**Universidad de las Ciencias Informáticas**


# Título: Componente para la extracción de características en huellas dactilares

Trabajo de Diploma para optar por el título de

"INGENIERO EN CIENCIAS INFORMÁTICAS"

**Autores:** Edelsys Vivian Carrazana Paneque

Leandro Fortún Luna

**Tutor:** Ing. Ramón Santana Fernández

**Co-tutora:** Ing. Yulainne Alonso Hernández

**Consultante:** Ing. Yosbel Rodríguez Rodríguez

**La Habana, día 10 de junio de 2013.**

*"Nunca podría haber hecho lo que he hecho, sin los hábitos de puntualidad, orden y diligencia, sin la determinación de concentrar en mi un objetivo a la vez."*

*Charles Dickens*

## Declaración de autoría

Declaramos ser los autores legítimos del trabajo titulado **"***Componente para la extracción de características en huellas dactilares",* y delegamos 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.

**Autores:**

Edelsys Vivian Carrazana Paneque

Leandro Fortún Luna

**Tutor:**

Ing. Ramón Santana Fernández

**Co-tutora:**

Ing. Yulainne Alonso Hernández

## Agradecimientos

**De Edelsys Vivian:**

A mis padres por su infinito amor, abnegación y sabiduría, por la lección de vida, comprensión y apoyo incondicional.

A **Edelia** mi mamá, por ser mi amiga, mi confidente y haberme admitido en su cuerpo durante nueve meses, por compartir conmigo su espacio, su aire, su mundo y haber soportado valientemente, todo el dolor que en su momento implicó darme la vida.

A **Edelsio** mi papá, por mantenerse a mi lado aún cuando yo no deseaba que estuviera ahí, por todas las veces que calló, para que yo aprendiera con su silencio; por enseñarme a dar, a compartir, sin importar si la otra persona merecía recibir o no, y por nunca dejar que me alejara del buen camino.

A mis hermanas, por ser especiales y el mayor regalo de mi vida, por conquistar mi cariño con sólo una sonrisa y haber depositado toda su confianza en mí, por brindarme su apoyo y hacerme sentir orgullosa.

A mi hermana **Edelmis** por darme la mayor de las alegrías al tener un hermoso angelito (mi sobrinita **Nazli**).

A mi hermana **Edeleimis** por ser mi abogada defensora.

A **Wilbert** mi novio, por iluminar mis días con su sonrisa y apoyarme en mis decisiones, entender mis enfados y demostrarme que siempre se puede ser mejor persona, por no reprocharme nada y ser especial conmigo.

A mi abuelo **Pablo** por su humildad y amor incondicional.

Al primo **Rafe** por las tardes de tertulias y por ser mi consejero.

A mi compañero de tesis **Leo** por su abnegación, responsabilidad y compromiso.

A mis amistades por la confianza depositada y la convivencia en estos cinco años.

A los profesores que contribuyeron a mi formación profesional.

**De Leandro Fortún Luna:**

Le agradezco a la revolución y a Fidel por crear una universidad como la UCI.

A mi compañera de tesis por su esfuerzo, abnegación y por ser parte de este sueño.

A mi tutor por estar ahí cuando más lo necesitaba.

A mi primo Yosbel por ser parte fundamental en el desarrollo de la aplicación y por apoyarme en todos estos años de estudio.

A todos los profesores que forman parte de mi educación como profesional y en especial a: Matilde, Joel Arencibia y Sergio Reyes.

A mis compañeros de grupo que desde el primer día de clases estuvieron ahí para ayudarme sin pedir nada a cambio: Eliandis, Amed, Luis Miguel, Rafael Portilla, Olber, Rafal Quiles, Vladimir, Junior, Carlos, Fernando, Reicel, Lianne, Ernesto y todos los demás con los que compartí estos 5 años en la misma aula.

