Universidad de las Ciencias Informáticas

Facultad 1

Trabajo de diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: Componente para la comparación de huellas dactilares.

Autor: Asiel Echemendia Carrasco.

Tutor: Ing. Ramón Santana Fernández.

La Habana, 27 de junio de 2013

"Año 55 de la Revolución"

"La revolución es algo que se lleva en el alma, no en la boca para vivir de ella."

Declaro ser el autor del trabajo titulado Componente para la comparación de huellas dactilares de la Universidad de las Ciencias Informáticas y autorizo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo el presente a los ____ días del mes de ____ del año 2013.

_____	_____
Asiel Echemendia Carrasco.	Ing. Ramón Santana Fernández

Opinión del Tutor

Agradecimientos

Agradecerle a esta Revolución por haberme dado la oportunidad de formarme como persona y de poder estudiar en una Universidad como esta.

A mi mamá y a mi papá por darme la vida, por confiar en mí, por apoyarme siempre en mis decisiones y por estar siempre a mi lado cuando más lo he necesitado.

A mi queridísimo abuelo que más que abuelo ha sido un padre para mí, que luchó por mí sin importarle nada ni nadie. Te quiero Abu.

A mi novia Liuba Ma. Barroso (Tita) por ser la mujer de mi vida, por acompañarme en todo momento.

A mis hermanos Arianna, Marisel, Jorge Mario y Massiel porque los quiero con toda mi alma, porque saben que si tengo que dar la vida por ellos la doy y no pasa nada.

A mi abuela, a mi tío por parte de madre que hoy no está aquí porque está lejos pero sé que está orgulloso de mí, gracias por todo tío, te quiero mucho.

A mi abuela que espero que en el cielo estés orgullosa de mí. Te quise, te quiero y te querré.

A mi abuelo paterno que lo quiero mucho.

A mis tíos y tías por parte de padre que los quiero y le agradezco por todo el apoyo que de una forma u otra he recibido.

A mi madrastra que mucho ha luchado por mí y sabe que para mí es mi segunda madre.

A Massiel por haberme dado a mi niño lindo (mi sobrino del alma).

A toda mi familia que de verdad sin ustedes no hubiera sido nadie. Esto es por ustedes.

A todos mis amigos por ayudarme y aconsejarme cuando lo he necesitado.

A todos mis amigos (Redox, El Funny, Jesús, José (El niño), El flaco, Yordanys, Omarito, Leyva) de la Universidad y que seguro pueden estar que estarán por siempre en mi corazón y que esos recuerdos de escándalos, de fútbol, de fiesta y todos los momentos que pasamos juntos nunca los olvidaré.

A mi tutor Ramón por ayudarme al desarrollo de este trabajo y además sabes que puedes contar conmigo para lo que sea.

A todos los que de una forma u otra han compartido conmigo durante estos 5 años de carrera.

A los profesores que me han ayudado y me han formado como un verdadero profesional.

Dedicatoria

A mi familia que es mi motor impulsor, mi ejemplo y por haberme ayudado a ser quien soy hoy, por estar siempre a mi lado cuando más lo he necesitado. Los quiero a todos.

A mí querida Tita por todo lo que significa para mí y lo mucho que me ayudó en el desarrollo de este trabajo.

A todos mis amigos que siempre me han ayudado, criticado y aconsejado en los momentos difíciles de la vida.

Resumen

En la actualidad el empleo de técnicas biométricas en sistemas informáticos, ha permitido identificar y autenticar de forma segura a las personas. Entre los métodos de identificación biométrica más utilizados se encuentra el reconocimiento de personas a través de la huella dactilar debido a su facilidad de uso y la gran aceptación con la que cuenta por parte de los usuarios.

En Cuba diversas instituciones han impulsado el desarrollo tecnológico con la producción de sistemas de identificación de personas a través de huellas dactilares; ámbito en el cual la Universidad de las Ciencias Informáticas se encuentra actualmente, a través del desarrollo de un sistema de verificación de individuos mediante huellas dactilares. El objetivo general de la presente investigación consiste en desarrollar un componente para la comparación de minucias en huellas dactilares a partir de la biblioteca de clases SourceAfis desarrollada en ".Net".

El desarrollo del componente brindará la posibilidad de realizar el proceso de comparación a partir de las plantillas biométricas en formato ISO/IEC 19794-2:2011, utilizando tecnologías y herramientas informáticas que responden a las políticas de software libre. La solución será embebida en un applet para integrarlo al sistema de verificación de individuos mediante huellas dactilares y podrá ser utilizado de manera independiente en cualquier otra solución que requiera del servicio de comparación.

Palabras clave: sistemas biométricos, técnicas biométricas, plantillas biométricas.

Índice

Índice de Figuras

Índice de Tablas

Introducción

Con el desarrollo de las tecnologías informáticas, se ha agilizado la automatización de los procesos de identificación de individuos, y por tanto la Biometría[1] se ha transformado en un área importante. El empleo de las técnicas biométricas en los sistemas informáticos ha permitido identificar y autenticar de forma segura a las personas, convirtiéndose en técnicas cada vez más confiables que el uso de la clave personal como método corriente de identificación.

Entre los métodos de identificación biométrica más utilizados está el reconocimiento de personas a través de la huella dactilar, debido a que este cumple con los parámetros de universalidad (todas las personas deben poseerlo), unicidad (ningún otro individuo presenta las mismas características) y permanencia (esta característica es invariable en el tiempo). (1).

La técnica de reconocimiento de personas a través de las huellas dactilares es un método que tuvo su origen a mediados del siglo XIX, es el método de identificación biométrica por excelencia, ya que es fácil de usar y cuenta con una gran aceptación por parte de los usuarios. Este método de identificación es uno de los procedimientos biométricos más utilizados en el mundo. En sus inicios, las identificaciones las realizaba una persona especializada, de manera visual, ubicando las minucias en ambas huellas y luego comparándolas para determinar si pertenecían a la misma persona. En la actualidad se han desarrollado sistemas que realizan esta labor de manera automática, logrando comparar varios cientos de huellas en minutos. Estos avances en la identificación de las personas a través de las huellas dactilares han favorecido el desarrollo de un gran campo en el área de la seguridad y la identificación personal.

Las técnicas biométricas son utilizadas para la identificación de personas en muchos países desarrollados, siendo aplicables en diferentes medios o lugares donde se requieran altos niveles de seguridad, por ejemplo en bancos, pasaportes electrónicos y cajeros automáticos. En cada caso, se mide cierta combinación de las características inherentes y se comparan automáticamente con plantillas almacenadas en un archivo o en una base de datos para realizar la identificación.

[1] **Biometría:** es el estudio de métodos automáticos para el reconocimiento único de humanos basados en uno o más rasgos conductuales o rasgos físicos intrínsecos.

En Cuba, la Biometría es un campo joven, durante la investigación realizada se encontraron las empresas cubanas de Desarrollo de Aplicaciones, Tecnologías y Sistemas (Datys), el Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV) y la Universidad de la Ciencias Informáticas (UCI), las cuales han impulsado el desarrollo tecnológico con la producción de sistemas basados en el proceso de identificación Dactiloscópica[2]. DATYS cuenta con un software biométrico (Biomesys AFIS), el cual es un sistema automatizado de identificación Dactiloscópica, que permite la identificación y autentificación de forma rápida y segura. CENATAV contribuyó al desarrollo del Biomesys AFIS durante su primera fase de elaboración, además de llevar a cabo diferentes investigaciones en el campo de la biometría. (2).

La UCI constituye una universidad productiva en la cual se han desarrollado diferentes componentes que permiten determinar la calidad de las huellas dactilares, reconstruirlas a partir de plantillas de minucias y compararlas en tarjetas inteligentes. También existe un sintetizador de huellas dactilares.

La actividad productiva en la UCI está sustentada en la red de centros, entre los cuales se encuentra el Centro de Identificación y Seguridad Digital (CISED), en el cual se lleva a cabo el proceso de desarrollo de un sistema de verificación de individuos mediante huellas dactilares, tomando como punto de partida la biblioteca de clases de SourceAfis para la extracción y comparación de características. El SourceAfis está desarrollado en ".Net" y posee una versión alfa no funcional en java que cuenta con una versión del módulo de comparación muy primitivo y no funcional. A partir de la implementación existente en ".Net" se puede desarrollar un ActiveX para la comparación de huellas dactilares, pero el uso de tecnologías propietarias continuaría presente y estaría sujeto de forma obligatoria al uso del Internet Explorer como navegador, trayendo consigo que esta solución solo pueda ser usada en el sistema operativo Windows. Conformándose así la **situación problemática** de la investigación.

Antes esta inconveniente se plantea el siguiente **problema de investigación:** ¿Cómo llevar a cabo el proceso de comparación de huellas dactilares en un entorno multiplataforma?

Para solucionar el problema planteado se determinó como **objeto de estudio** los procesos de comparación de huellas dactilares.

[2] **Dactiloscopia**: es la ciencia que estudia la identificación de las personas por medio de las impresiones formadas por las crestas papilares de las yemas de los dedos de las manos.

Para dar solución al problema antes planteado se define como **objetivo general** desarrollar un componente para la comparación de minucias de huellas dactilares en formato ISO/IEC 19794-2:2011, para integrarlo como un applet de java al sistema de servicios de verificación dactilar que se desarrolla en el departamento de Biometría del Centro de Identificación y Seguridad Digital; el cual fue desglosado en los siguientes **objetivos específicos:**

- Determinar las tendencias mundiales de los algoritmos de comparación de minucias de huellas dactilares.
- Definir las tecnologías, metodologías y herramientas a utilizar para el desarrollo de dicho componente.
- Realizar análisis y diseño del sistema.
- Implementar un algoritmo para la comparación de minucias de huellas dactilares a partir de plantillas de minucias.
- Realizar pruebas al componente para garantizar la validación.

Las **preguntas científicas** por las que se rige esta investigación son:

- ¿Cómo llevar a cabo el proceso de comparación de huellas dactilares?
- ¿Cómo obtener una plantilla de minucias en el formato ISO/IEC 19794-2:2011?
- ¿Qué patrones de diseño son factibles utilizar para el diseño de la solución?

Para llevar a cabo el objetivo del trabajo se plantearon las siguientes **tareas investigativas:**

- Análisis del estado del arte en Cuba y en el mundo referente a los algoritmos de comparación de minucias en huellas dactilares.
- Análisis de los algoritmos para la comparación de minucias en huellas dactilares.

- Definición de las tecnologías, metodologías y herramientas a utilizar para el desarrollo del componente.
- Definición de la arquitectura del software.
- Confección de las historias de usuario.
- Levantamiento de los requisitos funcionales y no funcionales.
- Implementación del algoritmo para la lectura de la plantilla de minucias.
- Definición de los patrones del Diseño.
- Implementación del algoritmo para la comparación de minucias de huellas dactilares.
- Realización de las pruebas al componente como: FAR, FRR, ERR.

La **justificación de la investigación:**

Con el desarrollo de un componente para la comparación de minucias en huellas dactilares, el departamento de biometría contará con la posibilidad de realizar este proceso en el sistema de verificación dactilar, directamente en el navegador una vez que sea incluido el componente en el applet para la verificación dactilar, utilizando tecnologías libres. Además, su ejecución al embeberlo en un applet de java no dependería del tipo de navegador utilizado por el cliente, ni estaría sujeto a un sistema operativo determinado, contribuyendo así a la soberanía tecnológica del país.

Los **Métodos Científicos** empleados son:

- **Métodos Teóricos.**
  - ✓ **Analítico - Sintético:** Se utilizó en el análisis de las diferentes teorías de huellas dactilares para determinar sus generalidades y establecer relaciones entre estas. También para extraer los elementos más importantes relacionados con el tema de comparación de huellas dactilares.
  - ✓ **Análisis Histórico – Lógico:** Se utilizó en el estudio de los antecedentes, la evolución y el desarrollo que han tenido los sistemas biométricos de huellas dactilares, así como valorar las

tendencias actuales de los algoritmos de comparación de huellas dactilares a través de la búsqueda de información en diferentes bibliografías.

- **Métodos empíricos.**
  - ✓ **Entrevista:** se realiza con el propósito de obtener información, ideas, puntos de vista que contribuyan al desarrollo de la investigación y aporten conocimientos específicos del tema.

El presente trabajo de diploma constará de tres capítulos estructurado de la siguiente forma:

Capítulo 1: **Fundamentación Teórica**, en el presente capítulo se describen los principales conceptos asociados al tema, se realiza un estudio del estado del arte de la investigación tanto a nivel internacional como nacional y se realiza además un estudio de las principales tecnologías, metodologías y herramientas para dar solución al problema planteado.

Capítulo 2: **Características del Sistema**, en el presente capítulo se definen las características del sistema realizando un análisis del dominio del componente, se confeccionan las tarjetas CRC (Class-Responsibility-Collaboration) para dejar todo listo para la etapa de implementación. Se hace referencia a los patrones utilizados para la construcción del componente.

Capítulo 3: Implementación y Prueba, en el presente capítulo se analizan aspectos relacionados con la implementación del componente para la comparación de minucias en huellas dactilares y se realizan las pruebas del sistema, donde se valida el funcionamiento de los requisitos funcionales.

Capítulo 1: Fundamentación Teórica.

En el presente capítulo se realiza un estudio del estado del arte de la investigación en el mundo y en nuestro país, se analizan además las principales metodologías, tecnologías y herramientas existentes con el objetivo de seleccionar la adecuada para el desarrollo de la investigación.

1.1 Sistemas Biométricos.

Un sistema biométrico es aquel que fundamenta sus decisiones de reconocimiento mediante una característica personal que puede ser reconocida o verificada de manera automatizada. En esta sección son descritas algunas de las características más importantes de estos sistemas (3).

Actualmente existen en el mundo diversos sistemas biométricos que basan su reconocimiento en algunas características de la persona tanto fisiológicas como conductuales (ver Figura 1.). Ejemplo de estos son los basados en: la geometría de la mano, la estructura venosa, los patrones de la retina, el termograma facial, la huella dactilar, el iris, la voz, la firma y el rostro.

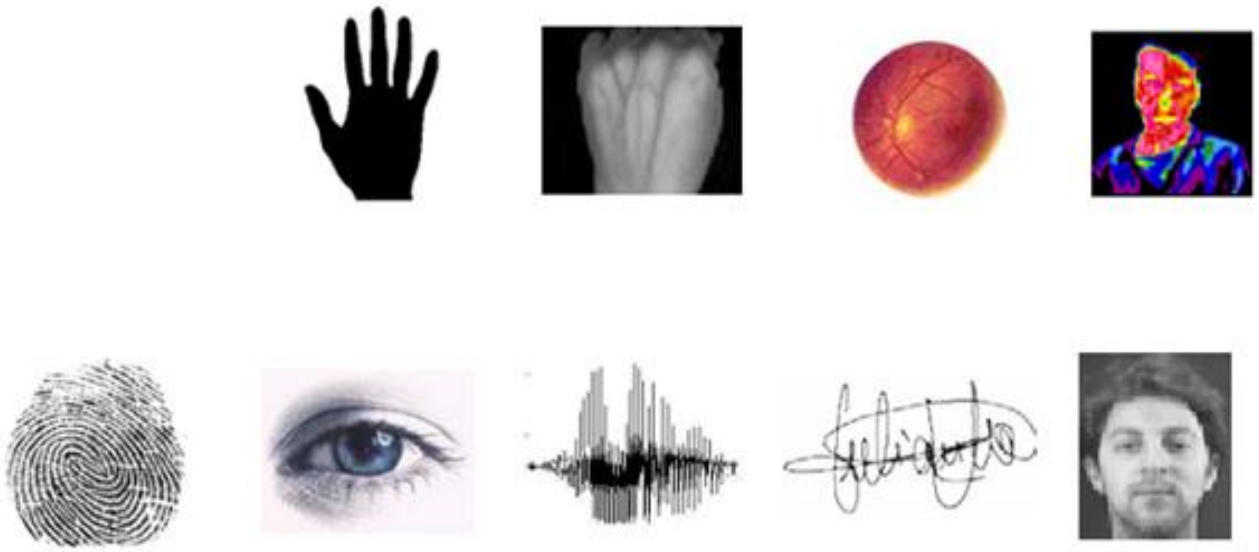

Figura 1: Técnicas biométricas.

