Universidad de las Ciencias Informáticas

Facultad 7

Título: Sistema de detección de personas duplicadas en un Sistema de Gestión de Información Médica

Trabajo de Diploma para optar por el título de Ingeniero Informático

Autor: Carlos Pedro Soberats Sánchez

Tutor: Ing. Alejandro Mario Velázquez

Cotutor: Ing. Gerardo Morgade Donato

La Habana, junio de 2011

"Año 53 de la Revolución"

Datos de Contacto

Ing. Alejandro Mario Velázquez Carralero

Ingeniero en Ciencias Informáticas, graduado en la UCI, en el curso 2006-07. Posee la Categoría Docente de Profesor Instructor. Actualmente imparte asignaturas de Práctica Profesional. Además, se desempeña como Arquitecto Principal de Área Temática Gestión Hospitalaria.

Correo electrónico: amvelazquez@uci.cu

Ing. Gerardo Morgade Donato

Graduado en el año 2009 de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas. Ha impartido las asignaturas Matemática III y Desarrollo de aplicaciones empresariales con Java y PostgreSQL. Actualmente se desempeña como Jefe de Módulo en el proyecto Sistema de Información Hospitalaria perteneciente al Departamento Gestión Hospitalaria del Centro de Informática Médica (CESIM).

Correo electrónico: gmorgade@uci.cu

AGRADECIMIENTOS

A mis padres, por ser ejemplo y guía en todas las actividades de mi vida, por ser la razón de mi esfuerzo y dedicación.

A mis hermanas, por su apoyo incondicional en todas las tareas de este proceso

A toda mi familia, por apoyarme en todo el transcurso de este proceso.

A mi esposa, por ser como es, por apoyarme en todo momento y darme ánimos para seguir adelante.

A mis tutores, por ser una fuente más de apoyo en el desarrollo de este trabajo.

A todos mis compañeros, por tener la difícil tarea de soportarme durante toda la carrera, para los presentes y para los que por una razón u otra no se encuentran entre nosotros.

DEDICATORIA

Dedico este trabajo de diploma a:

Mis padres y hermanas, por ser mi guía y ejemplo a seguir.

Mi esposa, por apoyarme en todo momento.

Gerardo Morgade Donato, por ser, más que un tutor, un amigo.

RESUMEN

En el mundo actual, la información ocupa un lugar de suma importancia, tanto por su valor como por los volúmenes que de ella se manejan en la sociedad moderna, por lo que resulta importante organizarla de forma eficiente con el fin de manipularla con mayor facilidad y extraer el máximo posible de ella. Los Sistemas de Gestión de Información Médica son aquellos sistemas encargados de gestionar todo este volumen de información, específicamente, la información referente al paciente. Uno de los principales datos manejados por la mayoría de estos sistemas son los datos personales. Con este trabajo se propone crear una herramienta capaz de detectar posibles personas duplicadas en un Sistema de Gestión de la Información Médica mediante la aplicación de algoritmos inteligentes.

MPI es una herramienta orientada a la detección de posibles personas duplicadas en un Sistema de Gestión de la Información Médica. La misma fue desarrollada siguiendo el flujo de procesos de la metodología de desarrollo RUP. Además, se implementó utilizando las librerías y clases que brinda la tecnología de desarrollo JAVA a través del IDE de desarrollo NetBeans.

La herramienta propuesta permitirá lograr una mayor centralización de los datos personales almacenados en los Sistemas de Gestión de Información Médica. Permitirá además, eliminar la duplicidad de las Historias Clínicas Electrónicas almacenadas pertenecientes a una misma persona. Además se logrará optimizar el proceso de búsqueda de personas en la base de datos y aumentar la calidad de los informes estadísticos generados.

PALABRAS CLAVES:

Sistema de Gestión de la Información Médica, algoritmos inteligentes.

TABLA DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el mundo actual, la información ocupa un lugar de suma importancia, si se tiene en cuenta su valor y los grandes volúmenes que se manejan en la sociedad moderna; resulta importante organizarla de forma eficiente con el fin de manipularla con mayor facilidad y extraer el máximo posible de ella.

La sociedad se encuentra en constante transformación y desarrollo por lo que es necesario almacenar una gran cantidad y variedad de datos en aplicaciones de gestión de información. El uso de dicha información afecta de manera directa e indirecta a todos los ámbitos de la sociedad, pues permite conocer la realidad, interactuar con el medio físico, apoyar la toma de decisiones y evaluar las acciones de individuos o grupos. La obtención y procesamiento de estos datos, referentes a las distintas áreas de la vida cotidiana, es realizada a través de los distintos medios y recursos tecnológicos con que cuenta la sociedad. Estas acciones se realizan de forma manual, lo cual disminuye la rapidez del análisis y procesamiento de datos.

En las instituciones hospitalarias se procesa un gran volumen de información referente al paciente, como son sus datos personales, resultados de exámenes entre otros. Existen sistemas encargados de procesar y almacenar todos estos datos, estos sistemas son conocidos como Sistemas de Gestión de Información, específicamente, Sistemas de Gestión de Información Médica. El autor asume el concepto planteado por la Universidad Veracruzana de que un Sistema de Información se puede definir como “Un conjunto de componentes interrelacionados que reúne procesa, almacena y distribuye información para apoyar la toma de decisiones y el control en una organización”. (1)

Uno de los componentes fundamentales de la mayoría de estos sistemas son los datos personales. Dichos datos, se refieren a toda aquella información relativa al individuo que lo identifica o lo hace identificable. Entre otras cosas, le dan identidad, lo describen, precisan su origen, edad, lugar de residencia, trayectoria académica, laboral o profesional. (2)

En la mayoría de los sistemas, los datos personales son obtenidos por el personal que labora con estos, lo que en ocasiones provoca la aparición de errores en la información y la duplicación de los datos personales referentes a una misma persona, poniéndose en riesgo la veracidad y calidad de esta información. Muchos de estos errores provocan inconsistencia en la generación de datos estadísticos, en la calidad de la información referente a un determinado grupo de personas plasmadas en informes, bases de datos y documentos de una alta importancia tanto para la persona como para la empresa encargada.

El proceso de búsqueda de posibles datos personales y personas duplicadas no es para nada trivial. La utilización de metodologías y procedimientos permite recorrer los diferentes sistemas de información médica, para identificar la existencia de errores en estos campos personales. La mayoría de los datos duplicados en estos sistemas están dados como consecuencias de errores en la recogida de la información. Los errores en la escritura de nombres, apellidos, carnet de identidad de personas, entre otros, los que principalmente son cometidos por el personal autorizado, propiciando la omisión de caracteres o la escritura errónea de estos.

Existen disímiles algoritmos para la búsqueda y comparación de personas duplicadas en la información referente a ellas. La mayoría de estos algoritmos son aplicados a la búsqueda de probabilidades de ocurrencia de datos, los cuales asignan prioridades, y delimitan los resultados de la búsqueda a grupos de personas en las que la información obtenida se considera significativa o común. Además, en ocasiones el personal encargado de interactuar con el sistema omite caracteres o introduce datos erróneos debido a la incongruencia en la escucha, es decir, en cuanto al entendimiento de la pronunciación de los datos.

En el año 2002 es creada en Cuba la Universidad de las Ciencias Informáticas (UCI), con el fin de lograr un avance en el estudio de esta rama tecnológica, además de potenciar el desarrollo de aplicaciones que, contribuyan a la informatización y desarrollo de la mayoría de los sectores del País.

La UCI cuenta con un gran número de centros de desarrollo de software, cada uno orientado a la informatización de un sector de la sociedad. El Centro de Informática Médica (CESIM) es un centro de desarrollo de software orientado a la informatización de las instituciones de la salud, principalmente, al desarrollo de aplicaciones de gestión de la información referente al paciente dentro de la institución de la salud. Entre las aplicaciones que más se destacan se encuentran: el Sistema de Información Hospitalaria alas HIS, el Sistema de Almacenamiento, Tratamiento y Comunicación de Imágenes Médicas alas PACS y el Sistema de Información Radiológica alas RIS. En su conjunto estos sistemas carecen de una herramienta capaz de detectar posibles datos duplicados en la información almacenada en sus bases de datos.

Partiendo de la necesidad de erradicar las dificultades existentes se pudo reconocer el siguiente **Problema a resolver**: ¿Cómo detectar datos personales duplicados en un Sistema de Gestión de Información Médica?

El **Objeto de estudio** está constituido por el proceso de limpieza de datos, enmarcado en el **Campo de acción:** el proceso de detección de personas duplicadas en un Sistema de Gestión de Información Médica.

Para solventar el problema antes planteado se propone el siguiente **Objetivo general**: Desarrollar una herramienta que detecte posibles datos personales duplicados en un Sistema de Gestión de Información Médica.

Para dar solución al problema y cumplimentar el objetivo anteriormente planteado se desarrollaron las siguientes **Tareas de investigación**:

1. Realizar un análisis de la lógica difusa y su aplicación en la comparación de datos personales.
2. Analizar los diferentes algoritmos inteligentes utilizados en la comparación de los datos personales mediante el uso de la lógica difusa.
3. Realizar un análisis de las principales herramientas y metodologías utilizadas para el desarrollo de sistemas de detección de personas duplicadas.
4. Aplicar la metodología seleccionada y generar los artefactos correspondientes a los Flujos de Trabajo: “Implementación” y “Pruebas”.
5. Implementar el sistema informático aplicando las pautas de diseño y siguiendo lo establecido en la Especificación de Requisitos de Software.
6. Implementar los diferentes algoritmos inteligentes a utilizar en la solución.
7. Realizar una comparación cualitativa y cuantitativa de los resultados obtenidos con cada uno de los algoritmos implementados.

La implementación de un sistema capaz de detectar posibles personas duplicadas en un Sistema de Gestión de Información Médica optimizaría el proceso de almacenamiento y procesamiento de los datos que en dichos sistemas se manejan, entre los principales beneficios que traería la solución propuesta se encuentran:

- Lograr una mayor centralización de los datos personales almacenados en los Sistemas de Gestión de Información Médica.
- Eliminar la duplicidad de las Historias Clínicas Electrónicas almacenadas pertenecientes a una misma persona.

- Optimizar el proceso de búsqueda de personas en la base de datos, así como, aumentar la calidad de los informes estadísticos generados.

El documento está estructurado de la siguiente forma:

Capítulo 1: Fundamentación Teórica: Estudio preliminar de los Sistemas de Gestión de Información Médica que puedan ser utilizados como sistemas detectores de personas duplicadas. Estudio de los algoritmos para el análisis de similitudes entre personas. Tecnologías y herramientas de desarrollo a utilizar.

Capítulo 2: Descripción de la Arquitectura del sistema: Descripción del modelo de dominio de la solución propuesta, así representaciones gráficas de las principales entidades del sistema. Además se realiza una breve descripción del patrón de diseño arquitectónico a utilizar.

Capítulo 3: Descripción y análisis de la solución propuesta: Se definen los diagramas de clases y el modelo de datos del sistema.

Capítulo 4: Validación de la solución propuesta: Se realiza el proceso de prueba y validación de la solución desarrollada.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En el presente capítulo se realiza una breve descripción de la problemática planteada. El autor asume como puntos esenciales a tratar los Sistemas de Gestión de Información Médica y los principales problemas asociados a ellos. Además se abordan brevemente las principales tecnologías y metodologías de desarrollo de software, para justificar la selección realizada y dar cumplimiento a la presente investigación.

1.1 PROBLEMÁTICA

Las acciones de obtención, análisis y procesamiento de la información se realizan de forma manual lo cual disminuye la rapidez del análisis y procesamiento de datos, por lo que se hace necesario el desarrollo de sistemas encargados de la gestión de dicha información. En la mayoría de estos sistemas los datos personales son obtenidos a través del personal encargado, lo que en ocasiones provoca la aparición de errores en la información y la duplicación de los datos personales referentes a una misma persona, poniéndose en riesgo la veracidad y calidad de esta información.

La mayoría de los datos personales duplicados en estos sistemas son consecuencia de errores en la recogida de los datos. Las omisiones o escrituras erróneas de caracteres en la escritura de nombres, apellidos, carnet de identidad de personas, entre otros, son principalmente cometidas por el personal autorizado, lo cual trae como consecuencia la posible duplicación de los datos almacenados pertenecientes a una misma persona, esta duplicación conlleva a su vez a la fragmentación de la información almacenada, generando informes estadísticos irreales. Como ejemplo de estos errores se pueden citar:

- El nombre **Carlo** omitiendo el carácter **s**
- El apellido **Rodrigues**, escribiendo erróneamente el último carácter.

Existen datos que toman una participación común en un determinado grupo de individuos. Al realizar una búsqueda de personas duplicadas en los sistemas, utilizando estas referencias, devolvería disimiles resultados, por lo que se deben utilizar otros métodos para la obtención de resultados, explorando otros campos de búsqueda, utilizando mayor cantidad de recursos para obtener un dato más específico, ejemplo:

- Apellido **García**, el cual, al realizar una búsqueda en el sistema, devolvería una inmensa cantidad de personas que presenten dicho apellido, no siendo, las mismas personas.

1.2 ALGORITMOS PRESENTES EN LA SOLUCIÓN

Existen disímiles algoritmos que se pueden emplear para la búsqueda y comparación de personas a través la información referente a ellas, además de la búsqueda de probabilidades de ocurrencia de datos. Estos asignan prioridades y delimitan los resultados de la búsqueda a grupos de personas en las que la información obtenida se considera significativa o común. Además, en ocasiones el personal encargado de interactuar con el sistema omite caracteres o introduce datos erróneos debido a la incongruencia en la escucha, es decir, en cuanto al entendimiento de la pronunciación de los datos.

Entre los tipos de algoritmos utilizados para realizar esta acción se encuentran los algoritmos fonéticos y los algoritmos comparativos. Los algoritmos fonéticos obedecen 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 (3).

La aplicación de algoritmos comparativos para la evaluación de los campos y la verificación de sus similitudes con información anteriormente incorporada en el sistema, propicia la asignación de prioridades y la obtención de la información realmente importante, es decir, de aquellos campos realmente significativos en la búsqueda de similitudes de personas. Estos campos son los utilizados para retornar un valor real o aproximado de las posibles personas duplicadas en el sistema.

1.2.1 ALGORITMOS FONÉTICOS

Los más usados en esta rama son: el **Soundex** desarrollado en 1918 por Robert Russell and Margaret Odell y patentado finalmente en 1922. Una variante de este algoritmo fue llamada American Soundex, y utilizada en 1930 para el análisis retrospectivo de los censos de Estados Unidos de 1890–1920. Además se encuentra el algoritmo **Double Metaphone**, desarrollado por Lawrence Phillips, y publicado en Computer Language, Vol. 7, No. 12 (December), 1990, como parte de una clase de algoritmo llamado como *phonetic matching o phonetic encoding*. Este algoritmo utiliza reglas de codificación mucho más extensas que sus predecesores, manteniendo componentes de caracteres no latinos, y retorna una codificación primaria y secundaria para diferentes pronunciaciones de una sola palabra en idioma Inglés (3).

1.2.2 ALGORITMOS COMPARATIVOS

1.2.2.1 ALGORITMO DE LEVENSHTEIN

Algoritmo desarrollado en 1965 por el científico ruso Vladimir Levenshtein. El algoritmo de Levenshtein o distancia de Levenshtein es un algoritmo tal que dadas dos cadenas de texto, devuelve un entero que da una idea de la distancia (o parecido) entre ellas. Este entero se calcula contando las transformaciones que es necesario hacer sobre una de estas cadenas para obtener la otra, algunas de estas transformaciones son la inserción, borrado y sustitución. A menor distancia, mayor será la correspondencia entre dos cadenas de texto comparados. Cuanto más corta es la distancia entre las dos cadenas, más parecidas son. Si la distancia es 0, las dos palabras son iguales. Este algoritmo es utilizado para corrección ortográfica, reconocimiento de voz, análisis de ADN y detección de plagio.

1.2.3 ALGORITMOS INTELIGENTES

La Inteligencia artificial es una disciplina empleada con el fin de lograr que un determinado programa se comporte de forma inteligente sin pretender tener en cuenta la "forma de razonamiento" empleada para lograr ese comportamiento. Para lograr este fin son utilizados los algoritmos inteligentes los cuales son procedimientos utilizados para implementar una situación determinada, en la que se pretende que determinados equipos adquieran una inteligencia similar a la del ser humano, es decir, que sean capaces de adaptarse y responder ante dicha situación.

1.2.4 REDES NEURONALES ARTIFICIALES

En 1950, el matemático inglés Alan Turing, en su artículo "Maquinaria Computacional e Inteligencia", introduce la idea de desarrollar la inteligencia artificial a partir de simular el funcionamiento del cerebro humano.