A mis compañeros de proyecto: Adrian, Roily, Melvis y todos los demás con los que compartí en el Laboratorio 104 del docente 6 aunque sea una línea de código.

A mis amigos y hermanos Alejandro Rodríguez, Noel y Alejandro Loyola por estar ahí en los buenos y malos momentos sin importar las consecuencias.

A Tony y Leticia por ser como mis propios padres y por estar ahí cuando más los necesité, gracias por ser la calidad de persona que son.

A Carlos y Yansi esos locos de los que aprendí que en la calle hace falta tener los ojos bien abiertos y la maldad siempre lista para usarla.

A mis hermanos, primos y tíos por siempre confiar en mí.

A mi Novia Yuly, por la dedicación, comprensión y todo ese amor que únicamente tu me sabes dar. Gracias por darme la oportunidad de estar a tu lado. Te amo.

A Manuel, gracias por permitirme ser como un hijo para ti, por enseñarme a enfrentar problemas y buscarle la mejor solución y por ayudar a mi mamá para que este sueño se hiciera realidad, sin ti muchas cosas no hubieran sido posibles.

A mis padres por permitirme estar hoy aquí y por haber depositado toda su confianza en mí. Es por

ustedes que hice tanto esfuerzo durante estos 5 largos y muy lindos años de universidad. Ustedes son mi razón de vivir gracias por permitirme ser su hijo.

Gracias a todos ustedes por compartir este día conmigo.

# Dedicatoria

**De Edelsys Vivian**:

A mis padres y hermanas por ser quienes merecen la realización de un sueño.

A mi mamá por perdonar con un simple beso.

A mi papá por ser mi límite antes los excesos.

A mis hermanas por ser mi inspiración.

A Nazlita por la esperanza de vida.

A mi novio por ser parte de lo que más amo en la vida.

**De Leandro Fortún Luna:**

Mi mamita linda, por ser mi razón de ser, el motor que bombea la sangre por mis venas y el ejemplo de novia, amiga, hermana y madre que eres. Además por apoyar todas mis decisiones por incorrectas que fueran, apostando siempre a tu niño.

Mi papito, por ser la figura importante de la que me siento orgulloso de hablar, la meta que siempre quise alcanzar y el ejemplo de esfuerzo, abnegación y voluntad por la que me levanto todos los días a cumplir con mi deber.

Mami y papi; sé que hoy no solo se cumple uno de mis sueños si no uno de los de ustedes también, gracias por ser un ejemplo para mí y por darme la oportunidad de ser su hijo. Los amo con todo lo que puede amar un hombre, con la vida.

A mis abuelitos por ayudarme a crecer y hacerme el hombre de bien que hoy soy.

A mi novia por ser la mujer que amo y por estar a mi lado en los momentos más difíciles de mi carrera y hacer de ellos un rato agradable. Gracias por apoyarme y ayudarme hacer el profesional que hoy soy.

A tía Caruca y a tío Robertico por ayudar a mi mamá durante mis primeros años de vida.

A mi primo Juan Carlos y Yosbel por quererme como a un hermano.

A mis compañeros tanto de la niñez como los de la juventud, por dejarme ser parte de sus vidas.

## Resumen

En la actualidad el reconocimiento de personas basado en rasgos característicos ha demostrado ser un método eficaz en recintos donde la seguridad es un tema de interés. Este proceso se efectúa con ayuda de los sistemas biométricos de identificación, que basan su funcionamiento en el reconocimiento de las características fisiológicas o conductuales de los seres humanos. En el departamento de Biometría, perteneciente al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas, se lleva a cabo el proceso de desarrollo de un sistema que realice la verificación de individuos mediante huellas dactilares. Para que este cumpla su objetivo, es necesario realizar del lado del cliente el proceso de extracción de características en huellas dactilares. La presente investigación tiene como objetivo, desarrollar un componente para la extracción de características en huellas dactilares, que genere una plantilla biométrica en formato ISO/IEC 19794-2:2011, y pueda ser ejecutado en un entorno multiplataforma. El desarrollo del componente para la extracción posee un aporte práctico, pues se contará con la posibilidad de realizar el proceso de extracción de características para el sistema de verificación dactilar utilizando tecnologías libres. Además, su ejecución no estaría condicionada por el tipo de navegador utilizado por el cliente, ni a un sistema operativo determinado, contribuyendo así a la soberanía tecnológica del país. La principal finalidad del componente es ser integrado en un Applet, pero puede ser publicado en un servidor o en cualquier sistema que requiera del servicio que brinda.