1.1.1 Características de un sistema biométrico para identificación personal.

Las características básicas de un sistema biométrico para la identificación de personas deben estar basadas en el cumplimiento de restricciones que permitan obtener una solución con utilidad práctica que cumpla con los requisitos de desempeño, aceptabilidad y fiabilidad (4).

- **El desempeño:** se refiere a la exactitud, rapidez y robustez alcanzada en la identificación.
- **La aceptabilidad:** indica el grado de aprobación que tiene la tecnología entre el público.
- **La fiabilidad:** refleja cuán difícil es burlar al sistema. El sistema biométrico debe reconocer características de una persona, pues es posible crear dedos de látex, grabaciones digitales de voz, prótesis de ojos, etc. Algunos sistemas incorporan métodos para determinar si la característica bajo estudio corresponde o no a la de una persona viva. Los métodos empleados son ingeniosos y usualmente más simples de lo que uno podría imaginar.

1.2 Huellas Dactilares.

Las huellas dactilares aparecen en el período fetal a partir del sexto mes y estas no varían durante el tiempo de vida del individuo a menos que sufran alteraciones debido a accidentes tales como cortes o quemaduras; por lo que constituyen una característica propia de cada persona a partir de la cual puede ser identificada ya que es casi imposible encontrar dos individuos con las mismas huellas dactilares.

Una huella dactilar está constituida por las crestas papilares de los dedos de las manos que aparecen visibles en la epidermis, generando configuraciones diversas. Las discontinuidades locales en el patrón del flujo de las crestas son conocidas como minucias (5).

1.2.1 Clasificación de huellas dactilares.

Las huellas dactilares se clasifican en 5 tipos (6): Ver **Anexo 1. Clasificación de las huellas dactilares.**

- **Espiral:** estos patrones tienen muchos círculos que salen de un punto de la huella, las mismas cuentan con dos puntos deltas.

- **Lazo derecho:** se caracterizan porque las crestas que forman su núcleo nacen en el costado izquierdo de la huella y hacen su recorrido hacia la derecha, para luego dar vuelta sobre sí mismas y regresar al mismo punto de partida. Estas tienen un punto delta que se encuentra ubicado al lado derecho de la huella.

- **Lazo izquierdo:** Tienen un punto delta que se ubica del lado izquierdo de la huella. Las crestas papilares que forman el núcleo nacen a la derecha y su recorrido es a la izquierda para dar vuelta sobre sí mismas y regresar al mismo punto de partida.

- **Arco:** Estos patrones tienen líneas que empiezan en un lado de la huella, van hacia el centro curvándose hacia arriba y salen del otro lado de la huella, formando lo que se asemeja a una fila de arcos apilados, no cuentan con un punto delta.

- **Arco tendido:** Es un tipo de línea que rápidamente nace y se pierde en un paso de ángulo y semeja un arco, estos tienden a ser asociados con los patrones de huellas tipo arco tendido.

1.2.2 Minucias.

Las minucias son puntos característicos que describen e identifican de modo inequívoco una huella dactilar. Estas son las características que más se utilizan para el análisis y comparación de las huellas dactilares, y se forman a partir de las discontinuidades que presentan las crestas dactilares.

- **Tipos de minucias.**

  Según la forma de las **crestas dactilares** se pueden identificar diversos tipos de minucias ya que pueden aparecer dos crestas que convergen en una sola, crestas que no tienen continuidad, crestas que se interrumpen, etc. En la Figura 2 se muestran los diferentes tipos de minucias.

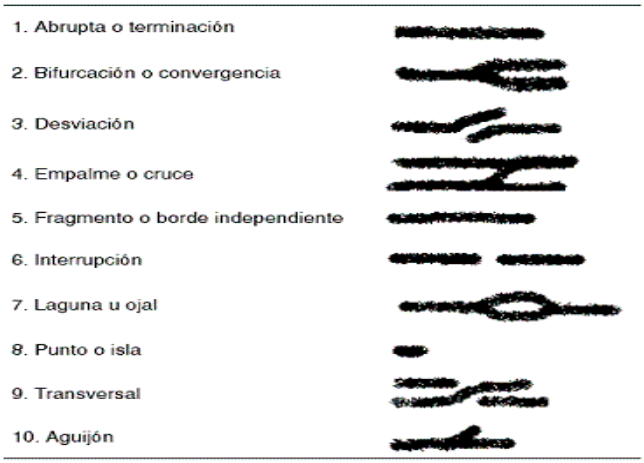

Figura 2: Tipos de minucias.

Durante el proceso de comparación de huellas dactilares los tipos de minucias que más se utilizan son las bifurcaciones y las terminaciones (ver Figura 3).

Figura 3: Clasificación de minucias.

1.3 Comparación de huellas dactilares.

Las minucias se utilizaban en el reconocimiento manual que realizaban los expertos antes de existir los AFIS y es por ello que al diseñar los primeros sistemas automáticos gran parte de los algoritmos se basaron en ellas. Esta tendencia ha seguido existiendo con el paso del tiempo y las técnicas basadas en minucias son las más estudiadas y sobre las que existe una mayor cantidad de algoritmos de identificación (7).

Durante la comparación para decidir si dos huellas dactilares corresponden o no a una misma persona se lleva a cabo un proceso que se inicia con la obtención de las plantillas biométricas o plantillas de minucias y termina con el resultado positivo o negativo resultante de la comparación de las minucias de ambas huellas.

Para realizar el cotejo de huellas se lleva a cabo una comparación de estas a partir de los vectores de características que se obtienen como resultado de la representación geométrica de cada una de las minucias con el objetivo de encontrar el grado de similitud entre ellos. Esta es una de las tareas más críticas de todo el proceso de identificación ya que durante su desarrollo hay que tener en cuenta las siguientes problemáticas (8):

- En las huellas no se dispone de ningún punto de referencia válido a partir del cual se pueda llegar a tomar medidas, y que dicho punto sea válido para todas las huellas. En otras técnicas se tiene un punto claro (el centro de la pupila en el caso de la técnica de reconocimiento por iris). Esta falta de referencia, hace que muchos métodos de comparación propuestos, no sean factibles. Algunos autores intentan utilizar como punto de referencia el núcleo[3], pero su localización exacta no es sencilla, y además, en algunas huellas no se obtiene ese punto singular.

- La elasticidad del dedo hace que, dependiendo de la postura y de la presión, se tenga una variabilidad en las medidas de un mismo usuario, que haga bastante difícil la comparación de las huellas obtenidas. De hecho, el problema puede llegar a ser más grave, ya que, dependiendo del grado de rotación que se haya dado al dedo, pueden llegar a capturarse un elevado número de minucias en la muestra, que no se encuentran consideradas en el patrón.

1.3.1 Dificultades en el proceso de comparación de huellas dactilares.

Las principales dificultades ante las que debe actuar un buen algoritmo de comparación son (8):

- **Desplazamiento y rotación:** son el resultado de que el usuario sitúe el dedo en un lugar diferente del sensor en cada ocasión. Pueden ocasionar que una parte de la huella se encuentre fuera del área de captura, por lo que las huellas a comparar tendrán un área menor

[3] **Núcleo:** Punto más elevado de la cresta más interna de la huella dactilar.

con relación a la huella almacenada. Este problema afecta en gran medida a los sensores con una reducida área de captura.

- **Transformación no lineal:** es la consecuencia de plasmar en una imagen de dos dimensiones una huella dactilar de 3 dimensiones, con una elasticidad que provoca deformaciones no lineales en su superficie.

- **Diferencias de presión y de las condiciones de la piel:** la presión que se ejerza contra el sensor y la humedad o sequedad de la piel, hacen que la captura sea diferente en cada situación. También afectan sustancias corporales como el sudor o la grasa.

- **Errores en la extracción de características:** los algoritmos de extracción de minucias tienden a producir minucias espurias en las huellas de baja calidad, que enmascaran a menudo las minucias reales.

1.3.2 Algoritmos de comparación de huellas dactilares.

- **Basados en correlación, normalmente a nivel de bloque:** las imágenes del par de huellas a comparar se superponen y se calcula la correlación entre píxeles equivalentes para diferentes alineamientos (variaciones en la posición y el ángulo).

- **Basados en minucias:** se trata de la técnica más extendida ya que sustenta la comparación manual de huellas por parte de los expertos forenses en la materia. Las minucias se extraen de ambas huellas y se almacenan en vectores como conjuntos de puntos en el plano bidimensional de la imagen. Estos algoritmos tratan de conseguir el mayor número posible de coincidencias entre pares de minucias de la huella modelo y la huella a comparar, incluso sin tener que alinear las huellas.

- **Basados en características del patrón de crestas o en texturas:** en imágenes de baja calidad la extracción de las minucias es bastante complicada y poco fiable mientras que otras características (orientación local, forma de las crestas, información de texturas), en general menos distintivas, pueden obtenerse de manera más robusta. Los algoritmos pertenecientes a

esta familia comparan las huellas en términos de las características antes mencionadas extraídas del patrón de crestas.

- **Los métodos basados en imágenes y en textura:** las características es el alineamiento, este consiste en tener ambas huellas a comparar en una misma orientación. Lograr esto puede llegar a ser un proceso bastante costoso a pesar de la existencia de diversas técnicas que hacen uso de puntos característicos como son los núcleos y los deltas[4], pero estos tienden a ser de difícil extracción y vulnerables a los errores. Una tendencia es el uso de coordenadas relativas entre las minucias, guardándose en estructuras especializadas. Esto hace innecesario el alineamiento por lo que los algoritmos que trabajan sobre esta tendencia poseen gran ventaja.

Durante la aplicación de los algoritmos descritos anteriormente se deben tener en cuenta las problemáticas que pueden aparecer durante el proceso de comparación (ver sección 1.3) por lo que en la actualidad los sistemas biométricos que usan como identificador las huellas dactilares y realizan el proceso de comparación basado en minucias tienden a utilizar la técnica de comparación elástica de A. K. Jain.

El uso de esta técnica comienza con la búsqueda de una minucia de referencia, analizando tanto el vector resultante de la muestra, como el de la huella almacenada, de forma que se pueda obtener una minucia cuya semejanza entre los dos vectores sea tan alta, que pueda considerar que pertenezca al mismo individuo. Posteriormente se hace un cambio de coordenadas de los dos vectores, para pasarlo a polares (radio y ángulo), con el centro en esa minucia de referencia. Una vez realizado esto, se comparan, uno a uno, todos los pares de minucias susceptibles de estar en el mismo sitio. Para considerar que están en el mismo sitio, se crea un área de influencia de cada minucia del patrón, que permita la tolerancia dada por la elasticidad del dedo. Si la minucia de la muestra está en la zona de influencia de la minucia del patrón, y su tipo y orientación son compatibles, entonces se podrá determinar que la minucia es la misma.

Realizando esto para todos los pares de minucias, y aplicando unas fórmulas de medida de proximidad, con factores de penalización para los casos en los que la comparativa haya sido infructuosa, se

[4] **Deltas**: Punto en una cresta más cercano donde dos crestas divergen.

consigue un resultado numérico que dará la probabilidad de que las huellas pertenezcan a la misma persona.

Como en todo sistema biométrico, el diseñador del sistema propone un umbral que se compara con el valor de similitud calculado en el proceso de comparación y el resultado de esta comparación hará que las huellas se consideren idéntica (aceptando al usuario), o falsa (rechazándolo). Se estudiarán entonces las posibles tasas de error: Falsa Aceptación (FAR) o Falso Rechazo (FRR), para indicar la calidad del sistema de identificación en la aplicación dada (9).

1.3.3 El uso de huellas dactilares en sistemas biométricos.

Actualmente la utilización de la huella dactilar como identificador biométrico se debe a su utilidad como recurso para autenticar de forma segura a las personas que pretenden acceder a un determinado servicio o recinto físico, también en gran parte a su temprano estudio científico como medio de identificación unívoco de las personas.

A finales del siglo XIX aparecieron dos estudios científicos relevantes en este ámbito: Galton, con el análisis de las huellas dactilares y la aparición de las minucias para la comparación, y Henry, con la clasificación de las huellas en función de la forma macroscópica formada por sus crestas. En los inicios del siglo XX las huellas dactilares comenzaron a utilizarse en la ciencia forense, proporcionando a la policía una forma para identificar a los criminales; y a mediados del siglo XX la amplia utilización de estas en el ámbito forense y policial y el número creciente de expertos inmersos en el análisis de los procesos de evaluación y comparación potenció la aparición de “Sistemas de reconocimiento automático de huellas dactilares” o “Automatic Fingerprint Identification System” (AFIS por sus siglas en inglés) (10).

1.4 Sistemas biométricos existentes para la comparación de huellas dactilares.

Actualmente con el desarrollo de las tecnologías se cuenta con herramientas y sistemas de alta potencia que permiten realizar los procesos de identificación y verificación de manera automatizada, y los sistemas de identificación basados en huellas dactilares (AFIS) se han extendido a diversas esferas de la sociedad.

A continuación se describen algunos componentes que realizan el proceso de comparación de huellas dactilares.

1.4.1 Nivel Internacional.

Actualmente en el mundo existen diversas soluciones que realizan el proceso de comparación de huellas dactilares como el SourceAfis y el VeriFinger SDK

1.4.1.1 SourceAfis

El SourceAfis es una biblioteca de clases para el reconocimiento de huellas dactilares, creado por Robert Vazan, cuenta con una interfaz de programación de aplicaciones fácil de usar soportada sobre ".Net" y cuenta además con una versión alfa no funcional para Java que no cuenta con el módulo de comparación de minucias. El SourceAfis se caracteriza por no poseer dependencia de los dispositivos de captura de las huellas dactilares, por tanto, acepta imágenes de huellas de todos los lectores comunes, e igualmente permite realizar búsquedas exhaustivas en la base de datos de las huellas dactilares a una velocidad de diez mil huellas por segundo. Otra de sus particularidades es que acepta y exporta plantillas biométricas basadas en el estándar ISO / IEC 19794-2. (11).

1.4.1.2 VeriFinger SDK

VeriFinger es una tecnología de identificación de huellas dactilares desarrollada desde 1998 por la compañía Neurotechnology. Desde que fue creada se han publicado más de 10 versiones y constituye una de las prestaciones de algoritmos de reconocimiento de huellas dactilares más potentes hasta la fecha. La última versión de VeriFinger es compatible con NIST MINEX, ya que se basa en el motor de identificación de huella dactilar del MegaMatcher. Aquel esquema que utiliza un conjunto de puntos característicos específicos (minucias) de las huellas dactilares, junto con un número de soluciones algorítmicas propietarias que mejoran el rendimiento y la fiabilidad del sistema. Permite lograr modos de cotejado de 1 a 1 y de 1 a muchos y una velocidad de comparación de cuarenta mil huellas por segundo. Está disponible como un conjunto de herramientas para el desarrollo del software (*SDK por sus siglas en inglés Software Development Kit)* para MS Windows, Windows CE 3.0 y Linux. (12).

