UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
FACULTAD 1

Sistema para la distribución del proceso de búsqueda de huellas dactilares en el banco de datos de un AFIS.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: *Melvis Machin Armas*
Royli Hernández Delgado

Tutor: *Dr.C. Yusnier Valle Martínez*
Co-tutor: *MSc. Rubén Cruzata Santos*

La Habana. Junio de 2013

DECLARACIÓN JURADA DE AUTORÍA

Declaramos que somos los autores del resultado que exponemos en la presente memoria titulada *Sistema para la distribución del proceso de búsqueda de huellas dactilares en el banco de datos de un AFIS*, para optar por el título de Ingeniero en Ciencias Informáticas.

El presente trabajo fue desarrollado en el transcurso de los años 2012-2013.

Finalmente declaramos que todo lo anteriormente expuesto se ajusta a la verdad, y asumimos la responsabilidad moral y jurídica que se derive de este juramento profesional. Autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Y para que así conste, firmamos la presente declaración jurada de autoría en La Habana a los 20 días del mes de Junio del año 2013.

Autor: Melvis Machin Armas

Autor: Royli Hernández Delgado

Tutor: Dr.C. Yusnier Valle Martínez

Co-tutor: MSc. Rubén Cruzata Santos

Agradecimientos

A mi madre por su amor incondicional, por tantos sacrificios, por ser un ejemplo para mí, por su fuerza y temple.
A mi padre, principal responsable de que haya escogido este camino, por confiar tanto en mí. Por ser un ejemplo de superación y responsabilidad.
A mi hermana, la persona que más amo en el mundo, gracias por estar a mi lado cuando más lo necesito.
A mi familia, por todo el apoyo que me han brindado para lograr cada una de mis metas. A mi segunda madre Lisi, a mis abuelas queridísimas, a las Nerytas.
Al amor de mi vida, mi secuaz, por impulsarme a ser cada día mejor, por sus constantes revisiones, por sus consejos y preocupación.
A Adriancito, mi tutor en off, por sus ideas geniales, por sus consejos, por tanto tiempo dedicado.
A Sergio, el logística de esta tesis, por sus largas y decisivas horas de configuraciones en el nodo.
A todos mis amigos, en especial a Eidys, Dayana, Betty Boo, Mari, Aly, Fonsi y Soto, quienes han estado siempre a mi lado.
A Royli, mi compañero en estos cinco años, lo mejor que me ha ocurrido en esta universidad, mi hermano para toda la vida.

Melvis

A mis queridos padres por guiarme por el camino correcto, por quererme tanto, por confiar en mí y por ayudarme a llegar hasta aquí.
A mis abuelos por su constante preocupación y por darme tanto cariño.
A mi querida hermana Julita, que ha sido la principal fuente de inspiración en mis estudios, mi vida y mis contraseñas. A mi hermano por estar siempre a mi lado.
A mi tío Migue por ayudarme cuando lo he necesitado.
A mi familia por apoyarme y estar siempre de mi lado.
A mi novia Yanara por su amor, su dedicación y su comprensión en estos últimos años de mi carrera.
A mis super amigos Nilberto, Herbert, Sergio y Yaciel Orlando por nunca decir que no y seguirme incondicionalmente en todas mis ocurrentes aventuras.
A mi amigo y mejor compañero de máquina Adrian por ser incondicional y por sus ideas siempre tan oportunas.
A mis amigos del apartamento los TUFES, Daniel, José, Aynel, Yam y Yohevis, a Ariel y Alisnay por tener siempre un lugar para mí donde estuviesen y por los inolvidables momentos que compartimos.
A mi Melvis por demostrarme ser una verdadera amiga desde el día que me conoció y por ayudarme a escalar en estos 5 años esta cima de mis sueños.

Royli

A todos los profesores que contribuyeron a nuestra formación como profesionales, en especial a nuestros tutores Yusnier Valle y Rubén Cruzata, al profesor Yassef por enseñarnos a codificar los primeros algoritmos, al insuperable profesor Joel por enseñarnos a poner el corazón en lo que hacemos, al profesor Yadier por sus irónicos pero constructivos comentarios donde siempre hubo algo que aprender, y al profesor Landrián por estar disponible para aclarar nuestras dudas siempre que lo necesitamos.

A todos aquellas personas que de una forma u otra nos ayudaron en la realización de este trabajo.

Dedicatoria

A la memoria de mi abuelo Andrés, quien sé, estaría muy orgulloso de mí.

A mi abuela Carmita, quien con su amor sobrenatural

ha llenado mi vida de felicidad.

Melvis

A mis padres y a mi tío Michel por lo mucho que representan para mí.

Royli

Resumen

El proceso de búsqueda en el banco de datos de un sistema de identificación basado en huellas dactilares (AFIS), influye significativamente en la obtención de tiempos de respuesta adecuados y en la efectividad del sistema en general. Entre las estrategias de búsqueda pueden mencionarse la centralización en un nodo que procese toda la información y la distribución del proceso en nodos que participen paralelamente en la búsqueda.

En el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) se desarrolló un AFIS para ser incluido en sus soluciones informáticas. Este sistema reporta una baja tolerancia a fallos y tiempos de respuestas ineficientes utilizando un banco de datos considerablemente pequeño con respecto a los manejados por otros sistemas homólogos, debido a la implementación de un proceso de búsqueda centralizado. Este trabajo propone un sistema que brinde un servicio de distribución, de modo que la búsqueda ocurra concurrentemente en varios nodos, disminuyendo los tiempos de respuesta del AFIS y garantizando que el fallo de un nodo no determine el fallo de la operación en general. Este sistema tiene en cuenta además, la alta escalabilidad y el futuro crecimiento de los datos.

El documento recoge los resultados de la investigación realizada, haciendo un estudio de las principales características de los sistemas AFIS analizados y los procesos de búsqueda de huellas asociados a estos. Se explica la arquitectura y el diseño del sistema propuesto. Se describen las herramientas y tecnologías utilizadas, así como los artefactos generados en el proceso de desarrollo.

Palabras clave: AFIS, banco de datos, búsqueda, distribución, escalabilidad, tiempos de respuesta, tolerancia a fallos.

Índice general

Índice de figuras

Índice de tablas

Introducción

La seguridad se ha convertido en un asunto de actualidad que es motivo de preocupación en la protección de los bienes de cualquier entidad. Muchos autores se refieren a este tema como un proceso continuo en el que, entre otros aspectos, se definen los niveles de acceso a recursos e información en correspondencia con la importancia que estos poseen para una entidad, de manera que la identificación de los individuos se convierte en factor fundamental. Para garantizar la identificación de un individuo existen diferentes métodos. Entre los más convencionales se encuentra el uso de tarjetas identificativas tales como el DNI (Documento de Identificación Nacional) o credenciales especiales creadas para un ambiente específico. Sin embargo, con el incremento de la potencia de cómputo en los últimos años y las ciencias informáticas en general, el uso de otros métodos como la biometría han sido aplicados a sistemas modernos que permiten identificar unívocamente a un individuo.

La biometría es la ciencia que se encarga del estudio de las características físicas o conductuales inherentes a un individuo que son permanentes hasta su muerte y de gran singularidad e invariabilidad. Entre estos rasgos, los más usados son el iris, la geometría de la mano, la huella dactilar, la voz, la firma manuscrita y la manera de caminar. De ellos, las huellas dactilares (patrones gráficos de crestas presentes en los dedos humanos) se encuentran entre las características humanas más fiables utilizadas para la identificación.

Los sistemas biométricos basados en el reconocimiento de huellas dactilares, también conocidos como AFIS (Automatic Fingerprint Identification Systems) por sus siglas en inglés, comenzaron a desarrollarse a partir de la década del sesenta del pasado siglo, y su desarrollo vertiginoso y aplicación incremental han permitido utilizar las huellas dactilares como medio de identificación en múltiples escenarios.

Uno de los componentes fundamentales de estos sistemas es el algoritmo de comparación de huellas dactilares, encargado de comparar dos huellas y determinar el grado de similitud entre ellas. Para la identificación de un individuo es necesario un proceso de búsqueda que obtenga la similitud arrojada por este algoritmo entre la huella capturada por el sistema y las huellas almacenadas en el banco de datos, que resulta esencial para el éxito de la identificación. La robustez de este proceso influye significativamente en la obtención de tiempos de respuestas adecuados y en la efectividad del sistema, aun cuando pudiera ocurrir un crecimiento futuro del banco de datos que este maneja.

Debido al gran tamaño que suelen alcanzar los bancos de datos de los AFIS es necesario la reducción del espacio de búsqueda. Por ejemplo, una de las formas más conocidas de lograrlo es mediante la indexación. Esta técnica consiste en definir una clasificación única de las huellas para que sean indexadas y clusterizadas de acuerdo a los patrones presentes en ellas. Por el contrario, en muchos casos no se cuenta con algoritmos de indexación o con otras técnicas que agilicen el proceso de búsqueda, por lo que se realiza un acercamiento más rudimentario y simple mediante la comparación de la huella introducida en el sistema con cada una de las huellas almacenadas en el banco de datos. Aunque

resulte impracticable esta opción, la elección de una adecuada estrategia de búsqueda puede influir en la disminución de los tiempos del proceso.

La estrategia de búsqueda utilizada está estrechamente vinculada a la eficiencia del proceso en general, dentro de la cual se incluyen factores importantes como la potencia de hardware y la arquitectura del sistema. Aumentar la potencia de hardware de las unidades de cómputo u optar por una arquitectura distribuida en la cual se puedan integrar paulatinamente nodos que participen en la búsqueda mediante una variante "divide y vencerás" podrían ser alternativas a tener en cuenta cuando se plantea una estrategia. Sin embargo, cada una de estas propuestas tiene inconvenientes muy serios.

La centralización del proceso de búsqueda en un nodo con alta potencia de hardware permite fortalecer la seguridad y en la práctica suele ser inevitable. Este nodo procesa toda la información de forma lineal afectando directamente al tiempo de búsqueda, y su fallo podría provocar un colapso del sistema. Por otra parte, utilizar un modelo distribuido aportaría mayor flexibilidad en cuanto a la cantidad de unidades participantes en la búsqueda, de modo que el proceso ocurriría concurrentemente en todos los nodos disminuyendo teóricamente el tiempo de búsqueda. El fallo de un nodo no determina el fallo de la operación en general, ya que los demás podrían completar el proceso de búsqueda. Sin embargo, este modelo es difícil de implementar.

El Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) necesita un AFIS para ser utilizado en sus soluciones. En su Departamento de Biometría se desarrolló una variante de un AFIS que no incluye todos los servicios estandarizados en estos sistemas, y cuya estrategia de búsqueda de huellas dactilares en el banco de datos se basa en un modelo centralizado. Su tiempo de respuesta es de aproximadamente 3 segundos en identificar a un individuo comparando la huella de un pulgar contra los dos pulgares almacenados por usuario en un banco de datos de 10 mil usuarios. Este tiempo de respuesta resulta ineficiente teniendo en cuenta que el número de huellas almacenadas no es significativo comparado con otros bancos de datos cuyas cifras oscilan en el orden de los millones, y previendo que el aumento futuro de huellas registradas influirá directamente en el incremento de este tiempo, demorando la identificación y por consiguiente la pérdida de calidad del AFIS como producto.

Se podría pensar en la reutilización de componentes destinados al proceso de búsqueda en los AFISs más conocidos y con mejores resultados. Sin embargo, estos sistemas han sido desarrollados bajo licencias privativas y de ellos solo se conoce la información referente a sus principios de funcionamiento y estrategias de búsqueda.

Estos factores que dificultan la identificación de un individuo en un tiempo adecuado permiten identificar una **problemática** de la cual emerge el **problema de investigación**: ¿Cómo disminuir los tiempos de respuesta del proceso de identificación del AFIS del Departamento de Biometría del CISED?

Para encontrar una solución al problema planteado se define como **objeto de estudio** el proceso de

identificación de personas mediante huellas dactilares.

El **objetivo general** será desarrollar un sistema que permita distribuir el proceso de búsqueda de huellas dactilares en el banco de datos del AFIS del Departamento de Biometría del CISED, de modo que se obtengan mejores tiempos de respuesta en el proceso de identificación, teniendo en cuenta la escalabilidad del AFIS y el crecimiento futuro de su banco de datos.

Para cumplir el objetivo general se han establecido los siguientes **objetivos específicos**:

- Confección del marco teórico de la investigación para lograr un mayor entendimiento sobre el estado del arte en Cuba y en el mundo referente a los sistemas de identificación mediante huellas dactilares y los procesos de búsqueda de huellas asociados a estos.
- Realizar el diseño del sistema para su posterior implementación.
- Implementar un sistema que permita distribuir el proceso de búsqueda de huellas dactilares en el banco de datos de un AFIS.
- Realizar pruebas de software al sistema desarrollado para verificar su correcto funcionamiento.

El resultado de la investigación influirá directamente en el proceso de búsqueda de huellas dactilares en el banco de datos de un AFIS como **campo de acción**.

La investigación se rige por la siguiente **hipótesis**: El desarrollo de un sistema que distribuya el proceso de búsqueda de huellas dactilares en el banco de datos del AFIS del Departamento de Biometría del CISED permitirá obtener mejores tiempos de respuesta, teniendo en cuenta la escalabilidad del AFIS y el crecimiento futuro de su banco de datos (Ver Anexo Operacionalización de variables).

Para llevar a cabo el objetivo del trabajo se plantean las siguientes **tareas de investigación**:

1. Analizar los referentes teórico-prácticos que preceden la realización del presente trabajo, en relación a los sistemas de identificación mediante huellas dactilares y los procesos de búsqueda de huellas asociados a estos (Melvis Machin Armas).
2. Definición de la metodología, las tecnologías y las herramientas a utilizar para el desarrollo del sistema (Melvis Machin Armas y Royli Hernández Delgado).
3. Confección del modelo de dominio del sistema para identificar los principales conceptos involucrados en la propuesta de solución del problema planteado (Melvis Machin Armas).
4. Discusión y definición de la arquitectura del sistema (Melvis Machin Armas y Royli Hernández Delgado).
5. Diseño del sistema.
   5.1. Definición de los patrones de diseño (Royli Hernández Delgado).
   5.2. Confección del diagrama de clases del diseño del sistema (Melvis Machin Armas).
6. Implementación del sistema.
   6.1. Definición del estándar de codificación (Royli Hernández Delgado).
   6.2. Definición de las tareas de ingeniería (Melvis Machin Armas).

6.3. Codificación de las historias de usuario (Melvis Machin Armas y Royli Hernández Delgado).
7. Realización de pruebas de software al sistema (Melvis Machin Armas y Royli Hernández Delgado).

Los **métodos científicos** que se emplearán son:

Métodos Teóricos

- Analítico - Sintético: Se utilizará en el análisis de los elementos más importantes relacionados con el modelo distribuido, para determinar las características generales de este tipo de arquitectura y los requisitos para su implementación.
- Análisis Histórico - Lógico: Se utilizará en el estudio de los antecedentes, la evolución y el desarrollo que han tenido los AFISs, así como valorar las tendencias actuales de los procesos de búsqueda de huellas asociados a ellos.
- Modelación: Se utilizará en la modelación de los diagramas generados en el diseño para un mejor entendimiento del sistema.

Métodos Empíricos

- Entrevista: Se utilizará para realizar entrevistas a especialistas en huellas dactilares, con el objetivo de obtener información relacionada con el tema de los procesos de búsqueda de huellas en el banco de datos de un AFIS, precisar el problema a resolver y definir los procesos que se llevan a cabo.
- Experimental: Se utilizará para registrar los tiempos de respuesta del AFIS desarrollado en el Departamento de Biometría del CISED con y sin el sistema propuesto.

Justificación de la investigación

Los sistemas biométricos basados en las huellas dactilares son medulares en la garantía de un alto nivel de seguridad e identificación personal en múltiples ámbitos de la sociedad. De la robustez de su proceso de búsqueda dependen el éxito de la identificación, tiempos de respuestas adecuados y la efectividad del sistema ante un crecimiento futuro del banco de datos.

Los procesos de búsqueda de huellas con mejores resultados forman parte de AFISs propietarios, cuya adquisición resulta muy costosa para el país. Además, el estudio y reutilización de los componentes destinados al proceso de búsqueda de huellas en sus bancos de datos se torna imposible, conociéndose solo el principio de funcionamiento de cada uno de ellos. Con este trabajo se pretende ofrecer como aporte práctico un sistema que permita distribuir el proceso de búsqueda de huellas dactilares en el banco de datos del AFIS del Departamento de Biometría del CISED para obtener mejores tiempos de respuesta en el proceso de identificación.

Estructura del documento

El presente documento se encuentra dividido en tres capítulos estructurados de la siguiente forma:

Capítulo 1: Fundamentación teórica, se describen los principales conceptos relacionados con el dominio del problema, se realiza un estudio del estado del arte sobre los AFISs y se analizan las principales metodologías, tecnologías y herramientas para darle solución al problema planteado.

Capítulo 2: Características del sistema, se presentan las fases de Planificación y Diseño definidas por la metodología XP para dar solución al problema planteado. Entre otros se define el modelo de dominio, se identifican las historias de usuario y los requisitos no funcionales, y se propone la arquitectura del sistema, así como el plan de iteraciones.

Capítulo 3: Implementación y prueba, se cumplen los planes trazados mediante la codificación de la solución propuesta, describiéndose elementos relacionados con ella, finalmente se realizan las pruebas de software al sistema.

Capítulo 1: Fundamentación teórica

1.1. Introducción

En este capítulo se abordan una serie de aspectos que fundamentan teóricamente y describen el dominio de esta investigación. Se realiza un estudio del arte sobre los AFISs en el mundo, en Cuba y en la Universidad, así como de las principales metodologías, tecnologías y herramientas, analizando sus características, ventajas y desventajas, buscando las más indicadas para la construcción de la solución a desarrollar.

1.2. Huella dactilar

Las huellas dactilares se encuentran en la parte posterior de los dedos de las manos como reproducción de la epidermis. Están constituidas por rugosidades que forman salientes y depresiones. Los salientes se denominan crestas papilares y las depresiones surcos interpapilares (o valles). Una cresta está definida como un segmento de curva y un valle es la región entre dos crestas adyacentes (Ver Figura 1)[1].

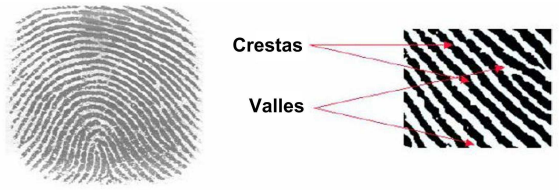


Figura 1: Crestas y valles de una huella dactilar.

1.3. AFIS

Los AFISs son los sistemas biométricos especializados en el reconocimiento de las huellas dactilares y sus inicios datan de la década del sesenta del pasado siglo, aunque la utilización de las huellas dactilares en ámbitos forenses se realizaba ya desde finales del siglo XIX por especialistas que examinaban dos pares de huellas de forma visual [2].

El proceso de identificación llevado a cabo por un AFIS está compuesto por un conjunto de etapas. Primero se adquiere la huella del sujeto que será identificado mediante un sensor especializado para ello. El paso siguiente es preprocesar la imagen con el objetivo de aumentar su calidad. Llegado a este punto se aplican algoritmos para la extracción de las características que serán comparadas (entre las características más utilizadas se encuentran las minucias, puntos singulares que presentan las crestas). El resultado de la comparación entre las características obtenidas con las características de otra huella determinará bajo un umbral si las dos muestras pertenecen al mismo individuo. Nótese que el algoritmo de comparación solo es responsable de decidir entre dos huellas un grado de semejanza, por lo que es el proceso de búsqueda en el banco de datos el encargado de obtener este resultado para

cada una de las huellas almacenadas que sean sometidas al análisis [3].

1.4. Procesamiento distribuido

En la actualidad los AFISs y los sistemas de información en general que necesitan manejar grandes cantidades de datos en tiempos cada vez menores, así como compartir recursos para llevar a cabo tareas cada vez más complejas, utilizan el procesamiento distribuido. Este tipo de enfoque se refiere a varias computadoras autónomas conectadas mediante una red de comunicaciones y equipadas con programas que les permitan coordinar sus actividades y compartir recursos [4]. Permite la ejecución de procesos paralelamente entre computadoras, que no tienen necesariamente por qué estar físicamente cercanas y que son independientes en cuanto a sus propios recursos, colaborando para la realización de una tarea que puede ser tan sencilla como distribuir la carga de trabajo entre procesos idénticos o tan compleja como distribuir multitud de procesos interdependientes. Los sistemas que emplean este procesamiento, y que comúnmente se les llama sistemas distribuidos, tienen un conjunto de características que constituyen requisitos para su diseño y desarrollo.

1.4.1. Transparencia

Un sistema distribuido es transparente cuando los usuarios y las aplicaciones pueden percibirlo como si fuese un único sistema de computación cuyos recursos no se encuentran distribuidos físicamente. Existen ocho tipos o grados de transparencia [5]: acceso, migración, recolección, replicación, persistencia, ubicación, concurrencia y frente a fallos. De ellos los tres últimos son medulares en un sistema distribuido.

La transparencia de ubicación permite ocultar la localización de los recursos de forma tal que puedan ser accedidos sin conocer su localización. La compartición de recursos por distintos usuarios sin que entre ellos mismo ocurra interferencia es denominada transparencia concurrente, mientras que la recuperación ante un fallo permitiendo que los usuarios continúen sus tareas sin que estos noten el estado del sistema es conocida como transparencia frente a fallos.

1.4.2. Escalabilidad

La escalabilidad de un sistema distribuido se refiere a su capacidad de continuar siendo efectivo cuando incrementa significativamente la carga de trabajo debido al aumento del número de usuarios y recursos [6]. Los sistemas distribuidos deben seguir funcionando con nuevas configuraciones en su entorno y en sus tareas, de forma tal que la atención a una petición sea independiente a los cambios a los que sean sometidos.

Existen diversas técnicas para lograr la escalabilidad como son la eliminación de la latencia en las comunicaciones, la distribución y la replicación [4].

1.4.3. Concurrencia

La concurrencia es una de las características más importantes presentes en un sistema distribuido, consiste en el número de procesos simultáneos activos, que en el peor de los casos será proporcional a la cantidad de procesadores centrales contenidos en el sistema. Está relacionada estrechamente con la consistencia de los datos que se procesan, dado que su objetivo es mantener siempre un entorno distribuido multiusuario [4].

1.4.4. Heterogeneidad

Todo sistema distribuido debe ser independiente a la diferencia y variedad de los distintos elementos que forman la red de computadoras sobre la que se ejecuta. La heterogeneidad es aquella característica orientada a la abstracción de un sistema sobre las redes, hardware, sistemas operativos, lenguajes de programación y las implementaciones en las que trabajan los diferentes desarrolladores [4].

Middleware

Existe un término fundamental que no puede dejar de ser mencionado para lograr que un sistema sea heterogéneo: middleware. Es una capa de software que ofrece una capa de abstracción para los programadores, ocultando la variedad existente en los elementos antes mencionados (Ver Figura 2). Provee un modelo computacional uniforme para la distribución, dado que permite la invocación de objetos remotos, notificación de eventos remotos, acceso a bancos de datos remotos y procesamiento distribuido de transacciones [7].

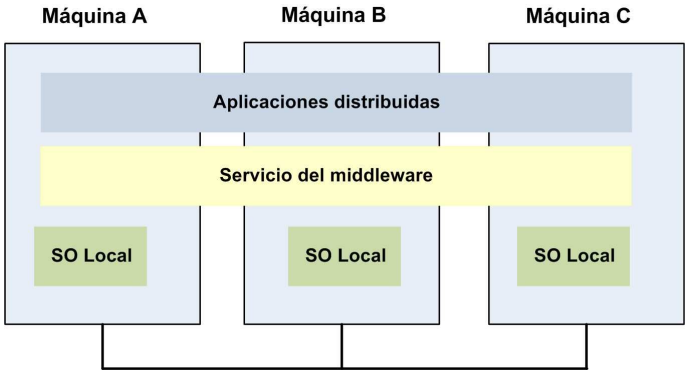


Figura 2: Localización de un middleware.

Existen varios tipos de middleware en dependencia de la integración que incluyen, o sea, teniendo en cuenta los protocolos de comunicación que utilizan o las formas de operar. Además, según las funciones deseadas para el problema que se pretenda resolver se necesitarán diferentes tipos de servicios de middleware [8]. Los middlewares se clasifican como: orientados a mensajes (MOM), basados en llamadas a procedimientos remotos (RPC) y basados en objetos [9].

Middleware Orientado a Mensajes (MOM)

Un Middleware Orientado a Mensajes (Message Oriented Middleware) es un intermediario en la comunicación entre aplicaciones distribuidas. Los mensajes, compuestos por cabeceras y datos, son el medio de integración utilizado por este middleware que brinda mecanismos para la creación, manipulación, almacenamiento y transmisión de los mensajes [10].

Las aplicaciones se comunican con los sistemas de mensajería a través de clientes provistos por el middleware, que brinda además, interfaces mediante las cuales las aplicaciones pueden enviar y recibir mensajes (Ver Figura 3) [11].

Figura 3: Comunicación basada en mensajes a través de un MOM.

El intercambio de información se realiza mediante canales virtuales llamados destinos, o sea, cuando se envía un mensaje no se envía a una aplicación específica sino a un destino determinado, mientras que las aplicaciones receptoras se dirigirán al mismo para obtener sus mensajes. Luego la comunicación establecida entre el emisor y el receptor es asíncrona pues las aplicaciones emisoras no necesitan confirmación de recepción, sino que siguen su procesamiento normal. La abstracción más significativa aportada por este tipo de middleware son las colas de mensajes, que se caracterizan por:

- Los receptores controlan cuándo procesar el mensaje. Además, no tienen que estar continuamente a la espera de mensajes, ni siquiera tienen que existir cuando se envía el mensaje.
- Las colas de mensajes pueden ser compartidas. Pueden ser leídas como en una pila FIFO (First In First Out) o teniendo en cuenta prioridades.
- A los mensajes puede asignárseles prioridades.

Las distintas implementaciones actuales de MOMs responden a diferentes arquitecturas, pero en esencia todas se basan en los clientes y el propio MOM, entendiendo como cliente cualquier aplicación que envíe o reciba mensajes a través del MOM.

Protocolo AMQP

AMQP (Advanced Message Queue Protocol) es un protocolo de estándar abierto en la capa de aplicaciones de un sistema de comunicación a partir del cual se puede lograr interoperabilidad entre MOMs. Su arquitectura está orientada a resolver problemas de procesamiento de datos mediante la abstracción

del problema de productores-consumidores, en donde los productores son los programas que envían mensajes y los consumidores aquellos que los reciben. Este protocolo está caracterizado por el manejo de mensajes y colas, el enrutamiento (puede ser punto a punto[1] o publicación-subscripción[2]), la exactitud y la seguridad [12].

Los mensajes son, básicamente, cualquier conjunto de bytes que se desee enviar; pueden ser una simple cadena ASCII (American Standard Code for Information Interchange), JSON (JavaScript Object Notation) o un objeto binario serializado. Los mensajes son enviados a colas, garantizando que sean entregados en el mismo orden en el que son enviados. Para recibir mensajes, la cola primeramente debe tener establecido un enlace o vínculo con un intercambiador, quien es el encargado de entregar los mensajes a las colas correspondientes. Parte de establecer la unión incluye especificar qué mensajes deben ser enviados a las colas (Ver Figura 4).