**Palabras clave:** extracción, huellas dactilares, minucias, reconocimiento, sistemas biométricos.

# Índice

# Índice de figuras

# Índice de tablas

# Introducción

El Procesamiento Digital de Imágenes (*PDI*) es el conjunto de técnicas y procesos para descubrir o hacer resaltar información contenida en una imagen, usando como herramienta principal una computadora y su objetivo es visualizar o evaluar estadísticamente algunos aspectos de la imagen que no se encuentran en su forma original. En las últimas dos décadas se ha fortalecido en áreas fundamentales como: la medicina, robótica, electrónica y la biometría, donde las Tecnologías de la Información y las Comunicaciones (*TIC*) tienen un papel importante a partir de los sistemas de identificación biométricos.

Estos sistemas surgen dada la necesidad de reemplazar y fortalecer los métodos tradicionales de identificación (*basados en la clave personal*), y fundan su funcionamiento en el reconocimiento de las características fisiológicas o conductuales de los seres humanos. Para este reconocimiento uno de los métodos más populares es el basado en huellas dactilares, debido a que estas cumplen con las tres leyes fundamentales de la Dactiloscopia: perennidad, inmutabilidad y diversidad infinita. [1]

A finales del siglo XIX, el Sir Francis Galton publicó el libro "FingerPrints", donde realiza un estudio detallado de las huellas dactilares y enuncia un método que clasifica e identifica las características (*también denominadas "minucias"*) por las que las huellas dactilares pueden ser clasificadas hoy en día. Su hijo, quien continuó la investigación, estableció a partir del cálculo de probabilidad que dos huellas serían iguales en 1:64.000.000.000. Estos aportes son la base para la ciencia de identificación de personas a partir de las huellas dactilares. [2]

En el proceso de identificación dactiloscópica, la huella es procesada en diferentes etapas, las que posibilitan su tratamiento con el objetivo de obtener una muestra mejorada con alta calidad. Este proceso comienza con la adquisición de la imagen, luego se procesa la muestra obtenida y el resultado es almacenado en una plantilla biométrica para su uso en la etapa de comparación. Esta fase consiste en encontrar el grado de similitud entre dos vectores de características, donde sus componentes representan las minucias de cada huella dactilar.

En el mundo existen empresas que se dedican a la producción de software biométrico entre las que se encuentran Griaule FingerPrint Recognition, Innovatrics y Safran Morpho. En esta esfera de producción Cuba no se encuentra exenta, hecho que lo demuestran la empresa de Desarrollo de Aplicaciones, Tecnologías y Sistemas (*DATYS*), el Centro de Aplicaciones de Tecnologías de Avanzada (*CENATAV*) y la Universidad de las Ciencias Informáticas (*UCI*), las cuales han impulsado el desarrollo tecnológico con la producción de sistemas basados en el proceso de identificación

dactiloscópica. DATYS incluye entre sus productos el Sistema Automatizado de Identificación Dactiloscópica (*Biomesys AFIS*), que permite la identificación y autenticación de personas de forma rápida y segura. También cuenta con el sistema BIOMESYS Control de Asistencia, que aprovecha 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. El 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.

