UNIVERSIDAD DE SANTIAGO DE CHILE
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA EN INFORMÁTICA

LENGUAJE Y MÉTODOS DE PROGRAMACIÓN

BUSCADOR EN C#

Profesor: Roberto González
Alumno: Roberto Orellana

Fecha de Entrega: 11/12/2017

Santiago de Chile

11 - 12 – 2017

TABLA DE CONTENIDOS

CAPÍTULO 1. INTRODUCCION

El siguiente informe trata de explicar cómo llegar a la solución del laboratorio N°4 de Lenguaje y métodos de Programación.

1.1 ANTECEDENTES Y MOTIVACIÓN

El laboratorio consta de crear un Buscador en cuatro lenguajes distintos de programación, siendo este el último en Lenguaje C# que incluye los paradigmas de Orientado a Objetos, Concurrencia y Eventos. Es un lenguaje bastante completo, eficiente, potente y amigable para el programado.

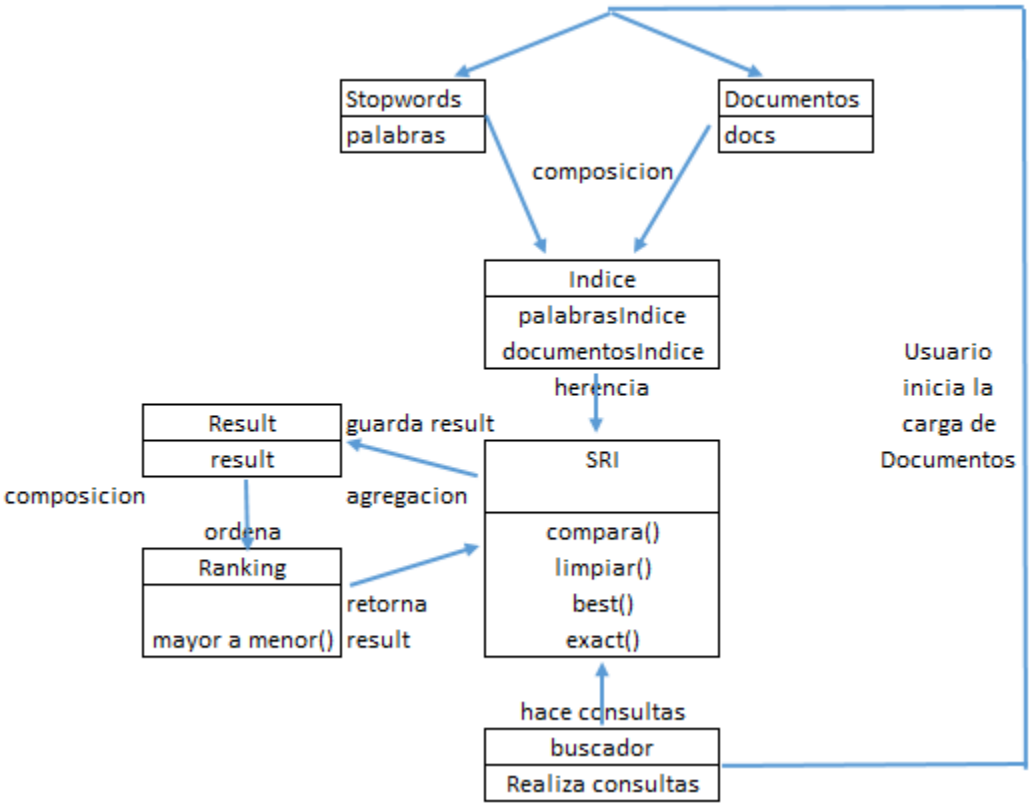
1.2 DESCRIPCIÓN DEL PROBLEMA

El Problema consiste en crear un sistema de recuperación de Información en el lenguaje multiparadigmatico de C# el cual consiste en cargar Stopwords en memoria, cargar documentos, crear un índice y almacenarlo en memoria, hacer una consulta de alguna palabra o frase completa, puede ser del tipo exactmatch o bestmatch y mostrar el resultado de los documentos de mayor a menor según las palabras encontrada.