Figura 4: Entidades del estándar AMQP.

Tanto los mensajes, como las colas y los vínculos entre el intercambiador y las colas poseen propiedades configurables, lo que hace de AMQP un protocolo más flexible, al no tener una preconfiguración estática, así todos los recursos pueden ser creados y destruidos dinámicamente por los clientes. AMQP es un modelo simple que cuenta además, con disímiles librerías en diversos lenguajes de forma gratuita [13].

Algunas implementaciones de este protocolo son OpenAMQ, ApacheQpid y RabbitMQ, todas ellas proyectos de código abierto.

1.5. Estudio de sistemas de interés

Los AFISs han sido desarrollados como el componente principal que vincula el análisis de huellas dactilares y los sistemas informáticos. Entre los primeros trabajos relacionados con estos aspectos se destacan aquellos realizados por el Dr. Henry Faulds en 1880 cuando propuso por primera vez que los detalles de las crestas en las huellas dactilares eran únicos, y por tal motivo, las huellas podían ser clasificadas y utilizadas en la resolución de crímenes. Pero no sería hasta 1892 que se comprobaría científicamente la unicidad de las huellas dactilares, en el trabajo publicado por Sir Francis Galton titulado *Fingerprint* que constituyó un hito en el campo de la identificación [14].

En los finales del siglo XIX destaca también el estudio del militar británico Sir Edward Henry, quien desarrolló un método de identificación basado en la huella dactilar del pulgar y cuyos fundamentos fueron publicados en 1900 en su trabajo *Classification and Use of Fingerprints*. Dicho método se

[1]Los mensajes van dirigidos a un único receptor. El mensaje queda almacenado en una cola hasta que el receptor quiera o pueda consumirlo.

[2]Intervienen dos actores: los emisores de información y los subscriptores o consumidores de dicha información.

convirtió en un estándar de facto para la mayoría de los sistemas de identificación de todo el mundo, siendo asumido por el FBI (Federal Buro of Investigation) que se encargó de sustentarlo [14].

Los primeros AFISs que surgieron en el mundo lo hicieron sin el beneficio de estándares internacionales. Con el apoyo del NIST (National Institute of Standards and Technology), fundado por el FBI, se comenzó a trabajar en la fabricación e internacionalización de un estándar para la transmisión de imágenes de huellas dactilares. Dicho estándar fue adoptado por el ANSI (American National Standards Institute), para convertirse en el ANSI/NIST-CSL 1-1993 *American National Standard for Information Systems-Data Format for the Interchange of Fingerprint Information*, el cual se encuentra vigente en el estándar ANSI/NIST ITL 1-2007 y al que se le han hecho peticiones de actualización por parte de diversos organismos tales como la Interpol y el FBI.

Actualmente se discute la aprobación de un nuevo estándar, *Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information*, referenciado como ANSI/NIST-ITL 1-2011, cuyas versiones previas han sido usadas por los Estados Unidos y extendidas a muchos países para proveer un lenguaje común en el intercambio de datos biométricos y metadatos asociados a los mismos. En este nuevo estándar se han incorporado nuevos registros de tipos existentes, así como ADN y huellas plantares (de pies) [15].

Con los estándares desarrollados y debidamente documentados, la principal barrera de los AFISs es el costo de su compra y mantenimiento. La organización americana que se encarga del desarrollo de la historia criminal nacional NCHIP (National Criminal History Improvement Project), ha inyectado al negocio de los sistemas de investigación millones de dólares desde el año 1995. Esta infusión de capital ha creado grandes oportunidades de negocio, concediendo la demanda de contratos multimillonarios que requieren una significativa inversión de capital para investigación y desarrollo (I+D). Compañías como Sagem Morpho, Printrak Motorola, NEC, Lockheed Martin y Cogent Systems han emergido como grandes contratistas para aplicaciones de uso civil y criminalístico. Compañías tales como AWARE, AFIX Tracker, Comnetix y FORAY se encuentran entre los principales mercados consumidores [14].

1.5.1. Soluciones existentes en el mundo

Entre las principales potencias en el mundo que desarrollan tecnologías relacionadas con los AFISs se encuentran Estados Unidos y Francia. A continuación se describirán los AFISs de mayor prestigio y con mejores resultados en el mercado internacional; nótese que todos son comercializados bajo licencias propietarias y en la mayoría de los casos demandan altas prestaciones de hardware, además de costosos mantenimientos de software que actualmente solo pueden ser pagados por las mayores agencias del mercado.

IAFIS

IAFIS (Integrated Automated Fingerprint Identification System), es el sistema nacional de identifica-

ción de huellas dactilares e historia criminal mantenido por el FBI. IAFIS posee el banco de datos biométrico más grande del mundo con cerca de 70 millones de historias criminales, 31 millones de huellas dactilares de civiles y 73 mil de conocidos terroristas o sospechosos de actos delictivos que han sido procesados por el sistema americano u otras agencias internacionales. El tiempo de respuesta promedio de este sistema para solicitudes criminales es de 27 minutos, mientras que para civiles se comporta con un tiempo medio de una hora y 12 minutos. IAFIS procesa más de 61 millones de solicitudes de huellas dactilares latentes durante un año [14].

El FBI pretende suplantar IAFIS por el NGI (Next Generation Identification System) desarrollado por el grupo Safran en asociación con la empresa Lockheed Martin. NGI no es más que una masiva actualización por 1 000 millones de dólares que contendrá escaneos de iris, fotos investigables con tecnología de reconocimiento facial, huellas palmares, mediciones de modos de andar y grabaciones de voz junto a registros de huellas digitales, cicatrices y tatuajes. La tecnología que usará este sistema está suministrada por MorphoTrak, un reconocido líder mundial en identificación de huellas dactilares, la cual ha sido escogida luego de un largo proceso de estudio de mercado que evaluó las soluciones presentadas por las principales empresas en el campo, para seleccionar la mejor tecnología biométrica. La solución técnica de MorphoTrak está basada en sus poderosos algoritmos de comparación y una arquitectura abierta [16].

ExpressID AFIS

ExpressID AFIS es un software para la identificación mediante huellas dactilares a gran escala desarrollado por la compañía Innovatrics, cuyos locales comerciales se encuentran en Francia y Eslovaquia. Este sistema ha sido integrado con otras aplicaciones de identificación para diversos clientes como gobiernos, empresas, bancos, hospitales, entre otros.

Comparado con otros AFISs del mercado ExpressID AFIS ha demostrado [17]:

- Rapidez: es capaz de comparar hasta 85 millones de huellas por segundo con un solo ordenador.
- Más exacto: ejecuta una búsqueda exhaustiva en el banco de datos sin usar técnicas de clasificación o indexación.
- Menos costoso: el sistema se ejecuta en el estándar de hardware presente en la mayoría de los ordenadores actuales sin recurrir a altos costos para adquisición de servidores y otras tecnologías.

El paquete que comprende ExpressID AFIS contiene todas las partes necesarias de un complejo sistema de identificación: un motor de alta velocidad, un administrador de la búsqueda en el banco de datos y una arquitectura cliente-servidor. Presenta una arquitectura escalable, permitiendo a los componentes del servidor de comparación ser instalados en varios ordenadores para distribuir la carga de trabajo e incrementar la velocidad de comparación, siendo capaz de realizar búsquedas en bancos de datos de millones de huellas dactilares [17].

ExpressID AFIS ha tenido resultados muy importantes en el mercado, alcanzando premios a nivel

internacional como son [18]:

- Año 2005: ExpressID AFIS obtiene el primer lugar en escáneres ópticos de huellas dactilares durante la FVC[3] de 2004.
- Año 2007: ExpressID AFIS obtiene el primer lugar en escáneres de 500 dpi (Dots per Inch) durante la FVC de 2006.
- Febrero 2009: ExpressID AFIS fue reconocido por el evento de Innovación Tecnología Global acerca de huellas dactilares del año 2008. Los analistas de Frost & Sullivan reconocieron no solo la tecnología, sino también los beneficios del cliente, la cuota de mercado y la estrategia de la compañía Innovatrics.
- Año 2011: ExpressID AFIS ANSI & ISO SDK (Software Development Kit), recibe un certificado del NIST que prueba el rendimiento de su algoritmo con plantillas ANSI-378.

1.5.2. Soluciones existentes en Cuba

Cuba ha incursionado también en el área de la identificación y en específico en el desarrollo de AFISs. Entre las instituciones que se destacan en estas actividades pueden mencionarse DATYS, Tecnologías y Sistemas, fundada en el 2005 para la producción de bienes y servicios informáticos, con una división destinada al desarrollo de software biométrico, Biomesys SUITE. Por otro lado en la UCI el CISED, fundado en 2008, posee un Departamento de Biometría que se especializa de forma moderada al estudio, investigación y desarrollo de sistemas biométricos. Seguidamente se describen los productos AFISs creados en estas instituciones.

BIOMESYS AFIS

BIOMESYS AFIS es el Sistema Cubano de Identificación Dactiloscópica desarrollado a partir de 2006 por Biomesys SUITE, que ha ahorrado al país más de 10 millones de dólares y ha sido un paso de avance en su soberanía tecnológica [19].

Este sistema responde a solicitudes de identificación, inserción, autenticación, actualización y recuperación de datos utilizando el formato ANSI/NIST ITL-1-2000 y el protocolo SMTP. Ha demostrado una efectividad de un 99,5 % con valores de FAR[4] y FRR[5] por debajo del 1 % [20].

El BIOMESYS AFIS está compuesto por dos componentes fundamentales. El primero de ellos es un AFIS Civil que es el centro modular del sistema para resolver y certificar las solicitudes. Proporciona la intercomunicación con otros sistemas a través de una interfaz basada en estándares internacionales reconocidos para la codificación y encriptación de información mediante el protocolo ANSI/NIST.

[3] Acrónimo de Fingerprint Verification Competition, competencia mundialmente reconocida que es patrocinada por la Universidad de Bologna, Italia, en ella se prueban múltiples sistemas de alto nivel en el mundo.

[4] Acrónimo de False Accepted Rate, es el error que se produce cuando el sistema indica que la información adquirida del usuario en la entrada sí se corresponde con la plantilla almacenada, cuando realmente se trata de otra persona.

[5] Acrónimo de False Reject Rate, es el error que se produce si el sistema indica que el usuario en la entrada no se corresponde con la plantilla almacenada, y realmente sí es la misma persona.

Mientras que el otro componente está integrado por módulos biométricos para las estaciones de registro que permiten la captura de datos alfanuméricos y de las impresiones dactilares con control de la calidad.

El tipo de búsqueda y comparación dependen del tipo de solicitud que recibe el sistema: 1:N si se requiere una identificación y de 1:1 si es requerida una autenticación, verificación o una actualización. Las características físicas de cómputo necesarias para estas solicitudes son adaptables en dependencia del alcance del sistema, que puede cubrir un simple control de acceso biométrico hasta sistemas de identidad nacional que impondrían grandes capacidades de cálculo y almacenamiento [20].

El AFIS Civil está basado en una arquitectura de servidores que incluye clústeres de banco de datos, Web, correo electrónico, procesamiento NIST y búsqueda. De ellos el clúster de búsqueda y comparación es el elemento principal, basándose en un procesamiento distribuido. Este clúster tiene una infraestructura en la que se incluyen un servidor que distribuye la búsqueda (master), un servidor de contingencia (spare) y n servidores de comparación.

La solución a las solicitudes de búsqueda se realiza de forma paralela íntegramente en memoria RAM en cada servidor de comparación, en los cuales está segmentado el banco de datos; con los resultados arrojados por cada uno de ellos se conforma una respuesta.

El propio diseño utilizado para el proceso de búsqueda ofrece las posibilidades de:

- Explotación desde estaciones remotas de búsqueda.
- Búsqueda automática al introducir impresiones en el banco de datos.
- Escalable según las necesidades de búsquedas simultáneas.
- Alto nivel de tolerancia a fallos tanto por hardware como por software.

La velocidad de comparación que se alcanza es superior al millón de comparaciones por segundo por CPU de procesamiento y con menos de un 1 % de penetración [20].

El uso de BIOMESYS AFIS se realiza mediante una licencia por cantidad de personas a enrolar en el AFIS. Debe señalarse que entre los requerimientos de software para la instalación del sistema se encuentra Oracle Database Enterprise Edition 11 G R2, para lo cual es necesario comprar dos licencias propietarias. Además, en igual medida a los AFIS internacionales descritos anteriormente, precisa de altos requerimientos de hardware.

El CISED en el desarrollo de AFIS

El CISED es un centro destinado a la producción de sistemas informáticos cuyas características fundamentales sean la seguridad y calidad. Para ello necesita un AFIS que no solo pueda ser comercializado como un producto independiente, sino que también pueda ser embebido en sus soluciones informáticas, y en específico en sistemas de control de acceso. La variante de AFIS desarrollada allí con este objetivo, aunque no constituye aún un sistema robusto y con el nivel impuesto en el mercado internacional, automatiza los procesos de identificación y enrolamiento, pero debe señalarse que no brinda muchos de

los servicios estandarizados para estos sistemas.

Esta variante realiza 8 mil comparaciones por segundo en un ordenador con propiedades físicas tales como un procesador Dual Core de 1 GB de RAM. Utiliza para el reconocimiento de huellas dactilares el kit de herramientas SourceAFIS, cuyas características son descritas a continuación.

SourceAFIS 1.7

SourceAFIS 1.7 es la versión más actualizada, hasta la fecha de escritura de este documento, del kit de herramientas de reconocimiento de huellas dactilares. Posibilita la extracción de plantillas, la verificación (1:1), identificación (1:N) que incluye búsqueda exhaustiva utilizando un algoritmo de comparación basado en minucias.

Otras de sus características son el procesamiento de imágenes, la extracción múltiple de huellas dactilares y el enrolamiento. Soporta librería compactada, plantillas ISO/IEC 19794-2:2005[6] y XML (eXtensible Markup Language).

Posee compatibilidad con las plataformas .NET, .NET CF (Compact Framework), actualmente obsoleto, y Java en las versiones posteriores a la 6.0, aunque para este caso en específico se encuentra en estado experimental de desarrollo. En cuanto a sistemas operativos es soportado por Windows para las versiones desde XP hasta Windows 8 y Linux para 32 y 64 bit. Presenta SDK para el desarrollo en lenguajes como C#, Java, VB.NET y ASP.NET. Ofrece una API (Applications Programming Interface) simple e incorpora documentación en C# con referencias para la API y ejemplos en consola; mientras que para Java aunque se encuentra en proceso experimental presenta también documentación.

El rendimiento de SourceAFIS 1.7 arroja los siguientes resultados [21]:

- Tiempo de extracción: 180 ms, este resultado solo es obtenido en .NET, pues la extracción de características solo es posible hasta el momento en esta plataforma.
- Velocidad de comparación: 10 mil huellas por segundo, este resultado es alcanzado en .NET
- Exactitud: (FVC-OnGoing[7])
  - 3,6 % EER[8], 10,9 % FRR, 0,01 % FAR en comparaciones 1:1.
  - 1,17 % EER, 2,5 % FRR, 0,01 % FAR en pruebas ISO.

[6]Especifica los conceptos y formatos de datos para la representación de huellas dactilares utilizando la noción fundamental de minucias.

[7]Sistema basado en la evaluación automatizada de algoritmos de reconocimiento de huellas dactilares. Las pruebas se llevan a cabo con una serie de conjuntos de datos y los resultados se reportan en línea mediante el uso de indicadores y métricas de resultados bien conocidos.

[8]Acrónimo de Equal Error Rate, estadística utilizada para mostrar el rendimiento biométrico, es la ubicación en una curva ROC (Característica de funcionamiento del receptor) o DET (Compensación por error de detección) donde la tasa de falsa aceptación y la tasa de falso rechazo son iguales.

1.6. Metodología de desarrollo de software

Una metodología de desarrollo de software es una filosofía o marco de trabajo para estructurar, planificar y controlar el proceso de desarrollo de software. Especifica un conjunto de pasos o procedimientos que definen quién debe hacer qué, cuándo y cómo debe hacerlo. Su objetivo principal es guiar a los desarrolladores de software en la obtención de un producto de calidad que cumpla con los requerimientos establecidos por el cliente ajustándose a los recursos apropiados y a un costo razonable [22].

Cada software a desarrollar implica condiciones muy diversas, debido a ello existen disímiles metodologías para guiar su creación, las cuales pueden ser agrupadas en metodologías pesadas o tradicionales y en metodologías ligeras o ágiles.

1.6.1. Metodologías pesadas

Las metodologías pesadas están orientadas al control de procesos. Se caracterizan por establecer estrictamente las tareas, la definición de roles, los artefactos que se deben obtener, así como las herramientas y notaciones a utilizar. Además, implican una detallada planificación y exhaustiva documentación. Son muy efectivas para proyectos de desarrollo de software complejos, aunque su aplicación en entornos volátiles no es recomendada [22].

Rational Unified Process (RUP)

RUP es un proceso de desarrollo de software, que integra los elementos esenciales del ciclo de vida del software, permitiendo ser adaptado a las características específicas de cada organización [23], aunque está especialmente diseñado para llevar a cabo grandes y complejos proyectos.

En el ciclo de vida de RUP las distintas actividades se han organizado en nueve flujos de trabajo bien definidos. De ellos seis son llamados flujos de ingeniería y los restantes tres, flujos de apoyo. Los flujos de trabajo son: Modelamiento del negocio, Requerimientos, Análisis y diseño, Implementación, Prueba, Instalación, Administración del proyecto, Administración de configuración y cambios, y Ambiente (Ver Figura 5).

RUP está dividido en cuatro fases: Inicio, Elaboración, Construcción y Transición. En la primera de ellas el objetivo esencial es delimitar el alcance del proyecto. También se explora el problema, lo cual permite describir el negocio y determinar los casos de uso críticos en él.

La fase de Elaboración está centrada a la definición de la arquitectura [24] mediante la comprensión de los requisitos funcionales y no funcionales identificados en la fase previa.

Obtener una versión del producto lista para ser usada y documentada es la finalidad principal de la fase de Construcción (todos los requisitos son implementados, integrados y probados), siendo esta fase la más prolongada de todas y en la que se pone a consideración de los usuarios el producto final.

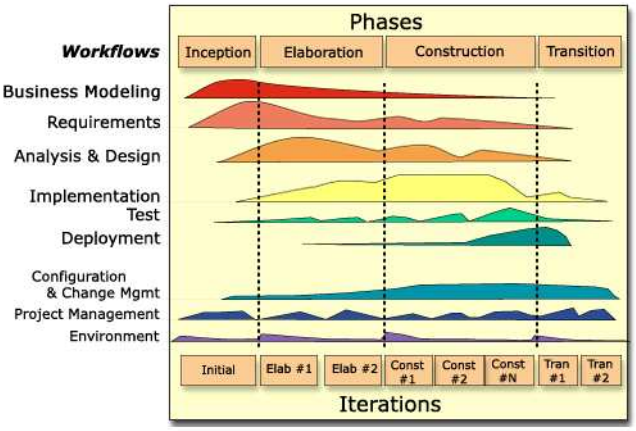


Figura 5: Relación entre las fases y flujos de trabajo de RUP.

La fase de Transición, como indica su nombre, está basada en la transición de la versión final del sistema, del entorno de desarrollo a las manos de los usuarios. Para ello es necesaria la corrección de los últimos errores, la completitud de la documentación y el entrenamiento del usuario en su interacción con el software [25].

El ciclo de vida de RUP tiene tres características fundamentales:

- Dirigido por casos de uso: Los casos de uso representan las funcionalidades del software que el usuario final necesita, los cuales son recogidos en el modelado del negocio y posteriormente transformados en requisitos funcionales. Constituyen la línea principal de todas las actividades que se realizarán a partir de ese momento, pues todos los artefactos resultantes de los flujos de trabajos representan su realización [26].
- Centrado en la arquitectura: Todos los implicados en el proceso de desarrollo de software deben tener una visión clara de la arquitectura. La arquitectura es el conjunto de sistemas, reglas, estándares, convenciones y procesos que entrelazados representan la base para la comprensión y construcción del sistema.
- Iterativo e incremental: Cada fase del ciclo de vida de RUP se realiza mediante iteraciones. Las iteraciones comprenden actividades de todos los flujos de trabajo que tienen como resultado un incremento o crecimiento en el producto software y un refinamiento de la arquitectura [26].

1.6.2. Metodologías ágiles

Las metodologías ágiles están enfocadas al factor humano, la estrecha comunicación con el cliente y al producto software. Se caracterizan por desarrollar el software de forma incremental y en iteraciones muy cortas, con la premisa de que es más importante su funcionamiento a escribir documentación exhaustiva. Su utilización es recomendada en entornos volátiles, puesto que la prioridad es responder a los cambios más que seguir estrictamente un plan. Además, resultan muy efectivas en proyectos pequeños, *"(. . .) aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas*

esenciales para asegurar la calidad del producto." [27].

Extreme Programming (XP)

XP guía el desarrollo de software haciendo énfasis en las relaciones interpersonales, fomentando el trabajo en equipo y la estrecha comunicación con el cliente. El producto software se construye teniendo en cuenta que la solución más simple es la mejor. Está orientada a la adaptación paulatina de los requisitos y de sus cambios en cualquier punto de la vida del proyecto.

El ciclo de desarrollo en XP es iterativo e incremental (Ver Figura 6); consiste en seis fases: Exploración, Planificación de la entrega, Iteraciones, Producción, Mantenimiento y Muerte del proyecto.

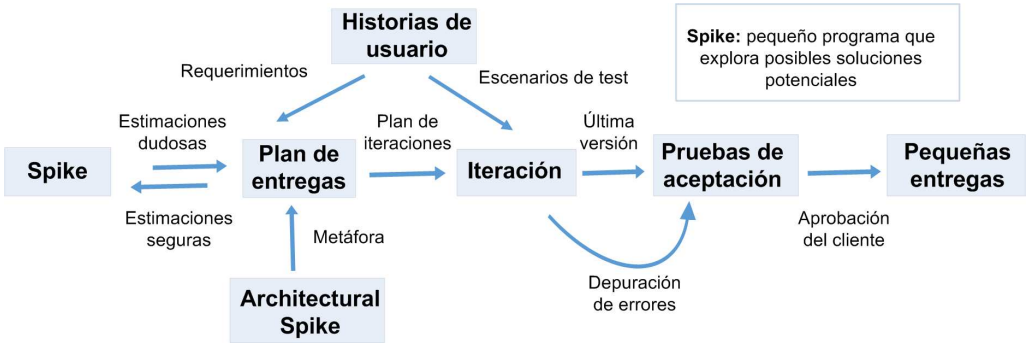


Figura 6: Flujo de trabajo en XP.

Está fundamentado no solo en la asunción de un conjunto de buenas prácticas en el desarrollo de software, sino en la sinergia obtenida de su aplicación conjunta y equilibrada. Entre las prácticas más importantes se pueden mencionar la codificación por parejas posibilitando que el código sea discutido y revisado mientras se escribe, pequeñas entregas del sistema en forma de versiones operativas aunque no incluyan todas las funcionalidades, diseño simple, dirección de las pruebas unitarias en la codificación, refactorización del código dirigida a simplificarlo para facilitar sus cambios en el futuro, presencia continua del cliente en la producción y utilización de estándares de programación, entre otras [27].

Durante todo el desarrollo está presente la retroalimentación entre el cliente y el equipo de desarrollo. El cliente guía el trabajo siempre hacia los elementos de mayor valor en el negocio, o sea, sus prioridades y restricciones, mientras que los programadores estiman el esfuerzo necesario para su codificación. Esta unión es la clave del éxito de XP, pues es el factor que posibilita asumir los cambios sin mayores dificultades [28].

Scrum

Scrum es una metodología de desarrollo de software iterativa e incremental. Define un marco para la gestión de proyectos orientado a qué construir y en qué orden hacerlo; debido a ello se utiliza en otras prácticas de ingeniería de software tales como RUP o XP. Es muy sencilla y responde a los principios de inspección constante y continua, adaptación del producto a las necesidades del cliente en tiempo real e innovación [29].

Tiene como meta fomentar la efectividad y productividad de los equipos. Entre sus principales características destacan los ciclos de desarrollo llamados sprints con una duración de 30 días, cuyo resultado es un incremento funcional del producto. Establece reuniones durante todo el proyecto, de ellas las más importantes son las diarias, en las cuales los miembros del equipo disponen de 15 minutos para chequear el trabajo realizado hasta la fecha y las previsiones para el día siguiente [30].

1.7. Herramientas y tecnologías para la solución del problema

Las herramientas y tecnologías son vitales en el desarrollo de software al organizar, facilitar, agilizar y automatizar el proceso. De la correcta selección, el buen uso y el dominio que tenga el equipo de desarrollo sobre ellas dependen en gran medida la duración del proyecto, la completa utilización de sus ventajas y por último, pero no menos importante, la calidad con que transcurre el proceso de desarrollo de software y del producto final.

1.7.1. Lenguaje de modelación

El lenguaje de modelado es vital para el diseño y posterior construcción del producto software. Es un conjunto estandarizado de notaciones que incluye símbolos y las distintas formas de organizarlos, estructurarlos y disponerlos lógicamente.

Unified Modeling Language (UML)

UML es un lenguaje de modelación que unifica los mejores resultados de pasadas y nuevas técnicas de modelado de forma estándar. Permite especificar, visualizar, construir y documentar un sistema. UML incluye los conceptos necesarios para detallar los procesos de negocio, las funcionalidades del sistema y otros elementos más específicos como esquemas de bancos de datos, expresiones de lenguajes de programación, plataformas de desarrollo y sistemas reutilizables.

Es fundamental aclarar que UML no es un lenguaje de programación, sino un lenguaje de modelado de propósito general, que ha demostrado su efectividad en el área del análisis y diseño de sistemas de cómputo [31].

1.7.2. Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering), cuyas siglas traducidas al español significan Ingeniería de Software Asistida por Computadora, incluyen métodos, técnicas, utilidades y documentación orientadas a la automatización del ciclo de vida del software [32]. Son fundamentales en la disminución del tiempo de desarrollo del software y en el aumento de la productividad, dado que permiten enfatizar en el análisis y el diseño para minimizar el esfuerzo de codificación y prueba.

Rational Rose Enterprise

Rational Rose es una herramienta CASE desarrollada y mantenida por Rational Corporation.

Está orientada a objetos sobre la base de UML, facilita el proceso de modelado con un número significativo de estereotipos predefinidos y permite además, la generación de la documentación del software. Esta herramienta posee un mecanismo para generar el código de las clases definidas en el diseño UML, aunque no soporta varios lenguajes de programación [33].

Visual Paradigm

Visual Paradigm es una herramienta CASE multiplataforma que incluye UML y soporta el ciclo de vida completo del desarrollo de software. Permite el modelado visual del software con el paradigma orientado a objetos. Entre sus características más significativas se encuentran la ingeniería directa e inversa, la modelación de todos los tipos de diagramas de clases y la generación de documentación en varios formatos [34].

Es importante destacar que brinda la facilidad de generar diagramas con las descripciones de casos de uso [35].