La UCI constituye una universidad productiva en la cual se han desarrollado diferentes componentes, los 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, el Sistema Único de Identificación Nacional (*SUIN*) y el Servicio Autónomo de Identificación, Migración y Extranjería (*SAIME*) desarrollado en colaboración con el gobierno de la República Bolivariana de Venezuela, nación para la cual se implementó. Cabe destacar además, la implementación de un algoritmo de extracción de minucias para un sistema de verificación de personas.

La actividad desarrollo–producción en la universidad es soportada sobre la red de centros, entre los que se destaca el Centro de Identificación y Seguridad Digital (*CISED*), donde 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 del SourceAFIS para la extracción y comparación de características. El SourceAFIS está desarrollado en ".Net" y posee una versión alfa no funcional en Java que no cuenta con el módulo de extracción. A partir de la implementación existente se puede desarrollar un ActiveX para la extracción de características, pero el uso de tecnologías propietarias continuaría presente y estaría sujeto de forma obligatoria al uso del Internet Explorer como navegador, lo que implica que esta solución sólo pueda ser usada en el sistema operativo Windows.

Una vez analizada la problemática anterior se formula el siguiente **problema científico**: ¿Cómo realizar el proceso de extracción de características en huellas dactilares en un entorno multiplataforma?

Para solucionar el problema propuesto se define como **objeto de estudio** los procesos de extracción de características en huellas dactilares.

Como **objetivo general** se plantea desarrollar un componente para la extracción de características en huellas dactilares, que genere una plantilla biométrica en formato ISO/IEC 19794-2:2011, y pueda ser ejecutado en un entorno multiplataforma, para integrarlo en un sistema de verificación dactilar, que da lugar a los **objetivos específicos** siguientes:

- Caracterizar diferentes sistemas que realicen el proceso de extracción de características en huellas dactilares.
- Analizar las tecnologías, metodologías y herramientas existentes que contribuyan al desarrollo del componente.
- Desarrollar el proceso de extracción de características en huellas dactilares.
- Realizar pruebas al componente.

Las **preguntas científicas** que guían esta investigación son las siguientes:

- ¿Cuáles serían las acciones a realizar en el pre-procesamiento para obtener una imagen con calidad?
- ¿Por qué la calidad de la imagen influye en el proceso de extracción de características?
- ¿Qué condiciones se deben tener en cuenta para realizar un buen post-procesamiento de la imagen?

Para dar cumplimiento a los objetivos planteados se proponen las siguientes **tareas de investigación:**

- Análisis del estado del arte en Cuba y en el mundo, referente a la extracción de características en huellas dactilares.
- Análisis del algoritmo de extracción de características de la biblioteca de clases SourceAFIS.
- Definición de la metodología, tecnologías y herramientas a utilizar para el desarrollo del componente.
- Identificación de los requisitos funcionales y no funcionales.
- Definición de la arquitectura del componente.
- Definición de los patrones del diseño.
- Diseño de la interfaz de prueba del componente.
- Implementación de los algoritmos de segmentación y normalización de la huella dactilar.
- Implementación del algoritmo para la estimación del campo de orientación de la huella dactilar.
- Implementación de los algoritmos de binarización y adelgazamiento de la huella dactilar.
- Implementación del algoritmo de extracción de características en huellas dactilares.
- Implementación de los algoritmos de post-procesamiento de la huella dactilar.

- Implementación del formato ISO/IEC 19794-2: 2011.
- Ejecución de pruebas sobre el componente para verificar su correcto funcionamiento.

**Justificación de la investigación:**

El desarrollo del componente para la extracción posee un aporte práctico, pues se contará con la posibilidad de realizar el proceso de extracción de características para el sistema de verificación dactilar utilizando tecnologías libres, además, su ejecución no estaría condicionado por el tipo de navegador utilizado por el cliente, ni a un sistema operativo determinado, contribuyendo así a la soberanía tecnológica del país. La principal finalidad del componente es ser integrado a un Applet, pero puede ser publicado en un servidor o en cualquier sistema que requiera del servicio que brinda.