Las **redes neuronales artificiales** son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico. Corresponden a un modo de computación analógica no lineal, distribuida, organizada por capas, las cuales son un conjunto de neuronas cuyas entradas provienen de la misma fuente y cuyas salidas se dirigen al mismo destino. (4)

Existen diversos tipos de redes neuronales como son:

Redes neuronales supervisadas: pueden emplearse como clasificadores de patrones, estimadores de funciones multivariadas o memorias asociativas. (5)

Redes auto organizadas o no supervisadas: en su entrenamiento no presentan las salidas objetivo que se desean asociar a cada patrón de entrada, será la red la que proporcione cierto resultado. La principal aplicación es la realización de agrupamiento de patrones (clustering), visualización de datos y representación de densidades de probabilidad, es por tanto, la más utilizada en el campo de la documentación. (5)

Redes neuronales realimentadas: son más complicadas que las dos anteriores. En este caso la información se propaga tanto hacia delante como hacia atrás, comportándose como un sistema dinámico, de difícil análisis y en el que deberá garantizarse la estabilidad de su respuesta. (5)

Las redes neuronales artificiales pueden aplicarse para resolver problemas en diversas ramas como son:

- En finanzas: se utilizan para prever la evolución de los precios, valorar el riesgo de los créditos, identificar falsificaciones, interpretar firmas, entre otras.
- En Medicina: en analizadores de habla para la ayuda de audición de sordos profundos y en el diagnóstico y tratamiento a partir de síntomas y/o de datos analíticos.
- Monitorización en cirugía
- Usos militares: para clasificar las señales de radar, creación de armas inteligentes y optimización del uso de recursos escasos.
- En procesos de Manufacturación: en control de producción de líneas de proceso, filtrado de señales e inspección de calidad, entre otras.

1.2.5 ARBOLES DE DECISIÓN

Un **árbol de decisión** es un modelo de predicción utilizado en el ámbito de la inteligencia artificial. Son muy similares a los sistemas de predicción basados en reglas. Un árbol de decisión es un diagrama que representa en forma secuencial condiciones y acciones; muestra qué condiciones se consideran en primer lugar, en segundo lugar y así sucesivamente. (6)

Un árbol de decisión tiene unas entradas las cuales pueden ser un objeto o una situación descrita por medio de un conjunto de atributos y a partir de esto devuelve una respuesta la cual en últimas es una decisión que es tomada a partir de las entradas.

Los valores que pueden tomar las entradas y las salidas pueden ser valores discretos o continuos. Se utilizan más los valores discretos por simplicidad, cuando se utilizan valores discretos en las funciones de una aplicación se denomina clasificación y cuando se utilizan los continuos se denomina regresión.

1.2.6 INTELIGENCIA BASADA EN CASOS

El Razonamiento Basado en Casos (CBR) es una rama de la inteligencia artificial que se preocupa por el estudio de los mecanismos mentales necesarios para repetir lo que se ha hecho o vivido con anterioridad, ya sea por uno mismo, o ya sea por casos concretos recopilados en la bibliografía o en la sabiduría popular. Es un paradigma de solución de problemas que difiere de otros enfoques y técnicas en que es capaz de utilizar el conocimiento específico adquirido en situaciones previas y utilizarlo en la situación presente.

Un problema nuevo se resuelve buscando en la memoria un caso similar resuelto en el pasado. Además, incrementa su conocimiento almacenando el nuevo caso para ser usado en situaciones futuras. Esto permite que el mismo se mantenga actualizado en todo momento. Este método intenta llegar a la solución de nuevos problemas, de forma similar a como lo hacen los seres humanos. Es una tecnología de la inteligencia artificial que representa el conocimiento como una base de datos de casos y soluciones. (7)

1.2.7 CLUSTERING

El algoritmo de clustering es la clasificación no supervisada de patrones (observaciones, elementos de datos o vectores de características) en grupos (clúster). Los algoritmos de clustering permiten clasificar un conjunto de elementos de muestra en un determinado número de grupos basándose en las semejanzas y diferencias existentes entre los componentes de la muestra.

Un algoritmo de agrupamiento o clustering es un procedimiento de agrupación de una serie de vectores según criterios habitualmente de distancia; se tratará de disponer los vectores de entrada de forma que estén más cercanos aquellos que tengan características comunes. Un algoritmo de este tipo permite extraer representantes de un conjunto de datos, que pueden ser posteriormente usados para transmisión, para eliminación de ruido o con una fase posterior de calibración, para clasificación de vectores en diferentes conjuntos. (8)

Algunas de las aplicaciones de este algoritmo son:

- Marketing: Encontrando grupos de clientes con un comportamiento similar, lo cual proporciona una extensa base de datos con datos de los clientes, almacenando, sus propiedades y las compras anteriormente realizadas.
- Biología: En la clasificación de plantas y animales proporcionando sus características.
- Web: En la clasificación de documentos; agrupando datos de los sitios web para descubrir grupos similares de patrones de acceso a ellos.

Posteriormente a la aplicación de algoritmos y al análisis de la información, es obtenido el resultado de similitud, el cual proporciona información acerca de la probabilidad de que existan dos o más datos personales similares en el sistema y, de que estos pertenezcan a una misma persona. Este resultado es empleado por los sistemas de información para brindar probabilidades e informes acerca de los errores cometidos en estos sistemas, además permite corregir estas incoherencias y eliminar la información innecesaria y errónea.

1.3 SISTEMAS ASOCIADOS AL CAMPO DE ACCIÓN

Actualmente en el mundo existen un sin número de sistemas encargados de la gestión de la información y de la búsqueda de la posible duplicación de sistemas en ellos. A continuación se relacionan los sistemas más significativos en esta rama.

1.3.1 ORACLE HEALTHCARE MASTER PERSON INDEX

Es un sistema diseñado para proveer una vista sencilla de la referencia de la información de un paciente, clínico, ordenante u otra institución de salud dentro de dicha organización. Este sistema permite a la organización, garantizar la disponibilidad unificada y confiada de los datos desde varios sistemas que referencian a la institución con diferentes identificadores o nombres. (9)

Características

Crea e integra una vista consistente de los datos personales basada en aplicaciones actuales y sistemas. Provee una visión única basada en la Web de los datos de la persona. Mantiene un sistema de base de datos centralizado y permite la integración de datos en toda la empresa mientras que permite a los sistemas locales operar de forma independiente. Es una entidad de modelo de objetos flexible y configurable. Contiene interfaces programables para crear y configurar la aplicación. Los procesos de actualizaciones de datos de sistemas externos son en tiempo real e incorpora más de 10 años de experiencia en la solución al problema. Es un sistema desarrollado sobre la plataforma Oracle Business Intelligent.

1.3.2 SUN MASTER PERSON INDEX

Sistema desarrollado sobre la plataforma **Sun Master Index**. Proporciona una vista confiable y exhaustiva de la información del paciente mediante un único identificador. Con este sistema es posible crear un patrón sencillo de la información del paciente sincronizado con el sistema existente. Realiza una referencia de datos cruzada con datos almacenados en otros sistemas conectados y automatiza los registros a través de sistemas dispares facilitando y simplificando la búsqueda de datos entre los departamentos. (10)

Dicho sistema centraliza la información acerca del paciente que participa dentro de la organización, y mantiene una base de datos centralizada para múltiples sistemas, lo que permite al sistema mantener la integración de los datos a través de toda la empresa. La base de datos, la cual es accesible a través de dicha empresa, almacena copias de archivos locales del sistema, además de los principales archivos que contienen la información principal de cada paciente.

Sun Master Person Index provee una estructura de datos predefinida basada en estándares de requerimientos de datos para la salud que pueden ser usados como tal o pueden ser personalizados fácilmente de ser necesario. La estructura de datos y el procesamiento lógico es almacenada en una colección de ficheros de configuración XML, los cuales son predefinidos pero que pueden ser modificados. Estos ficheros son definidos dentro del contexto del proyecto de capas de Java y son modificados usando un editor XML suministrado por NetBeans.

1.4 TECNOLOGÍAS UTILIZADAS

1.4.1 JAVA ENTERPRISE EDITION 5 (JEE5)

Es la plataforma que provee Sun Microsystem 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. Con el lanzamiento de esta edición, la especificación dio un gran paso hacia la excelencia, pues incorporó nuevas tecnologías de punta que no estaban contempladas anteriormente y que indiscutiblemente le incorporan a la plataforma gran robustez y simplicidad a la hora de trabajar, suficientes razones para no dudar en establecerla como base para el desarrollo. (11)

1.4.2 MÁQUINA VIRTUAL DE JAVA (JVM)

Es la base sobre la que se ejecutan todas las aplicaciones de java, encargándose de interpretar todo el código java y convertirlo al lenguaje nativo del sistema operativo en uso, esta es la razón por la que la mayoría de las aplicaciones java son altamente portables.

1.5 METODOLOGÍAS DE DESARROLLO

1.5.1 RUP

El Proceso Racional Unificado o RUP (Rational Unified Process), es un proceso de desarrollo de software que apoyándose en el Lenguaje Unificado de Modelado UML, constituye una de las metodologías estándares más populares para el desarrollo de sistemas orientados a objetos. (12)

Sus principales características se centran en:

- Implementar las mejores prácticas en Ingeniería de Software.
- Forma disciplinada de asignar tareas y responsabilidades, quién hace qué, cuándo y cómo.
- Administración de requisitos.
- Uso de arquitectura basada en componentes.
- Control de cambios y modelado visual del software.

RUP propone un desarrollo iterativo e incremental, donde se divide el proceso de elaboración del software en ciclos y se obtiene un producto final al término de cada uno de estos. Esta metodología está guiada por los casos de uso, siendo un caso de uso aquello que describe un fragmento de las funcionalidades del sistema. También es centrada en la arquitectura dándole a los desarrolladores una mayor visibilidad del sistema. RUP utiliza como lenguaje de modelado UML (Unified Modeling Language).

Este lenguaje permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, como consecuencia de que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

1.6 HERRAMIENTAS DE DESARROLLO

1.6.1 VISUAL PARADIGM

Es una herramienta CASE profesional de licencia gratuita que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Se puede definir a las herramientas CASE como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. (13)

Este software de modelado UML proporciona una más rápida y mejor construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Soporta ingeniería inversa, generador de informes, editor de figuras, integración IDE con Eclipse, NetBeans y otros. Además, entre sus ventajas se incluyen el modelado colaborativo con CVS (Concurrent Versions System*)* y subversión, la generación de documentación y de código base para diferentes lenguajes de programación como Java y su exportación como HTML. Cabe destacar igualmente su robustez, usabilidad y portabilidad.

1.6.2 ENTORNO DE DESARROLLO (IDE)

El IDE NETBEANS es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. A pesar de estar escrito en Java, es compatible con otros lenguajes de programación como C/C++, Ruby y JavaScript. Usando la plataforma NETBEANS el NETBEANS IDE soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring. Esta desarrollado bajo dos licencias: The Common Development and Distribution of License (CDDL) y la General Public License (GNU). (14)

1.6.3 POSTGRESQL

PostgreSQL es un sistema gestor de bases de datos Objeto-Relacionales basado en POSTGRES Versión 4.2, desarrollado por la Universidad de California en su departamento Berkeley de Ciencias de la Computación. POSTGRES fue pionero en muchos conceptos que solo estuvieron disponibles en algunos sistemas gestores de bases de datos comerciales mucho después. (15)

PostgreSQL es un sistema de código abierto descendiente del código original de Berkeley. Soporta una gran parte del SQL estándar y ofrece muchas funcionalidades modernas como:

- Consultas complejas
- Llaves foráneas
- Disparadores
- Vistas
- Integridad transaccional
- Control de concurrencia multiversiones

Además, PostgreSQL puede ser extendido por el usuario de muchas formas, por ejemplo adicionando nuevos:

- Tipos de datos
- Funciones
- Operadores
- Funciones agregadas
- Métodos de indexado
- Lenguajes procedurales

Producto de su licencia BSD, PostgreSQL puede ser usado, modificado y distribuido por cualquiera libre de costo para cualquier propósito, sea privado, comercial o académico.

1.6.4 PGADMIN III

Es una popular plataforma de administración y desarrollo para el gestor de bases de datos PostgreSQL, pues permite a los usuarios realizar desde simples consultas SQL hasta desarrollar bases de datos complejas.

Esta herramienta fue diseñada para responder a las necesidades de todos los usuarios, cuenta con una

interfaz gráfica que soporta todas las características de PostgreSQL y facilita enormemente su administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados y soporte para el motor de replicación Slony-I. (16)

Posterior al análisis y realización de la fundamentación teórica de la solución propuesta se concluye que el sistema alas HIS carece de un sistema que permita conocer la posible duplicación de las personas almacenadas en su base de datos, propiciando errores en la generación de informes estadísticos y documentos de un alto grado de importancia. Al realizar un análisis de los sistemas existentes para la búsqueda de posibles personas duplicadas en los Sistemas de Gestión de Información Médica se aprecia que dichos sistemas son sistemas desarrollados por empresas privadas, los cuales operan sobre las plataformas desarrolladas por dichas empresas. Además, de presentar un alto costo de adquisición, lo cual dificulta su aplicación en el sistema alas HIS.

Por lo anteriormente explicado se concluye en la necesidad de la realización de un sistema que permita la búsqueda de posibles personas duplicadas en los Sistemas de gestión de información médica, un sistema flexible que se adecue a las características tecnológicas de cada empresa, lo que permite el correcto funcionamiento y corrección de datos erróneos en las bases de datos de dichos Sistemas de Gestión de Información Médica.

Para el desarrollo de un sistema con las características antes explicadas se propone la utilización de las siguientes herramientas y tecnologías:

- Utilización de la metodología de desarrollo RUP debido a que permite la mitigación de los riesgos en forma temprana y continua, con un progreso demostrable y frecuentes releases ejecutables, permite además tener claro el proceso de desarrollo a seguir.
- Java como tecnología de desarrollo de software.
- Java como lenguaje de programación.
- NetBeans IDE como entorno de desarrollo.

CAPÍTULO 2: DESCRIPCIÓN DE LA ARQUITECTURA.

En el presente capítulo se plasman los requerimientos no funcionales con que debe contar el sistema. Se realiza una breve descripción de los principales conceptos del dominio, además se especifican los patrones arquitectónicos a utilizar en el desarrollo de la aplicación y se realiza un análisis de los posibles componentes y posibles implementaciones a reutilizar.

2.1 MODELO DE DOMINIO

El Modelo de Dominio constituye una representación visual estática del entorno real objeto del proyecto, pues, este no representa la interacción en el tiempo de los objetos, sino que representa una visión "parada" de las clases y sus interacciones. Los objetos o conceptos incluidos en el modelo de dominio no describen clases u objetos del software; sino entidades o conceptos del mundo real que están asociadas al problema en cuestión. Dicho modelo podrá ser utilizado como una base de las abstracciones relevantes en el proceso de construcción del sistema. (12)

2.2 CONCEPTOS FUNDAMENTALES DEL DOMINIO

Los conceptos del dominio constituyen los atributos del sistema o solución propuesta que permiten tener una mayor comprensión de la misma así como del modelo de dominio. Estos conceptos son:

Paciente: Persona que padece física y corporalmente, y especialmente quien se halla bajo atención médica. (17)

Comparadores: Conjunto de algoritmos encargados de realizar la comparación entre dos pacientes.

Comparaciones: Acciones a aplicar a un conjunto de datos específicos de los pacientes.

Documentos Clínicos: Registro donde se almacenan los episodios clínicos de los pacientes.

2.3 DIAGRAMA DEL MODELO DE DOMINIO

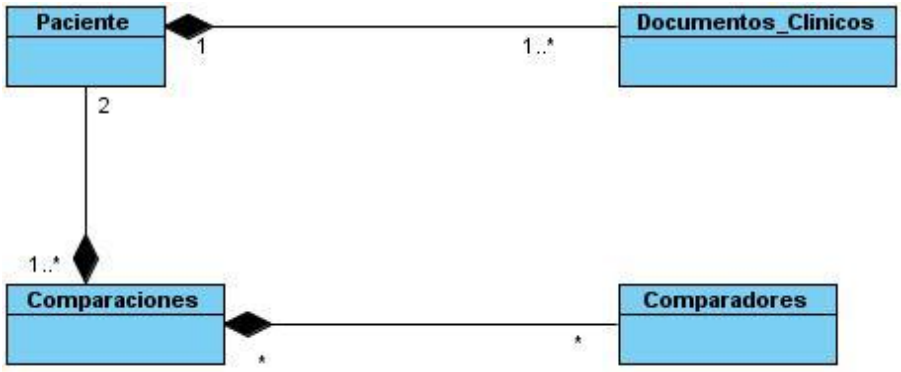


Imagen 1: Diagrama del Modelo de Dominio

2.4 ESPECIFICACIÓN DE LOS REQUISITOS DEL SOFTWARE

Un requisito de software podría definirse como una condición o capacidad que debe encontrarse o estar en un sistema o componente para satisfacer un contrato, norma, especificación u otro documento impuesto formalmente. El conjunto de todas las necesidades es el fundamento para el consiguiente desarrollo del sistema o componente. (12)

2.4.1 REQUISITOS NO FUNCIONALES DEL SISTEMA

Los requisitos no funcionales son cualidades o propiedades con las que debe de cumplir la solución. Son las características que logran que la solución sea más atractiva, usable, rápida y confiable.

Requisitos de rendimiento

RNF1- Permitir la realización de varias acciones por parte del sistema, garantizando que la rapidez de realización de dichas acciones sea máxima.

Requisitos de soporte

RNF2- El sistema debe contar con un mantenimiento y una revisión periódica, garantizando la disponibilidad de los servicios que brinda.

Requisitos de portabilidad

RNF3- El sistema podrá ser utilizado bajo cualquier arquitectura o sistema operativo donde exista instalada la máquina virtual de java (JVM).

Requerimientos de Software:

RNF4- Debe contar con Sistema Operativo Windows, Linux o UNIX. Además debe contar con la máquina virtual de java (JVM) 1.6 y como gestor de base de datos PostgreSQL 8.3.

Requerimientos de Hardware:

RNF5- Procesador Intel Quad Core o superior, 2 GB de memoria RAM, disco duro con capacidad de 40 GB.