Durante el estudio de los sistemas biométricos existentes para la comparación de huellas dactilares se encontraron dos sistemas más: el **Efinger** (sistema basado en la identificación y verificación de huellas dactilares el cual esta implementado en los lenguajes C++, VC++ y Matlab para ser utilizado en la plataforma Windows de 32 bits) y **FingerPrint-POC** (utiliza un comparador de comparación de imágenes débil, sin documentos y no hay estadísticas de precisión. Este sistema está desarrollado en el lenguaje C y para el sistema operativo Linux) pero no se encontraron datos suficientes para que fueran estudiados.

1.4.2 Nivel Nacional.

Cuba ha desarrollado productos de software que aplican la biometría, los cuales forman parte del desarrollo tecnológico del país. Ejemplos de estos productos son:

1.4.2.1 BIOMESYS Control de Asistencia.

Es un sistema que se beneficia de las bondades de las tecnologías que aplican la biometría, para registrar los eventos de asistencia en una empresa por medio de la identificación de los empleados y de la autenticación de su identidad mediante un censor biométrico de huellas dactilares. La infraestructura general del sistema está compuesta por varios módulos, entre los que se destaca el módulo de procesamiento de imágenes; que entre otras tareas permite realizar un control de calidad sobre la imagen de la huella dactilar, y ejecutar el proceso de extracción de características; además, implementa algoritmos propios para el reconocimiento de la huella dactilar. (13).

1.4.2.2 BYOMESYS AFIS

BYOMESYS AFIS es un Sistema Automatizado de Identificación Dactiloscópica, diseñado para reducir los tiempos y aumentar la efectividad, en la identificación y autenticación de personas, a partir de las impresiones dactilares. En su versión Civil permite la identificación y autenticación de personas de forma rápida y segura detectando los intentos de suplantación de identidad de personas, viabilizando la entrega de documentos de identidad con un grado de confianza acorde con la relevancia de la información que se procesa. Se integra a un motor de identificación que responde a solicitudes de identificación y autenticación utilizando datos en formato Ansi-Nist. El módulo de procesamiento de imágenes permite la compresión en formato de cuantificación escalar

Wavelet (*WSQ por sus siglas en inglés Wavelet Scalar Quantization*), aplicar control de calidad y realizar el proceso de extracción de características de las imágenes de las huellas dactilares. (14).

1.5 Valoración del estado del arte.

Luego del estudio realizado sobre las herramientas informáticas que realizan el proceso de comparación de huellas dactilares ninguno de los sistemas analizados resultó ser candidato para integrarlo como componente del sistema de verificación dactilar debido a que estos están soportados sobre tecnologías propietarias y no contribuyen a dar solución al objetivo planteado, pero se decide utilizar la biblioteca de clases SourceAfis aunque esta soportada en ".Net", teniendo en cuenta que es de código abierto y cuenta con una versión alfa no funcional en Java que se utiliza como base para la implementación del proceso de comparación de huellas dactilares.

1.6 Tecnología, metodología y herramientas empleadas.

1.6.1 Metodología de desarrollo de software.

Lograr la construcción de un sistema informático eficiente, que cumpla con los requerimientos planteados, es una tarea realmente intensa y sobre todo difícil de cumplir. Las metodologías para el desarrollo del software imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Una metodología de desarrollo de software tiene como principal objetivo aumentar la calidad del software que se produce en todas y cada una de sus fases de desarrollo. No existe una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto (equipo de desarrollo, recursos, etc.) exigiéndose así que el proceso sea configurable. Las metodologías de desarrollo se pueden dividir en dos grupos de acuerdo con sus características y los objetivos que persiguen: ágiles y robustas (15).

1.6.1.1 Metodologías ágiles.

Las metodologías ágiles se caracterizan principalmente por el uso de técnicas para agilizar el desarrollo del software, así como de una mayor flexibilidad para adaptarse a los cambios en los requisitos del proyecto, los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados, es más importante crear un producto de software que

funcione, que escribir documentación exhaustiva, la capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan, la colaboración con el cliente debe prevalecer sobre la negociación de contratos. Ejemplo de las metodologías ágiles: XP (por sus siglas en inglés *eXtreme Programming*), *Scrum* y *Crystal Methodologies* (16).

Programación Extrema (XP): esta metodología ágil está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores y propiciando un buen clima de trabajo. Esta metodología se basa en la realimentación continua entre el cliente y el equipo de desarrollo, la simplicidad en las soluciones implementadas y el coraje para enfrentar los cambios. XP se define como la adecuada especialmente para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. Entre las principales características que presenta XP se encuentran las historias de usuario, técnica utilizada para especificar los requisitos del software (17).

***Crystal Methodologies*:** esta metodología ágil define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características son: El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Estas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para la coordinación e integración (18).

1.6.1.2 Metodologías pesadas.

Las metodologías pesadas o tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del *software*, con el fin de conseguir un *software* más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan

adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno donde los requisitos no pueden predecirse o bien pueden variar. Una de las metodologías pesadas más conocida y usada es la Metodología RUP (Rational Unified Process) (19).

Rational Unified Process (RUP): es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado **UML**, constituye la metodología estándar más empleada para el análisis, implementación y documentación de sistemas.

RUP está definido de manera general en 9 flujos de trabajo. Existen 6 que se conocen como flujo de ingeniería: Modelado del negocio, Requisitos, Análisis y diseño, Implementación, Pruebas y Despliegues. Los tres flujos restantes se conocen como actividades sombrilla o flujo de apoyo y tienen gran significación durante las 4 fases de desarrollo del software: Gestión del cambio y configuraciones, Gestión del proyecto y Entorno.

RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades. Estas fases son conocidas como Inicio, Elaboración, Construcción y Transición (20).

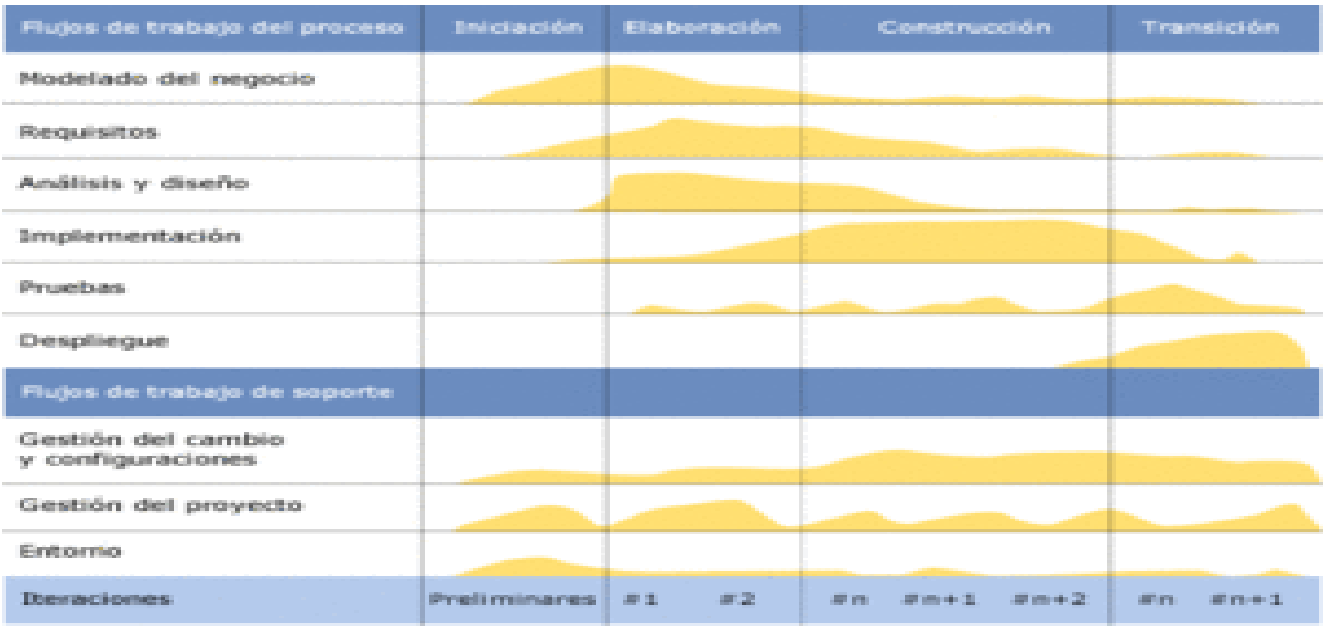

Figura 4: Ciclo de vida de RUP.

El ciclo de vida de RUP está caracterizado por:

- **Centrado en la arquitectura:** La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente.

- **Iterativo e Incremental:** RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros. En cada ciclo de iteración, se hace exigente el uso de artefactos lo que permite alcanzar un grado de certificación en el desarrollo del software.

- **Dirigido por Casos de Uso:** Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo pues los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

1.6.2 Lenguaje de modelado.

Un lenguaje para el modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software. El uso de un lenguaje de modelado es más sencillo que la auténtica programación, pues existen menos medios para verificar efectivamente el funcionamiento adecuado del modelo. Esto puede suponer también que las interacciones entre partes del programa den lugar a sorpresas cuando el modelo ha sido convertido en un software que funciona (21).

Lenguaje unificado de modelado (UML): este lenguaje de modelado de sistema es el más conocido y utilizado en la actualidad. Es un lenguaje gráfico que se desarrolló con el objetivo de visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del

sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Hay que recalcar que UML es un **lenguaje** para especificar y no para describir métodos y procesos. UML es un sistema definido para detallar los artefactos del sistema, para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo (22).

1.6.3 Herramienta CASE.

CASE (por sus siglas en inglés ***C**omputer **A**ided **S**oftware **E**ngineering*, y en español significa Ingeniería de Software Asistida por Computadora): El concepto de CASE es muy amplio; y una buena definición genérica, que pueda abarcar esa amplitud de conceptos, sería la de considerarla como la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, procedimientos y su respectiva documentación. El uso de estas herramientas, permite desarrollar proyectos informáticos que tengan como objetivo la automatización de procedimientos administrativos; o sea las herramientas CASE representan una forma que permite modelar los Procesos de Negocios de las empresas y desarrollar los Sistemas de Información Gerenciales (23).

Visual Paradigm: es un potente conjunto de herramientas CASE que utiliza UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Su diseño está centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad, permite realizar ingeniería directa e inversa sobre el software, también dibujar todos los tipos de diagramas de clases, además de generar documentación, etc. Proporciona abundantes tutoriales de UML, demostraciones interactivas de UML y proyectos UML. Presenta licencia gratuita y comercial, además que es fácil de instalar, actualizar y es compatible entre ediciones (24).

Rational Rose: es una herramienta de producción y comercialización establecida por Rational Software Corporation. Utiliza el Lenguaje Unificado (UML) para facilitar la captura de dominio de la semántica, la arquitectura y el diseño. Este software tiene la capacidad de crear, ver, modificar y

manipular los componentes de un modelo. No es gratuito, admite la integración con otras herramientas de desarrollo. Habilita asistentes para crear clases y provee plantillas de código que pueden aumentar significativamente la cantidad de código fuente generada (24).

1.6.4 Lenguajes de Programación.

Un lenguaje de programación es un idioma artificial diseñado para diseñado para expresar procesos que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Los lenguajes de programación están formados por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Un proceso el cual se escribe, se depura, se prueba, se compila y se mantiene el código fuente de un programa informático se le denomina "**programación**". (25).

Lenguaje C#: es un lenguaje de programación orientado a objeto, fue elaborado por Microsoft para su plataforma ".Net". Las principales figuras en la creación de este lenguaje fueron Scott Wiltamuth y Anders Hejlsberg, este último también diseñó el lenguaje Turbo Pascal y la herramienta RAD Delphi. (26).

Aunque en la plataforma ".Net" es posible escribir código en varios lenguajes, C# es el único que ha sido diseñado para ser utilizado en ella, por lo que desarrollar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes. Por esta razón, se suele decir que C# es el lenguaje nativo de ".Net". (26).

Características del lenguaje C# (26):

- Sencillez.
- Modernidad.
- Orientación a objetos.
- Orientación a componentes.

- Gestión automática de memoria.
- Seguridad de tipos.
- Instrucciones seguras.
- Sistema de tipos unificado.
- Eficiencia.
- Compatibilidad.

Lenguaje Java: fue desarrollado por Sun MicroSystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria (27).

Este es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar de un humano, el lenguaje genera ficheros de clases compiladas, pero estas son en realidad interpretadas por la "Máquina Virtual de Java", la cual mantiene el control sobre las clases que se estén ejecutando. El mismo es multiplataforma y seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

Su sintaxis ha sido trabajada mejorando la de C++ logrando mayor sencillez y legibilidad. Presenta mayor robustez al simplificar la gestión de memoria y eliminar las complejidades del manejo explícito de punteros (27).

1.6.5 Entornos Integrados de Desarrollo.

Un **entorno de desarrollo integrado**, (**IDE** por sus siglas en inglés de *integrated development environment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa que consiste en un editor de código, un compilador, un depurado y un constructor de interfaz gráfica. Estos IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes, además son proveedores de un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, PHP, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede trabajar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (28).

NetBeans: es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundo el proyecto de código abierto NetBeans en junio de 2000 y continúa siendo el patrocinador principal de los proyectos.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con los APIs de NetBeans y un archivo especial que lo identifica como un módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. (29).

Eclipse: fue desarrollado originalmente por la organización IBM. En la actualidad quien se encarga de su elaboración es la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. Es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores, es una potente y completa plataforma de Programación,

desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Además contiene una atractiva interfaz que lo hace fácil y agradable de usar. (30).

La base para Eclipse es la Plataforma de cliente enriquecido (del Inglés Rich Client Platform RCP). Los siguientes componentes constituyen la plataforma de cliente enriquecido.

- Plataforma principal: inicio de Eclipse, ejecución de plugins.
- OSGi: una plataforma para bundling estándar.
- El Standard Widget Toolkit (SWT).
- JFace: manejo de archivos, manejo de texto, editores de texto.
- El Workbench de Eclipse: vistas, editores, perspectivas, asistentes.

1.6.6 Fundamentación de las herramientas, metodologías y tecnologías a utilizar.

Después de un estudio de las herramientas, metodologías y tecnologías más usadas en la actualidad se propone XP como la metodología de desarrollo de software para el desarrollo del componente ya que resulta la más viable, debido a que se cuenta con un equipo de desarrollo pequeño a los cuales se les imposibilita asumir una metodología robusta por la cantidad excesiva de roles y documentación generada en el ciclo de vida del proyecto. Además la documentación generada en XP, aunque no es de gran alcance posee los instrumentos necesarios para soportar la solución. Donde las historias de usuario son su principal herramienta. Otro de los factores determinantes es la presencia del cliente en el grupo de desarrollo, además, XP es una metodología flexible y orientada a la entrega rápida de resultados tangibles. Independientemente de que es la metodología utilizada en el proyecto.

Se usa java porque se desea que sea multiplataforma, libre y además teniendo en cuenta que el principal objetivo de este componente es que sea utilizado en un applet de java para la integración con el sistema de verificación dactilar.

