Universidad de las Ciencias Informáticas
Facultad 6

Título: Componente para la indexación y búsqueda contextual de información audiovisual.

Trabajo de Diploma para optar por el título de ingeniero en ciencias informáticas.

Autor:
Grechin Guzmán González
Tutor:
Ing. Cesar Santos Sanabria
Co-tutor:
Lic. Alexander Fernández Castro

La Habana, junio de 2011
"Año 53 del Triunfo de la Revolución"

"…aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos…"

CHE

DECLARACION DE AUTORIA

Declaro que soy el único autor de este trabajo y autorizo al <nombre área> de la Universidad de las Ciencias Informáticas; así como a dicho centro para que hagan el uso que estimen pertinente con este trabajo.

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

Grechin Guzmán González

_____

Firma del Autor

Cesar santos Sanabria

_____

Firma del Tutor

OPINIÓN DEL USUARIO DEL TRABAJO DE DIPLOMA

El Trabajo de Diploma, titulado **< Componente para la Indexación y Búsqueda Contextual de Información Audiovisual** fue realizado en **la UCI**. Esta entidad considera que, en correspondencia con los objetivos trazados, el trabajo realizado le satisface

- ☐ Totalmente
- ☐ Parcialmente en un ____ %

Los resultados de este Trabajo de Diploma le reportan a esta entidad los beneficios siguientes (cuantificar):

_____
_____
_____
_____
_____
_____
_____
_____

Como resultado de la implantación de este trabajo se reportará un efecto económico que asciende a **<valor en MN o USD del efecto económico>**

Y para que así conste, se firma la presente a los ____ días del mesde _____ del año _____.

_____ _____

Representante de la entidad Cargo

_____ _____

Firma Cuño

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: **<título del trabajo de diploma>**
Autor: **<nombres y dos apellidos del o los autores>**

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

<Aquí el tutor debe expresar cualitativamente su opinión y medir (usando la escala: muy alta, alta, adecuada) entre otras las cualidades siguientes:

- **Independencia**
- **Originalidad**
- **Creatividad**
- **Laboriosidad**
- **Responsabilidad>**

<Además, debe evaluar la calidad científico-técnica del trabajo realizado (resultados y documento)y expresar su opinión sobre el valor de los resultados obtenidos (aplicación y beneficios) >

Por todo lo anteriormente expresado considero que el estudiante está apto para ejercer como Ingeniero Informático; y propongo que se le otorgue al Trabajo de Diploma la calificación de **<nota>**. **<Además, si considera que los resultados poseen valor para ser publicados, debe expresarlo también>**

<Nombre tutor>

_____ _____
Firma Fecha

Datos de contacto

Autor: Grechin Guzmán González

Correo electrónico: gguzman@estudiantes.uci.cu

Tutor: Ing. Cesar Santos Sanabria.

Correo electrónico: csanabria@uci.cu

Graduado del 2010 con Titulo de Oro en la carrera Ingeniería en Ciencias Informáticas con un promedio de 4.82. Imparte la asignatura Administración del Sistema Operativo GNU/Linux de segundo perfil. Se desempeñó como jefe de programación del proyecto Factoría por varios meses y actualmente se desempeña como jefe de módulo en el proyecto Video Vigilancia.

Co-tutor: Lic. Alexander Fernández Castro

Correo electrónico: alexanderfc@uci.cu

Graduado en el año 2003 en la carrera Ciencia de la Computación. Se ha desempeñado como profesor en las asignaturas de Programación II, Programación III, Programación IV, Teleinformática I y Teleinformática II, Instructor de varios postgrados de Redes, Servicios Telemáticos y Linux, profesor de asignaturas de perfil en la Facultad 2; entre otros. Con mas de 20 años de experiencia en la programación con lenguajes de alto nivel, entre ellos Object Pascal, C++, C# y mas recientemente Python, una experiencia de unos 14 años en los sistemas operativos GNU/Linux, mas de 6 años en el procesamiento de grandes volúmenes de trazas de seguridad y alrededor de 4 años en la Administración de Servicios Telemáticos. Actualmente se desempeña como Especialista General en la Dirección de Redes y Seguridad Informática.

Agradecimientos

Este es el único momento de mi tesis, en el cual no hablo de buscadores, de indexación, de búsqueda, metadatos y descriptores. Es el momento para agradecer de todo corazón, a todas estas personas que también hicieron suyo, mi sueño y es por ello que voy a intentar darle la máxima importancia, porque creo que se lo merece. Muchas gracias a Dios y a todos ustedes. Agradezco a:

A mi Madre y a mi Padre por ser los principales tutores de mi vida. Por apoyarme y confiar en mi todos estos años. Gracias por la educación que me dieron, por inculcarme los valores que me dieron, por haberme hecho como soy. Por llorar y reír conmigo en todo los momentos que lo precisaron. Por todas las veces que me he sentido apoyada, atendida y confiada junto a ustedes. Les debo todo lo que soy. La vida no me alcanzaría para mostrarles mi gratitud. Los amo. Muchas gracias.

A mis Tíos Sucel y Migue, muchas gracias por ser mis segundos padres, por aconsejarme y cuidarme como su propia hija. Por su constante insistencia, por sus constantes atenciones, porque el saber que personas tan luchadoras por sus sueños, son mis tíos, me hacen sentir la sobrina más orgullosa del mundo. Gracias por ayudarme a lograr también, mi sueño.

Agradezco a mi familia en general, por siempre estar pendientes todos de mí, a mis tíos queridos y a todos mis primos. En especial a mi abuela Eutimia, gracias abuelita por quererme tanto y por mostrar que el lazo que nos une es más fuerte que el de sangre. Y a mi primo Ale, gracias primito por estar conmigo siempre.

Alexander, muchas gracias mi amor por cruzarte en mi camino, por ser esa persona especial que cambió mi vida en tan poco tiempo. Por el tiempo que me has dedicado y tus buenos consejos y enseñanzas. Gracias por mostrarme mí verdadero camino y porque me haces olvidar todo y ser feliz. Gracias por hacer tuya también, esta tesis. Las palabras no me alcanzan para agradecerte todo lo que has hecho por mí. Gracias

Agradecimientos

Agradezco a mis Hermanas del alma, amigas inseparables que han estado incondicionalmente conmigo a desde el momento exacto que entraron en mi vida (Gelin, por estar conmigo desde la niñez y ser esa amiga inseparable, a Arelis por enseñarme el verdadero concepto de la amistad y por no olvidarme nunca a pesar de la distancia, a Sol por estar siempre conmigo, en las buenas y en las malas y brindarme esa amistad tan pura a lo largo de estos 5 años y a Mailen, por tu sinceridad y por ser esa persona tan incondicional y desinteresada que admiro mucho,ah y por ser la mejor deportista de la UCI)

Agradezco a mis amigas del apartamento 107102 (la Miche, la Claria, la Pina, a mi Negrita, a Yanecita y a la Lusme). Gracias a todas por hacer de este curso el mejor de todos mis años. Las quiero mucho.

Agradezco a Anyer, por compartir conmigo todos estos años, por ser tan paciente y perseverante conmigo. Por brindarme todo su amor, cariño y amistad.

Agradezco a Noida y a Robert por todo lo que han hecho por mí, por aceptarme en su familia y quererme como una hija, por todas sus atenciones conmigo y todos los buenos momentos que juntos pasamos.

Agradezco a Delmis, Felix y a Yasnary por ser buenos amigos y estar siempre que los necesité. Los quiero.

Agradezco a Pacheco, Sandy, Jose y a Reynier, por ayudarme cuando me hacía falta, además de todos los consejos que me brindaron cuando los necesité. Gracias por ayudarme a construir mi sueño.

Agradezco a todos mis compañeros y amigos, a los que les ha tocado avanzar, adelantar, quedarse atrás, estudiar, no estudiar, ir a clases y demás hazañas conmigo, muchas gracias. Estos años, gracias a ustedes fueron los más bonitos, rápidos y divertidos. Muchas gracias por todo lo compartido.

Por último y no menos importante, agradezco en general a todas las personas que de una forma u otra me apoyaron en algún momento determinado.

A todos ustedes, infinitas gracias.

Grechin

Dedicatoria

Con todo el amor del mundo les dedico mi tesis, a ustedes, las personas más importantes en mi vida.

A mi madre, a mi padre y a mi hermano.

Resumen

El avance en las tecnologías para el almacenamiento de datos y el aumento de la información multimedia existente en la actualidad, han dado lugar a un creciente número de bibliotecas digitales, en estas se almacena tanto información textual, como información audiovisual, es decir: imágenes, audio y videos. La organización, recuperación y búsqueda de dicha información se hace una tarea de gran dificultad.

El objetivo de este trabajo es diseñar e implementar una solución eficiente para la recuperación de información multimedia. Esta centrado en la recuperación de archivos de audio y video y hace uso de las ventajas ofrecidas por los sistemas de recuperación y los motores de reconocimiento del habla. Con el fin de indexar los archivos audiovisuales, será extraído de ambos el audio, este a su vez será convertido en texto y así será más fácil su indexación y posterior búsqueda.

Para la elaboración del sistema se utilizó el lenguaje de programación C++ y RUP como metodología de desarrollo del software. El componente fue desplegado en el sistema operativo Linux, utilizando Debían como distribución, haciendo uso de las actuales tecnologías libres. La realización de este trabajo ha posibilitado obtener un producto para contribuir al desarrollo de sistemas de búsqueda e indexación más eficientes y flexibles que permitan cumplir las necesidades actuales.

Palabras Claves:

Búsqueda, Indexación, Información multimedia, Recuperación de información.

Abstract

The technology advances in data storage and the increasing of the multimedia information available today, have led to a growing number of digital libraries, these containers holds textual information and also audiovisual information, i.e. images, audio and video. The arranging, searching and retrieval of that information usually becomes in a very difficult work.

The aim of this work is to design and implement an efficient solution for multimedia information retrieval. It focuses on the recovery of audio and video files, and makes use of the advantages offered by the recovery systems and speech recognition engines. In order to index the audiovisual archives, from both audio and video will be extracted the associated speech, this will be converted to text and it will be easier to be indexed for the subsequent search.

In this system the language used was C++ and RUP as software development methodology. The component was deployed over one open source platform, specifically Debian Linux system. This work helped to finish a product to contribute on the search and indexation systems, been this one flexible and efficient according to the actual needs.

Key words:

Search, Indexation, multimedia information, data retrieval

Índice

Indice

Índice de Figuras

Introducción

El avance en la tecnología para el almacenamiento de datos y el aumento de la información multimedia existente en la actualidad, han dado lugar a un creciente número de bibliotecas digitales, en estas se almacena tanto información textual, como información audiovisual, es decir imágenes, audio y videos. La organización, recuperación y búsqueda de dicha información se hace una tarea de gran dificultad.

El concepto de búsqueda ha estado presente desde los inicios de la informática hasta nuestros días. Los primeros algoritmos de búsqueda consistían en localizar datos almacenados en una pequeña base de datos local. Así surgieron los primeros sistemas de búsqueda, cuyo objetivo fundamental consistía en acelerar el proceso de búsqueda de cualquier información en una base de datos. Estos sistemas solo realizaban búsquedas textuales, no eran capaces de realizar búsquedas de contenido.

Con el surgimiento de Internet y las bases de datos distribuidas[1] se hizo necesario introducir un nuevo concepto de búsqueda, aquel que sea capaz de registrar información ordenadamente independientemente de que se solicite su búsqueda o no. Este proceso es conocido como indexación. Con su aparición se aceleró considerablemente el proceso de búsqueda. Así llegamos al concepto de sistema de búsqueda existente en la actualidad: un procedimiento informático que con la ayuda de su spider[2] busca archivos almacenados en las bases de datos.

La recuperación de información tiene como objetivo principal satisfacer las necesidades de información de un usuario expresadas en lenguaje natural. Como resultado le devuelve al usuario una serie de informaciones ordenadas según un grado de relevancia. La recuperación de la información se basa en la búsqueda continua en la web o en servidores locales.

Los sistemas de recuperación de información son tecnologías robustas y flexibles a la vez .Sin embargo con los avances de las tecnologías de la información y las comunicaciones, estos sistemas están llegando a un punto crítico debido a la aparición desmesurada de contenido audiovisual en Internet. Las

[1] Es un conjunto de base de datos relacionadas, las cuales se encuentran distribuidas en diferentes espacios lógicos
[2] Programa informático que inspecciona las páginas y crea una copia de cada página visitada para su procesado posterior

tecnologías maduras que se utilizan para localizar, indexar y buscar contenidos textuales no son válidas para estos nuevos contenidos audiovisuales que suponen un problema de otra naturaleza. En este sentido la información multimedia es difícil de analizar y comprender por un programa informático, mucho más que el texto. Por lo que se hace necesario realizar sistemas capaces de enfrentarse a estos retos que están imponiendo los avances de la ciencia.

Tras todos estos cambios, el concepto de sistema de recuperación de información cambiará radicalmente dejando obsoleto al actual vigente. Las ideas concebidas sobre documentos textuales no tienen en cuenta las futuras posibilidades técnicas que vendrán de la mano del lenguaje y de las entradas audiovisuales. En los sistemas documentales existe siempre una descripción textual que aporta información sobre cada documento, los describe y los representa facilitando su indexación. Mientras que en los documentos audiovisuales esta descripción se realiza mediante el uso de metadatos y descriptores, por lo que la búsqueda, organización y recuperación de la información multimedia deriva en sistemas de recuperación con una nueva perspectiva.

Después de estudiar el contexto actual en el que se percibe un gran necesidad tecnológica que hace inminente un nuevo encausamiento para realizar la indexación y la búsqueda contextual de la información audiovisual de manera más simple y automática. Se determinó que el **problema científico** estaría enmarcado en: El aumento del volumen de información multimedia en las bibliotecas digitales hace más difícil su búsqueda y organización**.**

Se define como **objeto de estudio** el proceso de análisis de las técnicas de indexación y los métodos de búsqueda contextual. El **campo de acción** se enfoca en la informatización del proceso de identificación y clasificación de las técnicas de indexación y la búsqueda contextual de la información audiovisual.

Como **objetivo general** para lograr solucionar el problema existente se tiene: Implementar un componente que permita la indexación y la búsqueda contextual de la información audiovisual. Para el desarrollo de la investigación se parte de la **idea a defender:** Disponer de un componente que permita indexar y realizar búsquedas contextuales de la información audiovisual, facilitará el proceso de búsqueda y recuperación de dicha información.

Para el cumplimiento del objetivo trazado se definieron las **tareas de la investigación** que se proponen a continuación que ayudaran a que el desarrollo fluya de una forma eficaz:

1. Describir el estado del arte de las técnicas de indexación y búsqueda contextual de la información audiovisual.
2. Valorar algoritmos y librerías que permitan que puedan ser utilizados en el desarrollo del componente.
3. Seleccionar y argumentar la Metodología de Desarrollo de Software a usar en el proceso.
4. Diseñar la solución propuesta.
5. Implementar la solución propuesta.
6. Desarrollar los Casos de Prueba que certifiquen la veracidad de los algoritmos empleados.

El desarrollo de la investigación precisó el empleo de varios **métodos teóricos** y **empíricos.** Entre los métodos teóricos utilizados el **histórico-lógico** el cual permitió estudiar la trayectoria de los acontecimientos relacionados con el tema de la investigación. El **analítico-sintético** se utilizó para analizar toda la información relacionada con la búsqueda e indexación de información audiovisual, para extraer los elementos más significativos existentes en el análisis realizado y elaborar conclusiones.

En calidad de los métodos empíricos se utilizó el **análisis documental** el cual permitió el estudio crítico de la documentación relacionada con la información audiovisual. Otro de los temas usados fue el de la **Modelación**: Fue necesario la modelación de la arquitectura propuesta, es por eso que se utilizó este método para crear modelos con vistas que favorezcan la comprensión de la misma.

La siguiente investigación tiene como propósito fundamental realizar el ciclo completo de desarrollo de la solución propuesta, basándose en la planificación que a continuación se describe:

Capítulo1: “Fundamentación teórica”, se hace una descripción detallada de los conceptos fundamentales vinculados al objeto de estudio. Se valoran las tendencias actuales referentes al campo de acción y se escogen las principales modelos, metodologías y estándares a utilizar en el desarrollo de la investigación. Así como una fundamentación de la elección.

Capítulo2:"Características del sistema", describe los procesos mediante un modelo del negocio, identificando los actores, trabajadores y casos de uso del negocio. Quedan definidas las funcionalidades del sistema a través de los requisitos funcionales y no funcionales.

Capítulo 3:"Análisis, diseño e implementación del sistema", se modelan las clases del análisis, así como los diagramas de clases del análisis. Se plantean los principios a seguir en la implementación del sistema. También se modela el diagrama de despliegue y el de componentes referente al sistema.

Capítulo 4:"Desarrollo de los casos de prueba al sistema", se diseñan los casos de prueba, se seleccionan las pruebas utilizar y se aplican al sistema.

Capitulo1. Fundamentación Teórica

En el presente capítulo se brinda una visión general de todos los conceptos principales correspondientes a la investigación y se definen las metodologías fundamentales para el desarrollo de la misma.

1.1 Tendencias actuales de los sistemas de recuperación de información.

Los avances tecnológicos de las últimas décadas han provocado un aumento considerable en el intercambio de datos. Hoy nos encontramos inmersos en la era de la informática y cada vez son más los métodos para acceder a la información. El proceso de digitalización de los documentos es un claro ejemplo de la revolución del formato de presentación, lo cual ha permitido su acceso y uso por un número incalculable de usuarios.

El adelanto de servicios multimedia ocurridos en el proceso tecnológico y la posibilidad de compartir y distribuir datos a través de las redes de comunicación han acentuado la importancia de herramientas para la recuperación de información multimedia. Las bases de datos de imágenes se emplean en un vasto abanico de aéreas, como son el entretenimiento, el arte, la publicidad, la medicina y la industria entre otros. En todos estos contextos el problema principal está relacionado con la necesidad de un acceso eficiente a la información.

Cada vez más aumenta el contenido audiovisual y su búsqueda, organización y recuperación se dificulta a medida que avanza la ciencia. Los sistemas de recuperación han surgido como consecuencia positiva de estos avances tecnológicos y al referirnos a ellos se hace fundamental tener en cuenta algunos conceptos básicos para un mejor entendimiento del problema planteado.

1.1.1 Conceptos básicos relacionados con el problema planteado.

Componente

Es un objeto o programa reutilizable que efectúa una función específica y que está diseñado para funcionar con otros componentes y aplicaciones. Después de analizar varios de los conceptos referentes a

un componente se llega a la conclusión de que un componente también podría ser denominado como la parte de un sistema capaz de operar independientemente, pero diseñada, construida y operada como parte integral del sistema. (Fuentes, y otros, 2006)

Recuperación de información

Un concepto tan ampliado como lo es el de recuperación de información presenta varias opiniones a nivel internacional. Una de estas opiniones es la del Diccionario Millán de Tecnologías de la Información que considera como recuperación de información al conjunto de "técnicas empleadas para almacenar y buscar grandes cantidades de datos y ponerlos a disposición de los usuarios" (Lon, 1989). Meadow introduce un concepto diferente, piensa que la recuperación de información "se trata de una disciplina que involucra la localización de una determinada información dentro de un almacén de información o una base de datos" (Meadow, 1992).

Pérez-Carballo y Strzalkowski consideran que "una típica tarea de la recuperación de información es traer documentos relevantes desde un gran archivo en respuesta a una pregunta formulada por un usuario y ordenar estos documentos de acuerdo con su relevancia" (Caraballo, y otros, 2000). Salton indica que "la recuperación de información tiene que ver con la representación, almacenamiento, organización y acceso a los ítems de información". (Salton, 1999)

Tomando como base de partida todas las opiniones planteadas se concluye que la recuperación de la información es la forma de satisfacer la necesidad de información de un usuario, normalmente expresada en lenguaje natural, solicitando una información almacenada en una base de datos.(Ver Figura 1)

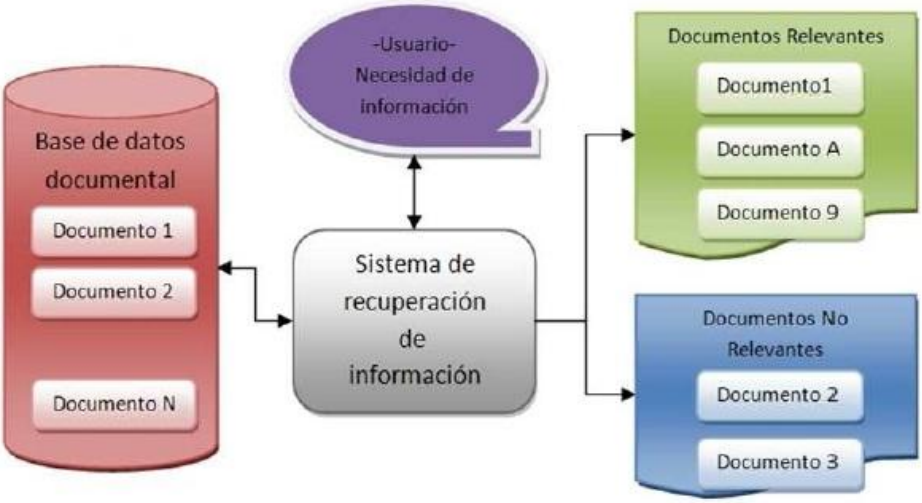


Figure 1: Recuperación de Información

Indexación

En la actualidad, en plena etapa de desarrollo tecnológico ha generado la creación de varios tipos de información, las cuales se deben indexar en las bibliotecas digitales. Como en las bibliotecas tradicionales, gracias a la indexación esta información puede ser ordenada y clasificada según tipo de información y materias. Concretamente la cantidad de información del sector audiovisual se ha masificado en los últimos años, con lo que adquiere mayor importancia la necesidad de indexarla.