1.3 SOLUCIÓN PROPUESTA

Para obtener una solución real al problema que se propone, se estudió cuidadosamente los requerimientos funcionales y no funcionales pedidos en el enunciado, de esta forma, se logra detallar las funcionalidades que este contendrá de forma correcta.

Crear un UML Inicial



Almacenar Stopwords

Debido a que se trata de un buscado, la primera problemática que presenta el proyecto es la eliminación de las palabras comunes que se encuentran en el documento y en las palabras que consultara el usuario. Teniendo en cuenta que la eliminación de estas palabras permite una búsqueda mucho más eficiente de las palabras.

Almacenar Documentos y crear Índice.

Para el almacenamiento de Documentos la representación elegida fue la de una matriz en el cual una dimensión representa la cantidad de documentos y la otra dimensión representa la secciones del documento (id, titulo, autor, biblio, texto) y la creación del índice la representación elegida fue la de matriz dispersa en el cual una dimensión representa la cantidad de palabras y la otra dimensión los documentos, almacena un contador cada vez que encuentra una incidencia de una palabra en un documento, un arreglo de string donde se almacena cada una de las palabras sin repetirse.

ExactMatch y BestMatch.
Son tipos de búsquedas en la cual ExactMatch nos permite encontrar aquellos documentos que contengan todas las palabras de una frase, pero en el mismo documento. Y BestMatch nos permite encontrar aquellos documentos o más que contengan uno o más términos de una consulta de tipo frase

Consultar y Mostrar Resultados.
La creación de consultas es muy parecida a la de la creación del índice donde elimina las palabras comunes que se encuentran en las Stopwords, dependiendo del tipo de consulte (Best o Exact) se consultan las palabras y se retorna la cantidad de veces que se repiten en cada documento gracias a la matriz dispersa. Los resultados son ordenados en un ranking de mayor a menor por la cantidad de palabras encontradas en cada documento, luego los resultados llegan de forma aleatoria en una lista debido a la concurrencia.

1.4 OBJETIVOS Y ALCANCES DEL PROYECTO

1.4.1 Objetivo General

El Objetivo general del proyecto consta en lograr el correcto funcionamiento del programa en el lenguaje C#, utilizando lo aprendido en clases y laboratorio sobre paradigma de programación Orientado a Objetos, Concurrencia y Eventos.
Conocer el correcto uso de cada clase utilizada, los objetos instanciados, separar en Backend del Frontend y la relación entre cada una de las clases.

1.4.2 Objetivos Específicos

Lograr que el usuario pueda interactuar con el buscador, cargar Palabras comunes para que sean ignoradas por el buscador, cargar los documentos y crear un índice limpio libre de Stopwords y caracteres que compliquen la búsqueda, hacer una consulta de una frase compuesta o palabra, y mostrar el resultado de la búsqueda en un ranking de mayor a menor según la cantidad de palabras encontradas en cada documento

1.4.3 Alcances

Obtener un correcto funcionamiento del programa gracias a los objetivos y problemas que se plantearon anteriormente.

1.5 METODOLOGÍA Y HERRAMIENTAS UTILIZADAS

Las herramientas y técnicas utilizadas para la solución de final del problema constan de lo aprendido en clases, investigación propia del alumno y las metodologías enseñadas por el profesor, analizando el problema y teniendo una abstracción de cómo resolverlo, iniciando por reconocer cada una de las clases, sus métodos y atributos y luego la creación de un diagrama de clases (UML). La aplicación utilizada en mi caso fue la de C# de .Net que gracias a su fácil compilación y ejecución hace que sea una interfaz más amigable para el programador.