Para la selección del entorno de desarrollo se tuvieron en cuenta aspectos determinantes como el lenguaje de programación a utilizar, el objetivo específico que se persigue y el dominio o familiarización

por parte del equipo de desarrollo del IDE que contribuyera a la implementación del componente. Como resultado del análisis se decidió utilizar el IDE NetBeans en su versión 7.2.1.

Para modelar el componente se propone Visual Paradigm para UML en su versión 8.0, ya que utiliza un lenguaje estándar común para todo el equipo de desarrollo, facilitando así la comunicación. Permite realizar ingeniería tanto directa como inversa, es de software libre y es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o PDF y es también una herramienta multiplataforma.

1.7 Propuesta de solución.

En la investigación realizada no se encuentran evidencias de sistemas que realicen el proceso de comparación de huellas dactilares en el navegador, pero se encontraron diferentes sistemas que enfocan este proceso de diversas formas. Uno de ellos es la biblioteca de clases SourceAfis la cual es un componente de código abierto, soportado sobre la tecnología ".Net". Esta circunstancia trae como consecuencia que la integración del componente de comparación y el sistema de servicios de verificación en línea sea un proceso difícil y costoso. Por tales razones se propone realizar la implementación del componente de comparación de huellas dactilares del SourceAfis en java ya que tiene una parte implementada que permite comenzar el trabajo, el cual tiene como principal finalidad ser integrado al sistema de servicios de verificación en línea del departamento de Biometría. Facilitando así tanto la integración como la ejecución de dicho proceso directamente desde el navegador

1.8 Conclusiones del Capítulo.

En el presente capítulo se mostraron los resultados del análisis realizado sobre los diferentes aspectos y conceptos involucrados en el proceso de comparación de huellas dactilares con el objetivo de esclarecer el contexto en el que se desarrollará el trabajo investigativo.

El estudio de los diferentes sistemas que realizan el proceso de comparación de huellas dactilares permitió conocer las tendencias actuales de estos procesos para la realización del componente de comparación.

Se explicaron los diferentes algoritmos utilizados para la comparación de huellas dactilares, así como algunas dificultades que pueden existir en este proceso. Además se analizaron algunas de las metodologías, herramientas y tecnologías existentes a nivel mundial para el desarrollo del componente.

Capítulo 2: Características del sistema.

En el presente capítulo se definen las características del sistema, se realiza una descripción del dominio del problema, permitiendo especificar correctamente los requisitos funcionales y se realiza además una descripción de los requisitos a través de las historias de usuario. Se realiza una propuesta de la arquitectura del sistema, especificando los patrones de diseño que se utilizarán en el desarrollo de la solución.

2.1 Descripción del problema.

El departamento de Biometría del Centro de Identificación y Seguridad Digital se encuentra desarrollando un sistema de verificación en línea de personas a través de huellas dactilares, entre los componentes que lo integran se encuentra un applet para la comparación de minucias en huellas dactilares en el navegador y que su ejecución no dependa del tipo de navegador utilizado por el cliente; el cual debe desarrollarse usando tecnologías de software libre. Los desarrollos previos de algoritmos que realizan dicho proceso fueron implementados utilizando tecnologías propietarias lo que impide que pueda ser integrado al sistema de verificación dactilar, además en soluciones anteriores todo el proceso de comparación de huellas dactilares es realizado de forma centralizada en el servidor en el que está publicado el sistema, provocando sobrecarga del mismo, además de ralentizar la respuesta ofrecida al cliente y que la integración sea un proceso difícil y costoso. Para que el centro cumpla su objetivo durante el presente trabajo investigativo se desarrollará un componente que pueda ser utilizado del lado del cliente para el proceso de comparación de características en huellas dactilares.

2.2 Modelo de dominio

En el modelo de dominio se muestran los principales conceptos que ayudan a obtener una mejor comprensión del problema. Este modelo contribuirá a describir las clases más importantes dentro del contexto del componente. (31).

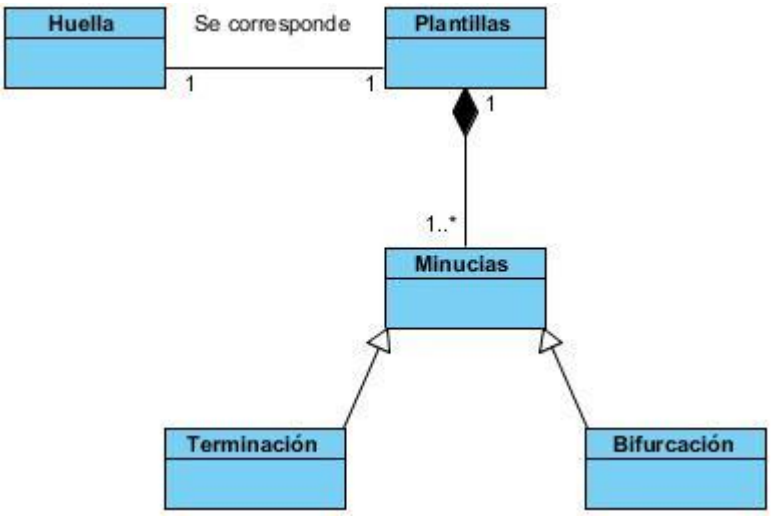

Figura 5: Modelo de dominio.

2.3 Exploración

La primera fase de la metodología XP es la de Exploración, en ella el cliente a través de un proceso de identificación plantea a grandes rasgos las historias de usuario (HU) que son de interés para la primera entrega del producto. En esta fase se realiza una exploración de las posibilidades de la arquitectura del sistema construyendo un prototipo. (32).

2.3.1 Propuesta del sistema

El componente para la comparación de huellas dactilares recibirá dos plantillas biométricas en formato ISO/IEC 19794-2:2011 provenientes del componente de extracción. El objetivo de este componente es la realización de la comparación de plantillas de minucias para conocer si las dos plantillas corresponden o no a la misma persona.

2.3.2 Descripción de la arquitectura.

La arquitectura de software representa un modelo estructurado, y consiste en un conjunto de patrones que brindan un marco de referencia para guiar el desarrollo de aplicaciones informáticas. Permitiendo a los desarrolladores enfocarse en una misma línea de trabajo y cubrir todas las restricciones y

necesidades del producto software. Estableciendo además la estructura, funcionamiento e interacción entre las partes del mismo.

En la arquitectura **cliente/servidor** las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Y se caracteriza por ser el modelo de interacción más común entre aplicaciones en una red. La arquitectura Cliente/Servidor es la integración distribuida de un sistema en red, que cuenta con los recursos, medios y aplicaciones definidos modularmente en los servidores. Los cuales administran, ejecutan y atienden las solicitudes de los clientes; estableciendo así un enlace de comunicación transparente entre los elementos que conforman la estructura.

Por otra parte, el patrón arquitectónico **modelo vista controlador** (MCV) separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El modelo (model) contiene la información central y los datos. Las vistas (view) despliegan información al usuario. Los controladores (controllers) capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario.

El estilo arquitectónico **tuberías y filtros** descompone el sistema en módulos funcionales, permitiendo transformar el flujo de datos y la interacción sucesiva entre estos. Se aplica cuando los datos de entrada son transformados a través de una serie de componentes computacionales o manipulativos en los datos de salida. Un patrón tubería y filtro tiene un grupo de componentes llamados filtros, conectados por tuberías que transmiten datos de un componente al siguiente. El filtro está diseñado para recibir entrada de datos de una forma y producir la salida de una forma específica.

El estilo arquitectónico **orientado a objetos** se basa en objetos que permite organizar un programa de acuerdo a abstracciones de más alto nivel. Los datos globales desaparecen y se convierten en parte interna de los objetos, por lo tanto los cambios generados en algún objeto solamente afectarían a él sin modificar a los datos referenciado.

Una vez analizados los estilos arquitectónicos anteriores y teniendo en cuenta las características del componente que se desea desarrollar se decide utilizar el estilo arquitectónico Orientado a objetos con el objetivo de asegurar la reutilización del mismo.

2.3.3 Requisitos funcionales (RF)

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares.

Los requisitos funcionales describen lo que el sistema debe hacer. Estos requisitos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requisitos. (33).

- **RF 1:** Obtener plantillas de minucias a utilizar.
- **RF 2:** Efectuar la comparación de las plantillas.
- **RF 3:** Mostrar resultados de la comparación.

2.3.4 Historias de Usuarios (HU)

Durante el desarrollo de la solución las historias de usuario permiten realizar una especificación detallada de las funcionalidades que el sistema debe implementar para obtener un resultado acorde a las necesidades del cliente.

A continuación se describe la HU asociada a la funcionalidad Efectuar la comparación de plantillas. Para consultar el resto de las HU dirigirse al **Anexo 2. Historias de usuario:**

Historia de Usuario	
Número: 1	**Usuario:** Sistema
Nombre historia: Efectuar la comparación de plantillas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 12	**Iteración asignada:** 2 y 3

Programador responsable: Asiel Echemendia Carrasco
Descripción: Con el objetivo de realizar el proceso de comparación de huella dactilar es necesario realizar los siguientes escenarios que dan paso al cumplimiento de esta funcionalidad: **Escenario Efectuar la comparación:** La comparación consiste en encontrar el mayor número posible de coincidencias entre pares de minucias de la huella modelo y la huella a comparar.
Observaciones: Para realizar este proceso se debe disponer de dos plantillas biométricas o plantillas de minucias resultantes del proceso de extracción de características.
Prototipo de interfaces: Sin prototipo de interfaz.

Tabla 1: HU_Efectuar la comparación de plantillas.

2.3.5 Requisitos No Funcionales (RNF)

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. (35).

NÚMERO RNF	DESCRIPCIÓN	CLASIFICACIÓN
1	➢ JRE en su versión 7.5	Software
2	➢ PC Intel Pentium 4 o superior. ➢ CPU 1.0 GHz o superior. ➢ 80 GB HDD o superior. ➢ 1 GB de Memoria RAM o superior.	Hardware
3	El componente está concebido para un ambiente donde se integre a un sistema de identificación, por tanto los tiempos de respuestas deben ser rápidos, así como la velocidad de procesamiento de la información.	Rendimiento

4	La implementación del componente debe ser desarrollada en el lenguaje java.	Restricciones en el Diseño y en la Implementación

Tabla 2: Requisitos no funcionales del componente.

2.4 Planificación

La planificación es otra de las fases de la metodología XP, la cual se plantea como un diálogo continuo entre las partes involucradas en el proyecto. Es una fase corta, donde el cliente y el grupo de desarrollo priorizan el orden de implementación de las historias de usuario, así como también su entrega. Esta planificación debe de ser flexible para que pueda adaptarse a los posibles cambios que puedan surgir. (32).

2.4.1 Estimación del esfuerzo por historia de usuario

Las estimaciones del esfuerzo asociado a la implementación de las historias de usuario la establecen los programadores utilizando como medida el valor estimado que le asigna el grupo de desarrollo al esfuerzo necesario para implementar la historia de usuario. Dicho valor, equivale a una semana ideal de programación. Los resultados obtenidos de acuerdo a las HU en esta estimación se exponen en la siguiente tabla:

No.	Historia de Usuario	Estimación (Semanas)
1	Obtener plantillas de minucias a utilizar.	6
2	Efectuar la comparación de las plantillas.	12

3	Mostrar resultado de la comparación.	4

Tabla 3: Estimación del esfuerzo por historia de usuario.

2.4.2 Plan de iteraciones

Una vez identificadas y descritas las historias de usuarios; y estimado el esfuerzo dedicado a la realización de cada una de ellas, se procede a confeccionar el plan de iteraciones. El cual constituye una breve planificación de lo que sería la fase de implementación. En esta fase las historias de usuario son traducidas como tareas de programación, que serán desarrolladas y probadas a través del orden que se define en el plan. El plan de iteraciones que se elaboró para el desarrollo del componente consta de cuatro iteraciones las cuales se corresponden con cada una de las HU:

Iteración 1

En esta primera iteración se va a implementar la historia de usuario HU_Obtener .plantillas de minucias a utilizar.

Iteración 2, 3

En estas iteraciones se van a implementar la historia de usuario HU_Efectuar la comparación de las plantillas.

Iteración 4

En esta iteración se va a implementar la historia de usuario HU_Mostrar resultados de la comparación.

2.4.3 Plan de duración de las iteraciones

El plan de duración de las iteraciones persigue el objetivo de mostrar en qué orden serán implementadas las historias de usuario en cada iteración, así como el tiempo destinado a cada una de ellas, permitiendo al grupo de trabajo y al cliente, tener una idea general de cuánto durará la confección del componente.

NO.	HISTORIA DE USUARIO	DURACIÓN DE LAS ITERACIONES
Iteración # 1	Obtener plantillas de minucias a utilizar.	6
Iteración # 2, 3	Efectuar la comparación de las plantillas.	12
Iteración # 4	Mostrar resultados de la comparación.	4

Tabla 4: Plan de iteraciones.

2.4.4 Plan de entrega

El plan de entrega está compuesto por una aproximación de la fecha límite que se corresponde con la entrega de artefactos de funcionalidad resultante de cada iteración.

NO. DE ITERACIÓN	FECHA LÍMITE
Iteración # 1	20/12/2012
Iteración # 2, 3	20/04/2013
Iteración # 4	10/05/2013

Tabla 5: Plan de entrega.

2.5 Diseño

La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e implementable que a la larga costará menos tiempo y esfuerzo desarrollar. (32).

2.5.1 Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Se consideran como procedimientos para solucionar diversas veces un problema del mismo tipo y son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Los desarrolladores lo usan como una forma de reutilizar la experiencia, clasificando las soluciones con términos de común denominación (36).

En el diseño del componente se utilizarán los patrones generales de software para la asignación de responsabilidades o patrones GRASP (***G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns*), los cuales describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. Los patrones GRASP empleados se describen a continuación (37):

Experto: el uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide (se conserva el encapsulamiento), esto favorece la existencia de mínimas relaciones entre las clases permitiendo contar con un sistema sólido y fácil de mantener (soporta un bajo acoplamiento). El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando con ello definiciones de clases sencillas y más cohesivas siendo más fáciles de comprender y de mantener. (36, 37).

Creador: se considera para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por el objeto que contiene la información necesaria para ello. El uso de este patrón admite crear las dependencias mínimas necesarias entre las clases, beneficiando el mantenimiento del sistema y brindando mejores oportunidades de reutilización (brinda soporte a un bajo acoplamiento). (36, 37).

```
public class PreparedProbe
{
    public ProbeIndex ProbeIndex = new ProbeIndex();
}
```

Figura 6: Ejemplo del Patrón Creador.

Bajo acoplamiento: soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta Cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades. (36, 37).

```
public class AfisEngine
{
    private int DPIVALUE = 500;
    private int MINMATCHES = 1;
    private float THRESHOLD = 25;

    private ParallelMatcher matcher;

    public AfisEngine()
    {
        matcher = new ParallelMatcher();
    }
```

Figura 7: Ejemplo del Patrón Bajo Acoplamiento.

Alta Cohesión: una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla, además puede destinarse a un propósito muy específico. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos. A menudo se genera un bajo acoplamiento. (36, 37).

```
public final class IsoFormat extends TemplateFormatBase<byte[]>
{
    @Override
    public byte[] exportTemplate(TemplateBuilder builder)
    {...}

    @Override
    public TemplateBuilder importTemplate(byte[] template)
    {...}

    @Override
    public  void serialize(OutputStream stream, byte[] template)
    {...}

    @Override
    public  byte[] deserialize(InputStream stream)
    {...}
}
```

Figura 8: Ejemplo del Patrón Alta Cohesión.