1.7.3. Lenguaje de programación

El lenguaje de programación es el intermediario en la comunicación entre el humano y un ordenador, mediante la creación de programas cuyo fin es representar algoritmos precisos que controlen el comportamiento de los dispositivos de hardware y de software. Agrupa símbolos, instrucciones, y reglas sintácticas y semánticas que expresan su estructura. Sus expresiones ayudan en la escritura, prueba y mantenimiento del código fuente, siendo fácilmente escritas y leídas por personas en el proceso de programación [36].

CSharp (C#)

C# es un lenguaje de programación orientado a objetos desarrollado por Microsoft específicamente para ser utilizado en su plataforma .NET, es por ello que desarrollar empleando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes con los que se puede codificar en esta plataforma. Aunque en la actualidad ya se ha implementado un compilador para generar programas codificados con C# en otras plataformas como Unix, Mac OS y GNU/Linux.

Está basado en las experiencias adquiridas de los lenguajes anteriores a él. Su sintaxis básica proviene de C y C++, y combina las mejores cualidades de un lenguaje de programación de alto nivel, en la misma medida en que brinda un gran rendimiento. Es un lenguaje que posee muchas semejanzas con Java como son el estándar de código y el tipado seguro y unificado [37].

Java

Java es un lenguaje de programación portable sobre diferentes plataformas que fue desarrollado originalmente por la compañía Sun Microsystems (la cual fue adquirida por la compañía Oracle en 2010). Es un lenguaje de propósito general, orientado a objetos, basado en clases y concurrente, cuyo diseño

está guiado a tener pocas dependencias de implementación y su sintaxis se deriva de C++. Integra librerías estándar para interfaces de usuario, objetos distribuidos, hilos de ejecución, XML, web, móviles, tv, entre otras.

Las aplicaciones escritas en Java hacen un uso intensivo de los recursos del ordenador, como son memoria y procesador. Además, la ejecución de estas aplicaciones es lenta cuando se realizan cálculos matemáticos complejos o el diseño contiene gran cantidad de elementos visuales [38].

1.7.4. Entorno de desarrollo integrado (IDE)

Un entorno de desarrollo integrado o IDE (Integrated Development Environment), por sus siglas en inglés, es un programa que incluye un conjunto de herramientas mediante las cuales los programadores pueden codificar [39]. Entre las herramientas que componen un IDE están un editor de texto, un compilador, un intérprete, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario (GUI) y, opcionalmente, un sistema de control de versiones [40].

Netbeans

Netbeans es un producto libre y gratuito sin restricciones para su uso comercial o no comercial. Es un proyecto fundado y patrocinado por Sun Microsystems y apoyado por una gran comunidad de usuarios. Se distribuye bajo las licencias CDDL (Common Development and Distribution License) y GNU GPL (General Public License) en su versión 2 [41].

Este IDE multiplataforma está orientado a la modularidad, o sea, todas sus funciones son provistas por módulos (plugins). Las aplicaciones desarrolladas a partir de módulos pueden extenderse adicionándole nuevos módulos que pueden ser codificados independientemente, de esta forma otros programadores pueden extender las aplicaciones desarrolladas en Netbeans. Debido a los módulos es posible utilizar varios lenguajes como C/C++, Java y Python, entre otros.

Con Netbeans es posible crear todo tipo de aplicación Java, así como aplicaciones de escritorio, web, móviles, entre otras. Brinda herramientas de desarrollo visuales, de esquemas y modelados UML [42]. En la actualidad Netbeans es muy popular entre muchos desarrolladores tecnológicos, la propia Sun Microsystems lo ha utilizado para realizar algunos de sus productos [43].

Visual Studio .Net

Visual Studio es un entorno de desarrollo solamente para los sistemas operativos Windows. Con este IDE es posible crear aplicaciones de escritorio, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET y aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles. Soporta varios lenguajes de programación tales como C++, C#, J#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros [44].

Su interfaz es limpia y personalizable. Brinda herramientas que facilitan todo el proceso de desarrollo de software simplificando procesos tales como el diseño visual de interfaces, la implementación, depuración y prueba. Incluye mecanismos de colaboración para la administración del ciclo de vida de las aplicaciones, chequeando en tiempo real el estado de los proyectos [45].

Entre sus principales características se encuentran completamiento de código, resaltado de sintaxis, refactorización y control de versiones, entre otras.

1.7.5. Servidor distribuidor de mensajes

Los servidores distribuidores de mensajes son las implementaciones de los MOMs, que como se explicó en la sección 1.4.4, permiten a las aplicaciones comunicarse mediante el intercambio de mensajes.

RabbitMQ

RabbitMQ es un servidor de mensajería multiplataforma escrito en el lenguaje de programación Erlang, líder en la implementación completa del protocolo AMQP. Es un proyecto de código abierto bajo la licencia Mozilla Public License versión 1.1 y comercialmente soportado por Rabbit Technologies Ltd (RTL). Aprovecha no solo las potencialidades de AMQP sino también las de OTP (Open Telecom Platform), uno de los sistemas más utilizado en el mundo por las empresas de telecomunicaciones. O sea, este servidor combina la robustez y escalabilidad de OTP con la flexibilidad del modelo de mensajería de AMQP.

Las librerías de OTP Erlang le proporcionan a RabbitMQ una base sólida para la construcción de software fiable. Por ejemplo el reconocido proyecto OpenStack lo emplea para controlar el envío de mensajes en sus diferentes servicios [46].

RabbitMQ permite alcanzar una alta escalabilidad y disponibilidad con la creación de un corredor virtual. Para ello varios nodos son clusterizados con el fin de replicar y compartir información.

Con este servidor una sola cola es capaz de almacenar cerca de 30 mil msj\seg[9], dependiendo de la cantidad y carga de los clientes, así como de las propiedades y tamaño de los mensajes. Ofrece la posibilidad de escribir mensajes en el disco duro para liberar la presión sobre la memoria, lo que aumenta la capacidad de almacenamiento de mensajes y el crecimiento de las colas, limitados solo por la capacidad del disco y no por la memoria RAM. Colas de hasta 10 millones de mensajes pueden ser acomodadas fácilmente. Además, permite el manejo de mensajes que son mayores que 1/4 de la memoria RAM instalada [47].

El monitoreo y control de todos los aspectos del corredor es muy sencillo en RabbitMQ, debido a que cuenta con una interfaz web para ello. Además, es posible extenderlo con una gran variedad de plugins ofrecidos por su equipo de desarrollo y por los propios usuarios [48].

[9] msj\seg: mensajes que son transmitidos por segundo.

ApacheQpid

ApacheQpid es un servidor de mensajería que implementa el protocolo AMQP completamente en dos corredores, uno escrito en C++ y otro en Java. Es un proyecto de la Apache Software Foundation (ASF) de código abierto que se distribuye bajo la licencia Apache License 2.0 [49]. Dispone de librerías para los clientes de C++, Java, .Net, Python y Ruby, los cuales son interoperables con los dos corredores.

Al igual que otros productos de Java este servidor es altamente configurable. Ha comenzado a ser reconocido en la actualidad, pues incluso la empresa Microsoft lo ha escogido para contribuir con un proyecto de código abierto, ayudando a esta comunidad a desarrollar la integración con la plataforma .NET. Otro ejemplo notable de su uso son los resultados logrados por el corredor Qpid C++ en el producto de Red Hat MRG, demostrando de 760 mil a 6 millones msj\seg con altas prestaciones de hardware [50].

Las cifras de latencia de mensajes son de aproximadamente 3 ms a 5 ms con un 1 Gbit TCP y de 18 ms a 22 ms con 10 Gbit TCP. La velocidad del servidor en modo no persistente oscila en los 5 000 msj\seg, mientras que en modo persistente se observan cifras de 1 100 msj\seg [50].

El corredor de Java soporta JMX (Java Management eXtensions) y proporciona un plugin para Eclipse y una herramienta para la ejecución de comandos que permitan la interacción con JMX en las tareas de monitoreo y control. El corredor de C++ tiene otras vías de control con mayor soporte.

OpenAMQ

El servidor OpenAMQ es un servidor de mensajería que implementa parcialmente el protocolo AMQP, pues deshecha funcionalidades del protocolo como las correspondientes a la mensajería JMS (Java Message Service). Es mantenido por una pequeña, pero activa comunidad que incluye a usuarios y al equipo iMatix. Es un software libre que se distribuye bajo los términos de la licencia GNU GPL según es publicada por la Free Software Foundation, ya sea la versión 2 de la licencia o cualquier versión posterior. Es portable en Linux, Windows, Solaris, y en otros sistemas Unix.

Su desempeño es de 130 mil msj\seg en la pila de una aplicación cliente y de 600 mil msj\seg en un corredor sostenido por un día de trabajo completo. Su latencia de mensajes es de aproximadamente 300 ms. En pruebas realizadas se ha registrado la transferencia de 600 KB por segundo en dependencia del hardware utilizado, debe tenerse en cuenta que esta cifra es del rendimiento total del corredor, es decir, para todos los mensajes entrantes y salientes del corredor [51].

La administración de los distintos elementos del corredor se realiza mediante una consola de comandos.

1.7.6. Gestores de bancos de datos

Un Sistema Gestor de Bancos de Datos (SGBD) permite procesar, describir, administrar y recuperar los datos almacenados en un banco de datos. Estos sistemas brindan un conjunto de programas, pro-

cedimientos y lenguajes para consultar los datos y realizar acciones sobre ellos, garantizando además, la seguridad de los mismos [52].

PostgreSQL

PostgreSQL es un SGBD objeto-relacional disponible en un amplio rango de plataformas y que se distribuye bajo la licencia BSD (Berkeley Software Distribution) de software libre. Es mantenido por la organización PostgreSQL Global Development Team y por la colaboración de una amplia comunidad de usuarios y programadores denominada PGDG (PostgreSQL Global Development Group). Es en la actualidad el SGBD de código abierto más potente del mercado [53]. Disímiles son las empresas y compañías que construyen sus productos software utilizando este gestor, entre ellas Debian, Apple, Red Hat y Sun Microsystems [54].

Cumple con los estándares SQL de interoperabilidad y compatibilidad, destacándose por su rendimiento, seguridad y alta disponibilidad [55]. Utiliza la tecnología MVCC (Multi-Version Concurrency Control) para conseguir una mejor respuesta en ambientes de grandes volúmenes de datos e implementa el modelo cliente-servidor.

Entre las principales características de PostgreSQL se encuentran el uso de procedimientos almacenados, el soporte de integridad referencial y manejo de distintos tipos de datos. Brinda una API flexible para programar en C/C++, Java y Python, entre otros.

MySQL

MySQL es un SGBD relacional que posee una licencia GNU GPL y una licencia comercial para aquellas empresas que quieran incorporarlo en productos privativos. Es ampliamente conocido y utilizado por su simplicidad y notable rendimiento y es además, muy confiable en términos de estabilidad [56]. La empresa MySQL AB, además de ser la encargada de la venta de licencias privativas y poseer el copyright de la mayor parte del código, es la que ofrece el soporte [57]. Según cifras del fabricante, en la actualidad existen más de seis millones de copias de MySQL funcionando y cuenta con usuarios de renombre como Google, Yahoo!, Nokia y Wikipedia [58].

Este gestor es multiplataforma y tiene un diseño multihilo, lo que le permite soportar una carga considerable de manera eficiente. Entre sus principales características se encuentran que es multiusuario, cuenta con APIs que posibilitan a aplicaciones escritas en diversos lenguajes de programación acceder a los bancos de datos MySQL, incluyendo, entre otros, C++, Java y Python; uso de procedimientos almacenados y vistas actualizables, gestión de usuarios y contraseñas. Soporta una gran cantidad de tipos de datos.

1.8. Selección de la metodología y el marco de trabajo

El sistema a desarrollar para la distribución del proceso de búsqueda de huellas dactilares en el banco de datos de un AFIS pertenece al Departamento de Biometría del CISED. Este centro especifica las herramientas, metodologías y tecnologías para el desarrollo de sus productos informáticos, teniendo en cuenta el estado actual y el creciente avance de estos elementos. Seguidamente se describe y fundamenta la selección del marco de trabajo para el desarrollo del sistema, según las líneas definidas por el centro y las decisiones propias del equipo de proyecto después del análisis de los mejores candidatos en cuanto a metodologías, herramientas y tecnologías.

La metodología RUP es ventajosa para el análisis, diseño, implementación, soporte y documentación de sistemas informáticos orientados a objetos, como es el caso del sistema a desarrollar. Es muy organizada y genera una documentación robusta y detallada de todo el proceso en general. Al estar pensada para llevar a cabo grandes y complejos proyectos, donde el equipo de desarrollo suele ser numeroso, no se ajusta ni a la solución que propone esta investigación, la cual no se considera un proyecto de grandes dimensiones, ni a la cantidad de desarrolladores para llevarlo a cabo, que en este caso son solo dos. Además, al no existir una definición detallada y estricta de los requisitos, es imperiosa la necesidad de adaptarse paulatinamente a los cambios que surjan durante la vida del proyecto, por lo que RUP no constituye la opción más viable para un entorno tan volátil, mientras que las metodologías ágiles se presentan como candidatos más fiables.

El uso de una metodología ágil permitirá el desarrollo de pequeños prototipos funcionales cada determinado tiempo, lo cual no solo ordenaría y simplificaría la creación del software de forma incremental, sino también aportaría una mayor retroalimentación de la investigación en sí misma.

Scrum es una metodología que aporta un fuerte marco para la gestión de proyectos. La inspección constante y continua, así como la adaptación del producto a las necesidades del cliente en tiempo real, son principios muy ajustables al sistema a desarrollar, aunque se ha considerado que su escasa generación documental no es apropiada para la certificación y futuro mantenimiento del producto final.

XP será la metodología que guíe el desarrollo de la solución. Su enfoque de simplicidad y adaptabilidad son características deseadas en el proceso de desarrollo. Además, sus prácticas son ajustables al entorno de trabajo del CISED, entre las que destaca la codificación en pareja. XP se centra también en la generación de una documentación, aunque no exhaustiva, suficiente para soportar la solución, teniendo en cuenta la necesidad de resultados tangibles a corto plazo.

No existen muchas diferencias entre Visual Paradigm y Rational Rose en cuanto a las funcionalidades básicas necesarias para la modelación del sistema. Sin embargo se seleccionó Visual Paradigm debido a que el equipo de proyecto está familiarizado con su empleo y ya se tiene una experiencia previa positiva.

El IDE seleccionado para el desarrollo es Visual Studio. Este IDE facilita la construcción de aplicaciones

de forma rápida y sencilla, garantizando que sean seguras, confiables y administrables, características que son deseadas y fundamentales en el sistema a desarrollar. Mientras que la implementación se realizará mediante el lenguaje de programación de propósito general C#. Debe señalarse que a pesar de ser Netbeans un fuerte candidato por su modularidad y extensibilidad, además de su condición de ser libre y gratuito, el sistema a desarrollar será utilizado por un AFIS que se ejecuta sobre la plataforma .Net, por lo que se necesita compatibilidad en este aspecto, requisito que sería imposible si el sistema utilizase la JVM (Java Virtual Machine) una vez escrito en Java.

RabbitMQ, ApacheQpid y OpenAMQ son todos proyectos de código libre que implementan el protocolo AMQP y cuyos resultados son satisfactorios para lograr un entorno distribuido. OpenAMQ posee una velocidad y latencia de mensajes adecuada para la presente solución, pero tiene el inconveniente de no ser completamente fiel al estándar, pues deshecha parte de sus funcionalidades.

ApacheQpid junto con RabbitMQ son los únicos servidores que implementan en su totalidad el protocolo, ambos son líderes en su aplicación y han sido utilizados en proyectos de gran envergadura con excelentes índices de distribución. Sin embargo, al emparejarlos se determinó que las potencialidades de OTP Erlang para la ejecución distribuida y conmutación ante errores presentes en RabbitMQ, aportará un mayor rendimiento al sistema a desarrollar. Otra de las razones por las que RabbitMQ se presenta como mejor candidato es la posibilidad que brinda de crear clústeres, característica que permitirá alcanzar una alta disponibilidad en las comunicaciones.

En cuanto a la administración de los distintos elementos del corredor, con RabbitMQ es mucho más sencilla que con OpenAMQ y ApacheQpid, pues cuenta con una interfaz de usuario para el control de los mismos.

Entre MySQL y PostgreSQL, se decidió que este último era el que con mayor completitud ofrecía un balance entre la integridad, la eficiencia y las funcionalidades para la consulta de los datos. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios sin presentar incoherencias en la escritura de los datos debido al MVCC. Además, es muy tolerante a fallos debido al uso de multiprocesos y no de multihilos. Estas son características únicas de PostgreSQL que lo hacen la opción más idónea para un sistema de tiempo de respuesta rápido como es el caso del sistema a desarrollar.

1.9. Conclusiones parciales

En el estudio realizado a los diferentes AFISs se encontraron evidencias de que aquellos con mejores resultados en el mercado internacional, e incluso el desarrollado en el país por DATYS, son software propietario, precisan de grandes prestaciones de hardware y utilizan el procesamiento distribuido para lograr tiempos de respuestas adecuados y una alta escalabilidad. El AFIS desarrollado en el CISED debe ser optimizado con el fin de realizar el proceso de identificación en un tiempo menor previendo el futuro crecimiento del banco de datos, por lo que el desarrollo del sistema para la distribución

del proceso de búsqueda en el banco de datos de un AFIS cobra especial importancia por el aporte económico que supondría su inclusión en las soluciones informáticas del CISED y particularmente su utilización en los sistemas de control de acceso dentro de la Universidad y el país.

En esta primera fase se ha hecho un estudio profundo de las metodologías, lenguajes, tecnologías y herramientas, que ayudó a determinar y fundamentar, según las necesidades de la solución, la utilización de XP como metodología de desarrollo, Visual Paradigm como herramienta de modelado, C# como lenguaje de programación, Visual Studio como entorno de desarrollo, RabbitMQ como servidor distribuidor de mensajes y PostgreSQL como SGBD.

Capítulo 2: Características del sistema

2.1. Introducción

En este capítulo se abordan las principales características de la solución propuesta. Debido a la poca estructuración de los procesos identificada después de una valoración del negocio, se precisa la definición de un modelo de dominio donde se agrupen los principales conceptos a tratar y de una metáfora global que describa el funcionamiento del sistema. Se especifica el levantamiento de los requisitos funcionales y no funcionales y son representados en las historias de usuario. Se realizan los diagramas de clases del diseño a partir de las tarjetas CRC y una descripción de la arquitectura del sistema, así como de los patrones de diseño utilizados, logrando de esta forma un entendimiento mayor del sistema para su posterior implementación.

2.2. Modelo de dominio

En el modelo de dominio se ilustran los principales conceptos con los que trabaja el sistema a desarrollar, ayudando a comprender su entorno y a definir sus clases más significativas (Ver Figura 7).

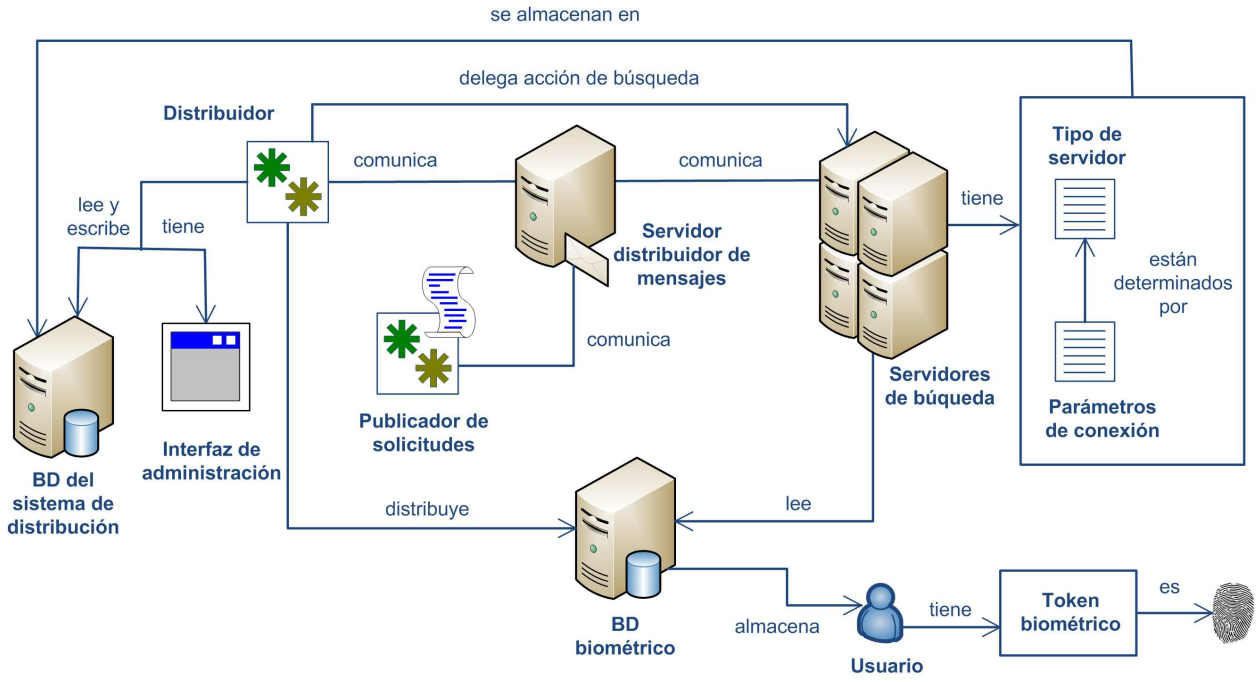


Figura 7: Modelo de dominio.

El sistema propuesto surge en el marco de los problemas acarreados por el modelo centralizado que utiliza el AFIS del Departamento de Biometría del CISED en su proceso de búsqueda de huellas dactilares en el banco de datos. Aunque su objetivo principal está dirigido a solucionar estos problemas mediante un enfoque de distribución, el modelo de dominio presentado abarca conceptos más generales

debido a que el diseño e implementación de la solución estarán orientados a su futura utilización por otros sistemas biométricos.

2.2.1. Glosario de conceptos del modelo de dominio

A continuación se definen brevemente los conceptos ilustrados en el modelo de dominio.

- **Distribuidor:** Es el encargado de particionar el banco de datos entre los distintos servidores de búsqueda. Maneja las solicitudes de distribución e identificación, enviando peticiones a los servidores de búsqueda y posteriormente procesando las respuestas recibidas.
- **Interfaz de administración:** Interfaz que permite monitorear los servidores de búsqueda y ejecutar las acciones relacionadas con la distribución.
- **Banco de datos del sistema de distribución:** Contiene todos los datos persistentes del sistema, como son los tipos de servidores de búsqueda y los parámetros de conexión a los bancos de datos.
- **Servidor de búsqueda:** Recibe un token biométrico en correspondencia con su tipo. Busca y compara el token con los tokens almacenados en la porción del banco de datos que se le ha asignado.
- **Tipo de servidor de búsqueda:** Está determinado por la combinación del token biométrico que maneja y del motor de identificación utilizado[1]. Las peticiones realizadas por el distribuidor estarán en correspondencia con el tipo de servidor de búsqueda.
- **Parámetros de conexión:** Son los parámetros de conexión al banco de datos del sistema biométrico que serán configurados según el tipo del servidor de búsqueda.
- **Publicador de solicitudes:** Es el responsable de publicar las solicitudes de distribución e identificación para que el distribuidor pueda procesarlas.
- **Banco de datos biométrico:** Contiene todos los datos persistentes de un sistema biométrico.
- **Usuario:** Entidad dentro del banco de datos del sistema biométrico que contiene los datos de una persona, entre ellos el token biométrico.
- **Token biométrico:** Representa una característica biométrica que puede ser física (la huella dactilar, el iris, la geometría de la mano) o conductual (la firma manuscrita, la manera de caminar). Es almacenado como parte de los datos de un usuario.
- **Huella dactilar:** Es la representación superficial de la epidermis de un dedo. Están constituidas por rugosidades que forman salientes (crestas) y depresiones (valles).
- **Servidor distribuidor de mensajes:** Es el intermediario en la comunicación entre el publicador de solicitudes, el distribuidor y los servidores de búsqueda.

[1]Se refiere al método implementado en cada servidor de búsqueda para realizar la identificación. Está determinado por el sistema biométrico.

2.3. Metáfora

Una vez descrito el entorno del sistema a desarrollar en el modelo de dominio, es preciso crear una metáfora que describa cómo este debe funcionar. A continuación se define la metáfora global que guía el desarrollo del presente proyecto.

El sistema propuesto en esta investigación está conformado por dos módulos para lograr la distribución del proceso de búsqueda de huellas dactilares en el banco de datos del AFIS del Departamento de Biometría del CISED.

En el primer módulo dedicado a la distribución, se chequeará periódicamente la disponibilidad de los distintos servidores de búsqueda. Se podrán seleccionar los servidores de búsqueda con los que se desee trabajar, teniendo como criterio para esta decisión aquellas propiedades de hardware de cada uno de ellos que influyen directamente en la distribución, como son la memoria RAM disponible y la velocidad de procesamiento de CPU. Este módulo es responsable de distribuir el banco de datos entre los servidores de búsqueda que han sido seleccionados para trabajar, balanceando la carga de forma tal que se maximice la utilización de los recursos y se obtenga el mejor desempeño del sistema. También se atenderán solicitudes de identificación, delegando la búsqueda en los servidores destinados a esta tarea, para posteriormente procesar sus respuestas. Se podrá monitorear el estado de los servidores de búsqueda, así como generar logs sobre las solicitudes atendidas.

El segundo módulo lo constituyen los servidores de búsqueda, los cuales buscarán en su porción del banco de datos el individuo al que corresponde la huella dactilar cuya comparación se requiere y responderán en consecuencia al resultado obtenido.

2.4. Principales funcionalidades

Los requisitos funcionales son características del sistema que expresan funcionalidades, es decir, describen lo que el sistema debe hacer. A continuación se enumeran los requisitos funcionales del sistema a desarrollar, los cuales serán descritos con mayor detalle en las historias de usuario.

El requisito funcional que permitirá la futura utilización del sistema de distribución por otros sistemas biométricos de forma simultánea es:

1. Gestionar tipos de servidores de búsqueda:
   - 1.1. Adicionar un tipo de servidor de búsqueda: Para realizar esta acción deben introducirse los siguientes datos:
     - Motor de identificación
     - Token biométrico
     - Descripción
   - 1.2. Generar la llave de identificación correspondiente al tipo de servidor de búsqueda adicionado.

1.3. Eliminar un tipo de servidor de búsqueda.

El siguiente requisito funcional permitirá configurar los parámetros de conexión de los servidores de búsqueda al banco de datos del sistema biométrico de forma tal que esta operación sea realizada solo una vez para un tipo de servidor de búsqueda determinado, sin la necesidad de hacerlo manualmente en cada servidor de búsqueda.

2. Gestionar conexiones a bancos de datos.
   2.1. Adicionar la conexión a un banco de datos: Para realizar esta acción deben introducirse los siguientes datos:
      - Usuario
      - Contraseña
      - Puerto
      - Nombre del servidor del banco de datos
      - Nombre del banco de datos
      - Tipo de servidor de búsqueda
   2.2. Modificar la conexión a un banco de datos.
   2.3. Eliminar la conexión a un banco de datos.