CAPITULO 2. FUNDAMENTOS TEORICOS

2.1 definiciones

- **Paradigma de programación Orientada a Objetos**: es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos manipulan los datos de entrada para la obtención de datos de salida específicos, donde cada objeto ofrece una funcionalidad especial.

- **Clase:** Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella.

- **Herencia**: la Clase padre hereda todos sus atributos y métodos a la clase hija, Por ejemplo, herencia de la clase C a la clase D, es la facilidad mediante la cual la clase D hereda en ella cada uno de los atributos y operaciones de C, como si esos atributos y operaciones hubiesen sido definidos por la misma D.

- **Objeto:** Instancia de una clase. Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos), los mismos que consecuentemente reaccionan a eventos. Se corresponden con los objetos reales del mundo que nos rodea, o con objetos internos del sistema (del programa).

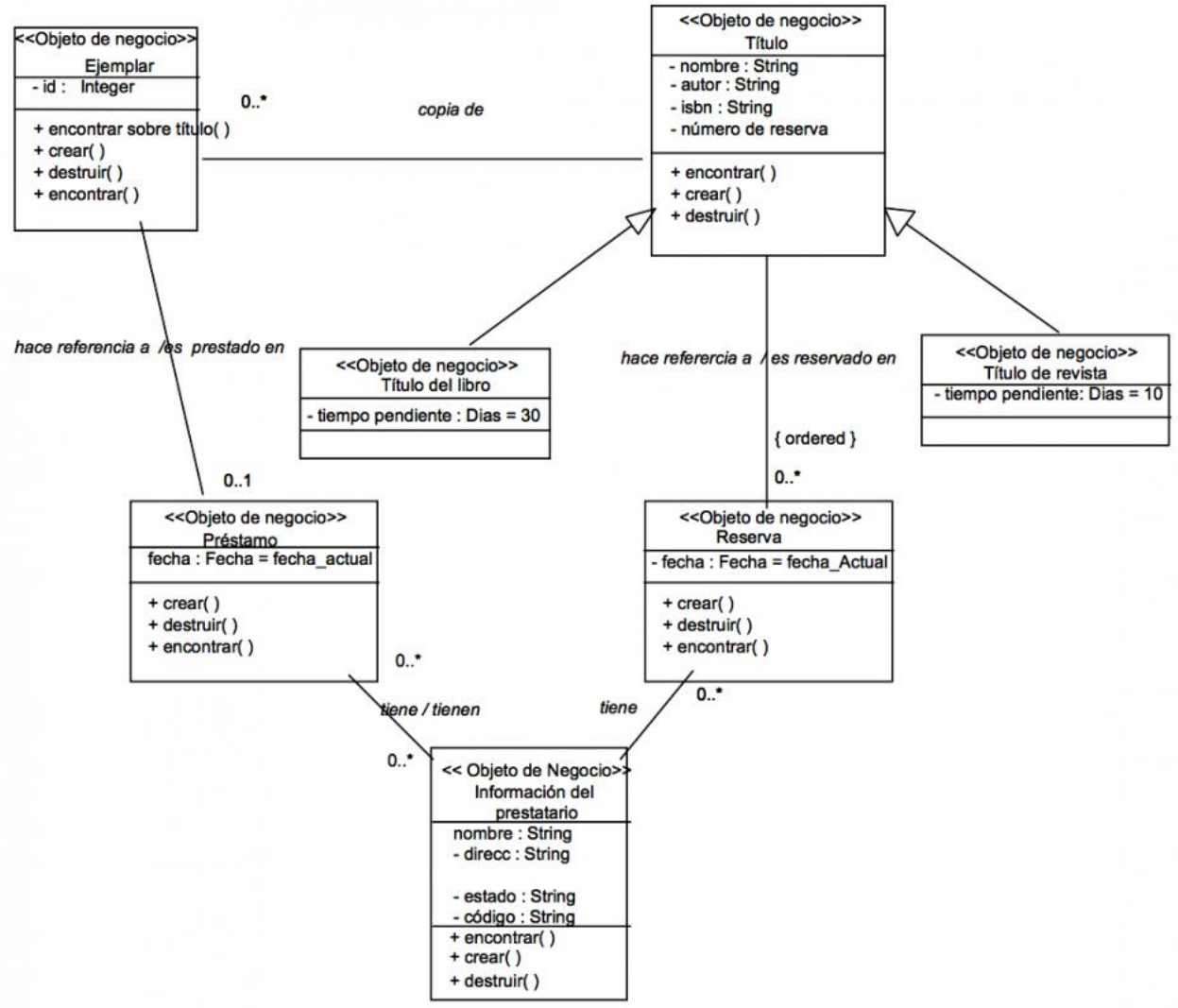
- **Programación Concurrente:** La computación concurrente es la simultaneidad en la ejecución de múltiples tareas interactivas. Estas tareas pueden ser un conjunto de procesos o hilos de ejecución creados por un único programa.