2.5 DESCRIPCIÓN DE LA ARQUITECTURA

La arquitectura del software es el diseño de más alto nivel de la estructura de un sistema. Aporta una visión abstracta de alto nivel y pospone el detalle de cada uno de los módulos definidos a pasos posteriores del diseño. Constituye la organización fundamental de un sistema formada por sus componentes, las relaciones entre ellos y el contexto en el que se implantarán, y los principios que orientan su diseño y evolución. El objetivo principal es apoyar a la toma de desiciones, proporcionando además, conceptos, y un lenguaje común que permitan la comunicación entre los equipos que participen en un proyecto. (18)

Para el desarrrollo del sistema y con las herramientas, tecnologías y metodologías propuestas para el desarrollo del mismo, se define como parte de la línea base de la Arquitectura del sistema la implementación del patrón de diseño pipeline o tubería-filtros. Pipeline es considerada una arquitectura que define y enlaza a la vez una o más etapas de un proceso de negocio, ejecutándolas en secuencia para completar una tarea específica. (19)

Este patrón se encuadra dentro de las llamadas arquitecturas de flujo de datos debido a que espera que termine la ejecución de un componente para ejecutar el siguiente.Los filtros no realizan tareas de filtrado

como pueden ser la eliminación de campos o registros, sino que ejecutan formas variables de transformación, una de las cuales puede ser el filtrado.

Esta popular arquitectura conecta componentes computacionales a través de conectores, de modo que las computaciones se ejecutan a la manera de un flujo. Los datos son transportados a través de estos conectores(tuberías) entre los componentes(filtros), transformando gradualmente las entradas en salidas. Se considera que este estilo se debe usar cuando:

- Se puede especificar la secuencia de un número conocido de pasos.
- No se requiere esperar la respuesta asincrónica de cada paso.

Dentro de las principales ventajas de este patrón se encuentran:

- Es simple de entender e implementar. Es posible implementar procesos complejos coneditores gráficos de líneas de tuberías o con comandos de línea.
- Fuerza un procesamiento secuencial.
- Es fácil de envolver (wrap) en una transacción atómica.
- Los filtros se pueden empaquetar, y hacer paralelos o distribuidos. (20)

Enfocado a las características arquitectónicas de dicho patrón se define la siguiente estructura de componentes y conectores:

Imagen 2: Estructura del sistema

En el componente Algoritmos Comparadores se encuentran aquellos algoritmos encargados de realizar la comparación entre dos atributos en dependencia del criterio de comparación seleccionado, devolviendo un valor numérico entre 0 y 100. Este valor es recibido por el componente Analizador de atributos a través del conector, para realizar un análisis del atributo en general retornando la característica de la comparación. Esta característica puede ser Bueno, Malo, Regular o No Presente. El componente Analizador de datos personales recibe las características de todos los atributos y sigue la secuencia de ejecución del flujo de datos, y devuelve el grado de similitud existente entre los datos personales de los pacientes.

2.6 VISTA DE DESPLIEGUE

En el lenguaje UML existen diversas formas de vizualizar, especificar y documentar la arquitectura de un sistema, una de ellas es la vista de despliegue, la cual se encarga de mostrar las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

La arquitectura en tiempo de ejecución del Sistema de detección de posibles personas duplicadas se modeló utilizando 2 tipos de dispositivos de hardware: un servidor de aplicación y un servidor de base de datos. La comunicación entre los nodos se realiza mediante el protocolo TCP/IP para establecer la conexión entre ambos servidores.

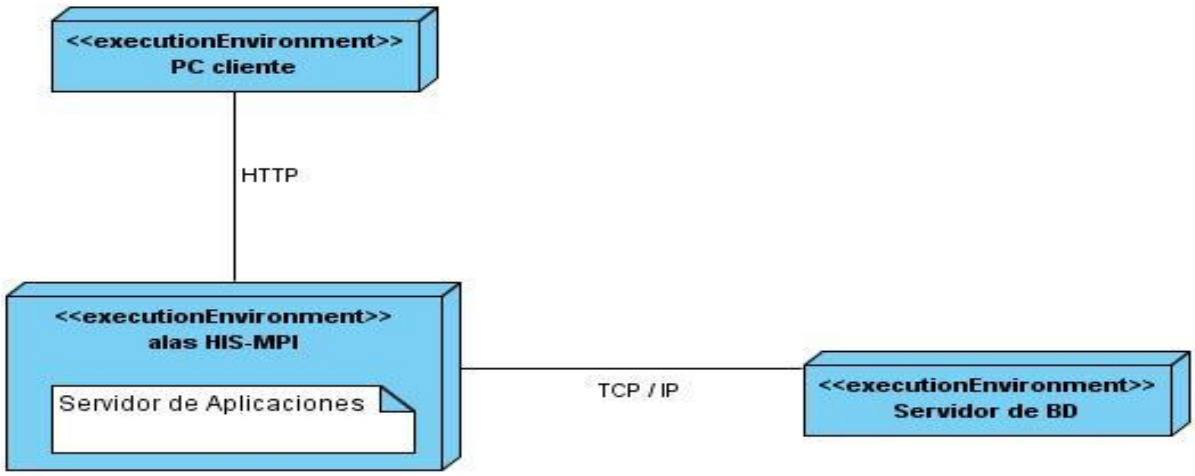


Imagen 3: Vista de Despliegue del Sistema

2.7 ESTRATEGIAS DE CODIFICACIÓN. ESTÁNDARES Y ESTILOS A UTILIZAR

Un estándar de codificación comprende todos los aspectos de la generación de código, de tal manera que sea prudente, práctico y entendible para todos los programadores. La utilización de las buenas prácticas de programación así como de una correcta codificación es de gran importancia para la calidad del producto, además de garantizar un buen rendimiento del mismo.

Para el desarrollo del sistema propuesto se utiliza un estándar de codificación acorde a la tecnología seleccionada garantizando la homogeneidad del estilo de programación durante la implementación del sistema. A continuación se describen parte de los elementos que conforman el estándar de codificación seleccionado:

Para lograr una correcta *indentación,* cuyo objetivo principal es el de lograr una estructura uniforme de los bloques de código, así como para los diferentes niveles de anidamiento se definieron los siguientes aspectos:

- Inicio y fin de bloque: se deben dejar dos espacios en blanco desde la instrucción anterior para el inicio y fin de bloque {}, así mismo para el caso de las instrucciones if, else, for, while, switch.
- Aspectos generales: evitar el uso del tabulador; pues este puede variar según la PC o la configuración de dicha tecla. Los inicios ({) y cierre (}) de ámbito deber estar alineados debajo de

la declaración a la que pertenecen y deben evitarse si hay sólo una instrucción. No se debe colocar ({) en la línea de un código cualquiera, pues esto requiere una línea propia.

Los comentarios, separadores, líneas, espacios en blanco y márgenes fueron otros de los elementos a considerar en el estándar, con estos se logra establecer un modo común para comentar el código de forma tal que se comprenda con sólo leerlo una vez.

- Ubicación de comentarios: estos se ubican al inicio de cada clase o función y al final de cada bloque de código, especificando el objetivo de la misma así como el tipo de dato y el objetivo de cada parámetro.
- Líneas en blanco: esta se emplean antes y después de métodos, clases y estructuras.
- Espacios en blanco: se recomienda entre operadores lógicos y aritméticos para lograr una mayor legibilidad en el código.

En cuanto al uso de variables y contantes se tuvo en cuenta los siguientes elementos:

- Apariencia de variables: el nombre de las variables debe comenzar con la primera letra en minúscula, en caso de que sea un nombre compuesto se empleará la notación Camell Casing, lo cual especifica que la palabra de inicio del identificador comienza con minúscula. Si el identificador está compuesto por más de una palabra entonces estás deben comenzar con mayúscula.
- Apariencia de constantes: se declaran con todas sus letras en mayúscula.
- Aspectos generales: los nombres de las variables y constantes deben tener nombres que con sólo leerlas se conozca el propósito.

Para el uso de clases y objetos se definieron un conjunto de elementos que garantizan nombrar las clases e instancias de la misma forma para toda la aplicación:

- Apariencia de clases y objetos: los nombres de las clases deben comenzar con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación Pascal Casing que especifica que la palabra de inicio del identificador comienza con

mayúscula. Si el identificador está compuesto por más de una palabra entonces estás deben comenzar con mayúscula seguido de Underscoard: Carácter ASCII representado como "_".

- Para el caso de las instancias se comenzará con un prefijo que identificará el tipo de dato, este se escribirá en minúscula.
- Apariencia de atributos: en caso de que sea un nombre compuesto se empleará notación Camell Casing.
- Apariencia de las funciones: el nombre de las funciones con verbos que denoten la acción que realiza y se empleará la notación Pascal Casing. Las funcionen que obtienen algún dato emplean el prefijo get y si fijan algún valor se emplea el prefijo set
- Declaración de parámetros en funciones: se declaran agrupados por tipos, defiendo primero los string y luego los numéricos, además se agrupan tenido en cuenta los valores por defecto.

En el presente capítulo se definieron los requerimientos no funcionales con que debe contar el sistema, se elaboró el diagrama del modelo de dominio basándose en los conceptos del dominio identificados. Se realizó además una descripción de la concepción arquitectónica del sistema, así como de la estrategia de codificación a seguir para el desarrollo del mismo. Se realizó el diagrama de despliegue, lo cual constituye una vista de la estructura física del sistema.

CAPÍTULO 3: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA.

En el presente capítulo se realiza el análisis y descripción de la solución propuesta, donde se definen los diagramas de clases y el modelo de datos del sistema. Se describen las principales clases y entidades asociadas a cada diagrama, confeccionándose además, el diagrama de componentes, el cual permite demostrar la dependencia entre los componentes del sistema.

3.1 MODELO DE DATOS DEL SISTEMA

Un modelo de datos realiza el estudio de los datos independientemente del procesamiento que los transforma. Constituye una visión abstracta de cómo son representados estos datos en la aplicación. Este modelo define además las estructuras permitidas y las restricciones a aplicar con el fin de representar los datos del dominio de aplicación. Hace uso del diagrama entidad relación con el fin de identificar los objetos de datos y sus relaciones y define los datos que se introducen, almacenan, transforman y se producen dentro de una aplicación. (12)

3.2 DESCRIPCIÓN DE LAS TABLAS PRESENTES EN EL SISTEMA.

Nombre: mpi.algoritmos		
Descripción: Tabla para registrar los datos de los algoritmos comparadores.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del algoritmo comparador
nombre	varchar	Nombre del algoritmo comparador
direccion	varchar	Define el tipo de algoritmo comparador: de atributos o de datos personales.
nivel	boolean	Define el nivel del algoritmo comparador
ultimo_entrenamiento	int	Valor del último entrenamiento realizado por el algoritmo.
id_jars	int	Identificador del jars de comparadores.

Tabla1: Descripción de la tabla algoritmos

Nombre: mpi.parametros		
Descripción: Tabla para registrar los parámetros de configuración de los algoritmos comparadores.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del parámetro.
nombre	varchar	Nombre del parámetro.
valor	int	Almacena el valor numérico del parámetro
id_algoritmo	int	Identificador del algoritmo al que pertenece el parámetro
descripcion	varchar	Descripción del parámetro.

Tabla 2: Descripcion de la tabla parámetros

Nombre: mpi.jars		
Descripción: Tabla para registrar los jars que contienen los algoritmos comparadores.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del jars
nombre	varchar	Nombre del jars
direccion	varchar	Dirección en la que se encuentra el jars
version	int	Define la versión del jars
eliminado	Boolean	Marca como eliminado o no el jars
cid	int	Identificador de la conversación del jars de comparadores.

Tabla 3: Descripción de la tabla jars

Nombre: mpi.comparadores		
Descripción: Tabla para registrar los datos de los comparadores.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del comparador
nombre	varchar	Nombre del comparador
id_jars	int	Identificador del jars de comparadores.
version	Int	Define la versión actual del comparador
eliminado	boolean	Marca como eliminado o no el comparador
cid	int	Identificador de la conversación del comparador

Tabla 4: Descripción de la tabla comparadores

Nombre: mpi.comparador_in_atributo		
Descripción: Tabla que relaciona el identificador de un comparador con el identificador de un atributo		
Atributo	**Tipo**	**Descripción**
id_comparador	int	Identificador del comparador
id_atributo	int	Identificador del atributo

Tabla 5: Descripción de la tabla comparador_in_atributo

Nombre: mpi.comparison_result		
Descripción: Tabla para registrar los resultados de las comparaciones realizadas.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del resultado de la comparación
id_hc_similares	int	Identificador de las historias clínicas similares
id_atributo	int	Identificador del atributo sobre el que se realizó la comparación
id_comparador	Int	Identificador del comparador que realizó la comparación
resultado	boolean	Almacena el resultado de la comparación realizada

Version	int	Versión de la comparación
eliminado	boolean	Marca como eliminado o no el resultado de la comparación
cid	int	Identificador de la conversación del resultado

Tabla 6: Descripción de la tabla comparison_result

Nombre: mpi.atributos_comparar		
Descripción: Tabla para registrar los datos de los atributos sobre los que se realizará la comparación		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del atributo
nombre	varchar	Nombre del atributo
version	Int	Define la versión actual del atributo
eliminado	boolean	Marca como eliminado o no el atributo
cid	int	Identificador de la conversación del atributo

Tabla 7: Descripción de la tabla atributos_comparar

Nombre: mpi.hc_similares		
Descripción: Tabla para registrar los resultados de las comparaciones de las historias clínicas		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del resultado de las comparaciones de las historias clínicas
id_nueva	int	Identificador de la nueva hoja frontal
id_viejo	int	Identificador de la hoja frontal almacenada en el sistema
es_igual	Boolean	Devuelve si las historias clínicas fueron marcadas como iguales o no.
porciento	int	Valor que representa el grado de similitud existente en dos hojas frontales.
version	Int	Define la versión actual del resultado de la comparación

eliminado	boolean	Marca como eliminado o no el resultado
cid	int	Identificador de la conversación del resultado de la comparación
fecha	date	Fecha en que se realizó la comparación y fue almacenado dicho resultado.

Tabla 8: Descripción de la tabla hc_similares

Nombre: mpi.hoja_frontal		
Descripción: Tabla para registrar los datos de la hoja frontal de un paciente		
Atributo	**Tipo**	**Descripción**
id	int	Identificador de la hoja frontal
numero_hc	varchar	Número de la hoja frontal
foto	Varchar	Foto referente al paciente
nombres	Varchar	Nombre del paciente
apellido1	Varchar	Primer apellido del paciente
cedula	Varchar	Cédula del paciente
fecha_nacimiento	Date	Fecha de nacimiento del paciente
apellido2	Varchar	Segundo apellido del paciente
version	Int	Versión actual de la hoja frontal
cid	Int	Identificador de la conversación de la hoja frontal
eliminado	boolean	Marca como eliminada o no la hoja frontal
id_sexo	Int	Identificador del sexo del paciente
id_nacion	Int	Identificador del país del paciente

Tabla 9: Descripción de la tabla hoja_frontal

Nombre: mpi.cambios_similares		
Descripción: Tabla para registrar los cambios realizados sobre las historias clínicas similares		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del cambio de similares
id_cambio	Int	Nombre de las historias clínicas similares
esquema	Varchar	Esquema donde se encuentra el cambio a realizar
tabla	Varchar	Tabla donde se encuentra el cambio a realizar
atributo	Varchar	Atributo sobre el cual se realizará el cambio
version	int	Versión actual del cambio
eliminado	Boolean	Marca como eliminado o no el cambio realizado
cid	Int	Identificador de la conversación del cambio
id_tabla	varchar	Tabla donde se encuentra el cambio realizado

Tabla 10: Descripción de la tabla cambios_similares

Nombre: mpi.fuzzy_comparison_result		
Descripción: Tabla para registrar los resultados de las comparaciones realizadas por los comparadores de atributos.		
Atributo	**Tipo**	**Descripción**
id	int	Identificador del resultado de la comparación
id_hc_similares	Int	Identificador de las historias clínicas similares
id_atributo	int	Identificador del atributo comparado
resultado	Int	Resultado de la comparación de atributo realizada

version	Int	Versión actual del resultado de la comparación realizada
eliminado	Boolean	Marca como eliminado o no el resultado de la comparación
cid	Int	Identificador de la conversación del resultado de la comparación
pertenencia	double	Grado de pertenencia que presenta el resultado de la comparación respecto al atributo seleccionado.