Seguidamente se listan los requisitos funcionales de mayor importancia para el negocio:

3. Detectar servidores de búsqueda.
4. Modificar el estado de trabajo de un servidor de búsqueda.
5. Obtener propiedades de los servidores de búsqueda:
   - Memoria RAM disponible
   - Velocidad de procesamiento de CPU
   - Valor de aptitud
6. Calcular la capacidad de carga de los servidores de búsqueda.
7. Atender una solicitud de distribución.

El sistema debe permitir además:

8. Atender una solicitud de identificación.
9. Configurar opciones de distribución: Para realizar esta acción deben introducirse los siguientes datos:
   - Número de peticiones sobre cola de mensajes[2]: Varía en dependencia del proceso que se ejecute.
     - Número de peticiones para la detección de servidores de búsqueda (El valor por defecto

[2]Debe ser establecido en correspondencia con la infraestructura de red donde se despliegue el sistema. La modificación de este valor debe tener en cuenta que la distribución del banco de datos es el proceso que más demora, seguido por la obtención de propiedades y la detección. Este valor se encuentra en una escala de $[1, 100]$.

estará definido en 20)
    - Número de peticiones para la distribución del banco de datos (El valor por defecto estará definido en 20)
    - Número de peticiones para la obtención de propiedades de los servidores de búsqueda (El valor por defecto estará definido en 40)
  - Período de tiempo de detección de servidores de búsqueda[3] (El valor por defecto estará definido en 4 segundos).

10. Generar logs del sistema.

2.5. Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Seguidamente se especifican los requisitos no funcionales del sistema a desarrollar.

1. Seguridad: Se utilizará una contraseña de bloqueo para la interfaz de administración.
2. Soporte: El sistema contará con un grupo de soporte cuyo objetivo es brindar asesoría al cliente y soporte técnico al producto.
3. Software: Plataforma .Net versión 4.0, servidor RabbitMQ versión 3.0 y PostgreSQL 9.1.
4. Requisitos de hardware:
   4.1 En el distribuidor: CPU 3.0 GHz o superior y 1 GB de memoria RAM o superior.
   4.2 En los servidores de búsqueda: CPU 2.13 GHz o superior y 1 GB de memoria RAM o superior.

2.6. Historias de usuario

La gestión de los requisitos, tanto funcionales como no funcionales, es realizada mediante las historias de usuario, artefacto generado por la metodología XP donde el cliente describe las características del producto final. A continuación se muestran las historias de usuario relacionadas con la atención de solicitudes de distribución e identificación (Ver Anexo Historias de usuario para consultar el resto de las historias de usuario).

Tabla 1: HU Atender una solicitud de distribución.

	Historia de usuario
Número: CDHU05	**Usuario:** Royli Hernández Delgado
Nombre: Atender una solicitud de distribución.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 3

[3]Expresa el período de tiempo en el que constantemente el sistema realizará la detección.

Programador responsable: Melvis Machin Armas
Descripción: La presente historia de usuario tiene como objetivo indicar a cada servidor de búsqueda la porción del banco de datos que le corresponde.
Observaciones: Se debe tener en cuenta para la distribución un conjunto de métricas que posibiliten una asignación equilibrada y que aseguren un mayor rendimiento. Esta funcionalidad requiere el tamaño del banco de datos a distribuir.
Requisitos no funcionales: Software, Seguridad
Asuntos pendientes: Ninguno

Tabla 2: HU Atender una solicitud de identificación.

	Historia de usuario
Número: CDHU06	**Usuario:** Melvis Machin Armas
Nombre: Atender solicitud de identificación.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Royli Hernández Delgado	
Descripción: La presente historia de usuario tiene como objetivo atender una solicitud de identificación.	
Observaciones: Cada servidor de búsqueda realizará el proceso de identificación en la porción del banco de datos que se le haya asignado con anterioridad, emitiendo una respuesta.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

2.7. Planificación

La planificación en XP está basada en un conjunto de decisiones tomadas por el cliente de conjunto con los programadores. Los clientes representan las necesidades propias del negocio como son el alcance del proyecto, la prioridad de las historias de usuario y las fechas de entrega de cada versión del producto final. Mientras que los programadores definen los requerimientos técnicos que complementan las necesidades del negocio, como por ejemplo la duración estimada de la implementación de las historias de usuario y la organización del proceso de desarrollo en general.

Seguidamente se define el plan de entrega y el plan de iteraciones que regirán el desarrollo.

2.7.1. Plan de entrega

Después de establecida las prioridades de las historias de usuario por el cliente se obtiene un cronograma sobre las versiones resultantes al final de cada iteración. Para la presente solución se han definido 4 iteraciones en las que se codificarán las 10 historias de usuario obtenidas, para una duración total del proyecto de 16 semanas, según el plan de entrega realizado.

Tabla 3: Plan de entrega.

Entregable	Iteración	Fin de iteración
Esquema de detección	1	Enero 2013
Esquema de balanceo de carga	2	Febrero 2013
Esquema de distribución	3	Marzo 2013
Gestión y configuración del sistema	4	Abril 2013

2.7.2. Plan de iteraciones

Para obtener un prototipo funcional al final de cada iteración, se ha definido un plan de iteraciones con el orden en el que se implementarán las historias de usuario.

Tabla 4: Plan de iteraciones.

Iteración	No. HU	Historia de usuario	Semanas estimadas
1	CDHU01	Detectar servidores de búsqueda.	3
	CDHU02	Modificar el estado de trabajo de un servidor de búsqueda.	
2	CDHU03	Obtener las propiedades de los servidores de búsqueda.	3
	CDHU04	Calcular la capacidad de carga de los servidores de búsqueda.	
3	CDHU05	Atender una solicitud de distribución.	5
	CDHU06	Atender solicitud de identificación.	
4	CDHU07	Gestionar tipos de servidores de búsqueda.	5
	CDHU08	Gestionar conexiones a bancos de datos.	
	CDHU09	Generar logs del sistema.	
	CDHU10	Configurar opciones de distribución.	

En las dos primeras iteraciones se priorizarán las historias de usuario de las que depende la distribución, relacionadas con la detección de los servidores de búsqueda, su estado de trabajo y capacidad de carga. En la tercera iteración se codificarán las historias de usuario de mayor criticidad para el negocio como son la atención de solicitudes de distribución e identificación. Finalmente se desarrollarán las historias de usuario de menor complejidad e impacto en el negocio.

2.8. Diseño

En el diseño se indica cómo se construirá la solución mediante la definición de una estructura lo más sencilla posible que responda a la satisfacción de los requisitos funcionales y no funcionales. Además, se realiza la identificación de los objetos lógicos que componen al software para su posterior implementación, de ellos se definen sus atributos y operaciones.

2.8.1. Tarjetas CRC

Las tarjetas CRC son una técnica utilizada en XP para diseñar la solución informática según el paradigma orientado a objetos. Las siglas CRC se refieren a Class, Responsibilities y Collaborators que son precisamente los elementos fundamentales de estas tarjetas, o sea, para su confección se identifican las responsabilidades del software, los objetos que las ejecutan, las clases que los definen, así como aquellos objetos que colaboran en una determinada responsabilidad. A continuación se hace referencia a las clases identificadas y se muestran las tarjetas CRC de las principales (Ver el Anexo Tarjetas CRC para consultar el resto de las tarjetas CRC).

Para lograr que el sistema a desarrollar sea independiente del sistema biométrico que requiera la distribución de la búsqueda en su banco de datos se definieron clases, que aunque no contienen responsabilidades propician dentro del negocio la abstracción. Estas clases son: Persona y Token biométrico. Las clases Tipo de servidor de búsqueda y Conexión al banco de datos contribuyen también a la extensibilidad, pues por cada sistema biométrico existirá un tipo de servidor de búsqueda y un banco de datos al que consultar.

Entre las clases encargadas de establecer la comunicación con el servidor RabbitMQ se encuentran: Comunicador, Comunicador con intercambiador de tipo fanout, Comunicador con intercambiador de tipo direct y Mensaje. Existe además, un conjunto de clases que definen los identificadores de mensajes, colas e intercambiadores que varían en dependencia de los procesos involucrados en la distribución y que son fundamentales en la comunicación con el middleware.

Tabla 5: Tarjeta CRC Comunicador.

Comunicador	
Responsabilidades	**Colaboradores**
Abrir un canal de comunicación	• RabbitMQ.Client.dll

Cerrar un canal de comunicación.	• RabbitMQ.Client.dll
Enviar un mensaje (debe ser definido según el tipo de comunicador).	• Mensaje
Chequear una cola de mensajes.	• Mensaje
Recibir un mensaje.	• Mensaje
Cerrar una cola de mensajes.	–

Para el módulo de distribución las principales clases identificadas fueron Distribuidor y Manejador de peticiones, la primera se encarga de la atención de solicitudes y la segunda es responsable del flujo de peticiones realizado por el distribuidor a los servidores de búsqueda.

Tabla 6: Tarjeta CRC Distribuidor.

Distribuidor	
Responsabilidades	**Colaboradores**
Atender una solicitud de distribución.	• Manejador de peticiones • Tipo de servidor de búsqueda • Comunicador con intercambiador de tipo direct
Atender una solicitud de identificación.	• Token biométrico • Persona • Manejador de peticiones • Tipo de servidor de búsqueda • Comunicador con intercambiador de tipo direct
Registrar servidores de búsqueda activos.	• Manejador de peticiones • Tipo de servidor de búsqueda • Información de trabajo de servidor de búsqueda
Registrar propiedades de los servidores de búsqueda.	• Manejador de peticiones • Información de trabajo de servidor de búsqueda
Calcular la capacidad de carga de los servidores de búsqueda.	–

La lógica de los servidores de búsqueda será definida en dependencia de su tipo, es decir, el sistema biométrico que en el futuro utilice el servicio de distribución deberá implementar las peculiaridades de

sus servidores de búsqueda. Las clases Controlador de peticiones, Servidor de búsqueda y Propiedades de cómputo estructuran el comportamiento de los servidores de búsqueda del AFIS del CISED, utilizando como motor de identificación el paquete de clases SourceAFIS.

Tabla 7: Tarjeta CRC Servidor de búsqueda.

Servidor de búsqueda	
Responsabilidades	**Colaboradores**
Cargar el banco de datos.	–
Calcular el valor de la función de aptitud.	–
Obtener propiedades de hardware.	• Propiedades de cómputo
Identificar.	• Paquete de clases SourceAFIS

En ambos módulos del sistema se hace uso de la clase Información de trabajo de servidor de búsqueda. El distribuidor maneja una instancia de esta clase por cada servidor de búsqueda detectado. Sin embargo, es responsabilidad de los servidores de búsqueda actualizar su información cada vez que el distribuidor lo requiera.

2.8.2. Diagrama de clases del diseño

Después de definidas las tarjetas CRC, donde se identificaron las clases del sistema y sus principales responsabilidades, se está en condiciones de confeccionar el diagrama de clases del diseño. Este diagrama aunque no es un artefacto propio de XP, permitirá representar gráficamente las clases (atributos y métodos), las relaciones entre clases (dependencias y asociaciones) y la navegabilidad.

En el diagrama de clases del diseño del sistema (Ver Anexo Diagrama de clases del diseño) puede observarse la estructura del distribuidor y de un servidor de búsqueda, los cuales comparten un conjunto de clases (Object Domain) que deben manipular cuando interactúan.

Como se mencionó con anterioridad, las clases que estructuran un servidor de búsqueda están definidas específicamente para el AFIS del CISED. Aquellos sistemas biométricos que deseen utilizar el sistema de distribución deberán estructurar específicamente la arquitectura de sus servidores de búsqueda. Para ello el diseño ha sido orientado a que los servidores de búsqueda puedan ser implementados en cualquier plataforma. Esto es posible por la notación utilizada en los mensajes a través de los cuales se establece la comunicación entre el distribuidor y los servidores de búsqueda. Los mensajes son enviados en formato JSON, permitiendo de esta forma que sin importar el lenguaje en el que se desee implementar el servidor de búsqueda, este siempre podrá procesar las instrucciones contenidas en el mensaje que el distribuidor le ha enviado.

Para la realización de las solicitudes al sistema de distribución se diseñó RequestsPublisher.dll. Esta

librería será referenciada por la capa del AFIS encargada de hacer las peticiones de distribución e identificación. Finalmente la comunicación con el servidor RabbitMQ será establecida mediante RabbitMQManagement.dll, que recoge las operaciones básicas que se cargarán bajo la demanda del distribuidor y de los servidores de búsqueda. Mediante su uso se promueve un diseño modular que reduce la duplicación de código y facilita la implementación.

2.8.3. Modelo de datos

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases entidad son las que contienen toda la información persistente manejada por el sistema, cuya estructura física y lógica describe el modelo de datos. A continuación se muestra el modelo de datos del sistema a desarrollar (Ver Figura 8).

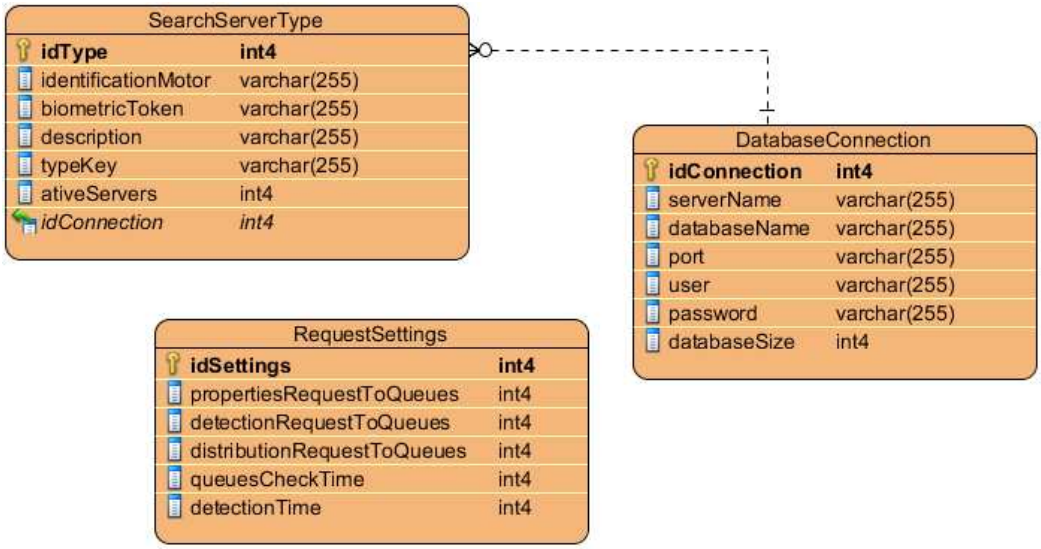


Figura 8: Diagrama entidad-relación del sistema.

La entidad SearchServerType almacena la información de los distintos tipos de servidores de búsqueda que estarán registrados en el sistema. Nótese que cada tipo introducido significa que un nuevo sistema biométrico requerirá el servicio de distribución. Por otra parte la entidad ConnectionDatabase describe los parámetros de conexión al banco de datos del sistema biométrico.

Puede ocurrir que varios sistemas biométricos basados en la identificación de un mismo token biométrico, pero con distinto motor de identificación, accedan a un mismo banco de datos. Esto supondría que los servidores de búsqueda de estos sistemas biométricos tendrían que consultar un mismo banco de datos, razón por la cual la relación entre las entidades SearchServerType y ConnectionDatabase es de uno a muchos.

La entidad RequestSettings almacena información referente a los parámetros de configuración implicados en la distribución.

Debe tenerse en cuenta que:

- La simplicidad del modelo de datos se debe a que la principal función del sistema de distribución es optimizar el proceso de búsqueda, sin interés en gestionar la información de los recursos que utilizará.
- Los servidores de búsqueda del AFIS del CISED utilizarán el modelo de datos establecido por el mismo (Ver Anexo Modelo de datos del AFIS).

2.8.4. Patrones de diseño

Los patrones de diseño expresan esquemas para solucionar problemas del mismo tipo que comúnmente se presentan en el diseño del software [59].

Los patrones GRASP (General Responsibility Assignment Software Patterns) describen los principios fundamentales de la asignación de responsabilidades a objetos. Los patrones de este tipo usados son:

- **Controlador:** Es un intermediario entre la capa de presentación y el núcleo de las clases donde reside la lógica del sistema, que coordina la actividad de otros objetos. Por ejemplo, en el sistema se hace uso del patrón controlador definiendo la clase RequestsController para el módulo de los servidores de búsqueda y la clase Distributor para el módulo de distribución, las cuales gestionan todo el flujo de sus correspondientes módulos y manejan varias instancias de objetos.
- **Experto:** Establece que cada responsabilidad sea asignada a la clase contenedora de la información necesaria para cumplirla. Garantiza el mínimo de relaciones entre las clases, para así obtener un sistema sólido y fácil de mantener. Por ejemplo, este patrón se pone de manifiesto en la clase Distributor que es la encargada de distribuir el banco de datos entre los distintos servidores de búsqueda.
- **Creador:** Establece que una instancia de un objeto solo pueda ser creada por la clase que contiene la información necesaria para ello. Su uso permite crear las dependencias mínimas necesarias entre las clases, beneficiando el mantenimiento del sistema y brindando mejores oportunidades de reutilización. Por ejemplo, en el sistema se pone en práctica en la clase SearchServer, donde se crea un objeto de la clase Message, en la cual para poder crear un instancia de ella hay que suministrarle la información necesaria, o sea, el tipo y el cuerpo o datos del mensaje.
- **Alta Cohesión:** Basado en la asignación de responsabilidades teniendo en cuenta que la cohesión permanezca alta. Su utilización facilita la comprensión del diseño e incrementa las capacidades de reutilización.
- **Bajo Acoplamiento:** Plantea que debe existir una alta reutilización entre las funcionalidades de las clases con una mínima dependencia, contribuyendo así al mantenimiento de las mismas. El empleo de los patrones Experto y Creador favorecen al bajo acoplamiento entre las clases del sistema.

Los patrones GOF (Gang of Four) describen las formas en las que pueden ser organizados los objetos

para trabajar unos con otros, formando estructuras de mayor complejidad. Los patrones de este tipo usados son:

- **Acción:** Es un patrón de comportamiento a nivel de objetos. Se evidencia cuando en una interfaz gráfica de usuario existe más de una alternativa para acceder a una misma función de la aplicación. La solución brindada por este patrón consiste en eliminar la redundancia a la hora de implementar dicha función. Por ejemplo, en la interfaz de administración del sistema se pondrá de manifiesto este patrón al detectar los servidores de búsqueda, ya que a esta funcionalidad se podrá acceder de dos maneras: a través de un menú desplegable, o a través de un botón Iniciar. Para ello se implementará una acción nombrada UpdateDetection, que será invocada por los dos elementos visuales.
- **Observador:** Es un patrón de comportamiento a nivel de objetos que favorece el bajo acoplamiento. Define una dependencia de uno a muchos entre objetos, para de esta forma permitir que el cambio de estado de un objeto sea notificado y actualizado automáticamente en todos los objetos que dependen de él. Por ejemplo, en la solución este patrón se evidencia específicamente en la relación entre las entidades SearchServerType y ConnectionDatabase.

2.9. Arquitectura

La arquitectura es la representación de más alto nivel de la estructura de un sistema informático. Define los componentes funcionales que la integran y las interacciones entre ellos. Además, incluye un conjunto de patrones y restricciones que supervisan su composición como estándares, convenciones, reglas y procesos [60].

Para el desarrollo del sistema se utilizará un estilo arquitectónico basado en la configuración cliente-servidor. Este modelo se basa en repartir el tratamiento de la información y los datos por todo el sistema, permitiendo así un mayor rendimiento. Proporciona un acceso transparente a los distintos recursos que se comparten, brindando un entorno en el que las tareas son llevadas a cabo por los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Ambos, clientes y servidores, se encuentran conectados a través de una red de comunicaciones [61].

En consecuencia con este modelo el sistema a desarrollar soportará el procesamiento distribuido, previendo su crecimiento modular y la alta escalabilidad, y permitiendo la implementación de mecanismos eficientes de tolerancia a fallos. La arquitectura propuesta (Ver Figura 9) garantizará en general expandir al AFIS para mejorar su rendimiento en forma dinámica, aumentando el tamaño del banco de datos y su carga de trabajo con el mínimo aumento de componentes, minimizando así el costo de las expansiones a largo plazo.

2.9.1. Patrón arquitectónico

Para la implementación de la arquitectura cliente-servidor se utilizará como patrón arquitectónico n-capas, específicamente 4 capas.

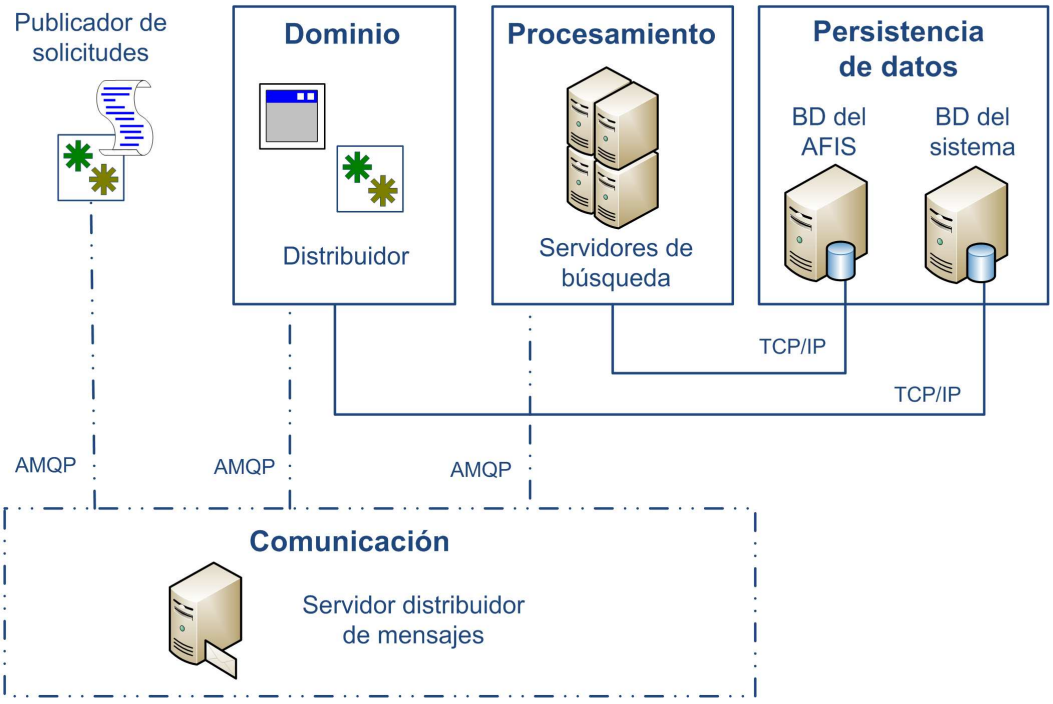


Figura 9: Arquitectura del sistema para la distribución de la búsqueda.

- **Capa de Dominio:** Contiene la interfaz de administración, así como toda la lógica del sistema, desde la atención de las solicitudes de distribución e identificación hasta la gestión de las entidades del sistema.
- **Capa de Procesamiento:** Contiene la lógica de los servidores de búsqueda, que incluye consultas al banco de datos biométrico y la búsqueda de la huella dactilar de un individuo que se desee identificar.
- **Capa de Persistencia de datos:** Está conformada por el banco de datos biométrico al que los servidores de búsqueda consultarán siempre que se requiera distribuirlo, así como por el banco de datos del sistema que almacenará toda la información persistente.
- **Capa de Comunicación:** Contiene un servidor distribuidor de mensajes encargado de distribuir mensajes de forma segura y eficiente a través del middleware RabbitMQ. Es la capa intermediaria en la comunicación entre:
  - Publicador de solicitudes y distribuidor
  - Distribuidor y servidores de búsqueda

La arquitectura tiene como restricción que el crecimiento del banco de datos está limitado por las condiciones de memoria RAM disponible en los distintos servidores de búsqueda.

2.9.2. Descripción de los flujos de distribución e identificación

Para tener una mayor comprensión de cómo se interconectarán y colaborarán los componentes de la arquitectura seguidamente se describirá el comportamiento del sistema ante solicitudes de distribución e identificación.

Para atender una solicitud de distribución el sistema primeramente debe detectar los servidores de búsqueda que estén conectados al servidor RabbitMQ. Una vez obtenida esta información el administrador podrá determinar, según sus características de hardware, aquellos servidores de búsqueda que no estarán activos para trabajar, cambiando su estado de activo a disponible. Estos servidores no intervendrán en los procesos del sistema hasta que su estado sea cambiado nuevamente. Cuando una solicitud de distribución es requerida se envía el tamaño del banco de datos al distribuidor para que balancee la carga de trabajo entre los servidores de búsqueda. Seguidamente se notifica a cada servidor de búsqueda los índices que delimitan la porción del banco de datos del AFIS que se le ha asignado. Cuando a un servidor de búsqueda llega esta petición consulta el banco de datos del AFIS y carga la información en su memoria RAM. Debe señalarse que el sistema es capaz de redistribuir siempre que se detecte algún cambio en los servidores de búsqueda de trabajo, como por ejemplo el fallo de uno de ellos.

Con el banco de datos del AFIS distribuido puede ser atendida una solicitud de identificación. La huella dactilar del individuo a identificar es enviada al distribuidor, quien delega la tarea de búsqueda en los servidores e informa finalmente el resultado del proceso.

2.10. Conclusiones parciales

En este capítulo se expusieron las características del sistema después del análisis de los procesos involucrados en la distribución de la búsqueda de huellas dactilares en el banco de datos de un AFIS. Se definió el modelo de dominio que ayudó a identificar los requisitos funcionales y no funcionales descritos en las historias de usuario.

La solución propuesta implementa una arquitectura cliente-servidor mediante 4 capas: Dominio, Procesamiento, Persistencia de datos y Comunicación. El diseño está orientado a la extensibilidad pues permite que el sistema a desarrollar sea utilizado por otros sistemas de identificación, además, cumple con los patrones de diseño GRASP y GOF que garantizan la organización de las clases y de la estructura de trabajo.

Capítulo 3: Implementación y prueba

3.1. Introducción

En el presente capítulo se describe la implementación del software, fase donde finalmente se materializa el producto final y se cumple con los requisitos obtenidos al inicio de la investigación. Para lograr esto se generan los diagramas de componentes y de despliegue, donde se observan las dependencias lógicas entre los elementos de software y los nodos necesarios para el despliegue del sistema respectivamente. Además, se muestran las interfaces de usuario y se da una explicación de cada una de ellas para un mayor entendimiento acerca del funcionamiento del sistema. Finalmente se muestran las pruebas realizadas para validar el correcto funcionamiento de la solución.

3.2. Tareas de ingeniería

El trabajo de una iteración es expresado en tareas de ingeniería que emergen de las historias de usuario. Cada una de estas tareas es asignada a un programador como responsable, pero llevada a cabo por una pareja de programadores. Las historias de usuario brindan un escaso nivel de detalle para afrontar la implementación por lo que las tareas de ingeniería juegan un papel fundamental al indicar a los programadores las acciones a realizar por cada una de ellas.

A continuación se muestran las tareas de ingeniería implicadas en la atención de las solicitudes de distribución e identificación (Ver el Anexo Tareas de ingeniería para consultar el resto de las tareas de ingeniería).