- **Sección Critica:** porción de código donde se accede a un recurso compartido que no debe ser accedido por más de un proceso o hebra a la vez o puede provocar inconsistencia en los datos

- **Exclusión mutua:** Evita el ingreso a la sección critica por más de un proceso a la vez
- **Programación dirigida por Eventos:** es un paradigma de programación en el que tanto la estructura como la ejecución de los programas van determinados por los sucesos que ocurran en el sistema, definidos por el usuario o que ellos mismos provoquen
- **Delégate:** Puntero a función
- **Evento:** Es el que notifica que algo ocurrió (uno a otros)
- **Manejador de Eventos:** es la acción en respuesta a 1 evento

2.2 Diagrama de Clases

Ejemplo índice biblioteca:



2.3 Métodos Utilizados

Los métodos usados dentro de la organización de un algoritmo que realice un procedimiento correctamente son diversos.

Si el algoritmo es demasiado extenso para ser abordado en su totalidad, se divide en sub problemas que permitan tener soluciones parciales, en donde se aplica una especie de principio de superposición, o sea, la suma de las soluciones, resultando la solución general del algoritmo.

CAPITULO 3, DESARROLLO DE LA SOLUCION

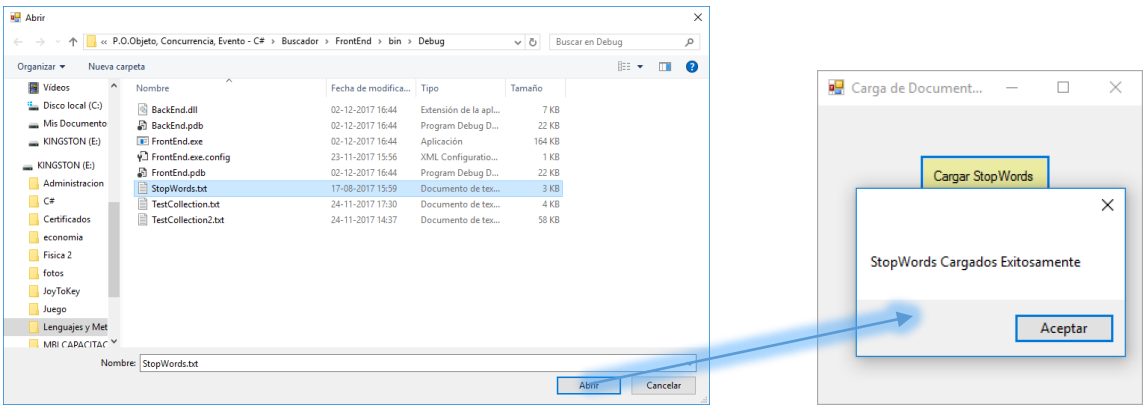
3.1 Análisis

En el análisis, la deducción principal fue la creación de Clases que leerán y almacenaran las palabras de las StopWords y Documentos, además la creación del Índice de matriz dispersa, que gracias a estos se podrá trabajar con los tipos de consultas, ya sea ExactMatch o BestMatch y realizar consultas de palabras o frases completas.