Tabla 11: Descripción de la tabla fuzzy_comparison_result

Nombre: mpi.config		
Descripción: Tabla para registrar la configuración de un comparador		
Atributo	**Tipo**	**Descripción**
key	int	Identificador de la configuración
data	byte	Datos de la configuración

Tabla 12: Descripción de la tabla config

Nombre: mpi.configuracion		
Descripción: Tabla para registrar los cambios realizados sobre los comparadores		
Atributo	**Tipo**	**Descripción**
id	Int	Identificador del cambio
Fecha	Date	Fecha en que se realizó el cambio
Umbral	Int	Umbral de los comparadores
Version	Int	Versión actual de los cambios realizados sobre los comparadores
Eliminado	Boolean	Marca como eliminado o no un cambio realizado sobre los comparadores
cid	Int	Identificador de la conversación de la configuración

Tabla 13: Descripción de la tabla configuración

Nombre: mpi.data_rules		
Descripción: Tabla para registrar las reglas de los datos del sistema		
Atributo	**Tipo**	**Descripción**
id	int	Identificador de las reglas
esquema	varchar	Esquema donde se encuentran las reglas
tabla	Varchar	Tabla donde se encuentran las reglas
atributo	varchar	Atributo sobre el que se aplicaron las reglas

Tabla 14: Descripción de la tabla data_rules

3.3 MODELO DE DATOS DEL SISTEMA

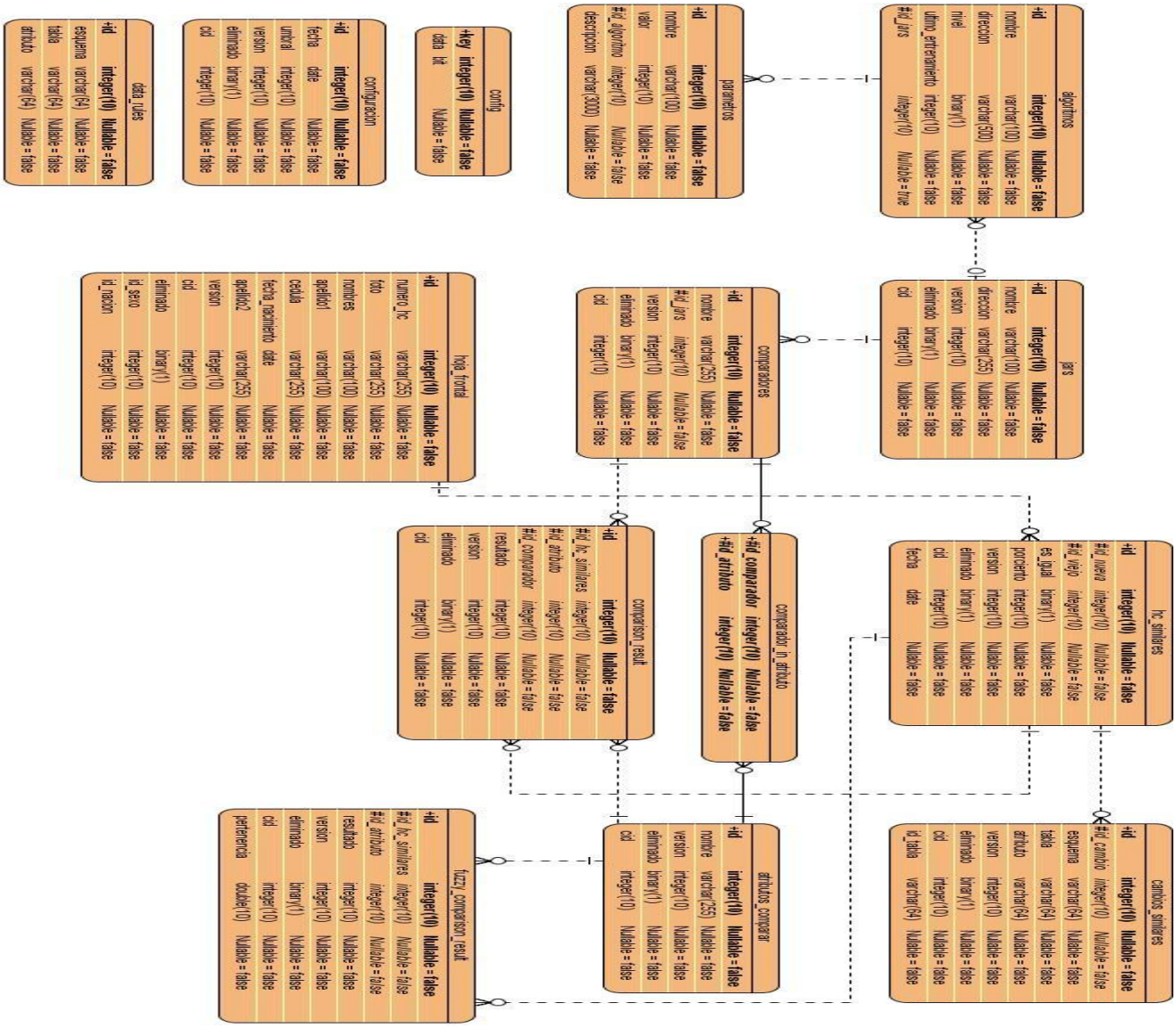


Imagen 4: Modelo de Datos del sistema de detección de personas duplicadas.

3.4 DIAGRAMA DE CLASES DEL DISEÑO

El diagrama de clases del diseño, es aquel diagrama compuesto por las clases refinadas del análisis. Dichas clases proporcionan detalles del diseño que permitirán la implementación de las clases de la solución propuesta. (**Imagen5: Diagrama de clases del diseño**). (12)

3.5 DESCRIPCIÓN DE LAS CLASES CONTROLADORAS MÁS SIGNIFICATIVAS

A continuación se describen las principales clases controladoras del sistema, estas son las encargadas de regir el funcionamiento de cada uno de los algoritmos comparadores.

3.5.1 CLASE INTELLIGENTALGORITM

Esta clase es la encargada de controlar el funcionamiento y ejecución de todos los algoritmos comparadores de atributos y comparadores generales de datos personales. De esta clase heredan las clases controladores de comparadores de atributos y de los comparadores generales de datos personales. **(Tabla 15: Descripción de la clase IntelligentAlgoritm)**.

3.5.2 CLASE FUZZYATTRIBUTECOMPARER

Esta clase es la encargada de controlar el funcionamiento de los algoritmos comparadores de atributos. Estos algoritmos retornan la característica del atributo como resultado de la comparación realizada. **(Tabla 16: Descripción de la clase FuzzyAttributeComparer).**

3.5.3 CLASE FUZZYHCCOMPARER

Encargada de controlar el funcionamiento de los comparadores generales de datos personales. Dichos comparadores retornan un valor numérico, significando el grado de similitud existente entre dos personas. **(Tabla 17: Descripción de la clase FuzzyHCComparer).**

3.6 DIAGRAMA DE CLASES DEL DISEÑO DEL SISTEMA

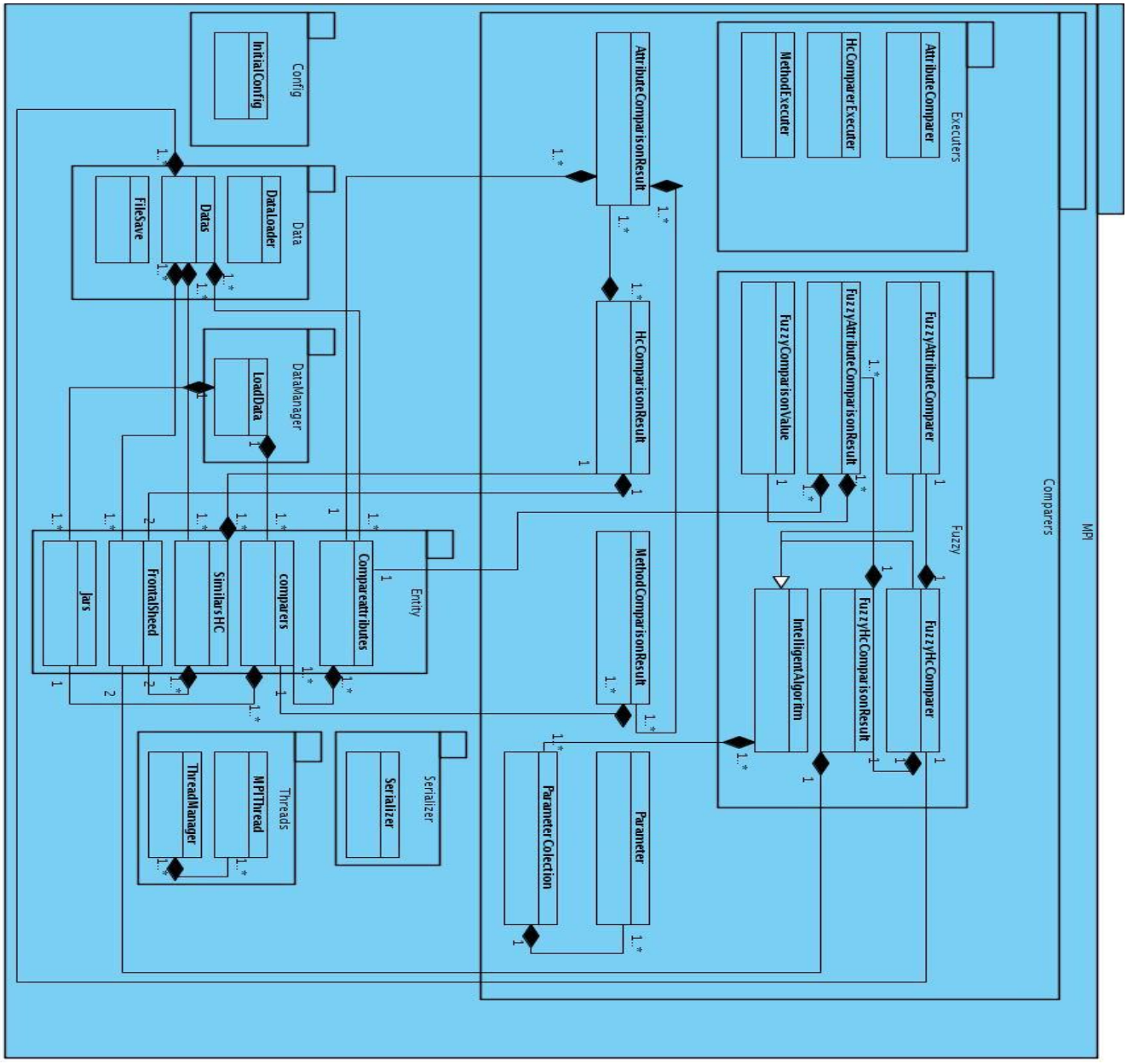


Imagen 5: Diagrama de Clases del Diseño del Sistema de detección de personas duplicadas

3.7 DESCRIPCIÓN DE LAS PRINCIPALES CLASES CONTROLADORAS PRESENTES EN LA SOLUCIÓN.

Nombre: IntelligentAlgoritm	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Id	int
lastTraining	int
parameters	ParameterCollection
Para cada responsabilidad:	
Nombre:	**IntelligentAlgoritm()**
Descripción:	Constructor de la clase en cuestión. Se encarga de cargar la configuración del algoritmo seleccionado.
Nombre:	**abstract void LoadIntelligency(Connection connection)**
Descripción:	Recibe la configuración de la conexión del sistema y carga los datos asociados a las comparaciones anteriores.
Nombre:	**abstract void RecalculateIntelligency(Connection connection)**
Descripción:	Recibe la configuración de la conexión y se encarga de re calcular los datos de la comparación realizada.
Nombre:	**abstractList<Parameter>getParametersName()**
Descripción:	Obtiene la lista de nombres de los parámetros de los comparadores.
Nombre:	**voidIntelligencyRecalculated(Connection connection)**
Descripción:	Recibe la configuración de la conexión y realiza la salva de los datos de la última comparación realizada.
Nombre:	**void save(Connection connection)**
Descripción:	Recibe la configuración de la conexión y realiza la salva de los datos de los comparadores después de realizada la comparación.
Nombre:	**List<Parameter>getParameterList()**
Descripción:	Retorna una lista con los parámetros de entrenamiento, cada parámetro estará compuesto por el nombre del parámetro, el id del algoritmo al que pertenece y la descripción de dicho parámetro.
Nombre:	**int getLastTraining()**

Descripción:	Retorna el valor del último entrenamiento realizado
Nombre:	**void setLastTraining(int lastTraining)**
Descripción:	Recibe y modifica el valor del último entrenamiento realizado.
Nombre:	**ParameterCollection getParameters()**
Descripción:	Retorna la colección de parámetros del algoritmo seleccionado.
Nombre:	**void setParameters(ParameterCollection parameters)**
Descripción:	Recibe y modifica la colección de parámetros del algoritmo seleccionado.
Nombre:	**String getName()**
Descripción:	Retorna el nombre del algoritmo seleccionado
Nombre:	**String getCode()**
Descripción:	Retorna el código perteneciente al algoritmo seleccionado
Nombre:	**boolean needTraining()**
Descripción:	Verifica si el algoritmo necesita ser entrenado nuevamente. Se basa en el valor de la diferencia entre el último entrenamiento salvado y el valor del entrenamiento actual. Si este valor es menor que el valor del parámetro "intervalo de entrenamiento" entonces realiza el entrenamiento.
Nombre:	**int getId()**
Descripción:	Retorna el valor del identificador del algoritmo seleccionado.
Nombre:	**void setId(int id)**
Descripción:	Recibe y modifica el valor del identificador del algoritmo.

Tabla 15: Descripción de la clase IntelligentAlgoritm

Nombre: FuzzyAttributeComparer	
Tipo de clase: Controladora.	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	**FuzzyAttributeComparisonResultattributeAnalyzis(AttributeComparisonResult comparison)**
Descripción:	Recibe el resultado de la comparación del algoritmo y realiza el análisis del atributo, implementando la operación attributeAnalysisCode y retornando la característica presente en dicho atributo.

Nombre:	**Abstract FuzzyAttributeComparisonResultattributeAnalyzisCode(AttributeComparisonResult comparison)**
Descripción:	Recibe el resultado de la comparación realizada sobre un atributo y realiza el análisis de la característica presente en dicho atributo.
Nombre:	**List<AttributeComparisonResult>getPrevious(Connection connection, CompareAttribute attribute)**
Descripción:	Recibe la configuración de la conexión y el atributo a comparar. Retorna una lista con todos los resultados de las comparaciones anteriormente realizadas sobre dicho atributo.

Tabla 16: Descripción de la clase FuzzyAttributeComparer

Nombre: FuzzyHCComparer	
Tipo de clase: Controladora.	
Atributo	**Tipo**
attributeComparer	FuzzyAttributeComparer
fuzzyComparison	FuzzyHcComparisonResult
Para cada responsabilidad:	
Nombre:	**FuzzyHcComparer()**
Descripción:	Constructor de la clase en cuestión.
Nombre:	**Integer Similitud(HcComparisonResult comparison)**
Descripción:	Recibe el resultado de la comparación de los datos personales y retorna un valor numérico entero, este valor es el grado de similitud existente entre dos personas.
Nombre:	**abstract Integer Similarity(FuzzyHcComparisonResult comparison)**
Descripción:	Recibe el resultado de la comparación de los datos personales y retorna la similitud existente entre estos datos.
Nombre:	**abstract Boolean show(Integer similitud)**
Descripción:	Recibe el valor de la similitud existente entre dos personas y en dependencia de dicho valor decide si mostrar este resultado o no.
Nombre:	**void saveData(int id)**
Descripción:	Recibe el id del atributo y almacena el resultado de la comparación realizada.

Nombre:	**FuzzyAttributeComparer getAttributeComparer()**
Descripción:	Retorna el comparador de atributo seleccionado.
Nombre:	**voidsetAttributeComparer(FuzzyAttributeComparerattributeComparer)**
Descripción:	Recibe y modifica el comparador de un atributo.
Nombre:	**List<FuzzyHcComparisonResult>getPrevious()**
Descripción:	Retorna una lista con todos los resultados de las comparaciones de datos personales anteriormente realizadas.

Tabla 17: Descripción de la clase FuzzyHCComparer

3.8 DESCRIPCIÓN DE LOS PRINCIPALES ALGORITMOS COMPARADORES IMPLEMENTADOS

Los algoritmos comparadores presentes en el sistema exhiben un parámetro de configuración común, dicho parámetro viene dado por el intervalo de entrenamiento, el cual es el número de comparaciones sobre la cual se debe realizar un nuevo entrenamiento del algoritmo.

3.8.1 ANALIZADORES DE IMPORTANCIA DE ATRIBUTOS

3.8.1.1 ÁRBOL DE DECISIÓN C4, 5

El algoritmo Árbol de decisión C4, 5 se encarga de conformar un árbol donde sus nodos están constituidos por las variables predictivas, y cada una de las ramas generadas son el conjunto de rangos de valores que puede tomar dicha variable. Estos rangos están divididos en 4 posibles grupos: 0-25, 26-50, 51-75 y 76-100. Las hojas del árbol serán una respuesta conformada por la característica presente en la variable objetivo, siempre y cuando se cumplan los valores necesarios de las variables para conformar el camino desde la raíz hasta dicha hoja. La generación de dichos árboles busca como objetivo general minimizar la entropía de las matrices asociadas a cada nodo en dependencia de la base de entrenamiento. Esta entropía está dada por la cantidad de información que se espera observar cuando ocurra el evento según una distribución de probabilidades.

La principal diferencia que presenta este algoritmo en comparación con el algoritmo ID3 es, que los nodos pueden contener un conjunto de valores pertenecientes a un mismo atributo, mientras que, los nodos del ID3 solo pueden contener un posible valor del atributo seleccionado.

