Universidad de las Ciencias Informáticas

Facultad 2

Trabajo de diploma para optar por el título Ingeniero en Ciencias Informáticas

Componente de búsqueda para el Sistema de Información Hospitalaria del Centro de Informática Médica

Autores: Dayana González Vera

Ernesto Cabrera Viciana

Tutores: Ing. Nadiezka Milan Cristo

Ing.Yasser Manuel Garbey Bermudes

La Habana, junio de 2015

"Año 57 de la Revolución"

"La alegría de ver y entender es el más perfecto don de la naturaleza"

Albert Einstein

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los __ días del mes de __ del año ____.

_____	_____
Dayana González Vera	Ernesto Cabrera Viciana
Autora	Autor
_____	_____
Ing. Nadiezka Milan Cristo	Ing. Yasser Manuel Garbey Bermudes
Tutora	Tutor

Datos de Contacto

Ing. Nadiezka Milan Cristo

Graduado en el año 2007 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Jefa del departamento de desarrollo de componentes. Ha impartido las asignaturas Inteligencia Artificial, Práctica Profesional y Gestión de Software.

Correo electrónico: nmilan@uci.cu

Ing. Yasser Manuel Garbey Bermudes

Ingeniero en Ciencias Informáticas, graduado en la Universidad de las Ciencias Informáticas en el año 2009. Posee la categoría docente de Instructor Recién Graduado. Ha impartido asignaturas de Introducción a la Programación, Programación 2, Ingeniería de Software 1 e Ingeniería de Software 2. Ha participado en varios proyectos de desarrollo vinculados al perfil de salud. Actualmente es especialista del departamento de desarrollo de componentes.

Correo electrónico: ymgarbey@uci.cu

Agradecimientos

Dayana

A mis padres, mi novio, familiares, tutores y amigos.

Ernesto

A mis padres, hermana, a mi novia, familiares, tutores y amigos.

Dedicatoria

Dayana

A mis padres.

Ernesto:

A mis padres, hermana, a mi novia, familiares y amigos.

Resumen

El Sistema de Información Hospitalaria del Centro de Informática Médica cuenta con mecanismos que garantizan la realización de los procesos de búsqueda. Actualmente estos muestran resultado cuando en la base de datos existe representación exacta del término buscado, pero su implementación no permite satisfacer todas las necesidades reales.

Con el objetivo de garantizar la obtención de los resultados esperados por el usuario, se determina la necesidad de desarrollar el componente de búsqueda para el Sistema de Información Hospitalaria del Centro de Informática Médica que permita estimar la similitud entre cadenas de textos. Este centra su funcionamiento en la implementación de los algoritmos Damerau-Levenshtein y Metaphone.

El desarrollo de las funcionalidades es guiado por el Proceso Unificado de Desarrollo de Software. Se utiliza Java como lenguaje de programación, Visual Paradigm para el modelado y Jboss Developer Studio como herramienta de desarrollo. Se selecciona como servidor de aplicaciones Jboss Server y PostgreSQL como Sistema Gestor de Base de Datos. Además se incorporan a estas herramientas *frameworks* y librerías.

Con el desarrollo del componente se espera brindar sugerencias del término buscado, lo que permite aumentar la precisión de las búsquedas en el sistema.

Palabras clave: búsqueda, Damerau-Levenshtein, Metaphone, Sistema de Información Hospitalaria.

Tabla de Contenidos

Tabla de Contenidos

Figuras

Tablas

Introducción

El avance científico y tecnológico que adquiere el hombre a finales del siglo XX permite consolidar a la informática como uno de los frentes tecnológicos con mayor capacidad de desarrollo e innovación. La informática es una ciencia que desarrolla métodos, procedimientos y técnicas con el objetivo de almacenar información para procesar datos y transmitirlos en formato digital. La introducción y ascendente utilización de productos informáticos beneficia diferentes esferas causando un positivo impacto en la sociedad. En la rama de la salud pública también es evidente, brindando soluciones concretas a problemas como el considerable aumento de los volúmenes de datos que genera el uso de las tecnologías de la información.

Debido a la demanda que existe en las instituciones hospitalarias de información precisa y oportuna, surge la necesidad de desarrollar diversas aplicaciones informáticas que tienen como objetivo, mejorar la gestión de información durante el proceso de atención al paciente, con el fin de incrementar su eficiencia, entre estas aplicaciones se encuentran los Sistemas de Información Hospitalaria (HIS, por sus siglas en inglés).

Los HIS son sistemas de gestión que permiten la recolección, almacenamiento, procesamiento, recuperación y comunicación de información de atención al paciente para todas las actividades relacionadas con la institución de la salud (1). Además, posibilitan controlar y manejar toda la documentación que allí se administra, pues la misma genera una inagotable fuente de conocimientos que habitualmente se pierde porque se encuentra dispersa, deteriorada o simplemente no está disponible cuando se necesita.

El desarrollo de la informática en la salud es también visible en Cuba, incrementándose para esto la creación de centros de desarrollo de software para este sector. Tal es el caso del Centro de Informática Médica (CESIM), que desarrolla productos informáticos de alta calidad para instituciones hospitalarias. El HIS que desarrolla el CESIM permite la gestión de la información que se obtiene en múltiples instituciones hospitalarias. Para ello cuenta con 17 módulos relacionados entre sí que responden a cada una de las áreas de un hospital. Este sistema contempla además un conjunto de funcionalidades que permiten gestionar la información, desde su creación hasta finalmente su visualización a través de las disímiles búsquedas sobre los diferentes términos y conceptos que maneja el sistema.

Con el objetivo de generalizar la terminología que se maneja en las diferentes instituciones de la salud se usan diversos estándares de codificación. En el HIS del CESIM, la terminología asociada

a las enfermedades se rige por la Clasificación Internacional de Enfermedades en su décima versión (CIE-10). El mismo consta de 3 volúmenes: el primero contiene los grupos de enfermedades con su estructura; el segundo agrupa las recomendaciones para certificación y clasificación que incluye el 1er volumen, así como el uso para codificar las causas de muerte y el tercer volumen es el índice general. (2)

Los nombres de los medicamentos se encuentran almacenados según el Sistema de Clasificación Anatómica, Terapéutica, Química (ATC por sus siglas en inglés), el código asignado a cada fármaco recoge el sistema u órgano sobre el que actúa, el efecto farmacológico, las indicaciones terapéuticas y la estructura química del medicamento (3) y los diagnósticos de enfermería son clasificados según la Asociación Norteamericana de Diagnósticos de Enfermería (NANDA por sus siglas en inglés).

Durante la explotación del sistema se comprueba que el uso de estos estándares en reiteradas ocasiones, dificulta los procesos de búsqueda que realizan los diferentes usuarios del sistema; debido a que estos no están familiarizados con la terminología que se utiliza para nombrar las enfermedades, medicamentos y diagnósticos de enfermería ya que es compleja y variada. En la actualidad el sistema cuenta con diferentes mecanismos que posibilitan realizar búsquedas, pero no están implementados para detectar errores de escritura y sugerir el término más semejante que se encuentra en el diccionario sobre el que se realiza la búsqueda.

En sistemas donde son más frecuentes las búsquedas y existe variedad de contenido, se recomienda crear diferentes bases terminológicas o diccionarios especializados en cada uno de los temas que integran la base de datos. Esto, posibilita contar con un componente especializado en los procesos de búsqueda, que implemente mecanismos de reconocimiento terminológico. Los mismos se basan fundamentalmente en comparar los elementos del texto fuente con los contenidos del término base y si encuentra una coincidencia, el registro de la expresión en cuestión se muestra para que el usuario lo consulte.

Teniendo en cuenta la situación anterior, se identifica el siguiente **problema a resolver:** El proceso de búsqueda del Sistema de Información Hospitalaria del Centro de Informática Médica, dificulta la obtención de los resultados esperados durante la realización de esta acción en el sistema.

Introducción

El **objeto de estudio** está constituido por el proceso de búsqueda en sistemas de gestión de información, enmarcado en el **campo de acción:** los mecanismos de búsqueda en sistemas de gestión de información.

Para darle solución al problema identificado se propone el siguiente **objetivo general:** Desarrollar un componente de búsqueda para el Sistema de Información Hospitalaria del Centro de Informática Médica que facilite la obtención de los resultados esperados.

Para dar solución al objetivo planteado se definen las siguientes **tareas de la investigación**:

1. Analizar los diferentes algoritmos relacionados con los procesos de búsquedas que permitan dar solución al problema planteado.
2. Asimilar la arquitectura y pautas definidas para el desarrollo de los artefactos, el diseño y las aplicaciones del Sistema de Información Hospitalaria del Centro de Informática Médica.
3. Obtener mediante el Proceso Unificado de Desarrollo, los artefactos correspondientes a las disciplinas de Modelado de negocio, Gestión de requisitos, Diseño e Implementación.
4. Implementar las funcionalidades necesarias para desarrollar el componente de búsqueda del Sistema de Información Hospitalaria del Centro de Informática Médica.

Con el desarrollo del componente de búsqueda del Sistema de Información Hospitalaria del Centro de Informática Médica se espera obtener el siguiente **beneficio**:

- Aumentar la precisión de las búsquedas en el Sistema de Información Hospitalaria del Centro de Informática Médica, siendo capaz de brindar, del término buscado, el resultado más semejante que exista en la base de datos.

El documento se encuentra estructurado en cuatro capítulos, organizados de la siguiente manera:

CAPÍTULO 1: Fundamentación teórica de la investigación. Se realiza un análisis de los algoritmos y funciones existentes que permiten estimar la similitud entre cadenas de textos y obtener la fonología de los términos. Se plantean además las tendencias y tecnología utilizada para la elaboración de las funcionalidades a desarrollar.

CAPÍTULO 2: Características del componente de búsqueda. Contiene un marco conceptual asociado a la información que es manipulada por el sistema, llegándose a un acuerdo sobre las funcionalidades, requisitos deseados y el objeto de automatización.

CAPÍTULO 3: Diseño del componente de búsqueda. Se centra en el modelado detallado y la construcción de la estructura de la aplicación.

CAPÍTULO4: Implementación del componente de búsqueda. Se describen las clases y subsistemas en términos de componentes. Se presenta la propuesta de solución para lograr una gestión más eficiente de los procesos de búsqueda del Sistema de Información Hospitalaria del Centro de Informática Médica.

Capítulo 1. Fundamentación teórica de la investigación

En el presente capítulo se exponen los principales conceptos y características que constituyen las bases para la implementación del componente de búsqueda. Se realiza un estudio de los algoritmos vinculados al campo de acción que pueden brindar una solución al problema planteado, de igual forma se describe la tecnología con la que se lleva a cabo el proceso de desarrollo del componente.

1.1 Conceptos básicos relacionados con el dominio del problema

La búsqueda de información en sistemas informáticos es el proceso mediante el cual el sistema examina las representaciones de la información y las compara con las de la base de datos, dichas tareas son ejecutadas por los componentes de búsquedas. Estos permiten definir tanto los índices por los que se realiza la búsqueda como el modo en el que esta se efectúa.

En el ámbito informático una base de datos es una entidad que permite almacenar datos de manera estructurada, con la menor redundancia posible (4). Si contiene solo información específica de alguna disciplina determinada se denomina diccionario especializado, ya que maneja un metalenguaje[1] que no constituye la totalidad de la lengua común y además aporta información descriptiva de los términos que almacena. En cambio, si la base de datos almacena terminología e información relacionada entre sí se denomina base de datos terminológica. Comúnmente esta terminología es compleja, variada y en ocasiones está clasificada por un grupo de estándares.

Los estándares son normas documentadas que contienen requisitos, especificaciones, directrices o características que pueden emplearse consistentemente para asegurar que los materiales, productos, procesos y servicios son adecuados para su propósito. (5)

Alguno de los estándares que se utilizan en el Sistema de Información Hospitalaria del Centro de Informática Médica son:

- CIE10: Clasificación Internacional de Enfermedades en su décima versión: Creada por la Organización Mundial de la Salud (OMS) en 1948, con el objetivo de estandarizar los nombres de las enfermedades, y además, asignarles a cada una de ellas un código único que la representa permitiendo un fácil almacenamiento y posterior recuperación para el análisis de la información. (2)

[1]Es el lenguaje utilizado para referirse no al objeto del discurso, sino al lenguaje que se refiere al objeto del discurso.

- NANDA: Asociación Norteamericana de Diagnósticos de Enfermería (NANDA por sus siglas en inglés): Es una sociedad científica de enfermería cuyo objetivo es estandarizar los diagnósticos de enfermería.
- ATC: Código o Sistema de Clasificación Anatómica, Terapéutica, Química (ATC por sus siglas en inglés): Es un índice de sustancias farmacológicas y medicamentos, organizados según grupos terapéuticos. Este sistema fue instituido por la Organización Mundial de la Salud y es adoptado en Europa. (3)

Para realizar búsquedas sobre bases de datos que contienen términos poco comunes es necesario contar con mecanismos que implementen algoritmos capaces de reconocer y comparar a través de un umbral[2] de semejanza varias expresiones.

Los algoritmos son secuencias de instrucciones que representan un modelo de solución para determinados tipos de problemas y son independientes del lenguaje de programación. (6)

1.2 Descripción de los algoritmos estudiados

La búsqueda informática sobre diccionarios de datos toma valor desde que comienza el empleo de los diferentes sistemas de gestión de información. Para lograr la eficiencia en el manejo de los grandes volúmenes de datos que estos sistemas generan, es necesario contar con mecanismos encargados de estimar el grado de similitud que existe entre dos términos.