La indexación de documentos es la operación más significativa del análisis documental referido al proceso de descripción y representación del contenido de un documento, mediante un número limitado de conceptos extraídos del texto de los documentos (palabras clave) y de vocablos controlados (descriptores, términos o encabezamiento de materia). (Carlos, 2001)

Indexación audiovisual

La indexación audiovisual utiliza un conjunto de técnicas que permiten el etiquetado de los documentos de audio y de video. Habitualmente para la indexación de cualquier tipo de datos se generan metadatos o

etiquetas, que son datos altamente estructurados y que contiene información relevante de cada documento por lo que intervienen positivamente en la indexación de contenido audiovisual.

Metadatos

Son varias las opiniones que existen a cerca de la definición exacta de los metadatos. Según la más difundida son “datos sobre datos” (Jeffery, y otros, 2006), pero además de esta hay varias declaraciones más como “informaciones sobre datos”, “datos sobre informaciones”, y “informaciones sobre informaciones”. Se puede concluir que los metadatos son cualquier información estructurada y descriptiva de los recursos, incluyendo tanto los digitales como los que no lo son.

Los metadatos se encargan de mantener un registro sobre la estructura, el significado, el contexto o el propósito de un objeto, permitiendo extraer y administrar ese objeto. En general los metadatos contienen información concisa del objeto, lo cual le permite a un máquina, su localización, identificación, organización y descripción.

Estándares de metadatos

Son una serie de estructuras o palabras clave que se usan para detallar los datos. Existen distintos tipos de estándares con diferentes esquemas de descripción, pero principalmente detallan una serie de atributos de un objeto asignándole valores que sirven para recuperar el objeto. En general se pueden encontrar metadatos administrativos, descriptores y técnicos. Dentro de estas categorías es relevante destacar los metadatos descriptores, que son especialmente utilizados para manejar recursos digitales, particularmente los recursos multimedia. (Gonzalo, y otros, 2002)

Descriptores

Los descriptores visuales describen las características visuales de los contenidos dispuestos en imágenes, audio y video. Describen características elementales como el color, la forma y la textura o el movimiento. Éstos tienen un buen conocimiento de los objetos y eventos presentes en un contenido audiovisual y permiten la búsqueda rápida y eficiente de la multimedia. (Collada Perez, 2009)

Búsqueda contextual

La búsqueda contextual es una de los aspectos más complejos en el campo de la recuperación de información y cada vez son más avanzados, los métodos utilizados para el análisis del contenido. En el modelo clásico de recuperación de información, el objetivo del empleo de la búsqueda contextual es de encontrar, a partir de una consulta en lenguaje natural, la mayor cantidad de información relevante.

Reconocimiento automático del habla

Se denomina reconocimiento automático del habla al proceso de realizar una transcripción audio-texto de un audio, es decir dado un audio identificar las palabras del mismo en el orden y el tiempo correctos. (IDE, 1990) También es denominado como el proceso automático por el cual un sistema es capaz de identificar una señal de voz producida por un individuo o conjunto de ellos. Esta señal es sometida a procesos de digitalización con el fin de obtener determinadas características y propiedades que permitan posteriores procesos, en este caso la indexación.

1.1.2 Sistemas de Recuperación de Información

Con el fin de recuperar los documentos más significativos y de cumplir con las especificaciones en lenguaje natural, por parte del usuario, se han creado varios sistemas de recuperación de código libre los cuales presentan dos clasificaciones:

Los **sistemas de recuperación basados en texto:** Se han desarrollado con gran éxito en las últimas décadas y son útiles solamente para recuperar coincidencias en tiempo real, y ofrecen la ventaja de indicar la relevancia de un documento con respecto a una pregunta mediante la aplicación de una serie de medidas que valoran la frecuencia de aparición de los términos de la pregunta en el documento completo, además permiten localizar que parte del documento es realmente relevante.

Sin embargo aún no se ha podido lograr que estos sistemas tengan un comportamiento óptimo y buena efectividad a la hora de seleccionar las partes relevantes del documento. Estos sistemas también soportan varias operaciones básicas tales como: introducción de nuevos documentos, modificación de los que ya estén almacenados y eliminación de los mismos.

Los **sistemas de recuperación basados en contenido**: Es decir aquellos que utilizan información intrínseca a cada recurso, son más difíciles de implementar. Los sistemas de recuperación basados en contenido se basan en la búsqueda específica y exclusiva de cada información. Están implementados concretamente para la información de tipo audiovisual.

Estos sistemas usan características de bajo nivel como el color, la textura y la forma para indexar y tienen la capacidad de acceder tanto al contenido como a los metadatos que los describen. Presentan como objetivo principal, encontrar formas automatizadas de creación de resúmenes que permiten extraer y devolver al usuario la información más relevante de cada uno de los contenidos consultados. (Amengual, y otros, 2006)

1.1.3 Análisis de soluciones existentes

Un sistema de recuperación es un proceso donde se accede a una información textual o audiovisual previamente almacenada en una base de datos, mediante herramientas informáticas que permiten establecer búsquedas específicas.

A continuación se describen algunos sistemas de recuperación de información de código libre

Xapian

Es una herramienta de software libre que le permite a los desarrolladores añadir a sus aplicaciones la posibilidad de indexar y realizar búsquedas contextuales. Está escrita en C++, con enlaces para permitir el uso de Perl, Python, PHP, Java, C# entre otros. Es altamente portable y se ejecuta en Linux, Solaris, Microsoft Windows, FreeBSD, NetBSD y OpenBSD.

Sus principales características se describen a continuación:

- Transacciones: si la actualización de base de datos falla en medio de una transacción, la base de datos está garantizada para permanecer en un estado coherente.
- Búsqueda y actualización simultánea, con nuevos documentos inmediatamente visibles.
- Soporte para grandes bases de datos: Xapian ha demostrado ser escalable a cientos de millones de documentos.
- Precisa clasificación probabilística: más documentos pertinentes se muestran primero.
- Frase y búsqueda por proximidad.

- Consultas booleanas estructuradas, por ejemplo, " Raza y ‘condición’ "
- Comodín de búsqueda.
- Corrección de ortografía.
- Solo realiza búsquedas por texto.

Lucene

Es un sistema de recuperación de código libre que permite indexar y realizar búsquedas contextuales de información textual exclusivamente. Entre sus principales características ofrece la de alto rendimiento, presenta algoritmos de búsqueda eficientes, exactos y robustos. Al igual que el Xapian es independiente de la plataforma y se encuentra implementado en varios lenguajes tales como Perl, C++, Python y PHP.

Sus principales características se describen a continuación:

- Alto rendimiento: pocos requerimientos de RAM e indexación incremental.
- Es independiente del formato de ficheros.
- Algoritmos de búsquedas eficientes, exactos y robustos.
- Presenta clases clave utilizadas para construir el motor de búsqueda.
- Solo realiza búsquedas textuales.

Swish-E

Simple Web Indexing System for Humans – Enhanced es un sistema gratuito, flexible y rápido que permite la indexación de colecciones de páginas web u otros archivos. Está diseñado para colecciones de hasta un millón de documentos y utiliza un modelo de recuperación booleano.

Sus principales características de destacan a continuación:

- Indexa una gran cantidad de archivos utilizando filtros.
- Soporta expresiones regulares para seleccionar los archivos a indexar.
- Búsqueda por frases, oraciones y comodines.
- Los resultados de la búsqueda son ordenados por relevancia, por algún campo o propiedad.
- Búsquedas textuales.

Como es evidente estas aplicaciones de recuperación de información son eficientes pero se están quedando obstaculizadas por el contenido audiovisual generado por los adelantos científicos en el mundo.

En la tecnología de la indexación se distingue un hecho clave, la localización de una palabra clave o de una frase en un fichero de audio. La localización de estas palabras sirve de gran ayuda a los usuarios para acceder a partes específicas dentro de un archivo multimedia. El indexado de audio y video conlleva reconocimiento de voz para la inspección total de un fichero ya que las técnicas para el análisis de las imágenes no están lo suficiente avanzadas como para realizar un indexado basado en fotogramas. Por lo tanto se realiza un proceso de extracción del audio ya que será el audio del video, el que representará a dicho video. La indexación basada en texto de documentos multimedia hace imprescindible el uso de sistemas de reconocimiento de voz, es decir sistemas que convierten el audio en texto, facilitando el proceso de indexado de los archivos multimedia que generalmente son los más dificultosos.

A continuación se describen diferentes sistemas de reconocimiento automático del habla comerciales:

Dragon NaturallySpeaking

Dragon NaturallySpeaking es un software de reconociendo de voz desarrollado por Dragon Systems.es uno de los primeros programas en realizar reconocimiento de voz en ordenadores personales (Nuance, 2009).

Principales características:

- Dictado: el lenguaje hablado es transformado a texto.
- Comandos de control: el lenguaje hablado es reconocido como un comando para controlar el ordenador.
- Sintetizador de voz: transformación del texto escrito, en voz.
- Aprende de forma interactiva las características del habla del usuario.
- Crea un texto con una velocidad de hasta 120 palabras por minuto.

Via Voice

Es un programa de reconocimiento del habla desarrollado por IBM, que proporciona una versión que permite su uso en dispositivos móviles (Nuance, 2009).

Principales características:

- Reconoce listas de vocabulario que exceden las 200 000 palabras en tiempo real y en diferentes idiomas.

- Dispone de una arquitectura completamente integrada que proporciona reconocimiento automático del habla, síntesis de voz y otras tecnologías.
- Funciona en diversos ordenadores y sistemas operativos.
- Se encuentra disponible en una gran variedad de idiomas para proporcionar reconocimiento del habla, síntesis de voz y reconociendo del hablante.
- Incluye un editor de gramáticas y plantillas, un compilador de pronunciación y una interfaz de seguimiento y depuración.

Media Mining Indexer.

Media Mining Indexer (MMI) es un software de reconocimiento de voz desarrollado por Sail Labs. Permite procesar voz de múltiples fuentes en varios formatos y produce texto en tiempo real.

Principales características:

- Es un sistema independiente del hablante.
- Una característica destacable es que la salida del sistema de reconocimiento de voz es en XML, por lo que permite su posterior uso en sistema de búsqueda y recuperación de información.
- Permite una conversión precisa de habla espontánea a texto incluso con altos niveles de ruido de fondo.
- Detección del cambio del hablante analizando segmentos de audio para la recuperación de información
- Detección de entidades que permiten encontrar información relevante, como palabras pertenecientes a categorías como personas, lugares, organizaciones y etc.
- Detección del tema.
- Identificación del hablante.

A continuación se describen diferentes sistemas de reconocimiento automático del habla de código abierto:

Sphinx

Es uno de los sistemas de reconocimiento automático del habla de código abierto más interesante, adaptable al español y moderno (Sphinx, 2009). Este sistema utiliza Modelos Ocultos de Markov[3] y funciones de densidad probabilísticas[4].

Principales características:

- Puede trabajar en modo cliente-servidor, de esta manera un cliente puede solicitar servicios de reconocimiento de voz a un servidor que se ejecuta en otro equipo.
- Presenta un módulo de entrenamiento, SphinxTrain, que sirve para generar modelos que se podrán utilizar en diferentes versiones de Sphinx.

Principal desventaja:

El funcionamiento de Sphinx carece de cierta precisión y eficacia. Suele ser muy complejo y difícil de configurar. No existe información clara sobre su funcionamiento.

Versiones 2, 3 y 4 de Sphinx respectivamente

Principales características:

Está orientada a brindar el mejor rendimiento posible en tiempo real, sacrificando en cierto modo la precisión.

Está orientada a brindar la mayor precisión posible con vocabularios extensos.

Presenta las mismas características de Sphinix-3, sin embargo presenta una mejor simplicidad, al estar escrita en Java.

Julius

Es un motor de reconocimiento acústico de dictado continuo en tiempo real, basado en la interpretación de modelos de Markov.

Principales características:

- Permite reconocer voz mediante dictado continuo o mediante una gramática previamente introducida.

[3]Es un modelo estadístico en el que se asume que el sistema a modelar es un proceso de Markov de parámetros desconocidos.

[4] Es una función matemática que caracteriza el comportamiento probable de una población

- Se puede utilizar en aplicaciones en las que sea necesario obtener la transcripción de documentos de audio en tiempo real e incluso como herramienta de dictado.
- Adopta modelos acústicos y diccionario de pronunciaciones en formato HTK[5].
- Adopta modelos de lenguajes basados en trigramas en formato ARPA[6].
- Está diseñado para plataformas UNIX, pero funciona correctamente en Windows.

[5] Es un paquete de programas que permite realizar todas las fases de procesamiento de señales para crear sistemas de reconocimiento de voz con Modelos Ocultos de Markov
[6] Logaritmo para computar las probabilidades.

1.2 Tendencia y tecnologías actuales a considerar

La informatización de la sociedad en los últimos años ha sido el motor impulsor del desarrollo de nuevas tecnologías, metodologías y herramientas para facilitar este propósito. En el ámbito mundial existen herramientas que permiten lograr en tiempo considerable el desarrollo de grandes sistemas manipuladores de magnos volúmenes de información. Además de posibilitar un alto grado de calidad gracias a las facilidades que implementan dichas herramientas.

El principal propósito de estas tecnologías es desarrollar los programas en un marco de trabajo claramente definido y estandarizado, que permita obtener productos que garanticen los requerimientos de calidad, cumplan con las expectativas del cliente, se desarrollen en un tiempo determinado y ajustados al presupuesto. En la actualidad existen ciertos criterios a tener en cuenta como son los lenguajes de programación a utilizar, gestores de base de datos, patrones arquitectónicos y de diseño, metodologías entre otros. A continuación se presenta un resumen de todos estos aspectos antes expuestos, utilizados en la investigación.

1.2.1 Arquitectura

La arquitectura de software es la estructura fundamental de un sistema, representada en sus componentes, las relaciones entre ellos y los principios que orientan su evolución. Toda arquitectura de software debe describir diversos aspectos del software. Generalmente, cada uno de estos aspectos se describe de una manera más comprensible si se utilizan distintos modelos o vistas. En la mayor parte de los casos los sistemas usan estilos arquitectónicos tales como: Modelo vista controlador (MVC), arquitectura basada en objetos, arquitectura basada en servicios (SOA) y arquitectura en n capas.

Sistemas Distribuidos. Modelo Cliente Servidor

Esta arquitectura consiste básicamente en un programa, el cliente informático, realiza peticiones a otro programa, el servidor, que se encarga de darle respuesta. La arquitectura cliente servidor sustituye a la arquitectura monolítica en la que no hay distribución, tanto a nivel físico como a nivel lógico. En ella la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las

ventajas de tipo organizativo debidas a la centralización de la gestión de información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema. (De la Torre, y otros, 2010)

Sus principios claves son:

- El cliente realiza una o más peticiones, espera por las respuestas y las procesa a su llegada.
- El cliente se conecta solo a uno o a un número reducido de servidores al mismo tiempo.
- El cliente interactúa directamente con el usuario, por medio de una interfaz gráfica
- El servidor no realiza ninguna petición al cliente.
- El servidor envía los datos en respuesta a las peticiones realizadas por parte de los clientes conectados.
- El servidor normalmente autentifica y verifica primero al usuario y después procesa la información y envía los datos.

Sus principales beneficios son:

- Más seguridad, ya que los datos se almacenan en el servidor que generalmente ofrece más control sobre la seguridad.
- Acceso centralizado a los datos que están almacenados en el servidor, lo que facilita su acceso y actualización.
- Facilidad de mantenimiento ya que los roles y las responsabilidades se distribuyen entre los distintos servidores a través de la red, lo que permite que un cliente no se vea afectado por un error en un servidor determinado.

Arquitectura en n capas

En el diseño actual de sistemas informáticos se suele usar este tipo de arquitectura. La esencia del desarrollo en capas es el concepto de mantener cada componente tan separado del contexto global como sea posible y cada capa es la agrupación de los componentes que tengan una funcionalidad común. Este concepto es muy importante para el desarrollo de cualquier tipo de aplicación ya sea un software o una aplicación web. (De la Torre, y otros, 2010)

Sus principales características son:

- Descomposición de los servicios de forma que la mayoría de interacciones ocurre solo entre capas vecinas.

- Las capas de una aplicación pueden residir en una misma máquina o pueden estar distribuidos en varios equipos.
- Los componentes de cada capa se comunican con los componentes de otras capas a través de interfaces bien conocidos.
- Cada nivel agrega las responsabilidades y abstracciones del nivel inferior.
- Separa de forma clara la funcionalidad de cada capa.

Principales beneficios:

- Rendimiento ya que distribuyendo las capas en distintos niveles físicos se puede mejorar la escalabilidad, la tolerancia a fallos y el rendimiento.
- Testeabilidad ya que cada capa tiene una interfaz bien definida sobre la que realizar las pruebas y la habilidad de cambiar entre diferentes implementaciones de una capa.
- Independencia ya que elimina la necesidad de considerar el hardware y el despliegue así como las dependencias con interfaces externas.

Arquitectura basada en componentes

Uno de los enfoques en los cuales se trabaja actualmente constituye lo que se conoce actualmente como Desarrollo de Software basado en Componentes, o Arquitectura basada en componentes (Ver Figura 2) que trata de sentar las bases para el diseño y desarrollo de aplicaciones distribuidas basadas en componentes de software reutilizables, este principio promueve que los componentes sean implementados de una forma que permita su utilización funcional sobre diferentes sistemas en el futuro. (De la Torre, y otros, 2010)

Sus principales características son:

- Es un estilo para realizar aplicaciones a partir de componentes individuales.
- Enfatiza la descomposición del sistema en componentes con interfaces bien definidas.
- Define una aproximación al diseño a través de componentes que se comunican mediante interfaces que exponen métodos, eventos y propiedades.

Principales beneficios:

- Fácil despliegue ya que se puede sustituir un componente por su nueva versión sin afectar a otros componentes o al sistema.

- Reducción de costos en el desarrollo y mantenimiento, ya que se pueden usar componentes de terceros.
- Reusables ya que son independientes del contexto, se pueden emplear a otras aplicaciones y sistemas.
- Reducción de la complejidad gracias al uso de contenedores de componentes que realizan la activación y gestión del ciclo de vida.

Figure 2: Arquitectura basada en componentes

Fundamentación de la arquitectura a utilizar.

Luego del estudio realizado a varias arquitecturas se escogió esta última para estructurar el componente, debido a que los continuos avances de las tecnologías están haciendo cambiar la forma en la que se desarrollan las aplicaciones. Esto ha provocado, entre otras cosas que lo modelos de programación existentes se vean desbordados, por lo que se hace necesario un nuevo enfoque, y en el que se trabaja actualmente es en la arquitectura basada en componentes la cual describe un acercamiento al diseño de sistemas como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver el problema.

1.2.2 Metodologías para el desarrollo del software

En todo proceso de desarrollo del software es fundamental un profundo análisis de las metodologías y estándares a utilizar para una ejecución exitosa del software. A nivel internacional varias empresas dedican su producción a la realización de modelos, metodologías y procedimientos estándares que permiten uniformar la filosofía del trabajo en aras de lograr la obtención de un producto de software con calidad. En la actualidad existe gran cantidad de metodologías orientadas al proceso de desarrollo del software, entre las más considerables se pueden destacar el Rational Unified Process (RUP), Extreme Programing (XP) y Microsoft Solution Framework (MSF).

Rational Unified Process (RUP)

Constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos y optimiza su comunicación proporcionando a cada miembro una aproximación al desarrollo de software con una base de conocimientos de acuerdo a las necesidad es específicas del proyecto. RUP se caracteriza por ser **Dirigido por casos de uso** donde los casos de uso definen lo que el usuario desea a partir de la captura de los requisitos y la modelación del negocio. Es **Centrado en la Arquitectura** característica que brinda una visión completa del sistema y se describen los procesos del negocio. **Iterativo e incremental** donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su desarrollo y compresión. (Pressman, 2006)

De forma general el ciclo de vida del RUP, como se conoce al trazado de las actividades de desarrollo del proyecto en el tiempo, está dividido en 4 fases (Ver Figura 3):

- **Inicio** : en esta fase se establece la oportunidad y el alcance del proyecto,
- **Elaboración**: se define una arquitectura base sólida,
- **Construcción**: en esta fase se desarrollan todos los componentes.
- **Transición:** el software desarrollado se publica a la comunidad de usuarios, cada fase corresponde respectivamente a los 4 hitos principales del RUP: ciclo de vida del proyecto, ciclo de vida de la arquitectura, capacidad operacional y release del proyecto, donde cada fase cambia el foco del equipo del trabajo para alcanzar cada uno de los hitos.