Tabla 8: Tareas de ingeniería de la tercera iteración.

Iteración 3	
Historia de usuario	**Tareas**
Atender una solicitud de distribución.	1. Dividir el banco de datos según la capacidad de carga de cada servidor de búsqueda. 2. Enviar un mensaje identificado como petición de asignación de datos con el intervalo del banco de datos correspondiente a cada servidor de búsqueda. 3. Cargar la porción del banco de datos en cada servidor de búsqueda. 4. Enviar un mensaje identificado como respuesta de distribución con detalles del proceso desde cada servidor de búsqueda.

	5. Esperar y procesar respuestas de confirmación de los servidores de búsqueda.
Atender solicitud de identificación.	1. Enviar un mensaje identificado como petición de identificación con la imagen del token biométrico (huella dactilar) introducido en el sistema, a los servidores de búsqueda. 2. Realizar la identificación en cada servidor de búsqueda. 3. Realizar tratamiento de fallos. 4. Enviar un mensaje identificado como respuesta de identificación con el resultado obtenido desde cada servidor de búsqueda. 5. Esperar y procesar los resultados del proceso en cada servidor de búsqueda.

3.3. Estándares de codificación

Establecer un estándar de codificación que sea aceptado por todo el equipo de desarrollo es muy importante en una metodología como XP que promueve la propiedad colectiva del código y la constante refactorización. Además, influye directamente en la facilidad con que pueda mantenerse el código si fuese necesario añadir nuevas funcionalidades al software, modificar las ya existentes, depurar errores o mejorar el rendimiento. Para la implementación del sistema se definió el estándar de codificación que se describe a continuación.

3.3.1. Estilos para la capitalización de los identificadores

Para la capitalización de los identificadores se utilizaron los convenios:

- **Pascal:** La primera letra en el identificador y la primera letra de cada subsiguiente palabra concatenada se capitalizan. Por ejemplo: SendMessage.
- **Camel:** La primera letra en el identificador se pone en minúscula y la primera letra de cada subsiguiente palabra concatenada en mayúscula. Por ejemplo: requestMessage.

La convención Pascal se empleó en los identificadores de las clases, métodos y nombres de ficheros. Mientras que la convención Camel es el estilo de los identificadores de las variables, atributos y parámetros.

En los identificadores de las variables se tuvo en cuenta emplear su significado y obviar las abreviaturas, específicamente en las variables booleanas sus identificadores usan el prefijo “is”. Los identificadores de los métodos están en correspondencia a lo que hacen. Los ficheros son identificados por el nombre de la única clase que contienen.

3.3.2. Comprensión y legibilidad del código

Con el objetivo de incrementar la legibilidad del código se comentaron todas las declaraciones de clases y funciones más complejas. Los atributos, propiedades y constructores aparecen en la parte superior de las clases, mientras que los métodos privados y públicos están a continuación. Además, se organizó el código de forma estructurada en bloques de código, para una mejor comprensión del mismo.

3.4. Diagrama de componentes

El diagrama de componentes representa, como indica su nombre, la división del software en partes más pequeñas denominadas componentes. Muestra la organización y las dependencias entre los distintos elementos de software, sean estos código fuente, binario o ejecutable. Las dependencias en este diagrama indican que un componente utiliza los servicios ofrecidos por otro componente. A continuación se describe el diagrama de componentes del sistema (Ver Figura 10).

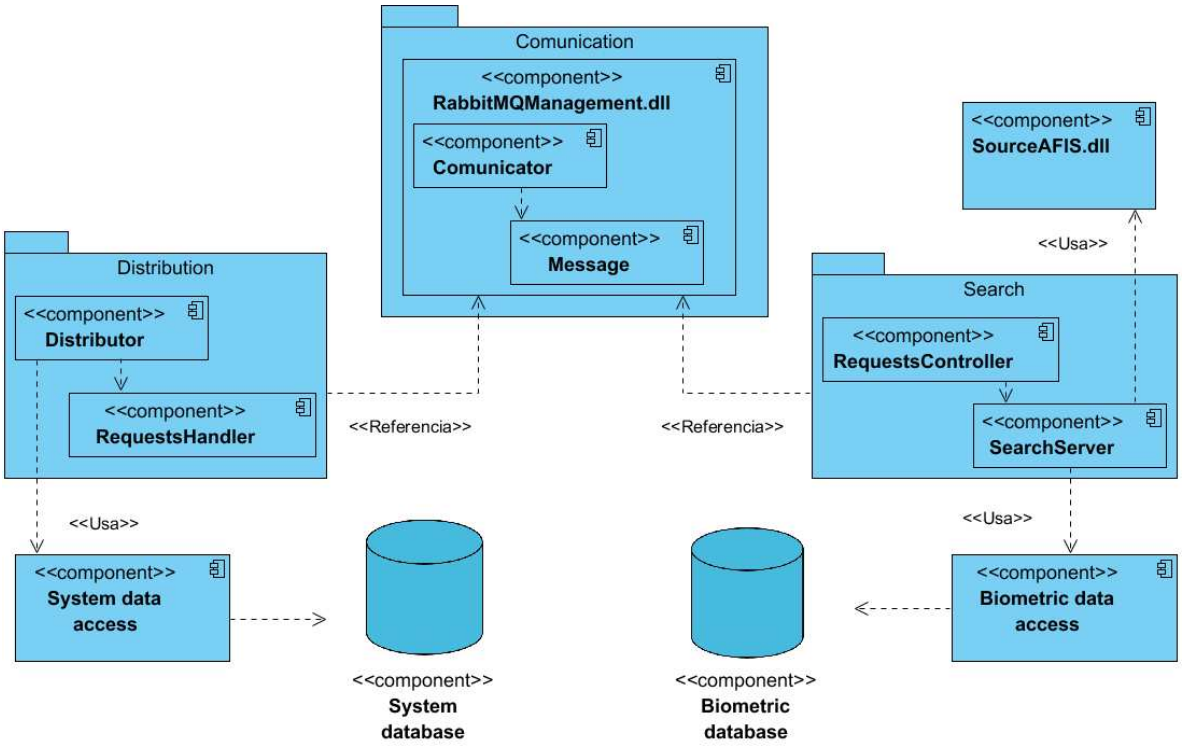


Figura 10: Diagrama de componentes del sistema.

3.4.1. Descripción del diagrama de componentes

En el diagrama se muestra la conformación de la solución por proyectos, de ellos el de Distribución y el de Búsqueda son los fundamentales.

El proyecto de Distribución contiene el elemento Distributor, encargado de manejar la lógica relacionada con la distribución del banco de datos y la atención de solicitudes de identificación, utilizando

RequestsHandler para el manejo de las peticiones realizadas a los servidores de búsqueda.

Los componentes de software RequestsController y SearchServer se encuentran en el proyecto de Búsqueda, el primero controla todo el flujo de peticiones enviadas a un servidor de búsqueda, mientras que el segundo las procesa utilizando SourceAFIS.dll.

Estos proyectos hacen referencia a la librería RabbitMQManagement.dll mediante la cual se establece la comunicación con el middleware RabbitMQ. Los componentes de acceso a datos (acceso al banco de datos del sistema y acceso al banco de datos biométrico) contienen los elementos que modelan los datos y el acceso a estos en los bancos de datos físicos.

3.5. Diagrama de despliegue

El diagrama de despliegue describe la distribución física de un software en un ambiente de producción o prueba. En él se grafican los nodos y las conexiones necesarias para el funcionamiento exitoso del producto software. Muestra también, dónde se localizan los componentes del software, o sea, en qué servidores, ordenadores o dispositivo hardware. A continuación se describe el diagrama de despliegue del sistema (Ver Figura 11).

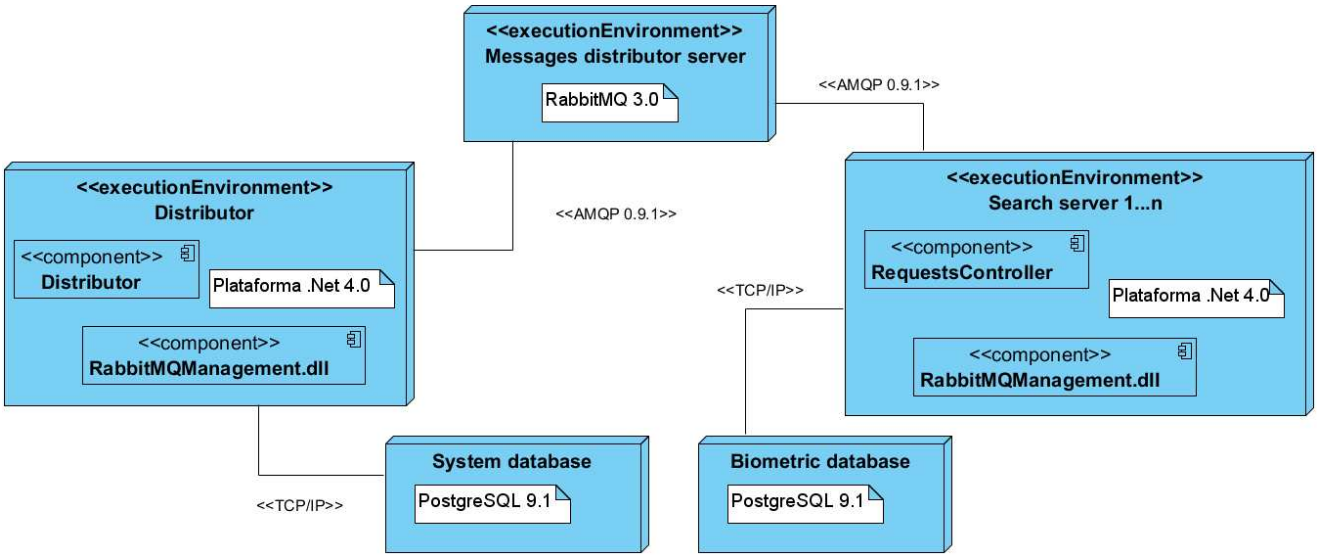


Figura 11: Diagrama de despliegue del sistema.

3.5.1. Descripción del diagrama de despliegue

Toda la comunicación, que incluye peticiones y respuestas, entre el distribuidor y los servidores de búsqueda se realiza teniendo como intermediario al servidor distribuidor de mensajes RabbitMQ 3.0 y utilizando para ello el protocolo AMQP en su versión 0.9.1. Entre las peticiones más importantes se encuentra la de distribución del banco de datos, para lo cual es preciso que los servidores de búsqueda consulten el banco de datos biométrico utilizando el protocolo TCP/IP. También el distribuidor

accederá al banco de datos del sistema de distribución utilizando este protocolo.

3.6. Método para el balanceo de carga

El balanceo de carga consiste en distribuir la carga de trabajo entre las computadoras de un sistema distribuido de acuerdo a los recursos y disponibilidad de cada una de ellas. Su objetivo es maximizar la utilización de los recursos y obtener el mejor desempeño del sistema [62].

El sistema que propone esta investigación distribuye la tarea de búsqueda entre varios servidores, para lograrlo el balanceo de carga juega un papel muy importante, ya que es muy difícil predecir el tamaño de la porción del banco de datos que debe asignársele a cada uno de ellos, de modo que la carga computacional se mantenga uniforme y se consiga la máxima velocidad de ejecución posible.

El método utilizado para realizar la asignación tiene en cuenta qué tan apto es un servidor para ejecutar la búsqueda, tomando en consideración dos propiedades fundamentales:

- **Memoria RAM disponible:** Una vez asignada la porción del banco de datos cada servidor de búsqueda debe cargarla en su memoria RAM, por lo que debe tenerse la precaución de no colapsar el servidor con una asignación que no esté en correspondencia con su disponibilidad.
- **Velocidad de procesamiento de CPU:** Cada servidor buscará en la porción del banco de datos que se le haya asignado cuando se le solicite realizar una identificación, por lo que la velocidad del CPU influye en qué tan rápido se ejecute este proceso.

La función de aptitud se define como la rapidez asociada a la memoria RAM disponible y a la velocidad de CPU. El cálculo de esta función corresponde a la sumatoria entre las propiedades ya mencionadas, donde cada una de ellas estará multiplicada por un peso que dependerá de su importancia para el desempeño del sistema en general.

$$\boldsymbol{Fa_i = a * dRAM_i + b * vCPU_i} \quad (1)$$

Donde

$\boldsymbol{dRAM_i}$ **:** Memoria RAM disponible del servidor de búsqueda i.
$\boldsymbol{vCPU_i}$ **:** Velocidad de procesamiento del CPU del servidor de búsqueda $\boldsymbol{i}$.
$\boldsymbol{a}$ y $\boldsymbol{b}$ **:** Pesos asignados a la memoria RAM disponible y a la velocidad de procesamiento de CPU respectivamente en una escala de $\mathbf{[0;1]}$ ($\boldsymbol{a = 0{,}6}$ y $\boldsymbol{b = 0{,}4}$).

Para realizar el cálculo de la función de aptitud es necesario llevar las variables a una escala común,

ya que si se trabaja con los valores originales la función de aptitud se volvería muy variada para cada servidor de búsqueda, debido a que la variable con mayor escala definiría por completo la función haciendo insignificante el valor total de la otra variable. En este caso la memoria RAM es expresada en MB y la velocidad de CPU en MHz, garantizando de esta forma que ambas variables se encuentren en la misma escala.

Una vez obtenida la función de aptitud se calcula la capacidad de carga de cada servidor de búsqueda. Consiste en el porciento que representa la función de aptitud del servidor i con respecto a la sumatoria de los valores de aptitud de todos los servidores activos.

$$\boldsymbol{CC_i = \frac{FA_i * 100}{SFA}} \tag{2}$$

Donde

$\boldsymbol{CC_i}$ **:** Capacidad de carga del servidor de búsqueda i.
$\boldsymbol{SFA}$ **:** Sumatoria de los valores de aptitud de todos los servidores activos. Se calcula mediante la fórmula:

$$\boldsymbol{SFA = \sum_{i=1}^{N} FA_i} \tag{3}$$

Con los porcentajes de carga calculados ya se está en condiciones de determinar el tamaño de la porción del banco de datos que puede asignársele a cada servidor de búsqueda, estableciendo una correspondencia entre el porcentaje de carga y el tamaño del banco de datos.

$$\boldsymbol{PBD_i = \frac{CC_i * BD}{100}} \tag{4}$$

Donde

$\boldsymbol{PBD_i}$ **:** Porción del banco de datos que debe asignársele al servidor de búqueda i.
$\boldsymbol{BD}$ **:** Tamaño del banco de datos.

PBD_i representa la cantidad máxima de usuarios que puede ser asignada al servidor i para la búsqueda. Si se le asigna un valor mayor que este se estaría sobrecargando al servidor, mientras que si se le asigna un valor menor se desaprovecha su capacidad.

Este método posibilita que todos los servidores activos se involucren en la búsqueda y garantiza la uniformidad en el procesamiento, debido a que cada servidor trabajará proporcionalmente a sus recursos.

3.7. Tratamiento de fallos

Por más confiable que pudiese parecer un sistema es necesario estar siempre preparado para cuando este eventualmente falle. Particularmente en un sistema distribuido los fallos suelen ser parciales, pues solo dejan de brindarse los servicios ofrecidos por el componente que presente problemas, mientras que el resto de los componentes no son afectados y continúan su normal funcionamiento.

Si se tiene en consideración que el sistema propuesto distribuye la búsqueda involucrada en un proceso tan sensible como lo es la identificación, se comprenderá cuán importante resulta la adopción de una estrategia efectiva para el tratamiento a fallos. Esta estrategia está orientada a enfrentar situaciones relacionadas con los servidores de búsqueda, o sea, fallos parciales en el sistema.

3.7.1. Detección de fallos

Para tratar un fallo primeramente es necesario tener conocimiento de que está ocurriendo un evento adverso. Para ello el sistema cuenta con varias vías para detectar que un servidor de búsqueda ha colapsado[1] y para informar cuando una funcionalidad no ha podido ser ejecutada.

La primera de ellas está asociada a la detección de los servidores de búsqueda que realizará el distribuidor frecuentemente en intervalos de tiempo. Esta inspección permitirá no solo conocer cuándo ha dejado de ofrecer servicios un servidor, sino también, detectar nuevos servidores de búsqueda.

Por el contrario el caso más crítico que puede presentarse es la caída de un servidor durante el proceso de búsqueda. Como cada servidor tiene cargada en su memoria RAM la porción del banco de datos que se la ha asignado, un fallo significa que existe un rango de datos en el cual no se realiza la búsqueda y que en el peor de los casos contiene el individuo a identificar. Cuando esta situación se presenta, otro servidor, una vez que ha terminado su búsqueda, asume el trabajo que no se ha realizado (Ver Sección 3.7.3) y envía una notificación al distribuidor informando el nombre en la red del servidor que ha interrumpido sus servicios.

Los servidores de búsqueda no solo pueden colapsar, también pueden tener dificultades en la ejecución de sus funciones por circunstancias como la corrupción de los mensajes enviados por el distribuidor. Para enfrentar esta eventualidad se ha realizado un exhaustivo tratamiento de excepciones donde se notifica al distribuidor sobre la acción que no pudo ser ejecutada.

[1] Un servidor de búsqueda puede dejar de prestar servicios no solo cuando presente problemas de hardware o software, sino también cuando su conexión con el servidor RabbitMQ sea interrumpida.

3.7.2. Enmascaramiento de fallos

El sistema ha sido desarrollado con la premisa de que una vez detectado un fallo este sea corregido internamente. El sistema biométrico que consume el servicio de distribución no percibirá que está ocurriendo un fallo y continuará su trabajo normal. Los fallos son enmascarados recurriendo a una nueva redistribución del banco de datos ante el colapso de un servidor de búsqueda, mientras que cuando se corrompa un mensaje y una funcionalidad no pueda ser ejecutada, el distribuidor es responsable de enviar un nuevo mensaje.

3.7.3. Recuperación ante fallos

La recuperación ante fallos ha sido definida para que el sistema regrese al estado en el que se encontraba antes de ser detectados los fallos. Siempre que exista un cambio en el número de servidores de búsqueda se redistribuirá el banco de datos con la condición de que en ese preciso momento no se esté atendiendo una solicitud de identificación, pues en ese caso se esperará a que culmine la identificación para luego distribuir.

Como se explicó anteriormente el colapso de un servidor de búsqueda mientras se atiende una solicitud de identificación es la situación de mayor delicadeza que puede presentarse. Para evitar que el proceso no se detenga se ha creado el siguiente mecanismo:

Cuando una solicitud de identificación es atendida se envía a los servidores de búsqueda un mensaje con:

- La imagen de la huella dactilar (o el token biométrico en dependencia del sistema de identificación que realice la solicitud) del individuo a identificar.
- Una lista con la información de cada servidor de búsqueda activo, que incluye su nombre en la red[2] y el intervalo del banco de datos[3] que se le ha asignado.

Cada servidor una vez recibido el mensaje realizará la búsqueda en la porción del banco de datos que tenga cargada en su memoria RAM. Al consumir un mensaje este no se elimina por completo, sino que pasa a una estructura de almacenamiento temporal hasta que la cola sea eliminada, en cuyo caso el mensaje desaparece por completo. Esta forma de consumir garantiza que si el servidor colapsa antes de terminar la búsqueda el mensaje será ubicado nuevamente en la cola indicando que no ha sido procesado.

Cuando un servidor termina su tarea e informa al distribuidor del resultado obtenido, procede a chequear las colas de mensajes del resto de los servidores, utilizando para ello la lista antes mencionada. Si encuentra un mensaje en una cola significará que el servidor que la ha publicado ha sufrido algún fallo. Seguidamente el servidor que ha detectado el problema carga en su memoria RAM la porción del

[2]El nombre en la red de un servidor de búsqueda coincide con el nombre de su cola publicada en el servidor RabbitMQ.
[3]Un intervalo está conformado por los índices inicial y final que delimitan una porción del banco de datos

banco de datos del servidor caído accediendo al intervalo del mismo.

El chequeo de las colas de mensajes se ejecutará mientras el distribuidor no informe que la búsqueda debe culminar, lo cual puede evidenciarse de dos formas:

- El distribuidor recibe un mensaje con una identificación positiva.
- El número de mensajes con respuestas de la identificación recibidos por el distribuidor es igual al número de servidores de búsqueda activos inicialmente, dado que si un servidor falla otro se encargará de enviar dos mensajes correspondientes a la búsqueda realizada tanto en su porción del banco de datos como en la del servidor que ha fallado.

El servidor que ha detectado la falla informa, como ya se ha explicado con anterioridad, al distribuidor requiriendo una nueva redistribución.

Pudiera ocurrir que un servidor con escasa disponibilidad de recursos asuma la tarea de un servidor con altas prestaciones y por tanto con una mayor asignación del banco de datos. Sin embargo, en estos casos se ha preferido sacrificar el tiempo de respuesta, demorando un poco más de lo esperado, a cambio de no detener el proceso por completo, y perder de esta forma mucho más tiempo.

3.8. Interfaz gráfica

Al estar conformado el sistema en dos módulos principales, este cuenta a su vez con dos interfaces gráficas, una interfaz accesible en cada servidor de búsqueda y una interfaz principal de administración. Seguidamente se describirán estas interfaces para que se alcance un mayor entendimiento del sistema.

3.8.1. Interfaz de servidores de búsqueda

Cuando un servidor de búsqueda es iniciado una aplicación comienza a ser ejecutada en segundo plano. El acceso a la interfaz del servidor de búsqueda se realiza mediante un icono de notificación ubicado en la Barra de Tareas que permite además, detener el servidor de búsqueda cuando sea requerido (Ver Figura 12).

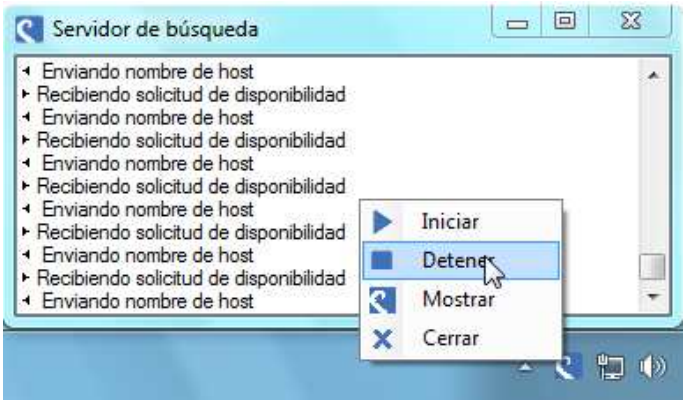


Figura 12: Procesos ejecutándose en el servidor de búsqueda.

Esta interfaz es solo para monitorear el trabajo del servidor de búsqueda, o sea, no brinda opciones para la ejecución de ninguna funcionalidad, pues su objetivo fundamental es consultar las tareas que se están realizando.

3.8.2. Interfaz de administración

La interfaz de administración permite controlar y monitorear el proceso de distribución. Cuando el sistema se inicia este detecta automáticamente los servidores de búsqueda activos, mostrando por cada uno de ellos su nombre en la red, tipo, propiedades de hardware y valor de aptitud. Como la detección es ejecutada a cada cierto intervalo de tiempo, estos datos son constantemente actualizados.

El administrador podrá ver el listado de los servidores de búsqueda activos y disponibles, obtener detalles de la distribución filtrando por tipos de servidores y monitorear los procesos que se están ejecutando en el distribuidor (Ver Figura 13).

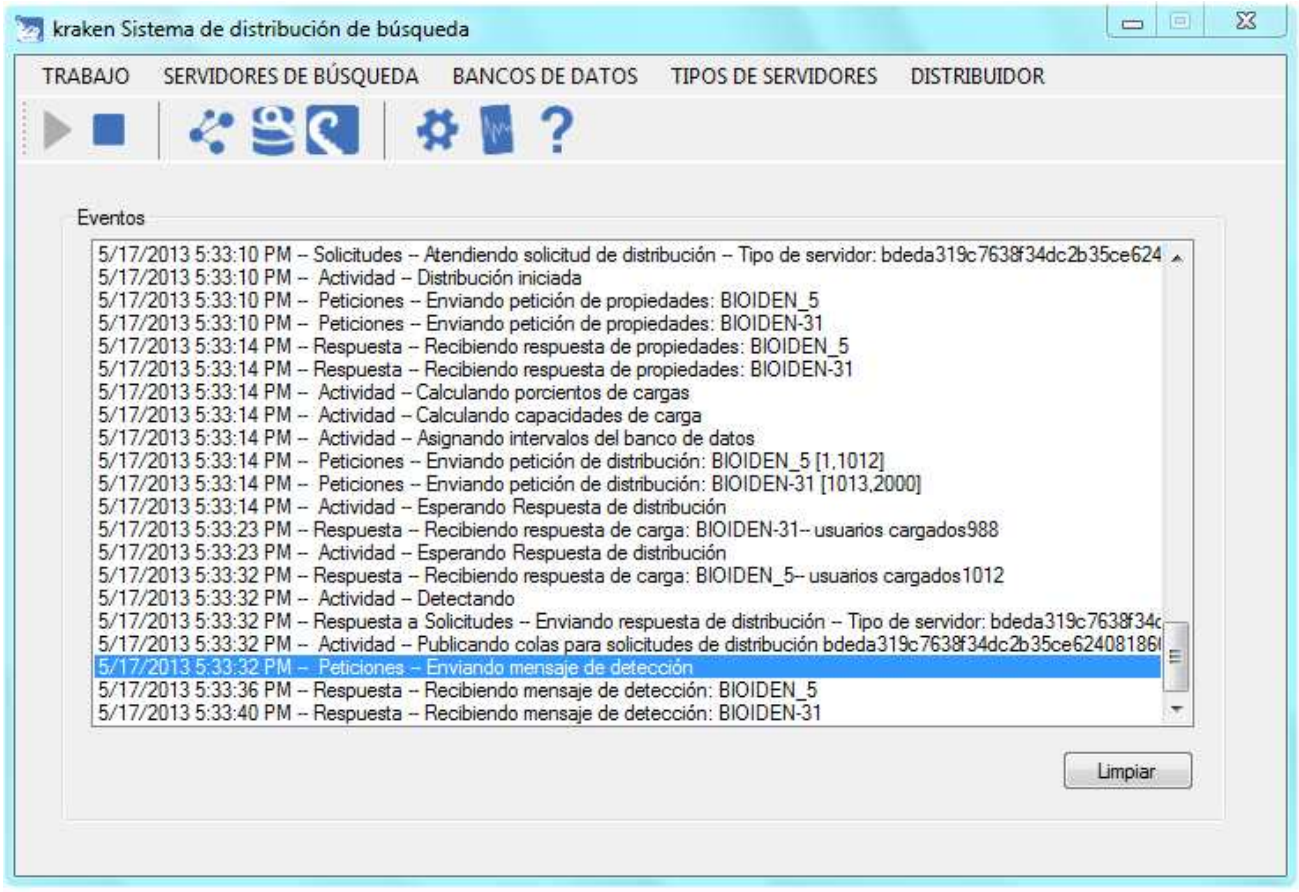


Figura 13: Procesos ejecutándose en el distribuidor.