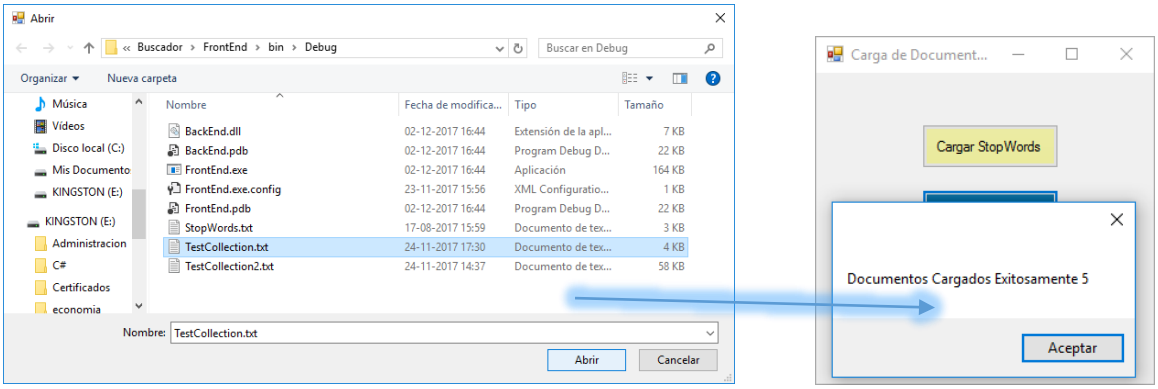
3.2 Diseño

Está diseñado básicamente con un menú de opciones que permite ingresar a diferentes funcionalidades del programa, por ejemplo:

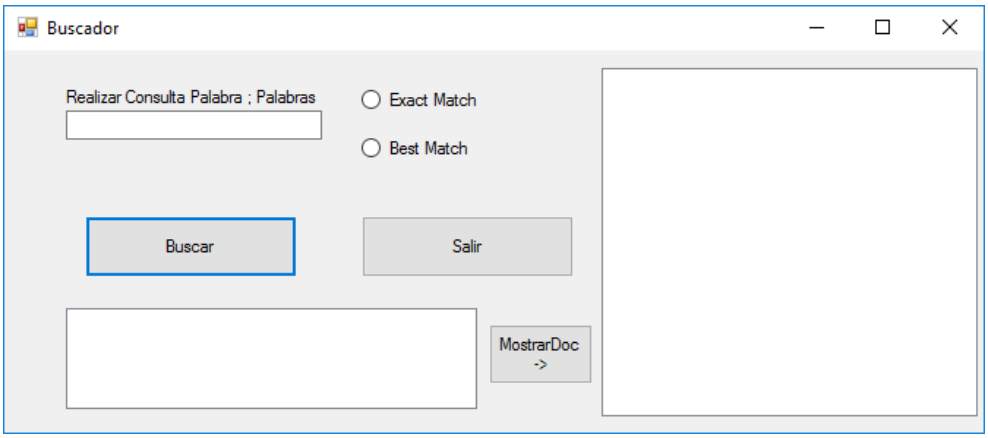
Cargar StopWords



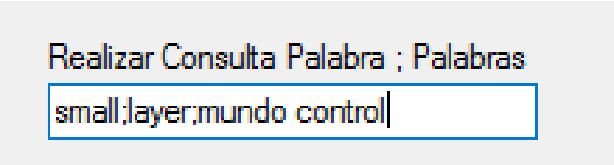
Cargar Documentos



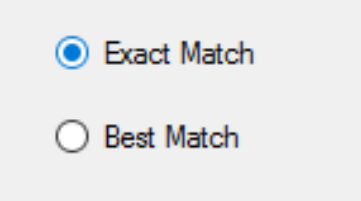
Buscador



Cuadro de consultas

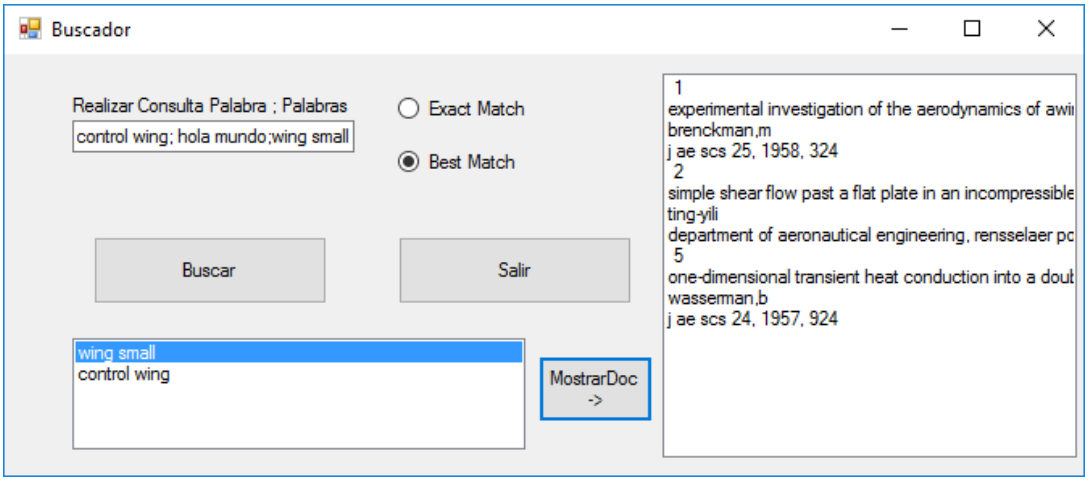


Opciones de Búsqueda (ExactMatch o BestMatch)