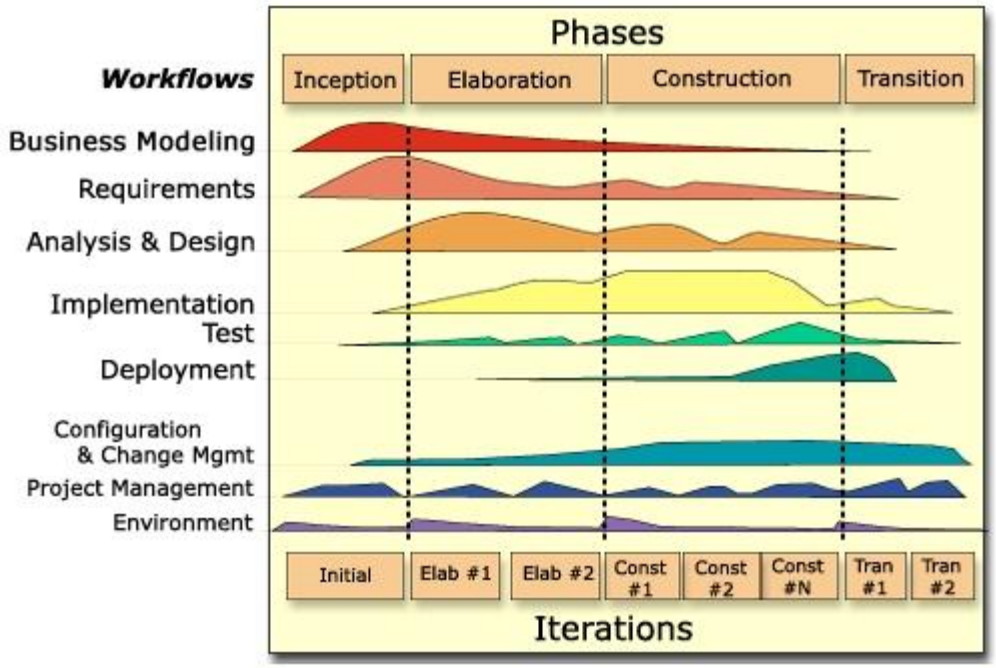


Figure 3: RUP en 2 Dimensiones

Extreme Programing (XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo. Es una de las más usadas en la actualidad y consiste en una programación rápida o extrema, cuya particularidad es tener al usuario final como parte del equipo de desarrollo.

Este método se basa en tres características fundamentales, las **Pruebas Unitarias Continuas**, estas pruebas son establecidas antes del código y se ejecutan constantemente ante cada modificación del sistema. **La Refactorización** es una actividad constante de reutilización del código con el objetivo de remover código repetido, mejorar su legibilidad, simplificarlo y hacerlo un poco más flexible ante los posteriores cambios. **La programación en parejas** consiste en la participación en conjunto de dos programadores, esto posibilita que la tasa de errores al final del desarrollo del software sea más baja, los diseños son mejores y se reduce considerablemente el tamaño del código, además de que permite la transferencia de conocimientos de programación entre los miembros del equipo. (Pressman, 2006)

Microsoft Solution Framework (MSF)

Microsoft Solution Framework es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos. Se centra en los modelos de proceso y de equipo dejando en segundo plano las elecciones tecnológicas.

MSF divide el desarrollo del software en cinco fases (Pressman, 2006):

Visión: es donde se obtiene un panorama del proyecto compartido, comunicado y alineado a los objetivos del proyecto.
Planeación: es donde se planifica un cronograma de trabajo siguiendo las especificaciones de la fase Visión, ajustado al presupuesto, tiempo y recursos.
Desarrollo: es en esta fase donde los miembros del equipo de trabajo elaboran y prueban la solución a implementar y obtiene iterativamente de la fase de Planeación y de la de Estabilización, versiones del producto, entregables y medibles.
Estabilización: se obtiene una versión final del proyecto, probada, ajustada y aprobada en su totalidad.
Implementación: es donde el equipo implementa la solución y sus componentes. Se le entrega al cliente el producto finalizado en su totalidad y se obtiene su aprobación final.

Fundamentación de la selección de la metodología a utilizar.

En correspondencia con el análisis realizado, se ha podido evidenciar que cada metodología presenta una serie de características particulares para diferentes situaciones. RUP es perfectamente competente para proyectos a largo plazo, aunque es caracterizada por ser una metodología compleja. MSF por su parte es adaptable a proyectos de cualquier dimensión y basa su funcionamiento en el control de entregables por parte de los desarrolladores y no en la documentación del software. La metodología XP está destinada para proyectos a corto plazo, es iterativa y está dirigida fundamentalmente a los clientes.

Teniendo en cuenta las particularidades de cada una, se ha seleccionado como apoyo en el desarrollo del software la metodología RUP, debido a varias prestaciones que brinda como son: provee alta calidad del resultado de los sistemas, aumenta la productividad de los desarrolladores mediante el acceso a las bases de conocimientos, documentos y herramientas. Presenta herramientas de apoyo en cada fase del proyecto. Se centra en la producción y tiene un alto potencial de organización.

1.2.3 Herramientas de modelado de los artefactos del RUP

Visual Paradigm

Es una herramienta UML[7] profesional que soporta el ciclo de vida completo del desarrollo del software: análisis, diseño, construcción, pruebas y despliegue. Este software de modelado ayuda a una rápida construcción de aplicaciones de calidad y menos costosas. Permite dibujar todos los tipos de diagrama de clases, código inverso, genera código desde diagramas y documentos.

Presenta características adicionales como:

- Ingeniería inversa: Código a modelo, código a diagrama
- Diagramas de Procesos de Negocio - Proceso, Decisión, Actor de negocio, Documento.
- Modelado colaborativo con CVS[8] y Subversión.
- Generación de código - Modelo a código, diagrama a código.
- Diagramas de flujo de datos
- Generación de bases de datos - Transformación de diagramas de Entidad-Relación en tablas de base de datos
- Generador de informes para generación de documentación

Rational Rose

Es una herramienta UML que proporciona un lenguaje común para el equipo de desarrolladores. (GSI, 2007)

Presenta características adicionales como

- Modelado UML para trabajar en diseños de bases de datos, con capacidad de representar la integración de los datos y los requerimientos de aplicación a través de diseños lógicos y físicos.
- Capacidad de crear definiciones de tipo XML (DTD[9]) para el uso en la aplicación.
- Integración con otras herramientas Rational.
- Capacidad de análisis de calidad de código
- Capacidad para integrarse con cualquier sistema de control de versiones SCC[10].
- Soporte de ingeniería reversa.

[7] Unified Modeling Language
[8] Concurrent Versions System
[9] Document Type Definition
[10] Source Code Control

Fundamentación de la selección de la herramienta de modelado

Se seleccionó Visual Paradigm para el modelado de los artefactos del RUP ya que es una herramienta multiplataforma, además de poseer un entorno de creación de diagramas UML 2.1. Soporta una gama de lenguajes en la generación de código e ingeniería inversa como son Java, C++, Python, C#, Ruby, Delphi y Perl lo cual posibilita el desarrollo del software. Es de fácil uso y permite generar código desde diagramas, así como generar documentación, agilizando el trabajo del desarrollador. Teniendo en cuenta todo lo anterior el Visual Paradigm se convierte en una herramienta potente para el modelado y desarrollo del producto.

1.2.4 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para controlar el comportamiento de una computadora. Está compuesto por un conjunto de símbolos y reglas, la unión de estos dos componentes permite expresar instrucciones que luego serán interpretadas. Atendiendo al número de expresiones que utilicen para realizar una determinada acción, los lenguajes pueden clasificarse en dos tipos: lenguajes de alto nivel, que son los que se asemejan a las lenguas humanas usando palabras y frases fáciles de entender. Por otra parte están los lenguajes de bajo nivel, que son los mejor entiende y sigue la máquina.

Los lenguajes de programación pueden clasificarse según el paradigma que usan en: orientado a objetos, funcionales, lógicos, estructurales, etc., para el desarrollo de esta aplicación se usara un lenguaje orientado a objetos, por la facilidad de uso que estos permiten, ejemplos de ellos son C Sharp, C++, Java entre otros. (De los Lobos, 2005)

C Sharp

C# o C Sharp es un lenguaje orientado a objetos, desarrollado por Microsoft como parte de su plataforma .Net. C# intenta ser el lenguaje para escribir aplicaciones .Net. Algunas características del C# son (Corporation., 2001):

- Es un lenguaje simple, seguro, moderno y de alto rendimiento.
- Incluye un amplio soporte de estructuras, componentes, programación orientada a objetos, manipulación de errores y recolección de basura.
- Está construido sobre los principios de C++ y Java.
- C# contiene las herramientas para definir nuevas clases, sus métodos y propiedades.

- Presenta una sencilla habilidad para implementar encapsulación, herencia y polimorfismo que son los tres pilares de la programación orientada a objetos.
- C# tiene un nuevo estilo de documentación XML[11] que incorpora a lo largo de la aplicación
- Soporta interfaces, una forma de estipular los servicios requeridos de una clase.
- Las clases en C# pueden heredar de un padre pero pueden implementar varias interfaces.
- C# provee características de componentes orientados como propiedades, eventos y atributos.
- C# provee soporte para acceder a la memoria usando el estilo de punteros de C++.
- Soporta los modificadores de acceso privado, protegido, público y agrega un cuarto modificador interno.
- No es posible usar variables no inicializadas.
- Los parámetros que son pasados son de tipo seguro.
- C# depende de tiempo de ejecución que provee la plataforma .NET, el mismo administra la ejecución del código.

C++

Es un lenguaje imperativo y orientado a objetos. La intención de su creación fue extender el exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. En este sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Posteriormente se añadieron facilidades de programación genérica, que se sumó a los dos paradigmas que ya estaban admitidos (programación estructurada y programación orientada a objetos). Es por esta razón se conoce al lenguaje C++ como multiparadigma (Bustamante, y otros, 2004).

En la actualidad, el C++ es un lenguaje versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones. A la vez es un lenguaje procedural y orientado a objetos. Como lenguaje procedural se asemeja al C, es compatible con él, y tiene como ventaja, las modificaciones menores. Como lenguaje orientado a objetos se basa en una filosofía completamente diferente. Las características propias de la Programación Orientada a Objetos de C++ son modificadores mayores que cambian radicalmente su naturaleza.

[11] Extensible Markup Language

Este lenguaje está constituido por tres elementos fundamentales: el compilador, encargado de traducir al lenguaje máquina, el programa C++ contenido en uno o más ficheros fuente. El preprocesador, actúa sobre el programa fuente, antes de que empiece la compilación para realizar ciertas operaciones y la librería estándar que no es más que, el conjunto de funciones pre programadas que se venden o se entregan junto al compilador (Bustamante, y otros, 2004).

C++ es un lenguaje de programación extremadamente complejo, son cuantiosas sus funcionalidades y siempre tiene algo que mostrar al programador. Ha experimentado un gran éxito desde su creación por lo que muchos sistemas operativos, compiladores e intérpretes han sido escritos en C++. A continuación se describen sus principales características (Bustamante, y otros, 2004).

- La posibilidad de orientar la programación a objetos permite al programador realizar aplicaciones más cercanas a la vida real.
- Presenta una gran portabilidad debido a que los códigos escritos en C++ pueden ser compilados en casi todo tipo de ordenadores y sistemas operativos con el menor número de cambios.
- El código escrito en C++ es breve en comparación con otros lenguajes, gracias al uso de caracteres especiales.
- El código resultante de una compilación en C++ es muy eficiente, gracias a su capacidad de actuar como lenguaje de alto y bajo nivel.
- Un cuerpo de aplicación en C++ puede estar hecho con varios ficheros de código fuente, que son compilados por separado y después unidos. Esta característica permite unir código C++ con otros lenguajes como ensamblador y C.

Fundamentación de la selección del lenguaje a utilizar.

Se ha seleccionado como lenguaje a utilizar en la implementación de la aplicación de escritorio C++. Este es un lenguaje muy sólido, puede crear desde sencillas aplicaciones hasta sistemas operativos. Es portable, un programa escrito en C++ se puede compilar en cualquier sistema operativo o sistema informático si apenas cambiar el código. Otra de sus ventajas es que es un lenguaje multi-nivel, es decir puedes usarlo para programar directamente el hardware y para realizar aplicaciones tipo Windows.

1.2.5 IDE a utilizar en el desarrollo.

Netbeans

Es una herramienta de código abierto para programadores, pensada para escribir, compilar, ejecutar y depurar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Soporta el desarrollo de aplicaciones en Java, C y C++. Incluye herramientas de desarrollo visuales de SOA[12], herramientas de esquema XML, orientación a servicios web y modelado UML. Todas sus funciones son provistas por módulos y cada módulo posee una función bien específica tales como la edición y el soporte para el control de versiones.

Eclipse

Es un entorno de desarrollo de código abierto multiplataforma. Detrás de este entorno se encuentra una gran comunidad de usuarios, extendiendo continuamente las áreas de aplicación. Eclipse fue desarrollado por IBM13 como el sucesor de su familia de herramientas VisualAge. Eclipse es ahora desarrollado por la Fundación Eclipse, una organización que promueve una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Una de sus desventajas es que toma una largo tiempo al usuario configurarlo, aún más que otros IDEs.

Geany

Es un IDE de programación compatible con multitud de lenguajes como C, C++, Java, PHP, HTML, Python, Perl, Pascal, Ruby, Fortran, Haskell entre otros. Entre sus muchas funciones, Geany incluye resaltado de síntesis, completado de código, autocompletado de funciones habituales y de etiquetas XML, HTML y lista de símbolos. Permite compilar y ejecutar programas creados o editados, ya sea desde el terminal o desde el menú. (Lopez, 2009) Dispone de plugins, destacando autoguardado, buscador de archivos y exportador.

[12] Arquitectura orientada a servicios.
[13]International Business Machines.

Fundamentación del IDE a utilizar

Para el desarrollo de software se escogió el Geany como entorno. Según la investigación realizada se ultimó que es muy factible programar en él y de fácil configuración, es un entorno de desarrollo ligero, pero muy potente. Compatible con la mayoría de los lenguajes, presenta varios paneles para acceder mejor a los datos, soporte multidocumento y soporte de proyectos. Está disponible en varios sistemas operativos como GNU/Linux, Mac OS X, BSD, Solaris y Windows. Y una de sus fundamentales características es que esta distribuido como software libre bajo la Licencia Publica General.

1.2.6 Compilador a utilizar en el desarrollo.

g++

Es un compilador creado por el proyecto GNU[14], es de software libre y está distribuido bajo la licencia GPL[15]. Es estándar para los sistemas operativos derivados de Unix de código abierto y es el encargado de ejecutar el compilador de C++. Es un programa independiente que toma como entrada código fuente y produce código en ensamblador. Tiene como estructura interna un font end por lenguaje que procesa el lenguaje y produce un árbol de sintaxis y un back end que convierte esos árboles en lenguaje de transferencia de registros. Luego realiza varias optimizaciones y produce un ensamblador utilizando un reconocimiento de patrones específicos.

1.2.7 Sistemas Gestores de Bases de Datos (SGBD).

Un sistema gestor de base de datos (SGBD) es un conjunto de programas específicos dedicados a servir de interfaz entre las bases de datos y las aplicaciones que lo utilizan. En las bibliografías de este tema se refieren a SGBD y DBMS, siendo ambos equivalentes y acrónimos respectivamente de Sistema Gestor de Base de Datos y *Data Base Management System*. Como propósito general tienen, manejar de manera clara, sencilla y ordenada un conjunto de datos relevantes para un proyecto. (Castillo, 2008)

[14] Es una acrónimo recursivo que significa GNU No es Unix y es el nombre de un proyecto creado por Richard Stallman con el objetivo de crear un sistema operativo completamente libre.
[15] Licencia Pública General.

Un SGBD permite: ahorrar a los usuarios detalles acerca del almacenamiento físico de los datos. La independencia entre los datos y los programas de aplicación. Minimiza las redundancias y garantiza la integridad, la seguridad y la protección de los datos. Facilita la manipulación de la información y permite un control centralizado. Entre los principales gestores de bases de datos se encuentran el Oracle, Microsoft SQL Server, MySQL y el PostgreSQL. A continuación se expondrán las principales características de cada uno de ellos, lo que permitirá la selección del que se utilizará en el desarrollo de la aplicación.

SQL Server

Es un sistema gestor de base de datos muy completo. Posee una buena velocidad y soporta un buen volumen de datos. Presenta una plataforma de desarrollo fácil y abierta integrada por las mejores tecnologías como ActiveX, ADC y Microsoft Transaction Server y con las mejores herramientas de gestión y desarrollo como FrontPage97, Microsoft Office97 y Visual InterDev. Se adapta a las necesidades de la empresa, soportando desde unos pocos usuarios a varios miles y es compatible con varias herramientas como Visual Basic, Visual C++, Visual J++, Visual InterDev y Microfocus Cobol (Casaares, 2005).

Sus principales características son:

- Soporte de transacciones.
- Escalabilidad, estabilidad y seguridad.
- Soporta procedimientos almacenados.
- Permite trabajar en modo cliente servidor, donde la información y datos se alojan en el servidor, y los clientes solo acceden a la información.
- Además permite administrar información de otros servidores de datos.

Sus principales desventajas son:

- Solo permite albergar un máximo de 64GB de memoria compartida.
- No maneja compresión de datos, por lo que puede llegar a ocupar mucho espacio en disco.
- Requiere de un sistema operativo de Microsoft Windows.

MySQL Server

MySQL por su parte es un servidor de base de datos que ostenta la cualidad de poseer un rápido nivel de procesamiento. Se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia. Es el más usado cuando se emplean lenguajes de programación como PHP y Perl. Es un software confiable, fácil de

usar, multiplataforma, multiusuario y permite elaborar consultas con el robusto SQL. Una de sus principales ventajas es la velocidad de lectura de datos. Es el mejor cuando se trabaja con indexación.

Es un servidor multi-hilos de base de datos de código abierto confiable, multiproceso, y poderoso. El objetivo principal de MySQL es la velocidad y la robustez, escrito en C y C++ posee un sistema de contraseñas y privilegios fiable y seguro. Es muy usado en aplicaciones web como Drupal y PhpBB, en plataformas como Linux y Windows y en herramientas de seguimiento de errores como Bugzilla. Fue desarrollado inicialmente para manejar grandes bases de datos mucho más rápidamente que las soluciones existentes y ha sido usado exitosamente por muchos años en ambientes de producción de alta demanda.

Características Distintivas:

- Múltiples motores de almacenamiento, permitiendo al usuario escoger lo que sea más adecuado para cada tabla de la base de datos.
- Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.

Oracle

Es el sistema gestor de base de datos más completo que existe, destacando entre sus características el soporte de transacciones, presenta una buena escalabilidad y estabilidad. Soporta varias plataformas. Presenta una disponibilidad controlada de los datos de las aplicaciones y una eficaz conectabilidad. Su elevado costo hace que solo lo usen las grandes empresas y multinacionales y que no esté tan extendido como otras bases de datos como MySQL, PostgreSQL y Access. Un aspecto que ha sido criticado por algunos especialistas es la seguridad de la plataforma y las políticas de seguridad, modificadas a partir del 2005 y que incrementan el nivel de exposición de los usuarios.

Entre sus características se destacan:

- Maximiza la disponibilidad y elimina la redundancia del centro de datos inactivos.
- Protege con seguridad la información almacenada.
- Comprime datos en particiones de almacenamiento de bajo coste para un rendimiento más rápido.

Presenta desventajas tales como:

- El mayor inconveniente de Oracle es su precio, incluso las licencias de Personal Oracle son excesivamente caras.
- Necesidad de ajustes. Un Oracle mal configurado puede ser enormemente lento
- Elevado coste de la formación. Poca información gratuita sobre su instalación y administración.

PostgreSQL

Es el servidor de bases de datos de código libre más potente que existe. Permite métodos almacenados, restricciones de integridad y vistas. Es el servidor más usado por aquellos programadores que requieren realizar aplicaciones cliente servidor, complejas o críticas.

PostgreSQL cuenta con una comunidad global de miles de contribuyentes y usuarios, y docenas de compañías y organizaciones. El Proyecto PostgreSQL tiene más de 20 años de ingeniería, desde sus inicios en la Universidad de Berkeley, California, con un ritmo de desarrollo actual sin precedentes. El conjunto de funcionalidades de PostgreSQL no sólo es comparable a los mejores sistemas gestores de datos propietarios, sino que las superan en características avanzadas, extensibilidad, seguridad y estabilidad.