Cada servidor de búsqueda podrá ser consultado mediante una interfaz que permite cambiar su estado de trabajo y muestra, en el caso de que esté activo, datos como su capacidad de carga, porciento de carga, intervalo del banco de datos y cantidad de usuarios que se le ha asignado (Ver Figura 14).

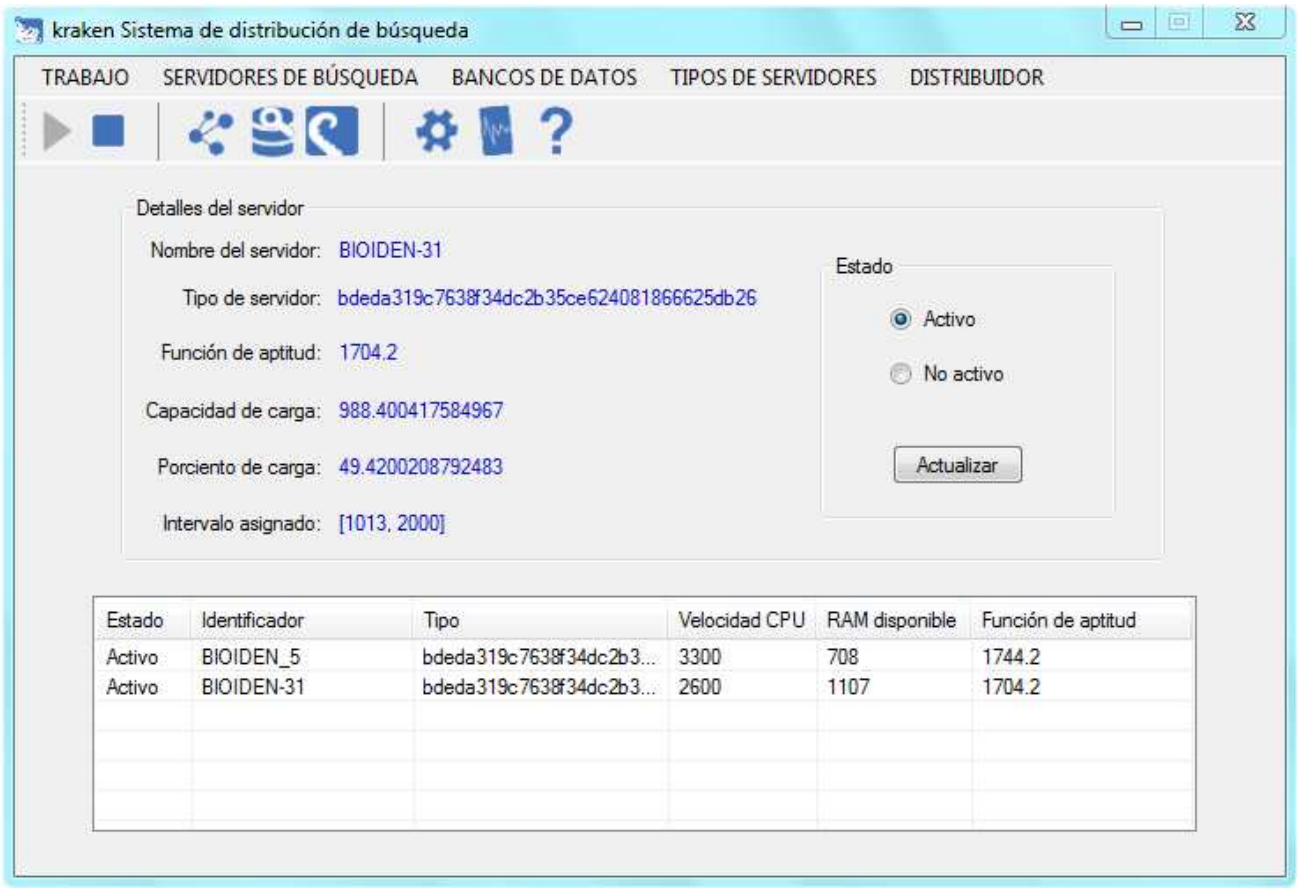


Figura 14: Detalles de servidores de búsqueda.

En cuanto a la configuración, la interfaz de administración permite gestionar los tipos de servidores de búsqueda, las conexiones a los bancos de datos y las opciones de distribución.

3.9. Pruebas

Las pruebas de software son fundamentales para verificar la calidad del producto software teniendo como base el control de la satisfacción de los requisitos. Su principal objetivo es detectar errores. La metodología XP establece dos tipos de pruebas: pruebas unitarias y pruebas de aceptación o funcionales.

3.9.1. Pruebas unitarias

Las pruebas unitarias son escritas por los programadores antes de comenzar la codificación. El objetivo de estas pruebas es aislar pequeñas e individuales porciones del código para verificar que no tengan errores. Para la realización de estas pruebas se utilizó un Add-in de Visual Studio llamado ReScharper (R#) que permite detectar errores y brinda soluciones instantáneas para corregirlos. Los resultados de estas pruebas pueden consultarse en el Anexo Código y resultados de pruebas unitarias.

3.9.2. Pruebas de aceptación

Las pruebas de aceptación o pruebas funcionales son definidas por el cliente y son esenciales en su satisfacción con el producto desarrollado. Constituyen el fin de una iteración y el comienzo de la siguiente que asumirá la corrección de los errores encontrados.

Al sistema se le realizaron dos tipos de pruebas de aceptación: las pruebas alfa y las pruebas beta. Las pruebas alfa fueron realizadas a las historias de usuario implementadas en las dos primeras iteraciones, etapa en la que el sistema era aún inestable y se le continuarían agregando características. Cuando el sistema se encontraba en su primera versión completamente funcional fueron verificadas mediante pruebas beta las historias de usuario restantes, o sea, aquellas que requerían la interacción entre el sistema de distribución y la capa del AFIS encargada de publicar solicitudes.

A continuación se presentan los casos de prueba de las historias de usuario relacionadas con la atención de solicitudes de distribución e identificación (Consultar el resto de los casos de prueba en el Anexo Casos de prueba alfa y en el Anexo Casos de prueba beta).

Tabla 9: CP Atender una solicitud de distribución.

Caso de prueba de aceptación	
Código: CDHU5CP1	**Historia de usuario:** Atender solicitud de distribución.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que se distribuya el banco de datos entre los servidores de búsqueda activos.	
Condiciones de ejecución: El publicador de solicitudes, el distribuidor y los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se selecciona la opción proceso distribuido en el cliente AFIS. 2. Se ejecuta la acción Distribuir en el cliente AFIS.	
Resultado esperado: Mostrar el intervalo y la cantidad de usuarios asignados a cada servidor de búsqueda en las interfaces correspondientes (interfaz de distribución e interfaz con los detalles de un servidor específico).	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 10: CP Atender una solicitud de identificación.

Caso de prueba de aceptación	
Código: CDHU6CP1	**Historia de usuario:** Atender solicitud de identificación.
Responsable de la prueba: Royli Hernández Delgado	

Descripción: Prueba para verificar que se identifique a un individuo.
Condiciones de ejecución: El publicador de solicitudes, el distribuidor y los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.
Entrada/Pasos de ejecución: 1. Se accede a la interfaz principal del AFIS. 2. Se introduce la huella dactilar. 3. Se ejecuta la acción Identificar.
Resultado esperado: Mostrar el resultado de la identificación en la interfaz principal del AFIS, en caso de ser positiva se muestran los datos del individuo y su foto, de lo contrario se alerta de que el individuo no se ha encontrado en el banco de datos.
Evaluación de la prueba: Prueba satisfactoria.

3.9.3. Pruebas de rendimiento

El sistema desarrollado fue sometido a pruebas de rendimiento para medir el funcionamiento del AFIS una vez utilizado el servicio de distribución. Estas pruebas estuvieron enfocadas al análisis del comportamiento de los tiempos de respuesta del AFIS en dependencia de condiciones variables como el crecimiento de los datos y la escalabilidad, esta última expresada en la cantidad de servidores involucrados en la búsqueda (Ver el entorno en el que se ejecutaron estas pruebas en el Anexo Datos de pruebas de rendimiento).

Debido a la necesidad de medir el rendimiento del sistema con un banco de datos de tamaño representativo para la validación de la hipótesis planteada en la investigación, se utilizó un banco de datos con 14 mil usuarios para un total de 28 mil huellas dactilares almacenadas, pues por cada usuario se tomaron las impresiones dactilares de sus dos pulgares.

A continuación se explican los resultados obtenidos en estas pruebas, donde se hace una comparación detallada entre los tiempos de respuestas reportados por el AFIS con la centralización y con la distribución del proceso de búsqueda.

Análisis de resultados

En la Figura 15 se pueden observar los resultados obtenidos al identificar a individuos en distintos intervalos del banco de datos. Para ello el sistema de distribución empleó 3 y 6 servidores de búsqueda respectivamente.

Al aumentar los datos a procesar (aumento del número de iteraciones) el modelo centralizado relentiza la identificación con un incremento promedio de 117,0449 ms por cada 2 000 huellas a comparar, o sea, 1 000 individuos a analizar. Por el contrario, con el sistema de distribución los tiempos de respuesta

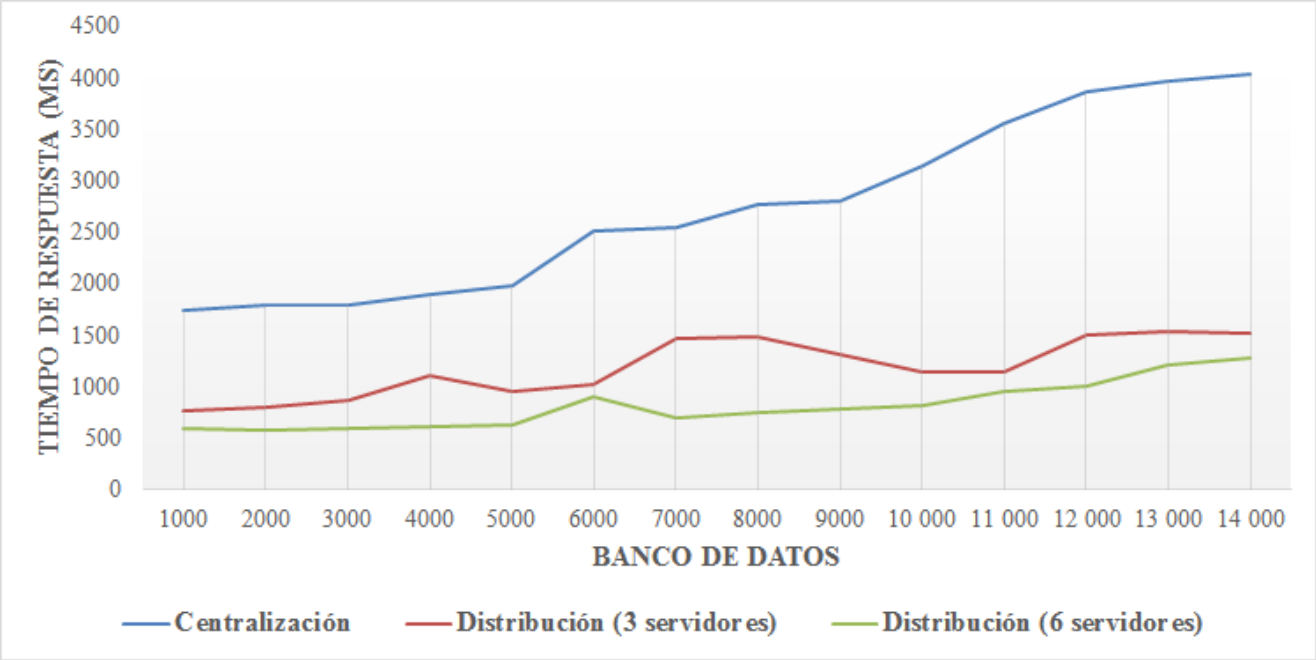


Figura 15: Comparación de tiempos de respuesta del AFIS centralizado y distribuido.

del AFIS se reducen en más de un 50 % con solo 3 servidores de búsqueda. Nótese que al aumentar el número de servidores de búsqueda disminuyeron aún más los tiempos de respuesta (Consultar los tiempos de respuestas en el Anexo Datos de pruebas de rendimiento).

Debe tenerse en consideración que los tiempos de respuesta del AFIS con el sistema de distribución están afectados por la latencia de la infraestructura de red sobre la que se ejecute el sistema. Es por ello que en la Figura 15 se perciben incrementos y decrementos bruscos en los tiempos de respuesta, en lugar de una función creciente. Aunque los resultados anteriores demuestran la efectividad de la distribución para optimizar el proceso de búsqueda, a continuación se presentarán los tiempos de búsqueda locales obviando la latencia de la red, o sea, el tiempo que demora el servidor de búsqueda donde se hace la identificación positiva[4] en culminar el proceso, obviando el tiempo que transcurre en la notificación del resultado a la capa del AFIS encargada de procesarlo (Ver Figura 16).

El promedio de tiempo perdido por la latencia de la infraestructura de red donde se realizaron las pruebas fue de aproximadamente 254,054 ms (Consultar los tiempos de búsqueda locales en el Anexo Datos de pruebas de rendimiento). Sin embargo, como se ha podido apreciar, este tiempo de demora no afectó en gran medida los tiempos de respuesta reportados por el AFIS con el sistema de distribución.

[4]Se refiere al servidor de búsqueda donde se encuentra la huella dactilar del individuo a identificar.

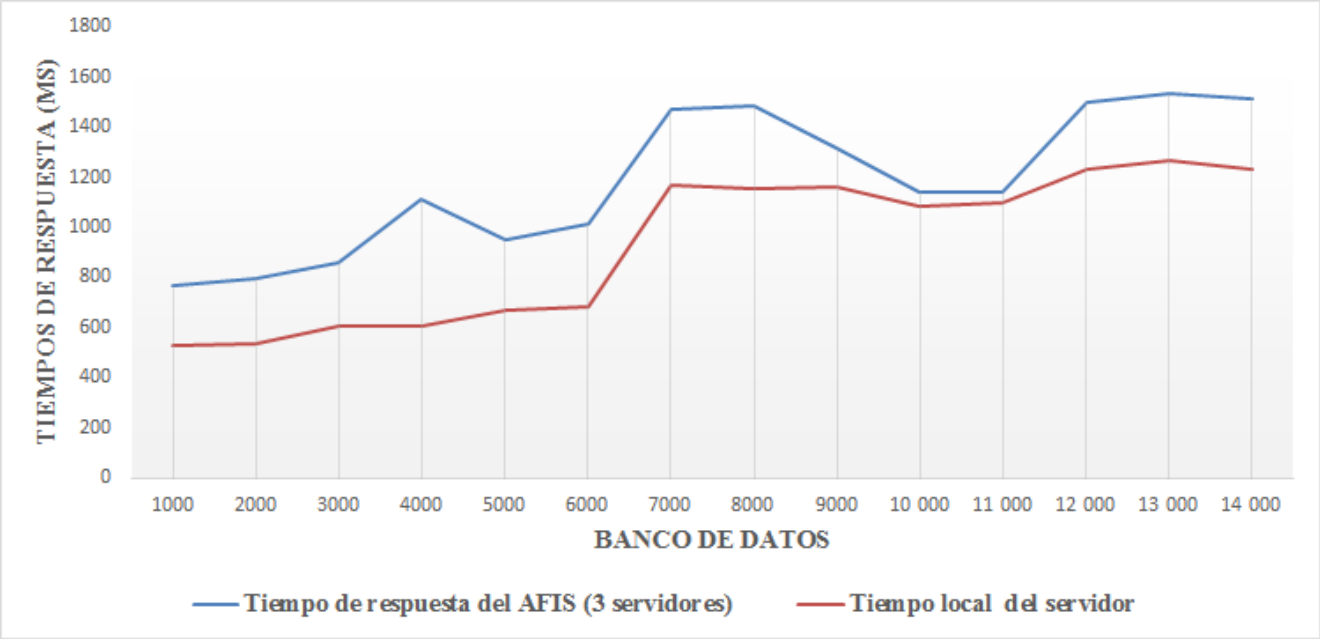


Figura 16: Tiempos de respuesta del AFIS y tiempos locales en los servidores de búsqueda.

Después de analizados todos estos datos arrojados por las pruebas de rendimiento, se puede afirmar que la hipótesis que ha regido la investigación es válida, ya que el sistema de distribución del proceso de búsqueda permite obtener mejores tiempos de respuesta en el proceso de identificación del AFIS del CISED.

3.10. Conclusiones parciales

En este capítulo se transitó a través de la codificación y prueba del sistema. El desglose de las historias de usuario en tareas de ingeniería contribuyó a facilitar el trabajo de programación al indicar específicamente las funcionalidades a desarrollar. El modelo de despliegue definido mostró con mayor claridad la distribución física del sistema sobre una arquitectura de hardware, mientras que el diagrama de componentes describió cómo interactúan los paquetes de clases y las diferentes capas del sistema dentro del marco de trabajo. Se constató que el desarrollo guiado por pruebas asegura la ejecución correcta de la solución en todo el período de implementación, disminuyendo el tiempo invertido en el ciclo compilación-ejecución. Además, las pruebas de aceptación culminaron exitosamente con la satisfacción del cliente. Finalmente las pruebas de rendimiento fueron esenciales en la validación de la hipótesis, demostrando que el sistema de distribución permite que el AFIS reporte mejores tiempos de respuesta.

Conclusiones generales

Durante el transcurso de esta investigación se realizó un análisis sobre las condiciones actuales en que se realiza el proceso de búsqueda de huellas dactilares en el AFIS desarrollado en el Departamento de Biometría del CISED, así como las principales características de los AFISs con mejores resultados en el mercado y los procesos de búsqueda asociados a estos. A partir de esta información se encontraron evidencias de que estos AFISs, dentro de los que se incluye el desarrollado en el país por DATYS, son software propietario, precisan de grandes prestaciones de hardware y utilizan el procesamiento distribuido para lograr tiempos de respuesta adecuados y una alta escalabilidad, siendo imposible reutilizar sus componentes de búsqueda.

Llegado a este punto el sistema desarrollado para la distribución del proceso de búsqueda de huellas dactilares, cobra especial importancia por el aporte económico que supone la inclusión de un AFIS eficiente en las soluciones informáticas del CISED, y particularmente su utilización en los sistemas de control de acceso dentro de la Universidad y el país. Algunas de las conclusiones obtenidas luego de terminada la primera versión del sistema para la distribución de la búsqueda son:

1. La definición de la arquitectura cliente-servidor y el uso del patrón n-capas permitieron que el sistema soporte el procesamiento distribuido, previendo el crecimiento modular y la alta escalabilidad, así como la implementación de mecanismos eficientes de tolerancia a fallos.
2. El componente encargado de la distribución que forma parte del sistema final, tiene un alto nivel de independencia, que puede ser usado por otros sistemas biométricos que solo tendrían que implementar la arquitectura específica de sus servidores de búsqueda.
3. El diseño está orientado a que los servidores de búsqueda de otro sistema biométrico que desee utilizar el servicio de distribución, se ejecuten en cualquier plataforma.
4. Se demostró que el uso y escritura de pruebas unitarias sobre partes y funciones sensibles del sistema asegura la escritura de código sin errores y aumenta la consistencia de la implementación.
5. La realización de pruebas de aceptación alfa y beta permitieron satisfacer las expectativas del cliente.
6. Las pruebas de rendimiento permitieron validar la hipótesis planteada, ya que el AFIS del CISED haciendo uso del sistema de distribución, reporta mejores tiempos de respuesta y una alta tolerancia a fallos, teniendo en cuenta la escalabilidad y el futuro crecimiento de los datos.

Recomendaciones

Después de cumplido el objetivo general y haber validado con resultados positivos y tangibles la hipótesis que rigió la investigación, se recomienda:

- Optimizar la búsqueda en los servidores dedicados a esta tarea mediante la implementación de una capa de indexación, con lo que se reducirían aún más los tiempos de respuesta.
- Utilizar un servidor de contingencia que asuma el rol del distribuidor en caso de que este falle.
- Utilizar el sistema desarrollado para la distribución de la búsqueda asociada a otros sistemas de identificación que son productos del Departamento de Biometría del CISED, para lo cual es necesario estructurar e implementar los servidores de búsqueda específicos para cada uno de ellos.

Referencias bibliográficas

[1] **Maltoni, Davide y Maio, Dario**. *Handbook of fingerprint recognition*. Second Edition. Londres : Springer, 2009. ISBN: 978-1-84882-253-5.

[2] **Muñoz, Almudena Lindoso**. *Tesis Doctoral: Contribución al reconocimiento de huellas dactilares mediante técnicas de correlación y arquitecturas hardware para el aumento de prestaciones*. Madrid : Universidad Carlos III de Madrid. Departamento de Tecnología Electrónica, 2009.

[3] *Reconocimiento de Huellas Dactilares Usando*. **Aguilar, Gualberto**. Diciembre 2008, Rev. Fac. Ing. Univ. Antioquia, págs. 101-109. 46. [En línea] [Citado el: 12 de Enero de 2013.] http://jaibana.udea.edu.co/grupos/revista/revistas/nro046/46_10.pdf.

[4] **Coulouris, George**. *Sistemas Distribuidos*. Madrid : Addison Wesley, 2001.

[5] **Vallecillo, Antonio Moreno**. *RM-ODP: El Modelo de Referencia de ISO para el procesamiento abierto distribuido*. [En línea] [Citado el: 12 de Enero de 2013.] http://www.lcc.uma.es/~av/Publicaciones/00/odpesp.pdf.

[6] **Rojo, J. Oscar**. *Introducción a los Sistemas Distribuidos*. [En línea] [Citado el: 13 de Enero de 2013.] http://augcyl.org/?page_id=231.

[7] **Puder, Arno**. *Distributed Systems Architecture: A Middleware Approach*. s.l. : Morgan Kaufmann Publishers, 2005, págs. 21-24.

[8] **Sommerville, Ian**. Arquitecturas de sistemas distribuidos. *Ingeniería del software 7ma Edición*. 2005, págs. 242-244.

[9] **Rob, Peter y Coronel, Carlos**. Clasificaciones del middleware. *Sistemas de bases de datos: diseño, implementación y administración*. s.l. : EDICIONES PARANINFO, S.A, 2004, págs. 595-596.

[10] **Caponi, Marcelo; Rodriguez, Pablo Defino y Zamudino, Pablo**. *Mensajería en Sistemas de Información*. Montevideo, Uruguay : s.n., 2008.

[11] **Woolf, Gregor Hohpe**. *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. s.l. : Addison-Wesley Professional, Octubre 2003.

[12] **Roebuck, Kevin**. *Advanced Message Queuing Protocol (Amqp): High-Impact Strategies - What You Need to Know: Definitions, Adoptions, Impact, Benefits, Maturity, Vendors*. 2011.

[13] **Tarkoma, Sasu**. *Standars and Products. Publish / Subscribe Systems: Design and Principles*. s.l. : John Wiley & Sons, 2012, págs. 127-129

[14] **Komarinski, Peter**. *History of Automated Identification Systems. Automated Fingerprint Identification System (AFIS).* s.l. : Elsevier Academic Press, 2005.

[15] **Podio, Fernando L., Yaga, Dylan J. y McGinnis, Christofer J**. *Conformance Testing Methodology for ANSI/NIST-ITL 1-2011, Data Format for the Interchange of Fingerprint, Facial & Other Biometric Information (Release 1.0).* s.l. : The National Institute of Standards and Technology (NIST), 2012.

[16] **Fillon, Eve**. *MorphoTrak Chosen as Biometric Provider for FBI Next Generation Identification Program.* Alexandria : MorphoTrak, SAFRAN Group, 2009.

[17] 18. **Innovatrics**. ExpressID AFIS. [En línea] Innovatrics, 2013. [Citado el: 16 de Enero de 2013.] http://www.innovatrics.com/products/expressid-afis.

[18] 19. -. Awards. [En línea] Innovatrics, 2013. [Citado el: 16 de Enero de 2013.] http://www.innovatrics.com/about-us/awards.

[19] **Datys**. Sistema automatizado de Identificación civil. [En línea] Datys, 2011. [Citado el: 16 de Enero de 2013.] http://www.datys.cu/wpinfproducto.aspx?22.

[20] -. Especificaciones Técnicas BIOMESYS AFIS Civil. [En línea] [Citado el: 16 de Enero de 2013.] http://www.datys.cu/docs/Documentaci%C3%B3n%20Productos/Biomesys%20AFIS/Civil/Especificaciones%20T%C3%A9cnicas/Biomesys%20AFIS%20Civil%20V2.0%20ET_20.04.10_%20E03%20_Es_D.pdf.

[21] **Važan, Robert**. SourceAFIS. [En línea] [Cited: 1 16, 2013.] http://www.sourceafis.org/blog/datasheet/.

[22] **Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A**. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES. [En línea] [Citado el: 17 de Enero de 2013.] http://www.google.com.cu/url?sa=t&rct=j&q=metodologias+de+desarrollo+de+software+pesadas&source=web&cd=6&ved=0CEkQFjAF&url=http%3A%2F%2Fadonisnet.files.wordpress.com%2F2008%2F06%2Farticulo-metodologia-de-sw-formato.doc&ei=bTn0ULbfJoqH0QHcmIDACg&usg=AFQjCN.

[23] **Kruchten, Philippe**. *The Rational UnifiedProcess An Introduction.* s.l. : Addison WesleY, 2001.

[24] **Letelier, Patricio**. *Rational Unified Process (RUP).* [En línea] [Citado el: 17 de Enero de 2013.] http://eva.uci.cu/mod/resource/view.php?id=9303&subdir=/Metodologias/RUP.

[25] **Sommerville, Ian**. Procesos del software. *Ingenieria del Software 7ma Edicion.* 2005, págs. 76-78.

[26] **Kroll, Per y Kruchten, Philippe**. *The Rational Unified Process Made Easy: A Practitioner's Guide to the Rup*. s.l. : Addison-Wesley Professional, 2003.

[27] **Canós, José H., Letelier, Patricio y Penadés, María Carmen**. Métodologías Ágiles en el Desarrollo de Software. [En línea] [Citado el: 17 de Enero de 2013.] http://www.funtec.org.ar/pdf/Pdf_semin_completo_MetodologiasAgiles.pdf.

[28] **Beck, K**. *Extreme Programming Explained. Embrace Change*. s.l. : Addison Wesley, 1999.

[29] **Schwaber, Ken y Beedle, Mike**. *The Agile software development with Scrum*. s.l. : Prentice Hall, 2008.

[30] **Schwaber, Ken**. *Agile Project Management with Scrum*. s.l. : Microsoft Press, 2008.

[31] **Rumbauhg, James, Jacobson, Ivar y Booch, Grady**. *El lenguaje Unificado de Modelado. Manual de referencia*. s.l. : Addison Wesley, 2007.

[32] **Gónzalez, Pascual López, Gónzalez, Ana Amelia López y Gallud, José Antonio Lázaro**. *Herramientas CASE ¿Cómo incorporarlas con éxito a nuestra organización?* España : Universidad de Castilla-La Mancha, 2005.

[33] **Nobrega, Maria De**. Herramientas CASE: Rational Rose. [En línea] [Citado el: 17 de Enero de 2013.] http://curso_sin2.blogia.com/2005/060401-herramientas-case-rational-rose.-por-maria-de-nobrega.php.