2.5.2 Diagrama de clases

El diagrama de clases describe la estructura de un sistema mostrando sus clases y relaciones entre ellas. Son utilizados durante el proceso de análisis y diseño para elaborar un diseño conceptual de la información que se manejará en el sistema y los componentes por los que este estará integrado. (38). Para consultar el diagrama de paquetes dirigirse al **Anexo 3. Diagrama de paquetes.**

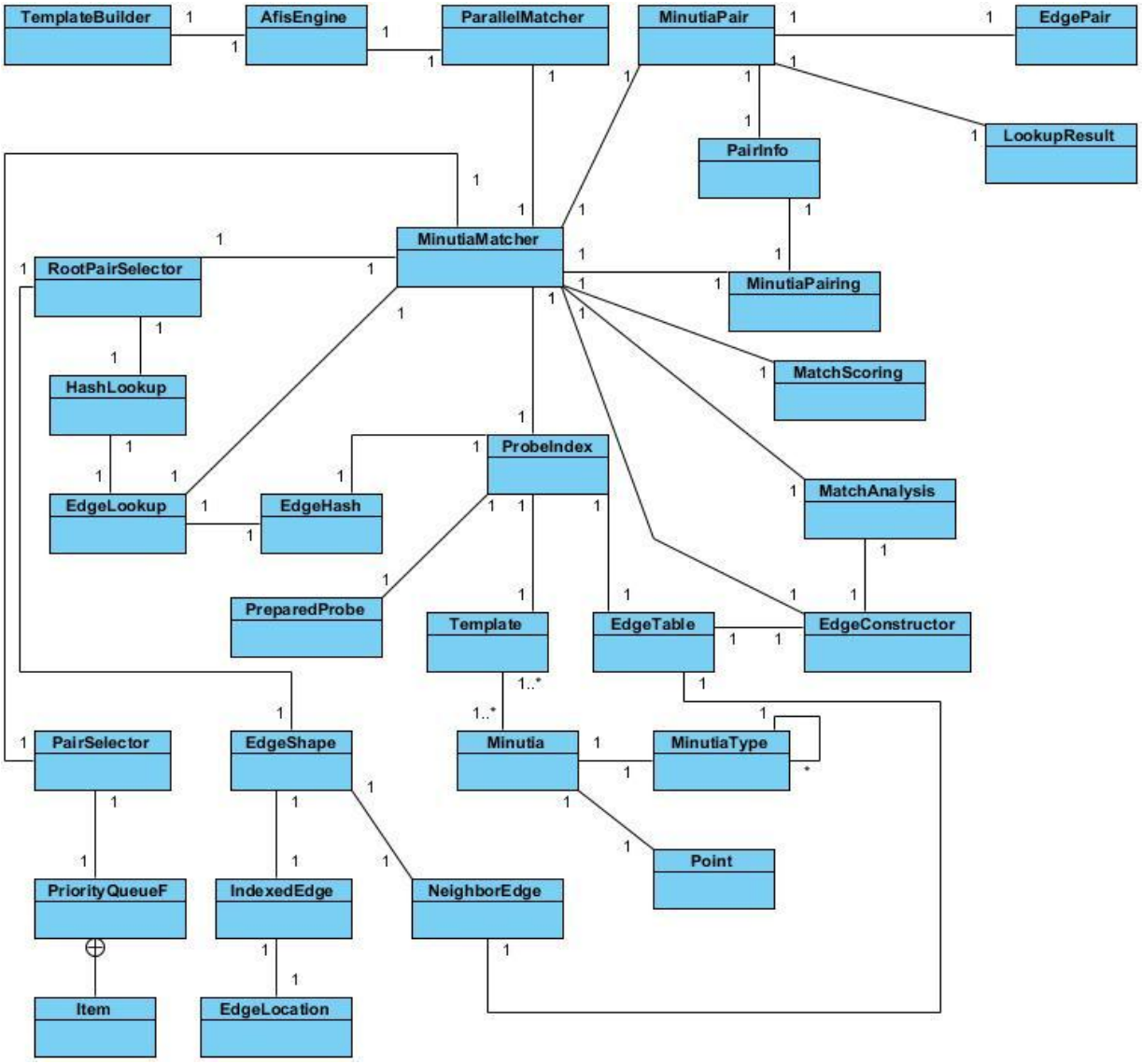

Figura 9: Diagrama de clases.

2.5.3 Tarjetas CRC

Las tarjetas CRC (Class-Responsibility-Collaboration) es la técnica de diseño orientada a objetos que Kent Beck y Ward Cunningham propusieron para la metodología XP. (39).

Las principales características de estas tarjetas son:

- Identificación de clases y asociaciones que participan del diseño del sistema.
- Obtención de las responsabilidades que debe cumplir cada clase.
- Establecimiento de cómo una clase colabora con otras clases para cumplir con sus responsabilidades.

El objetivo de las tarjetas CRC es tener el control de las clases que se necesitaran para la implementación del sistema y la forma que interactúan entre ellas, de esta forma se facilita el análisis y la discusión de las mismas por parte de varios actores del equipo de proyecto con el objetivo de que el diseño sea lo más simple posible verificando las especificaciones del sistema. (39).

A continuación se muestra la tarjeta CRC de la clase AfisEngine. Para consultar el resto de las tarjetas CRC dirigirse al **Anexo 4. Tarjetas CRC**:

AFISENGINE	
ESTA CLASE CUENTA CON LA FUNCIONALIDAD PRINCIPAL DEL SISTEMA. DESPUÉS DE HABER ESTABLECIDO PROPIEDADES RELEVANTES EN PARTICULAR EL UMBRAL (THRESHOLD), LA APLICACIÓN PUEDE LLAMAR A LOS MÉTODOS PRINCIPALES: VERIFY: ESTE MÉTODO REALIZA LA COMPARACIÓN DE DOS TEMPLATEBUILDER QUE SE PASAN POR PARÁMETRO CUANDO EL MÉTODO ES INVOCADO. VERIFY_TEMPLATEBUILDER: ESTE MÉTODO REALIZA LA COMPARACIÓN DE DOS FILE QUE A SU VEZ SON DESERIALIZADO Y CONVERTIDOS EN TEMPLATEBUILDER, DESPUÉS ES INVOCADO EL MÉTODO VERIFY.	➢ **PARALLELMATCHER** ➢ **PREPAREDPROBE** ➢ **BESTMATCHSKIPPER** ➢ **TEMPLATE** ➢ **TEMPLATEBUILDER** ➢ **ISOFORMAT** ➢ **FILE**

Tabla 6: Tarjeta CRC, Clase AfisEngine.

2.6 Conclusiones del capítulo

Durante el desarrollo de este capítulo se exploraron las principales características del componente, permitiendo construir el modelo de dominio para un mejor entendimiento del problema. Se identificaron los requisitos funcionales y no funcionales con los que debe cumplir el sistema, brindando al equipo de desarrollo una visión del producto deseado por el cliente. Como principal artefacto de la metodología XP se generaron las Historias de usuario que describen y se corresponden con los requisitos funcionales identificados. Estas permitieron una estimación del esfuerzo por Historias de usuario, obteniéndose así un plan de duración y de entrega de la primera versión del componente.

Capítulo 3: Implementación y Pruebas.

En el presente capítulo se analizan los aspectos relacionados con la implementación del componente para la comparación de huellas dactilares y se realizan las pruebas del sistema, donde se valida el funcionamiento de los requisitos funcionales.

3.1 Diseño de Implementación.

3.1.1 Estado de implementación en la versión no funcional del SourceAfis en Java.

En el momento de iniciar la implementación del componente, la versión alfa no funcional del SourceAfis en java contaba con un 50% de clases implementadas, de ellas el 45% eran clases totalmente implementadas y el 5% clases parcialmente implementadas. Con relación al 100% el 50% eran clases no implementadas que eran necesarias para poder realizar el proceso de comparación de minucias en huellas dactilares.

En el siguiente gráfico se representa lo analizado anteriormente:

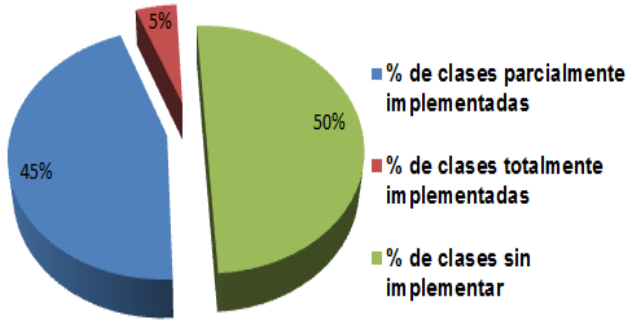

Figura 10: Representación del estado de la implementación de la versión no funcional del SourceAfis en Java.

Después de haber analizado la versión alfa no funcional del SourceAfis en java se detectó un error en la clase **Calc** que afecta los resultados del proceso de comparación de minucias en huellas dactilares.

En la implementación se programó lo siguiente:

```
public static Point Multiply(float scale, Point point) {
```

return new Point (Calc.toInt32(scale * point.X), **Calc.toInt32(scale) * point.Y)**;

}

Cuando realmente se debía haber programado:

public static Point Multiply(float scale, Point point) {

return new Point (Calc.toInt32(scale * point.X), **Calc.toInt32(scale * point.Y))**;

}

3.1.2 Plantilla a utilizar.

La plantilla a usar es basada en una estructura diseñada para almacenar la información relativa a cada minucia (obtenida del proceso de extracción), se podrá almacenar de cada minucia el punto de referencia en el plano bidimensional, además del tipo de minucia, entre otros elementos.

3.1.3 Descripción del algoritmo.

El algoritmo utilizado por el SourceAfis para la comparación de plantillas de minucias es el basado en el análisis de las minucias que se trata de la técnica más extendida. Las minucias se leen desde las plantillas de minucias y se almacenan en vectores como conjuntos de puntos en el plano bidimensional de la imagen. Este algoritmo basado en la comparación de minucias trata de conseguir el mayor número posible de coincidencias entre pares de minucias de la plantilla modelo y la plantilla a comparar para decidir si dos plantillas pertenecen o no a un mismo individuos en singular.

3.1.4 Problemas durante la implementación.

Durante el desarrollo del componente se detectaron errores asociados a la diferencia entre los lenguajes de programación Java y C#. A continuación se muestran de los problemas que se encontraron y la forma en la que se le dio solución.

- En la clase BestMatchSkipper en el método GetSortedScores() se observa este fragmento de código:

SourceAfis C#:

```
Array.Sort(results, (left, right) => Calc.Compare(right.Score, left.Score));
```

Componente Java:

```
Arrays.sort(results,new Comparator<PersonsSkipScore>()

{public int compare(PersonsSkipScore left, PersonsSkipScore right) { Calc new_calc = new Calc();

return new_calc.Compare(right.score, left.score);}});
```

- Durante la implementación de algunos métodos relacionados con el procesos de comparación se pudo apreciar que la representación del rango de valores del tipo de dato “byte” en C# (va de 0 a 255) y en Java (va de -128 a 127) son diferentes. Por este motivo a la hora de calcular algunos valores en Java el resultado no era el mismo al de C#. Este problema se puede observar en la clase EdgeLookup específicamente en el método HashCoverage.

SourceAfis C#:

```
int minReferenceBin = Angle.Difference(edge.ReferenceAngle, MaxAngleError)/MaxAngleError;

int maxReferenceBin = Angle.Add(edge.ReferenceAngle, MaxAngleError) / MaxAngleError;

int endReferenceBin = (maxReferenceBin + 1) % angleBins;

int minNeighborBin = Angle.Difference(edge.NeighborAngle, MaxAngleError) / MaxAngleError;
```

Componente Java:

```
int minReferenceBin = (angle.Difference(edge.referenceAngle, maxAngleError) & 0xFF) / maxAngleError;
```

int maxReferenceBin = (angle.Add(edge.referenceAngle, maxAngleError) & 0xFF)/ maxAngleError;

endReferenceBin = (maxReferenceBin + 1) % angleBins;

int minNeighborBin = (angle.Difference(edge.neighborAngle, maxAngleError) & 0xFF) / maxAngleError;

- En la implementación durante la declaración de variables se encontraron problemas de compatibilidad por el tipo de datos porque en C# existen tipos de datos que no son compatibles con java.

SourceAfis C#

var innerDistanceRadius;

var innerAngleRadius;

var probeEdge;

var candidateEdge;

Componente en Java:

int innerDistanceRadius;

int innerAngleRadius;

EdgeShape probeEdge;

EdgeShape candidateEdge;

3.2 Estilo de codificación.

Para que el código pueda ser entendido y modificado fácilmente por cada miembro del equipo es imprescindible el uso de estilos de codificación. Cuando una codificación es bien definida se logra que

todo el equipo se sienta cómodo con el código escrito por otro miembro del mismo. Los estilos definidos en este trabajo son los siguientes:

Se contemplaron dos estilos de espacios en blanco distintos, que se utilizan en determinados casos:

- Línea en blanco: se utilizará una línea en blanco en las clases para separar el fin de un método con el inicio del otro.
- Separaciones entre términos: en una lista de parámetros se pondrán espacios tras las comas, pero nunca tras o antes de los paréntesis. En las asignaciones se pondrán espacios antes y después del "=".

Estilos de capitalización:

- Capitalización Pascal: en este estilo se capitaliza la primera letra de cada palabra (ApplyThresHold), usada en las declaraciones de métodos.
- Capitalización Camel: en este estilo se capitaliza la primera letra de cada palabra excepto la primera (loadTemplate), usada en la declaraciones de variables.
- Capitalización Versal: es aquella en la que todo se escribe con mayúsculas (THRESHOLD), usada en la declaración de constantes.

Los comentarios:

- Deben ser suficientes para el mejor entendimiento del código.
- Se utilizará solamente el formato //, nunca se recurrirá al /*….*/. Cando existan más de una línea para comentar se utilizaran tantos // al principio de línea como convengan.

3.2.1 Ejemplo de código.

```
//Este método es invocado despues de cargar los template y deserializarlos
//para realizarles todos el proceso de comparación
public synchronized boolean Verify(TemplateBuilder loadTemplate,
        TemplateBuilder loadTemplate1)
{
    Template probe = new Template(loadTemplate);
    Template candidate = new Template(loadTemplate1);
    BestMatchSkipper collector = new BestMatchSkipper(1, MINMATCHES-1);
    List<Template> candidateTemplates=new ArrayList<Template>();
    candidateTemplates.add(candidate);
    PreparedProbe probeIndex = matcher.prepare(probe);
    float[] scores = matcher.Match(probeIndex, candidateTemplates);
    for (float score :scores)
    {
        collector.addScore(0, score);
    }
    return ApplyThresHold(collector.getSkipScore(0));
}
```

Figura 11: Ejemplo de código.

3.3 Componente de comparación de huellas dactilares.

Para el desarrollo de este componente se tomó como base la biblioteca de clases del SourceAfis en C#, después de un análisis profundo del funcionamiento del mismo, se decide pasar a la implementación del nuevo Componente de comparación de huellas dactilares en Java, realizando la transformación de lenguajes:

- Para la comparación no se trabaja directamente con una imagen sino con una plantilla que se obtiene como resultado del proceso de extracción.
- Con respecto al SourceAfis los tiempos de respuesta del componente son más rápidos.
- El componente podrá ser embebido en un applet para integrarlo a algún sistema de verificación de individuos mediante huellas dactilares y/o puede ser utilizado de manera independiente en cualquier otra solución que requiera del servicio de comparación.