Para el desarrollo de la investigación se utilizan varios **métodos científicos**, tanto teóricos como empíricos, que se ajustan al objeto de estudio y al cumplimiento de los objetivos trazados. Entre los que se encuentran:

**Métodos teóricos:**

- **Analítico-Sintético:** posibilita la consulta de diversas fuentes bibliográficas, que facilitará la confección del estado del arte de la investigación, haciendo énfasis en el proceso de extracción de características en imágenes de huellas dactilares, sintetizando con los elementos más importantes; lo cual viabiliza el establecimiento de una estrecha relación con el objeto de estudio.
- **Análisis Histórico-Lógico:** la utilización de este método permite la comprensión lógica del objeto de estudio, haciendo un análisis riguroso del proceso evolutivo por el que ha transitado la extracción de características basada en minucias, además de contribuir al cumplimiento de las tareas de la investigación.

**Métodos empíricos:**

- **Entrevista:** se realiza a personal capacitado con el propósito de obtener información, ideas, puntos de vista que contribuyan al desarrollo de la investigación y aporten conocimientos específicos del tema.
- **Experimental:** posibilita comprobar las funcionalidades en la medida que estas sean implementadas, con el objetivo de validar si cumplen o no con los requerimientos establecidos.

El presente trabajo de diploma consta de tres capítulos estructurados de la siguiente manera:

**Capítulo 1: Fundamentación teórica:** constituido por el estado del arte de sistemas que realizan el proceso de extracción de características en imágenes de huellas dactilares; además, se abordan

elementos teóricos de la investigación tales como: sistemas biométricos, proceso de identificación de individuos y conceptos relacionados con las huellas dactilares. En este capítulo se realiza la selección de la metodología, tecnologías y herramientas de desarrollo que serán utilizadas; así como la propuesta de solución.

**Capítulo 2: Características del componente:** en este capítulo se efectúa el levantamiento de los requisitos, que describirán en detalle las propiedades y condiciones que debe cumplir el componente. Se determina la arquitectura y se definen, describen y priorizan las historias de usuario, se hace referencia a los patrones utilizados para su diseño e implementación.

**Capítulo 3: Implementación y prueba:** se describen los artefactos relacionados con la implementación y las pruebas realizadas al componente, para validar el correcto funcionamiento de los requerimientos especificados.

# Fundamentación teórica

En el presente capítulo se define un conjunto de elementos que sitúan al lector en el contexto donde se desarrolla la investigación. Se describen sistemas que realizan el proceso de extracción de características en imágenes de huellas dactilares y las tecnologías que se consideran indicadas para implementar la solución.

## 1.1 Conceptos asociados al dominio del problema

**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. [3]

**ActiveX según Microsoft:** es un estándar que permite a los componentes de software interactuar en un entorno de red, independientemente del lenguaje en el que fue creado. [4] Al igual que los Applets, un ActiveX "Control" se puede incrustar en una página web u otro programa para que pueda reutilizar una funcionalidad empaquetada, programada por otra persona. A diferencia de Java, que es un lenguaje de programación independiente de la plataforma, los controles ActiveX son distribuidos como binarios ejecutables, y deben ser compilados para cada equipo destino. [5]

**ISO/IEC 19794-2:2011:** especifica formatos de conceptos y datos para la representación de las huellas dactilares utilizando el concepto fundamental de minucias. [6]

**Ruido en las imágenes:** es información no deseada provocada por fuentes ruidosas que contaminan la imagen, como los sensores ópticos y eléctricos. [7]

**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. [8]

## 1.2 Sistemas biométricos

Son sistemas automatizados que permiten verificar e identificar la identidad de las personas, basándose en algún rasgo físico o patrón de comportamiento; y se clasifican en: [1]