[34] **Paradigm, Visual**. Viual Paradigm for UML. [En línea] Visual Paradigm. [Citado el: 17 de Enero de 2013.] http://www.visual-paradigm.com/product/vpuml/.

[35] **Baeza, Pablo Nicolás**. Visual Paradigm DB Visual ARCHITECT SQL. [En línea] [Citado el: 17 de Enero de 2013.] http://www.docstoc.com/docs/96492173/Visual-Paradigm-Studio.

[36] **Rodríguez, Jesús J. Sala**. *Lenguajes de programación. Introducción a la programación: teoría y práctica*. s.l. : Editorial Club Universitario, 2003

[37] **Cerezo, Yolanda López**. *Iniciación a la programación en C#: un enfoque práctico*. s.l. : Delta Publicaciones, 2006.

[38] **Java**. [En línea] [Citado el: 17 de Enero de 2013.] http://arantxa.ii.uam.es/~castells/docencia/poo/2-java-esp.pdf.

[39] **Ramos, Isidro Salavert y Lozano, María Dolores Pérez**. *Entornos de Desarrollo Integrado. Ingeniería Del Software Y Bases de Datos: Tendencias Actuales*. s.l. : Univ de Castilla La Mancha, 2000.

[40] **Entornos integrados de desarrollo**. [En línea] [Citado el: 17 de Enero de 2013.] http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion.

[41] **NetBeans**. [En línea] NetBeans. [Citado el: 18 de Enero de 2013.] http://netbeans.org/index_es.html.

[42] **Boudreau, Tim**. *NetBeans: The Definitive Guide*. s.l. : O'Reilly Media, 2002.

[43] **FAQ-MAC**. [En línea] [Citado el: 18 de Enero de 2013.] http://www.faq-mac.com/noticias/netbeansorg-detalla-nuevo-mapa-marco-herramientas-java/5152.

[44] **Avery, James**. *Visual Studio Hacks: Tips & Tools for Turbocharging the IDE*. s.l. : O'Reilly Media, 2005.

[45] **Dunaway, Robert B**. *The Book of Visual Studio.Net: Aguide for Developers*. s.l. : No Starch Press, 2002.

[46] **Bryant, Russell**. [En línea] 2012. [Citado el: 18 de Enero de 2013.] http://fedoraproject.org/wiki/Features/OpenStack_using_Qpid.

[47] **Videla, Alvaro y Williams, Jason J.W**. *RabbitMQ in Action Distributed Messaging For Enyone*. s.l. : Manning Publications Company, 2012.

[48] **RabbitMQ**. [En línea] RabbitMQ. [Citado el: 19 de Enero de 2013.] http://www.rabbitmq.com/features.html.

[49] **Foundation, Apache Software**. ApacheQpid. [En línea] Apache Software Foundation, 2012. [Citado el: 18 de Enero de 2013.] http://qpid.apache.org/.

[50] –. **ApacheQpid**. [En línea] Apache Software Foundation. [Citado el: 18 de Enero de 2013.] https://cwiki.apache.org/qpid/faq.html.

[51] **Introduction to OpenAMQ** . OpenAMQ. [En línea] OpenAMQ. [Citado el: 18 de Enero de 2013.] http://openamq.wikidot.com/doc:user-1-introduction.

[52] **Asenjo, Jorge Sánchez**. Apuntes Completos sobre Sistemas gestores de Bases de Datos. [En línea] 2009. http___ubuntuone.pdf---http://ubuntuone.com/p/sqt/.

[53] **Martinez, Rafael**. Sobre PostgreSQL. www.postgresql.org.es. [En línea] 2010. http://www.postgresql.org.es/sobre_postgresql.

[54] **The PostgreSQL Global Development Group**. PostgreSQL: PostgreSQL Featured Users. [En línea] 2013. http://www.postgresql.org/about/users/.

[55] **Riggs, Simon y Krosing, Hannu**. *PostgreSQL 9 Administration Cookbook. Birmingham* : Packt Publishing Ltd., 2010. pág. 8.

[56] **Casillas, Luis Alberto Santillán, y otros**. Bases de datos en MySQL. [En línea] 2009. http://uoc.edu/computer-science-technology-and-multimedia/bases-de-datos/bases-de-datos/P06_M2109_02151.pdf.

[57] Panorámica de MySQL AB. [En línea] 2013. http://dev.mysql.com/doc/refman/5.0/es/what-is-mysql-ab.html.

[58] **Oracle**. About MySQL. [En línea] 2013. http://www.mysql.com/about/.

[59] **Larman, Craig**. *UML y Patrones. Introducción al análisis y diseño orientado a objetos*. México : Prentice Hall, 1999.

[60] **Alonso, Fernando Amo, Normand, Martínez Loïc A. and Segovia, Francisco Javier Pérez**. El Proceso Unificado está centrado en la arquitectura. *Introducción a la ingeniería del software*. s.l. : Delta Publicaciones, 2005, pp. 337-338.

[61] **Kendall, Kenneth E. and Kendall, Julie E**. Tecnología Cliente Servidor. *Análisis y diseño de sistemas*. s.l. : Pearson Educación, 2005, pp. 622-624.

[62] **Dormido, Sebastián**. *Procesamiento paralelo: teoría y programación*. Madrid : Sanz y Torres, 2003.

Bibliografía consultada

Ratha, Nalini y Bolle, Ruud. *Automatic Fingerprint Recognition Systems*. s.l. : Springer, 2004.

Monsó, Juliá i Bustio. *Sistemas de identificación y control automáticos* (Tomo II). s.l. : Marcombo, 1994 .

Policing in Modern Society. s.l. : Gulf Professional Publishing, 1999 .

Buquet, Alain. *Manual de criminalística moderna: la ciencia y la investigación de la prueba*. s.l. : Siglo XXI, 2006.

Komarinski, Peter. *Automated Fingerprint Identification Systems (AFIS)*. s.l. : Academic Press, 2005.

Puder, Arno, Römer, Kay y Pilhofer, Frank. *Access Online via Elsevier: A Middleware Approach*. s.l. : Elsevier, 2011.

Mahmoud, Qusay. *Middleware for Communications*. s.l. : John Wiley & Sons, 2005.

Sánchez, José Acedo. *Instrumentación y control de avanzado de procesos*. s.l. : Ediciones Díaz de Santos, 2006 .

Stamelos, Ioannis G. y Sfetsos, Panagiotis. *Agile Software Development Quality Assurance*. s.l. : Idea Group Inc (IGI), 2007 .

Ambler, Scott. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. s.l. : John Wiley & Sons, 2002.

Sommerville, Ian. *Ingeniería del software 7ma Edición*. 2005.

Rob, Peter y Coronel, Carlos.*Sistemas de bases de datos: diseño, implementación y administración*. s.l. : EDICIONES PARANINFO, S.A, 2004.

Alonso, Fernando Amo, Normand, Martínez Loïc A. and Segovia, Francisco Javier Pérez. *Introducción a la ingeniería del software*. s.l. : Delta Publicaciones, 2005.

Kendall, Kenneth E. and Kendall, Julie E. *Análisis y diseño de sistemas*. s.l. : Pearson Educación, 2005.

Hunt, John. *Agile software construction*. s.l. : Springer, 2006.

PostgreSQL. *PostgreSQL: A Comprehensive Guide to Building, Programming, and Administering PostgresSQL Databases*. s.l. : Sams Publishing, 2003 .

Group, Postgresql Global Development. *Postgresql 9.0 Official Documentation - Volume I the Sql Language*. s.l. : Fultus Corporation, 2011. Vol. I.

–.*Postgresql 9.0 Official Documentation - Volume II Server Administration*. s.l. : Fultus Corporation, 2011.

–. *Postgresql 9.0 Official Documentation - Volume III Server Programming*. s.l. : Fultus Corporation, 2011 . Vol. III.

–. *Postgresql 9.0 Official Documentation - Volume Iv Reference*. s.l. : Fultus Corporation, 2011. Vol. IV.

Russell, Jesse y Cohn, Ronald. *Rabbitmq*. s.l. : Book on Demand, 2012.

Teniente, Ernest López, et al. *Diseño de sistemas software en UML*. s.l. : Univ. Politèc. de Catalunya, 2004 .

Glosario de términos

API: Acrónimo de Applications Programming Interface, es un conjunto de funciones brindadas por una librería que puede ser utilizado por otro software como una capa de abstracción.

DPI: Acrónimo de Dots per Inch, se refiere al número de puntos individuales que pueden ser situados en una línea en el rango de una pulgada (2,54 cm).

JMS: Acrónimo de Java Message Service, es un estándar de mensajería creado por la Sun Microsystems para el uso de colas de mensajes.

JMX: Acrónimo de Java Management eXtensions, es una tecnología que especifica una arquitectura de gestión para administrar servicios y aplicaciones Java.

JSON: Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos que puede ser interpretado por cualquier lenguaje de programación. Es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

JVM: Acrónimo de Java Virtual Machine, es una máquina virtual de proceso nativo ejecutable en varias plataformas, que permite interpretar y ejecutar instrucciones expresadas en un código binario especial generado por el compilador del lenguaje Java.

Memoria RAM: Memoria temporal disponible para los datos que se están procesando y programas que se están ejecutando en el momento.

Multiplataforma: Se refiere a los programas informáticos que pueden funcionar en diversas plataformas.

MVCC: Acrónimo de Multi-Version Concurrency Control, permite el acceso a una tabla sin necesidad de bloqueos por aunque otros procesos estén escribiendo simultáneamente en ella.

Plataforma: Es un término que define una arquitectura de hardware y de software con la que son compatibles determinados lenguajes de programación, sistemas operativos y software en general.

SDK: Acrónimo de Software Development Kit, es un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

Tiempo de respuesta: Tiempo que necesita el sistema para completar una única tarea a partir del momento en que esta fue requerida.

Velocidad de procesamiento de CPU: Número de instrucciones que el procesador puede ejecutar por segundo.

XML: Acrónimo de eXtensible Markup Language, es un lenguaje de marcas que permite estructurar documentos para el intercambio de información entre diferentes plataformas.

Anexo Operacionalización de variables

Operacionalización de variables.

Operacionalización de Variables			
Variable	**Tipo de variable**	**Dimensiones**	**Indicadores**
Sistema de distribución	Independiente	Eficiencia	• Productividad • Disponibilidad • Utilización de recursos del sistema
		Escalabilidad	• Escalabilidad vertical • Escalabilidad horizontal • Escalabilidad en carga
		Tratamiento de fallos	• Tiempo esperado hasta la detección de un fallo del sistema • Tiempo medio de recuperación ante fallos • Tiempo medio entre fallos
Tiempo de respuesta	Dependiente	Capacidad del canal	• Ancho de banda y tiempo de transmisión de datos • Velocidad de transmisión de datos, infraestructura de red donde se despliega el sistema
		Latencia	• Velocidad de descompresión y compresión de datos • Tamaño de los mensajes enviados • Frecuencia de requerimientos

Conceptualización de variables

A continuación se definen las variables involucradas en la operacionalización:

Eficiencia: Óptima utilización de los recursos disponibles para la obtención de resultados deseados.

Productividad: Número de tareas que puede completarse en un intervalo de tiempo determinado.

Disponibilidad: Grado en el cual un sistema se encuentra específicamente operable y en estado confirmable.

Escalabilidad: Capacidad del sistema de continuar siendo efectivo cuando incrementa significativamente la carga de trabajo debido al aumento del número de usuarios y recursos.

Escalabilidad vertical: Aumento de recursos en un nodo del sistema.

Escalabilidad horizontal: Aumento de nodos del sistema.

Escalabilidad en carga: Crecimiento de los datos y aumento de peticiones.

Tratamiento de fallos: Detección, tratamiento y recuperación del sistema ante un fallo.

Tiempo de respuesta: Tiempo que necesita el sistema para completar una única tarea a partir del momento en que esta fue requerida.

Capacidad del canal: Límite superior de la capacidad de información que puede ser confiablemente transmitida sobre el canal de comunicación.

Latencia: Suma de retardos temporales dentro de una red[5].

[5]Un retardo es producido por la demora en la propagación y transmisión de paquetes dentro de la red.

Anexo Historias de usuario

HU Detectar servidores de búsqueda.

Historia de usuario	
Número: CDHU01	**Usuario:** Royli Hernández Delgado
Nombre: Detectar servidores de búsqueda.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 1
Programador responsable: Melvis Machin Armas	
Descripción: La presente historia de usuario tiene como objetivo la detección de los servidores de búsqueda que estén disponibles y cuyos tipos estén registrados en el sistema. Para ello se les enviará un mensaje al que responderán los servidores con su nombre en la red.	
Observaciones: Se debe realizar para chequear periódicamente los servidores de búsqueda activos.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

HU Modificar el estado de trabajo de un servidor de búsqueda.

Historia de usuario	
Número: CDHU02	**Usuario:** Melvis Machin Armas
Nombre: Modificar el estado de trabajo de un servidor de búsqueda.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: Royli Hernández Delgado	
Descripción: La presente historia de usuario tiene como objetivo cambiar el estado de un servidor de búsqueda. Los estados pueden ser: activo si el servidor está involucrado en la búsqueda o disponible si no lo está.	
Observaciones: Inicialmente todos los servidores detectados son activos.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

Obtener las propiedades de los servidores de búsqueda.

	Historia de usuario
Número: CDHU03	**Usuario:** Royli Hernández Delgado
Nombre: Obtener las propiedades de hardware de los servidores de búsqueda.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Melvis Machin Armas	
Descripción: La presente historia de usuario tiene como objetivo obtener las propiedades de los servidores de búsqueda, como son memoria RAM, velocidad de procesamiento de CPU y valor de aptitud.	
Observaciones: Se debe realizar siempre que se necesite distribuir el banco de datos entre los servidores de búsqueda.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

HU Calcular la capacidad de carga de los servidores de búsqueda.

	Historia de usuario
Número: CDHU04	**Usuario:** Melvis Machin Armas
Nombre: Calcular la capacidad de carga de los servidores de búsqueda.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Medio
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: Royli Hernández Delgado	
Descripción: La presente historia de usuario tiene como objetivo calcular la capacidad de carga de cada servidor de búsqueda considerando sus propiedades de memoria RAM disponible y velocidad de procesamiento del CPU.	
Observaciones: Se debe realizar siempre que se necesite distribuir el banco de datos entre los servidores de búsqueda.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

HU Gestionar tipos de servidores de búsqueda.

	Historia de usuario
Número: CDHU07	**Usuario:** Royli Hernández Delgado

Nombre: Gestionar tipos de servidores de búsqueda.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 2	**Iteración asignada:** 4
Programador responsable: Melvis Machin Armas	
Descripción: La presente historia de usuario tiene como objetivo adicionar y eliminar un tipo de servidor de búsqueda.	
Observaciones: En el caso de la adición se generará una llave que identifique a los servidores de búsqueda de este tipo. Debe actualizase la detección de los servidores de búsqueda de acuerdo a los cambios ocurridos en el sistema.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

HU Gestionar conexiones a bancos de datos.

	Historia de usuario
Número: CDHU08	**Usuario:** Melvis Machin Armas
Nombre: Gestionar conexiones a bancos de datos.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Royli Hernández Delgado	
Descripción: La presente historia de usuario tiene como objetivo adicionar y eliminar la conexión a un banco de datos, así como modificar sus parámetros. Cada conexión está determinada por un tipo de servidor de búsqueda.	
Observaciones: Facilita la configuración de la conexión para que un servidor de búsqueda acceda al banco de datos del sistema biométrico.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

HU Generar logs del sistema.

	Historia de usuario
Número: CDHU09	**Usuario:** Royli Hernández Delgado
Nombre: Generar logs del sistema.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo

Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Melvis Machin Armas	
Descripción: La presente historia de usuario tiene como objetivo generar logs del sistema que contendrán información referente a las solicitudes que se han atendido.	
Observaciones: Cada petición atendida, tanto de distribución como de identificación, debe ser registrada en los logs.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

HU Configurar opciones de distribución.

	Historia de usuario
Número: CDHU10	**Usuario:** Melvis Machin Armas
Nombre: Configurar opciones de distribución.	
Prioridad en negocio: Baja	**Riesgo en desarrollo:** Bajo
Puntos estimados: 1	**Iteración asignada:** 4
Programador responsable: Royli Hernández Delgado	
Descripción: La presente historia de usuario tiene como objetivo configurar determinados parámetros que intervienen en la distribución, como son: cantidad de peticiones a realizar sobre una cola de mensajes, tiempo de espera mientras se chequea una cola de mensajes y tiempo de detección.	
Observaciones: Estos parámetros deben tener establecidos valores por defecto.	
Requisitos no funcionales: Software, Seguridad	
Asuntos pendientes: Ninguno	

Anexo Tarjetas CRC

Tarjeta CRC Comunicador con intercambiador de tipo direct.

Comunicador con intercambiador de tipo direct	
Responsabilidades	**Colaboradores**
Define la responsabilidad enviar un mensaje.	• Mensaje

Tarjeta CRC Comunicador con intercambiador de tipo fanout.

Comunicador con intercambiador de tipo fanout	
Responsabilidades	**Colaboradores**
Define la responsabilidad enviar un mensaje.	• Mensaje
Subscribir una cola de mensaje a un intercambiador.	–

Tarjeta CRC Manejador de peticiones.

Manejador de peticiones	
Responsabilidades	**Colaboradores**
Detectar servidores de búsqueda activos.	• Comunicador con intercambiador de tipo fanout
Obtener las propiedades de los servidores de búsqueda.	• Comunicador con intercambiador de tipo fanout
Enviar petición de distribución.	• Comunicador con intercambiador de tipo direct
Enviar petición de identificación.	• Comunicador con intercambiador de tipo fanout

Tarjeta CRC Controlador de peticiones.

Controlador de peticiones	
Responsabilidades	**Colaboradores**
Procesar la orden de obtener propiedades.	• Comunicador con intercambiador de tipo direct • Servidor de búsqueda

Procesar la orden de cargar el banco de datos.	• Comunicador con intercambiador de tipo direct • Servidor de búsqueda
Procesar la orden de realizar una identificación.	• Comunicador con intercambiador de tipo fanout • Servidor de búsqueda
Ejecutar tratamiento de fallos.	• Servidor de búsqueda

Tarjeta CRC Propiedades de cómputo.

Propiedades de cómputo	
Responsabilidades	**Colaboradores**
Registrar velocidad de CPU	–
Registrar memoria RAM	–
Registrar nombre en la red	–

Anexo Diagrama de clases del diseño

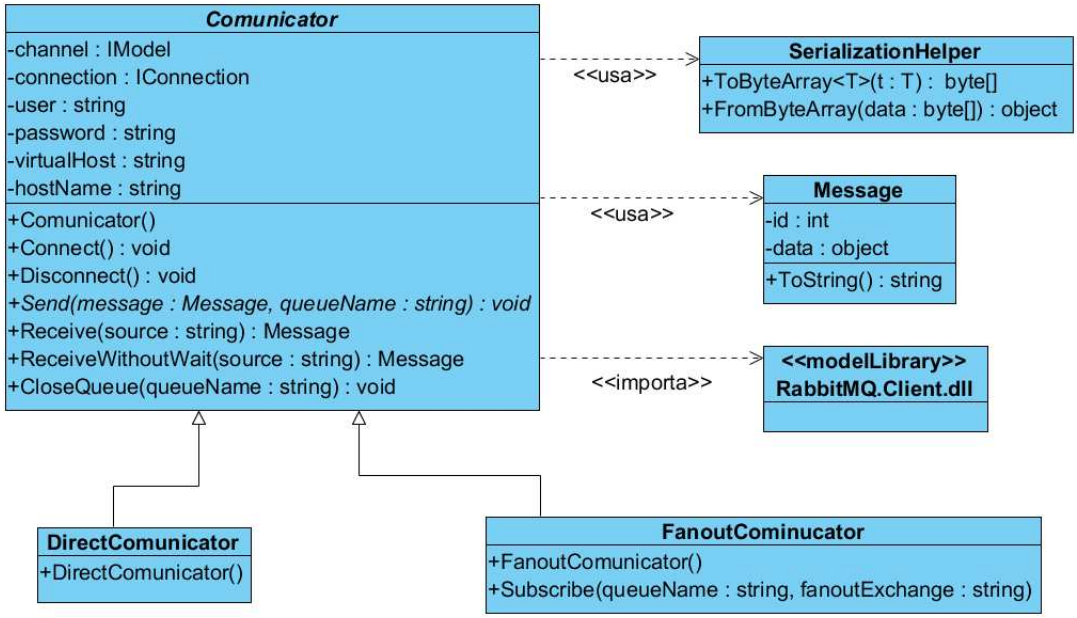


Diagrama de clases de RabbitMQManagement.dll.

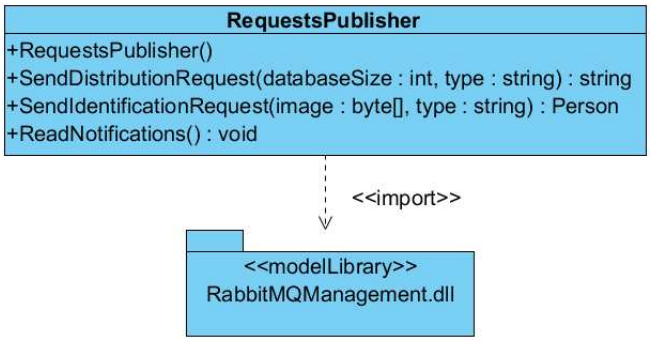


Diagrama de clases de RequestsPublisher.dll.

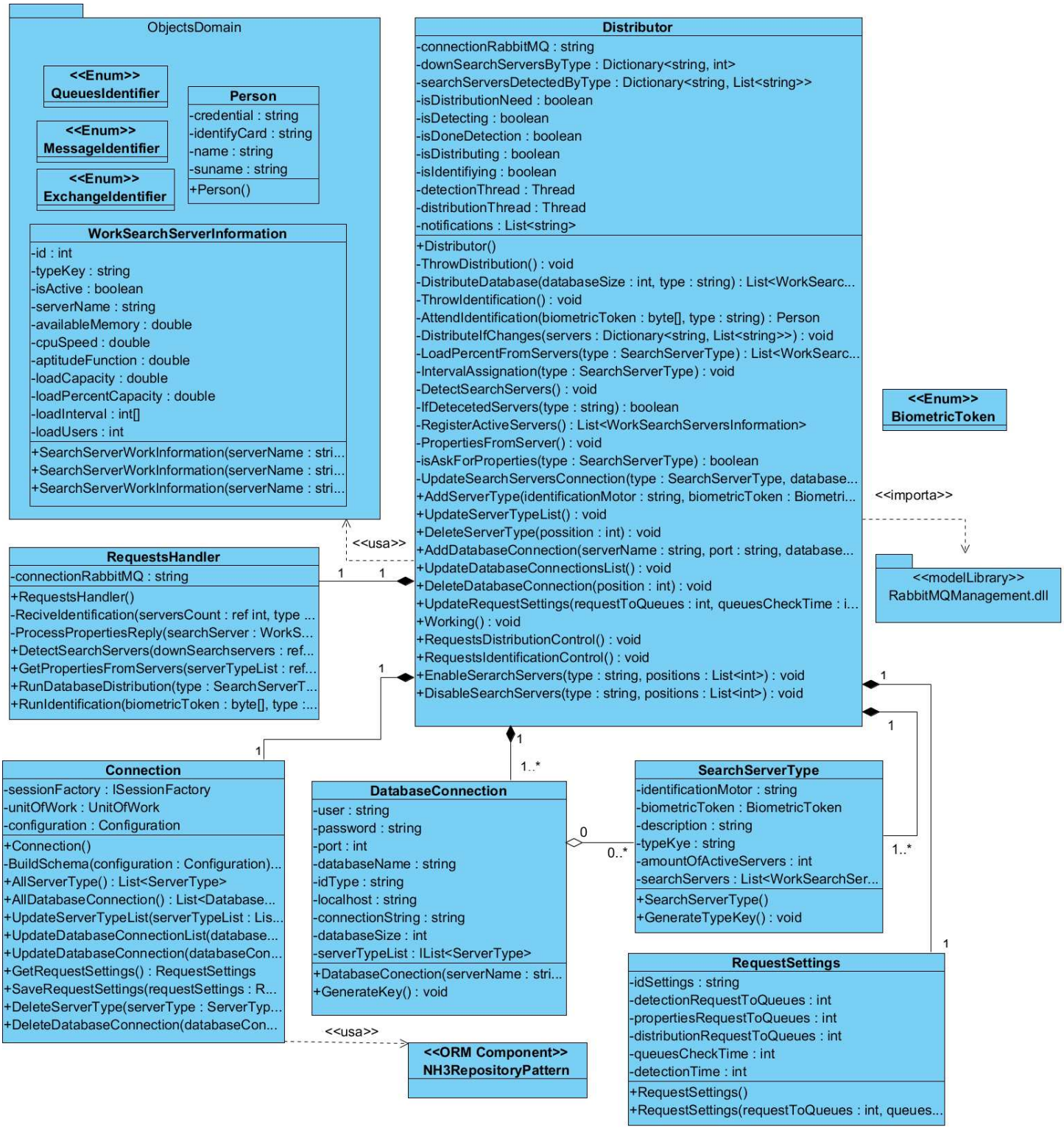


Diagrama de clases del distribuidor.

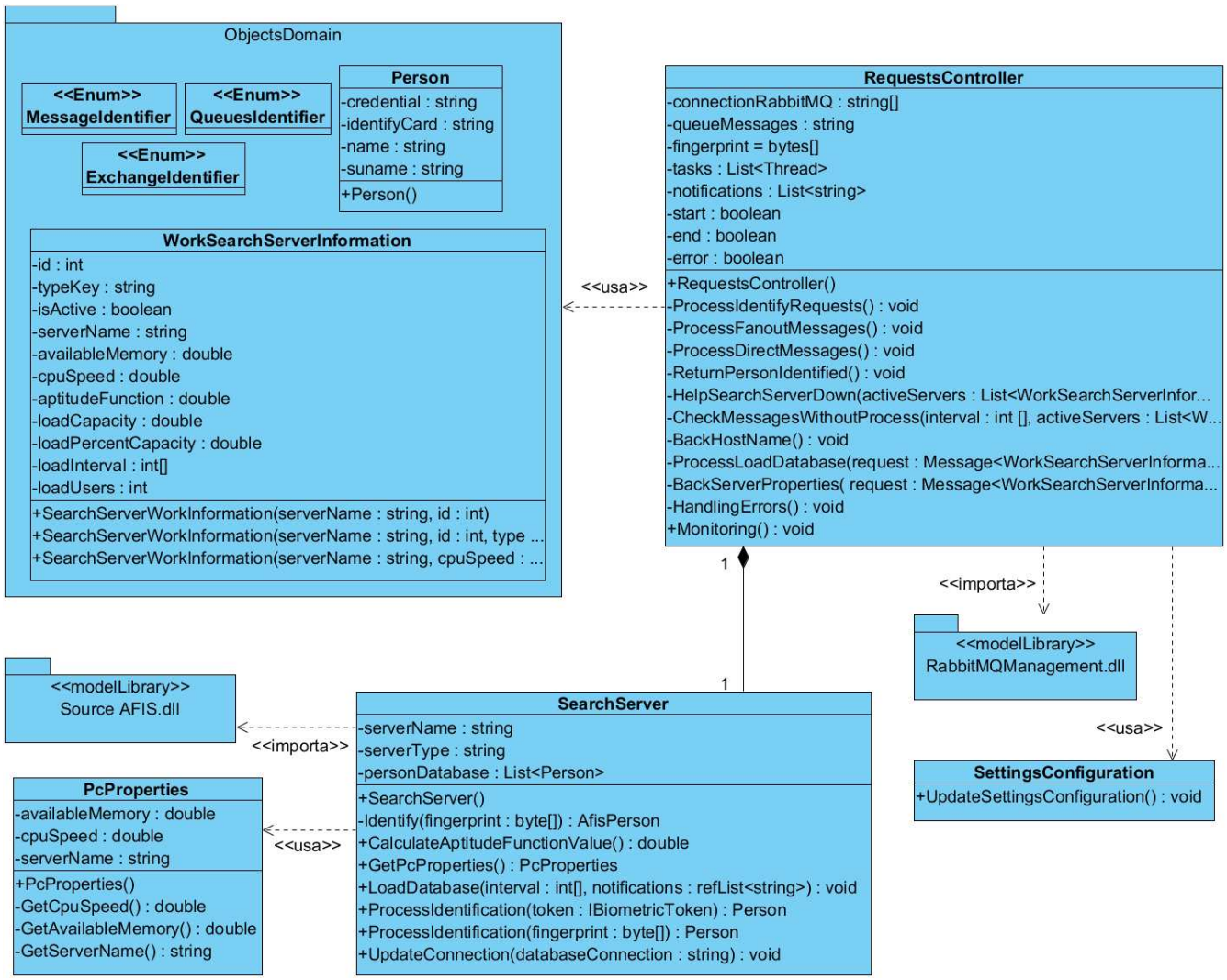


Diagrama de clases de un servidor de búsqueda.

