarios procede a lanzar la clasificación por dependencias .
Resultado	El sistema muestra una caja de confirmación que informa de la tarea que se va a llevar a cabo con los parámetros introducidos por el usuario, si el usuario está de acuerdo deberá aceptar dicho diálogo.
Evaluación	Correcta.

No se introduce ningún fichero para la generación del corpus	
Descripción	El usuario introduce los datos para proceder a actualizar el corpus de un tema e idioma pero no introduce ningún fichero.
Resultado	El sistema actualiza la vista sin mostrar mensaje.
Evaluación	Correcta.

El fichero introducido no tiene la extensión adecuada	
Descripción	El usuario introduce un nuevo fichero que no tiene la extensión esperada.
Resultado	El sistema ignora ese fichero.
Evaluación	Correcta.

No se introduce consulta	
Descripción	El usuario deja el campo de la consulta vacío.
Resultado	El sistema se mantiene hasta que el usuario escriba la consulta.
Evaluación	Correcta.

Tema repetido	
Descripción	El nombre del nuevo tema introducido por el usuario ya existe en el sistema.
Resultado	El sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Borrar tema	
Descripción	El usuario elige un tema para eliminar.
Resultado	El sistema informa de la operación que se va a llevar a cabo y de los directorios que se van a eliminar.
Evaluación	Correcta.

Idioma repetido	
Descripción	El nombre del nuevo idioma introducido por el usuario ya existe en el sistema.
Resultado	El sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Borrar idioma	
Descripción	El usuario elige un idioma para eliminar.
Resultado	El sistema informa de la operación que se va a llevar a cabo y de los directorios que se van a eliminar.
Evaluación	Correcta.

Abreviatura repetida	
Descripción	El nombre de la nueva abreviatura introducida por el usuario ya existe en el sistema.
Resultado	El sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Extractor repetido	
Descripción	El nombre del nuevo extractor introducido por el usuario ya existe en el sistema.
Resultado	El sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Borrar extractor	
Descripción	El usuario elige un extractor para eliminar.
Resultado	El sistema informa de la operación que se va a llevar a cabo.
Evaluación	Correcta.

Analizador repetido	
Descripción	El nombre del nuevo analizador introducido por el usuario ya existe en el sistema.
Resultado	El sistema informa de ello con una caja de diálogo.
Evaluación	Correcta.

Borrar analizador	
Descripción	El usuario elige un analizador para eliminar.
Resultado	El sistema informa de la operación que se va a llevar a cabo.
Evaluación	Correcta.

14.2. PRUEBAS DE INTEGRACIÓN

El principal objetivo de este tipo de pruebas es detectar fallos en la comunicación e interacción de las clases del sistema. Se incluye aquí además la integración tanto de diversas tecnologías en la aplicación como la integración de herramientas externas a las clases Java propiamente dichas.

De este modo tomando los módulos probados en las pruebas de unidad se trata de ver si en la estructura construida a partir de las clases, fluye la información de forma correcta y esperada.

Durante la implementación las clases fueron probadas de manera individual y progresivamente en su conjunto. Con la aplicación ya acabada se procedió a un conjunto de pruebas globales obteniendo unos resultados satisfactorios.

14.3. PRUEBAS DE VALIDACIÓN

En estas pruebas se validan los requisitos establecidos en el proceso de análisis de requerimientos, comparándolos con el sistema que ha sido construido.

Para ello se hacen pruebas para cada uno de los requisitos establecidos obteniendo resultados satisfactorios. Así, el sistema cumplen con todas aquellas bases establecidas en sus inicios y realiza las funcionalidades esperadas.

ÍNDICE DE FIGURAS