Lista de palabras o frases encontradas

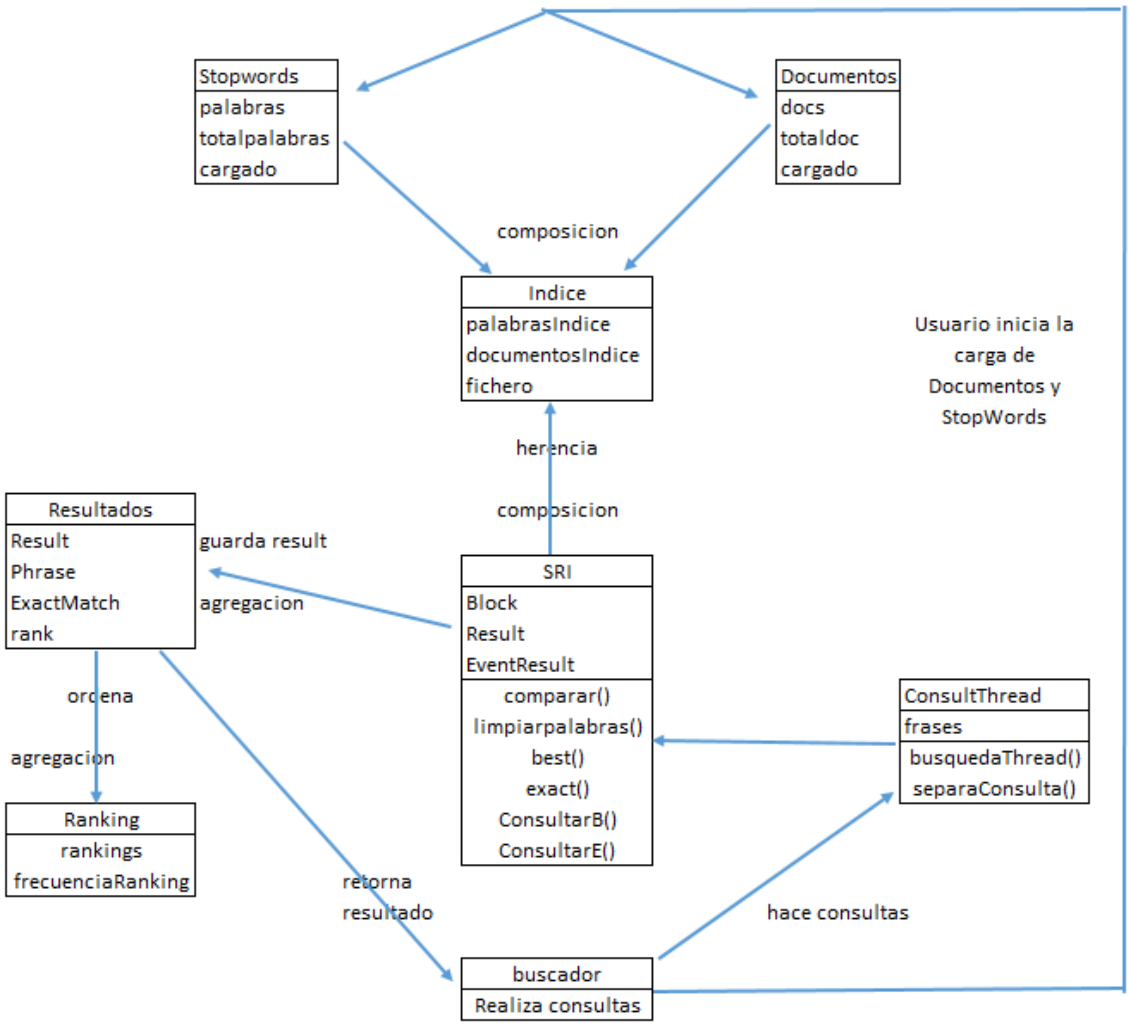
Mostrar Documentos por Ranking



3.3 Estructura del proyecto

La estructura del proyecto consta de 1 solución que contiene dos proyectos, el Backend y el Frontend, el proyecto realiza búsquedas a través de hebras para una búsqueda más práctica, y cada vez que encuentra alguna incidencia con el índice un evento notifica al Frontend para que este envié la respuesta a la interface del usuario
El programa no permitirá avanzar mientras no se carguen los archivos de Stopwordss y Documentos.

UML Final



CAPITULO 4. RESULTADOS

Se da cumplimiento a los objetivos del proyecto, creando las funciones requeridas; leer y almacenar las Stopwords y Documentos, crear un índice a partir de un archivo de entrada y limpiar el archivo eliminando las palabras comunes, realizar consultas concurrentes del tipo Exactmatch o Bestmatch de frases o palabra y eliminar las palabras comunes de la consulta, ordenar top k documentos de mayor a menor, generar un evento donde muestra cada una de las consultas realizadas y sus respectivos resultados.

CAPITULO 5. CONCLUSION

Gracias a este laboratorio logre adquirir más conocimientos sobre el lenguaje de programación C# y los paradigmas orientado a Objetos, concurrente y dirigida por eventos. en cuanto a los recursos de cada paradigma, como por ejemplo: Clases, métodos y atributos, Objetos, herencia, concurrencia, hebras, secciones críticas, eventos, manejador de eventos delégate, diagrama de clases, frontend y backend.

Es un leguaje y multiparadigma muy agradable de trabaja que nos permite hacer muchas cosas solo modelando objetos del mundo real

CAPITULO 6. REFERENCIAS

Nacho C, 2009, ***Introducción a la Programación con C#.***
http://www.tutorialesprogramacionya.com/csharpya/
http://www.udesantiagovirtual.cl/moodle2/course/view.php?id=9365
https://www.youtube.com/watch?v=FeeOntta6mk&list=PLhbcXfA7vHX64kzwPvdHGOsiPh9q3IOhF