- **Fisiológicos**


**Figura 1: Clasificación de las características fisiológicas de los seres humanos**

- **Conductuales**

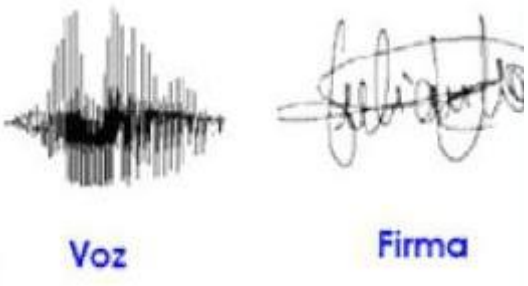


**Figura 2: Clasificación de las características conductuales de los seres humanos**

## Procesos principales en la identificación de individuos

Cada sistema biométrico utiliza una clase de interfaz, un censor o mecanismo de captura determinado y un software específico, que facilitan el proceso de identificación de individuos; y los pasos básicos son: [9]

- **Registro:** proceso de captura de una o más muestras biométricas que se utilizan como identidad biométrica del ciudadano para posteriores procesos de identificación.
- **Procesado:** fase orientada a la extracción de características que identifican al individuo.
- **Almacenamiento de la plantilla biométrica:** proceso que permite almacenar la plantilla biométrica en una base de datos con información biométrica de los individuos.
- **Verificación:** es la comparación de la plantilla biométrica almacenada con la que se acaba de capturar.
- **Identificación:** tarea en la cual el sistema biométrico busca en una base de datos una referencia que coincida con la muestra biométrica suministrada y, de encontrarla, devuelve la identidad correspondiente.

## 1.3 Huellas dactilares

Las huellas dactilares son patrones constituidos por las crestas papilares de los dedos de las manos que

aparecen visibles en la epidermis, generando configuraciones diversas que se forman en el período fetal, a partir del sexto mes, manteniéndose invariables a través de la vida del individuo, a menos que sufran alteraciones debido a accidentes tales como cortes o quemaduras. [10]

## Características de la huella dactilar

La particularidad de las huellas dactilares es que son únicas para cada individuo, argumento que permite su uso como uno de los métodos de reconocimiento más utilizados.

- **Valles y crestas**

Las huellas dactilares están constituidas por rugosidades que forman salientes y depresiones. Las salientes se denominan crestas papilares y las depresiones surcos inter-papilares o valles. [11]

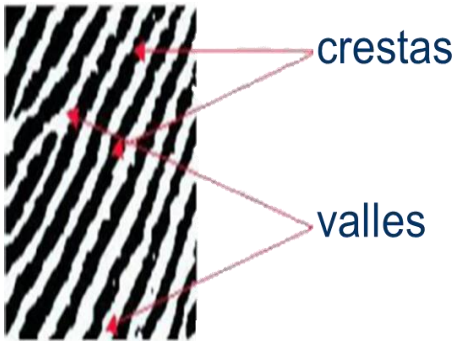


**Figura 3: Representación de crestas y valles**

- **Minucias**

Son los puntos singulares encontrados en el trazo de las crestas y en el proceso de identificación las que tienen mayor valor significativo son las terminaciones y las bifurcaciones. Cabe destacar que en una huella dactilar existen aproximadamente de cincuenta a ciento cincuenta minucias.

- **Núcleo y delta**

El núcleo puede definirse como el punto de máxima curvatura de la cresta más interna, mientras que el delta, es el centro de un conjunto de crestas inferiores al núcleo, que dibujan varios triángulos concéntricos.

## Clasificación de las huellas dactilares

Las huellas dactilares son únicas en cada individuo y se pueden clasificar en cinco tipos: arco, arco tendido, lazo derecho, lazo izquierdo y espiral: Ver **Anexo_Clasificación de las huellas dactilares**.

**Arco:** los patrones de arco tienen líneas que empiezan en un lado de la huella dactilar, van hacia el centro curvándose hacia arriba y salen al otro lado de la huella dactilar, formando lo que se asemeja a una fila de

arcos apilados y no cuentan con un punto delta.