Los términos pueden ser similares de dos formas: léxica y semánticamente. Dos cadenas se consideran léxicamente similares si contienen secuencias de caracteres semejantes. A continuación se explican los algoritmos que implementan las funciones más utilizadas y reconocidas para calcular la similitud léxica entre cadenas de texto.

1.2.1 Funciones de similitud basadas en caracteres

Estas funciones de similitud consideran cada cadena de caracteres como una secuencia ininterrumpida de caracteres.

Distancia de edición

La distancia de edición entre dos cadenas de texto A y B se basa en el conjunto mínimo de operaciones de edición necesarias para transformar A en B o viceversa. Las operaciones de edición permitidas son eliminación, inserción y sustitución de un caracter.

[2]Valor más bajo o pequeño de una magnitud que puede generar cierto efecto.

- **Distancia de Levenshtein[3]**: es un algoritmo que dadas dos cadenas de caracteres devuelve un número entero que expresa la distancia o parecido entre ellas. El número entero se calcula contando las transformaciones que son necesarias para obtener a partir de una de las cadenas la otra. Las posibles operaciones que permite realizar son: sustitución, inserción y eliminación de caracteres. (7)
- **Distancia de Damerau-Levenshtein:** Este algoritmo permite, además de las operaciones de inserción, borrado y sustitución de un único caracter, la transposición (o intercambio) entre dos caracteres adyacentes. Es esta última operación lo que la diferencia de la distancia de Levenshtein y es introducida por el autor, F.J Damerau[4], porque según él las 4 operaciones permiten corregir más del 80% de los fallos humanos en escritura con una sola operación. (8)
- **Distancia de Hamming[5]:** Este algoritmo solo permite una operación, la de sustitución, lo que implica que ambas cadenas han de tener una misma longitud. Esta restricción provoca que prácticamente esté en desuso, puesto que generalmente las cadenas no suelen tener la misma longitud. (8)

Distancia de Brecha Afín

La distancia de edición y otras funciones de similitud tienden a fallar identificando cadenas equivalentes que han sido demasiado truncadas, ya sea mediante el uso de abreviatura o la omisión de *tokens*[6]. Por ejemplo la cadena "Jorge Eduardo Rodríguez López" puede transformase a "Jorge E Rodríguez", de esta manera se omite el término "López" y "Eduardo" se sustituye por "E". La distancia de Brecha Afín ofrece una solución al penalizar la inserción o eliminación de k caracteres consecutivos (brecha) con bajo costo, mediante una función $p(k) = g+h\cdot(k-1)$, donde g es el costo de iniciar una brecha, h el costo de extenderla un caracter, y $h < g$. (8)

Similitud Smith-Waterman

La similitud Smith-Waterman entre dos cadenas A y B es la máxima similitud entre una pareja (A´, B´), sobre todas las posibles, tal que A´ es subcadena de A, y B´ es subcadena de B. Tal problema se conoce como alineamiento local. El modelo original de Smith[7] y Waterman[8] define las mismas

[3]Científico ruso que realizó investigaciones sobre la teoría de la información y los códigos de corrección de errores.
[4]Matemático y lingüístico estadounidense pionero en el campo de la inteligencia artificial.
[5]Matemático estadounidense que trabajó en temas relacionados con la informática y las telecomunicaciones.
[6]Cadena de caracteres que tiene un significado coherente en cierto lenguaje.
[7]Físico estadounidense que trabajó en el Centro de Investigaciones de Ingeniería Biomolecular.

operaciones de la distancia de edición, y además permite omitir cualquier número de caracteres al principio o final de ambas cadenas. Esto lo hace adecuado para identificar cadenas equivalentes con prefijos o sufijos que, al no tener valor semántico, pueden ser descartados. Esta similitud puede ser normalizada con base en la longitud de la cadena de mayor longitud, la de menor longitud o la longitud media de ambas cadenas. (8)

Similitud de Jaro

La similitud de Jaro es la función que define la transposición de dos caracteres como la única operación de edición permitida. Los caracteres no necesitan ser adyacentes, sino que pueden estar alejados cierta distancia *d* que depende de la longitud de ambas cadenas. La similitud de Jaro[9] puede ser calculada con un orden de complejidad de *O(n)*. Winkler propone una variante que asigna puntajes de similitud mayores a cadenas que compartan algún prefijo. (8)

Similitud de q-grams

Un q-gram, es una subcadena de longitud q. El principio tras esta función de similitud es que, cuando dos cadenas son muy similares, tienen muchos q-grams en común. Es común usar uni-grams (q=1), bi-grams (q=2) y tri-grams (q=3). Es posible agregar q-1 ocurrencias de un carácter especial al principio y final de ambas cadenas. Esto lleva a un puntaje de similitud mayor entre cadenas que compartan algún prefijo y/o sufijo, aunque presenten diferencias hacia el medio. (8)

1.2.2 Funciones de similitud basadas en *tokens*

Estas funciones de similitud consideran cada cadena como un conjunto de subcadenas separadas por caracteres especiales y calculan la similitud entre cada pareja de *tokens* mediante alguna función de similitud basada en caracteres.

Similitud de Monge-Elkan

Monge-Elkan plantean que la distancia entre A y B es la similitud máxima promedio entre una pareja (αI,,βj). Donde A y B son dos cadenas y α1, α2 …α k y β1, β2 …βL sus *tokens* respectivamente. Para cada *token* αI existe algún βj de máxima similitud. (8)

[8]Científico estadounidense que centró su estudio en la aplicación de las matemáticas, estadísticas y técnicas informáticas a diversos problemas de la biología molecular.
[9] Máster en Ciencias de la Computación y licenciado en Matemáticas de origen norteamericano.

Similitud cosena TF-IDF

Dadas dos cadenas A y B, sean α1, α2 …α k y β1, β2 …βL sus *tokens* respectivamente, que pueden verse como dos vectores VA y VB de K y L componentes, la similitud entre estas es el coseno del ángulo que forman sus respectivos vectores. No es eficaz bajo la presencia de variaciones a nivel de caracteres, como errores ortográficos o variaciones en el orden de los *tokens*. (8)

A continuación se muestra una comparación de las funciones mencionadas anteriormente, adaptando cada una de ellas a distintas situaciones problémicas.

Tabla 1: Comparación de las funciones de similitud (8)

Situación Problemática	**Posición**								
	1	2	3	4	5	6	7	8	9
Errores ortográficos	LE	SW	2G	3G	BA	ME	JA	SW	JW
Abreviaturas	BA	ST	SW	ME	3G	2G	LE	JA	JW
Tokens Faltantes	SW	3G	BA	LE	2G	SW	ME	JA	JW
Prefijos / Sufijos	ST	BA	3G	SW	2G	ME	LE	JA	JW
Tokens en desorden	3G	2G	ST	ME	JA	BA	JW	SW	LE
Espacios en blanco	SW	3G	LE	2G	BA	SW	ME	JA	JW

Leyenda:

BA: Brecha Afín ST: Coseno TF-IDF SW: Smith-Waterman LE: Levenshtein
2G: Bi-grams 3G: Tri-grams ME: Monge-Elkan JA: Jaro
JW: Jaro-Winkler

La tabla anterior muestra cuales son las funciones que más se adaptan a los diferentes escenarios. Por ejemplo, para detectar errores de escritura lo mejor es aplicar Levenshtein y en el peor de los casos se calcula la distancia de Jaro-Winkler, pues es la función menos apropiada ante esta situación, según la prioridad que muestra la tabla. Esto evidencia que no todas las funciones son adecuadas para los diferentes problemas, sino que su aplicación depende de la situación que pretenda resolver.

Después de aplicar alguna de estas funciones a una situación específica es necesario normalizar la distancia obtenida, debido a que tres errores son más significativos en una cadena de longitud 4 que en una de longitud 20. Para esto se proponen varias técnicas de normalización. Las más

simples normalizan dividiendo por la longitud de la cadena más larga o por la suma de la longitud de ambas cadenas. (8)

1.2.3 Algoritmos de emparejamiento fonético

La existencia de estos algoritmos obedece a la necesidad de recuperar información que tiene una semejanza sonora y cuya representación a través de la palabra escrita pueda diferir de su pronunciación.

Soundex

Es un algoritmo de codificación fonética, que convierte una palabra en un código. Consiste en sustituir las consonantes de la palabra afectada por un número, si es necesario se agregan ceros al final para conformar un código de 4 dígitos.

Debido a que en el idioma inglés, las letras A, E, I, O, U, H, W y Y no hacen diferenciación fonética, son descartadas. Adicionalmente existen otras reglas complementarias para la codificación de las letras dobles (si el texto tiene letras dobles, estas se deben tratar como una sola) y para letras a lado y lado con el mismo código (si el texto tiene diferentes letras lado a lado que tienen el mismo número en la guía de codificación Soundex, estas se tratan como una sola letra), entre otras.

Las limitaciones del algoritmo Soundex se encuentran ampliamente documentadas y dan lugar a varias mejoras, pero ninguna orientada hacia el idioma español. Es claro, que Soundex no está orientado al español ya que ni siquiera contempla el juego de caracteres (“ñ”, “ll”). Asimismo, la dependencia de la letra inicial, la agrupación por punto de articulación del idioma inglés y el estar limitado a cuatro caracteres implica que no es eficiente para detectar errores ortográficos comunes en el idioma español. (9)

Metaphone

El algoritmo Metaphone aparece como respuesta a las deficiencias de Soundex. La idea detrás del algoritmo es codificar explícitamente las reglas comunes de pronunciación, con las que Soundex no trabaja. El núcleo de su funcionamiento consiste en una serie de reglas que asignan a cada combinación de letras una determinada clase consonante, es por esto que supone una mejora sobre Soundex. (9)

Por lo descrito anteriormente se determina utilizar la distancia de edición implementando el algoritmo Damerau-Levenshtein, ya que es el más eficaz para detectar errores de escritura.

Además, por las mejoras que presenta sobre el Soundex se decide aplicar el algoritmo Metaphone para obtener la fonología de los términos a buscar.

1.3 Tecnología a utilizar

En este epígrafe se especifica la tecnología a utilizar en el proceso de desarrollo de la solución. La misma fue definida para el desarrollo de las aplicaciones del Sistema de Información Hospitalaria del Centro de Informática Médica.

1.3.1 Tecnología

A continuación se realiza una descripción de dicha tecnología.

- **XHTML 1.0:** Es el lenguaje de marcado pensado para sustituir a HTML como estándar para las páginas web. XHTML es solamente la versión XML de HTML, por lo que tiene, básicamente, las mismas funcionalidades, pero cumple las especificaciones, más estrictas de XML. (10)
- **Java Server Face (JSF) 1.2:** Es un *framework* Java que permite crear interfaces de usuario (UI) para una aplicación web, mediante el uso de componentes reutilizables. Permite el manejo de estados y eventos, así como la asociación entre los datos de la interfaz y los datos de la aplicación web. (11)
- **Facelets 1.1.15.B1:** Es un *framework* simplificado de presentación, en donde es posible diseñar de forma libre una página web y luego asociarle los componentes JSF específicos. Aporta mayor libertad al diseñador y mejora los informes de errores que tiene JSF. (12)
- **Ajax4JSF:** Es una librería de código abierto que se integra totalmente en la arquitectura de JSF y extiende la funcionalidad de sus etiquetas, dotándolas con tecnología Ajax de forma limpia y sin añadir código JavaScript. (11)
- **Richfaces 3.3.0.GA:** Es una biblioteca de componentes JSF y un avanzado *framework* para la integración de Ajax con facilidades en la capacidad de desarrollo de aplicaciones de negocio. Se integra completamente dentro del ciclo de vida JSF. Permite crear interfaces de usuario modernas de manera eficiente y rápida, basadas en componentes listos para usar, altamente configurables en cuanto a temas y esquemas de colores predefinidos por el propio *framework* o desarrollados a conveniencia. (13)

- **Seam UI 2.1.1 GA:** Es un conjunto de controles altamente integrables con JBoss Seam. Se utiliza para complementar los controles JSF incorporados y los controles de otras bibliotecas externas. (14)
- **Seam 2.1.1:** Es un poderoso y moderno *framework* creado para unificar todas las tecnologías estándares JSF, EJB3, JPA. Es un software de código abierto que asocia diferentes tecnologías y estándares Java, adicionando algunas funcionalidades no contempladas en *frameworks* anteriores. Seam integra transparentemente la administración de procesos del negocio vía jBoss-jBPM, haciendo muy fácil implementar y optimizar complejas colaboraciones e interacciones con el usuario. (14)
- **Hibernate 3.0:** Es una herramienta de Mapeo objeto-relacional para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) que establecen estas relaciones. Es un software libre, distribuido bajo los términos de la licencia GNU LGPL. (15)
- **Enterprise Java Bean 3 (EJB 3):** Es una arquitectura de componentes de servidor que simplifica el proceso de construcción de aplicaciones de componentes empresariales distribuidos en Java. Un EJB es un componente de software que se ejecuta del lado del servidor en una aplicación multicapa. Los clientes del EJB acceden a él por medio de una interfaz que esconde los detalles de implementación del componente. Esta interfaz debe cumplir la especificación EJB. (12)
- **Java Enterprise Edition 6 (JEE 6):** Es la plataforma que provee SunMicrosystem para dar soporte y desarrollar software para las empresas. El gran éxito de java como plataforma para el desarrollo de aplicaciones se encuentra en esta especificación, que no es más que un conjunto de librerías que establecen un estándar para lograr un producto altamente calificado. (15)
- **Java:** Es una plataforma de software. Está conformada por: el lenguaje de programación Java, la máquina virtual de Java, que permite la portabilidad en ejecución y el API Java, que es una biblioteca estándar para el lenguaje. Una de las características más importantes de Java es que los programas “ejecutables”, que crea su compilador, son independientes de la arquitectura. Además es gratis, universal y de fácil distribución. (15)

1.3.2 Metodologías de desarrollo

Para obtener el componente de búsqueda es necesario seguir una metodología que oriente todo el proceso de desarrollo. Una metodología es un conjunto de procedimientos y pasos que define lo que se tiene que hacer, cuándo y cómo. Para la elaboración del componente de búsqueda del Sistema de Información Hospitalaria se utiliza RUP y para la confección de los artefactos correspondientes a cada fase se usa UML como lenguaje de modelado.