Anexo Modelo de datos del AFIS

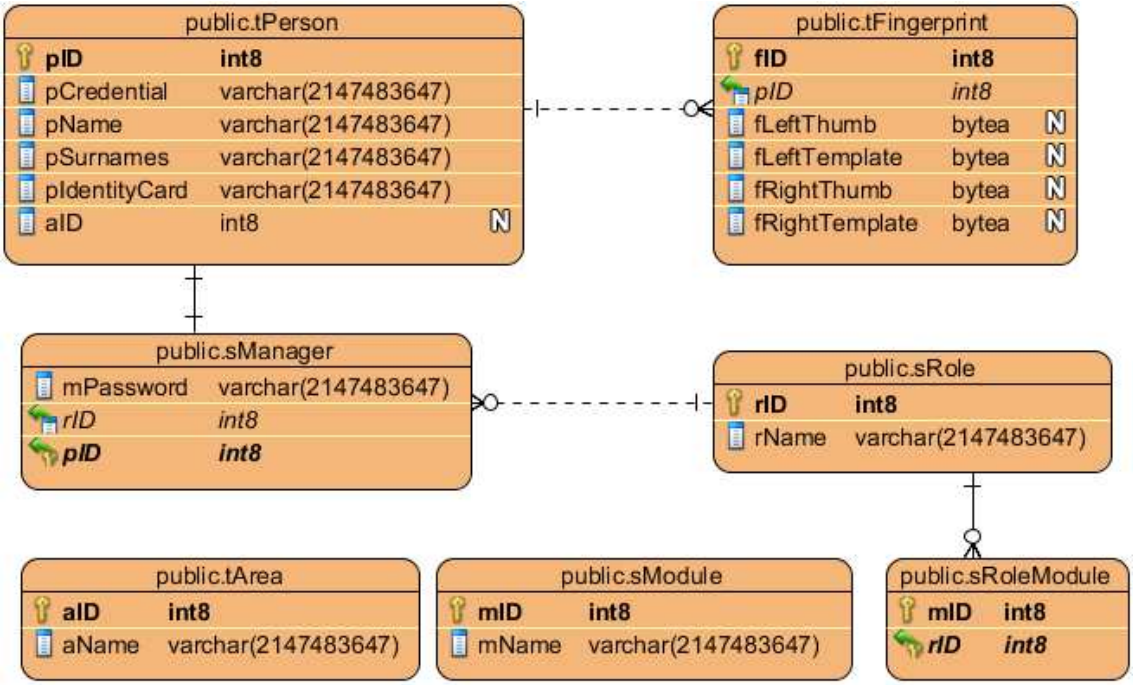


Diagrama entidad-relación del AFIS.

Anexo Tareas de ingeniería

Tareas de ingeniería de la primera iteración.

Iteración 1	
Historia de usuario	**Tareas**
Detectar servidores de búsqueda.	1. Enviar un mensaje identificado como una petición de disponibilidad a todos los servidores de búsqueda. 2. Enviar un mensaje identificado como respuesta de disponibilidad desde los servidores de búsqueda con su nombre en la red y tipo. 3. Esperar y procesar las respuestas de los servidores de búsqueda, registrando sus datos.
Modificar el estado de trabajo de un servidor de búsqueda.	1. Procesar la selección del usuario. 2. Cambiar el estado de los servidores de búsqueda.

Tareas de ingeniería de la segunda iteración.

Iteración 2	
Historia de usuario	**Tareas**
Obtener las propiedades de los servidores de búsqueda.	1. Registrar la memoria RAM disponible y la velocidad de procesamiento del CPU en cada servidor de búsqueda. 2. Calcular la función de aptitud de cada servidor de búsqueda. 3. Enviar un mensaje identificado como una petición de propiedades a todos los servidores de búsqueda. 4. Enviar un mensaje identificado como respuesta de propiedades desde los servidores de búsqueda. 5. Esperar y procesar las respuestas de los servidores de búsqueda, registrando sus propiedades de hardware y valor de aptitud.
Calcular la capacidad de carga de los servidores de búsqueda.	1. Consultar valores de aptitud de los servidores de búsqueda. 2. Calcular la capacidad de carga de los servidores de búsqueda con sus valores de aptitud.

Tareas de ingeniería de la cuarta iteración.

Iteración 4	
Historia de usuario	**Tareas**
Gestionar tipos de servidores de búsqueda.	1. Adicionar un tipo de servidor de búsqueda. 2. Generar la llave de identificación correspondiente al tipo de servidor de búsqueda adicionado. 3. Eliminar un tipo de servidor de búsqueda.
Gestionar conexiones a bancos de datos.	1. Adicionar la conexión a un banco de datos. 2. Modificar la conexión a un banco de datos. 3. Eliminar la conexión a un banco de datos.
Generar logs del sistema.	1. Acceder al fichero de logs. 2. Escribir información.
Configurar opciones de distribución	1. Procesar la configuración del usuario. 2. Guardar cambios registrados.

Anexo Código y resultados de pruebas unitarias

```
        /// <summary>
        ///A test for RegisterActiveServers
        ///</summary>
        [TestMethod()]
        public void RegisterActiveServersTest()
        {
            Distributor.Distributor target = new Distributor.Distributor(); // TODO: Initialize to an appropriate value
            List<InformationSearchServers> expected = new List<InformationSearchServers>(); // TODO: Initialize to an a
            List<InformationSearchServers> actual;
            actual = target.RegisterActiveServers();
            Assert.AreNotEqual(expected, actual);
        }
    }
```

100 %

Test Results

Royli@BIOIDEN5 2013-05-06 14:4 | Run | Debug | Group By: [None] | [All Columr

Test run completed Results: 1/1 passed; Item(s) checked: 1

Result	Test Name	Project	Error Message
Passed	RegisterActiveServersTest	TestDistribution	

PU Detectar servidores de búsqueda.

```
        /// <summary>
        ///A test for PropertiesFromServer
        ///</summary>
        [TestMethod()]
        public void PropertiesFromServerTest()
        {
            Distributor.Distributor  target = new Distributor.Distributor(); // TODO: Initialize to an appropriate valu
            target.Work();
            Thread.Sleep(5000);
            List<InformationSearchServers> expected = new List<InformationSearchServers>();// TODO: Initialize to an a
            List<InformationSearchServers> actual;
            actual = target.PropertiesFromServer();
            Assert.AreNotEqual(expected, actual);
        }
```

100 %

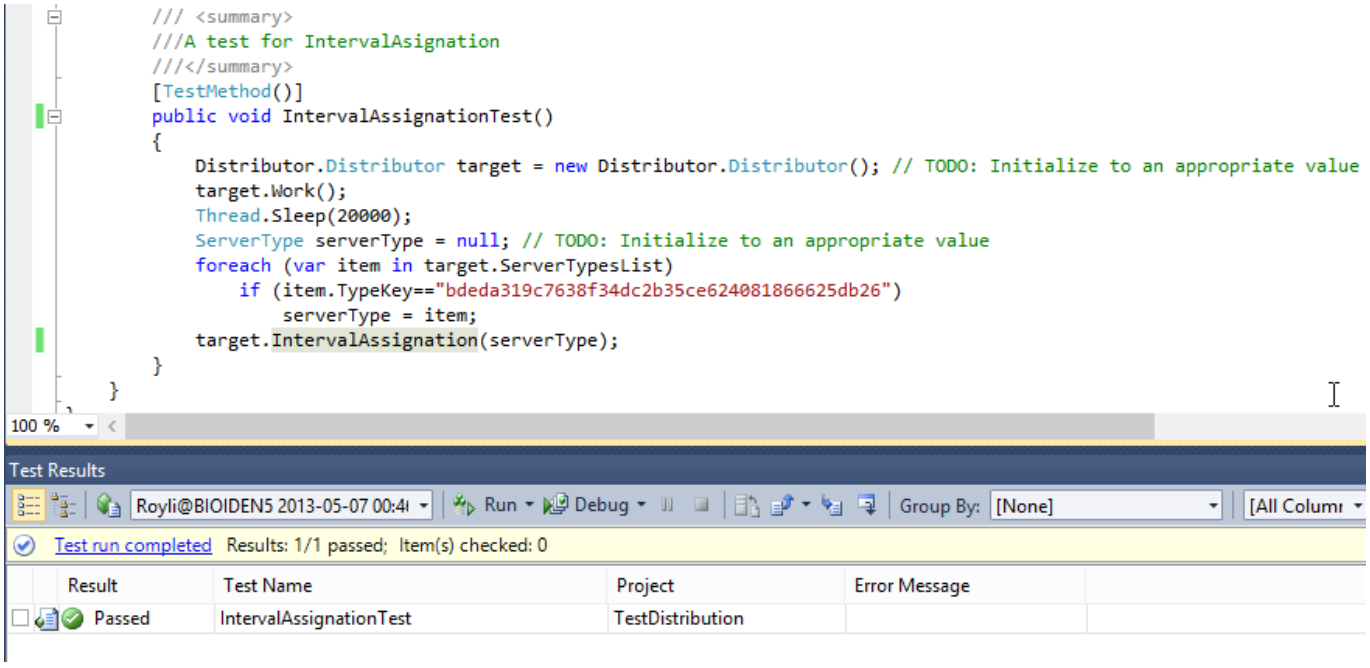
Test Results

Royli@BIOIDEN5 2013-05-06 15:3 | Run | Debug | Group By: [None] | [All Columr

Test run completed Results: 1/1 passed; Item(s) checked: 0

Result	Test Name	Project	Error Message
Passed	PropertiesFromServerTest	TestDistribution	

PU Obtener las propiedades de los servidores de búsqueda.



```
        /// <summary>
        ///A test for IntervalAsignation
        ///</summary>
        [TestMethod()]
        public void IntervalAssignationTest()
        {
            Distributor.Distributor target = new Distributor.Distributor(); // TODO: Initialize to an appropriate value
            target.Work();
            Thread.Sleep(20000);
            ServerType serverType = null; // TODO: Initialize to an appropriate value
            foreach (var item in target.ServerTypesList)
                if (item.TypeKey=="bdeda319c7638f34dc2b35ce624081866625db26")
                    serverType = item;
            target.IntervalAssignation(serverType);
        }
    }
```

100 %

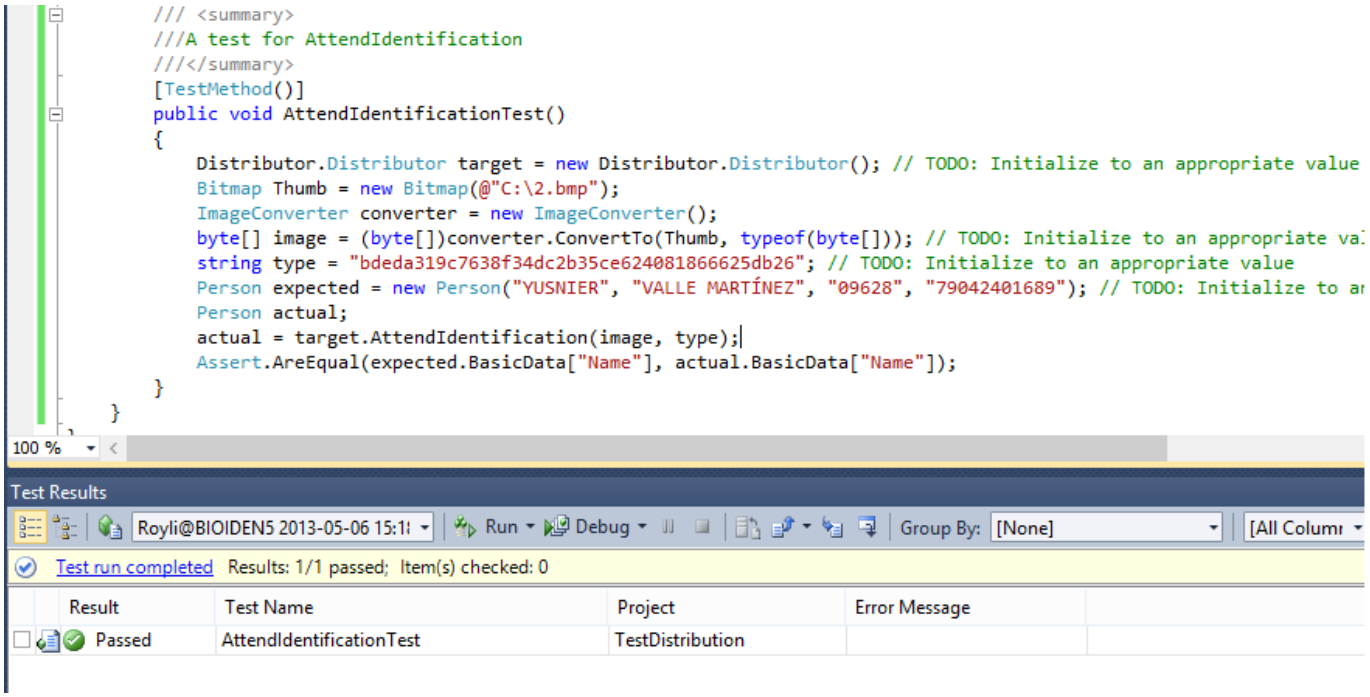
Test Results

Royli@BIOIDEN5 2013-05-07 00:4 | Run | Debug | Group By: [None] | [All Columr

Test run completed Results: 1/1 passed; Item(s) checked: 0

Result	Test Name	Project	Error Message
Passed	IntervalAssignationTest	TestDistribution	

PU Atender solicitud de distribución.



```
        /// <summary>
        ///A test for AttendIdentification
        ///</summary>
        [TestMethod()]
        public void AttendIdentificationTest()
        {
            Distributor.Distributor target = new Distributor.Distributor(); // TODO: Initialize to an appropriate value
            Bitmap Thumb = new Bitmap(@"C:\2.bmp");
            ImageConverter converter = new ImageConverter();
            byte[] image = (byte[])converter.ConvertTo(Thumb, typeof(byte[])); // TODO: Initialize to an appropriate va
            string type = "bdeda319c7638f34dc2b35ce624081866625db26"; // TODO: Initialize to an appropriate value
            Person expected = new Person("YUSNIER", "VALLE MARTÍNEZ", "09628", "79042401689"); // TODO: Initialize to ar
            Person actual;
            actual = target.AttendIdentification(image, type);
            Assert.AreEqual(expected.BasicData["Name"], actual.BasicData["Name"]);
        }
    }
```

100 %

Test Results

Royli@BIOIDEN5 2013-05-06 15:1 | Run | Debug | Group By: [None] | [All Columr

Test run completed Results: 1/1 passed; Item(s) checked: 0

Result	Test Name	Project	Error Message
Passed	AttendIdentificationTest	TestDistribution	

PU Atender solicitud de identificación.

Anexo Casos de prueba alfa

CP Detectar servidores de búsqueda.

Caso de prueba de aceptación	
Código: CDHU1CP1	**Historia de usuario:** Detectar servidores de búsqueda.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que se detecten los servidores activos.	
Condiciones de ejecución: Tanto el distribuidor como los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz principal. 2. Se acciona el botón Iniciar.	
Resultado esperado: Se listan de cada servidor de búsqueda su nombre en la red y tipo.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Notificación de detección.

Caso de prueba de aceptación	
Código: CDHU1CP2	**Historia de usuario:** Detectar servidores de búsqueda.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que los servidores de búsqueda respondan a la detección.	
Condiciones de ejecución: Tanto el distribuidor como los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz principal. 2. Se acciona el botón Iniciar.	
Resultado esperado: Mostrar en la interfaz de cada servidor de búsqueda la notificación de haber enviado su nombre en la red y tipo.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Modificar el estado de trabajo de un servidor de búsqueda.

	Caso de prueba de aceptación
Código: CDHU2CP1	**Historia de usuario:** Modificar el estado de trabajo de un servidor de búsqueda.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que se modifique el estado de un servidor de búsqueda.	
Condiciones de ejecución: Tanto el distribuidor como los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se selecciona en la interfaz principal el servidor de búsqueda. 2. Se modifica el estado del servidor de búsqueda en la interfaz correspondiente a los detalles del servidor.	
Resultado esperado: Mostrar en la interfaz correspondiente el servidor cuyo estado se ha modificado (interfaz de servidores activos, interfaz de servidores disponibles, interfaz con los detalles del servidor modificado).	
Evaluación de la prueba: Prueba satisfactoria.	

CP Obtener las propiedades de los servidores de búsqueda.

	Caso de prueba de aceptación
Código: CDHU3CP1	**Historia de usuario:** Obtener las propiedades de hardware de los servidores de búsqueda.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que se obtengan las propiedades de los servidores de búsqueda, específicamente memoria RAM disponible, velocidad de procesamiento de CPU y valor de aptitud.	
Condiciones de ejecución: Tanto el distribuidor como los distintos servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz principal. 2. Se acciona el botón Iniciar.	
Resultado esperado: Se listan de cada servidor de búsqueda su memoria RAM disponible, velocidad de CPU y valor de aptitud en las interfaces correspondientes (interfaz principal e interfaz con los detalles de un servidor específico).	
Evaluación de la prueba: Prueba satisfactoria.	

CP Notificación de propiedades.

	Caso de prueba de aceptación
Código: CDHU3CP2	**Historia de usuario:** Obtener las propiedades de hardware de los servidores de búsqueda.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que los servidores de búsqueda envíen sus propiedades.	
Condiciones de ejecución: Tanto el distribuidor como los distintos servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz principal. 2. Se acciona el botón Iniciar.	
Resultado esperado: Mostrar en la interfaz de cada servidor de búsqueda la notificación de haber enviado su memoria RAM disponible, velocidad de CPU y valor de aptitud.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Calcular la capacidad de carga de los servidores de búsqueda.

	Caso de prueba de aceptación
Código: CDHU4CP1	**Historia de usuario:** Calcular la capacidad de carga de los servidores de búsqueda.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que se calcule la capacidad de carga de cada servidor de búsqueda teniendo en cuenta su valor de aptitud.	
Condiciones de ejecución: Tanto el distribuidor como los distintos servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se selecciona la opción proceso distribuido en el cliente AFIS. 2. Se ejecuta la acción Distribuir en el cliente AFIS.	
Resultado esperado: Se listan de cada servidor de búsqueda su capacidad de carga en las interfaces correspondientes (interfaz principal, interfaz de distribución e interfaz con los detalles de un servidor específico).	
Evaluación de la prueba: Prueba satisfactoria.	

Anexo Casos de prueba beta

CP Notificación de distribución.

Caso de prueba de aceptación	
Código: CDHU5CP2	**Historia de usuario:** Atender solicitud de distribución.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que cada servidor de búsqueda cargue su porción del banco de datos.	
Condiciones de ejecución: El publicador de solicitudes, el distribuidor y los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se selecciona la opción proceso distribuido en el cliente del AFIS. 2. Se ejecuta la acción Distribuir.	
Resultado esperado: Mostrar en la interfaz de cada servidor de búsqueda la porción del banco de datos cargada.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Notificación de identificación.

Caso de prueba de aceptación	
Código: CDHU6CP2	**Historia de usuario:** Atender solicitud de identificación.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que cada servidor de búsqueda realice el proceso de identificación en su porción del banco de datos.	
Condiciones de ejecución: El publicador de solicitudes, el distribuidor y los servidores de búsqueda deben estar conectados al servidor distribuidor de mensajes RabbitMQ.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz principal del AFIS. 2. Se introduce la huella dactilar. 3. Se ejecuta la acción Identificar.	
Resultado esperado: Mostrar en la interfaz de cada servidor de búsqueda la notificación del resultado de la identificación en la porción del banco de datos asignada.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Adicionar un tipo de servidor de búsqueda.

Caso de prueba de aceptación	
Código: CDHU7CP1	**Historia de usuario:** Gestionar tipos de servidores de búsqueda.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que se adicione un tipo de servidor de búsqueda.	
Condiciones de ejecución: El tipo de servidor de búsqueda a adicionar no debe existir.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz de inserción de tipos de servidores de búsqueda. 2. Se introducen los datos requeridos.	
Resultado esperado: Mostrar el nuevo tipo insertado en la interfaz de tipos de servidores registrados con su llave de identificación. Actualizar la detección de los servidores de búsqueda de acuerdo a los cambios ocurridos en el sistema.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Eliminar un tipo de servidor de búsqueda.

Caso de prueba de aceptación	
Código: CDHU7CP2	**Historia de usuario:** Gestionar tipos de servidores de búsqueda.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que se elimine un tipo de servidor de búsqueda.	
Condiciones de ejecución: Debe detenerse el proceso de distribución.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz de principal de tipos de servidores de búsqueda. 2. Se selecciona el tipo de servidor a eliminar.	
Resultado esperado: Mostrar en la interfaz de tipos de servidores de búsqueda solo aquellos que estén registrados.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Adicionar la conexión a un banco de datos.

Caso de prueba de aceptación	
Código: CDHU8CP1	**Historia de usuario:** Gestionar conexiones a bancos de datos.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que se adicione la conexión a un banco de datos.	
Condiciones de ejecución: La conexión del banco de datos a adicionar no debe existir.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz de bancos de datos. 2. Se introducen los datos requeridos.	
Resultado esperado: Mostrar en la interfaz de los bancos de datos la nueva conexión registrada.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Modificar la conexión a un banco de datos.

Caso de prueba de aceptación	
Código: CDHU8CP2	**Historia de usuario:** Gestionar conexiones a bancos de datos.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que se modifique la conexión a un banco de datos.	
Condiciones de ejecución: Debe detenerse el proceso de distribución.	
Entrada/Pasos de ejecución: 1. Se accede a la interfaz de bancos de datos. 2. Se selecciona la conexión a modificar. 2. Se modifican los datos.	
Resultado esperado: Mostrar en la interfaz de los bancos de datos los cambios ocurridos en la conexión modificada.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Eliminar la conexión a un banco de datos.

Caso de prueba de aceptación	
Código: CDHU8CP3	**Historia de usuario:** Gestionar conexiones a bancos de datos.

Responsable de la prueba: Melvis Machin Armas
Descripción: Prueba para verificar que se elimine la conexión a un banco de datos.
Condiciones de ejecución: Debe detenerse el proceso de distribución.
Entrada/Pasos de ejecución: 1. Se accede a la interfaz de bancos de datos. 2. Se selecciona la conexión a eliminar.
Resultado esperado: Mostrar en la interfaz de los bancos de datos solo aquellas conexiones que estén registradas.
Evaluación de la prueba: Prueba satisfactoria.

CP Generar logs del sistema.

	Caso de prueba de aceptación
Código: CDHU9CP1	**Historia de usuario:** Generar logs del sistema.
Responsable de la prueba: Royli Hernández Delgado	
Descripción: Prueba para verificar que se generen los logs del sistema.	
Condiciones de ejecución: El publicador de solicitudes, el distribuidor y los servidores de búsqueda deben estar conectados al sistema distribuidor de mensajes.	
Entrada/Pasos de ejecución: 1. Se debe solicitar la distribución del banco de datos o la identificación de un individuo.	
Resultado esperado: Mostrar en un log detalles del proceso de atención de la solicitud.	
Evaluación de la prueba: Prueba satisfactoria.	

CP Configurar opciones de distribución.

	Caso de prueba de aceptación
Código: CDHU10CP3	**Historia de usuario:** Configurar opciones de distribución.
Responsable de la prueba: Melvis Machin Armas	
Descripción: Prueba para verificar que se configuren las opciones de distribución.	
Condiciones de ejecución: Debe detenerse el proceso de distribución.	
Entrada/Pasos de ejecución: 1. Acceder a la interfaz de configuración. 2. Modificar los parámetros deseados.	
Resultado esperado: Mostrar la nueva configuración.	

Evaluación de la prueba: Prueba satisfactoria.

Anexo Datos de pruebas de rendimiento

Entorno de pruebas.

Elemento	Software	Hardware
AFIS centralizado	Plataforma .Net versión 4.0	• CPU 2.13 GHz • 1 GB de memoria RAM
Distribuidor	Plataforma .Net versión 4.0	• CPU 3.0 GHz • 1 GB de memoria RAM
Servidor de búsqueda	Plataforma .Net versión 4.0	• CPU 2.13 GHz • 1 GB de memoria RAM

Comparación de tiempos de respuesta (ms).

Banco de datos	Centralización	Distribución (3 servidores)	Distribución (6 servidores)
1000	1732,6031	765,6601	582,0943
2000	1784,7629	796,9156	580,5692
3000	1790,5505	859,4153	588,1736
4000	1887,6033	1109,4292	602,006
5000	1981,2935	953,1745	618,421
6000	2511,6644	1015,6739	892,309
7000	2542,8044	1468,8214	700,548
8000	2761,2048	1484,448	740,623
9000	2808,005	1312,5645	778,4092
10 000	3142,6054	1140,6806	812,0234
11 000	3556,8062	1140,6806	948,6982
12 000	3853,2068	1500,073	1003,5463
13 000	3966,6436	1531,3245	1200,2819
14 000	4034,1873	1515,6979	1268,5095

Tiempos locales de búsqueda (ms).

Banco de datos	Tiempo de respuesta del AFIS (3 servidores)	Tiempo de búsqueda local
1000	765,6601	530,401

2000	796,9156	532,583
3000	859,4153	602,543
4000	1109,4292	608,401
5000	953,1745	670,8012
6000	1015,6739	686,4012
7000	1468,8214	1170,002
8000	1484,448	1154,2102
9000	1312,5645	1163,754
10 000	1140,6806	1082,532
11 000	1140,6806	1101,2021
12 000	1500,073	1234,4356
13 000	3966,6436	1265,6864
14 000	1515,6979	1234,4345