3.4 Descripción de la interfaz de prueba.

La interfaz es el medio por el cual el usuario interactúa con un sistema, los sistemas normalmente suelen tener interfaces que deben ser fáciles de entender y de accionar. El componente no tiene interfaz pero

para su desarrollo y prueba se hizo necesario desarrollar una interfaz de prueba la que presenta un diseño sencillo y amigable.

Utilizando la interfaz a la que se hizo referencia anteriormente se puede realizar la comparación entre dos plantillas o se puede comparar una plantilla con las restantes. Para realizar al comparación entre dos plantillas se presiona el botón match1x1 y luego se seleccionan los dos elementos que se deseen comparar. Para comparar una plantilla contra muchas plantillas, solo se debe dar clic en el botón matchAll y cargar la plantilla que se desea comparar con el resto de las plantillas; para ambos casos luego de realizar el proceso se muestra un mensaje informando si fue aceptado (Identificado) o rechazado (No identificado) y el tiempo que tardó en realizar el proceso. Ver **Anexo 6. Interfaz de prueba del componente.**

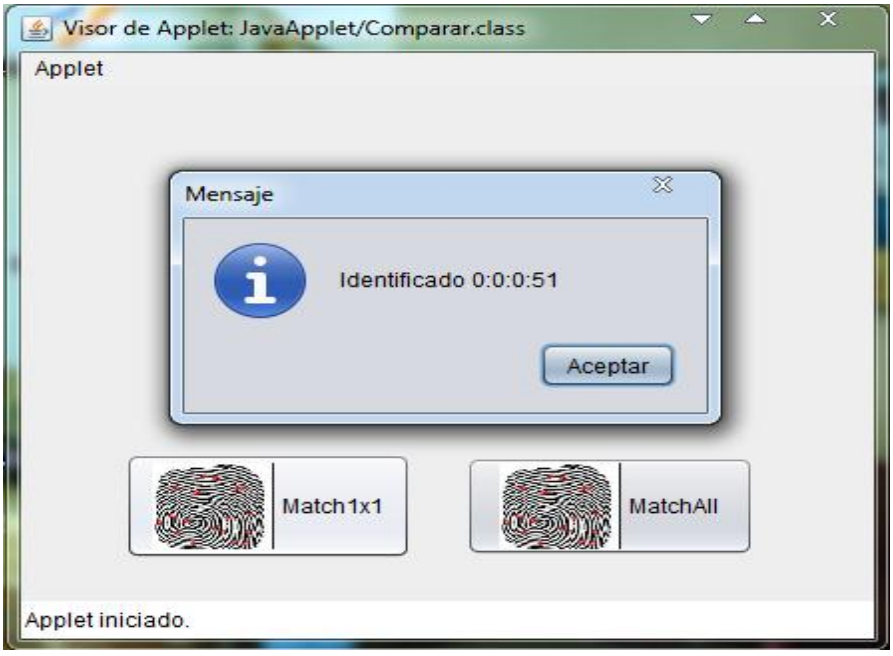

Figura 12: Vista del resultado de ejecutar la comparación una a una.

3.5 Diseño de pruebas.

Un aspecto muy importante de rendimiento en los sistemas biométricos es la precisión del proceso de verificación o identificación. En la actualidad se consideran tres parámetros que ayudan a determinar la precisión del proceso de una manera cuantitativa:

- **Tasa de falso rechazo (FRR):** es el porcentaje de usuarios autorizados que tratan de acceder al sistema y este los declara como no autorizados.
- **Tasa de falsa aceptación (FAR):** es el porcentaje de intentos de accesos de usuarios no autorizados los cuáles el sistema acepta como autorizados.
- **Tasa de igualdad de error (ERR):** es el punto en el cuál FRR y FAR son el mismo valor.

3.5.1 Etapa de Verificación

En la primera etapa del sistema se obtuvo un alto porcentaje de reconocimiento, sin embargo, en algunas pruebas el umbral (Threshold) fue superado por 15 plantillas de diferentes personas, es decir, los valores de coordenadas, ángulos y distancias entre minucias fueron semejantes para más de una persona. Por lo tanto, en esas pruebas el sistema entregaba a la salida más de una plantilla reconocida, esto ocasionaba también, que se aumentara el porcentaje de falsa aceptación. Por esta razón, se realizó una segunda etapa que consiste en verificar las plantillas resultantes. Con esta prueba se eliminan algunas plantillas similares, pero continuaban los problemas de la primera etapa con 8 plantillas. Por este motivo se realiza una tercera etapa donde se trabaja en el análisis de esas plantillas que continuaban con problema y se obtiene un resultado de una sola plantilla que sigue dando el mismo problema de las etapas anteriores.

Cuando dos plantillas de minucias son comparadas puede ocurrir que alguna de ellas aparezca trasladada, todas las minucias de la plantilla se mueven en la misma dirección y la misma cantidad de píxeles. La **Figura 13** muestra este ejemplo.

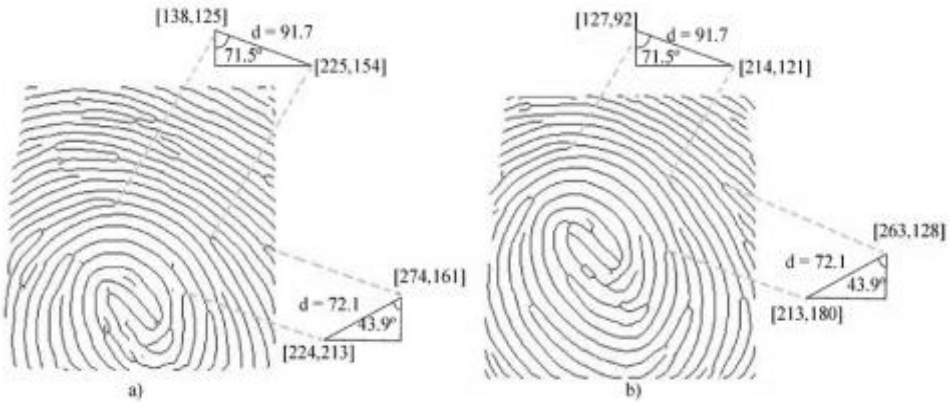

Figura 13: a) Imagen almacenada, b) Imagen de entrada.

La metodología XP tiene como pilar básico la fase de pruebas que promueve a los programadores a probar constantemente el desarrollo de la implementación. Mediante las pruebas que se realizan se minimiza la cantidad de errores no detectados. Esto contribuye a mejorar cada vez más la calidad del producto y la seguridad de los desarrolladores a la hora de realizar cambios.

La metodología XP propone que se hagan pruebas de aceptación para evaluar si al final de cada iteración se obtiene la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la deseada por el cliente.

En la evaluación del componente de comparación se realizaron las pruebas descritas anteriormente para la comprobación del buen funcionamiento del componente. A continuación se muestra el caso de prueba de aceptación que corresponde a la prueba realizada al terminar la primera iteración, asociada a la HU_Efectuar la comparación de las plantillas. Para consultar el resto de los casos de prueba dirigirse al **Anexo 5. Casos de pruebas:**

Caso de prueba de aceptación	
Código de caso de prueba: 1	**Nombre de la historia de usuario:** HU_Efectuar la comparación de las plantillas.

Responsable de la prueba: Asiel Echemendia Carrasco.
Descripción de la prueba: prueba comprobar que el componente realiza el proceso de comparación.
Condición de ejecución: para la realización del proceso se debe recibir dos plantillas de minucias.
Entrada/Pasos de ejecución: luego de cargar las plantillas se realiza el proceso de comparación calculando un valor (score) para compararlo con el umbral (Threshold).
Resultado esperado: se debe obtener como resultado verdadero (en caso de que el score > Threshold) o falso en caso contrario.
Evaluación de la prueba: prueba satisfactoria

Tabla 7: Caso de prueba de aceptación. HU_Efectuar la comparación.

3.6 Resultados obtenidos.

Después de haber realizado las etapas de pruebas se obtuvieron los siguientes resultados:

Etapas	Total de plantillas de minucias	Cantidad de plantillas con errores	% de error
1	900	15	1,67
2	900	8	0,89
3	900	1	0.11

Tabla 8: Resultados de las Pruebas.

La siguiente gráfica muestra los resultados obtenidos luego de las etapas de pruebas:

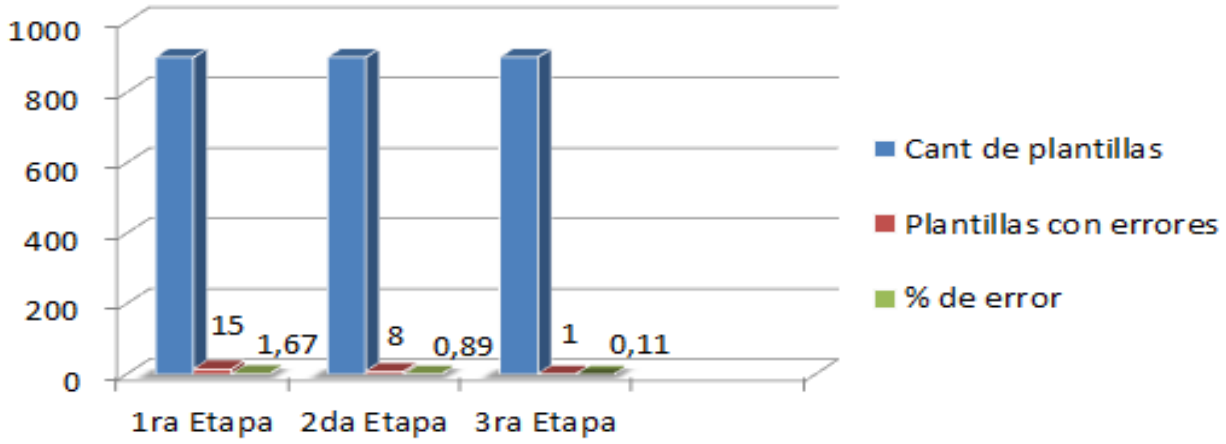

Figura 14: Gráfico del resultado de las pruebas.

3.6.1 Comparación de los resultados obtenidos del componente y los resultados obtenidos del SourceAfis en C#.

Para el análisis de los resultados arrojados por ambos sistemas se realizó en cada uno un método que guardara los resultados de la comparación en un archivo de texto “txt” para luego analizar estos resultados y comprobar que el componente de comparación de minucias en huellas dactilares realizaba el proceso correctamente.

Luego del análisis realizado se llega a la conclusión de que los resultados son satisfactorios, aunque hay que destacar que el valor de similitud calculador (Threshold) en ambos casos no es el mismo, pero esto no afecta en ningún momento la verificación, o sea no cambia la respuesta del componente.

A continuación de muestran resultados obtenidos del proceso de comparación de plantillas biométricas para ambos casos:

Resultados del SourceAfis en C#:

JIsoFormat_08786_1_1.bmp.txt, JIsoFormat_08786_2_1.bmp.txt -----score: 41,463

JIsoFormat_08786_1_1.bmp.txt, JIsoFormat_08786_2_10.bmp.txt -----score: 4,055444

JIsoFormat_08786_1_1.bmp.txt, JIsoFormat_08786_2_2.bmp.txt-----score: 4,33

Resultados del componente de comparación de minucias en Java:

JIsoFormat_08786_1_1.bmp.txt, JIsoFormat_08786_2_1.bmp.txt--------score: 78.416245

JIsoFormat_08786_1_1.bmp.txt, JIsoFormat_08786_2_10.bmp.txt--------score: 11.56121

JIsoFormat_08786_1_1.bmp.txt, JIsoFormat_08786_2_2.bmp.txt--------score: 12.953319

3.7 Conclusiones del capítulo

En el presente capítulo se elaboró la descripción más específica posible de las características del algoritmo de comparación a usar en el componente y además el diseño de la plantilla con que este trabaja, se expusieron las diferencias ante la aplicación SourceAfis. Además se mostraron los resultados de las pruebas de aceptación, comprobando y validando de esta forma el buen funcionamiento del componente.

Conclusiones

El desarrollo del presente trabajo de diploma permitió realizar un estudio sobre los procesos que se realizan para la comparación de huellas dactilares. Una vez concluida la investigación se obtuvo como resultado un componente que automatiza el proceso de comparación de huellas dactilares que podrá ser integrado al sistema de verificación dactilar dándole cumplimiento al objetivo de la investigación; específicamente a los objetivos específicos de la siguiente manera:

- Se realizó un estudio detallado de las tendencias actuales de los algoritmos para la comparación de huellas dactilares.
- Luego del estudio de las tecnologías, metodologías y herramientas que existen actualmente y que podían ser utilizadas en el desarrollo de la solución se escogieron las que más se ajustaban de acuerdo a las características del componente.
- Se realizó un análisis del sistema obteniendo como resultado los requisitos que debía cumplir el mismo y luego se realizó un diseño detallado del mismo.
- Se implementó un algoritmo para la comparación de huellas dactilares obteniendo como resultado un componente que puede ser utilizado para conformar un applet de java e integrarlo al sistema de servicios de verificación.
- Se le realizaron pruebas al componente con el fin de validar la solución desarrollada y verificar el correcto funcionamiento del componente.