El Proceso Unificado de Desarrollo (RUP, por sus siglas en inglés) junto al Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es una de las metodologías de desarrollo de software más utilizada en el mundo para el análisis, implementación y documentación de sistemas. Es dirigido por casos de uso, los mismos reflejan lo que el cliente necesita y desea, lo cual se capta al modelar el negocio y se representa a través de los requisitos. A partir de aquí los casos de uso guían el proceso de desarrollo, ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso. Además es centrado en la arquitectura, esto muestra la visión común del sistema en la que el equipo de desarrollo y los clientes deben de estar de acuerdo, a su vez es iterativo e incremental ya que cada fase se desarrolla en iteraciones. (16)

El Lenguaje Unificado de Modelado establece un conjunto de notaciones y diagramas para el modelado de sistemas. Está organizado en paquetes y estructurado por capas que permiten a los desarrolladores crear diseños de forma fácil para compartirlos con otros desarrolladores o clientes. (17)

1.3.3 Herramientas de desarrollo

A continuación se describen las herramientas que se usan para el desarrollo del componente de búsqueda.

PostgreSQL 8.4

PostgreSQL es un gestor de base de datos relacional de código abierto. Se ejecuta sobre la mayoría de los sistemas operativos que existen en la actualidad. Tiene como objetivo manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convierte en información relevante. (18)

PgAdmin III

PgAdmin es una aplicación gráfica para interactuar con el de bases de datos PostgreSQL. Está escrita en C++ usando la librería gráfica multiplataforma wxWidgets, lo que permite que se pueda usar en Linux, FreeBSD, Solaris, Mac OS X y Windows. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3 ejecutándose en cualquier plataforma, así como versiones comerciales de PostgreSQL como PervasivePostgres, EnterpriseDB, MammothReplicator y SRA PowerGres. (19)

Visual Paradigm 8.0

Visual Paradigm es una poderosa herramienta para visualizar y diseñar elementos de software. Está orientada a la creación de diseños usando el paradigma de programación orientada a objetos. Provee soporte para la generación de código, tiene integración con diversos IDE's como NetBeans (de SunMicrosystems), Developer (de Oracle), Eclipse (de IBM), JBuilder (de Borland), así como la posibilidad de realizar la ingeniería inversa para aplicaciones JAVA, .NET, XML e Hibernate. (20)

JBoss Developer Studio 7.1.1

Jboss Developer Studio es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés). Contiene un conjunto de herramientas para desarrollo de software. Está compuesto por un compilador, un depurador, un editor de código y un constructor de interfaz de usuario, además está construido sobre un mecanismo para descubrir, integrar y ejecutar módulos. (14)

JBoss Server 4.2.2 GA

Jboss Server es el servidor de aplicaciones de código abierto más ampliamente desarrollado del mercado. Por ser una plataforma certificada J2EE, soporta todas las funcionalidades de J2EE 1.4 e incluye servicios adicionales como *clustering*, *caching* y persistencia. JBoss es ideal para aplicaciones Java y aplicaciones basadas en la web. También soporta Enterprise Java Beans (EJB) 3.0, lo que hace el desarrollo de las aplicaciones mucho más simple. Además, al ser desarrollado con tecnología Java, es multiplataforma. (14)

1.4 Conclusiones del capítulo

- La utilización de algoritmos especializados en obtener la fonología y similitud entre términos facilita la realización de los procesos de búsqueda, pues estos permiten obtener de la base de datos las representaciones más semejantes del término que se está buscando.
- La tecnología definida para el Sistema de Información Hospitalaria del Centro de Informática Médica, permite la obtención de un componente flexible y multiplataforma.

Capítulo 2. Características del componente de búsqueda

En este capítulo, se explica el contexto en el cual se desarrolla el componente y se muestra el Modelo de dominio, donde se exponen los conceptos fundamentales relacionados con el componente y los vínculos que se establecen entre ellos. Se detalla el funcionamiento de los algoritmos Damerau-Levenshtein y Metaphone utilizados en la solución del problema. Por último se enumeran los requisitos funcionales y no funcionales a partir de los cuales se representan en un diagrama los casos de uso del componente de búsqueda.

2.1 Modelo de Dominio

Con el objetivo de lograr una mejor comprensión del funcionamiento actual de las búsquedas que se ejecutan en el Sistema de Información Hospitalaria se realiza una breve descripción de los conceptos asociados al Modelo de dominio. Un Modelo de dominio explica cuales son y cómo se relacionan los conceptos relevantes en la descripción del problema, siendo una útil herramienta de comunicación. (21)

Un usuario es cualquier persona que tiene acceso al sistema. Este, según los privilegios que tenga asignado, puede acceder a la información que se gestiona y que está almacenada en la base de datos del sistema. La misma contiene términos que abordan un tema específico y términos que son clasificados por distintos estándares. Ejemplos de estos últimos son: los diagnósticos de enfermería nombrados por la Asociación Norteamericana de Diagnósticos de Enfermería, la Clasificación Internacional de Enfermedades encargada de nombrar las enfermedades y el Sistema de Clasificación Anatómica, Terapéutica, Química se emplea para manejar la terminología asociada a los medicamentos.

Los usuarios dentro del sistema pueden hacer uso de las búsquedas para acceder a la información almacenada. Estos procesos de búsqueda se implementan a través de simples consultas SQL a la base de datos, que devuelven un resultado siempre y cuando exista representación exacta del término introducido. A menudo los usuarios cometen errores de escritura y más aún si realizan búsquedas de términos que no son comunes. Ante esta situación el sistema no es capaz de mostrar respuesta a la petición de la búsqueda, ya que no está implementado para distinguir dentro de la base de datos los términos más similares que existen y sugerirlos al usuario.

A continuación se muestra el diagrama del Modelo de dominio del componente con los conceptos que maneja y las relaciones entre ellos.

Diagrama de Modelo de dominio

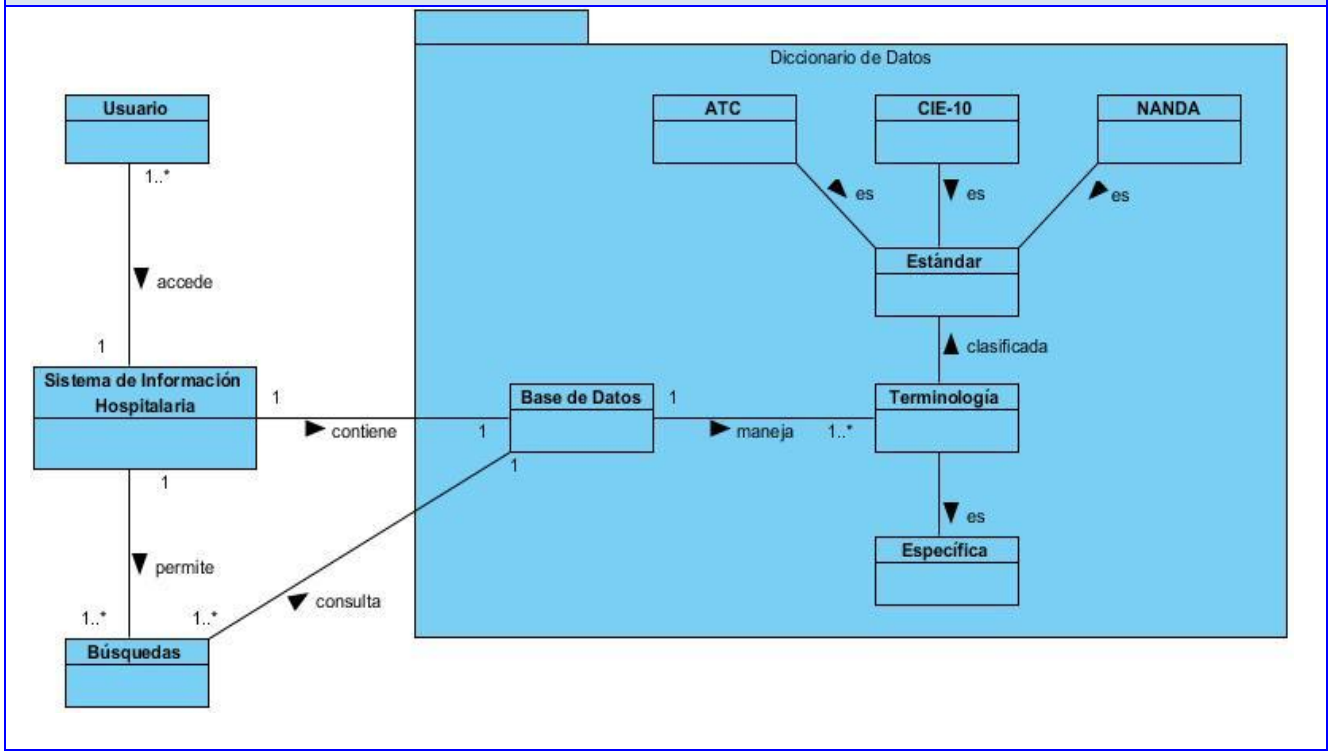

Figura 1: Modelo de dominio

2.2 Propuesta del componente de búsqueda

Después del estudio realizado y con el objetivo de mejorar los procesos de búsqueda en el Sistema de Información Hospitalaria del CESIM, se propone la creación de funcionalidades que faciliten las búsquedas de información en cualquiera de los módulos del sistema.

Estas funcionalidades permiten:

- Calcular la similitud que existe entre el término introducido por el usuario y el contenido que existe en los diccionarios de datos.
- Ordenar los términos obtenidos en la búsqueda y seleccionar a través de un umbral definido los mejores casos.
- Almacenar las búsquedas con su resultado en una tabla hash[10] para su posterior uso.
- Devolver el resultado de la búsqueda.

[10]Estructura de datos que asocia llaves o claves con valores.

Las funcionalidades propuestas posibilitan que el sistema busque sobre los diccionarios de datos el término introducido por el usuario sin importar que esté mal escrito, si no hay representación exacta del vocablo el componente se encarga de mostrar al usuario los términos más semejantes, a través de la implementación de algoritmos que permiten calcular la similitud entre cadenas y realizar el emparejamiento fonético entre ellas.

Para obtener el componente de búsqueda del Sistema de Información Hospitalaria de CESIM con las funcionalidades descritas anteriormente se implementan los algoritmos Damerau-Levenshtein y Metaphone. A continuación se explica de manera detallada el funcionamiento de cada uno de ellos.

2.2.1 Damerau-Levenshtein

El algoritmo funciona de la siguiente manera:

- La longitud de la palabra A es x, la longitud de la palabra B es y. Si x=0, retornar y; si y=0 retornar x.
- Construir una matriz con y+1 filas y x+1 columnas. Inicializar la primera fila de la matriz con la secuencia 0, 1, 2, ..., x; y la primera columna de la matriz con la secuencia 0, 1, 2, ..., y.
- Colocar cada caracter de la palabra A en su correspondiente celda i (i va de 1 a x).
- Colocar cada caracter de la palabra B en su correspondiente celda j (j va de 1 a y).
- Si A(i) es igual a B(j) el costo de la celda es 0.
- Si A(i) es diferente de B(j) el costo de la celda es 1.
- El valor de la celda d(i,j) es el mínimo entre:
    - El valor de la celda (i-1,j) + 1.
    - El valor de la celda (i,j-1) + 1.
    - El valor de la celda (i-,j-1) + costo de celda.
- La distancia es la celda d(x,y).

Ejemplo: La distancia de Damerau-Levenshtein para las palabras *p1=funcional* y *p2=disfuncional* es igual a 3, tal y como se puede observar en cada uno de los siguientes pasos.

Longitud(*p1*)=9 Longitud(*p2*)=12

Paso 1: Construcción de la matriz.

Tabla 2: Algoritmo Damerau-Levenshtein (Paso 1)

		D	I	S	F	U	N	C	I	O	N	A	L
	0	1	2	3	4	5	6	7	8	9	10	11	12
F	1												
U	2												
N	3												
C	4												
I	5												
O	6												
N	7												
A	8												
L	9												

Paso 2: Comienzan las iteraciones del algoritmo.