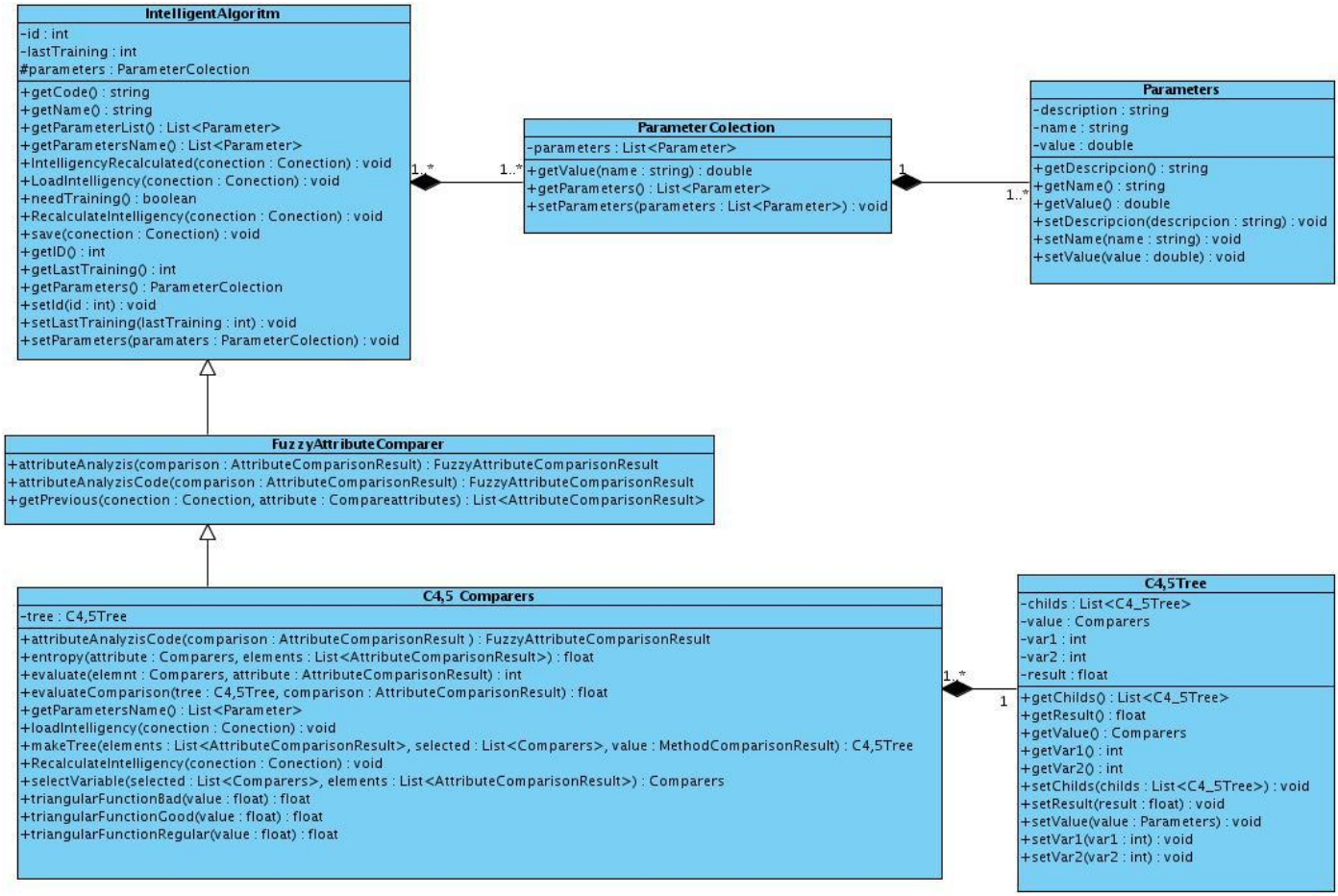


Imagen 6: Diagrama de clases del comparador de atributos mediante el Árbol de decisión C 4,5.

3.8.1.2 CLUSTERING VARIANTE DEL ALGORITMO CLUSTERING K-MEANS

Este algoritmo parte de conocer un valor numérico entero, el cual indica cual será la distancia máxima de separación que existirá entre cada uno de los puntos que conformen el clúster. Dicho clúster estará conformado además, por el valor de la comparación realizada sobre el atributo. El funcionamiento de este algoritmo se basa en seleccionar un atributo como un punto y verificar a que clúster se acerca, tomando como referencia, las coordenadas en el espacio de dicho punto. Posteriormente es obtenido el resultado de la importancia media del clúster al cual se asemeje en mayor medida. Este resultado es calculado de la siguiente forma:

- Si más del 70% de los miembros del clúster son parte de una comparación en la que los datos fueron iguales se le asigna el valor de “Bien”.
- Si entre un 40% y un 70% de los miembros del clúster son parte de una comparación en la que los datos fueron iguales se le asigna el valor de “Regular”.
- De lo contrario se asigna un valor de “Mal”.

Este algoritmo utiliza los siguientes parámetros de configuración:

Umbral: Distancia máxima que puede existir entre un punto y el punto centro del clúster.
Iteraciones: Cantidad máxima de iteraciones que debe realizar el algoritmo a la hora de conformar el clúster

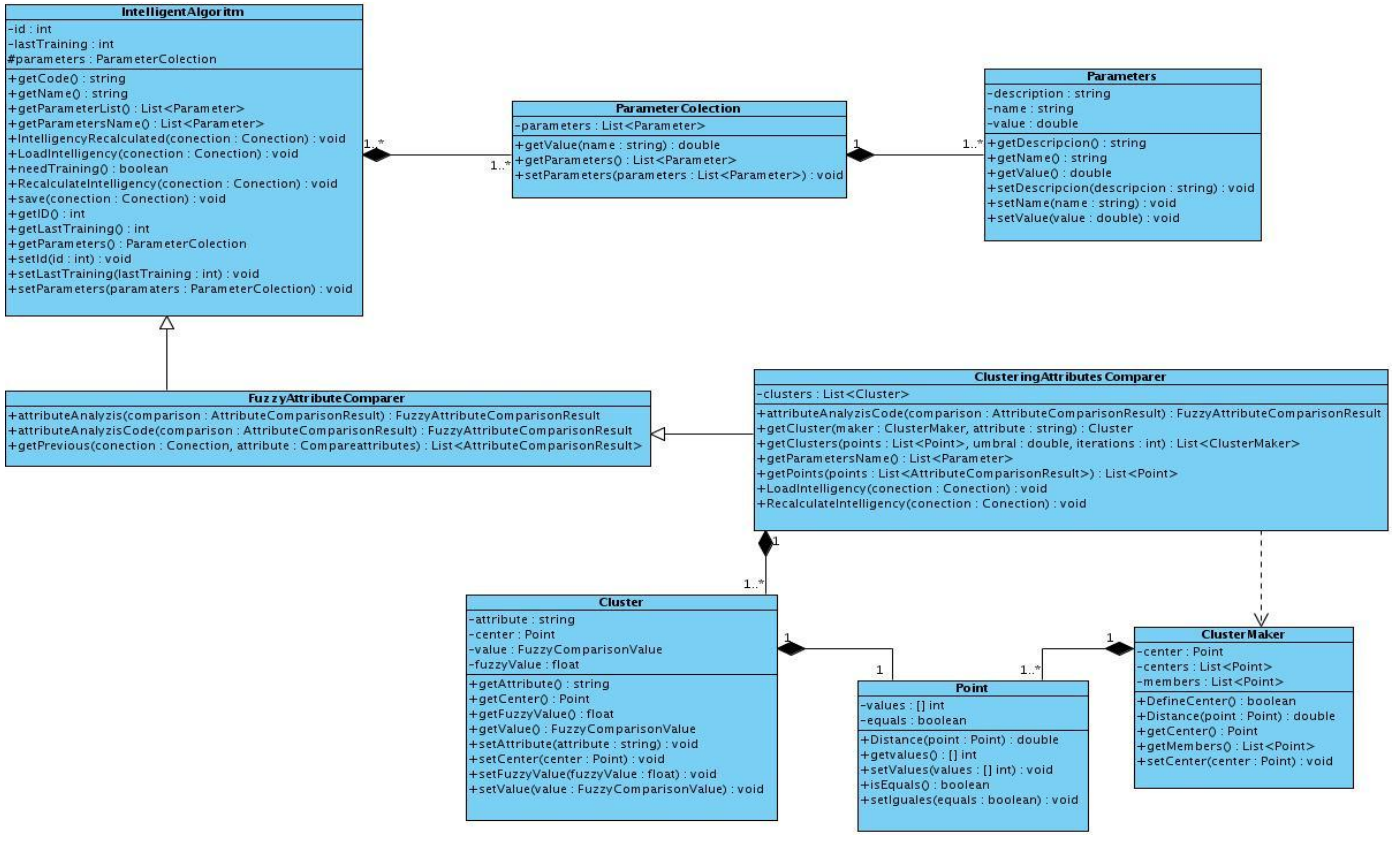


Imagen 7: Diagrama de clases del algoritmo de clustering, variante del clustering k-means

3.8.2 COMPARADORES DE DATOS GENERALES

3.8.2.1 ÁRBOL DE DECISIÓN ID3

Este algoritmo parte de la obtención de los resultados de las comparaciones de atributos realizadas, compuestos por la característica perteneciente a cada atributo. Cada rama de dicho árbol estará compuesta por el valor que puede tomar la variable objetivo. El grado de similitud retornado será el porciento de iguales positivos presentes en el nodo hoja al que se llegue por el camino que conforme el patrón previsto por los datos de entrada proporcionados al árbol.

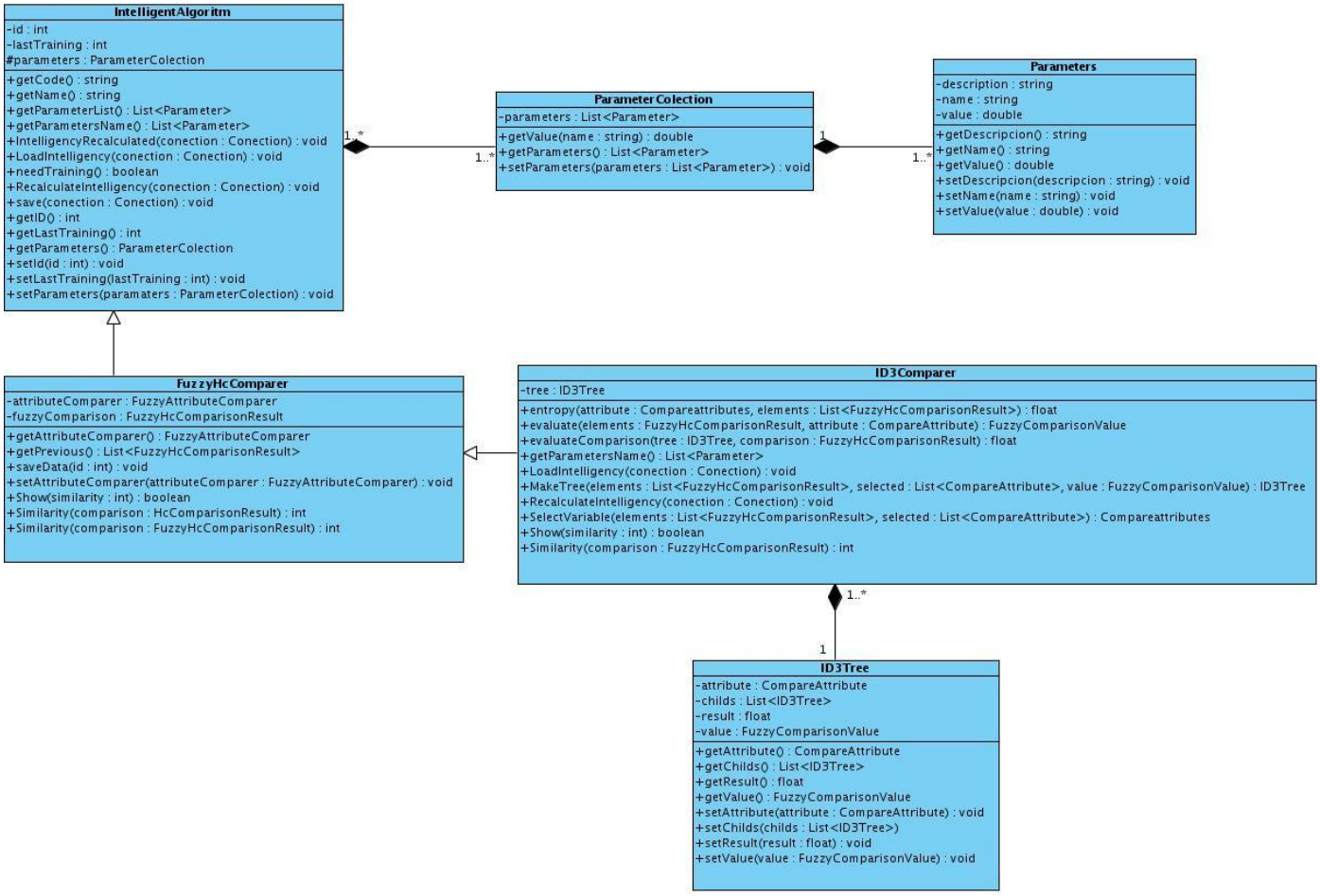


Imagen 8: Diagrama de clases del Árbol de decisión ID3

3.8.2.2 RED NEURONAL ARTIFICIAL

Para la implementación de un analizador de datos personales mediante la utilización de una red neuronal artificial, se decide implementar un perceptrón multicapa (multilayer perceptron). La red neuronal estará compuesta por una capa de neuronas de entrada, encargada de recibir las características de los atributos, dos capas ocultas y una capa salida. La capa de entrada está compuesta por 16 neuronas debido a que se toman dos neuronas para expresar en código binario el estado del atributo analizado. Esta codificación es la siguiente:

00 si la característica del atributo no está presente en al menos uno de los elementos.

01 si la característica del atributo es mala.

10 si la característica del atributo es regular.

11 si la característica del atributo es buena.

Las dos capas ocultas estarán compuestas por 13 y 10 neuronas respectivamente, dichas capas son las encargadas del procesamiento lineal de la información entre la capa de entrada y la capa de salida. Para la capa de salida se definen 7 neuronas, siendo este el menor número que se necesita para representar el valor máximo de similitud entre dos personas (100). La función de activación empleada para la activación de las neuronas es la función sigmoidea: $f(x) = \frac{1}{(1-\varepsilon^{-x})}$

Esta red es entrenada mediante el algoritmo de entrenamiento Backpropagation, el cual utiliza la variación del error para realizar la propagación del mismo hacia atrás, reajustando los pesos de las conexiones entre las capas de la red. Los pesos de las conexiones entre las neuronas fueron inicialmente establecidos utilizando un valor numérico entre 0 y 1, generado aleatoriamente.

Este algoritmo utiliza los siguientes parámetros de configuración:

Umbral: Valor numérico que debe sobrepasar la neurona para poder activarse.

LearningRate: Valor numérico que determina la velocidad de entrenamiento de la red.

CalmingRate, parámetro empleado para disminuir el LearningRate durante un proceso de entrenamiento en serie.

Iteraciones: Cantidad de iteraciones a realizar por el comparador.

Elasticity: Elasticidad de la función de activación.

CalmingElasticity: parámetro empleado para disminuir la Elasticity de la función de activación.

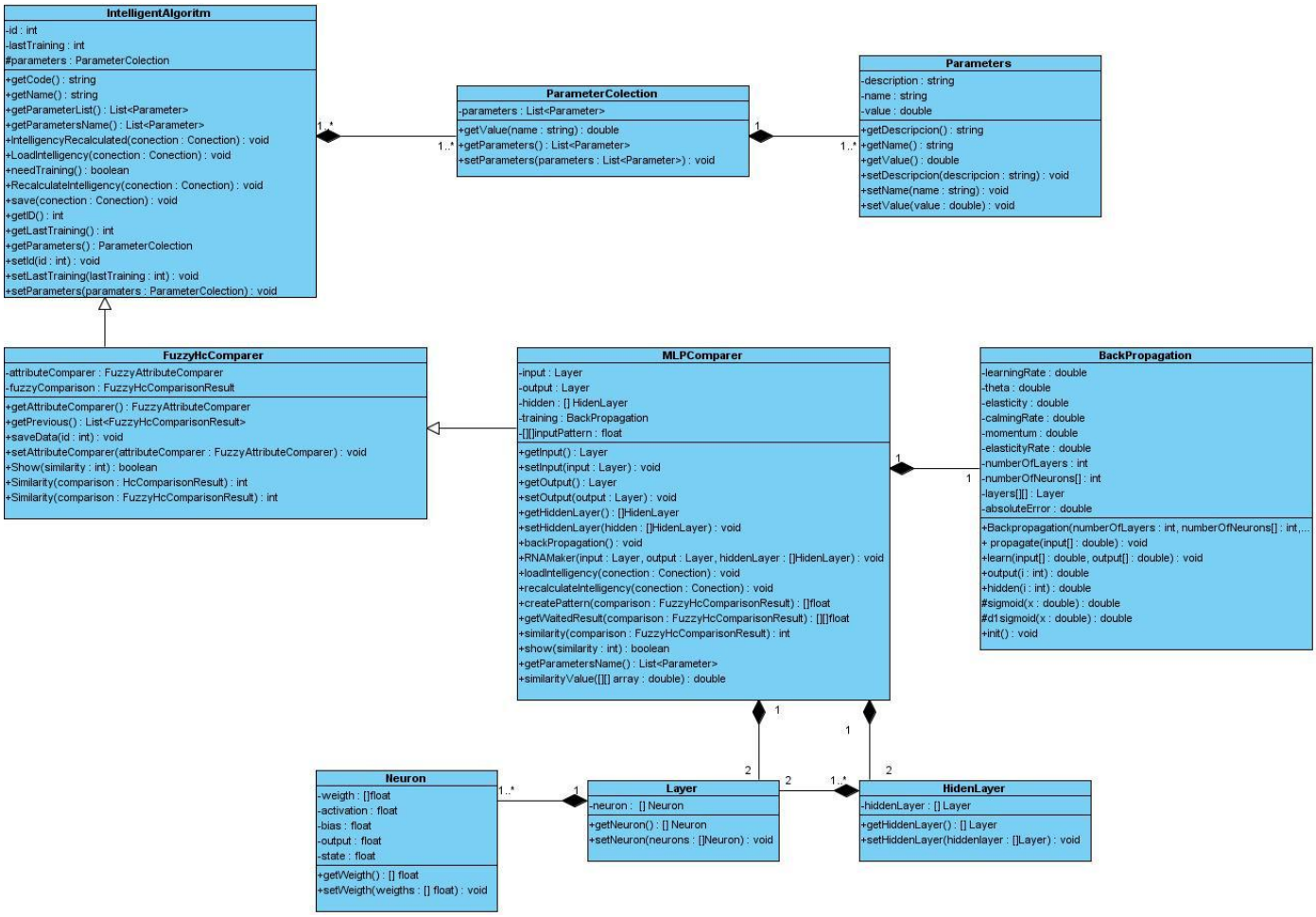


Imagen 9: Diagrama de Clases Red Neuronal Artificial

3.8.2.3 CLUSTERING K-MEANS

Para la implementación de uno de los comparadores encargado de realizar el análisis general de los datos personales, se decide la utilización del algoritmo de Clustering k-means. Este comparador recibe el resultado de las comparaciones de atributos realizadas. Con estos resultados conforma los puntos que a su vez forman parte de cada uno de los clústeres creados. Como resultado de la probabilidad de similitud de una nueva comparación se podrá asumir como la probabilidad de similitud que existe entre el clúster que presente mayor semejanza con la comparación actual.