Se destaca por algunas características tales como:

- Instalación ilimitada: varios clientes comerciales instalan las bases de datos en más servidores de los que permite la licencia y no hay demandas puesto que la licencia no tiene costo ninguno.
- Mejor soporte que otros proveedores: además del soporte que ofrece PostgreSQL, también presenta una importante comunidad de profesionales dispuestos a contribuir a su desarrollo.
- Extensible: el código fuente está disponible para todos sin costo alguno.

Fundamentación de la selección del sistema gestor de base de datos.

Después de realizar un análisis completo de los diferentes sistemas gestores de base de datos más utilizados en la actualidad y teniendo en cuenta los aspectos más significativos por los cuales se puede comparar, se llega a la conclusión de que el más idóneo para lograr los objetivos trazados es MySQL. Este sistema presenta una gran robustez y presta atención a las restricciones del cliente. En comparación

con otros sistemas de base de datos es extremadamente estable, MySQL nunca ha presentado caídas en varios años de operación de alta actividad. MySQL es de fuente abierta lo que significa que cualquier persona pueda usarlo y modificarlo. Además de que presenta determinadas características que solo son implementadas por él.

1.2.8 Sistemas externos empleados en el desarrollo del componente

Para comprender mejor el funcionamiento de la aplicación es necesario conocer qué sistema de reconocimiento del habla se utiliza para la extracción de los descriptores semánticos y que sistema se utiliza para la recuperación de información.

Sistema automático del habla seleccionado

Ante los distintos sistemas de reconocimiento automático del habla disponibles es necesario realizar una comparación de sus características para seleccionar el sistema más adecuado para el desarrollo de esta aplicación.

Tras un estudio de los sistemas de código libre, se descartó la posibilidad de su uso, ya que los sistemas gratuitos no disponen de modelos acústicos y de lenguaje para el español. Igualmente su eficiencia para los lenguajes disponibles, como el inglés es muy inferior a la de los sistemas comerciales, lo que hace inviable para una aplicación, donde es fundamental el rendimiento del sistema de reconocimiento de voz. NaturallySpeaking y Via Voice ofrecen funcionamiento y características similares, no obstante son dependientes del hablante. Ambos programas ofrecen una tasa baja de error, sin embargo es necesario entrenar el sistema, lo cual es un problema cuando se trata de obtener la transcripción de distintos locutores como es el caso de esta aplicación.

MMI por el contrario brinda una ventaja destacable, que cuenta con un modelo acústico independiente del hablante y tras procesar un archivo de audio genera una salida en formato XML. Incluyendo información adicional como el género del locutor y el momento del archivo de audio correspondiente a cada frase. Este software también brinda una API escrita en C++ que permite su integración con otras aplicaciones en Windows y en Linux. Estas significativas características lo convierten el sistema seleccionado para el reconocimiento del habla.

Sistema de recuperación de información seleccionado.

Tras estudiar las ventajas y desventajas de los sistemas de recuperación de código libre se decide utilizar Lucene como solución para el desarrollo de esta herramienta. El sistema de recuperación debe ser el encargado de indexar la información textual generada en el proceso de almacenamiento y a su vez sea capaz de realizar búsquedas realizadas en lenguaje natural en el índice creado.

Lucene es un sistema de recuperación de información que cuenta con operadores booleanos, es capaz de realizar consultas mediante frases y búsqueda por proximidad, además devuelve los resultados en orden de relevancia e incluso es capaz de obtener resultados parciales. Permite la indexación incremental y presenta una rápida respuesta, tanto como para realizar la indexación, como para realizar búsquedas complejas.

1.3 Conclusiones Parciales del capítulo.

Las soluciones de código abierto existentes en la actualidad referente a la recuperación, organización y búsqueda de la información audiovisual son eficientes pero se están quedando rezagadas en el avance tecnológico, pues solo realizan búsquedas textuales. En cambio, referente a las soluciones existentes para el reconocimiento del habla se determinó que las soluciones de código abierto no son eficientes, ni presentan las características necesarias para el desarrollo de la aplicación, no siendo así con las comerciales. Por otra parte se concluye que una buena selección de las herramientas y metodologías de desarrollo, deriva aplicaciones robustas, eficaces y óptimas.

Capítulo 2. Características del sistema.

En este capítulo se presenta formalmente la situación problemática. Así como una descripción detallada de los documentos que se procesan y la información que se manipula. También se hace una descripción general de la propuesta del sistema que propone este trabajo de diploma. Además se expone el modelo del negocio, la especificación de los requisitos funcionales y no funcionales y finalmente se muestran los diagramas de casos de uso.

2.1 Modelo del negocio

Unos de los flujos más significativos durante la fase de inicio en el desarrollo del software es el modelado del negocio el cual tiene como principales objetivos:

- Comprender la estructura y la dinámica de la organización en la cual se va a implantar un sistema
- Comprender los problemas actuales de la organización y e identificar las mejoras potenciales.
- Asegurar que los clientes, los usuarios finales y desarrolladores tengan un mejor entendimiento común de la organización.
- Derivar los requerimientos del sistema que va a soportar la organización.

Para lograr estos objetivos, el proceso de modelado permite obtener una excelente visión de la organización que permita definir los procesos, roles y responsabilidades de la organización en los modelos de casos de uso del negocio y de objetos. Dependiendo de la situación o escenario, hay varias alternativas de desarrollar este proceso, no siempre es necesaria o posible la completa realización de un modelo completo del negocio. Si se determina que no es necesario un modelado completo del negocio, se realizará lo que se conoce como modelo de dominio.

2.2 Modelo del dominio

El modelo del dominio se centra en una parte del modelo de negocio. Captura los tipos de objetos más importantes que existen o los eventos que suceden en el entorno donde estará el sistema. El modelo de domino ayuda a comprender los conceptos que utilizan los usuarios, los conceptos con los que trabajan y con los que deberá trabajar el sistema. Este modelo (Ver Figura 4) en conjunto con el Diccionario de Dominio ayudará a establecer un solo lenguaje entre los clientes y los desarrolladores.

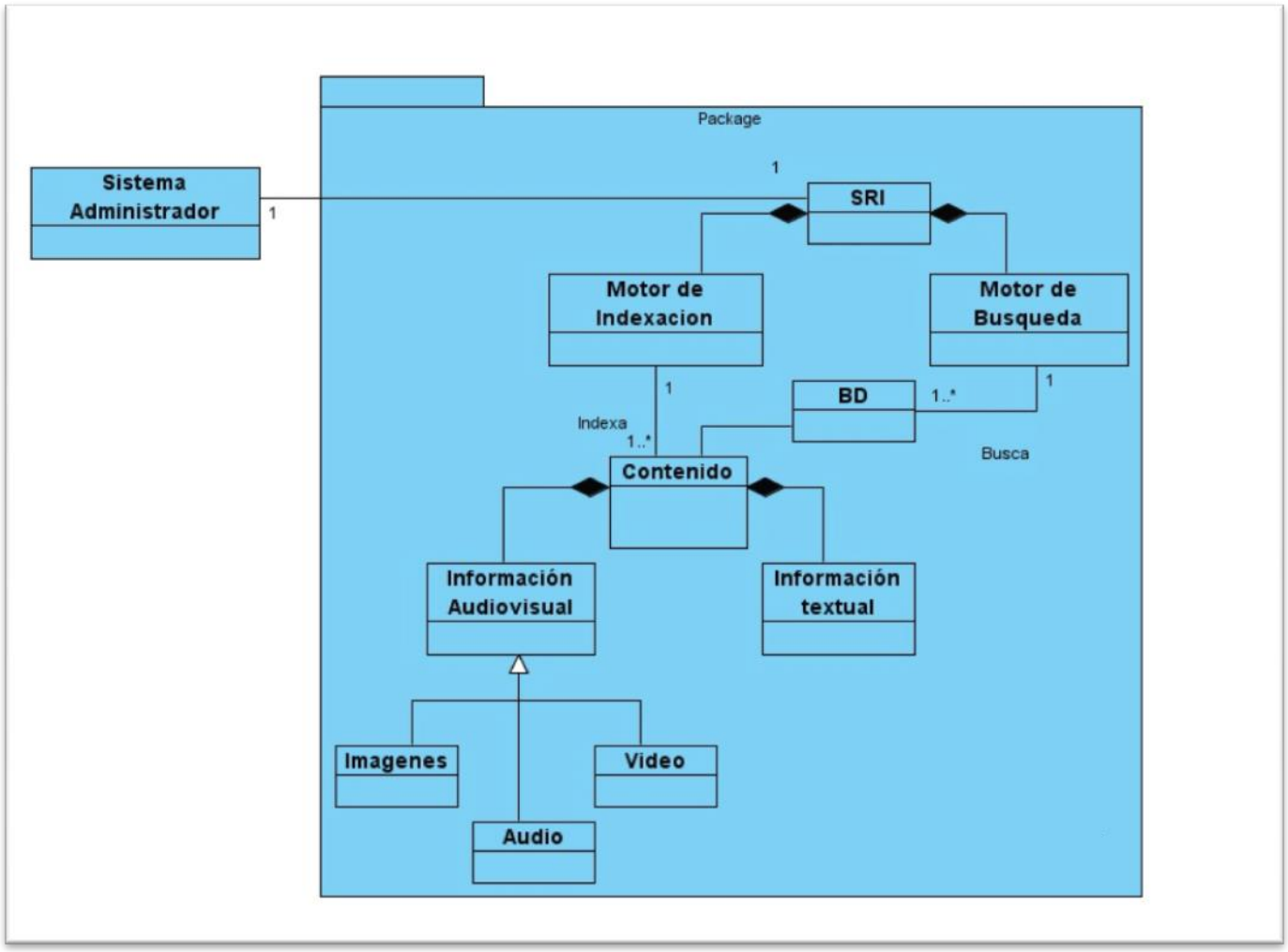


Figure 4: Modelo del Dominio del Componente

Diccionario del Dominio

Sistema Administrador: Sistema encargado de realizar todas las acciones en el componente.

SRI: Componente.

BD: Base de datos

Motor de Indexación: es la estructura del componente encargado de indexar la información contenida en la base de datos

Motor de Búsqueda: es la estructura del componente encargado de realizar la búsqueda de la información contenida en la base de datos.

2.3 Propuesta del sistema

De acuerdo con los estudios realizados y luego de materializar un profundo análisis del objeto de estudio se ha concebido implementar un sistema capaz de indexar y realizar búsquedas contextuales de la información audiovisual (Ver Figura 5). El sistema será el encargado de indexar los documentos y realizar búsquedas en las bases de datos. El servidor de indexación proporciona al sistema de recuperación de información los datos necesarios para realizar la indexación de los documentos

También es necesario destacar que para facilitar el uso de la herramienta que se desea implementar es importante el desarrollo de interfaces de usuario que den acceso a la aplicación. Puesto que el sistema permitirá al sistema administrador tanto indexar como buscar documentos, se desarrollaran dos interfaces distintas, una que permita el almacenamiento de audio, imágenes y videos por parte del usuario para su posterior indexación y otra que permita la búsqueda y recuperación de dichos documentos mediante consultas en lenguaje natural.

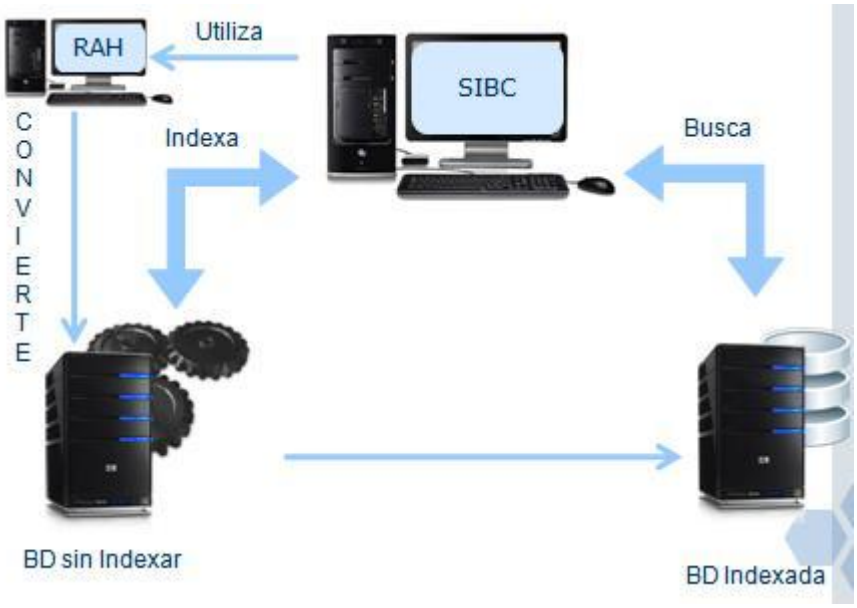


Figure 5 Propuesta del sistema

El necesario destacar que para indexar los archivos de audio y video, el sistema utilizará los beneficios brindados por un motor de reconocimiento del habla, encargado de convertir el audio en texto. Facilitando su indexación y posterior búsqueda.

Es imprescindible que los usuarios como principales beneficiarios de este sistema, noten una mejoría importante con su instauración y puesta en práctica. Para lograr esto, los requisitos funcionales se han capturado partiendo de la necesidad real de los procesos del negocio.

2.4 Especificación de los requerimientos del software

Un requerimiento es una funcionalidad que puede ser alcanzada o poseída por un sistema o un componente de un sistema, para satisfacer un contrato u otro documento impuesto formalmente entre los desarrolladores del software y los clientes del mismo. Define qué es lo que el sistema debe hacer.

2.4.1 Requerimientos Funcionales

Los requisitos funcionales especifican el comportamiento interno que tendrá la solución propuesta. Este comportamiento se podrá expresar como las tareas más importantes a desarrollar.

RF.1- Realizar indexación de información audiovisual.

RF.1.1 Listar la información indexada

RF.1.2 Mostar tiempo de ejecución del proceso de indexación

RF.2 Gestionar Información Indexada: Define las funcionalidades relacionadas con la gestión de la información.

RF.2.1 Actualizar la información indexada: Define la posibilidad de actualizar la información indexada.

RF .2.2 Modificar la información indexada: Define la posibilidad de modificar la información indexada.

RF.2.3 Eliminar la información indexada: Define la posibilidad de eliminar la información indexada.

RF.3-Realizar búsquedas contextuales: Define la posibilidad de realizar búsquedas mediante el uso de comodines, frases y oraciones.

RF.4-Realizar búsquedas textuales y de contenido: Define la posibilidad de especificar qué tipo de búsqueda desea realizar.

2.4.2 Requerimientos No Funcionales

- **Usabilidad**

La aplicación deberá presentar un diseño centralizado en una interfaz principal que permita la ejecución de las funcionalidades del sistema. Está diseñado de manera que los usuarios finales adquieran las habilidades para usarlo en un tiempo reducido:

Usuarios normales: 31 días

Usuarios avanzados 15 días

- **Seguridad**

Confidencialidad: La información manejada por el sistema debe estar protegida de acceso no autorizado y divulgación.

Integridad: La información manejada por el sistema debe ser objeto de cuidadosa protección contra la corrupción y estados inconsistentes, de la misma forma será considerada igual a la fuente o autoridad de los datos. Puede incluir mecanismos de chequeo de integridad y realización de auditorías.

El sistema permitirá la recuperación de la información a partir de los respaldos o copias de seguridad.

- **Portabilidad**

El sistema será desarrollado sobre una plataforma multi-sistema que posibilita la ejecución en varios sistemas operativos.

- **Disponibilidad**

A los usuarios autorizados se les garantizará el acceso a la información. Los dispositivos y mecanismos utilizados para lograr la seguridad no ocultarán o retrasarán a los usuarios para obtener los datos deseados en un momento dado.

- **Rendimiento**

El sistema minimizará el volumen de datos en las peticiones y además optimizará el uso de recursos críticos como la memoria.

El sistema respetará buenas prácticas de programación para incrementar el rendimiento en operaciones costosas como la creación de objetos.

- **Software**

La aplicación requiere para su ejecución MySQL y como IDE de desarrollo el Geany.

- **Hardware**

La aplicación debe ser compatible con bajas capacidades de hardware, a continuación se muestran datos de estos requerimientos.

Capacidades de hardware mínimas:

RAM: 128 MB

CPU: 1.0 GHz

Capacidades de hardware recomendadas:

RAM: 512 MB

CPU: 2.0 GHz

2.5 Modelo del sistema. Definición de los casos de uso

Los casos de uso son una técnica que proporciona uno o más escenarios de como el sistema debe interactuar con el usuario o con otro sistema para conseguir un objetivo específico. El modelo de casos de uso describe lo que hace el sistema para cada tipo de usuario.

2.5.1 Definición de actores

Los actores del sistema constituyen personas, sistemas u entidades abstractas, que eran trabajadores del negocio e interactúan de alguna forma con el sistema y están asociados al cumplimiento de los requisitos funcionales o procesos que responden a las funcionalidades definidas en los mismos.

Durante el desarrollo de la aplicación se definió un conjunto de actores que en un momento determinado desencadenaran un grupo de acciones en el sistema, los mismos se describen a continuación.

Tabla 1: Definición de actores

Actores	Descripción
Sistema Administrador	Actor con permisos de ejecución de todos los casos de uso

2.5.2 Casos de uso del sistema

Los casos de uso son artefactos narrativos que describen, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades que debe cumplir el sistema.

Tabla 2: Descripción textual del caso de uso Realizar Indexación.

Nombre del caso de uso	Realizar indexación de información audiovisual.
Actores	Sistema administrador
Propósito	Permite indexar la información contenida en la base de datos es decir por puede ser ordenada y

	clasificada según el tipo de información
Resumen El caso de uso comienza cuando el sistema administrador le muestra los metadatos al componente. Culmina una vez después de ser generados los descriptores para la información en la base de datos del componente.	
Referencias	RF1
Precondiciones	• Ruta de los ficheros a indexar, en este caso es un servidor local.
Pos condiciones	• Información indexada en la base de datos
Acción del actor	**Respuesta del Sistema**
El sistema administrador debe iniciar al componente con la petición de realizar la indexación de la información contenida en una base de datos.	El componente recibe la solicitud e inicia el proceso de indexado con la información contenida en la base de datos.
	El componente procede a comprobar si la petición es válida, verificando su estructura y su existencia en la base de datos e indexa la información relacionada y crea una lista ordenada por un orden de relevancia con la información indexada
	El componente muestra cada información indexada en el progreso del proceso de indexación y notifica la finalización del proceso con un conjunto de estadísticas relacionadas
Flujo Alternativo	
Acción del actor	**Respuesta del sistema**

El sistema administrador introduce una ruta no valida y solicita iniciar el proceso de indexación	El componente procede a comprobar si la petición inicial es válida , verificando su estructura y existencia, no indexa la información y así continua el proceso para cada información de entrada
Requerimientos especiales	

Descripción textual de los restantes casos de usos (Ver Anexo 1)

2.5.3 Diagrama de casos de uso del sistema

El diagrama de casos de uso del sistema (Ver Figura 6) define las relaciones entre los actores y los casos de uso. Para la realización del componente se definieron un conjunto de acciones que deben ser ejecutadas en este caso, por un único actor y que desencadenan un conjunto de operaciones. Las interrelaciones que se establece entre las acciones y el actor del componente son agrupadas en el diagrama de casos de uso del sistema que se muestra a continuación

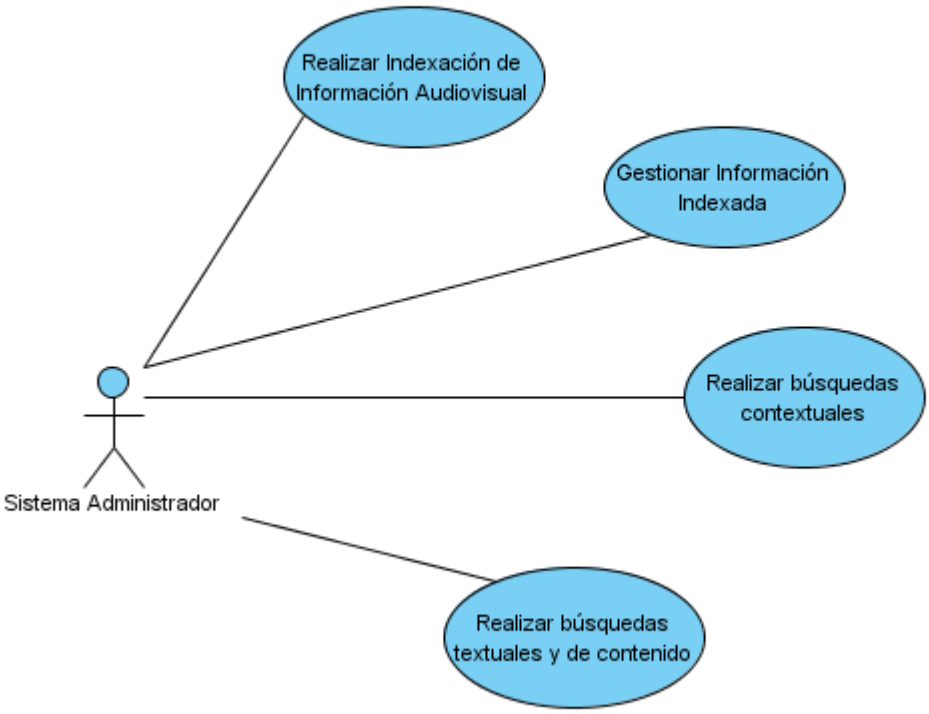


Figure 6: Diagrama de Casos de Uso

2.6 Conclusiones Parciales del capítulo.

El modelo de dominio es sumamente importante, ya que de su realización, depende una buena captura y entendimiento del área previa al diseño. El éxito o el fracaso del sistema dependen de este modelo. Es decir que el modelo de dominio ocupa un rol protagónico en el desarrollo moderno del software. Por otra parte la captura de requisitos es el hilo conductor de todo desarrollo de software. Una obtención de requisitos con calidad demuestra que el trabajo culminará con éxito.