Tabla 3: Algoritmo Damerau-Levenshtein (Paso 2)

		D	I	S	F	U	N	C	I	O	N	A	L
	0	1	2	3	4	5	6	7	8	9	10	11	12
F	1	1	2	3	3	4	5	6	7	8	9	10	11
U	2												
N	3												
C	4												
I	5												
O	6												
N	7												
A	8												
L	9												

Como *p1*(1)=F y *p2*(1)=D, A(1) ≠B(1) por tanto el costo de la celda es 1. El valor de la celda sería el menor de:

- El valor de la celda (i-,j) + 1: 1+1=2
- El valor de la celda (i,j-1) + 1: 1+1=2
- El valor de la celda (i-1,j-1) + costo de celda: 0+1=1

Se realiza el mismo procedimiento hasta llenar toda la fila y así sucesivamente hasta completar la matriz.

Tabla 4: Algoritmo Damerau-Levenshtein (Matriz final)

		D	I	S	F	U	N	C	I	O	N	A	L
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
F	**1**	1	2	3	3	4	5	6	7	8	9	10	11
U	**2**	2	2	3	4	3	4	5	6	7	8	9	10
N	**3**	3	3	3	4	4	3	4	5	6	7	8	9
C	**4**	4	4	4	4	5	4	3	4	5	6	7	8
I	**5**	5	4	5	5	5	5	4	3	4	5	6	7
O	**6**	6	5	5	6	6	6	5	4	3	4	5	6
N	**7**	7	6	6	6	7	6	6	5	4	3	4	5
A	**8**	8	7	7	7	7	7	7	6	5	4	3	4
L	**9**	9	8	8	8	8	8	8	7	6	5	4	③

Con la implementación de este algoritmo se obtiene la distancia entre las palabras comparadas, esta distancia expresa numéricamente cuan semejantes son dichos términos. Para normalizar la distancia, se divide el valor obtenido entre la longitud de la cadena de texto más larga, en este caso 12 que le corresponde a p(2). Mientras más cercano a 0 sea el valor obtenido luego de normalizar más similares son los términos.

2.2.2 Metaphone

El algoritmo Metaphone recibe un término y hace las transformaciones necesarias a través de reglas definidas de emparejamiento para obtener su fonética.

Ejemplo: El término bencilpenicilina es transformado a bnsilpnisilina.

La subcadena **be** queda sustituida por b, la subcadena **ci** por si y la subcadena **pe** por p.

Con la implementación de este algoritmo se obtiene el término transformado con su fonética según las reglas definidas. El uso del mismo es gran importancia para el funcionamiento del componente, pues en dependencia del resultado que arroje se decide si se aplica el algoritmo Damerau-Levenshtein a los términos o a su fonética.

2.3 Especificación de requisitos del componente

Los requisitos son las condiciones o capacidades que tiene que alcanzar el componente para satisfacer un acuerdo. Según Wiegers[11] se clasifican en funcionales y no funcionales.

[11]Norteamericano ingeniero de software, consultor y formador en las áreas de desarrollo de software, gestión y mejora de procesos.

2.3.1 Requisitos funcionales

Los requisitos funcionales definen las operaciones que el componente es capaz de realizar y describen las transformaciones que realiza este sobre las entradas para producir las salidas (22). Luego del análisis de las funcionalidades a implementar para el desarrollo del componente de búsqueda se define el siguiente requisito funcional:

RF1: Ejecutar componente de búsqueda.

Este requisito permite calcular la similitud entre el término a buscar y lo que hay almacenado en el diccionario de datos, luego muestra los casos más semejantes y guarda la consulta realizada con su respuesta en una tabla hash.

2.3.2 Requisitos no funcionales

Para realizar un componente que pueda integrarse al HIS se deben tener en cuenta los requisitos no funcionales del sistema. Los Requisitos no funcionales son las características que de una forma u otra pueden limitar el funcionamiento del componente. (22)

- **Portabilidad**

  El componente se puede utilizar de dos formas, añadiendo al sistema el paquete de clases java o simplemente agregándolo como una librería más.

- **Eficiencia**

  Las funcionalidades relacionadas con el componente de búsqueda del sistema, permiten que los usuarios sin importar la escritura del término obtengan siempre que exista una respuesta a su petición, facilitándole así su trabajo.

- **Hardware**

  Los requisitos de hardware se definen por la dirección del Centro de Informática Médica atendiendo a experiencias anteriores de despliegue.

  Servidor de aplicaciones: Disco duro 80 GB, RAM 16 GB, CPU 8 núcleos.

  Servidor de base de datos: Disco duro 80 GB, RAM 8 GB, CPU 8 núcleos.

  Dependiendo de la cantidad de estaciones de trabajo, las características de los servidores pueden aumentar. Para los servidores de aplicaciones y base de datos, es necesario aclarar que inicialmente requiere 80 GB pero con el tiempo será necesario contar con más espacio

disponible para poder almacenar toda la información que genera el sistema, dígase documentos clínicos, estudios y base de datos.

- **Software**

  El componente se integra al Sistema de Información Hospitalaria del Centro de Informática Médica, por lo que el sistema operativo deber ser de las distribuciones Linux de 64 bits Debian, Ubuntu u OpenSuse, utilizando la plataforma Java (Máquina Virtual de Java – Java Enterprise Edition), el servidor de aplicaciones Jboss AS y como sistema para la gestión de base de datos PostgreSQL. Los usuarios deben disponer de un navegador web, que puede ser Google Chrome, Firefox 3.6 o versiones superiores de estos para conectarse al sistema.

2.4 Modelo de casos de uso del componente

El comportamiento de un requisito funcional se representa a través de un caso de uso que describe la secuencia de interacciones entre el sistema y sus actores. La representación de todos los casos de uso conforma el Modelo de casos de uso del componente.

El Modelo de casos de uso es un diagrama del sistema que contiene actores, casos de uso y sus relaciones. Permite que los desarrolladores del software y los clientes lleguen a un acuerdo sobre los requisitos, es decir, sobre condiciones y posibilidades que debe cumplir el sistema. Su ventaja principal es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente. (21)

2.4.1 Diagrama de casos de uso

Figura 2: Diagrama de casos de uso

2.4.2 Especificación de los casos de uso

CU1: Realizar búsqueda. (Ver artefacto: Especificación de casos de uso del sistema HIS-Elementos comunes)

CU2: Ejecutar componente de búsqueda

Tabla 5: Descripción del caso de uso "Ejecutar componente de búsqueda"

Objetivo	Realizar búsquedas en el sistema.
Actores	Sistema
Resumen	El sistema ejecuta el componente de búsqueda, esto permite calcular la similitud que existe entre el término a buscar y lo que hay almacenado en el diccionario de datos, luego muestra los casos más semejantes y guarda la consulta realizada con su respuesta en una tabla hash.
Complejidad	Alta
Prioridad	Alta
Precondiciones	El componente reciba el término a buscar, el esquema de la tabla, el nombre de la tabla, el umbral, si se desea respetar el umbral, la cantidad de elementos que desea almacenar en la tabla hash y el Entity Manager.
Postcondiciones	Se encontró el término, no se encontró el término o se realiza una sugerencia del término buscado correctamente.
Flujo de eventos	
Flujo básico	
Sistema	

1. Comprueba si el término fue buscado anteriormente, de ser así devuelve la lista de elementos más semejantes que le corresponde a dicha búsqueda.
2. Si el término no ha sido buscado con anterioridad, se le halla la fonética con el algoritmo Metaphone.
3. Se ejecuta un método que comprueba si se va a utilizar la lista de sonido y otro para ver si lo introducido por el usuario es un término o frase. Luego se llena el diccionario de datos con los elementos tal y como están en la base de datos. En caso de utilizarse la lista de sonido se llena el diccionario de datos transformados con los elementos de la base de datos luego de aplicarles el algoritmo Metaphone. En caso contrario el diccionario de datos transformados contendrá los elementos tal y como están en la base de datos.
4. Para un término:

   Se comprueba si lo introducido por el usuario es número o una cadena, en el último caso si su longitud es igual o mayor que 1:

   4.1. Numero:

4.1.1. Se busca en el diccionario de datos. Si lo encuentra almacena en una tabla hash el código y la lista de términos iguales al introducido por el usuario.

4.1.2. Si no encuentra el término, devuelve un objeto vacío.

4.2. Cadena con longitud igual a uno (1):

4.2.1. Busca todos los términos del diccionario de datos que contienen la letra, selecciona y ordena ascendentemente los que la tienen en la primera posición, almacena en una tabla hash el código y la lista de términos que la contienen.

4.2.2. Si la letra no está en la primera posición ordena de forma ascendente todos los términos que la contengan, almacena en una tabla hash el código la lista de términos que la contienen.

4.2.3. Si no existe algún término del diccionario de datos que contenga la letra devuelve un objeto vacío.

4.3. Cadena con longitud mayor que uno (1):

4.3.1. Se busca el término introducido por el usuario en el diccionario de datos. Si lo encuentra almacena en una tabla hash el código y la lista de términos que son iguales o contienen al introducido por el usuario.

4.3.2. Si no lo encuentra y se decidió utilizar la lista de sonido, se busca en el diccionario de datos transformado el sonido del término introducido por el usuario.

4.3.2.1. Si lo encuentra almacena en una tabla hash el código y la lista de términos que son iguales o contienen el introducido por el usuario.

4.3.2.2. Si no lo encuentra se aplica el algoritmo Damerau-Levenshtein pasándole los parámetros determinados. Se fija un umbral para seleccionar las mejores respuestas.

4.3.2.2.1. Si encuentra términos que cumplan con el umbral de selección se ordenan de menor a mayor y se devuelven las que tengan el menor umbral y sean iguales. Se almacena en una tabla hash el código y la lista de términos más semejantes al introducido por el usuario.

4.3.2.2.2. Si ninguna de estas cumple con el umbral de selección devuelve el objeto vacío.

4.3.3. Si no lo encuentra y no se decidió utilizar la lista de sonido, se ejecuta desde el paso 4.3.2.2 hasta el 4.3.2.2.2

Para frase:

Lista de sonido:

Se ejecuta desde el paso 4.1 hasta el 4.3.2.1 • Se analiza si estos términos al menos se encuentran por separados en alguno de los posibles valores almacenados en el diccionario de datos transformado. ▪ Si lo encuentra almacena en una tabla hash el código y la lista de términos que son iguales o contienen al introducido por el usuario. Se ejecuta del paso 4.3.2.2 hasta 4.3.2.2.2 No lista de sonido: Se ejecuta desde el paso 4.3.2.2 hasta el 4.3.2.2.2		
Relaciones	**CU incluidos**	No existen
	CU extendidos	No existen

2.5 Conclusiones del capítulo

- La descripción del Modelo de dominio permite un mejor entendimiento del proceso de negocio, a partir del cual se identifican las funcionalidades necesarias para el desarrollo del componente.

Capítulo 3. Diseño del componente de búsqueda

En este capítulo se describe la concepción arquitectónica y los patrones de diseño definidos para el desarrollo del componente de búsqueda. Se exponen las estrategias de integración empleadas en el desarrollo del mismo. Además se realiza el diagrama de clases de diseño y el diagrama de paquetes.

3.1 Descripción de la arquitectura

Para obtener un componente que logre satisfacer los requisitos definidos inicialmente es necesario definir una arquitectura que guíe todo el proceso de desarrollo. “La arquitectura de un sistema es un marco conceptual completo que describe su forma y estructura (sus componente y la manera en que se integran)” (23), permite a los programadores, diseñadores y analistas trabajar bajo una línea común que les posibilite la compatibilidad necesaria para lograr el objetivo deseado.

La arquitectura de un sistema puede basarse en un modelo o estilo arquitectónico particular. Para la creación de las funcionalidades asociadas al componente de búsqueda se hará uso del patrón de diseño de arquitectura Modelo Vista Controlador (MVC).

Patrón arquitectónico Modelo-Vista-Controlador (MVC)

El patrón MVC está diseñado para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronización de datos haciendo a los sistemas más flexibles y adaptables. Su característica principal es que el modelo, las vistas y los controladores se tratan como entidades separadas, lo que permite su desarrollo independiente, garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo.

La capa de presentación está compuesta por páginas XHTML, desarrolladas básicamente con JSF, utilizando las librerías Ajax4JSF y RichFaces, que se complementan con la plataforma de integración Jboss Seam. Se utilizan también componentes Seam de interfaz de usuario y Facelets como motor de plantillas. El uso de estos componentes enriquece el diseño de la interfaz de usuario.

La capa de negocio es la encargada de manipular y procesar la información. Está compuesto por clases Java controladoras, que definen la lógica del negocio, a las cuales, mediante anotaciones que provee el *framework* Seam, se le especifica el contexto en el que se encuentran, los cuales definen el estado de las entidades y datos que manejan.

La capa de acceso a datos se encarga principalmente de la carga, modificación, eliminación y persistencia de la información en la base de datos. Esta capa valida también los datos antes de persistirlos. El encargado de realizar esta tarea es Hibernate, el cual abstrae al desarrollador del gestor de base de datos haciendo uso del mapeo de tablas, esto permite llevar las consultas a un lenguaje de objetos.

