(75.06) Organización de Datos
Trabajo Práctico Speaker – Segunda Entrega

Curso: Lic. Arturo Servetto
Ayudante Asignado: Renzo Navas
Grupo: Led Zeppelin

Integrantes:

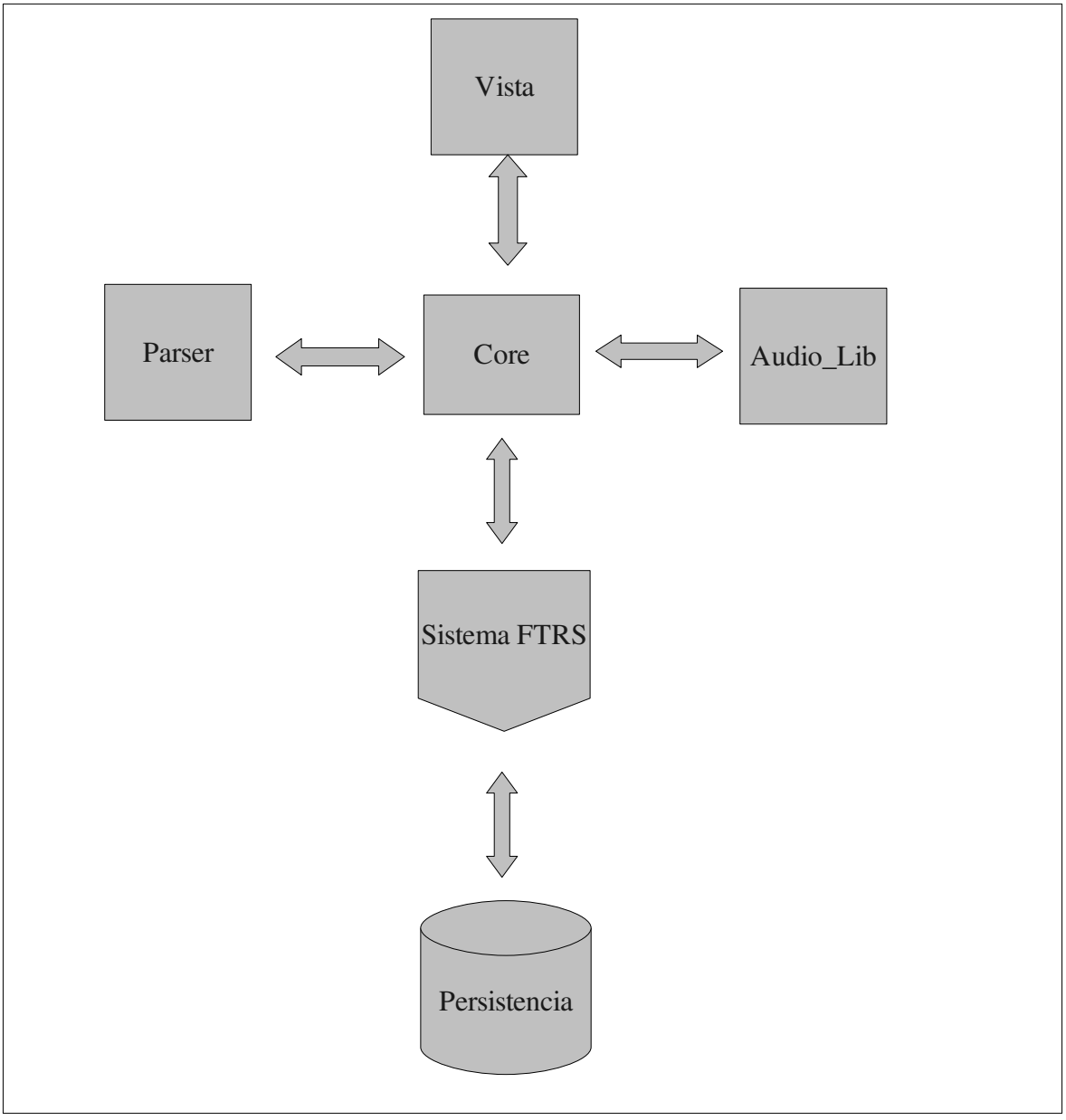
Gabriel Fusca	**86521**
Exequiel Leite	**82631**
Diego García Jaime	**78938**
Cristian Stügelmayer	**82521**
Ramiro Salom	**83350**

Índice de contenido

Arquitectura utilizada

La arquitectura utilizada se divide en 6 estructuras lógicamente diferenciadas:

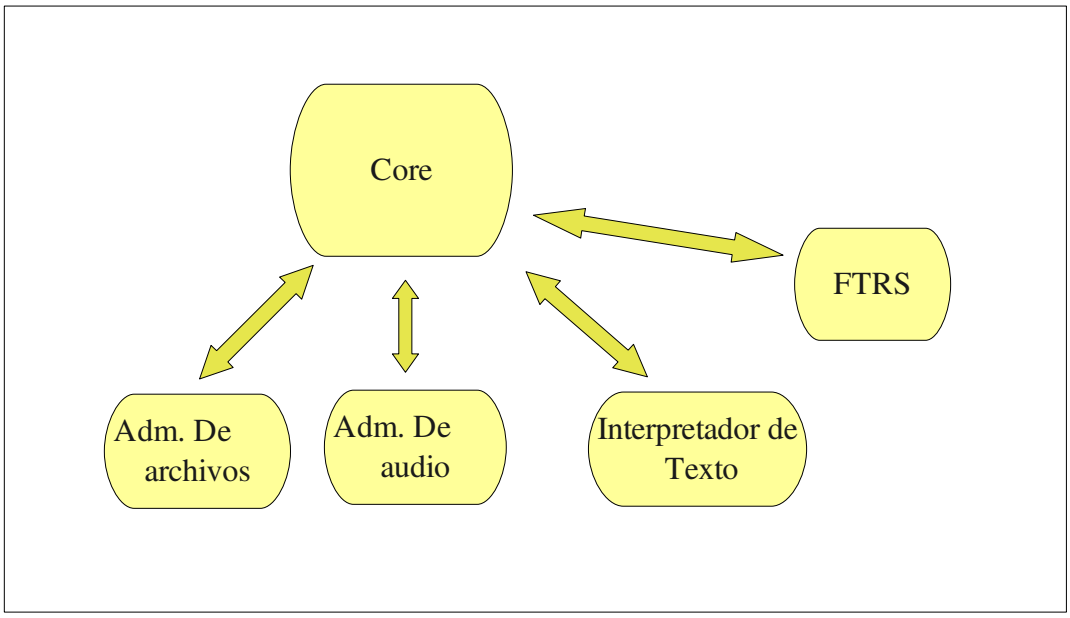
1. Vista
2. Core
3. Parser
4. Sistema FTRS
5. Persistencia
6. Audio_lib



Una vez que nos pusimos de acuerdo en la forma de encarar este nuevo requerimiento, se intentó ver como hacer evolucionar la solución propuesta para la segunda entrega sin tener que recurrir a desperdiciar todo el trabajo hecho anteriormente.
Cinco de estas estructuras ya fueron presentadas en la entrega anterior, de modo que en cada una se comentaran actualizaciones o modificaciones sobre las mismas.
Se ha prestado especial interés en esta entrega, en las capas de persistencia y FTRS.