Capítulo 3. Análisis, Diseño e Implementación

En este capítulo se presentan los temas referentes al análisis, diseño e implementación del sistema. Se hace referencia a las principales tareas que se llevan a cabo en este flujo de trabajo, así como la descripción de los artefactos que se generan.

3.1 Modelo de Análisis.

El modelo de análisis constituye un modelo que se emplea para obtener una visión del sistema sobre los requisitos funcionales, expresados en un lenguaje técnico. Es el resultado de la actividad de analizar los casos de uso. Este modelo ofrece una visión más precisa de los requisitos, los estructura de modo que facilita su preparación, comprensión, modificación y mantenimiento en general. Es tomado como una primera aproximación al diseño. Por su parte este se encargará de moldear el sistema y buscar una forma de arquitectura que de vida a los requerimientos del sistema.

En este modelo se precisa la elaboración de una serie de artefactos, el Diagrama de Clases del Análisis (Ver Figura 7) es un pilar fundamental en esta etapa de desarrollo para mostrar lo que el sistema puede hacer y cómo puede ser construido (Jacobson, y otros, 2000).Se centra en el tratamiento de los requisitos funcionales y pospone los no funcionales para el diseño. Según RUP siempre se ajusta a alguno de los estereotipos siguientes (Jacobson, y otros, 2000):

- Clase Interfaz: Modelan la interacción entre el sistema y sus actores.
- Clase Controladora: Coordina la realización de uno o unos pocos casos de uso, relacionando las actividades de los objetos que implementan sus funcionalidades.
- Clase Entidad: Modelan información que posee larga vida y que es a menudo persistente.

3.1.1 Diagramas de Clases del Análisis del componente.

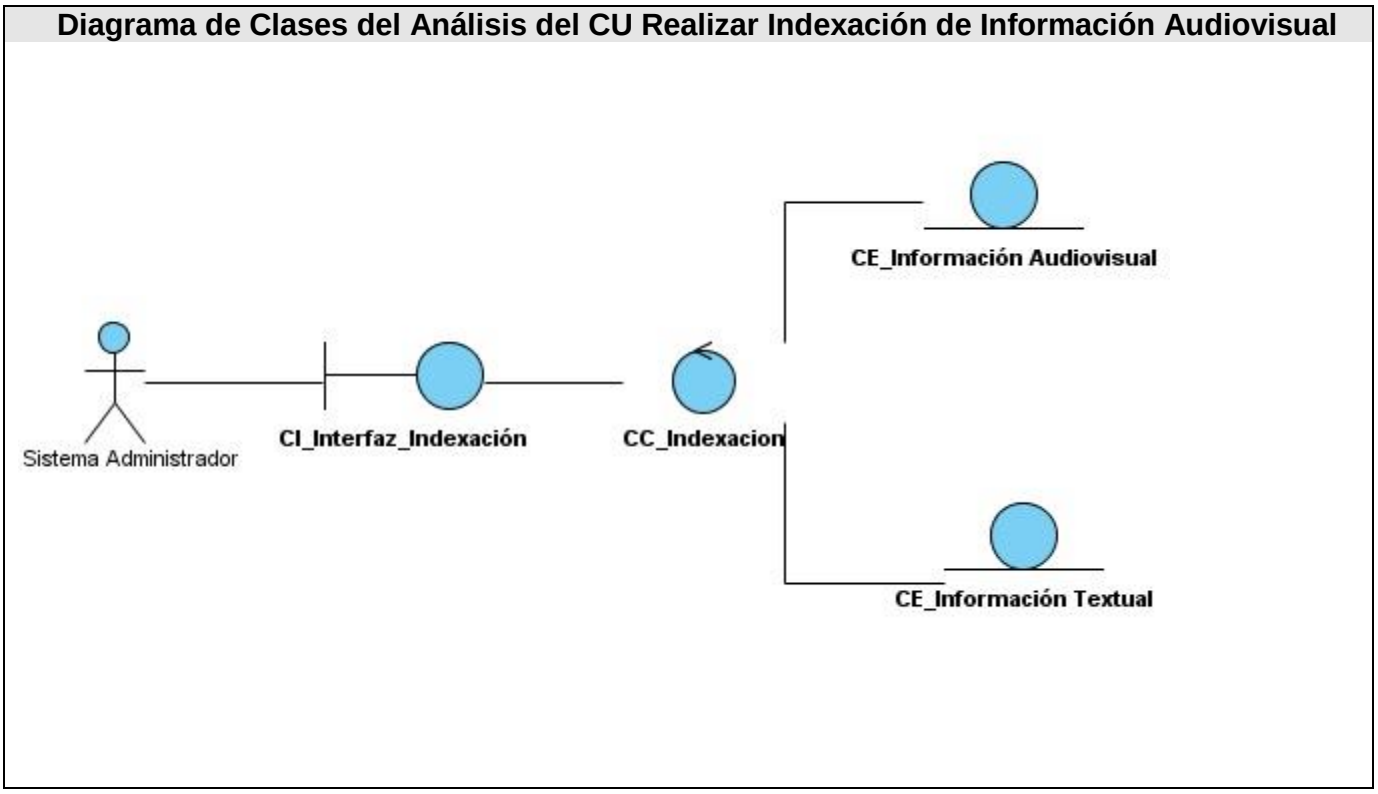


Figure 7: Diagrama de Clases del Análisis del CU Realizar Indexación de Información Audiovisual

Restantes diagramas de Clases del análisis (Ver Anexo 2)

3.1.2 Diagramas de Colaboración del Análisis del componente.

Los diagramas de colaboración (Ver Figura 11) son una forma alternativa a los diagramas de secuencia de mostrar un escenario. Estos diagramas muestran las interacciones entre los objetos y los enlaces entre ellos, contribuyendo a la representación de aspectos dinámicos del sistema.

Diagrama de colaboración del análisis del CU Realizar Indexación de Información Audiovisual

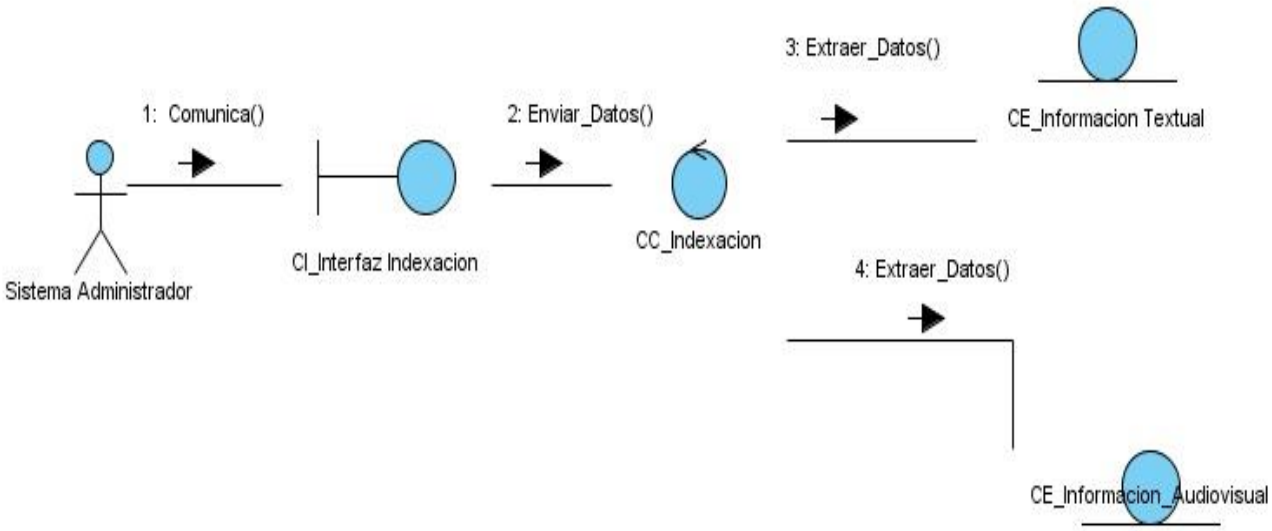


Figure 8: Diagrama de Colaboración del CU Realizar Indexacion de Informacion Audiovisual

Restantes diagramas de colaboración (Ver Anexo 3)

3.2 Modelo de Diseño

El objetivo fundamental de este flujo de trabajo es de traducir los requisitos funcionales a una descripción de cómo quedará implementado el sistema. Debe ser lo suficiente robusto para permitir ambigüedades. El objetivo de este flujo de trabajo es producir un modelo lógico del sistema a implementar.

Propósitos del diseño (Pressman, 2006):

- Adquirir una compresión en profundidad de los aspectos relacionados con los requisitos no funcionales, restricciones impuestas por el lenguaje de programación, el Sistema Operativo donde se va a ejecutar, el tipo de interfaz, entre otras.
- Punto de partida para la implementación capturando requisitos de las clases del análisis.
- Ser capaz de visualizar y razonar acerca del diseño usando una notación común.
- Crear una abstracción sin costuras de la implementación del sistema, en el sentido de que la implementación es un refinamiento directo del diseño que rellena lo existente sin cambiar la estructura.

3.2.1 Diagrama de clases del Diseño.

Durante el flujo de trabajo de diseño, se modela el sistema de manera que soporte todos los requerimientos, incluyendo a diferencia del análisis, los requerimientos no funcionales. Este modelo se puede utilizar para visualizar la implementación y para soportar las técnicas de programación grafica de la aplicación.

A continuación se muestran los diagramas (Ver Figura 15) de clases para cada caso de uso del sistema, de forma tal que se facilite la compresión de las relaciones entre los distintos componentes.

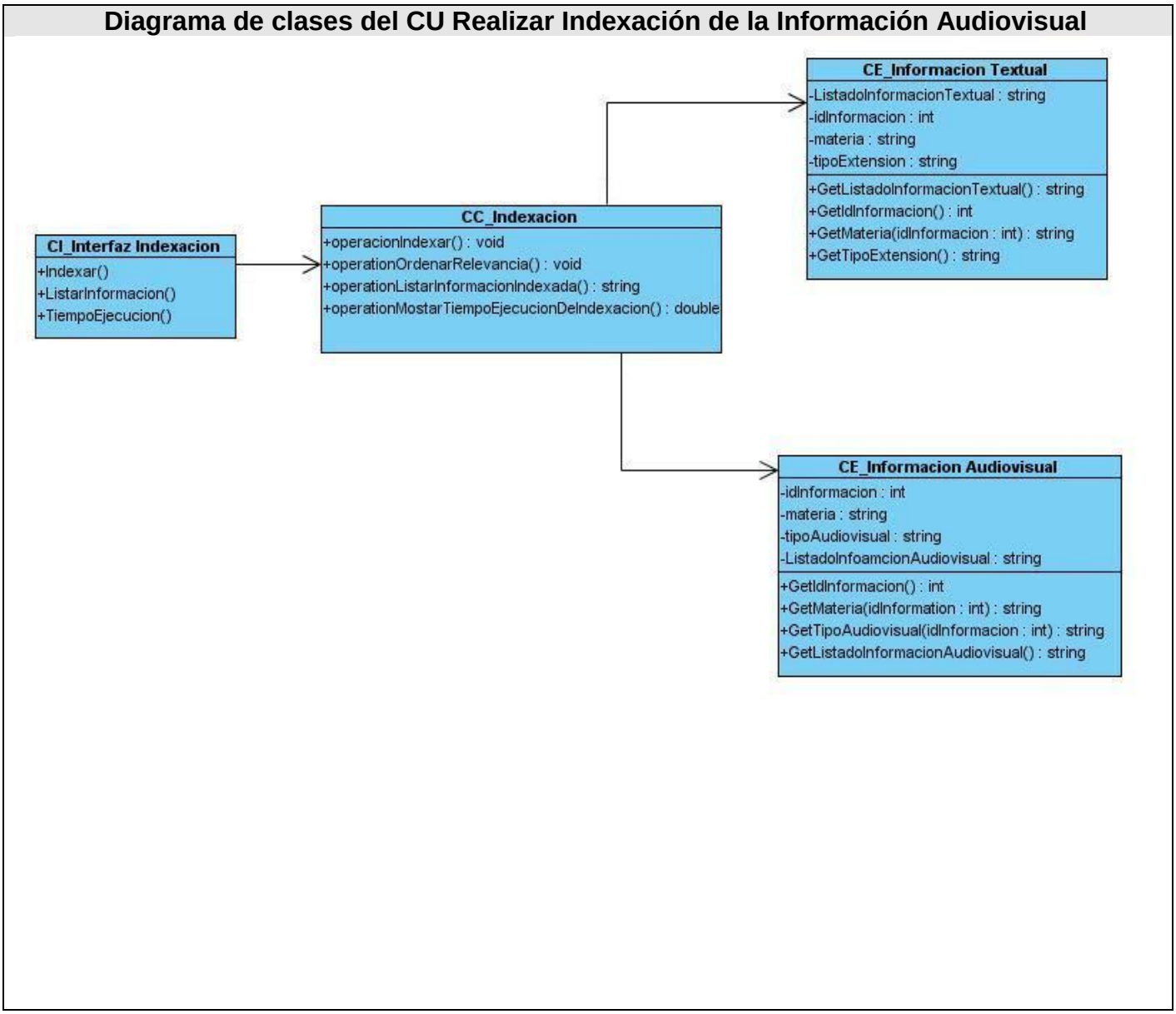


Figure 9: Diagrama de Clases del Diseño del CU Realizar Indexación de Información Audiovisual

Restantes diagramas de clases del diseño (Ver Anexo 4)

3.2.2 Descripción de las clases del diseño.

Tabla 6: Descripción textual de la clase Indexación.

Nombre: Indexación	
Tipo de clase (controladora)	
Atributo	**Tipo**
No tiene	No tiene
Para cada responsabilidad:	
Nombre:	operacionIndexar()
Descripción:	Función encargada de indexar la información contenida en la base de datos especificada
Nombre:	operacionOrdenarRelevancia()
Descripción:	Función encargada de ordenar la información según un orden de relevancia
Nombre:	operacionListarInformaciónIndexada()
Descripción:	Función encargada de listarle al sistema administrador la información una vez indexada
Nombre:	operacionMostrarTiempoEjecucion()
Descripción:	Función encargada de mostrarle al sistema administrador el tiempo que demoro el sistema en indexar.

Restantes tablas de descripción de las clases del diseño (Ver Anexo 5)

3.2.3 Patrones de diseño

Los patrones de diseño empleados en el desarrollo del componente fueron los patrones GRASP que son patrones de software para la asignación general de responsabilidades y describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades. Se pueden destacar como patrones fundamentales:

- Experto: Indica que la responsabilidad de la creación de un objeto debe recaer sobre la clase que conoce la información necesaria para crearlo.
- Creador: Este patrón ayuda a identificar quien debe ser el responsable de la creación de nuevos objetos o clases.
- Alta cohesión: Plantea que la información que almacena una clase debe ser coherente y está en la mayor medida de lo posible relacionada con la clase.
- Bajo Acoplamiento: Tratar de independizar entre sí lo menos posible, con el objetivo de que si existiera algún cambio sobre una de ellas no inicia directamente sobre las otras.
- Controlador: Fue utilizado como intermediario entre las interfaces y los algoritmos que la implementan, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

3.2.4 Modelo de Datos.

El modelo de datos no es más que la representación de la estructura o descripción física de las tablas de la base de datos. Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos: los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. A continuación se presenta el modelo de datos (Ver Figura 19) del componente (Marques, 2004).

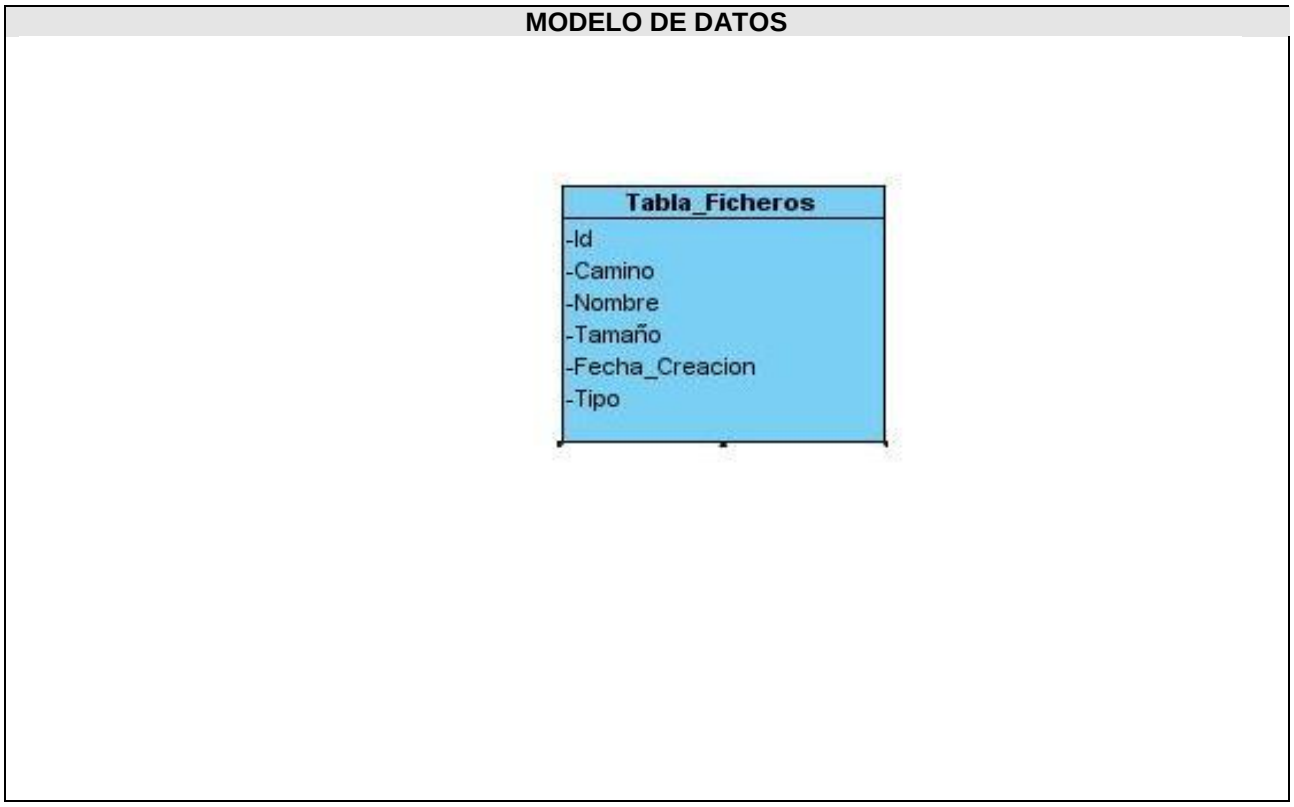


Figure 10: Modelo de Datos

Tabla 8: Descripción textual de la tabla Tabla_Ficheros

Nombre: Tabla_Ficheros		
Descripción: Tabla se encarga de almacenar los datos de los todos ficheros que posteriormente serán indexados		
Atributo	**Tipo**	**Descripción**
Id	int	Id del fichero
Camino	string	Localización de los ficheros que serán indexados
Nombre	string	Nombre del fichero
Tipo	string	Tipo de información (textual o audiovisual)

3.3 Modelo de Implementación.

En el flujo de trabajo de diseño se propone crear un plano del modelo de implementación, por lo que sus últimas actividades están vinculadas a la creación del modelo de despliegue. El flujo de trabajo de implementación describe como los elementos del diseño se implementan en términos de componentes y como estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue.

Los diagramas de despliegue y de componentes son artefactos generados en este flujo de trabajo, juntos conforman lo que se le conoce como un modelo de implementación, su función fundamental es describir los componentes a construir, su organización y su dependencia entre nodos físicos, que mostrará cómo funcionará la aplicación.

3.3.1 Diagrama de Componente.

Los diagramas de componentes se utilizan para modelar la vista estática de un sistema. Muestra la organización y las dependencias lógicas entre un conjunto de componentes de software, sean estos componentes de código fuente, librerías, ejecutables o binarios. No es necesario que un diagrama de componente incluya todos los componentes del sistema, normalmente se realizan por partes. A continuación se presenta el diagrama de componentes (Ver Figura 20) del componente.