Este algoritmo presenta los siguientes parámetros de configuración:

Umbral: valor numérico que representa el número de clústeres a generar por el algoritmo.

Iteraciones: cantidad máxima de iteraciones a realizar por el algoritmo en el momento de conformar los clústeres.

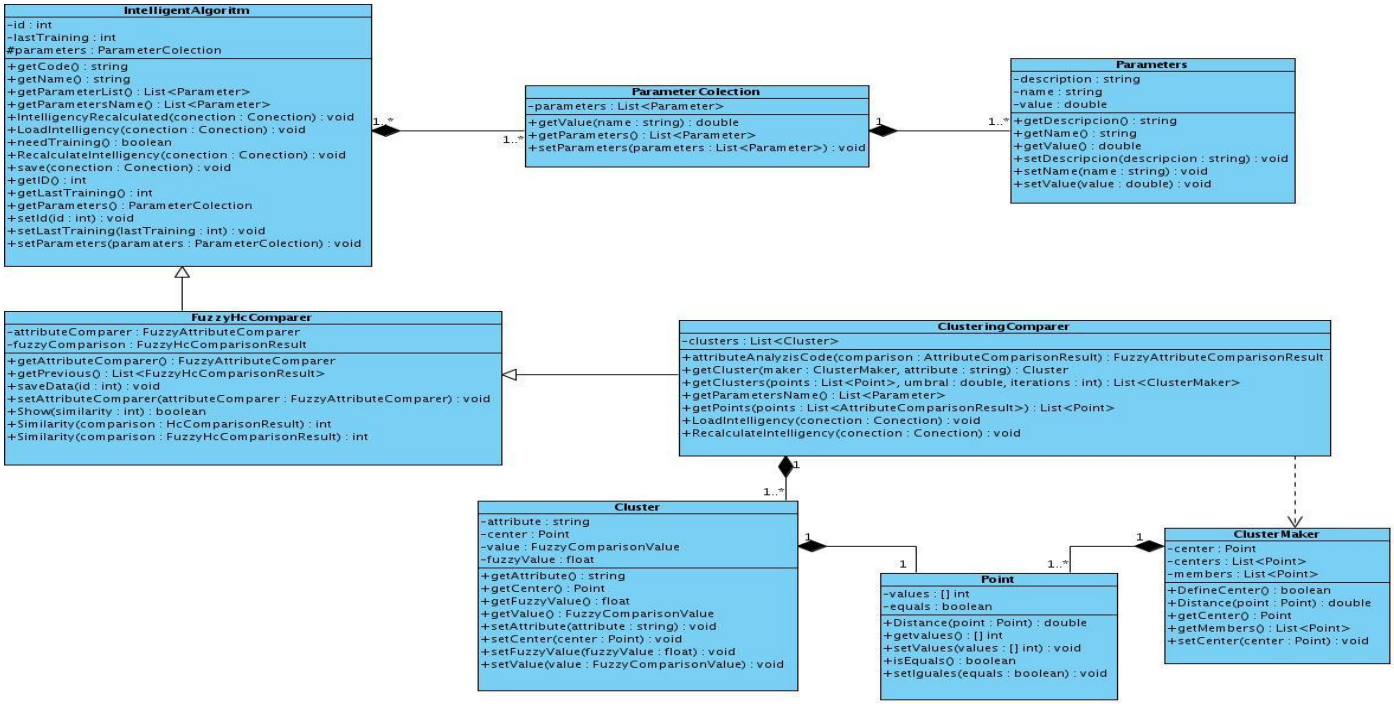


Imagen 10: Diagrama de clases del algoritmo Clustering k-means

3.9 DIAGRAMA DE COMPONENTES

Un componente es una parte física y reemplazable de un sistema. Es la implementación física de un conjunto de elementos lógicos, como clases y colaboraciones. Estos componentes son agrupados en diagramas que muestran la estructura de la aplicación así como la dependencia entre sus partes. (12)

3.9.1 DIAGRAMA DE COMPONENTES DEL SISTEMA

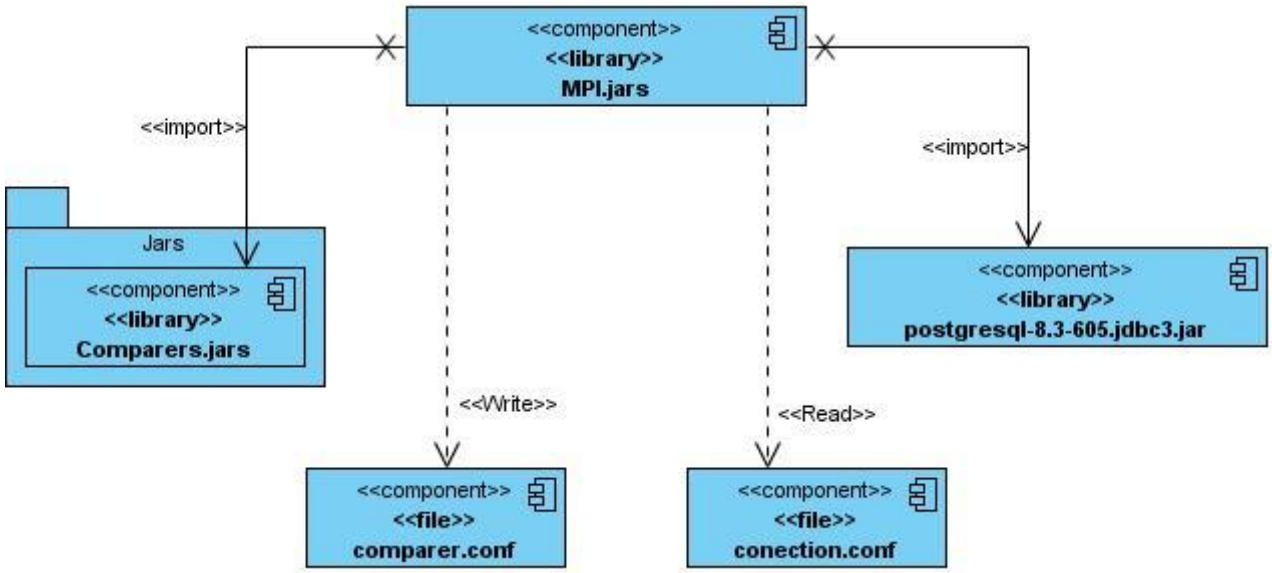


Imagen 11: Diagrama de componentes del sistema

En el presente capítulo se definieron los principales diagramas del diseño, como son diagrama de clases del diseño y diagrama de componentes, el cual, brinda una visión de las partes físicas con que cuenta la solución propuesta. Se describieron además los principales comparadores presentes en el sistema, obteniendo el diagrama de clases correspondiente a cada comparador. Se realizó una descripción ampliada de las principales clases controladores de los comparadores de atributos y de los comparadores generales de datos personales.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

En el presente capítulo se realiza la validación de la solución propuesta cumplimentar el objetivo planteado. Se realizan comparaciones entre los distintos comparadores presentes en el sistema, plasmando los resultados arrojados en cada ejecución. Se propone la configuración sobre la que debe ejecutarse el sistema, debido a que disminuye el tiempo de ejecución y la utilización de los recursos del servidor.

4.1 VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

La realización de pruebas o validaciones de datos sobre un sistema experto es una tarea un tanto difícil de lograr, debido a la gran utilización de recursos y medios tecnológicos para lograr este fin.

Para la validación de la solución se realizan comparaciones entre los distintos algoritmos con que cuenta la misma, plasmando en tablas comparativas cada uno de los resultados arrojados en cada ejecución de la solución.

Para lograr un mayor entendimiento y análisis de estos resultados se comparan cada uno de ellos con los resultados que se esperaba arrojara el sistema en cada ejecución, retornando así el número de falsos positivos y falsos negativos presentes.

Un resultado se considera un falso positivo cuando representa un valor que no es el valor real esperado, no siendo así cuando dicho resultado no representa un valor esperado, en este caso dicho resultado se considera un falso negativo.

En la realización de una primera ejecución del sistema, con el objetivo de poblar las distintas tablas presentes en la base de datos, de las cuales se nutrirán los algoritmos, y con la utilización de un CPU Intel Pentium 4 de 3GHz, con frecuencia de 2800.000 MHz y una memoria RAM de 1GB se obtuvieron los siguientes resultados.

Tabla de resultados de ejecución inicial			
Muestra Poblacional	**HC Similares**	**Resultados de las Comparaciones**	**Tiempo de Ejecución**
1224	9714	58284	135 Minutos con 26 Segundos

Tabla 18: Tabla de resultados de la ejecución inicial

Para la realización de las ejecuciones de los algoritmos se utilizó el siguiente tipo de hardware:

Tipo de Hardware Utilizado (1):

- CPU Intel Pentium 4 3GHz
- Frecuencia: 2800.000 MHz
- L2Cache: 2048 Kb
- Memoria RAM: 1 Gb
- Tarjeta de Red: Gigabit Fast Ethernet Controller

A continuación se plasman las configuraciones y los resultados de las ejecuciones de los comparadores presentes en el sistema. Algunos de estos comparadores presentan parámetros de configuración, los cuales se relacionan a continuación:

- Intervalo de Entrenamiento (IE).
- Umbral (U).
- LearningRate (LR).
- CalmingRate (CR).
- Elasticity (E).
- CalmingElasticity (CE)
- Iteraciones (IT)

La siguiente tabla muestra la configuración de los parámetros pertenecientes a cada uno de los comparadores presentes en el sistema:

Tabla de configuración de los parámetros de los comparadores			
Tipo Hardware	**Algoritmo**	**Muestra Poblacional**	**Parámetros**
1	Árbol de decisión ID3	1357	-
1	Árbol de decisión C 4,5	1357	IE:0.1
1	Clustering Variante k-means	1357	IE:0.1, U: 200.0
1	Clustering k-means	1357	U: 20.0 IT:1000
1	Red Neuronal Artificial	1357	U: 0.5,CR:0.4, LR: 0.7, E: 0.3 CE: 0.1, IT:1000

Tabla 19: Tabla de configuración de los parámetros de los comparadores

En la presente tabla se realiza una comparación de los resultados obtenidos posterior a la ejecución de los comparadores presentes en el sistema.

Tabla comparativa entre los Comparadores de atributos y los comparadores de datos personales					
Comparador de Atributos	**Comparador de Datos Personales**	**Resultado Obtenido**	**Tiempo de Ejecución**	**Falsos Positivos**	**Falsos Negativos**
Árbol de decisión C 4,5	Árbol de decisión ID3	7245	142 Min	126	37
Árbol de decisión C 4,5	Red Neuronal Artificial	6435	165 Min	133	59
Árbol de	Clustering K-	7527	153 Min	114	48

decisión C 4,5	means				
Clustering variante K-means	Árbol de decisión ID3	6126	123 Min	127	39
Clustering variante K-means	Red Neuronal Artificial	5233	145 Min	156	63
Clustering variante K-means	Clustering K-means	5196	119 Min	134	47

Tabla 20: Tabla comparativa entre los comparadores de atributos y los comparadores de datos personales

4.2 DESCRIPCIÓN DE LOS CASOS DE PRUEBA DE LA SOLUCIÓN PROPUESTA.

En el presente epígrafe se describe el modelo de prueba de caja negra para probar las funcionalidades de la solución propuesta, definiéndose además los modelos de casos de prueba asociados a las funcionalidades presentes en la solución.

Para lograr una buena calidad en la obtención de la solución se hace necesaria la confección de un correcto modelo de prueba.

Un buen diseño de prueba, sistemáticamente saca a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. Para garantizar la efectividad de las pruebas, estas deben ser realizadas por un equipo independiente al que realizó el sistema.

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software y son completamente indiferentes al comportamiento interno y la estructura del programa. Con estas se pretende demostrar que las funcionalidades del sistema son operativas, que la entrada se acepta de forma adecuada, que se produce un resultado correcto y que se mantiene la integridad de la información externa. Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están:

- Técnica de la partición de equivalencia.

- Técnica del análisis de valores límites.
- Técnica de grafos de causa-efecto. (21)

Técnica de partición de equivalencia

Divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para esta partición se basa en la evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada y regularmente estas condiciones pueden ser un valor numérico, un rango de valores, un conjunto de valores relacionados o una condición lógica.

Técnica del análisis de valores límites

Los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen. Las condiciones límite son aquellas que se hallan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. La técnica del análisis de valores límites complementa a la de partición equivalente, pues en lugar de centrarse solamente en las condiciones de entrada, deriva los casos de prueba también para el campo de salida.

Técnica de grafos de causa-efecto

En este método se debe entender los objetos (objetos de datos, objetos de programa tales como módulos o colecciones de sentencias del lenguaje de programación) que se modelan en el software y las relaciones que conectan a estos objetos. Una vez que se ha llevado a cabo esto, el siguiente paso es definir una serie de pruebas que verifiquen que todos los objetos tienen entre ellos las relaciones esperadas. En este método:

- Se crea un grafo de objetos importantes y sus relaciones.
- Se diseña una serie de pruebas que cubran el grafo de manera que se ejerciten todos los objetos y sus relaciones para descubrir errores.

Descripción de los casos de prueba.

A continuación se describen los casos de prueba para los principales casos de uso presentes en la solución.

Caso de uso: Listar Historias Clínicas similares.

Escenarios del Seleccionar historias clínicas similares	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Listar Historias Clínicas similares	Mostrar el listado de las Historias Clínicas similares satisfactoriamente.	Muestra un listado de las Historias Clínicas similares y el grado de similitud existente entre ellas.
EC 2: No se encuentra información	No existe información a mostrar.	Muestra el mensaje de información "No existe información a mostrar.
EC 3: Modificar historias clínicas similares	Permite acceder al caso de uso modificar historias clínicas similares satisfactoriamente.	Se selecciona la opción Modificar. Ver DCP **Ver detalles de Historias Clínicas similares.**
EC 4: Cancelar	Permite regresar a la interfaz de bienvenida del Módulo Admisión.	Regresa a la interfaz de bienvenida del módulo Admisión.

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Listar Historias Clínicas similares.	No se encuentra información	Modificar Historias Clínicas similares	Cancelar
Botón 1 (Cancelar)				NA
Botón 2 (Modificar)			NA	
Respuesta del Sistema	Muestra las Historias Clínicas similares agrupadas por la fecha en que se realizó la comparación. Además muestra: • Foto • Nombre • Primer Apellido • Segundo Apellido • Grado de similitud	Muestra el mensaje de información: "No existe información a mostrar."	Muestra la interfaz Ver datos de Historias Clínicas similares para igualar o descartar dichas Historias Clínicas.	Muestra la interfaz de entrada del módulo admisión.
Resultado de la Prueba				

Caso de uso: Ver detalles de Historias Clínicas similares

Escenarios del ver detalles de Historias Clínicas similares	**Descripción de la funcionalidad**	**Flujo Central**
EC1: Ver detalles de Historias Clínicas similares.	Ver detalles de Historias Clínicas similares satisfactoriamente.	Se muestra la interfaz Ver detalles de Historias Clínicas similares. Se muestran los datos de las Historias Clínicas similares. Selecciona la opción Salir. Regresa a la vista anterior.
EC2: Igualar Historias Clínicas similares.	Permite igualar las Historias Clínicas similares.	Se igualan las Historias Clínicas similares. Regresa a la vista anterior.
EC3: Descartar Historias Clínicas similares.	Permite descartar las Historias Clínicas similares	Se descartan las Historias Clínicas similares. Regresa a la vista anterior.
EC4: Salir	Permite regresar a la vista anterior.	Regresa a la vista anterior.

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Ver detalles de las Historias Clínicas similares satisfactoriamente	Igualar dos Historias Clínicas similares.	Descartar dos Historias Clínicas similares	Regresar a la vista anterior.
Botón 3 (Salir)				NA
Botón 4 (Igualar)		NA		
Botón 5 (Descartar)			NA	

Respuesta del Sistema	Muestra la interfaz Ver datos de Historias Clínicas similares. Muestra además: Foto, Nombre, Primer Apellido, Segundo Apellido, Sexo, Cédula, Fecha de Nacimiento y el grado de similitud existente entre las dos Historias Clínicas similares	Iguala las dos Historias Clínicas similares y regresa a la vista anterior	Descarta las dos Historias Clínicas similares y regresa a la vista anterior	Regresa a la vista anterior
Resultado de la Prueba				

Caso de uso: Deshacer cambios de Historias Clínicas similares.