Core (Capa de Control)

Diagrama de bloques

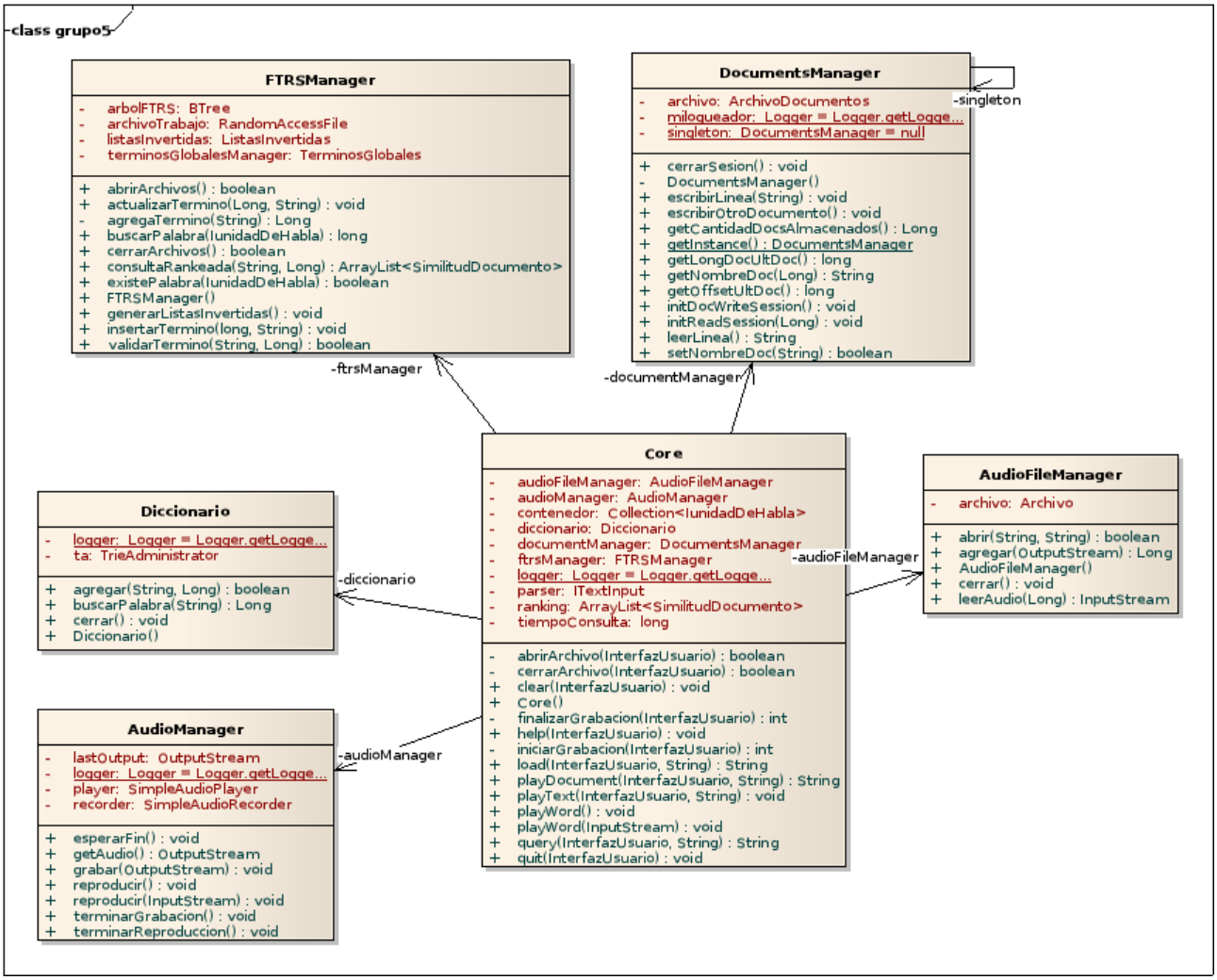


Bloque de Administracion de Archivos

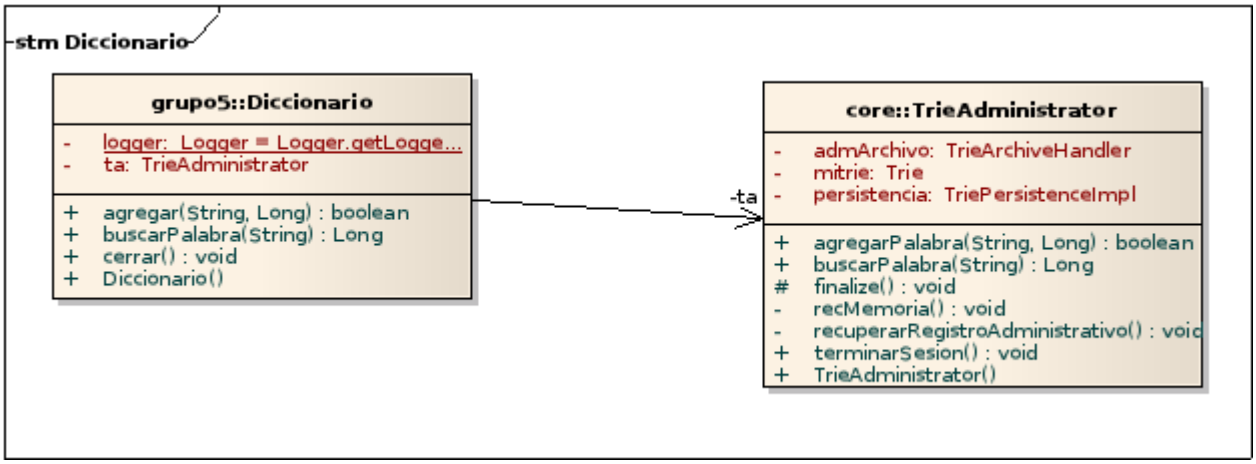
Este bloque se encarga de controlar la persistencia de datos. Esta compuesta por las clases:

1. AudioFileManager (ver las modificaciones).
2. Diccionario
3. DocumentsManager

Diagrama de Clases Core



Diccionario

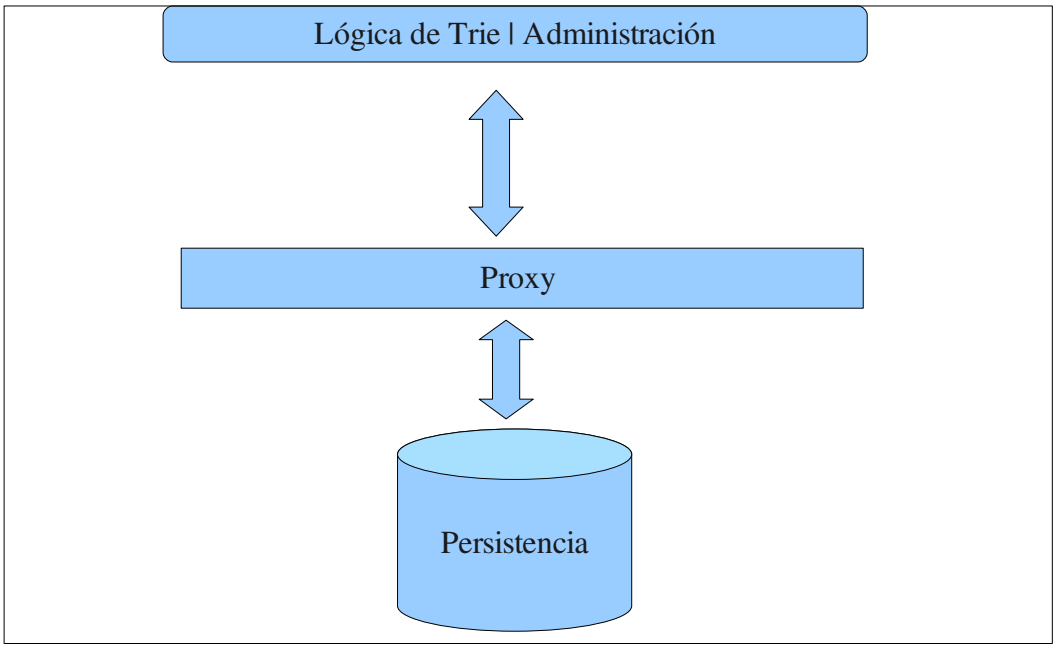


Es la clase encargada de la administración del léxico del sistema, contiene todas las palabras conocidas por el sistema.

A diferencia de la primera entrega, el nuevo requirimiento exigió el uso de un Trie para el manejo del léxico. Este es de profundidad parametrizable (fijada en 4 por requerimiento de esta entrga) y se encuentra encapsulado por la clase Diccionario.