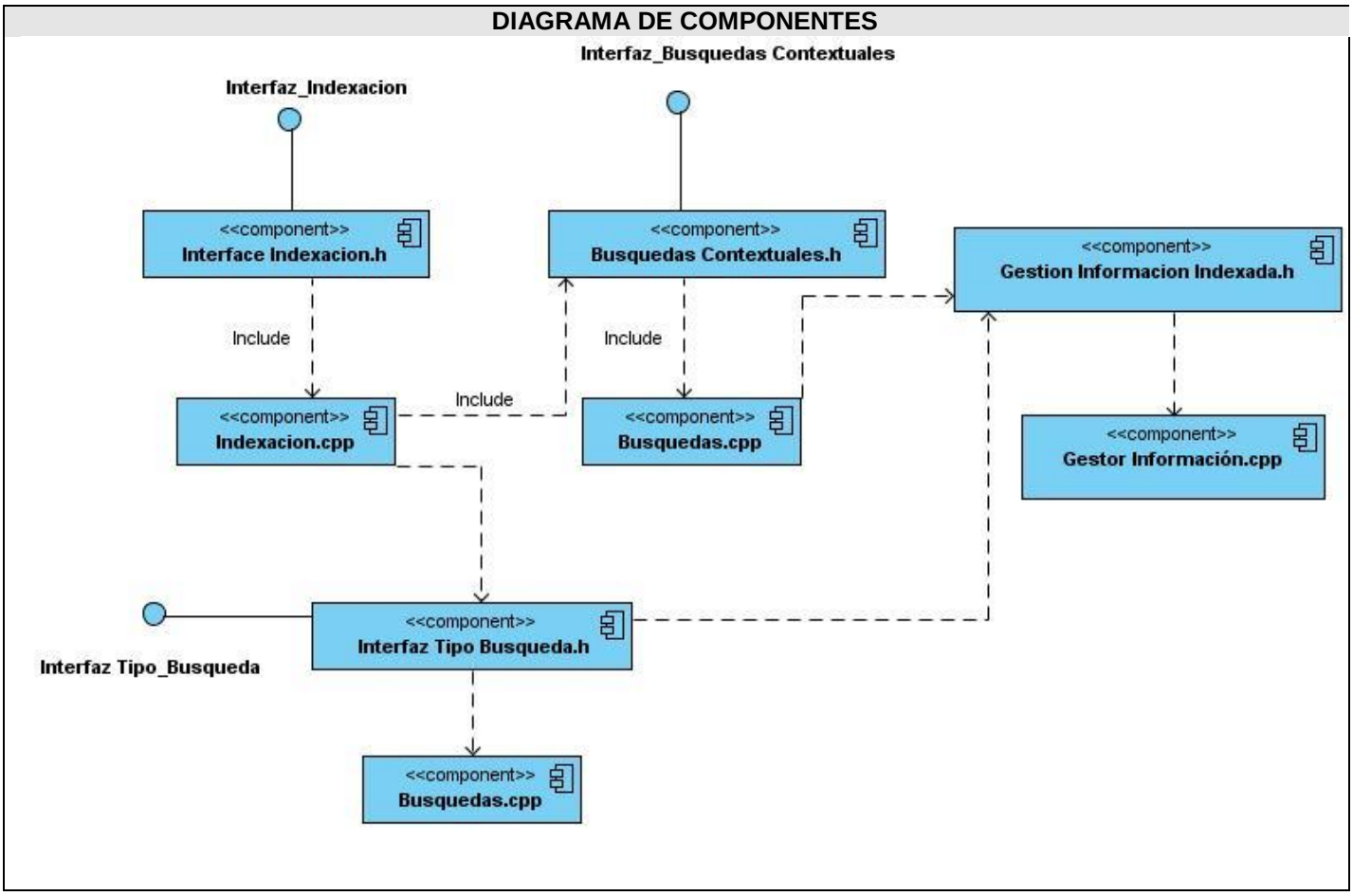


Figure 11: Diagrama de Componentes

3.3.2 Modelo de Despliegue.

El modelo de despliegue no es más que la representación física de los nodos, es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo. Es una colección de nodos y arcos; donde cada nodo representa un recurso de cómputo, normalmente un procesador o un dispositivo de hardware similar (Jacosbon, y otros, 2000). A continuación se presenta el diagrama de despliegue (Ver Figura 21) del componente.

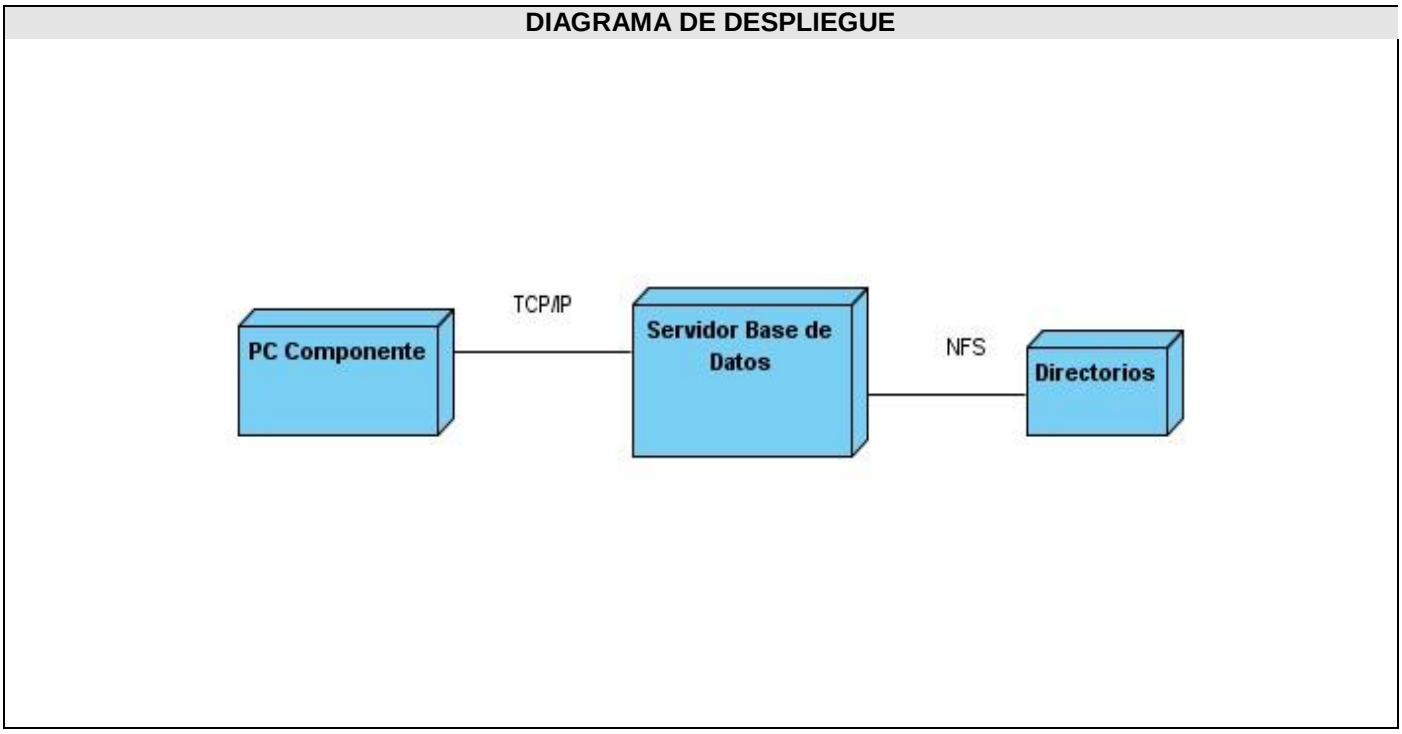


Figure 12: Diagrama de Despliegue

3.3.3 Conclusiones Parciales del capítulo.

La importancia del diseño del software se puede definir con una sola palabra “Calidad”, ya que dentro del diseño es donde se fomenta la calidad del proyecto y todos sus diagramas están enfocados en lograr este objetivo. Mientras que la implementación, es la fase cumbre de todo proceso de desarrollo de software. Es en esta fase donde se le da cumplimiento a todo lo analizado y diseñado, basándose en los diagramas de despliegue y componentes que describen la correspondencia entre la arquitectura de software definida y la del sistema, la dependencia entre los nodos físicos en los que funciona el sistema propuesto, así como los compontes a construir y su organización.

Capítulo 4. Modelo de Prueba

El desarrollo de un producto de software implica la realización de una serie de actividades encaminadas a encontrar errores. Es por ello que se deben incorporar acciones que evalúen la calidad del producto que se está desarrollando. Dentro de este proceso, el flujo de trabajo de Prueba, es mediante el cual se puede validar que las suposiciones hechas en el diseño y los requerimientos se estén cumpliendo satisfactoriamente.

Este flujo de trabajo brinda soporte para encontrar, documentar y solucionar problemas del sistema, por lo que debe estar presente en todo el ciclo de vida del desarrollo del producto para ir refinándolo paulatinamente y no al final del mismo.

Las pruebas se pueden realizar basándose en dos esquemas diferentes, demostrar a través de pruebas de caja blanca que las operaciones internas se ajustan a lo especificado y que los componentes internos funcionan correctamente, o mediante las pruebas de caja negra, conociendo la función del programa, intentar demostrar que las funciones están correctas.

4.1 Pruebas de caja blanca.

4.1.1 Pruebas de Camino Básico

Esta prueba le permite al diseñador de los casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. Los casos de prueba obtenidos del conjunto básico garantizarán que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa.

Los pasos del diseño de pruebas a seguir mediante el camino básico son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo de flujo.
3. Se determina el conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico.

Grafo de Flujo

Para la elaboración del mismo se utiliza los tres elementos siguientes:

- Nodos: representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión (bifurcación).
- Aristas: líneas que unen dos nodos.
- Regiones: áreas delimitadas por aristas y nodos. Cuando se contabilizan las regiones de un programa debe incluirse el área externa como una región más.

Complejidad ciclomática

El resultado obtenido en el cálculo de la complejidad Ciclomática define el número de caminos independientes dentro de un fragmento de código y determina la cota superior del número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

Existen varias formas de calcular la complejidad ciclomática de un programa a partir de un grafo de flujo:

1ra vía V (G)= Número de Regiones.

2da vía V (G)= Número de Aristas – Número de Nodos + 2

3ra vía V (G)= Número de Nodos Predicados +1

Camino Independiente

Un camino independiente es cualquier camino del programa que introduce un nuevo conjunto de sentencias. El conjunto de caminos independientes de un grafo no es único.

Derivación de casos de prueba

Luego de tener elaborados los grafos de flujos y los caminos a recorrer, se preparan los casos de prueba que forzarán la ejecución de cada uno de esos caminos. Se escogen los datos de forma que las condiciones de los nodos predicados estén adecuadamente establecidas, con el fin de comprobar cada camino**.**

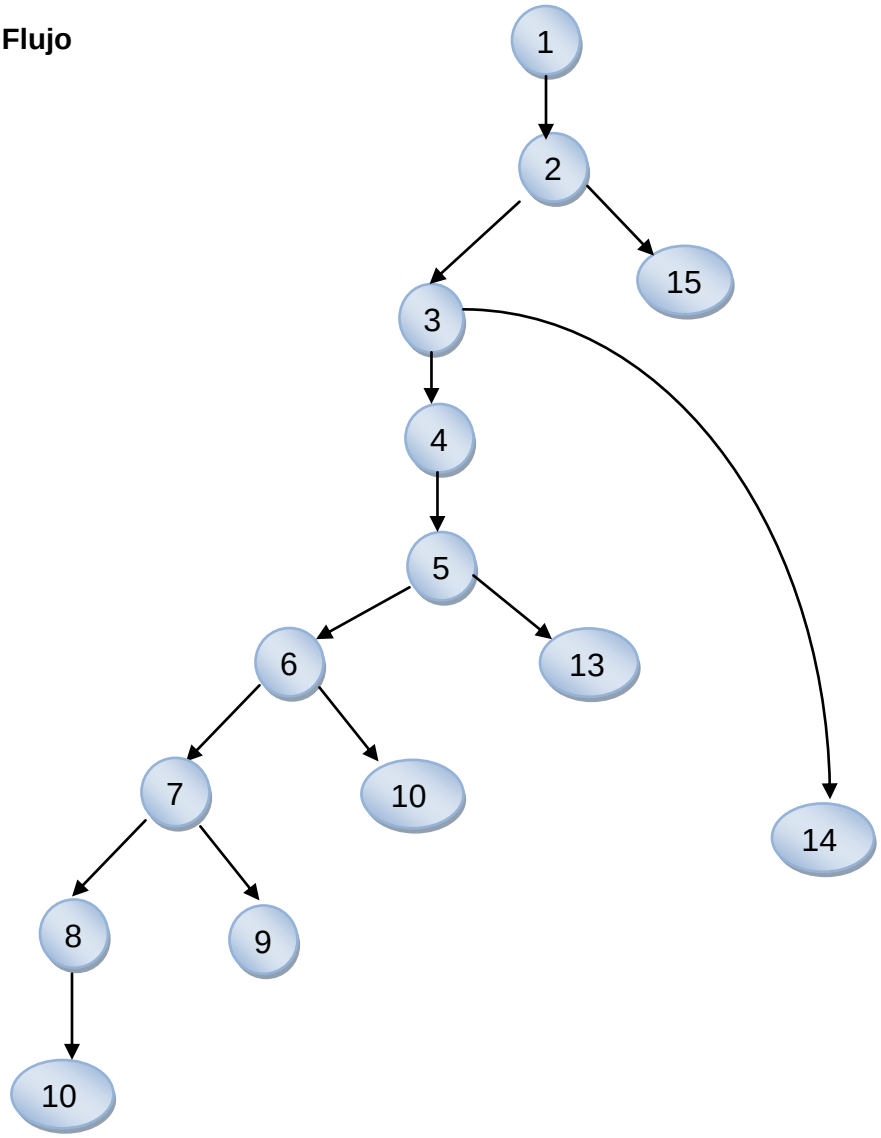
En este caso solo se expondrá en el trabajo las pruebas para el caso de uso Indexar Información Audiovisual, específicamente el método Cargar Ficheros

Pruebas de unidad para el caso de uso Indexar Información

```
int CInterfazBD::CargarFicheros(const string& inCamino)
{
  DIR* Directorio = opendir(inCamino.c_str());1
  string Camino = "";
  string Nombre = "";
  string Tipo = "";
  string Tamano = "";
  string Propietarios = "";1

  if (Directorio)2
    {
    struct dirent* Puntero;
    while ((Puntero = readdir(Directorio)))3
      {
      struct stat Buffer;
      string Nombre = Puntero->d_name;4
      string Camino = inCamino + "/" + Nombre;4
      if (!stat(Camino.c_str(), &Buffer))5
        {
        if (S_ISDIR(Buffer.st_mode))6
          {
          if (Nombre == "..");7
          else if (Nombre == ".");8
          else CargarFicheros(Camino);9
          }
        else 10
          {
            passwd *Propietario = getpwuid(Buffer.st_uid);10
            time_t Modificado = Buffer.st_mtime;10
            struct tm *Convertido;
            Convertido = localtime(&Modificado);10
            cout << "----------------" << endl;
            cout << "Camino + Nombre:" << Camino << endl;
            cout << "Tamaño en Bytes:" << Buffer.st_size << endl;
            cout << "Propietario:" << Propietario->pw_name << endl;
            cout << "Modificado:" << asctime(Convertido) << endl;
          }
        }13
      }
    closedir(Directorio);14
    }15
}
```

Grafo de Flujo



Calculo de Complejidad Ciclomática

V (G) = Numero de Aristas – Número de Nodos + 2

V (G) = 13 – 16 + 2

V (G) = 5

Caminos Independientes

1: 1-2-15

2: 1-2-3-14

3: 1-2-3-4-5-13

4: 1-2-3-4-5-6-10

5: 1-2-3-4-5-6-7-8-16

Resultados obtenidos para el caso de uso

Nombre Caso de uso: Indexar Información Audiovisual

Para el camino: 5: 1-2-3-4-5-6-7-8-16

Caso de Prueba: Probando la función CargarFicheros

Resultado: Cargar todos los ficheros en la base de datos que posteriormente serán indexados.

No del camino.	Caso de Prueba.	Objetivo.	Resultado.
5	Probando la función CargarFicheros	Cargar todos los ficheros en la base de datos	Satisfactorio.

4.2 **Conclusiones Parciales del capítulo.**

Con la realización de las pruebas se demuestra que los algoritmos aplicados a la indexación y la búsqueda contextual de documentos digitales son factibles, esto se logra a través de pruebas de unidades para inspeccionar y verificar la calidad y eficiencia de los requisitos especificados. Se demostró que es posible el desarrollo de un sistema con buenos resultados de eficiencia.

Conclusiones

Con la realización de este proyecto se cumplieron los objetivos trazados al inicio del mismo, desarrollando un conjunto de actividades para cumplir con el propósito del proyecto planteado. Se profundizó en el análisis del funcionamiento de las tecnologías de reconocimiento del habla y las técnicas de recuperación de información existentes. Posibilitando la implementación de un sistema de recuperación de documentos multimedia mediante el uso de consultas en lenguaje natural.

La herramienta desarrollada es una herramienta modular, siguiendo la tendencia del mercado tecnológico. El módulo desarrollado interactúa con los sistemas de recuperación de información y reconocimiento del habla y pese a las limitaciones de estos sistemas externos utilizados para el desarrollo de la aplicación final, el sistema ofrece la precisión necesaria para acceder correctamente a los documentos de bases de datos multimedia, mediante un procesamiento sencillo de dichos documentos. La principal ventaja de la modularidad es que se puede adaptar de manera sencilla a las necesidades del entorno en que se utilice.

Recomendaciones

Se recomienda:

- Enriquecer el sistema con nuevas funcionalidades, pues el campo del audiovisual es muy amplio y surgen nuevos avances cada día.
- La optimización del código, para lograr un funcionamiento óptimo de la aplicación y un uso adecuado de los recursos.
- Integración de la aplicación de recuperación de audio y video, con un sistema de recuperación de imágenes, permitiendo, de esta forma crear un sistema más completo, que permita la recuperación de cualquier archivo multimedia.
- Poner el sistema a prueba un tiempo determinado, para comprobar que sus funcionalidades se correspondan correctamente a las gestionadas por el cliente.
- Poner su utilización y generalización en la UCI y en Cuba.

Glosario de términos

Spider: es un sistema que inspecciona las páginas de la web de forma metódica y automatizada.

XML: (Extensible Markup Language- lenguaje de marcas extensible) Es un metalenguaje de etiquetas que permite definir la gramática de lenguajes específicos.

UML: (Unified Modeling Language-Lenguaje unificado de modelado) Es un lenguaje de modelado de software. Visualiza, especifica, construye y documenta un sistema.

Plataforma .NET: es un componente se software que posee un conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones.

Licencia GPL: (Licencia Publica General) Es una licencia y está orientada a proteger la libre distribución, modificación y uso del software.

Cómputo: Procedimiento que determina el valor de una cantidad por medio de operaciones matemáticas.

Patrones GRASP: acrónimo de “General Responsibility Assignment Software Patterns”.

Trabajos Citados

Amengual, Sebastian, y otros. 2006. *Motores de Busqueda para contenidos audiovisuales.* Madrid : Universidad Politec (Noeda, 2009)nica de Madrid, 2006.
Bustamante, Paul, Aguinaga, Iker y Aybar, Miguel. 2004. *Aprenda C++ basico.* 2004.
—. 2004. *Aprenda C++ Basico.* 2004.
Bustamante, Paul, Aguinagua, Iker y Aybar, Miguel. 2004. *Aprenda C++ Basico.* 2004.
Caraballo, Perez y Strzalkowski. 2000. *Natural Languaje Information Retrieval:progres reports.* 2000.
Carlos, Universidad III. 2001. Modelo de Recuperacion Probabilistico. Recuperacion y organizacion de la información. [En línea] 2001. http://modelosrecuperacion.tripod.com.
Casaares, Claudio. 2005. Tutorial de SQL Server. [En línea] 2005. www.clikear.com/manuales/sql/default.aspx.
Castillo, Carlos. 2008. *Sistemas gestores de base de datos.* 2008.
Collada Perez, Sonia. 2009. *Sistema de Indexacion y busqueda de documentos audiovisules.* Madrid : s.n., 2009.
Corporation., Microsoft. 2001 . *Introducction to C# Programming for de Microsoft.Net Plataform.* 2001 .
De la Torre, Cesar y Ramos Barroso, Miguel Angel. 2010. *Guia de Arquitecturas N-Capas Orientada al Dominio con .Net 4.0.* 2010.
—. 2010. *Guia de Arquitecturas N-Capas Orientada al Dominio con .Net 4.0.* 2010.
—. 2010. *Guia de Arquitecturas N-Capas Orientadas al Dominio con .Net 4.0.* 2010.
De los Lobos, Maria Elena. 2005. *Aprende a Programar.* 2005.
Fuentes, Lidia, Troya, Jose M. y Vallecillo, Antonio. 2006. *Arquitectura Basada en Componentes.* Malaga, España : s.n., 2006.
Gonzalo, Julio, Peñas, Ancelmo y Verdejo, Felisa. 2002. La Indexacion con Tecnicas Linguisticas en el Modelo Clasico de Recuperación de Información. [En línea] 2002. http://nlp.uned.es/anselmo/articulos/jotri2002.pdf.
GSI. 2007. Rational Rose Enterprise. *Grupo de Soluciones Innova.* [En línea] 2007. http://www.rational.com.ar/herramientas/roseenterprise.html.
IDE. 1990. 1990.
Jacobson y Ivar. 2000. *El Proceso Unificado de Desarrollo.* 2000.
—. 2000. *El Proceso Unificado del Software.* 2000.
Jacosbon, I, Booch, G. y Rumbaugh, J. 2000. *El Proceso Unificado de Desarrollo de software.* 2000.
Jeffery y G, Keith. 2006. The Future of the Information System . [En línea] 2006. http://www.wmo.int/pages/prog/www/WDM/ET-IDM/Doc-2-3.html..
Lon. 1989. *Diccionario Mac Milian de Tecnologias de la Información.* 1989.
Lopez Jose Maria. 2009. SofTonic. *Geany.* [En línea] 2009. http://geany.softonic.com/.