Escenarios del Seleccionar historias clínicas similares	**Descripción de la funcionalidad**	**Flujo Central**
EC1: Listar las Historias Clínicas similares igualadas o descartadas.	Mostrar el listado de las Historias Clínicas similares igualadas o descartadas satisfactoriamente.	Muestra un listado de las Historias Clínicas similares igualadas o descartadas con el grado de similitud existente entre ellas.
EC2: No se encuentra información	No existe información a mostrar.	Muestra el mensaje de información “No existe información a mostrar.
EC3: Modificar Historias Clínicas similares	Permite acceder al caso de uso modificar Historias Clínicas similares satisfactoriamente.	Se selecciona la opción Modificar. Ver DCP **Deshacer cambios de Historias Clínicas similares**
EC4: Salir	Permite regresar a la página de bienvenida del módulo Admisión.	Regresa a la interfaz de bienvenida del módulo Admisión.

Id del escenario	EC 1	EC 2	EC 3	EC 4
Escenario	Listar Historias Clínicas similares igualadas o descartadas.	No se encuentra información	Deshacer cambios de Historias Clínicas similares.	Cancelar
Botón 6 (Cancelar)				NA
Botón 7 (Modificar)			NA	
Respuesta del Sistema	Muestra las Historias Clínicas similares agrupadas por la fecha en que se realizó la comparación. Además muestra: • Foto • Nombre • Primer Apellido • Segundo Apellido • Grado de similitud existente entre dichas Historias Clínicas, acompañado de un símbolo de igualdad o descarte.	Muestra el mensaje de información: "No existe información a mostrar."	Muestra la interfaz Ver datos de Historias Clínicas similares para deshacer cambios sobre dichas Historias Clínicas.	Muestra la interfaz de bienvenida del módulo admisión.
Resultado de la Prueba				

Caso de uso: Ver detalles de deshacer cambios de Historias Clínicas similares

Escenarios del ver detalles de Historias Clínicas similares	**Descripción de la funcionalidad**	**Flujo Central**
EC 1: Ver detalles de Historias Clínicas similares	Ver detalles de dos Historias Clínicas similares satisfactoriamente.	Se muestra la interfaz Ver detalles de Historias Clínicas similares Se muestran los datos de las Historias Clínicas similares Selecciona la opción Salir. Regresa a la vista anterior.
EC 2: Deshacer cambios de Historias Clínicas similares	Permite deshacer los cambios realizados sobre las Historias Clínicas similares(Igualar o Descartar)	Se deshace la operación realizada sobre las Historias Clínicas similares. Regresa a la vista anterior.
EC 3: Salir	Permite regresar a la vista anterior.	Regresa a la vista anterior

Id del escenario	EC 1	EC 2	EC 3
Escenario	Ver detalles de dos Historias Clínicas similares satisfactoriamente	Deshacer cambios realizados sobre las Historias Clínicas similares	Salir
Botón 8 (Salir)			NA
Botón 9 (Deshacer)		NA	
Respuesta del Sistema	Muestra la interfaz Ver datos de Historias Clínicas similares. Muestra además: Foto, Nombre, Primer Apellido, Segundo	Deshace la operación realizada sobre las Historias Clínicas similares.	Regresa a la interfaz anterior.

	Apellido, Sexo, Cédula, Fecha de Nacimiento y el grado de similitud existente entre las dos Historias Clínicas similares		
Resultado de la Prueba			

Posterior a la ejecución y análisis de los comparadores presentes en la solución se concluye que:

- La configuración propuesta para la ejecución del sistema es:
    - La utilización del algoritmo Árbol de decisión C 4,5 como algoritmo comparador de atributos en conjunto con el algoritmo Árbol de decisión ID3 como algoritmo comparador de datos generales, debido a que en las ejecuciones realizadas arrojaron el menor número de falsos negativos para la muestra poblacional inicial utilizada para realizar las comparaciones.

Se describieron además los casos de prueba referentes a los principales casos de uso presentes en el sistema.

CONCLUSIONES

Con la realización de la solución propuesta se cumplió con el objetivo y las tareas de la investigación, se concluye que:

- Los Sistemas de Gestión de Información Médica presentes en el Centro de Informática Médica carecen de un sistema encargado de la búsqueda de posibles personas duplicadas en ellos.
- Los sistemas analizados para la búsqueda de posibles personas duplicadas no son aplicables a las necesidades del Centro de Informática Médica.
- Se obtuvo un sistema encargado de la búsqueda de posibles personas duplicadas en los Sistemas de Gestión de Información Médica.
- Se obtuvo un sistema configurable y adaptable a los distintos escenarios en los que pueden operar los Sistemas de Gestión de Información Médica.
- Se desarrolló un conjunto de comparadores aplicable a los Algoritmos Comparadores, al Analizador de Atributos y al Analizador de datos personales.

RECOMENDACIONES

Para la mejora y extensión de la aplicación a la detección de datos duplicados de cualquier tipo se recomienda que:

- Adicionar nuevos algoritmos comparadores al sistema.
- Extender el modelo propuesto a la detección de datos duplicados de cualquier tipo.

REFERENCIAS BIBLIOGRÁFICAS

1. **Veracruzana, Universidad.** Instituto de Contaduría Pública. [En línea] http://www.uv.mx/icp/gruposinv.html.

2. **Instituto Federal de Acceso a la Información y Protección de Datos.** Instituto Federal de Acceso a la Información y Protección de Datos. [En línea] http://www.ifai.org.mx/SectorPublico/informacionGeneral.

3. **Gonzales-Cam, Celso.** *Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos.*

4. **LABORATORIO DE SISTEMAS INTELIGENTES, Facultad de Ingeniería. Universidad de Buenos Aires .** REDES NEURONALES APLICADAS AL FRAUDE EN TELEFONÍA CELULAR. [En línea] http://laboratorios.fi.uba.ar/lsi/p-grosser-proyectodetesis.htm.

5. **NOVERGES NATIVIDAD, SACRISTÁN VICENTE , MARGAIX LOURDES.** ALGUNAS APLICACIONES DE REDES NEURONALES ARTFICIALES EN DOCUMENTACIÓN. [En línea] http://personales.upv.es/ccarrasc/doc/2000-2001/Redes%20Neuronales%20Excalibur%20-%20Nativ.%20Noverges/redes.htm.

6. **Grupo de Inteligencia Artificial en Contabilidad y Administración de Empresas Universidad de Huelva.** SISTEMAS DE INDUCCIÓN DE ÁRBOLES DE DECISIÓN. [En línea] http://www.ciberconta.unizar.es/Biblioteca/0007/arboles.html.

7. **Plaza, Agnar Aamodt & Enric.** Case Based Reasoning. [En línea] http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/CaseBasedReasoning.

8. *Data Clustering: A Review.* **Michigan State University,Indian Institute of Science,The Ohio State University.** 3, s.l. : ACM Computing Surveys, Vol. 1.

9. **ORACLE.** *ORACLE HEALTHCARE MASTER PERSON INDEX.* 2010.

10. **SunMicrosystems, Inc.** *Developing SunMaster Patient Indexes.* Santa Clara,USA : s.n., 2008.

11. **Sun Microsystem.** Conozca más sobre la tecnología Java. [En línea] [Citado el:] http://java.com/es/about/..

12. **Campderrich Falgueras Benet, Maspons Ramon.** *Ingeniería del software.* Cataluña : UOC, 2003. 9788484297932.

13. UML,BPMN and Database Tool for Software Development. [En línea] http://www.visual-paradigm.com/.

14. **NetBeans.** NetBeans. [En línea] http://www.netbeans.org.

15. **The PostgreSQL Global Development Group.** *PostgreSQL 8.1.0 Documentation.* California : University of California, 2005.

16. pgAdmin: PostgreSQL database administration and management tool. [En línea] http://www.pgadmin.org/.

17. **Real Academia Española.** Diccionario de la Real Academia Española. [En línea] [Citado el: 15 de enero de 2011.] www.rae.com.

18. **IEEE Standars Asociation.** IEEE SA. [En línea] http://standards.ieee.org/findstds/standard/1471-2000.html.

19. **Microsoft.** MSDN. [En línea] http://msdn.microsoft.com/en-US/library/ee824462%28v=CS.10%29.aspx.

20. **Kiccillof, Reynoso – Nicolás Carlos.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : Universidad de Buenos Aires, 2006.

21. **Juristo Natalia, Moreno Ana M., Vegas Sira.** *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 17 DE OCTUBRE DE 2006.

BIBLIOGRAFÍA

1. **Veracruzana, Universidad.** Instituto de Contaduría Pública. [En línea] http://www.uv.mx/icp/gruposinv.html.

2. **Instituto Federal de Acceso a la Información y Protección de Datos.** Instituto Federal de Acceso a la Información y Protección de Datos. [En línea] http://www.ifai.org.mx/SectorPublico/informacionGeneral.

3. **Gonzales-Cam, Celso.** *Algoritmos fonéticos en el desarrollo de un sistema de información de marcas y signos distintivos.*

4. **LABORATORIO DE SISTEMAS INTELIGENTES, Facultad de Ingeniería. Universidad de Buenos Aires .** REDES NEURONALES APLICADAS AL FRAUDE EN TELEFONÍA CELULAR. [En línea] http://laboratorios.fi.uba.ar/lsi/p-grosser-proyectodetesis.htm.

5. **NOVERGES NATIVIDAD, SACRISTÁN VICENTE , MARGAIX LOURDES.** ALGUNAS APLICACIONES DE REDES NEURONALES ARTFICIALES EN DOCUMENTACIÓN. [En línea] http://personales.upv.es/ccarrasc/doc/2000-2001/Redes%20Neuronales%20Excalibur%20-%20Nativ.%20Noverges/redes.htm.

6. **Grupo de Inteligencia Artificial en Contabilidad y Administración de Empresas Universidad de Huelva.** SISTEMAS DE INDUCCIÓN DE ÁRBOLES DE DECISIÓN. [En línea] http://www.ciberconta.unizar.es/Biblioteca/0007/arboles.html.

7. **Plaza, Agnar Aamodt & Enric.** Case Based Reasoning. [En línea] http://www.aaai.org/AITopics/pmwiki/pmwiki.php/AITopics/CaseBasedReasoning.

8. *Data Clustering: A Review.* **Michigan State University,Indian Institute of Science,The Ohio State University.** 3, s.l. : ACM Computing Surveys, Vol. 1.

9. **ORACLE.** *ORACLE HEALTHCARE MASTER PERSON INDEX.* 2010.

10. **SunMicrosystems, Inc.** *Developing SunMaster Patient Indexes.* Santa Clara,USA : s.n., 2008.

11. **Sun Microsystem.** Conozca más sobre la tecnología Java. [En línea] [Citado el:] http://java.com/es/about/..

12. **Campderrich Falgueras Benet, Maspons Ramon.** *Ingeniería del software.* Cataluña : UOC, 2003. 9788484297932.

13. UML,BPMN and Database Tool for Software Development. [En línea] http://www.visual-paradigm.com/.

14. **NetBeans.** NetBeans. [En línea] http://www.netbeans.org.

15. **The PostgreSQL Global Development Group.** *PostgreSQL 8.1.0 Documentation.* California : University of California, 2005.

16. pgAdmin: PostgreSQL database administration and management tool. [En línea] http://www.pgadmin.org/.

17. **Real Academia Española.** Diccionario de la Real Academia Española. [En línea] [Citado el: 15 de enero de 2011.] www.rae.com.

18. **IEEE Standars Asociation.** IEEE SA. [En línea] http://standards.ieee.org/findstds/standard/1471-2000.html.

19. **Microsoft.** MSDN. [En línea] http://msdn.microsoft.com/en-US/library/ee824462%28v=CS.10%29.aspx.

20. **Kiccillof, Reynoso – Nicolás Carlos.** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : Universidad de Buenos Aires, 2006.

21. **Juristo Natalia, Moreno Ana M., Vegas Sira.** *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 17 DE OCTUBRE DE 2006.

22. **Universidad de Extremadura (UEx).** [En línea] http://campusvirtual.unex.es/cala/epistemowikia/index.php?title=L%C3%B3gica_difusa.

23. **CECAM.** *Revista Cubana de Informática Médica.* [En línea] http://www.rcim.sld.cu/revista_2/articulos_html/febles.htm.

24. **Salgueiro, Sr. Fernando A.** *Sistemas Inteligentes para el Modelado del Tutor.* s.l. : Universidad de Buenos Aires, 2005.

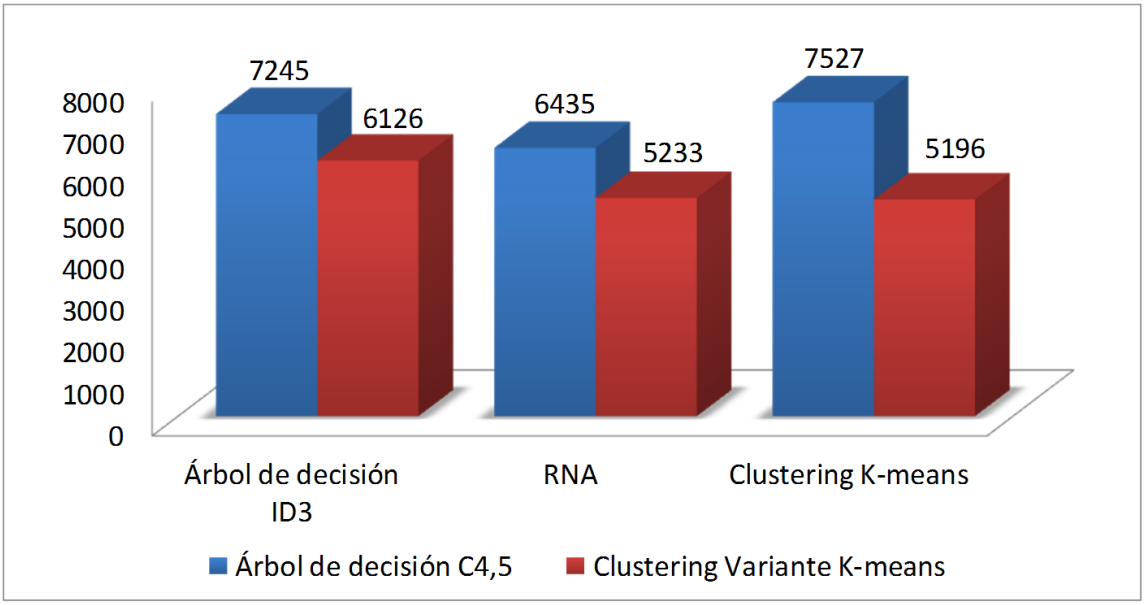
25. Tutorial UML, Diagramas de casos de uso. *Clickear.com.* [En línea] Noviembre de 2010. http://www.clikear.com/manuales/uml/diagramascasouso.aspx.

26. **Donato Morgade, Gerardo.** *ClioBD. Sistema de control de versiones para bases de datos.* 2009.