Esquema general de funcionamiento del arbol "Trie"

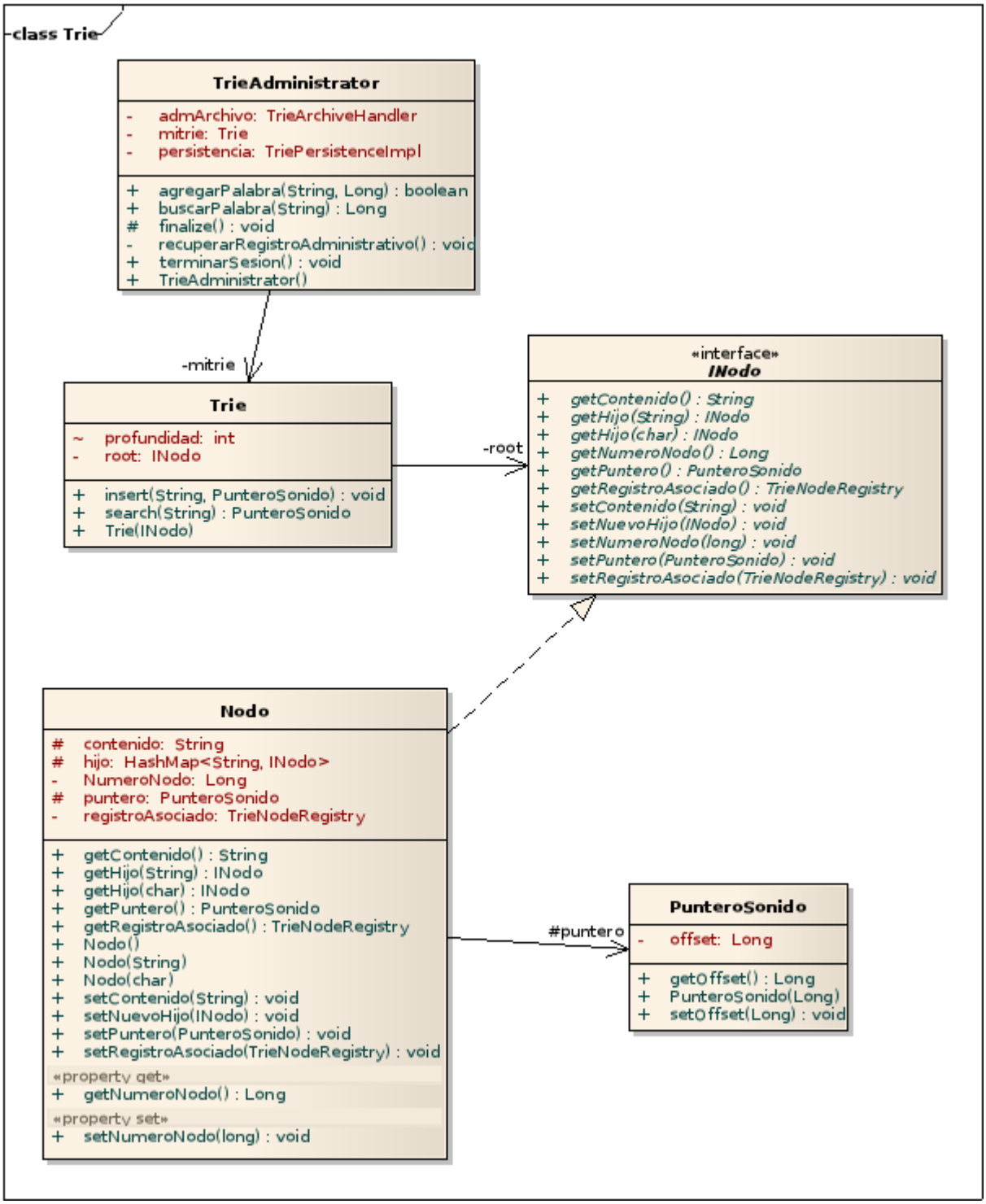
Para la implementación del Trie y su persistencia, se propuso un modelo de 3 capas:



Al desarrollar la arquitectura de este árbol, se pensó en un modelo que permita abstraer su lógica del lugar de funcionamiento, es decir, independientemente que el arbol funcione en memoria o en disco su algoritmia debía ser identica.
Este modelo permite tener, en la capa lógica, un algoritmo simple de administración de Trie en memoria, simplificando en gran medida el desarrollo de dicha funcionalidad.

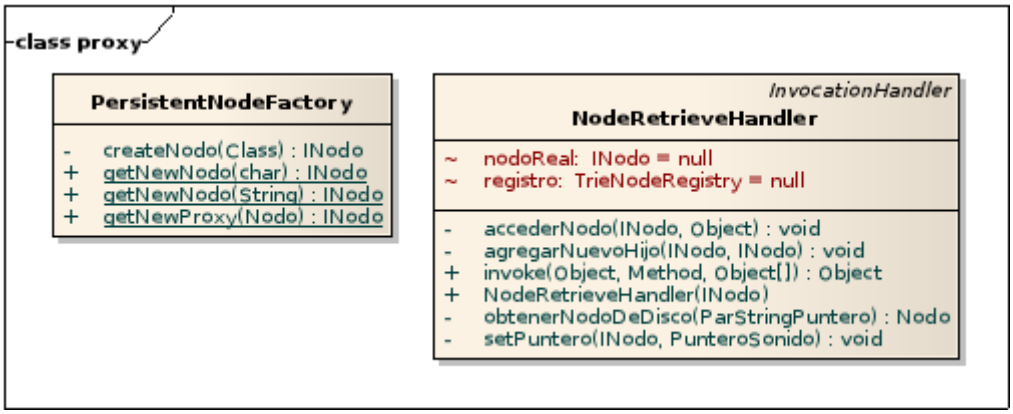
Capa Lógica o de Administración

Diagrama de Clases



Esta capa es la encargada del manejo del trie, a nivel lógico, haciendo transparente su manejo. Este sera el encargado de embocar al Trie para una búsqueda o inserción de una palabra.
El Trie hace a su vez de Diccionario conteniendo la palabra y un puntero (al final de la palabra) , indicando el Offset del audio asociado a esa palabra. El trie tiene una profundidad parametrizable (por defecto es 4, propuesta por la catedra).