Trabajos Citados

Marques, Maria Mercedes. 2004. Modelo de datos. [En línea] 2004. http://www3.uji.es/mmarques/f47/apun/node32.html.
Meadow. 1992. *CT.Text Information Retriveral Systems .* San Diego : s.n., 1992.
Nuance. 2009. Nuance. *NaturallySpeaking.* [En línea] 2009. http://www.spain.nuance.com/natutallyspeaking/products/default.asp.
—. 2009. Via Voice. *Nuance.* [En línea] 2009. http:www.nuance.com/viavoice.
Pressman, Roger. 2006. *Ingenieria de Software. Un enfoque practico.* 2006.
—. 2006. *Ingenieria de Software. Un enfoque practico.* 2006.
—. 2006. *Ingenieria de Software.Un enfoque practico.* 2006.
—. 2006. *Ingenieria de Sofware. Un Enfoque practico.* 2006.
Salton. 1999. *Introduction to Moderm Information Retrieval.* New York : s.n., 1999.
Sphinx, CMU. 2009. CMU Sphinx. *CMU Sphinx.* [En línea] 2009. http://cmusphinx.sourceforge.net/html/cmushpinx.php.

Bibliografía

Amengual, Sebastian, y otros. 2006. *Motores de Busqueda para contenidos audiovisuales.* Madrid : Universidad Politecnica de Madrid, 2006.
Bustamante, Paul, Aguinaga, Iker y Aybar, Miguel. 2004. *Aprenda C++ basico.* 2004.
—. 2004. *Aprenda C++ Basico.* 2004.
Bustamante, Paul, Aguinagua, Iker y Aybar, Miguel. 2004. *Aprenda C++ Basico.* 2004.
Caraballo, Perez y Strzalkowski. 2000. *Natural Languaje Information Retrieval:progres reports.* 2000.
Carlos, Universidad III. 2001. Modelo de Recuperacion Probabilistico. Recuperacion y organizacion de la información. [En línea] 2001. http://modelosrecuperacion.tripod.com.
Casaares, Claudio. 2005. Tutorial de SQL Server. [En línea] 2005. www.clikear.com/manuales/sql/default.aspx.
Castillo, Carlos. 2008. *Sistemas gestores de base de datos.* 2008.
Collada Perez, Sonia. 2009. *Sistema de Indexacion y busqueda de documentos audiovisules.* Madrid : s.n., 2009.
Corporation., Microsoft. 2001 . *Introducction to C# Programming for de Microsoft.Net Plataform.* 2001 .
De la Torre, Cesar y Ramos Barroso, Miguel Angel. 2010. *Guia de Arquitecturas N-Capas Orientada al Dominio con .Net 4.0.* 2010.
—. 2010. *Guia de Arquitecturas N-Capas Orientada al Dominio con .Net 4.0.* 2010.
—. 2010. *Guia de Arquitecturas N-Capas Orientadas al Dominio con .Net 4.0.* 2010.
De los Lobos, Maria Elena. 2005. *Aprende a Programar.* 2005.
Fuentes, Lidia, Troya, Jose M. y Vallecillo, Antonio. 2006. *Arquitectura Basada en Componentes.* Malaga, España : s.n., 2006.
Gonzalo, Julio, Peñas, Ancelmo y Verdejo, Felisa. 2002. La Indexacion con Tecnicas Linguisticas en el Modelo Clasico de Recuperación de Información. [En línea] 2002. http://nlp.uned.es/anselmo/articulos/jotri2002.pdf.
GSI. 2007. Rational Rose Enterprise. *Grupo de Soluciones Innova.* [En línea] 2007. http://www.rational.com.ar/herramientas/roseenterprise.html.
IDE. 1990. 1990.
Jacobson y Ivar. 2000. *El Proceso Unificado de Desarrollo.* 2000.
—. 2000. *El Proceso Unificado del Software.* 2000.
Jacosbon, I, Booch, G. y Rumbaugh, J. 2000. *El Proceso Unificado de Desarrollo de software.* 2000.
Jeffery y G, Keith. 2006. The Future of the Information System . [En línea] 2006. http://www.wmo.int/pages/prog/www/WDM/ET-IDM/Doc-2-3.html..
Lon. 1989. *Diccionario Mac Milian de Tecnologias de la Información.* 1989.

Bibliografía

Lopez Jose Maria. 2009. SofTonic. *Geany.* [En línea] 2009. http://geany.softonic.com/.
Marques, Maria Mercedes. 2004. Modelo de datos. [En línea] 2004. http://www3.uji.es/mmarques/f47/apun/node32.html.
Meadow. 1992. *CT.Text Information Retriveral Systems .* San Diego : s.n., 1992.
Nuance. 2009. Nuance. *NaturallySpeaking.* [En línea] 2009. http://www.spain.nuance.com/natutallyspeaking/products/default.asp.
—. 2009. Via Voice. *Nuance.* [En línea] 2009. http:www.nuance.com/viavoice.
Pressman, Roger. 2006. *Ingenieria de Software. Un enfoque practico.* 2006.
—. 2006. *Ingenieria de Software. Un enfoque practico.* 2006.
—. 2006. *Ingenieria de Software.Un enfoque practico.* 2006.
—. 2006. *Ingenieria de Sofware. Un Enfoque practico.* 2006.
Salton. 1999. *Introduction to Moderm Information Retrieval.* New York : s.n., 1999.
Sphinx, CMU. 2009. CMU Sphinx. *CMU Sphinx.* [En línea] 2009. http://cmusphinx.sourceforge.net/html/cmushpinx.php.

Anexos

Anexo 1 Descripción textual de los casos de uso

Tabla 3: Descripción textual del caso de uso Gestionar Información Indexada.

Nombre del caso de uso	Gestionar Información Indexada
Actores	Sistema administrador
Propósito	Permite gestionar la información indexada en la base de datos, es decir la actualiza y la elimina.
Resumen	
El caso de uso comienza cuando el sistema administrador le ordena al componente gestionar la información indexada en base de datos, adicionando nueva información y eliminado aquella que está obsoleta.	
Referencias	RF2
Precondiciones	• Información de la base datos indexada
Pos condiciones	• Base de datos actualizada.
Acción del actor	**Respuesta del Sistema**
El sistema administrador debe iniciar al componente con la petición de la gestión de la información indexada en la base de datos.	El componente recibe la solicitud e inicia el proceso de gestión de la información almacenada en la base de datos.
	El componente procede a comprobar si existe nueva información que se deba adicionar a la base de datos
	El componente procede a comprobar si existe información obsoleta que se deba eliminar de la base de datos y notifica la finalización del proceso con un conjunto de estadísticas relacionadas

Flujo Alternativo	
Acción del actor	**Respuesta del sistema**
El sistema administrador realiza la petición de la gestión de la información indexada en la base de datos.	El componente comprueba que la base de datos no necesita ser gestionada y finaliza la acción con una respuesta al sistema administrador.
Requerimientos especiales	

Tabla 4: Descripción textual del caso de uso Realizar búsquedas contextuales.

Nombre del caso de uso	Realizar búsquedas contextuales
Actores	Sistema administrador
Propósito	Permite realizar búsquedas mediante el uso de comodines, frases y oraciones.
Resumen	
El caso de uso inicia cuando el sistema administrador realiza la consulta de la información deseada y culmina una vez que el componente le devuelve un listado ordenado en un orden de relevancia relacionado con la información solicitada en la consulta.	
Referencias	RF3
Precondiciones	• Consulta realizada
Poscondiciones	• Listado de información relacionada con la consulta
Acción del actor	**Respuesta del Sistema**
El sistema administrador debe iniciar al componente con la consulta de la información que desea buscar en la base de datos.	El componente recibe la solicitud e inicia el proceso de búsqueda de la información ya indexada y almacenada en la base de datos.

	El componente procede a comprobar si existe dicha información
	El componente ordena toda la información por un orden de relevancia y devuelve al sistema administrador el resultado de la búsqueda.
Flujo Alternativo	
Acción del actor	**Respuesta del sistema**
El sistema administrador realiza un consulta	El componente comprueba que la información solicitada no existe en la base de datos y lo notifica al sistema administrador, concluyendo el caso de uso
Requerimientos especiales	

Tabla 5: Descripción textual del caso de uso Realizar búsquedas textuales y de contenido.

Nombre del caso de uso	Realizar búsquedas textuales y de contenido
Actores	Sistema administrador
Propósito	Permite realizar búsquedas de documentos (.pdf, .doc, .ppt, .html) entre otros y también permite realizar búsquedas de contenido audiovisual (audio y video).
Resumen	
El caso de uso inicia cuando el sistema administrador le especifica al componente el tipo de búsqueda que desea. Culmina una vez que el componente le devuelve un listado ordenado en un orden de relevancia relacionado con la información solicitada en la consulta.	
Referencias	RF4
Precondiciones	• Consulta realizada

Poscondiciones	• Listado de información relacionada con la consulta
Acción del actor	**Respuesta del Sistema**
El sistema administrador debe especificar al componente el tipo de búsqueda que desea realizar, es decir de texto o de contenido.	El componente recibe la solicitud e inicia el proceso de búsqueda del tipo de información solicitada por el sistema administrador
	El componente procede a mostrarle al sistema toda la información relacionada con el tipo de búsqueda especificado.
El sistema procede a realizar una consulta	El componente ordena toda la información por un orden de relevancia y devuelve al sistema administrador el resultado de la búsqueda.
Flujo Alternativo	
Acción del actor	**Respuesta del sistema**
El sistema administrador especifica al componente el tipo de búsqueda que desea realizar.	El componente comprueba que no existe información relacionada con el tipo de búsqueda que el sistema administrador especificó y lo notifica al sistema administrador, concluyendo el caso de uso.
Requerimientos especiales	

Anexo 2 Diagramas de clases del análisis.

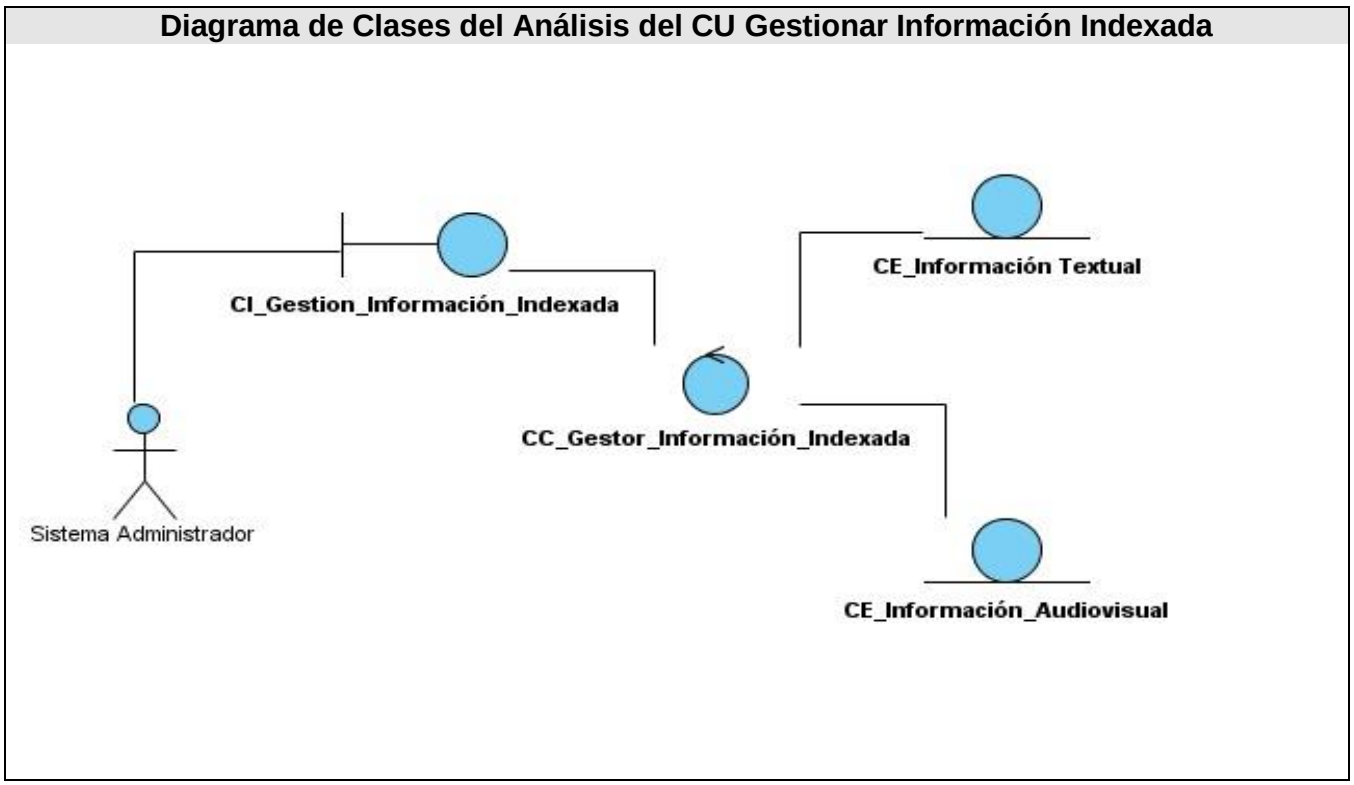


Figure 13: Diagrama de Clases del Análisis del CU Gestionar Información Indexada

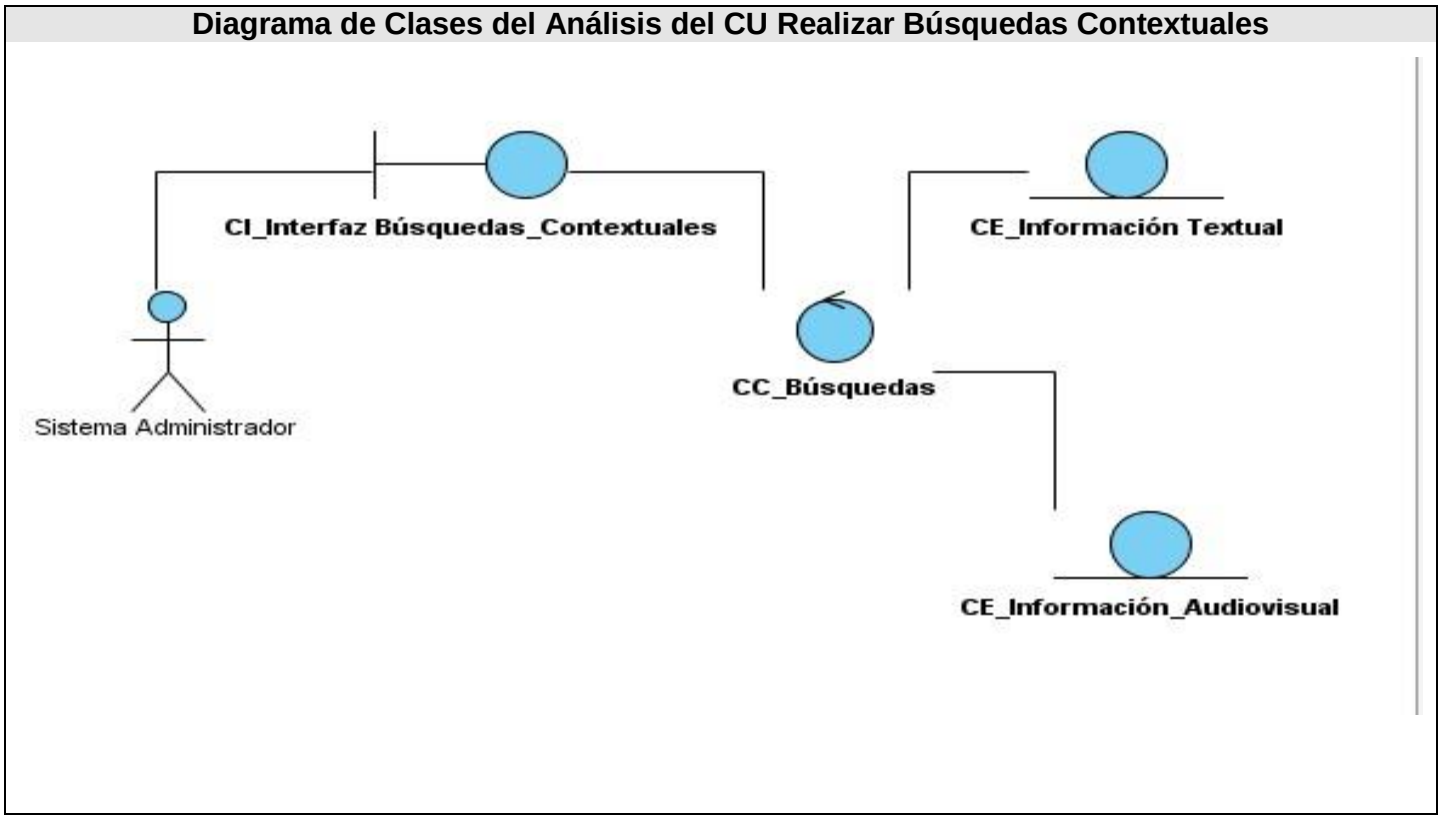


Figure 14: Diagrama de Clases del Análisis del CU Realizar Búsquedas Contextuales

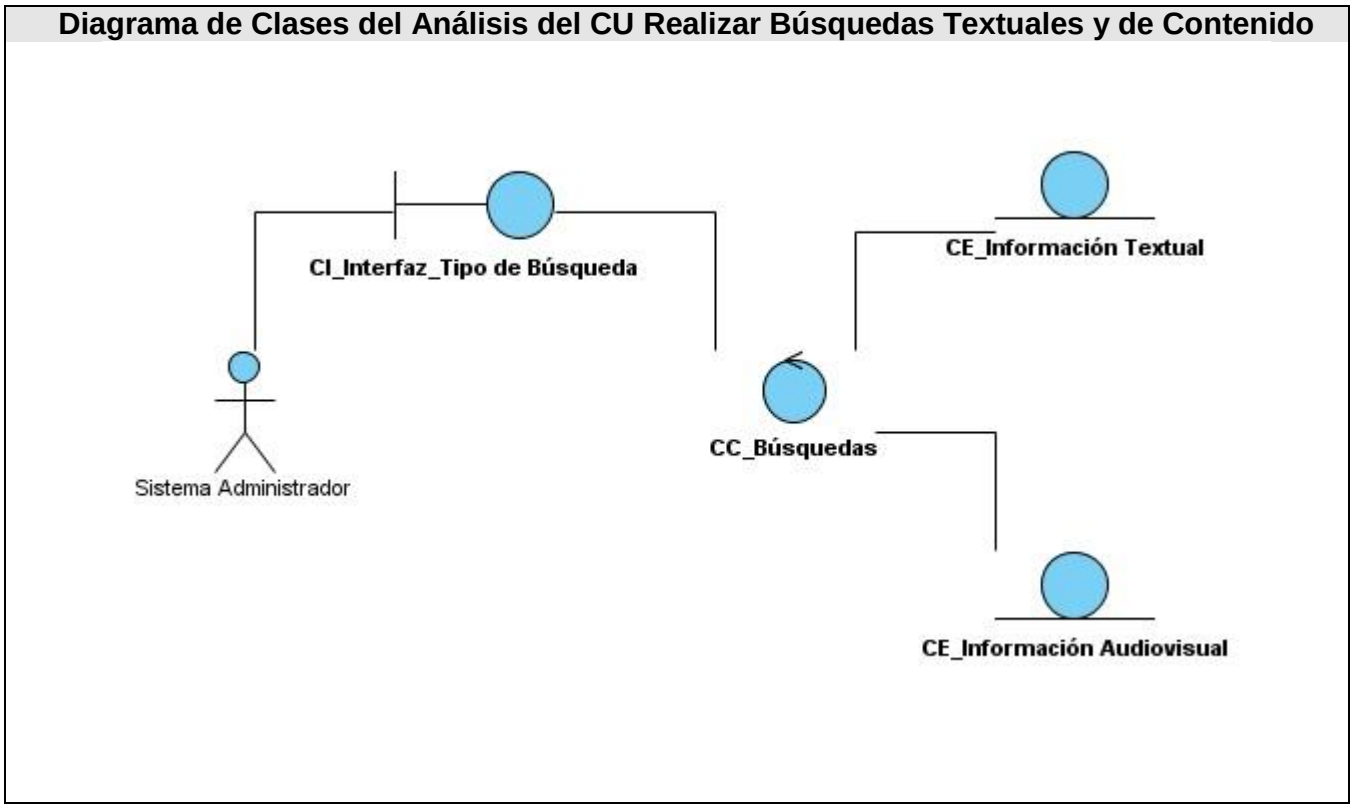


Figure 15: Diagrama de Clases del Análisis del CU Realizar Búsquedas Textuales y de Contenido

Anexo 3 Diagramas de colaboración del análisis.