Las principales ventajas que proporciona el uso de este patrón son:

- Soporte de vistas múltiples: Dado que la vista se halla separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación web pueden utilizar el mismo modelo de objetos, mostrado de maneras diferentes.
- Adaptación al cambio: Los requisitos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Dado que el modelo no depende de las vistas, agregar nuevas opciones de presentación generalmente no afecta al modelo. (23)

3.2 Estrategia de integración

El componente de búsqueda no responde a un módulo específico del Sistema de Información Hospitalaria, por lo que es diseñado para integrarse al sistema de dos formas, agregando el componente como una librería o añadiendo un paquete compuesto por las clases java: CC_StringUtils que realiza todo el trabajo con cadenas, CC_DamerauLevenshtein encargada de calcular la distancia entre dos términos, CC_Phonetic se usa para hallar la fonética de los términos, CC_QuickShort se utiliza para ordenar, CC_Suggestions que recibe los parámetros (término, esquema, tabla, lista de campos de la tabla, umbral, si desea respetar el umbral, cantidad de elementos que se guarda en la tabla hash, entityManager) y devuelve la sugerencia encontrada, y la clase CC_HashMap es la encargada de guardar las búsquedas realizadas.

3.3 Modelo de diseño

El modelo de diseño se define durante la disciplina de Diseño del Proceso Unificado de Desarrollo, este modelo describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema. Está compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.

Para la elaboración del diseño, generalmente se utilizan un grupo de patrones o modelos para lograr los objetivos específicos. Estos patrones expresan esquemas para definir estructuras de diseño con las que se construyen sistemas automatizados, son una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular. Facilitan la reutilización del conocimiento experto como componentes de diseño mejorando así la documentación, comprensión, comunicación del diseño final y el aprendizaje al programador inexperto, permitiendo que el software construido sea más fácil de mantener y extender. De manera general, los patrones constituyen soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos, basados en la experiencia de su funcionamiento. (23)

Para llevar a cabo el diseño del componente se tienen en cuenta varios de los Patrones para la Asignación General de Responsabilidades (GRASP, por sus siglas en inglés).

La clase CC_Suggestions se crea con el objetivo de dar sugerencias al término introducido por el usuario, en ella se representa el patrón alta cohesión ya que realiza solo funciones específicas y designa las demás gestiones a otras clases. Esta clase además es la encargada de crear y guiar la asignación de responsabilidades relacionadas con la creación de objetos de tipo phonetic, damerauLevenshtein, stringUtils, quickShort y hashMap, de esta forma se ve reflejado el uso del patrón creador.

Por otra parte el patrón experto se manifiesta en la clase CC_Phonetic, pues esta recibe de la clase CC_Suggestions a través del método phonetic.getMetaphone(getPalabra()) la información necesaria para cumplir su función. Por último se hace uso del patrón bajo acoplamiento que soporta el diseño de clases más independientes y reduce el impacto de los cambios, pues si se quisiera obtener la fonética de un término o calcular la distancia entre dos cadenas solamente habría que llamar a sus respectivas clases. De igual forma es fácil realizar algún cambio en la implementación de dichas clases.

En la confección del diseño se elabora el diagrama de paquetes y el diagrama de clases del diseño. El primero permite dividir el componente en partes manejables para su implementación y el segundo muestra la relación que existe entre las diferentes clases.

3.3.1 Diagrama de paquetes

Para elaborar el diseño se define una estructura de paquetes que permite dividir el sistema en fragmentos manejables para su implementación. Se emplea además el criterio de empaquetamiento por procesos y por clases, siguiendo la estructura que define el sistema. Los paquetes que hacen referencia a procesos están compuestos por subcarpetas que responden a las realizaciones de los casos de uso de los mismos.

Diagrama de paquetes

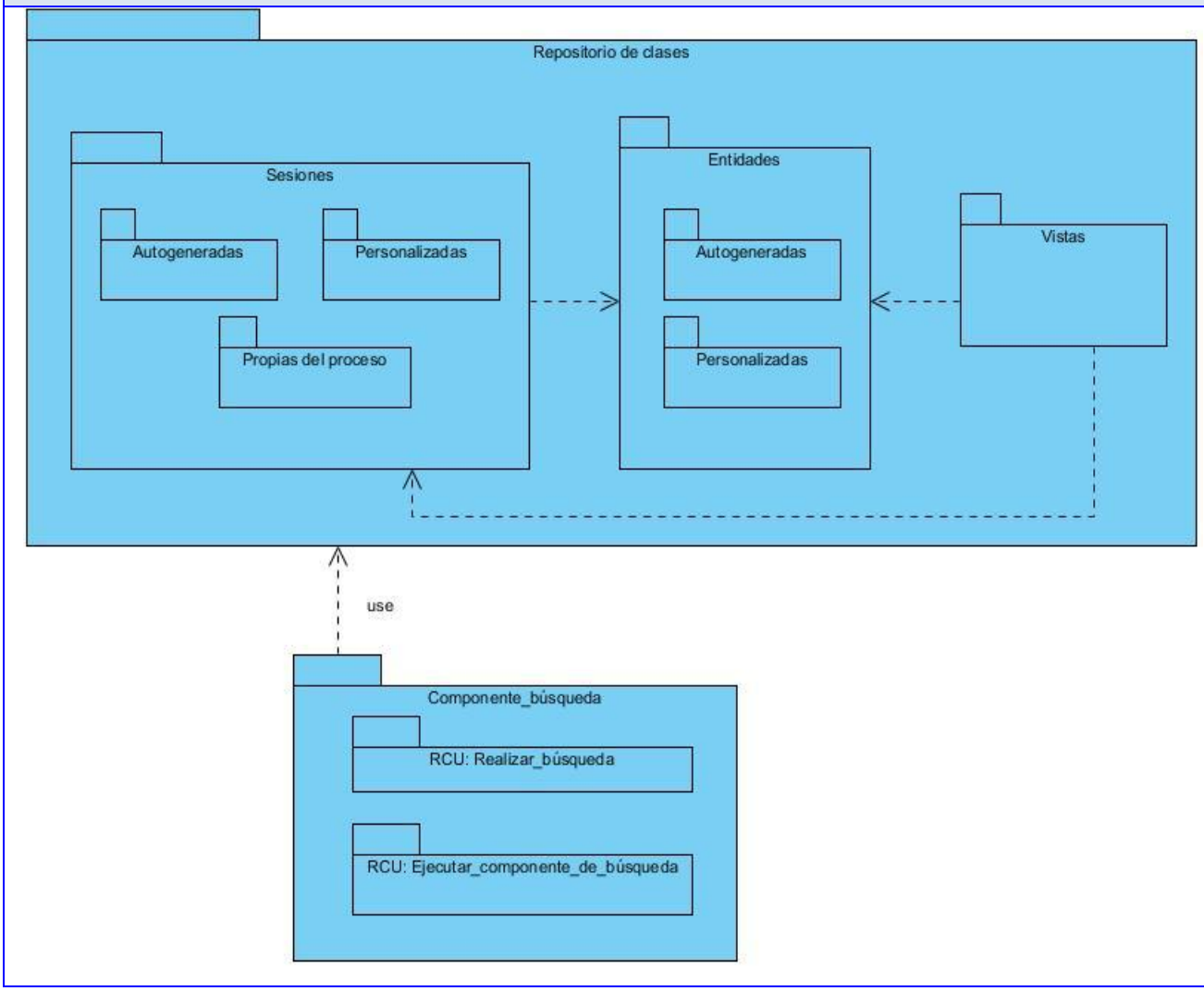

Figura 3: Diagrama de paquetes

Los casos de uso se implementan haciendo uso de la estructura de clases definida en el paquete Repositorio de clases. El mismo contiene tres subpaquetes: Vistas, Sesiones y Entidades. El paquete de las Vistas incluye los contenidos web referentes a las páginas clientes y los formularios que la componen. En el de Sesiones se agrupan las clases controladoras autogeneradas por el

entorno de desarrollo, las clases controladoras personalizadas y las clases controladoras propias del proceso. El paquete de las entidades contiene las entidades autogeneradas y las personalizadas. Las autogeneradas se obtienen mediante el mapeo de la base de datos y las personalizadas son las modificadas, o entidades que heredan de las autogeneradas que son cambiadas para lograr un mejor funcionamiento. A continuación se muestra el diagrama de paquetes correspondiente al componente de búsqueda.

3.3.2 Diagrama de clases del diseño

Los diagramas de clases del diseño están compuestos por páginas servidoras que construyen páginas clientes, estas últimas contienen formularios que capturan y muestran toda la información que será enviada a las páginas servidoras. Dichas páginas invocan métodos o responsabilidades en la clase controladora que según la acción solicitada puede modificar las entidades. A continuación de muestra el Diagrama de clases del diseño del componente de búsqueda.

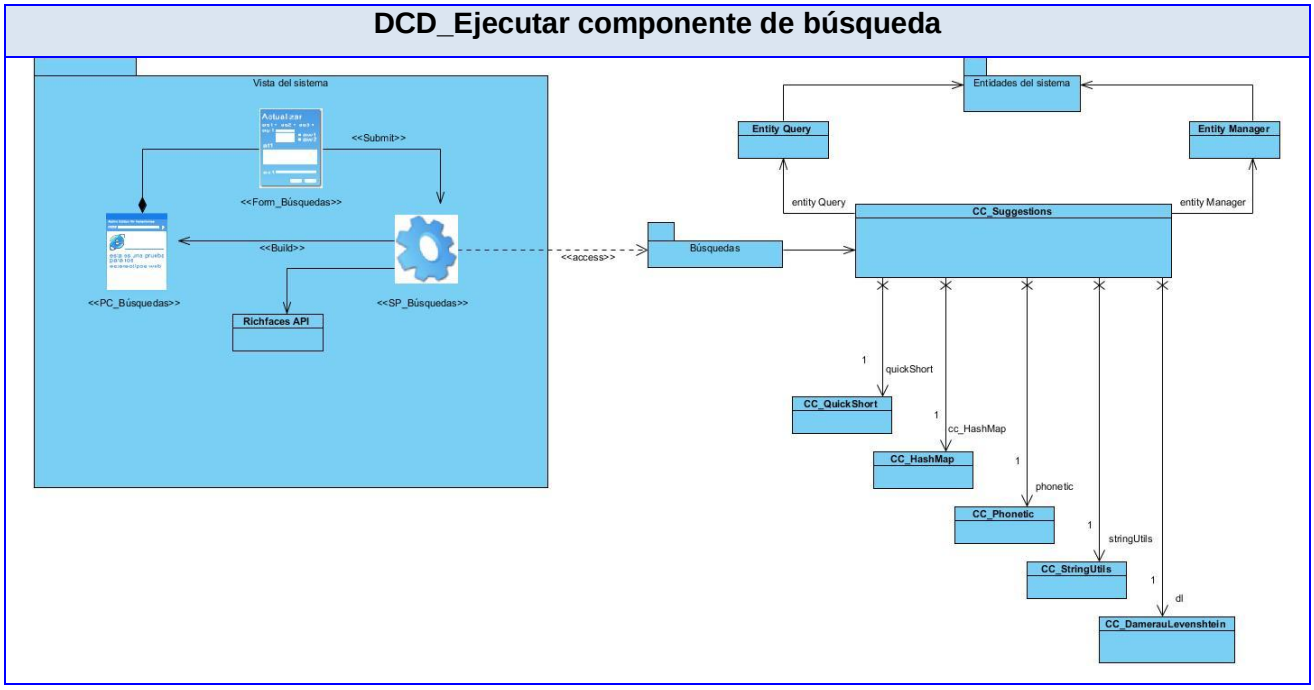

Figura 4: Diagrama de clases del diseño "Ejecutar componente de búsqueda"

3.4 Conclusiones del capítulo

- El diseño propuesto a partir del uso del patrón Modelo-Vista-Controlador facilita el mantenimiento y la reutilización del código.
- El empleo de los patrones de diseño: experto, creador, alta cohesión y bajo acoplamiento permite la asignación de responsabilidades a las clases definidas.

- La realización del diagrama de clases del diseño facilita la comprensión del funcionamiento del componente de búsqueda.

Capítulo 4. Implementación del componente de búsqueda

En este capítulo después de realizar el diseño de la solución propuesta, se describen las principales características de la implementación del componente. Se muestra a través del Modelo de datos la estructura donde se almacena toda la información requerida. Luego se muestra el Modelo de implementación, compuesto por el diagrama de componentes. Por último se expone como se realiza el tratamiento de errores.

4.1 Modelo de datos

La representación del Modelo de datos es una de las tareas que se desarrollan durante el flujo de trabajo correspondiente a la fase de implementación del componente. Este modelo es la traducción del análisis de requisitos al esquema conceptual mediante una representación gráfica de las entidades, atributos y sus relaciones. (23)

- Las entidades son objetos de los que los que el componente necesita guardar información. Por ser un concepto o abstracción se suele emplear el sustantivo en singular para nombrar la entidad y se representa a través de un rectángulo.
- Los atributos son cada una de las características asociadas a una entidad. Se representan colocando su nombre dentro del rectángulo de la entidad. Pueden ser clasificados en: obligatorios, opcionales, claves foráneas y claves primarias (estas se dividen en simples y compuestas). La representación gráfica de la clave primaria es un signo de suma y la de la foránea es un símbolo de número.

Las relaciones muestran la forma en que dos entidades se asocian. Se representan mediante una línea que une las dos entidades implicadas y tienen principalmente dos características: cardinalidad y obligatoriedad. (23)

El Modelo de datos del componente de búsqueda que a continuación se muestra es un modelo entidad relación (MER), las entidades que en él aparecen representan objetos de los que el componente necesita obtener información para comprobar su funcionamiento.