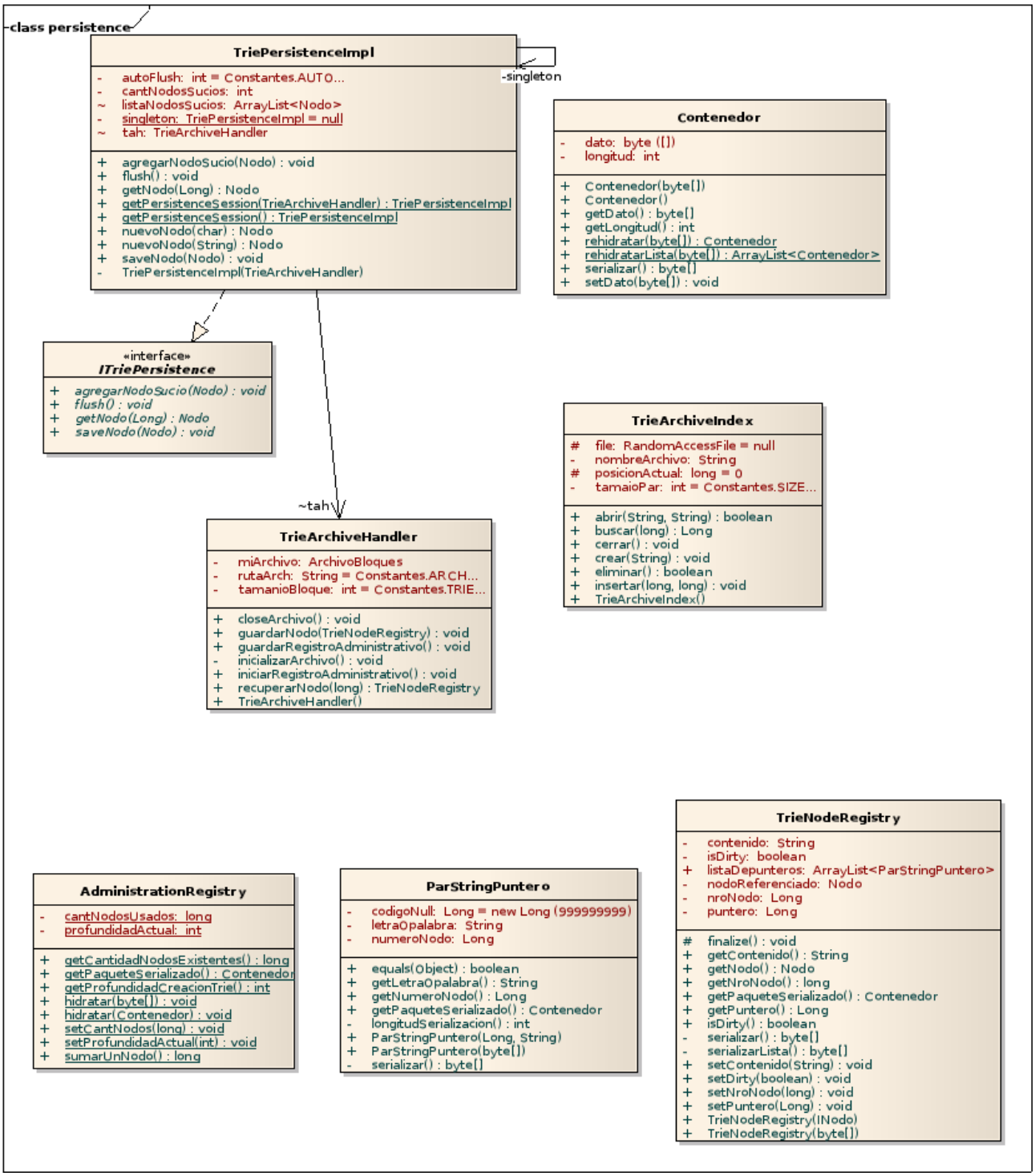
Proxy o Capa Intermedia



Esta capa hace de conexión entre la capa lógica y la de persistencia, proveyendole a la capa lógica los nodos requeridos para la busqueda o inserción, y también, la llamada actualización de los nodos. Para no tener que contar con todo el Trie en memoria, el trie siempre hace referencia a 2 nodos, la raíz y el actual, dejando al proxy la tarea de ir actualizando el actual en función del contenido fisico del Trie. Por cada llamada a insercion o a busqueda de un nuevo nodo (recorriendo el trie), el proxy intercepta el llamado y se encarga de interactuar con la capa física para pedir el siguiente nodo y llevarselo a la capa lógica para su utilzación, en el caso de búsqueda, o directamente la actualización del la capa física del Trie.

Capa Física

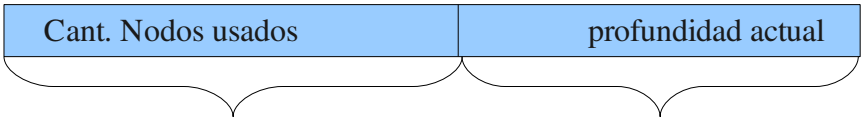
Diagrama de Clases



Es la encargada del manejo integro de la persistencia del Trie. La clase responsable de coordinar la persistencia es la llamada *TriePersistenceImpl.*

Para tener un control sobre la persistencia *TriePersistenceImpl* tiene un registro de control, llamado *AdministrationRegistry,* ubicado en el archivo en donde se almacenará el Trie.

Este tiene la siguiente esctructura:



Cantidad de nodos contenidos en el Archivo.

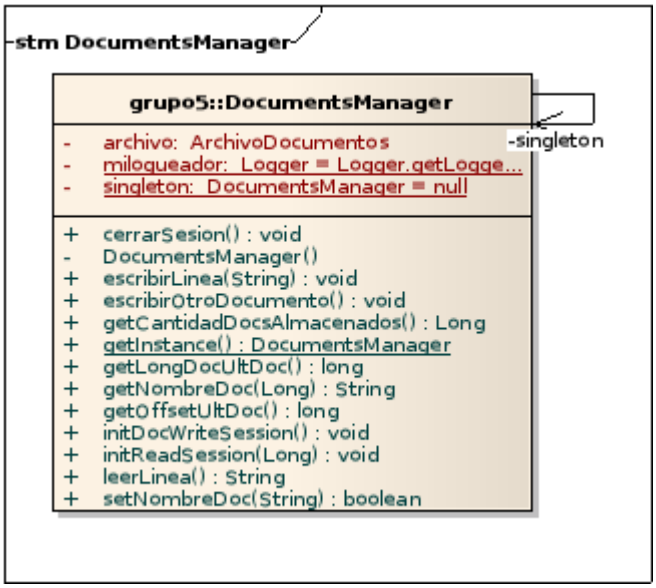
profundidad actual del Trie.

Cada Nodo del Trie esta compuesto por un Registro con la siguiente esctructura:

Nível	Nro Bloque	Long id Termino	Termino

- Nivel: nivel en el que se encuentra el Termino dentro del Trie.
- Nro Bloque: número de bloque en que se guarda el registro.
- Long. Id Termino: longitud del Termino almacenado
- Termino: el dato en sí.

DocumentsManager



Es la clase encargada de la administracion de documentos en disco. Para su diseño se utilizo un

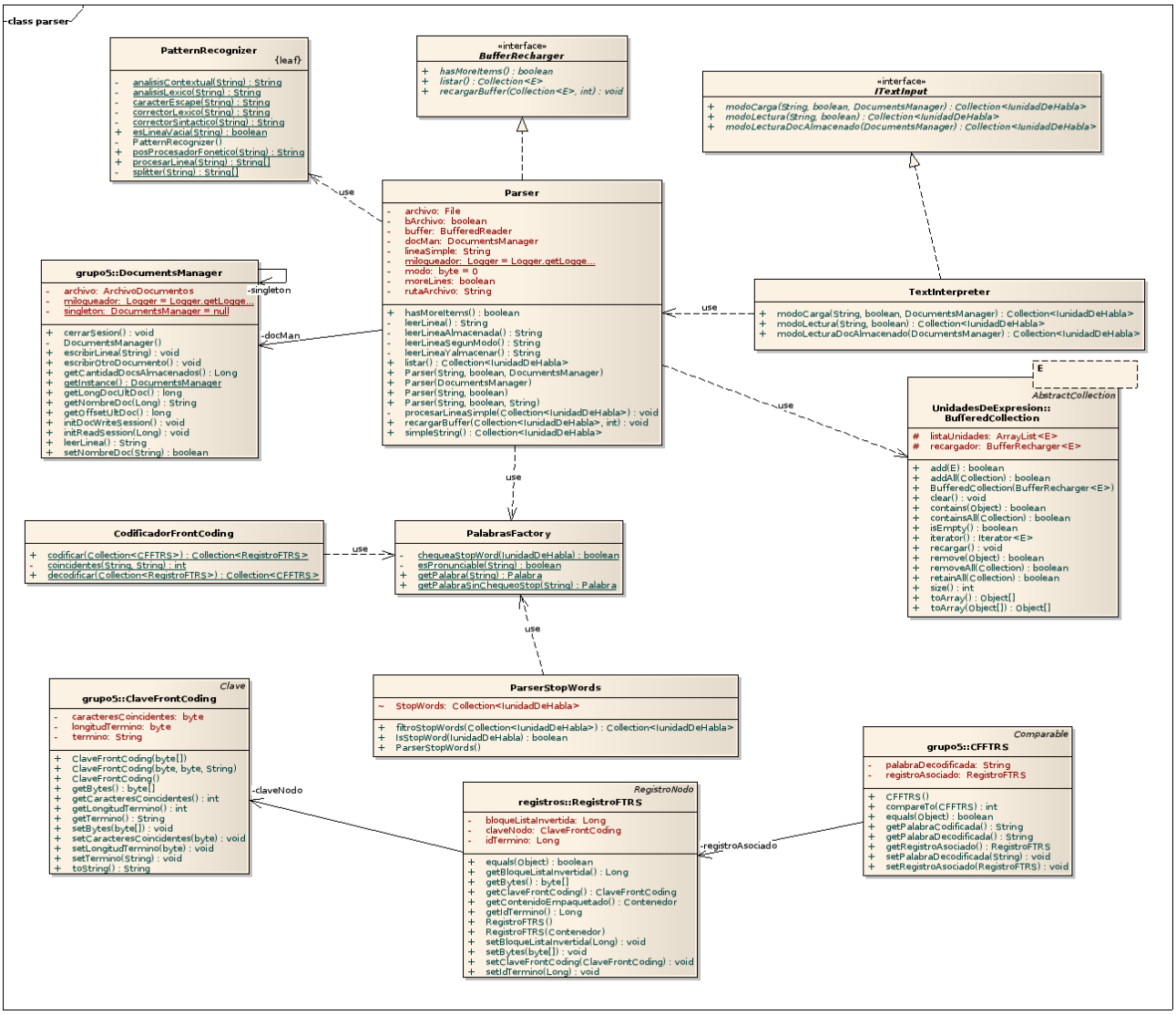
Singleton (patrón de diseño), para asegurarse de que siempre que se llame, sin importar de donde se llame, se hará referencia a la misma instancia.