Referencias Bibliográficas

1. Biometría - Dactilar. [online]. [Accessed 20 November 2012]. Available from: http://www.biometria.gov.ar/metodos-biometricos/dactilar/

2. Portal DATYS. [online]. [Accessed 20 November 2012]. Available from: http://www.datys.cu/

3. IEEEpaper.dvi - biometrics04.pdf [online]. [Accessed 26 May 2013]. Available from: http://bytelabs.org/pub/papers/biometrics04.pdf

4. Características de un sistema Biométrico. [online]. [Accessed 13 January 2013]. Available from: http://redyseguridad.fi-p.unam.mx/proyectos/biometria/basesteoricas/caracteristicassistema.html

5. Las palabras del silencio: Huellas dactilares... [online]. [Accessed 13 January 2013]. Available from: http://mar-palabrasilencio.blogspot.com/2012/10/huellas-dactilares.html

6. Huellas. [online]. [Accessed 13 January 2013]. Available from: http://www.slideshare.net/sotolargo/huellas-4519116

7. Biometrics: fingerprint: algorithms. [online]. [Accessed 15 January 2013]. Available from: http://fingerchip.pagesperso-orange.fr/biometrics/types/fingerprint_algo.htm

8. Tesis_Almudena_Lindoso_Munoz.pdf [online]. [Accessed 26 May 2013]. Available from: http://e-archivo.uc3m.es/bitstream/10016/5571/1/Tesis_Almudena_Lindoso_Munoz.pdf

9. doi:10.1016/j.patcog.2005.09.005 - fingerprintmatchingby06.pdf [online]. [Accessed 26 May 2013]. Available from: http://www.vislab.ucr.edu/PUBLICATIONS/pubs/Journal%20and%20Conference%20Papers/after10-1-1997/Journals/2006/fingerprintmatchingby06.pdf

10. SISTEMA BIOMÉTRICO DE RECONOCIMIENTO DE HUELLAS DIGITALES.PDF - 34215_1.pdf [online]. [Accessed 26 May 2013]. Available from: http://repositorio.ute.edu.ec/bitstream/123456789/5634/1/34215_1.pdf

11. SourceAFIS. [online]. [Accessed 20 January 2013]. Available from: http://www.sourceafis.org/blog/

12. VeriFinger SDK. [online]. [Accessed 20 January 2013]. Available from: http://www.goit.cl/vfinger.html

13. Biomesys Control de Asistencia V2.0_ET_Es - Biomesys Control de Asistencia V2.0_ET_Es.pdf [online]. [Accessed 26 May 2013]. Available from: http://www.datys.cu/docs/Documentaci%C3%B3n%20Productos/Biomesys%20Control%20de%20Asistencia/Biomesys%20Control%20de%20Asistencia%20V2.0_ET_Es.pdf

14. Especificaciones Técnicas BIOMESYS AFIS Civil - Biomesys AFIS Civil V2.0 ET_20.04.10_ E03 _Es_D.pdf [online]. [Accessed 26 May 2013]. Available from: 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

15. Metodologías de desarrollo de Software - EcuRed. [online]. [Accessed 6 February 2013]. Available from: http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software

16. Metodologías de desarrollo de software. ¿Cuál es el camino? (página 2) - Monografias.com. [online]. [Accessed 7 February 2013]. Available from: http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software2.shtml

17. XP - Extreme Programing Ingeniería de Software. [online]. [Accessed 7 February 2013]. Available from: http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html

18. Crystal Methodologies. [online]. [Accessed 9 February 2013]. Available from: http://crystalmethodologies.blogspot.com/

19. Metodologías tradicionales y metodologías ágiles. [online]. [Accessed 11 February 2013]. Available from: http://www.eumed".Net"/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

20. *METODOLOGIA Rational Unified Process (RUP) - RUP vs. XP.pdf* [online]. [Accessed 26 May 2013]. Available from: http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf

21. *JAVA - ActaUML.PDF* [online]. [Accessed 26 May 2013]. Available from: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF

22. *Uml.PDF - Uml.PDF* [online]. [Accessed 26 May 2013]. Available from: http://www.dccia.ua.es/dccia/inf/asignaturas/GPS/archivos/Uml.PDF

23. Capitulo I HERRAMIENTAS CASE. [online]. [Accessed 20 February 2013]. Available from: http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE

24. Herramientas CASE para el proceso de desarrollo de Software (página 2) - Monografias.com. [online]. [Accessed 21 February 2013]. Available from: http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml

25. *lenguajes_de_programacion_1.pdf* [online]. [Accessed 7 June 2013]. Available from: http://competenciastic.educ.ar/pdf/lenguajes_de_programacion_1.pdf

26. Lenguaje de programacion c#. [online]. [Accessed 27 February 2013]. Available from: http://www.slideshare".Net"/perezinho/lenguaje-de-programacion-c-7869638

27. Historia del lenguaje Java. [online]. [Accessed 27 February 2013]. Available from: http://www.cad.com.mx/historia_del_lenguaje_java.htm

28. CarlosBlanco.pro | Entornos de Desarrollo Integrado (IDE´s) – Introducción. [online]. [Accessed 2 March 2013]. Available from: http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion/

29. Bienvenido a NetBeans y www".Net"beans.org, Portal del IDE Java de Código Abierto. [online]. [Accessed 4 March 2013]. Available from: https://netbeans.org/index_es.html

30. Eclipse. [online]. [Accessed 7 March 2013]. Available from: http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html

31. Modelos de dominio. [online]. [Accessed 17 March 2013]. Available from: http://www.slideshare".Net"/jpbthames/modelos-de-dominio

32. Metodologías Ágiles Desarrollo Software (XP). [online]. [Accessed 17 March 2013]. Available from: http://www.slideshare".Net"/rtorres462003/metologa-agiles-desarrollo-software-xp-1709082

33. Requerimientos funcionales y no funcionales. [online]. [Accessed 20 March 2013]. Available from: http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales

34. Diapositivas xp. [online]. [Accessed 20 March 2013]. Available from: http://www.slisulbaranjosedeshare".Net"//diapositivas-xp

35. Requerimientos Funcionales y No Funcionales (RF/RNF). [online]. [Accessed 21 March 2013]. Available from: http://ingenieriadesoftware.bligoo.com.mx/requerimientos-funcionales-y-no-funcionales-rf-rnf

36. PATRONES DE DISEÑO. [online]. [Accessed 22 April 2013]. Available from: http://es.scribd.com/doc/27239031/PATRONES-DE-DISENO

37. Patrones para asignar responsabilidades. grasp. [online]. [Accessed 24 April 2013]. Available from: http://www.slideshare".Net"/jpbthames/patrones-para-asignar-responsabilidades-grasp

38. Diagrama de Clases. [online]. [Accessed 5 May 2013]. Available from: http://es.scribd.com/doc/35528344/Diagrama-de-Clases

39. A propósito de programación extrema XP (eXtreme Programming) (página 2) - Monografias.com. [online]. [Accessed 7 May 2013]. Available from: http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema2.shtml.

Bibliografía Consultada

1. A propósito de programación extrema XP (eXtreme Programming) (página 2) - Monografias.com. [online]. [Accessed 7 May 2013]. Available from: http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema2.shtml

2. Algoritmos de comparación mediante huellas dactilares | Domotiv@. El blog de la domótica. Miguel Ángel Peñalver. [online]. [Accessed 26 May 2013]. Available from: http://domotiva.wordpress.com/2012/06/26/algoritmos-de-comparacion-mediante-huellas-dactilares/

3. Bienvenido a NetBeans y www".Net"beans.org, Portal del IDE Java de Código Abierto. [online]. [Accessed 4 March 2013]. Available from: https://netbeans.org/index_es.html

4. Biomesys Control de Asistencia V2.0_ET_Es - Biomesys Control de Asistencia V2.0_ET_Es.pdf [online]. [Accessed 26 May 2013]. Available from: http://www.datys.cu/docs/Documentaci%C3%B3n%20Productos/Biomesys%20Control%20de%20Asistencia/Biomesys%20Control%20de%20Asistencia%20V2.0_ET_Es.pdf

5. Biometría - Dactilar. [online]. [Accessed 20 November 2012]. Available from: http://www.biometria.gov.ar/metodos-biometricos/dactilar/

6. Biometrics: fingerprint: algorithms. [online]. [Accessed 15 January 2013]. Available from: http://fingerchip.pagesperso-orange.fr/biometrics/types/fingerprint_algo.htm

7. Capítulo I HERRAMIENTAS CASE. [online]. [Accessed 20 February 2013]. Available from: http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE

8. Características de un sistema Biométrico. [online]. [Accessed 13 January 2013]. Available from: http://redyseguridad.fi-p.unam.mx/proyectos/biometria/basesteoricas/caracteristicassistema.html

9. CarlosBlanco.pro | Entornos de Desarrollo Integrado (IDE´s) – Introducción. [online]. [Accessed 2 March 2013]. Available from: http://carlosblanco.pro/2012/04/entornos-desarrollo-integrado-introduccion/

10. Crystal Methodologies. [online]. [Accessed 9 February 2013]. Available from: http://crystalmethodologies.blogspot.com/

11. DATYS. Datys, Tecnología y Sistemas, empresa desarrolladora de aplicaciones y soluciones de software. [online]. 24 September 2010. [Accessed 12 June 2013]. Available from: http://www.datys.cu/

12. Diagrama de Clases. [online]. [Accessed 5 May 2013]. Available from: http://es.scribd.com/doc/35528344/Diagrama-de-Clases

13. Diapositivas xp. [online]. [Accessed 20 March 2013]. Available from: http://www.slisulbaranjosedeshare".Net"//diapositivas-xp

14. GONZÁLEZ, Migcenel. Diseño de una arquitectura de software orientada a objetos para el sistema de comercialización de RCTV, C.A. utilizando Rational Unified Process [online]. Thesis. 2012. [Accessed 12 June 2013]. Available from: http://miunespace.une.edu.ve/jspui/handle/123456789/984

Submitted by Anabel Armas (anabel.armas@une.edu.ve) on 2012-03-20T15:37:55Z No. of bitstreams: 1 TG3630 resumen.pdf: 10895 bytes, checksum: 88553b99e34d3fc2e35a7226296f8276 (MD5)

15. Diseño de una arquitectura de software orientada a objetos para el sistema de comercialización de RCTV, C.A. utilizando Rational Unified Process [online]. Thesis. 2012. [Accessed 12 June 2013]. Available from: http://miunespace.une.edu.ve/jspui/handle/123456789/984

Submitted by Anabel Armas (anabel.armas@une.edu.ve) on 2012-03-20T15:37:55Z No. of bitstreams: 1 TG3630 resumen.pdf: 10895 bytes, checksum: 88553b99e34d3fc2e35a7226296f8276 (MD5)

16. doi:10.1016/j.patcog.2005.09.005 - fingerprintmatchingby06.pdf [online]. [Accessed 26 May 2013]. Available from: http://www.vislab.ucr.edu/PUBLICATIONS/pubs/Journal%20and%20Conference%20Papers/after10-1-1997/Journals/2006/fingerprintmatchingby06.pdf

17. doi:10.1016/j.patcog.2005.09.005 - fingerprintmatchingby06.pdf [online]. [Accessed 12 June 2013]. Available from:

http://www.vislab.ucr.edu/PUBLICATIONS/pubs/Journal%20and%20Conference%20Papers/after10-1-1997/Journals/2006/fingerprintmatchingby06.pdf

18. Eclipse. [online]. [Accessed 7 March 2013]. Available from: http://curso-sobre.berlios.de/introsobre/2.0.1/sobre.html/eclipse.html

19. Especificaciones Técnicas BIOMESYS AFIS Civil - Biomesys AFIS Civil V2.0 ET_20.04.10_ E03 _Es_D.pdf [online]. [Accessed 26 May 2013]. Available from: 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

20. Fingerprint-POC | Free Science & Engineering software downloads at SourceForge”.Net”. [online]. [Accessed 26 May 2013]. Available from: http://sourceforge”.Net”/projects/fpoc/

21. Herramientas CASE para el proceso de desarrollo de Software (página 2) - Monografias.com. [online]. [Accessed 21 February 2013]. Available from: http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml

22. Historia del lenguaje Java. [online]. [Accessed 27 February 2013]. Available from: http://www.cad.com.mx/historia_del_lenguaje_java.htm

23. Historias de usuario ¿Por qué? ¿Qué son? ¿Cómo son? [online]. [Accessed 7 June 2013]. Available from: http://www.slideshare”.Net”/MiquelMora/historias-de-usuario

24. Huellas. [online]. [Accessed 13 January 2013]. Available from: http://www.slideshare”.Net”/sotolargo/huellas-4519116

25. IEEEpaper.dvi - biometrics04.pdf [online]. [Accessed 26 May 2013]. Available from: http://bytelabs.org/pub/papers/biometrics04.pdf

26. JAVA - ActaUML.PDF [online]. [Accessed 26 May 2013]. Available from: http://www.disca.upv.es/enheror/pdf/ActaUML.PDF

27. Las palabras del silencio: Huellas dactilares... [online]. [Accessed 13 January 2013]. Available from: http://mar-palabrasilencio.blogspot.com/2012/10/huellas-dactilares.html

28. Lenguaje de programación c#. [online]. [Accessed 27 February 2013]. Available from: http://www.slideshare".Net"/perezinho/lenguaje-de-programacion-c-7869638

29. lenguajes_de_programación_1.pdf [online]. [Accessed 7 June 2013]. Available from: http://competenciastic.educ.ar/pdf/lenguajes_de_programacion_1.pdf

30. Manual del lenguaje C#. [online]. [Accessed 7 June 2013]. Available from: http://msdn.microsoft.com/es-es/library/zkxk2fwf(v=vs.90).aspx

31. METODOLOGIA Rational Unified Process (RUP) - RUP vs. XP.pdf [online]. [Accessed 26 May 2013]. Available from: http://www.usmp.edu.pe/publicaciones/boletin/fia/info49/articulos/RUP%20vs.%20XP.pdf

32. Metodologías de desarrollo de Software - EcuRed. [online]. [Accessed 6 February 2013]. Available from: http://www.ecured.cu/index.php/Metodologias_de_desarrollo_de_Software

33. Metodologías de desarrollo de software. ¿Cuál es el camino? (página 2) - Monografias.com. [online]. [Accessed 7 February 2013]. Available from: http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software2.shtml

34. Metodologías tradicionales y metodologías ágiles. [online]. [Accessed 11 February 2013]. Available from: http://www.eumed".Net"/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm

35. Metodología Agiles Desarrollo Software (XP). [online]. [Accessed 17 March 2013]. Available from: http://www.slideshare".Net"/rtorres462003/metologa-agiles-desarrollo-software-xp-1709082

36. MiUneSpace: Diseño de una arquitectura de software orientada a objetos para el sistema de comercialización de RCTV, C.A. utilizando Rational Unified Process. [online]. [Accessed 12 June 2013]. Available from: http://miunespace.une.edu.ve/jspui/handle/123456789/984

37. Modelos de dominio. [online]. [Accessed 17 March 2013]. Available from: http://www.slideshare".Net"/jpbthames/modelos-de-dominio

38. PATRONES DE DISEÑO. [online]. [Accessed 22 April 2013]. Available from: http://es.scribd.com/doc/27239031/PATRONES-DE-DISENO

39. Patrones para asignar responsabilidades. grasp. [online]. [Accessed 24 April 2013]. Available from: http://www.slideshare".Net"/jpbthames/patrones-para-asignar-responsabilidades-grasp

40. Portal DATYS. [online]. [Accessed 20 November 2012]. Available from: http://www.datys.cu/

41. PowerPoint Presentation. [online]. [Accessed 26 May 2013]. Available from: http://efinger.sourceforge".Net"/index_files/v3_document.html

42. Que Es Un Ide De Programacion - Ensayos - Betancourtrv. [online]. [Accessed 7 June 2013]. Available from: http://www.buenastareas.com/ensayos/Que-Es-Un-Ide-De-Programacion/163537.html

43. Requerimientos funcionales y no funcionales. [online]. [Accessed 20 March 2013]. Available from: http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales

44. Requerimientos Funcionales y No Funcionales (RF/RNF). [online]. [Accessed 21 March 2013]. Available from: http://ingenieriadesoftware.bligoo.com.mx/requerimientos-funcionales-y-no-funcionales-rf-rnf

45. SISTEMA BIOMÉTRICO DE RECONOCIMIENTO DE HUELLAS DIGITALES.PDF - 34215_1.pdf [online]. [Accessed 26 May 2013]. Available from: http://repositorio.ute.edu.ec/bitstream/123456789/5634/1/34215_1.pdf

46. SourceAFIS. [online]. [Accessed 20 January 2013]. Available from: http://www.sourceafis.org/blog/

47. TD_04925_11.pdf [online]. [Accessed 12 June 2013]. Available from: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/TD_04925_11/1/TD_04925_11.pdf

48. tema3.pdf [online]. [Accessed 12 June 2013]. Available from: http://wwwdi.ujaen.es/~barranco/publico/ofimatica/tema3.pdf

49. Tesis_Almudena_Lindoso_Munoz.pdf [online]. [Accessed 26 May 2013]. Available from: http://e-archivo.uc3m.es/bitstream/10016/5571/1/Tesis_Almudena_Lindoso_Munoz.pdf

50. Uml.PDF - Uml.PDF [online]. [Accessed 26 May 2013]. Available from: http://www.dccia.ua.es/dccia/inf/asignaturas/GPS/archivos/Uml.PDF

51. VeriFinger fingerprint recognition technology, algorithm and SDK for PC and Web. [online]. [Accessed 20 January 2013]. Available from: http://www.neurotechnology.com/verifinger.html?gclid=CPPIisCbs7cCFUXJtAodkVgArw

52. VeriFinger SDK. [online]. [Accessed 20 January 2013]. Available from: http://www.goit.cl/vfinger.html

53. XP - Extreme Programing Ingenieria de Software. [online]. [Accessed 7 February 2013]. Available from: http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

Glosario de Términos

AFIS: Sistemas Automáticos de Identificación por Huellas Dactilares.