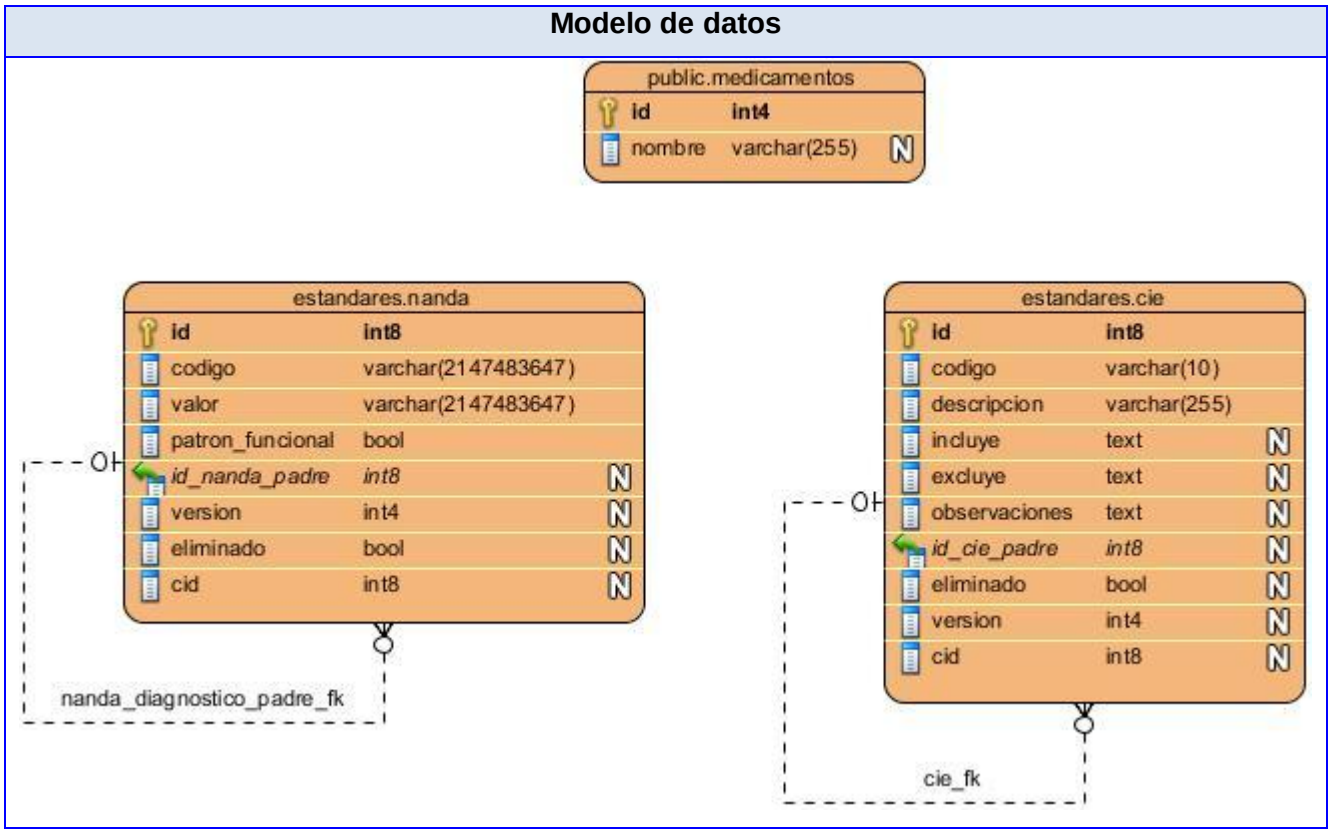

Figura 5: Modelo de datos

Para realizar los procesos de búsqueda el componente necesita los siguientes atributos:

- De la entidad public.medicamentos se utiliza el atributo **nombre** que contiene el principio activo[12] de los medicamentos.
- De la entidad estandares.nanda se utiliza el atributo **valor** que contiene el nombre de los diagnósticos de enfermería.
- De la entidad estandares.cie se utiliza el atributo **descripción** que contiene el nombre de las enfermedades.

4.2 Modelo de implementación

El modelo de implementación está compuesto por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes se encuentran: datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se

[12] Composición farmacológica del medicamento.

describen las relaciones que existen entre los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

4.2.1 Diagrama de componentes

El diagrama de componentes constituye otro de los artefactos de gran importancia para la arquitectura de un software ya que modela las relaciones entre los diferentes componentes (librerías, ejecutables, tablas, archivos, documentos y otros) que forman parte de un sistema de software. Sus interfaces pueden ser vistas como una firma del componente, el cliente no necesita saber sobre los funcionamientos internos del mismo (su implementación) para usarlo. Esto tiene como objetivo convertir el diseño de datos, interfaces y arquitectura en una representación intermedia que se pueda transformar fácilmente en código fuente. (23)

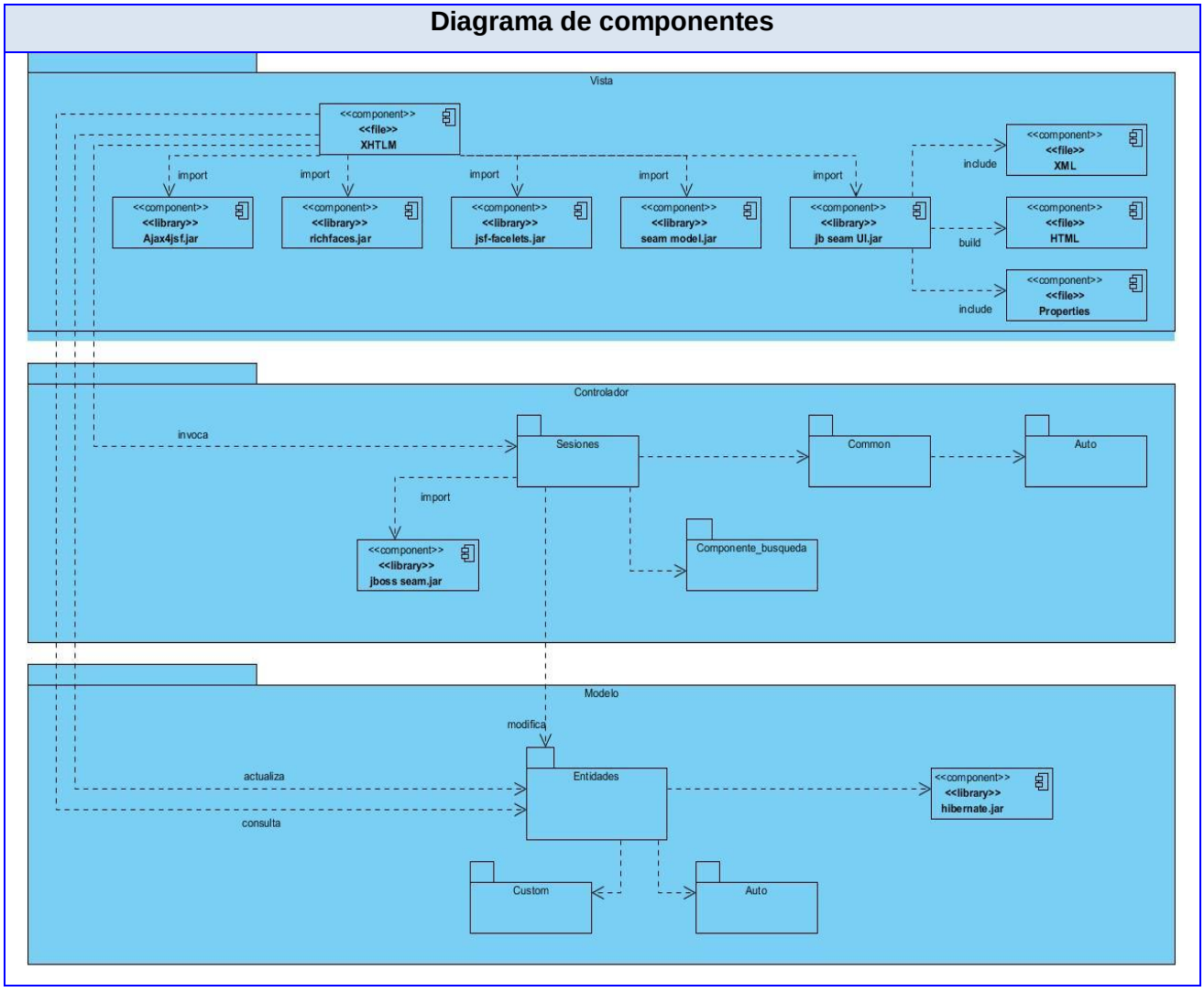

Figura 6: Diagrama de componentes

Como muestra la figura anterior la Vista está formada por las páginas XHTML que importan varias librerías para su construcción. El Controlador engloba el paquete de Sesiones que hace uso de otras librerías y contiene las controladoras autogeneradas, personalizadas y las propias del proceso, que son aquellas que permiten llevar a cabo los requisitos funcionales del sistema. En el Modelo están representadas las entidades autogeneradas y las personalizadas, contenidas todas en el paquete Entidades que utiliza la librería hibernate.jar. Estos paquetes mantienen una relación, las vistas consultan y actualizan las entidades e invocan a las controladoras y estas a su vez modifican las entidades.

4.3 Tratamiento de errores

El tratamiento de errores es uno de los aspectos más importante a tener en cuenta durante el desarrollo de cualquier sistema informático. En los entornos de programación se desarrolla una forma alternativa de manejar los errores, esta es conocida como manejo de excepciones. Las excepciones son situaciones anómalas que requieren un tratamiento especial.

En el componente de búsqueda el tratamiento de errores se realiza utilizando expresiones regulares que permiten validar los datos que recibe el componente, pues el funcionamiento de este depende de la información que recibe. Por ejemplo:

```
Pattern caracteresPermitido = Pattern
        .compile("^(([a-zA-Z]|[ñáéíóúü]|[0-9])|(([a-zA-Z]|[ñáéíóúü]|"
                + "[0-9])+\\s))+([a-zA-Z]|[ñáéíóúü]|[0-9])+$");
```

Figura 7: Ejemplo de expresión regular

Esta expresión regular valida que no se introduzcan caracteres extraños (@,*,<,&,^,%,$). Además se hace uso de bloques de código *try-catch*. El bloque *try* detecta la ocurrencia de algún fallo en el componente y el bloque *catch* maneja dichas excepciones mediante mensajes. Por ejemplo:

```
try {
    retornar = cc_Suggestions.suggestionsHibernate(objetoMapeado,
            getPalabraDelUsuario(), getCampos(),
            getUmbralDelProgramador(), isRespetarUmbralDelProgramdor(),
            getCantidadElementosAInsertar(), entityManager);
}



catch (CC_Excepciones e) {

    System.out.print(e.getMessage());
}
```

Figura 8: Ejemplo de bloque try catch

Este fragmento de código valida todos los elementos que recibe el componente para su ejecución.

4.4 Validación del componente

Para llevar a cabo la prueba de rendimiento se escoge como muestra varios términos. Se realizan 5 búsquedas con cada uno ellos para medir el tiempo de respuesta del sistema antes y después de su integración con el componente. También se comparan los resultados que muestran estos dos escenarios. La computadora donde se realizan las pruebas cuenta con las siguientes características.

- Microprocesador: Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz
- Memoria RAM: 3738 MB
- Sistema Operativo: Ubuntu 14.04
- Disco duro: 1 TB
- Mozilla Firefox en su versión 37 con una velocidad en la red de 100 Mb/s.

Además el componente utiliza un umbral de selección de 0.2 y el mismo es respetado.

Las siguientes tablas muestran los resultados obtenidos.

Tabla 6: Comparación de los resultados obtenidos

Término buscado	HIS		Caso	Respuesta esperada
	Con el componente	Sin el componente		
a	Acido folico	Acido folico	Una letra	Acido folico
	Atenolol	Atenolol		Atenolol
	Amlodipino	Amlodipino		Amlodipino
	Aluminio,hidroxido	Aluminio,hidroxido		Aluminio,hidroxido
	Magnesio, hidroxido	Magnesio, hidroxido		Magnesio, hidroxido
	Aminoacidos	Aminoacidos		Aminoacidos
salbutamol	Beclometasona	Beclometasona	Una palabra	Beclometasona
	Salbutamol	Salbutamol		Salbutamol
Saccharomyces boulardii	Saccharomyces boulardii	Saccharomyces boulardii	Dos palabras	Saccharomyces boulardii
1	Tiamina(vit B1)	Tiamina(vit B1)	Un número	Tiamina(vit B1)
55	No devuelve resultado, pues no existe en la base de datos ninguna cadena que sea igual o contenga el término introducido.	No devuelve resultado, pues no existe en la base de datos ninguna cadena que sea igual o contenga el término introducido.	Dos números	Ningún elementos
boulardii Saccharomyces	Saccharomyces boulardii	No devuelve resultado.	Transposición de palabras	Saccharomyces boulardii
Saccharomices	Saccharomyces boulardii	No devuelve resultado.	Sustitución de letra	Saccharomyces boulardii
Zecharomicse	No devuelve resultado.	No devuelve resultado.	Sustitución y transposición de letras	Saccharomyces boulardii
Saccharomycse	Saccharomyces boulardii	No devuelve resultado.	Transposición de letras	Saccharomyces boulardii
Leboflozasino	Levofloxacino	No devuelve resultado.	Sustitución de letra	Levofloxacino
klsio	Calcio, carbonato	No devuelve resultado.	Fónetica del término	Calcio, carbonato
Bnsilpnisilina	Bencilpenicilina	No devuleve resultado.	Fonética del término	Bencilpenicilina
	Benzatina bencilpenicilina			Benzatina bencilpenicilina
	Bencilpenicilina procaina			Bencilpenicilina procaina

Tabla 7: Comparación del rendimiento

Principio activo a buscar	Tiempo máximo		Tiempo promedio	
	Sin componente	Con componente	Sin componente	Con componente
a	2,43	3,08	2,26	2,98
salbutamol	2,13	2,76	2,03	2,6
Saccharomyces boulardii	2,49	2,92	2,26	2,84
1	2,42	3,61	2,25	3,49
55	1,82	2,72	1,64	2,63
boulardii Saccharomyces	3	3,27	2,89	3,17
Saccharomices	1,31	2,49	1,24	2,31
Zecharomicse	1,3	2,8	1,2	2,61
Saccharomycse	1,3	3,67	1,27	3,54
Leboflozasino	1,4	2,91	1,4	2,81
klsio	1,37	2,5	1,3	2,27
klsio krvonato	1,68	2,81	1,6	2,6
Bnsilpnisilina	1,78	2,25	1,7	2,4

Análisis de los resultados:

El sistema antes de la integración:

- Solo muestra resultados cuando el término a buscar constituye una subcadena de algún elemento del diccionario de datos.
- En el 33,3 % de los casos devuelve resultados.
- El tiempo de respuesta promedio para cuando se obtienen no excede los 2.26 segundos.