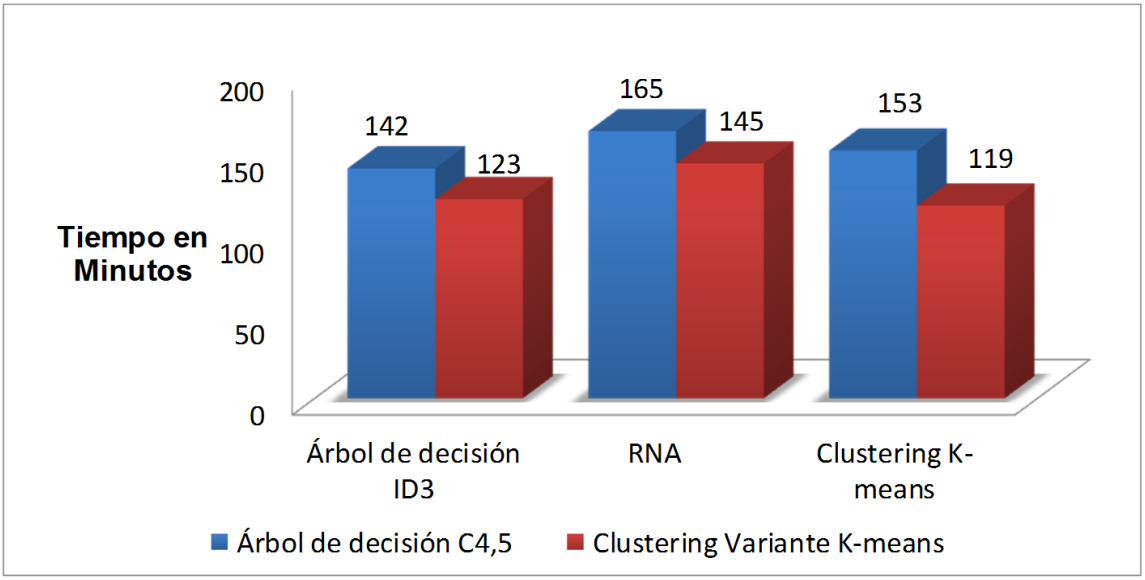
27. **Zubco Jose, Pardillo Jesús , Trujillo Juan.** *Minería de datos con clustering en espacios multidimensionales mediante modelos conceptuales extendiendo UML.*

ANEXOS

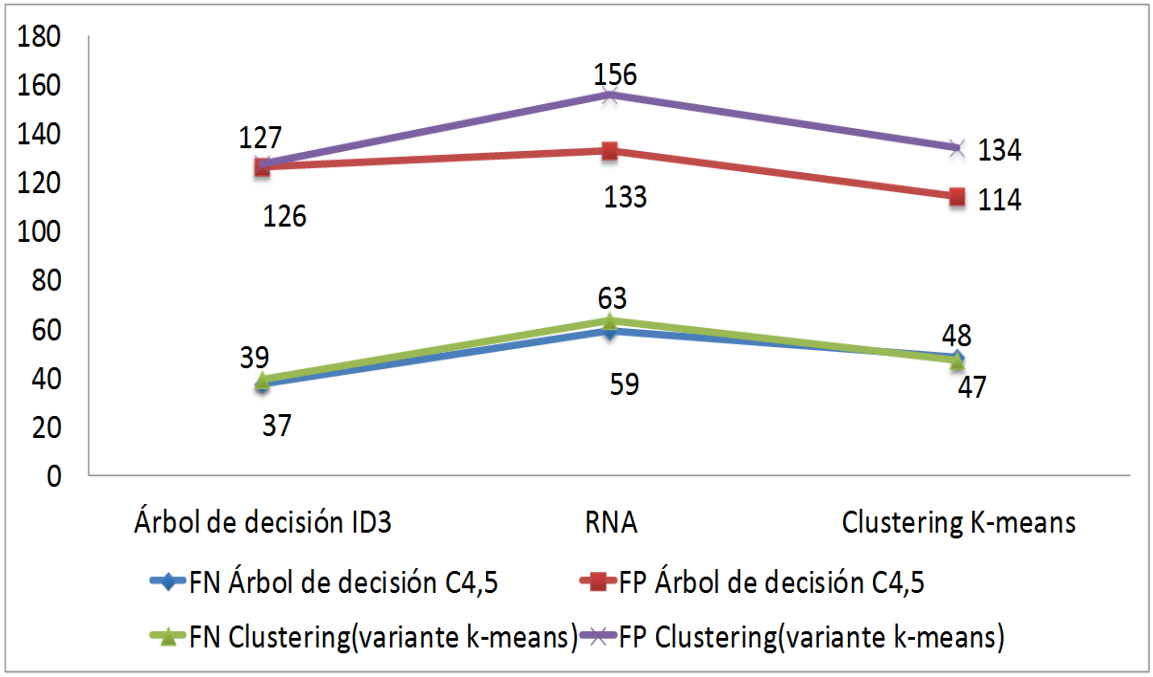
A continuación se muestran varias gráficas en las que se plasman los diferentes resultados obtenidos en las ejecuciones realizadas por el sistema.



Gráfica 1: Historias Clínicas Electrónicas similares obtenidas



Gráfica 2: Tiempo de ejecución de los algoritmos presentes en el sistema.



Gráfica 3: Falsos negativos y Falsos positivos obtenidos en la ejecución de los algoritmos.

A continuación se muestran algunas de las interfaces presentes en el Sistema de Información Hospitalaria alas HIS, dichas interfaces le permiten al usuario ver los resultados de las comparaciones realizadas por la solución propuesta para su aprobación o rechazo. Además le permite al usuario adicionar una nueva librería de comparadores al sistema y realizar una nueva configuración de la solución.

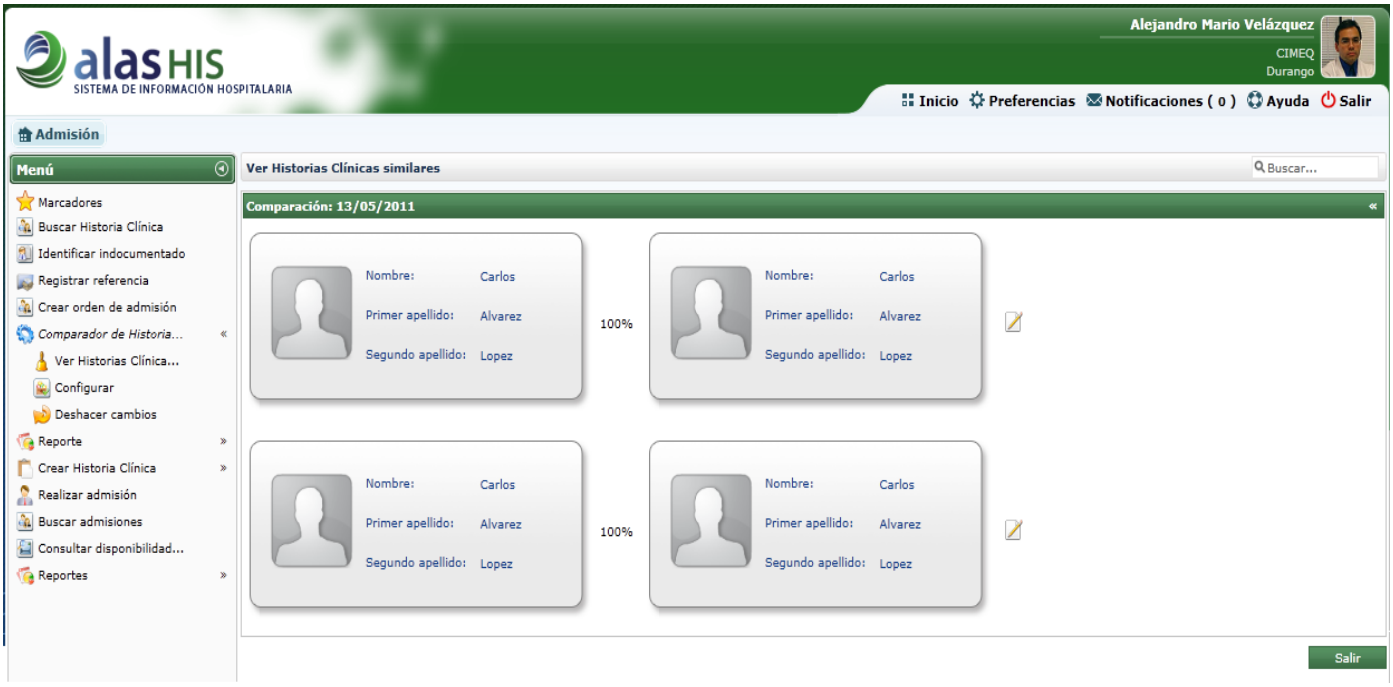


Imagen 12: Ver Historias Clínicas similares.

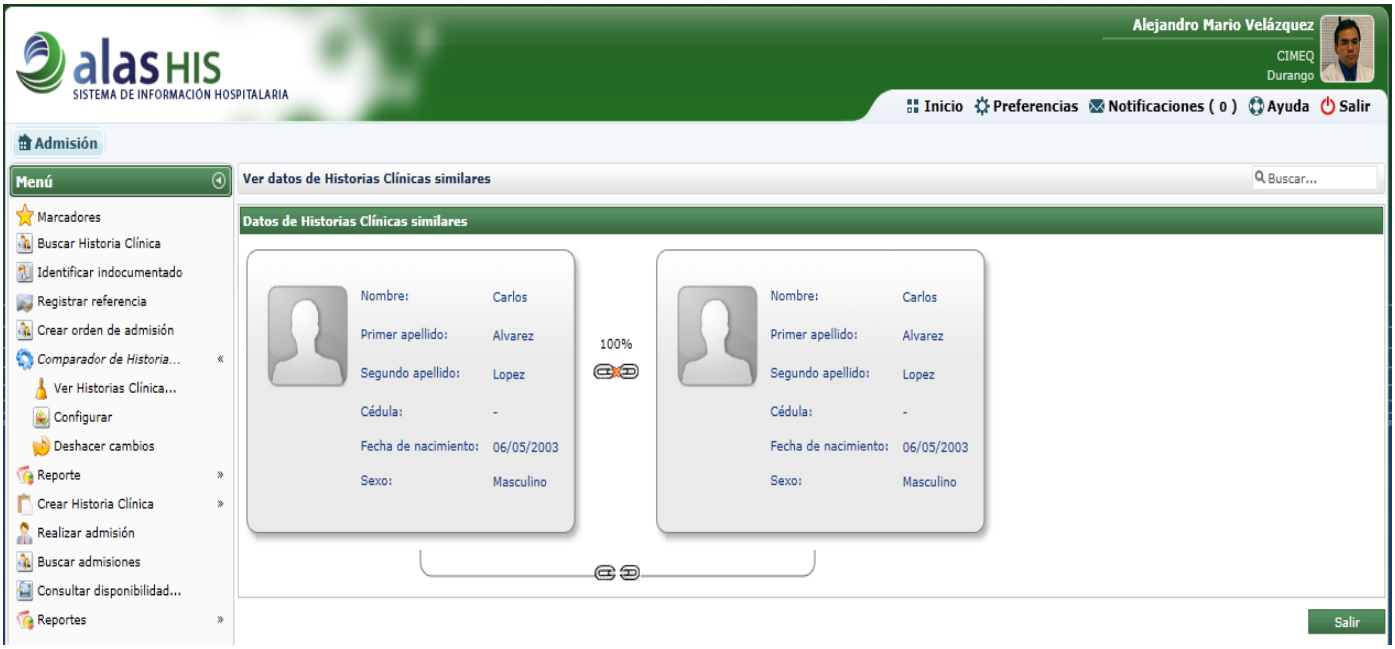


Imagen 13: Ver detalles de Historias Clínicas similares

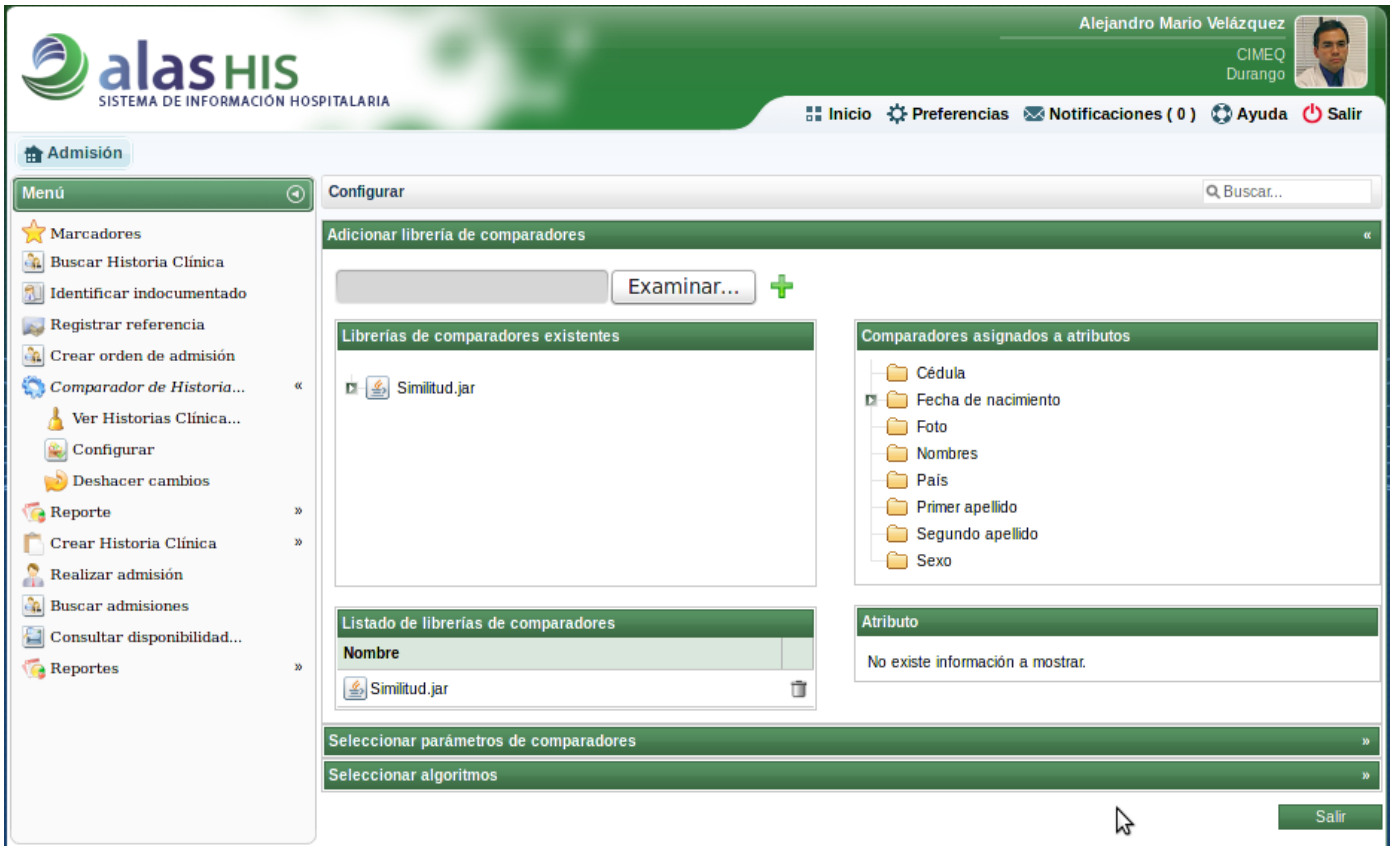


Imagen 14: Configuración

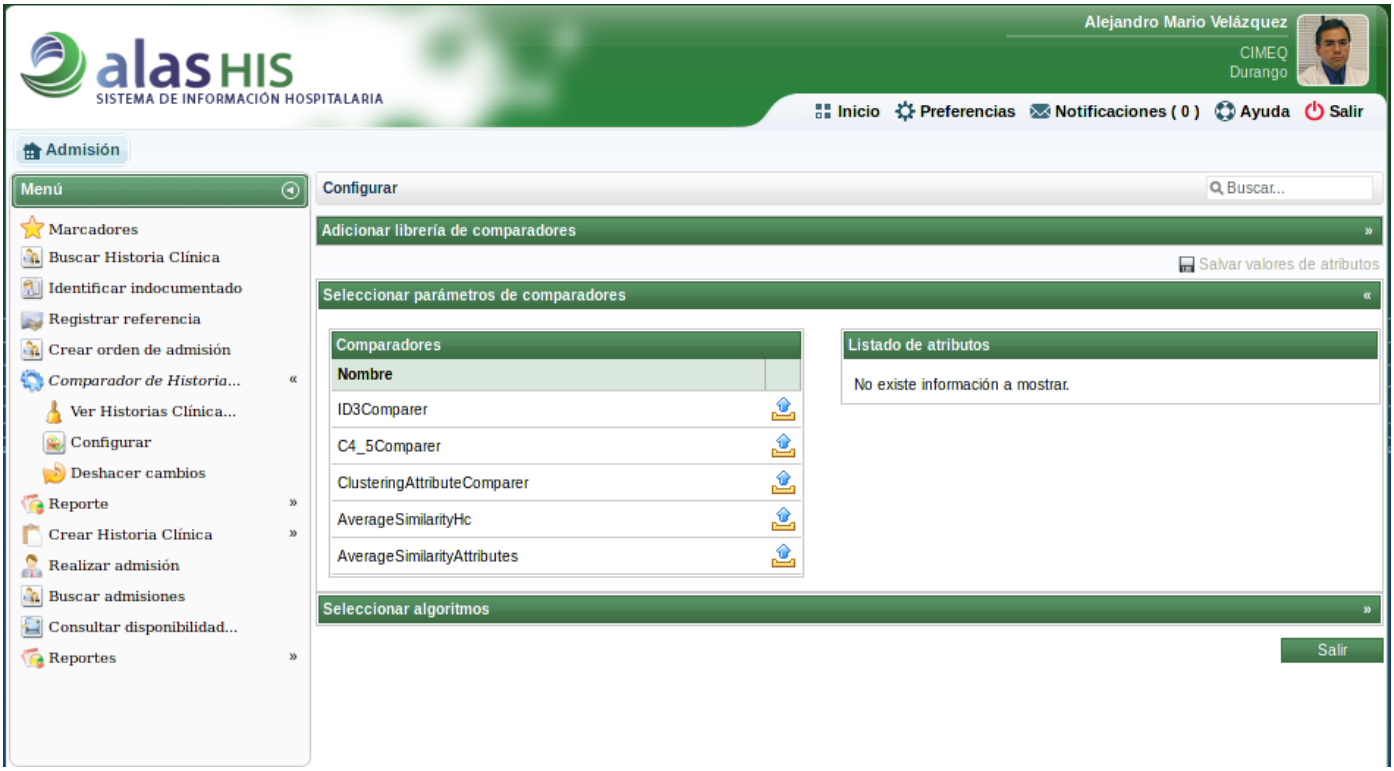


Imagen 15: Configuración de parámetros de algoritmos.

GLOSARIO DE TÉRMINOS

BSD: La licencia BSD pertenece al grupo de licencias de software libre. Es la licencia de software otorgada principalmente para los sistemas BSD (Berkeley Software Distribution). Permite el uso del código fuente en software no libre(o lo que es igual, desarrollo tremendamente rápido, tanto para uso comercial como otros). El autor únicamente mantiene la protección del copyright para posibles reclamaciones de correcta atribución de autoría de la obra o trabajos derivados.

GPL: La GPL (General Public License o licencia pública general) es una licencia creada por la Free Software Foundation. Esta licencia se enfoca más hacia los derechos del usuario, lo que permite la copia, modificación y redistribución del software.

Alan Turing: Matemático Inglés conocido como el padre de la Inteligencia Artificial.