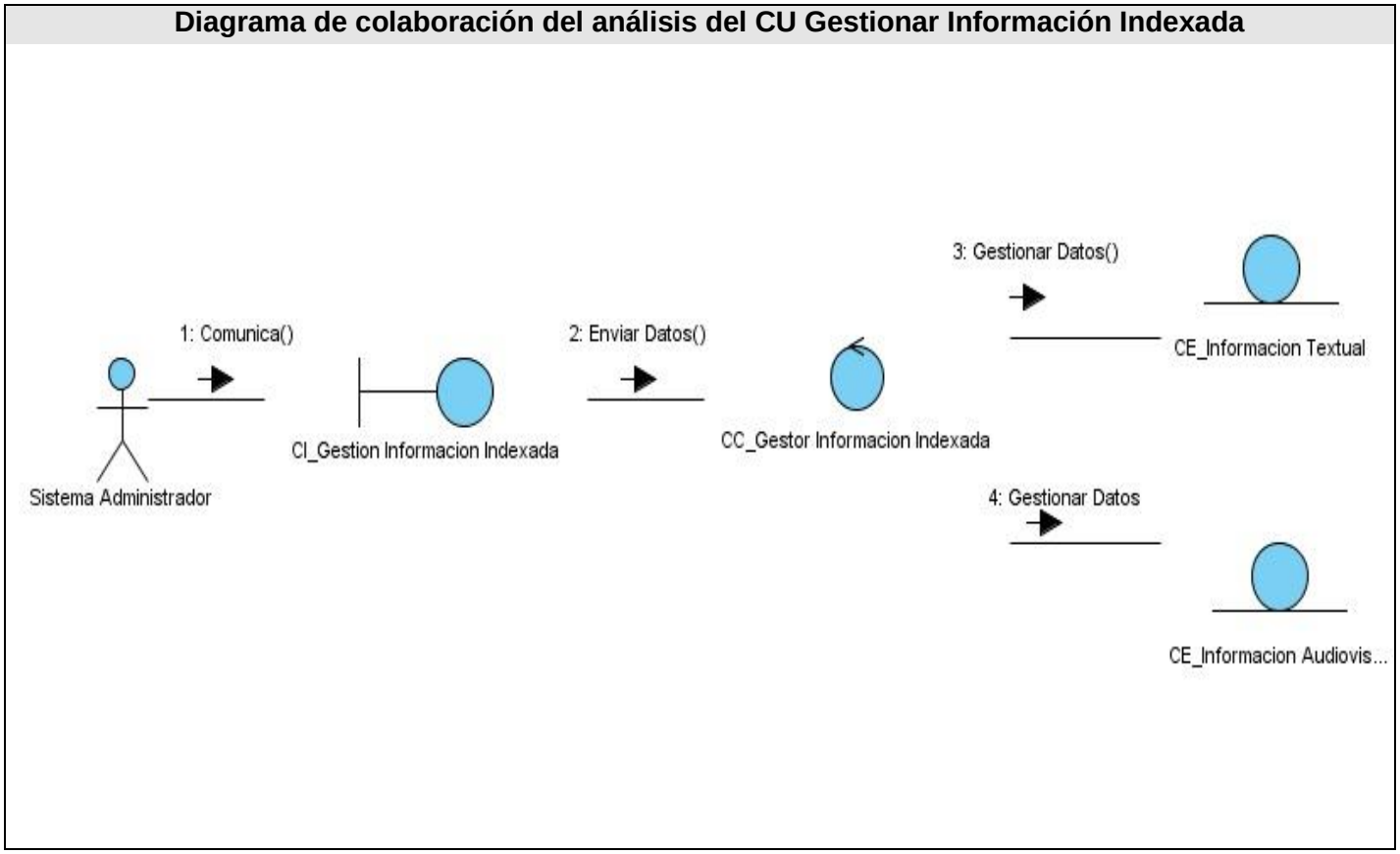


Figure 16: Diagrama de Colaboración del CU Gestionar Información Indexada

Diagrama de colaboración del análisis del CU Realizar Búsquedas Contextuales

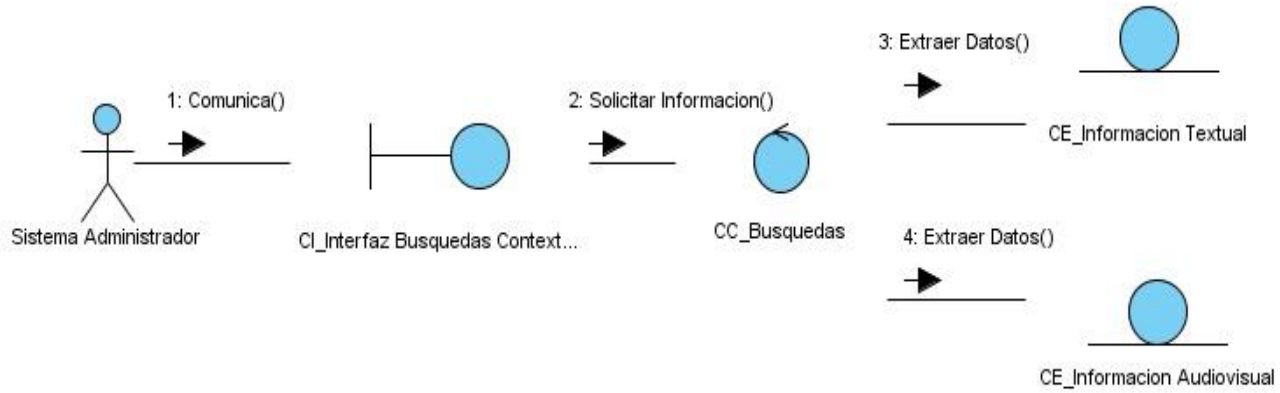


Figure 17: Diagrama de Colaboración del CU Realizar Búsquedas Contextuales

Diagrama de colaboración del análisis del CU Realizar Búsquedas Textuales y de Contenido

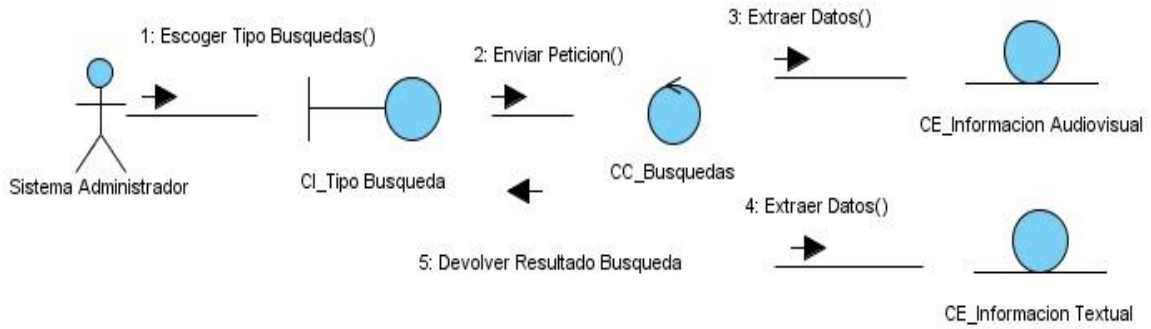


Figure 18: Diagrama de Colaboración del CU Realizar Búsquedas Textuales y de Contenido

Anexo 4 Diagramas de clases del diseño.

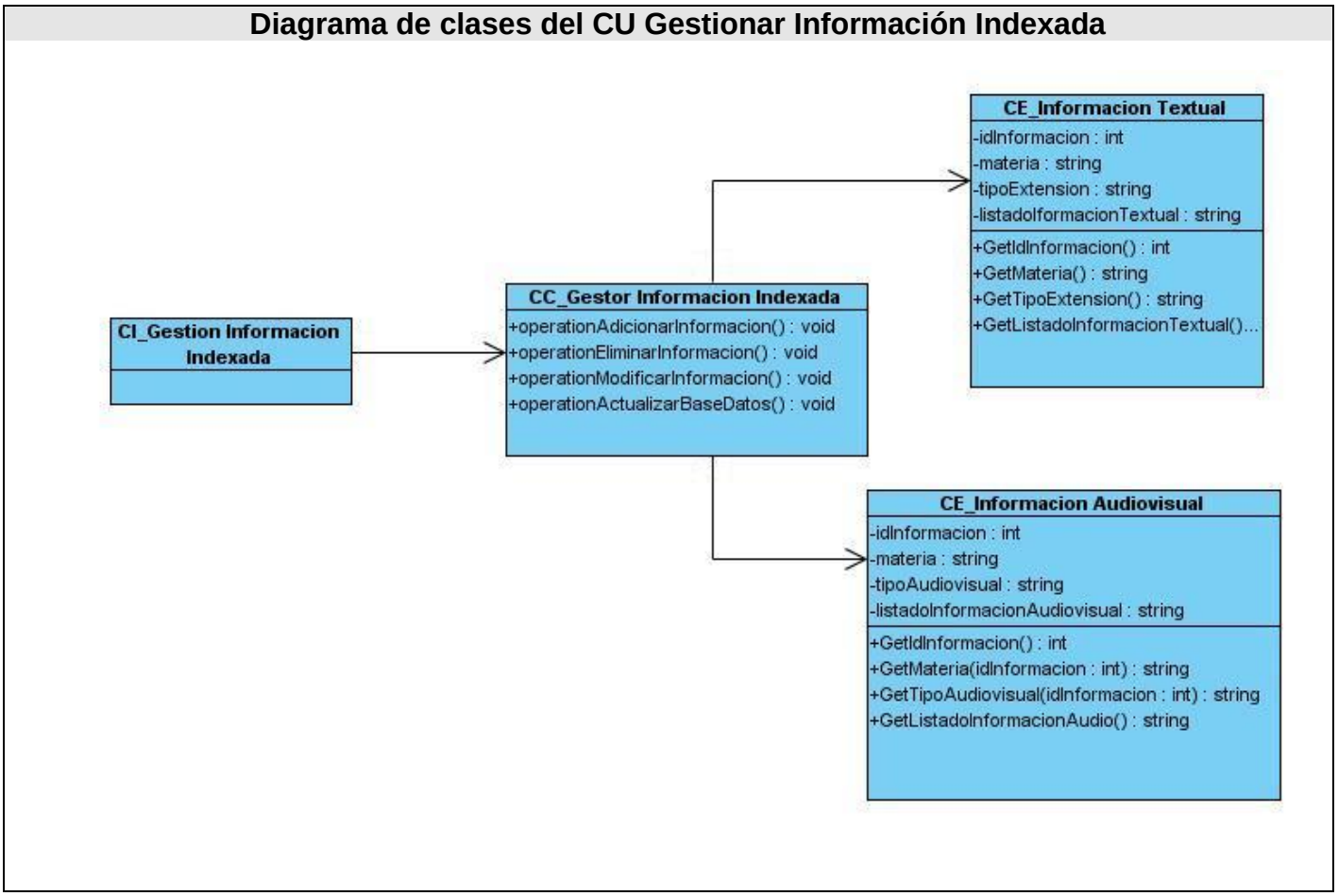


Figure 19: Diagrama de Clases del Diseño del CU Gestionar Información Indexada

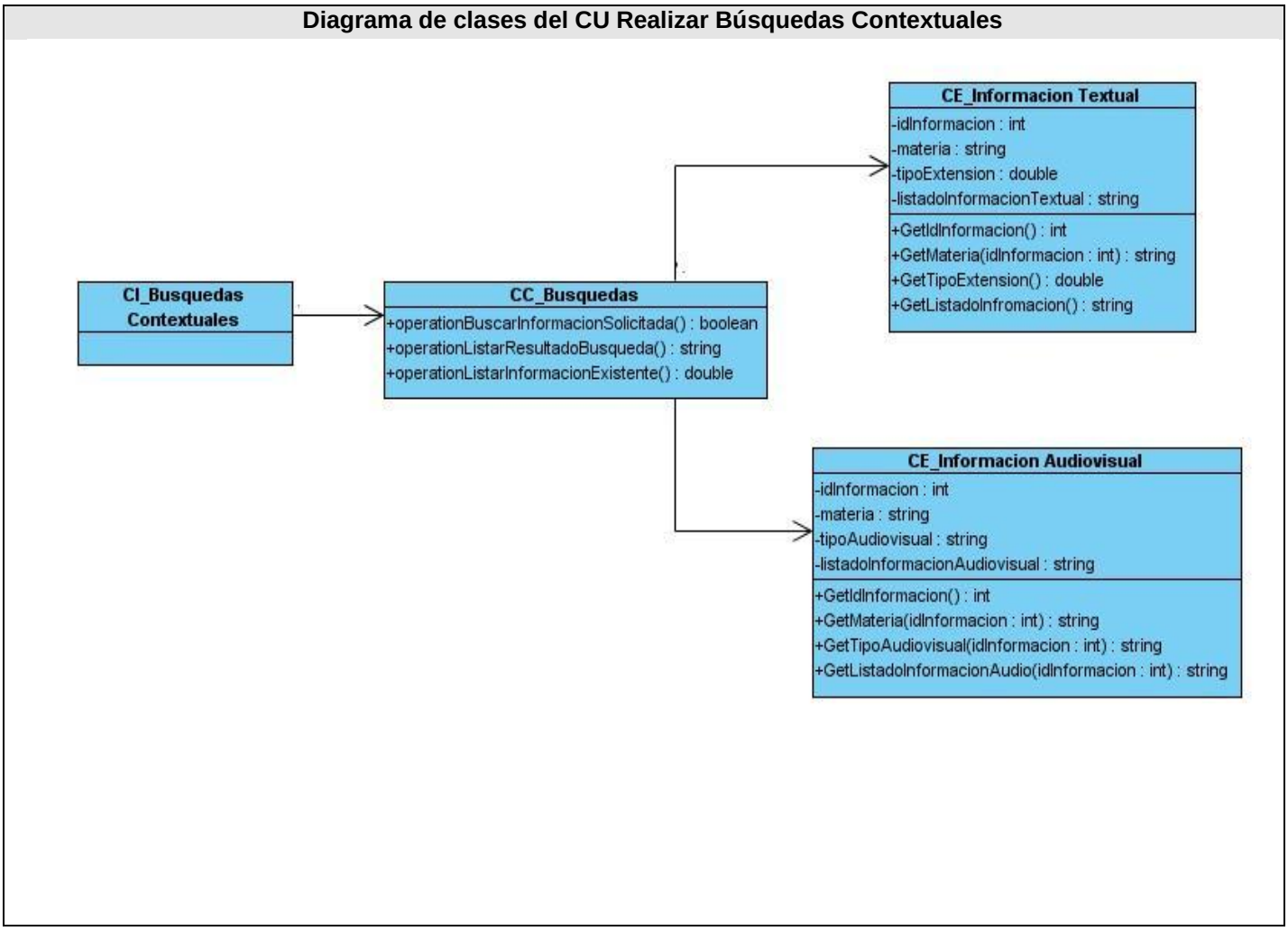


Figure 20: Diagrama de Clases del Diseño del CU Realizar Búsquedas Contextuales

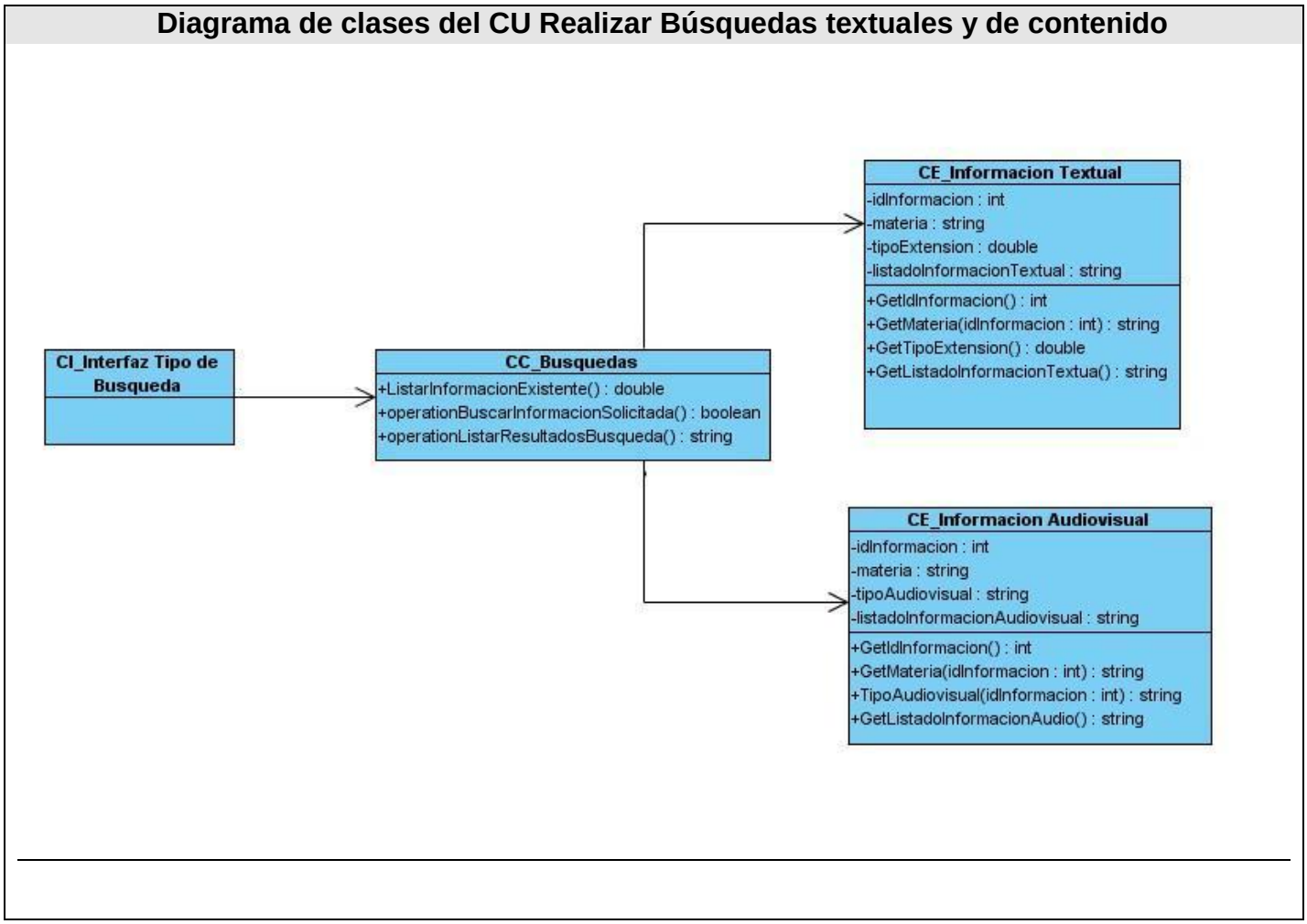


Figure 21: Diagrama de Clases del Diseño del CU Realizar Búsquedas Textuales y de Contenido

Anexo 5 Descripción textual de las clases

Tabla 7: Descripción textual de la clase Gestor Información Indexada.

Nombre: Gestor Información Indexada	
Tipo de clase (controladora)	
Atributo	**Tipo**
No tiene	No tiene
Para cada responsabilidad:	
Nombre:	operacionAdicionarInformacion()
Descripción:	Función encargada de adicionar nueva información a la base de datos.
Nombre:	operaciónEliminarInformacion()
Descripción:	Función encargada de eliminar información obsoleta de la base de datos.
Nombre:	operacionModificarInformación()
Descripción:	Función encargada de modificar alguna información contenida e la base de datos

Tabla 7: Descripción textual de la clase Gestor Información Indexada.

Nombre: Búsquedas	
Tipo de clase (controladora)	
Atributo	**Tipo**
No tiene	No tiene
Para cada responsabilidad:	
Nombre:	ListarInformacionExistente()
Descripción:	Función encargada de listar la información contenida en la base de datos correspondiente al criterio de búsqueda.
Nombre:	operacionBuscarInformacionSolicitada()
Descripción:	Función encargada de responder si existe o no el tipo de información especificada por parte del administrador.
Nombre:	operacionListarResultadosBusqueda()
Descripción:	Función encargada de devolver el resultado de la búsqueda.

Anexos