El sistema después de la integración:

- En el 83.3% de los casos devuelve resultado.
- El tiempo de respuesta promedio no excede los 3.54 segundos.
- Seleccionando un mayor valor para el umbral, el componente puede devolver más resultado.

Después de analizados los valores de las tablas 6 y 7 respectivamente, se obtienen las siguientes conclusiones:

El sistema sin hacer uso del componente de búsqueda implementado, devuelve los resultados 1.28 segundos más rápido que cuando se utiliza el componente, aunque es válido señalar que solo se obtienen resultados en 33.3% de los casos analizados. Sin embargo, el componente

integrado al HIS, muestra resultados en un 83.3% de los casos, pudiendo variar este por ciento en correspondencia con el umbral seleccionado.

4.5 Conclusiones del capítulo

- La disciplina de implementación permite obtener el Modelo de datos y el diagrama de componentes, garantizando la completa descripción de los datos, componentes y las dependencias entre estos.
- La utilización de expresiones regulares y bloques de código *try-catch* posibilita realizar el correcto tratamiento de errores del componente.
- Con la implementación de las funcionalidades definidas se obtuvo el componente de búsqueda para el Sistema de Información Hospitalaria del Centro de Informática Médica.
- La utilización del componente de búsqueda proporciona la respuesta deseada en un alto por ciento de las veces, aunque su tiempo de respuesta es superior al de los mecanismos de búsqueda que tiene implementado el HIS del CESIM.

Conclusiones

Con el desarrollo del componente de búsqueda para el Sistema de Información Hospitalaria del Centro de Informática Médica se concluye que:

- El proceso de búsqueda utilizando algoritmos especializados permite la comparación de términos de acuerdo a similitudes fonéticas y léxicas.
- La utilización de pautas en el proceso de desarrollo, garantiza la uniformidad y homogeneidad en los artefactos obtenidos.
- El componente de búsqueda para el Sistema de Información Hospitalaria del Centro de Informática Médica facilita al usuario final la obtención de los resultados esperados.

Recomendaciones

Con el objetivo de enriquecer la solución propuesta se recomienda:

- Incorporar al componente de búsqueda un diccionario de sinónimos de términos médicos que permita a los usuarios emplear un vocabulario más amplio y de fácil entendimiento.

Referencias Bibliográficas

1. **Lissete González Gallo, Filiberto López Palenzuela, Maikel David Ruenes Correa.** Convención internacional de salud pública. *Impacto de la plataforma tecnológica de salud para PDVSA.* La Habana : s.n., 3 de diciembre de 2012. 1637. 978-959-212-811-8.

2. **cie10.** cie10. [En línea] 2015. [Citado el: 12 de diciembre de 2014.] http://www.cie10.org.

3. **infomed.** infomed. [En línea] 29 de abril de 2011. [Citado el: 13 de diciembre de 2014.] http://www.sld.cu. 1537-1964.

4. **Abraham Silberschatz, Henry F. Korth, S. Sudarshan.** *Fundamentos de Bases de datos.* Madrid : s.n., 2002.

5. **ISO.** International Organization for Standardization. [En línea] 2015. [Citado el: 10 de diciembre de 2014.] http://www.iso.org/iso/home/standards.htm.

6. **Facultad de Ingeniería.** Facultad de Ingeniería. [En línea] 4 de mayo de 2015. [Citado el: 12 de enero de 2015.] http://ing.unne.edu.ar/pub/informatica/Alg_diag.pdf.

7. **Iván Amón, Francisco Moreno, Jaime Echeverria.** *Algoritmo fonético para la detencción de cadenas de texto duplicadas en el idioma español.* Medellín : s.n., 2012. 1692-3324.

8. *Funciones de similitud sobre cadenas de texto: Una comparación basada en la naturaleza de los datos.* **Iván Amón, Claudia Jiménez.** 2010. Association for information systems management.

9. **Gonzales-Cam, Celso.** *Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos.* Perú : s.n., 2008.

10. **Extensible Hypertext Markup Language.** Extensible Hypertext Markup Language. [En línea] 2008. [Citado el: 13 de enero de 2015.] http://xhtml.com/en/xhtml/reference.

11. **Java Server Faces.** Java Server Faces. [En línea] [Citado el: 10 de diciembre de 2014.] http://www.javaserverfaces.org.

12. **Marco de desarrollo de la junta de Andalucía.** Marco de desarrollo de la junta de Andalucía. [En línea] 2015. [Citado el: 20 de febrero de 2015.]

http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/capa-presentacion..

13. **RichFaces.** RichFaces. [En línea] 2015. [Citado el: 12 de diciembre de 2014.] http://richfaces.jboss.org/.

14. **Red Hat JBoss Middleware.** Red Hat JBoss Middleware. [En línea] 2012. [Citado el: 18 de enero de 2015.] http://www.redhat.com/es/technologies/jboss-middleware/developer-studio.

15. **Java The source for java technology collaboration.** Java The source for java technology collaboration. [En línea] 2015. [Citado el: 10 de enero de 2015.] https://www.java.net/.

16. **Ivar Jacobson, Grady Booch, James Rumbaugh.** *El proceso unificado de desarrollo de software.* Madrid : Pearson Educación, 2000. Vols. 84-7829-036-2.

17. **Orallo, Enrique Hernández.** *El lenguaje unificado de modelado(UML).*

18. **Martínez, Rafael.** Postgres-es. [En línea] 2 de octubre de 2010. [Citado el: 12 de enero de 2015.] http://www.postgresql.org.es/sobre_postgresql.

19. **Guía Ubuntu.** Guía Ubuntu. [En línea] 30 de mayo de 2014. [Citado el: 24 de febrero de 2015.] http://www.guia-ubuntu.com/index.php/PgAdmin_III.

20. **León, Eduardo.** *Tutorial Visual Paradigm for UML.*

21. **Pressman, Roger S.** *Ingeniería del Software: Un enfoque práctico.* 2005. 9701054733.

22. **Karl Wiegers, Joy Beatty.** *Software Requirements.* Washington : Christian Holdener, S4Carlisle Publishing Services, 2013. 978-0-7356-7966-5.

23. **Nicolás Kicillof, Carlos Reynoso.** *Introducción a los Patrones de Arquitectura. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft . .* Buenos Aires : s.n., 2004

Bibliografía

- **Abraham Silberschatz Henry F. Korth, S. Sudarshan** Fundamentos de Bases de datos [Libro]. - Madrid : [s.n.], 2002.
- **Carlos Reynoso Nicolás Kicillof** Introducción a los Patrones de Arquitectura [Sección del libro] // Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. - Buenos Aires : [s.n.], 2004.
- **cie10** cie10 [En línea]. - 2015. - 12 de diciembre de 2014. - http://www.cie10.org.
- **Dr. Denis Derivet Thaureaux Ing. Mirna Cabrera Hernández** Revista Cubana de Informática Médica [En línea]. - 16 de diciembre de 2014. - http://www.rcim.sld.cu/revista_14/articulos_htm/aplicacion.htm.
- **Enterría Josefa Gómez de** Los diccionarios especializados y la enseñanza de ELE [Libro]. - 2000.
- **Extensible Hypertext Markup Language** Extensible Hypertext Markup Language [En línea]. - 2008. - 13 de enero de 2015. - http://xhtml.com/en/xhtml/reference.
- **Facultad de Ingeniería** Facultad de Ingeniería [En línea]. - 4 de mayo de 2015. - 12 de enero de 2015. - http://ing.unne.edu.ar/pub/informatica/Alg_diag.pdf.
- **Gálvez Carmen** Identificación de nombres personales por medio de sistemas de codificación fonética [Libro]. - Granada : [s.n.], 2006.
- **García Juan Felipe** Métricas de Similitud para Búsqueda Aproximada.
- **Garrido Fernando Ripoll** Informática Médica [En línea]. - 2010. - 16 de enero de 2015. - http://www.informaticamedica.cl/2012/07/que-es-nanda.html..
- **Gonzales-Cam Celso** Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos [Informe]. - Perú : [s.n.], 2008.
- **Gracia Joaquin** Diseño de software orientado a objetos [Informe]. - 2005.
- **Guía Ubuntu** Guía Ubuntu [En línea]. - 30 de mayo de 2014. - 24 de febrero de 2015. - http://www.guia-ubuntu.com/index.php/PgAdmin_III.

- **Gutierrez Demián** UML Diagrama de paquetes [Informe]. - Venezuela : [s.n.], 2009.
- **infomed** infomed [En línea]. - 29 de abril de 2011. - 13 de diciembre de 2014. - http://www.sld.cu. - 1537-1964.
- Ingeniería del Software [En línea]. - 20 de noviembre de 2012. - 14 de enero de 2015. - https://arlethparedes.wordpress.com/.
- **ISO** International Organization for Standardization [En línea]. - 2015. - 10 de diciembre de 2014. - http://www.iso.org/iso/home/standards.htm.
- **Iván Amón, Claudia Jiménez** Funciones de similitud sobre cadenas de texto: Una comparación basada en la naturaleza de los datos [Conferencia] // Association for information systems management. - 2010.
- **Iván Amón Francisco Moreno, Jaime Echeverria** Algoritmo fonético para la detencción de cadenas de texto duplicadas en el idioma español [Informe]. - Medellín : [s.n.], 2012. - 1692-3324.
- **Ivar Jacobson Grady Booch, James Rumbaugh** El proceso unificado de desarrollo de software [Libro]. - Madrid : Pearson Educación, 2000. - Vols. 84-7829-036-2.
- **Java Server Faces** Java Server Faces [En línea]. - 10 de diciembre de 2014. - http://www.javaserverfaces.org.
- **Java The source for java technology collaboration** Java The source for java technology collaboration [En línea]. - 2015. - 10 de enero de 2015. - https://www.java.net/.
- **Karl Wiegers Joy Beatty** Software Requirements [Libro]. - Washington : Christian Holdener, S4Carlisle Publishing Services, 2013. - 978-0-7356-7966-5.
- **Larman Craig** UML y patrones : introducción al análisis y diseño orientado a objetos . [Libro]. - Madrid : Prentice-Hall, 2006.
- **León Eduardo** Tutorial Visual Paradigm for UML [Libro].
- **Lissete González Gallo Filiberto López Palenzuela, Maikel David Ruenes Correa** Convención internacional de salud pública // Impacto de la plataforma tecnológica de salud para PDVSA. - La Habana : [s.n.], 3 de diciembre de 2012. - 978-959-212-811-8.

- **Manuel Blázquez Ochando** Fundamentos y diseño de Bases de Datos [En línea]. - 20 de febrero de 2014. - 24 de enero de 2015. - http://ccdoc-basesdedatos.blogspot.com/2013/02/modelo-entidad-relacion-er.html.
- **Marco de desarrollo de la junta de Andalucía** Marco de desarrollo de la junta de Andalucía [En línea]. - 2015. - 20 de febrero de 2015. - http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/capa-presentacion..
- **Martínez Rafael** Postgres-es [En línea]. - 2 de octubre de 2010. - 12 de enero de 2015. - http://www.postgresql.org.es/sobre_postgresql.
- **Miguel Angel Alvarez Manu Gutierrez** Scribd [En línea]. - 2015. - 14 de enero de 2015. - http://es.scribd.com/doc/177026300/manual-programacion-javascript-parte1-pdf#scribd.
- **Nicolás Kicillof Carlos Reynoso** Introducción a los Patrones de Arquitectura. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft . [Libro]. - Buenos Aires : [s.n.], 2004.
- **Oracle** [En línea]. - 2015. - 16 de enero de 2015. - http://www.oracle.com/technetwork/java/javaee/documentation/index.htm.
- **Orallo Enrique Hernández** El lenguaje unificado de modelado(UML) [Libro].
- **Pressman Roger S.** Ingeniería del Software: Un enfoque práctico [Libro]. - 2005. - 9701054733.
- **Red Hat JBoss Middleware** Red Hat JBoss Middleware [En línea]. - 2012. - 18 de enero de 2015. - http://www.redhat.com/es/technologies/jboss-middleware/developer-studio.
- **RichFaces** RichFaces [En línea]. - 2015. - 12 de diciembre de 2014. - http://richfaces.jboss.org/.
- **Romaní Juan Cristóbal Cobo** Zer-Revista de Estudios de Comunicación [En línea]. - 10 de diciembre de 2014. - http://www.ehu.eus/ojs/index.php/Zer/index.
- **Sommerville Ian** Ingeniería de software [Libro]. - Madrid : PEARSON EDUCACIÓN, 2005.
- **Villaverde Alejandro Lucas** Evaluación automática e interactiva de destrezas mediante distancias entre cadenas [Informe]. - Valencia : [s.n.], 2012.