Bloque Administracion de Audio

Este bloque no presento cambios frente al nuevo requerimiento.

Interpretador de Texto

Diagrama de Clases Interpretador de Texto



Esta solución, compatible con la versión anterior, extiende sus funcionalidades brindando:

- Verificación de stop words, para la utilización, por ejemplo, de indexación de palabras con relevancia.
- Codificacion y decodificacion en front coding de una serie de palabras
- Se extendio la funcionabilidad sobre lectura y escritura de Documentos brindando una clase que administra estas operaciones

Verificacion de stop word

La clase encargada de manejar esta verificación se llama *ParserStopWords*. Cada palabra considerada stop word se almacena en un archivo xml , de tal forma que frente a una nueva palabra considerada stop word lo único que se tiene que hacer es agregarla al archivo de stopwords. Además esta clase brinda la posibilidad dada una lista de palabras filtrar las que son consideradas stop words.

Front coding

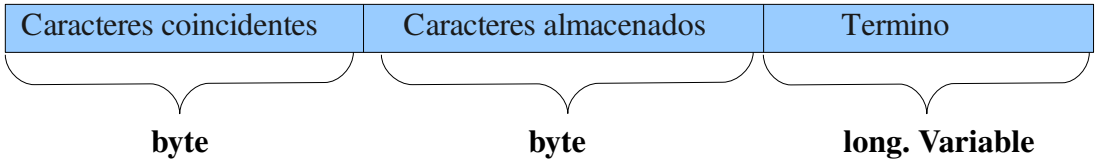
La clase encargada de la codificación y decodificación en front coding es la llamada *CodificadorFrontCoding*.
Para la codificación, este recibe una serie de CFFTRS, esta clase contiene un RegistroFTRS y la palabra decodificada.

Para la decodificicación, este devuelve una serie de RegistroFTRS el cual tiene:

- Id del Termino
- Referencia a bloque de la Lista Invertida
- Clave con front coding

La estructura utilizada para la codificación de una palabra en front coding fue:

Clave en front coding:



1. Caracteres coincidentes con el termino anterior.
2. Caracteres almacenados, se almacenan sin los caracteres coincidentes.
3. Termino restante sin caracteres coincidentes.

Administrador de Documentos

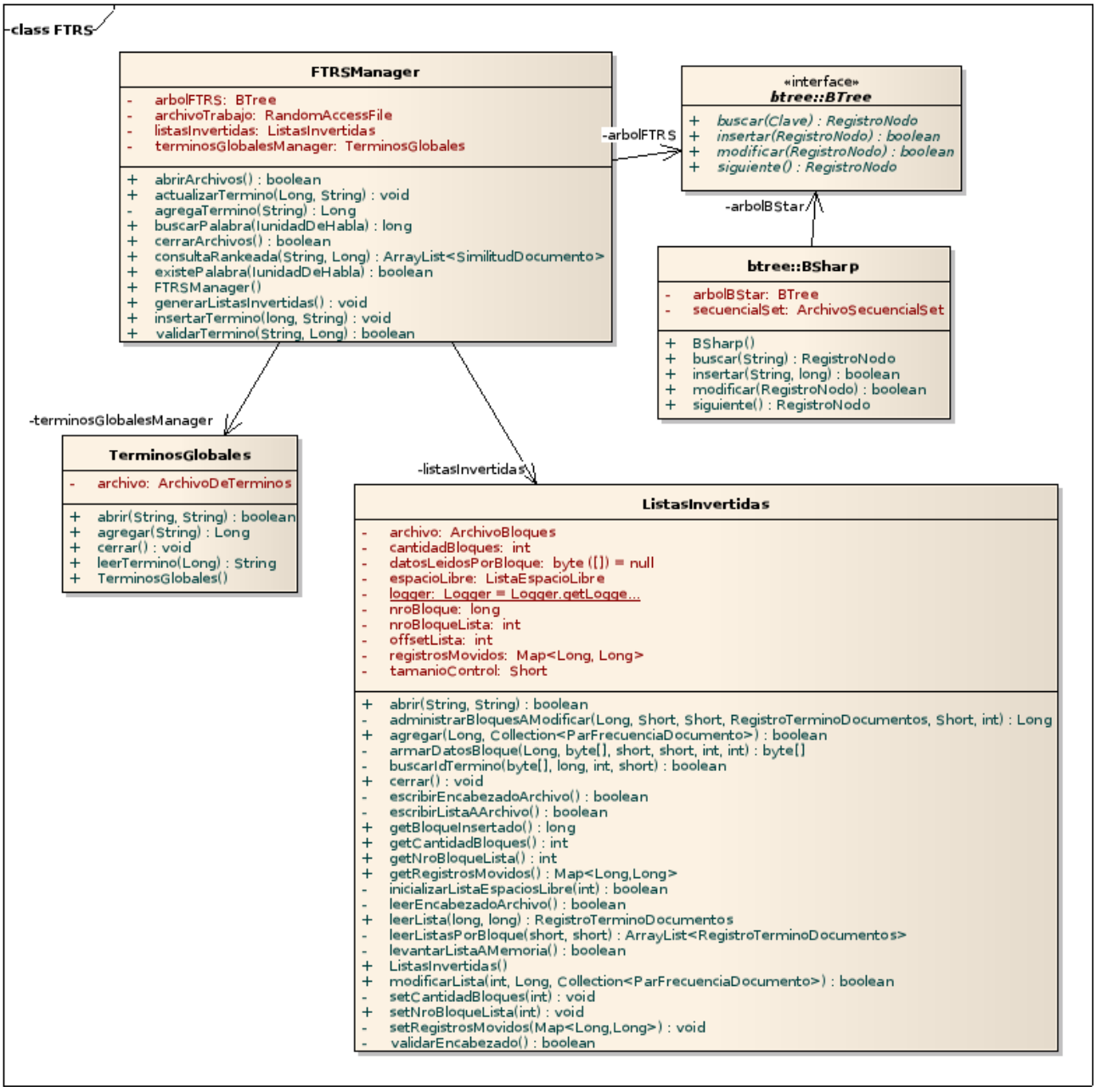
La clase encargada de administrar las operaciones sobre documentos se llama *DocumentsManager*. El parser brinda 4 modos para el manejo de escrtiura lectura:

1. Lectura de archivo y almacenamiento en *DocumentsManager*
2. Lectura sobre doc Almacenado
3. Lectura sobre String sin almacenar
4. Constructor sin almacenamiento con otro charset

Sistema FTRS

La clase encargada de manejar el sistema FTRS es la llamada *FTRSManager*. Esta es la responsable de administrar las consultas y manejar de los documentos almacenados.