Applet: programa informático desarrollado en Java, que entre sus particularidades tiene la enorme ventaja de ejecutarse en el navegador; forma parte de una página web y es utilizado para introducir acciones dinámicas en ella sin necesidad de enviar una petición del usuario al servidor.

CISED: Centro de Identificación y Seguridad Digital.

IDE: Entornos Integrados de Desarrollo.

Identificación: consiste en la comparación de la muestra recogida del usuario frente a una base de datos de rasgos biométricos registrados previamente. No se precisa de identificación inicial por parte del usuario, es decir, el único dato que se recoge es la muestra biométrica, sin apoyo de un nombre de usuario o cualquier otro tipo de reconocimiento. Este método requiere de un proceso complejo, puesto que se ha de comparar esta muestra con cada una de las previamente almacenadas en búsqueda de una coincidencia.

ISO (Organización Internacional de Normalización): organización de normalización en cuyo funcionamiento intervienen organismos del mundo interesados en regular y armonizar diversas áreas de la industria.

IEC (Comisión Electrotécnica Internacional): enfoca su atención a la existencia de un lenguaje técnico universal, que comprenda definiciones, símbolos eléctricos y electrónicos o unidades de medidas, rangos normalizados, requisitos y métodos de prueba, características de los sistemas como tensión e intensidad y frecuencia, requisitos dimensionales, requisitos de seguridad eléctrica, tolerancias de componentes de equipo eléctrico y electrónico.

ISO/IEC: definen los procedimientos básicos que deben seguirse en la elaboración de normas internacionales y otras publicaciones.

Kit de desarrollo: conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

MINEX de sus siglas en inglés o Prueba de Intercambio de Interoperabilidad de las Minucias: es una evaluación en curso de la plantilla de la huella dactilar INCITS 378. El programa de la prueba tiene dos premisas: proporcionar medidas del funcionamiento e interoperabilidad de las capacidades de codificación y comparación de la plantilla base a los usuarios, a los vendedores y a las partes interesadas, y establecer la conformidad para los codificadores y las unidades de comparación de la plantilla para el Programa de Verificación de Identidad del Personal del gobierno de Estados Unidos.

NIST: Instituto Nacional de Estándares y Tecnología.

Tasa de falsa aceptación: Estadística utilizada para medir el rendimiento biométrico, que calcula el porcentaje de veces que un sistema produce una falsa aceptación, lo cual ocurre cuando una huella dactilar es erróneamente vinculada a otra clasificación que no es la que le corresponde.

Tasa de falso rechazo: Estadística utilizada para medir el rendimiento biométrico, que calcula el porcentaje de veces que el sistema produce un falso rechazo, lo cual ocurre cuando una huella dactilar no es vinculada a su verdadera clasificación.

TIC (Tecnologías de la Información y las Comunicaciones): se refiere al conjunto de herramientas, aplicaciones y soportes informáticos que permiten procesar, almacenar y representar información en las más variadas formas.

UCI: Universidad de las Ciencias Informáticas.

Verificación: el primer paso del proceso es la identificación del usuario mediante algún nombre de usuario, tarjeta o algún otro método. De este modo se selecciona de la base de datos el patrón que anteriormente se ha registrado para dicho usuario. Posteriormente, el sistema recoge la característica biométrica y la compara con la que tiene almacenada. Es un proceso simple, al tener que comparar únicamente dos muestras, en el que el resultado es positivo o negativo.

Anexo 1. Clasificación de las huellas dactilares.

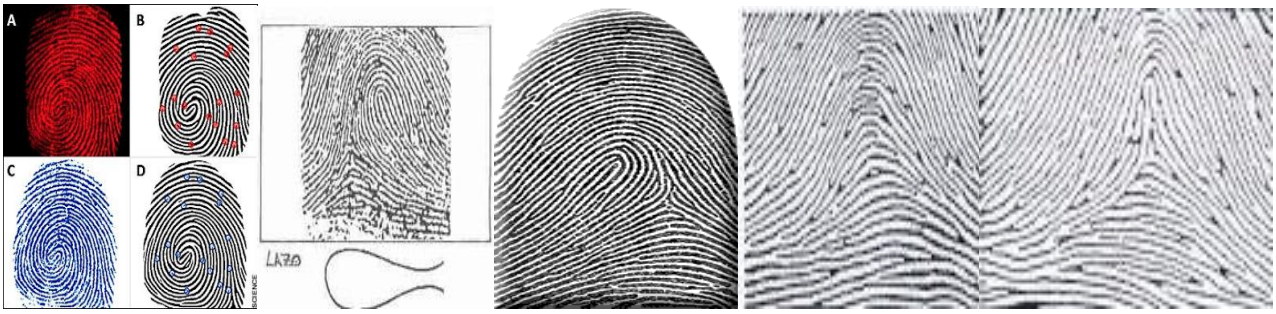

Figura 15: Clasificación de las huellas: espiral, lazo derecho, lazo izquierdo, arco y arco tendido.

Anexo 2. Historias de usuario

HISTORIA DE USUARIO	
Número: 1	**Usuario:** Sistema
Nombre historia: Obtener plantillas de minucias a utilizar.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 6	**Iteración asignada:** 1
Programador responsable: Asiel Echemendia Carrasco	
Descripción: Con el objetivo de la comparación de la huella dactilar es necesario realizar los siguientes escenarios que dan paso al cumplimiento de esta funcionalidad: **Escenario Obtener Plantilla:** Obtener la plantilla consiste en cargar la misma aplicarle el proceso de deserializar o descodificarla teniendo en cuenta que esta plantilla para almacenarla se debe serializar o codificar con el objetivo de que los datos que se almacenan en ella no puedan verse sin haberlas	

deserializado.
Observaciones: Para realizar este proceso se debe disponer de dos plantillas biométricas o plantillas de minucias resultante del proceso de extracción de características.
Prototipo de interfaces: Sin prototipo de interfaz.

Tabla 9: HU_Obtener plantillas de minucias a utilizar.

HISTORIA DE USUARIO	
Número: 1	**Usuario:** Sistema
Nombre historia: Mostrar resultados de la comparación.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 4	**Iteración asignada:** 4
Programador responsable: Asiel Echemendia Carrasco	
Descripción: Con el objetivo de la comparación de la huella dactilar es necesario realizar los siguientes escenarios que dan paso al cumplimiento de esta funcionalidad: **Escenario Mostrar resultado de la comparación:** Este escenario consiste en mostrar un resultado después de haberse ejecutado el proceso de comparación, donde se dará como resultado verdadero o falso.	

Observaciones: Para realizar este proceso se debe disponer de la plantilla resultante del proceso de extracción.
Prototipo de interfaces: Sin prototipo de interfaz.

Tabla 10: HU_Mostrar resultados de la comparación.

Anexo 3. Diagrama de paquetes.

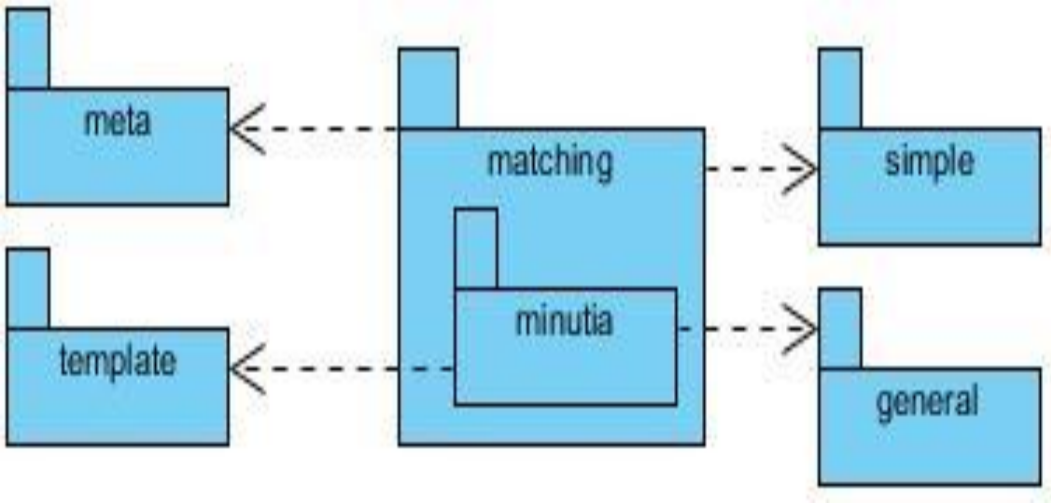

Figura 16: Diagrama de paquetes.

Anexo 4. Tarjetas CRC

MatchScoring	
Esta clase tiene la responsabilidad del calcular el valor (score), que es el valor de similitud entre un par de minucias de las plantillas cargadas.	➢ MatchAnalysis ➢ Parameter

Tabla 11: Tarjeta CRC. Clase MatchScoring.

EdgeLookup	
Esta clase tiene la responsabilidad de encontrar los correspondientes pares de minucias entre las plantillas cargadas.	➢ Angle ➢ Calc ➢ Parameter ➢ LookupResult ➢ Range ➢ NeighborEdge ➢ MinutiaPair

Tabla 12: Tarjeta CRC. Clase EdgeLookup.

MatchAnalysis	
Esta clase tiene la responsabilidad de calcular la suma de la distancia de error y la suma del ángulo de error que existe entre los pares de minucias encontradas en la clase EdgeLookup.	➢ Parameter ➢ NestedAttribute ➢ EdgeConstructor ➢ MinutiaPairing ➢ EdgeLookup ➢ Template ➢ PairInfo

Tabla 13: Tarjeta CRC. Clase MatchAnalysis.

EdgeTable	
Esta clase tiene la responsabilidad de crear la tabla de minucias vecinas.	➢ Parameter ➢ NestedAttribute ➢ NeighborEdge ➢ Template ➢ Calc ➢ EdgeConstructor

Tabla 14: Tarjeta CRC. Clase EdgeTable.

Anexo 5. Casos de pruebas

Caso de prueba de aceptación	
Código de caso de prueba: 1	**Nombre de la historia de usuario:** HU_ Obtener plantillas de minucias a utilizar.
Responsable de la prueba: Asiel Echemendia Carrasco.	
Descripción de la prueba: prueba para comprobar que el componente cargue correctamente las plantillas.	
Condición de ejecución: para la realización del proceso se debe cargar dos plantillas de minucias para deserealizarla.	
Entrada/Pasos de ejecución: luego de deserealizar las plantillas se verifica que todos los valores que se deben obtener de las plantillas se obtuvieron sin problema.	
Resultado esperado: se debe obtener como resultado que todos los parámetros que se esperan o que se	

obtienen de las plantillas estén correcto.
Evaluación de la prueba: prueba satisfactoria

Tabla 15: Caso de prueba de aceptación. HU_Obtener plantillas.

Anexo 6. Interfaz de prueba del componente.

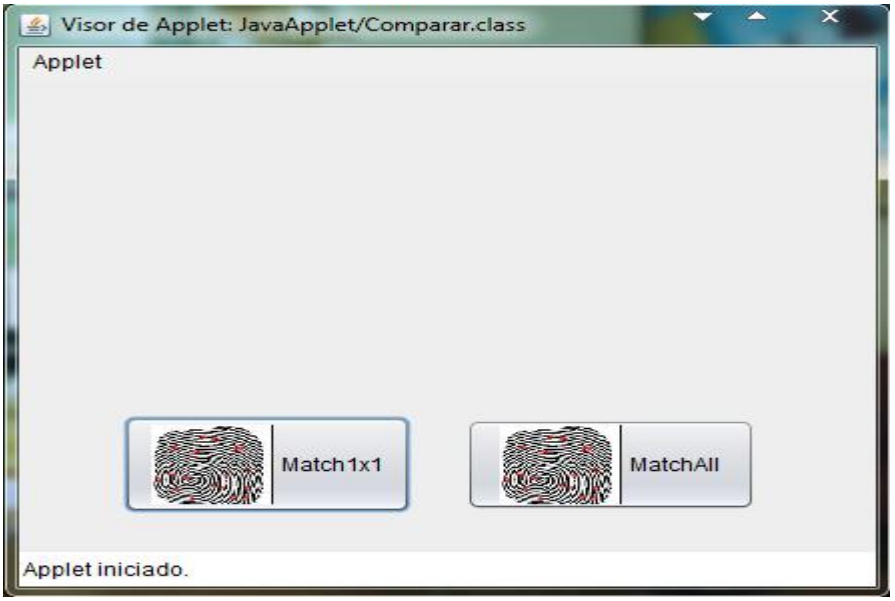

Figura 17: Vista inicial del componente.

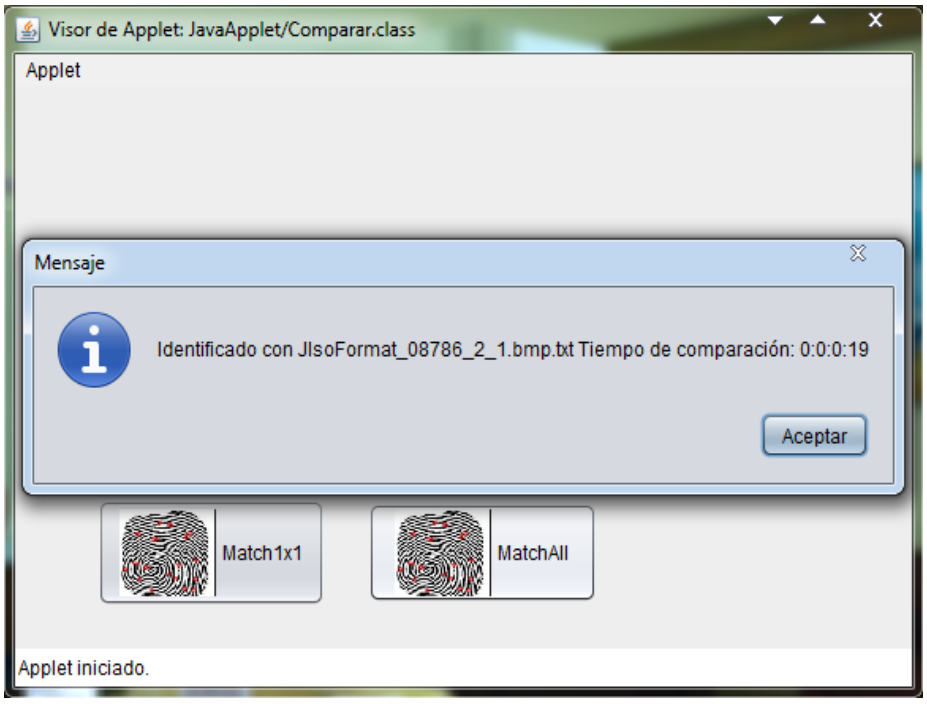

Figura 18: Vista del resultado de ejecutar la comparación una plantilla contra muchas.