Anexos

Anexo 1: Clases del diseño.

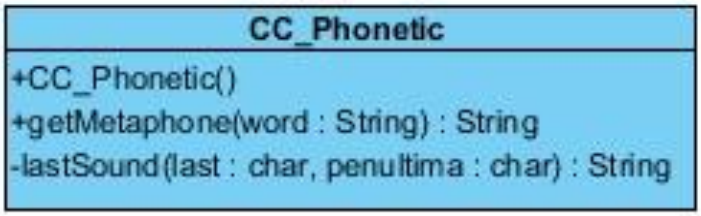

Figura 9: CC_Phonetic

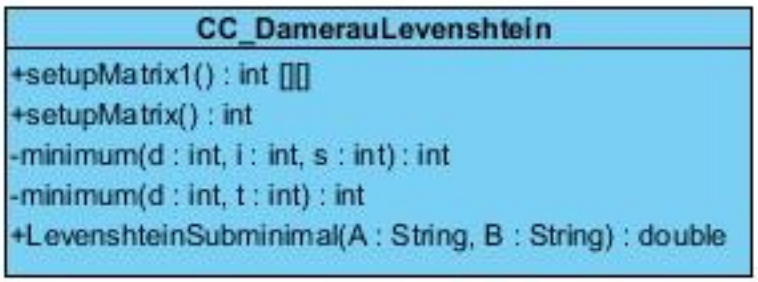

Figura 10: CC_DamerauLevenshtein

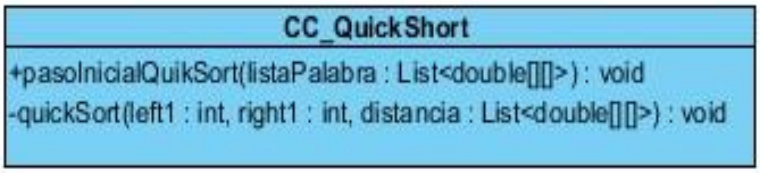

Figura 11: CC_QuickShort

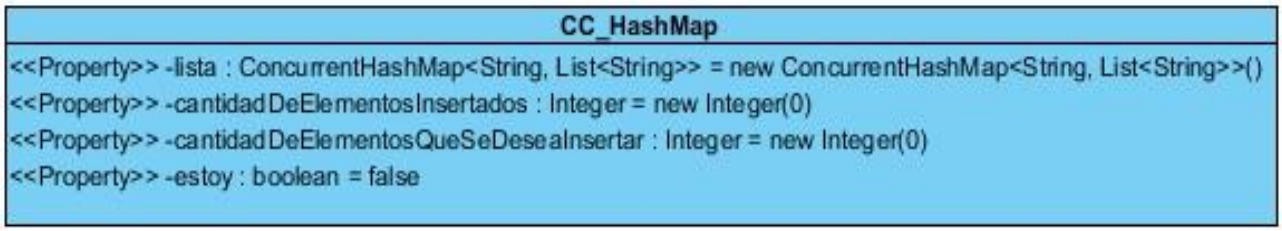

Figura 12: CC_HashMap

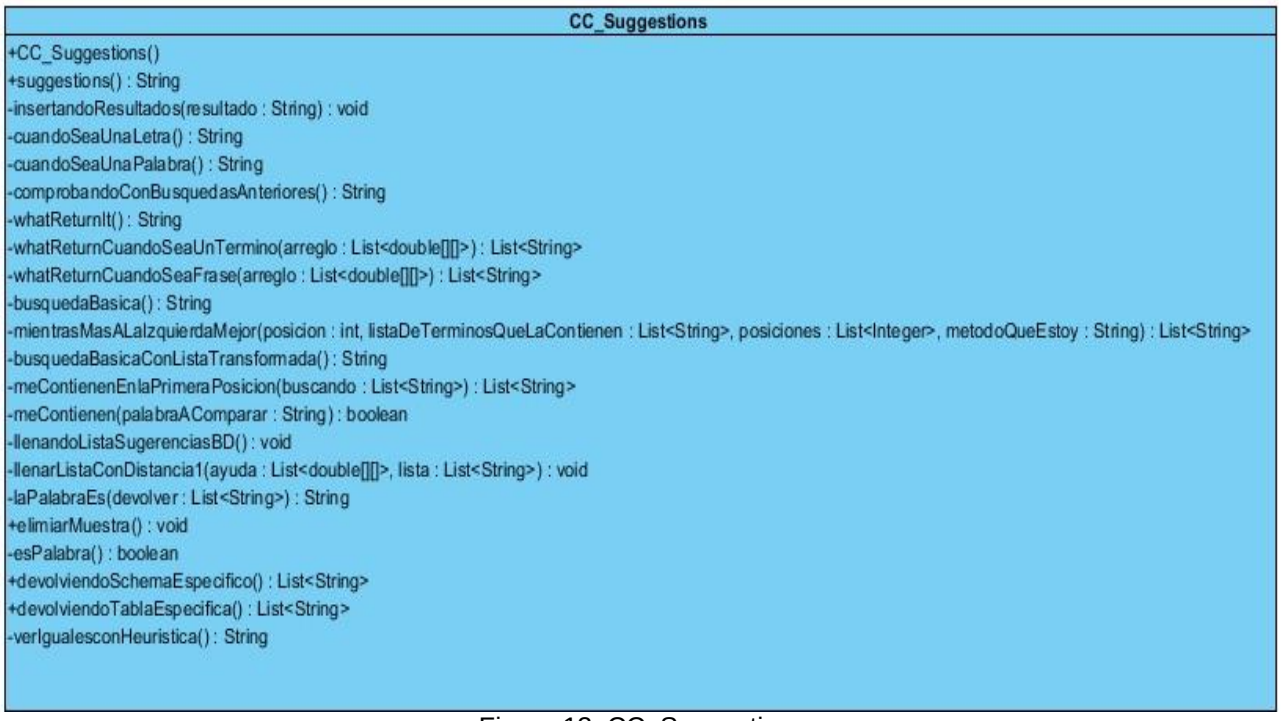

Figura 13: CC_Suggestions

Figura 14: CC_StringUtils

Anexo 2: Vistas de las búsquedas en el sistema.

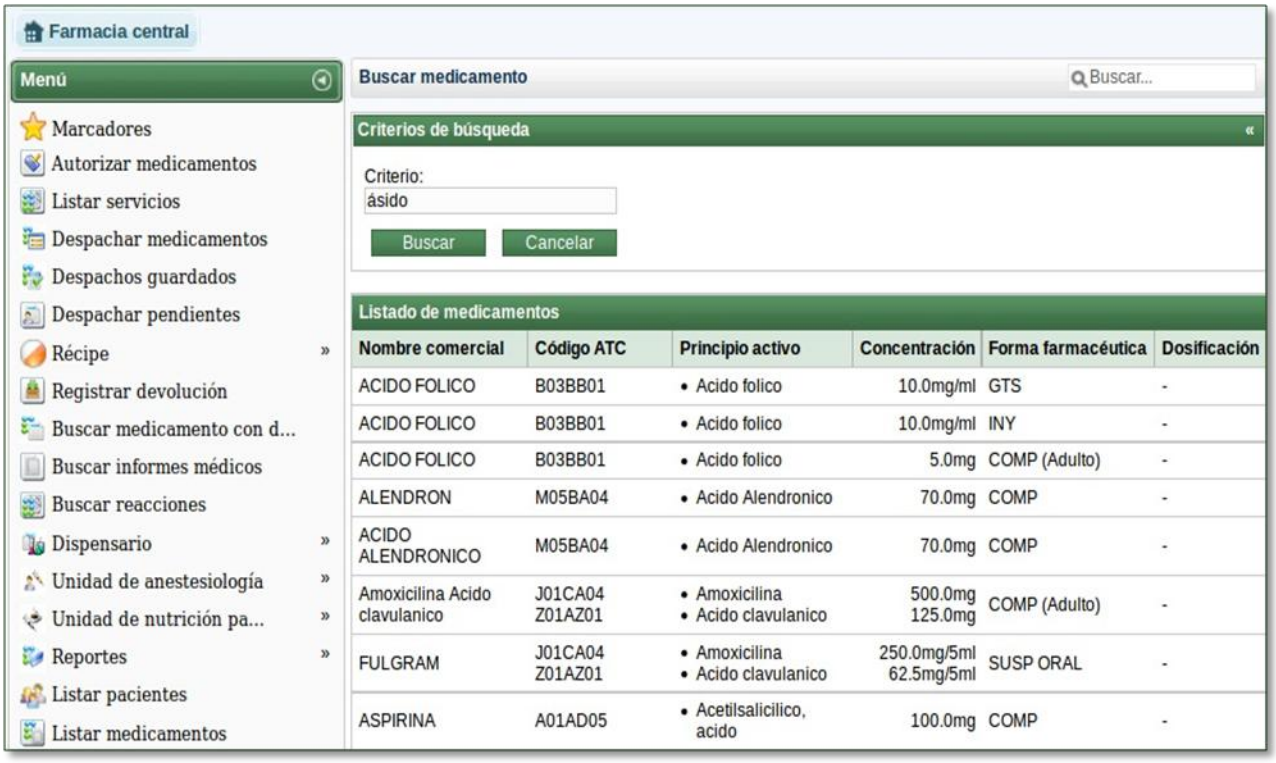

Figura 15: Búsqueda usando el componente

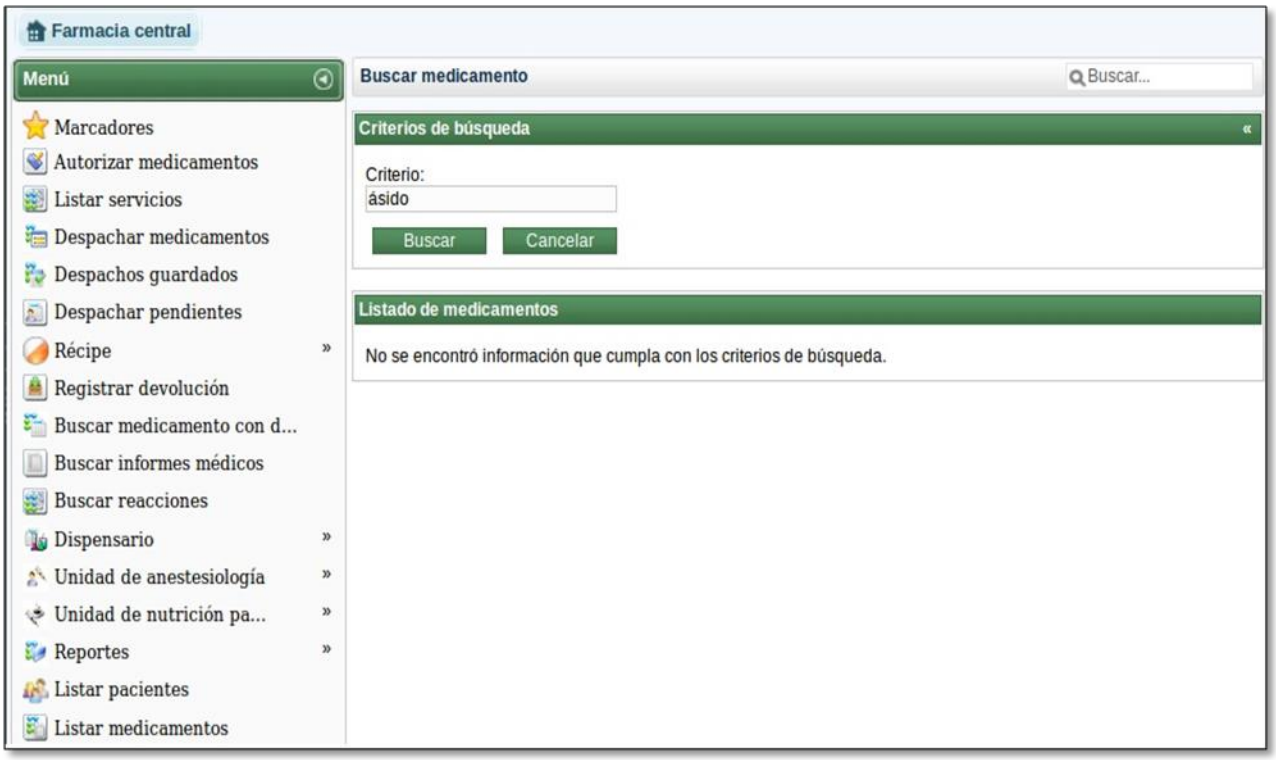

Figura 16: Búsqueda sin usar el componente