Diagrama de Clases Sistema FTRS



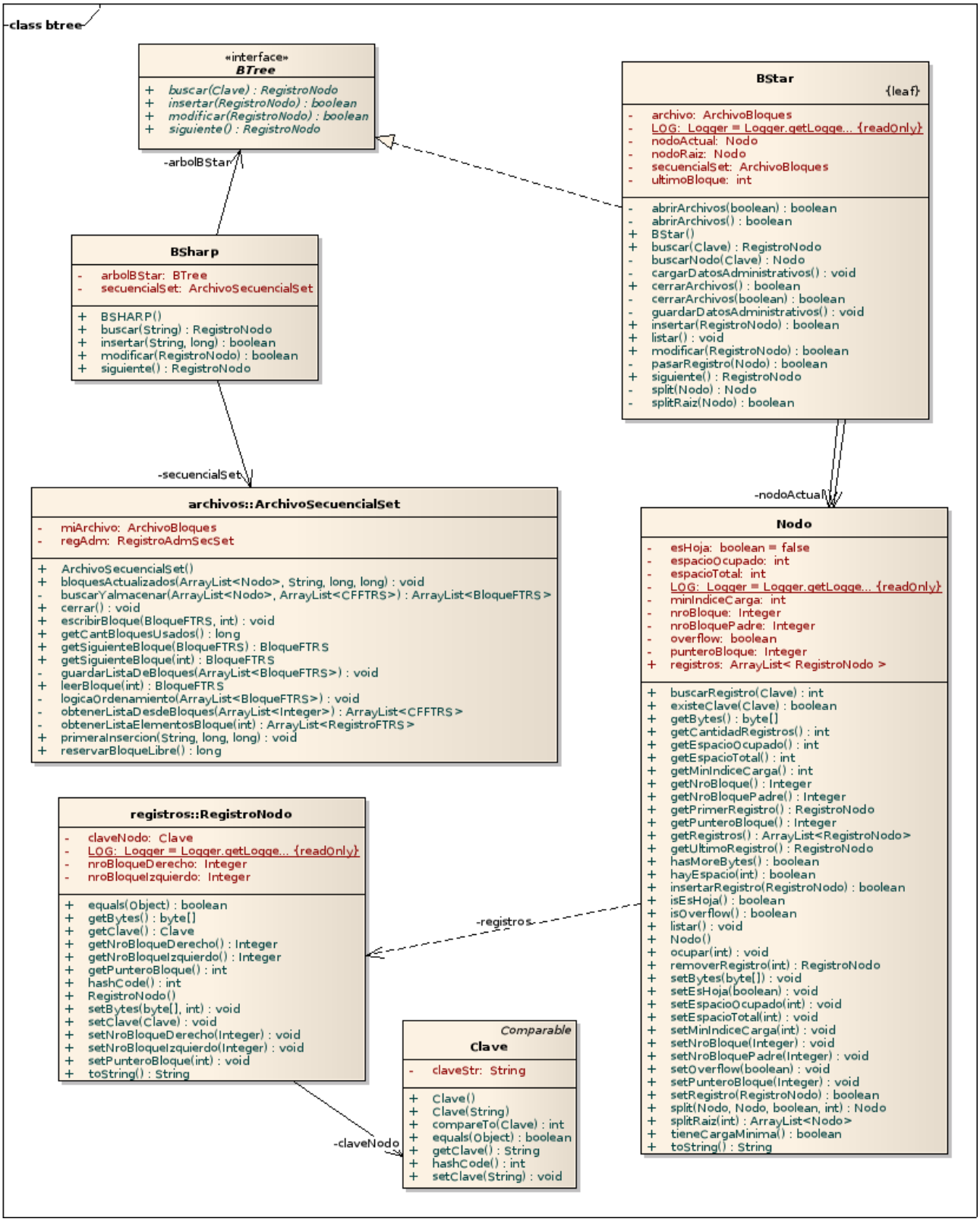
Para todo lo que es el Sistema de Recuperación de Texto se eligió el modelo Vectorial (sugerido por el requerimiento), de modo que todas las consultas se resolverán en base al modelo vectorial, y, todos los documentos se almacenarán siguiendo la misma lógica.

El sistema FTRS se compone de:

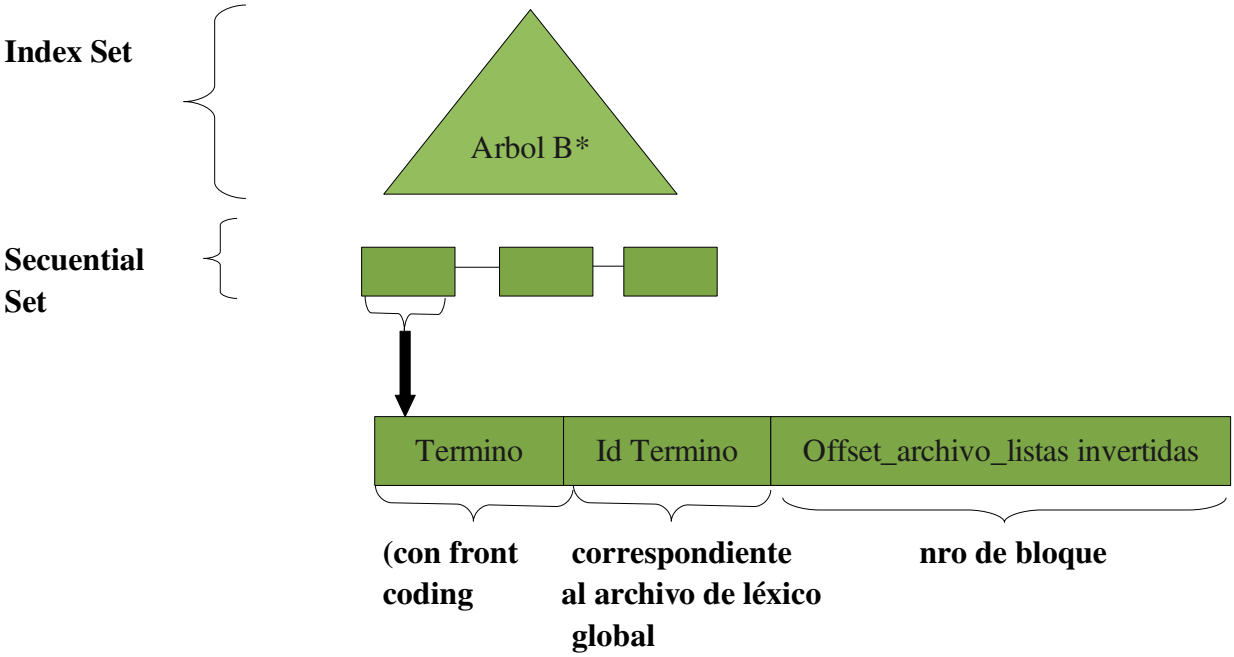
- Un árbol B#.
- Un administrador para la listas invertidas de los términos.
- Un administrador para el archivo de términos globales.
- Un administrador para el archivo de trabajo con duplas.

Arbol B#

Diagrama de Clases



Esctructura Arbol B#



El arbol B# esta compuesta por:

- Un arbol B* . (para los nodos índices).
- Un conjunto secuencial para los nodos hojas.

Cada nodo del arbol esta conformado por una serie de RegistrosNodos. La estructura de un RegistroNodo es:

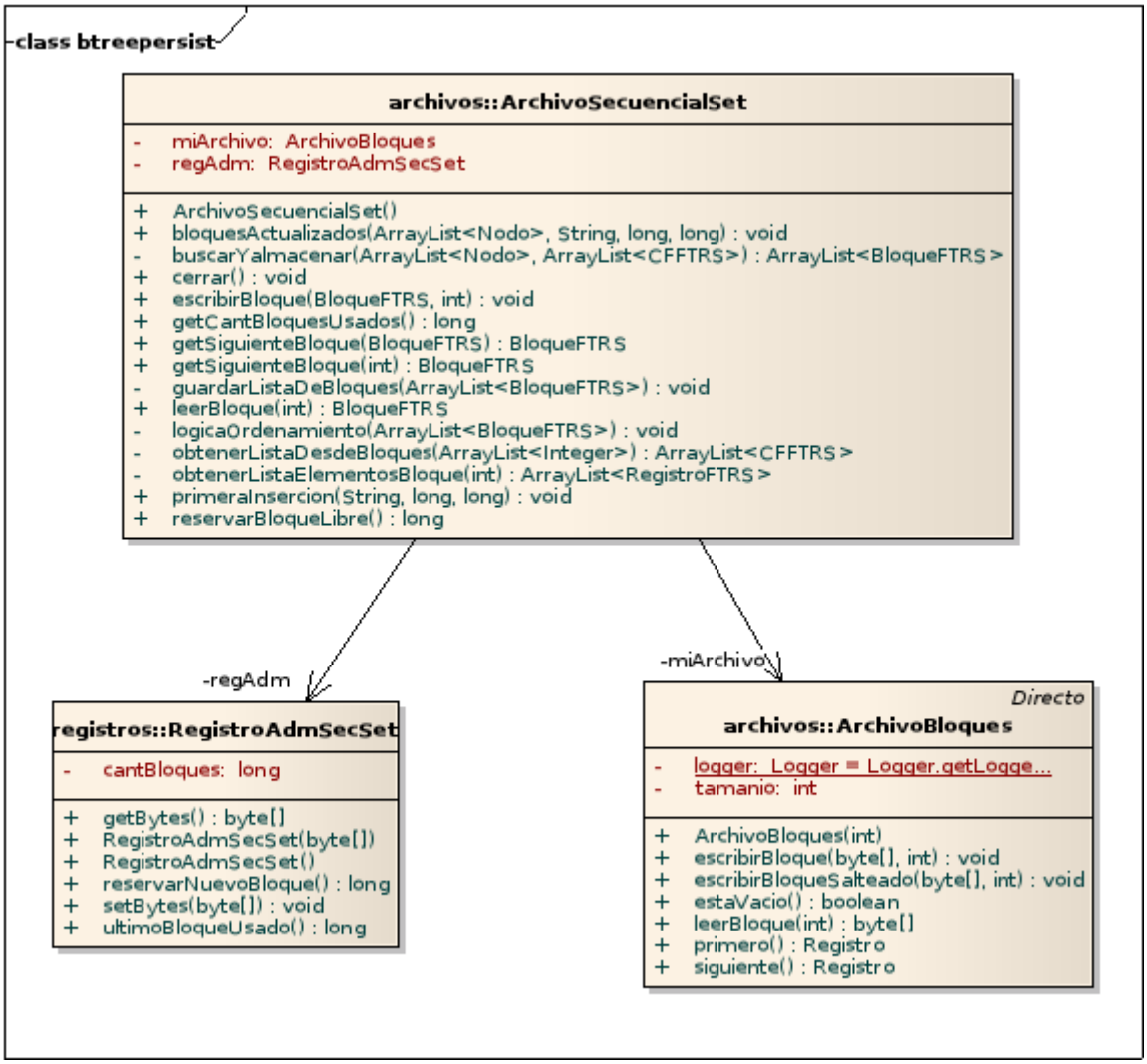
Clave	Nro bloque Izq	Nro Bloque Der.

- Clave: es el término en sí. Si es un nodo hoja el término se encontrará codigado en front coding, en cambio, si es un nodo índice, el termino estará sin codificar.
- Nro. Bloque Izq. : Número de bloque al que apunta a la izquiera.
- Nro. Bloque Der.: Número de bloeque al que apunta a la derecha.

Secuencial Set

La clase encargada administrar el secuencial set, es la llamada *SecuencialSet.* Este además del manejo en memoria, se encarga de la persistencia de los nodos hojas del arbol.
El archivo de nodos hojas tiene un registro encargado de la parte administrativa del archivo. Este tiene la cantidad de bloques almacenados en el archivo.

Diagrama de clases



Cada nodo de la lista del Secuencial Set tiene la siguiente estructura:

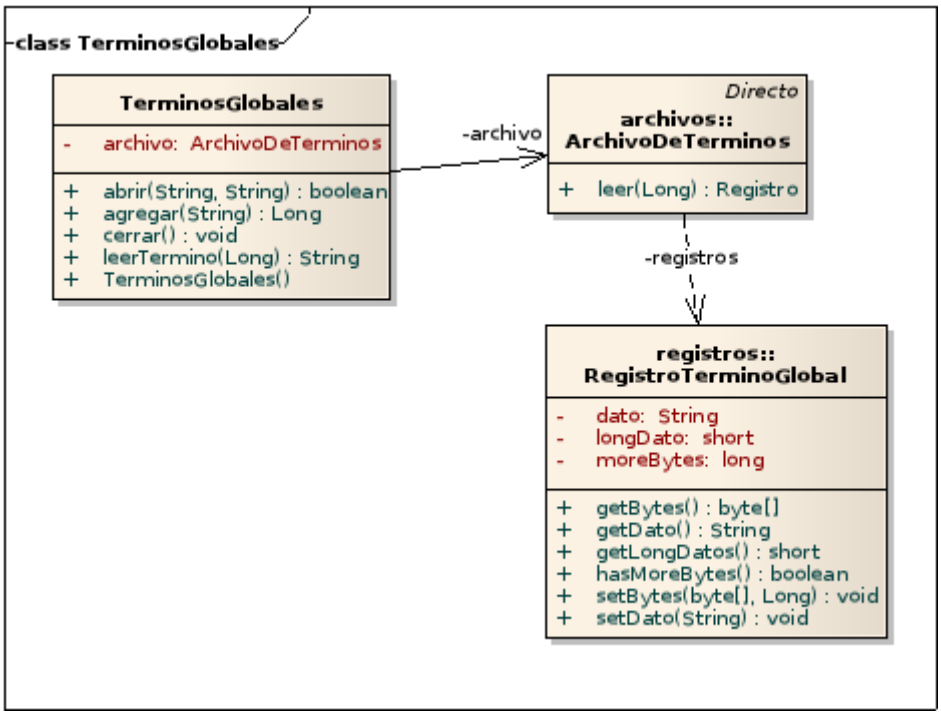
- Termino: termino codificado en front coding (explicado en otra sección).
- Id Termino: id correspondiente al archivo de léxico global (explicado en otra sección).
- Offset: nro de bloque del archivo de listas invertidas. (explicado en otra sección)

Administrador de archivo de lexico global

La clase responsable de la administración del archivo de léxico global es la llamada *TerminosGlobales.*

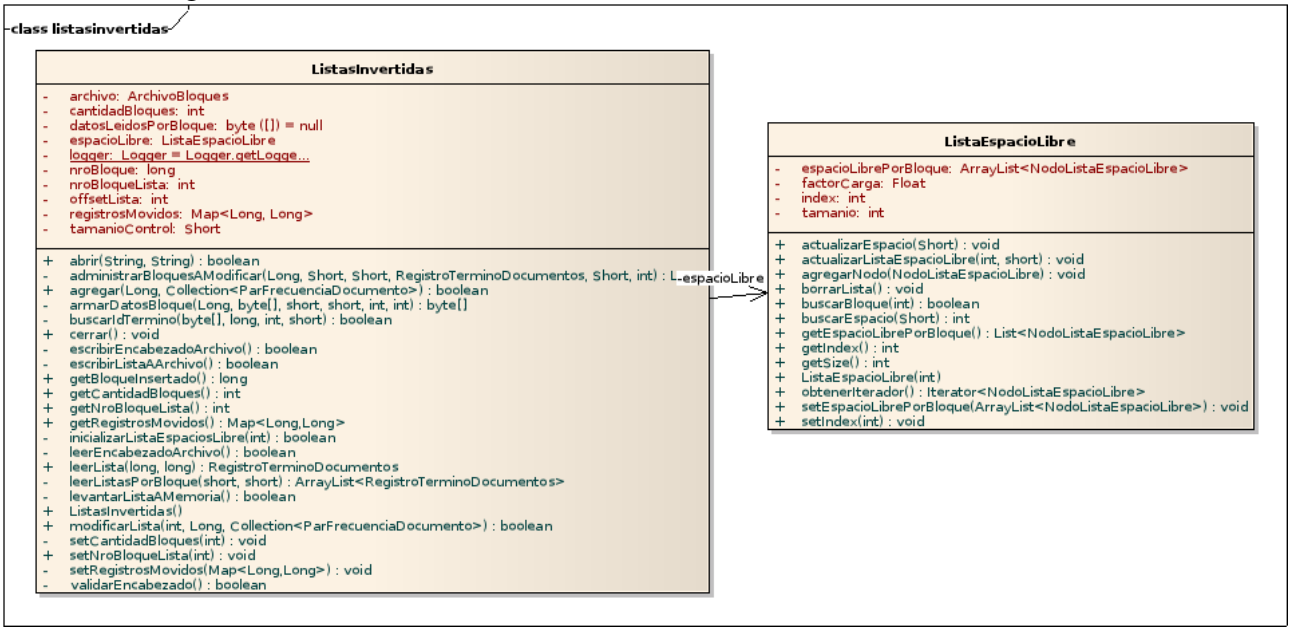
El archivo de terminos globales tiene como funcionabilidad el de brindar, dado un termino su id asociado (offset dentro del archivo de terminos globales) y, dado un id de un termino, dar su id asociado. Es el traductor entre el FTRS y el diccionario. De este modo se identifica univocamente cada término. Es un archivo organizado en bloques de registros de longitud variable. Cada registro tiene la estructura: | Longitud dato | dato | .

El archivo de léxico global trabaja en conunjunto con un archivo de duplas, temporal para cada documento, almacenando los términos de cada documento y todas sus apariciones. Cada registo, de longitud fija, del archivo de duplas tiene 2 campos : | Offset_termino | Offset_documento | (el offset del termino es el offset en bytes del archivo de léxico).



Administración de listas invertidas

La clase encargada de administrar el archivo de listas invertidas es la llamada *ListasInvertidas.*



El archivo de listas invertidas es una archivo de bloques de registros de longitud variable, donde cada registro tiene la siguiente estructura:

Id lista	Cant docs leidos	Cant docs lista	Pares frecuencia documento

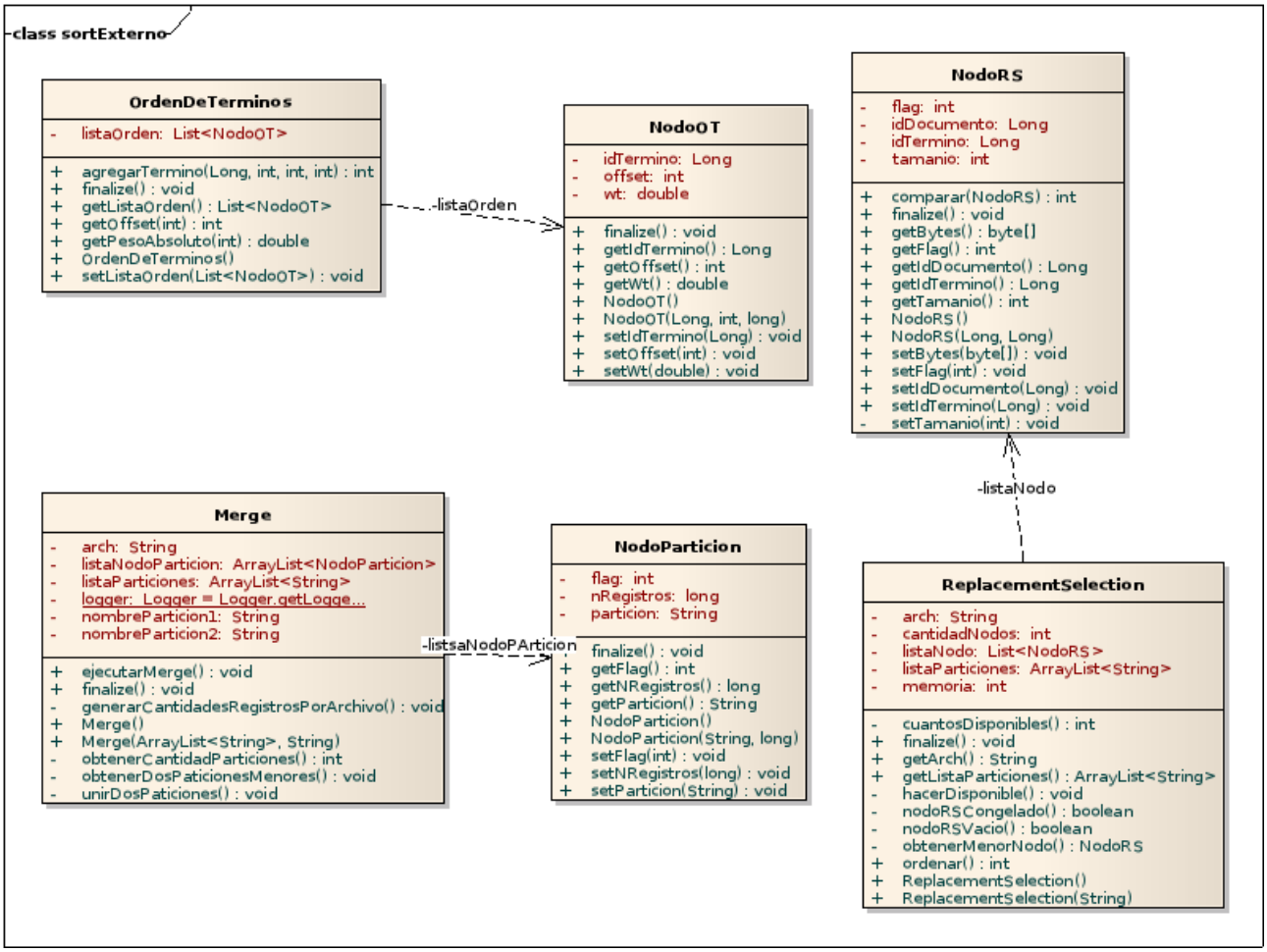
- Id listas: identificador de la lista
- Cant docs leídos: cantidad de documentos leídos en el bloque, sirve de control por si el registro se guarda en más de un bloque.
- Cant docs lista: cantidad total de documentos de la lista invertida.
- Pares frecuencia documento: contiene todos los pares de frecuencia, documento para la lista invertida.

Creación y actualización de listas invertidas

Para la generación y actualización de las listas invertidas se implemento un metodo ordenamiento de sort externo (Merge). El metodo consiste en tomar 2 particiones a ordenar, ordenar cada partición por separado, y luego mezclar ambas partes, manteniendo el orden, en una sola partición ordenada.

Para llevarlo a cabo se crearon las siguientes clases

Diag. Clases Sort Externo



Merge

Esta clase es la encargada de ejectuar el merge. Genera y controla todo el procedimiento de merge de las particiones disponibles.

Orden de Terminos

Contiene la lista de teminos ordenados (secuencia de NodoOT), y mantiene una inserción ordenada por peso global. Se utiliza para ordenar la partición.

ReplacementSelection

Esta clase es la encargada de manejar las particiones a ordenar del archivo. Es la responsable de verificar que particiones se encuentran disponibles.

NodoParticion

Estructura de la partición. Esta compuesta por:

- Nombre de la partición.
- Cantidad de registros en la partición.

Casos de uso del Sistema de Recuperación de Texto

La solución propuesta para el sistema FTRS cubre 2 casos de uso :

- Resolución de una Consutas por ranking.
- Carga de nuevos documentos al sistema FTRS (Indexación de documentos).

Resolución de Consultas por ranking

Para la resolución de una consulta se siguen los siguientes pasos:

1. El usuario ingresa la consulta
2. La consulta es administrada por el Core
3. El Core delega la solución de la consulta al Sistema FTRS
4. El Sistema FTRS toma la consulta
5. Por cada término relevante de la consulta, el Sistema FTRS se encarga de buscar en el arbol para verificar la existencia de ese término.

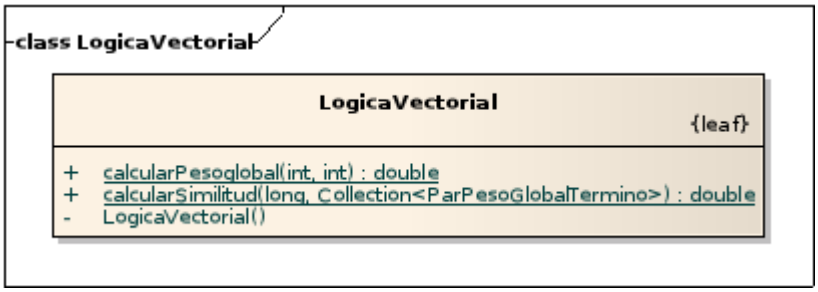
6. Por cada termino que se encuentra se pide la lista invertida correspondiente
7. Cuando ya no quedan mas término por buscar, se devuelve el control al FTRSManager con los términos y listas invertidas correspondientes.
8. Este calcula el peso global de todos los terminos y la similitud del documento de la consulta, y, devuelve la lista una cantidad de documentos prederterminada, y configurable, de documentos ordenados por similitud.

Carga de nuevos documentos al sistema FTRS

Para la carga de un nuevo documento al sistema FRTS se siguen los siguientes pasos:

1. Se busca la existencia del termino dentro del arbol FTRS.
2. Si no existe el término a agregar se inserta en el arbol (ver sección Arbol B# para más información
3. Se reordenan las listas invertidas utilizando el merge externo explicado anteriormente.

Lógica utilizada para la implementación del Modelo Vectorial



Cálculo de Peso Global

Para el cálculo del peso global de un término se utilizo la fórmula:

- **pg(th) = log10(n/fth)**

Cálculo de Similitud

Para el cálculo de similitud de un documento y la consulta se empleó la siguiente fórmula:

- **Similitud(dj,c) = Sumatεc(p(t, dj) * log(n / ft)).**

No se toma en cuenta la normalización de los pesos.