**Arco tendido:** es un tipo de línea que semeja a un arco, que nace y se pierde rápidamente en un paso de ángulo.

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

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

**Espiral:** contiene al menos una cresta que hace una trayectoria completa de trescientos sesenta grados alrededor del centro de la huella dactilar, y se pueden encontrar más de un núcleo y un delta. Aproximadamente del veinticinco al treinta y cinco por ciento de las huellas dactilares son espirales. [12]

## 1.4 Proceso de extracción de características en huellas dactilares

### Etapa de pre-procesamiento. Métodos

El pre-procesamiento de la huella dactilar es la clave del éxito de una buena extracción de características. Esta etapa aplica un conjunto de técnicas sobre la imagen de la huella dactilar con el objetivo de mejorar su calidad, para facilitar la búsqueda de información de interés en una etapa posterior. La calidad de la imagen se corresponde con la claridad de las estructuras de crestas en la imagen de la huella dactilar; además, se considera que posee buena calidad cuando el contraste es alto y están bien definidas las crestas y los valles. [13]

### Método tradicional

Conocido como el método que aplica secuencialmente sobre la imagen los siguientes procesos: segmentación, normalización, cálculo de la orientación, binarización y adelgazamiento.

### Segmentación

Existen dos regiones que describen cualquier imagen de la huella dactilar, la región que se encuentra en el primer plano y la región del fondo. La región del primer plano se corresponde con la zona de la huella dactilar que contiene las crestas y los valles, área también conocida como la zona de interés, que contiene los puntos característicos (*minucias*) de la imagen de la huella dactilar. Por su parte, la región del fondo es aquella región fuera de los límites de las huellas dactilares que no contienen ninguna información válida y es el área donde principalmente se introduce ruido en la imagen durante su adquisición. La esencia de la

segmentación radica en reducir la carga asociada, asegurándose de que solamente se enfoquen en la región del primer plano.

## Normalización

Es usada para estandarizar las variaciones de los valores de niveles de gris que constituyen la imagen de la huella dactilar, haciéndolos corresponder dentro de cierto rango, considerado lo suficientemente bueno para mejorar el contraste de la imagen. [14] Como parte de este proceso, se ecualiza el histograma previamente calculado.

El histograma de una imagen se puede definir como una función discreta que representa el número de píxeles en la imagen en función de los niveles de intensidad. En general se representa como un gráfico de barras, donde las abscisas son los distintos niveles de gris de la imagen y las ordenadas la cantidad de veces que aparece cada nivel en la imagen. Proporciona información sobre el contraste, y puede ser utilizado para ajustar este parámetro, eliminando ciertas tonalidades molestas. [15]

Con el objetivo de reducir el contraste en las áreas muy claras o muy oscuras de una imagen se utiliza la ecualización del histograma, que consiste en una transformación no lineal que considera la distribución acumulativa de la imagen original, para generar una imagen resultante cuyo histograma será aproximadamente uniforme. [15]

## Cálculo de la orientación

En cada huella dactilar, las crestas forman patrones que fluyen en diferentes direcciones. La dirección del flujo de una en particular durante un intervalo de píxeles, como se muestra en la figura constituye la orientación de la cresta. [14]

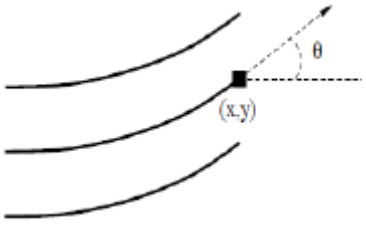


**Figura 4: Orientación de un píxel de cresta en una huella dactilar**

## Binarización

Es el proceso que convierte una imagen en escalas de grises a una imagen binaria, donde cada píxel toma valor de 0 o 1. La mayoría de los algoritmos de extracción de minucias operan en imágenes binarias donde hay sólo dos niveles de interés: los píxeles negros que representan las crestas y los blancos que

representan los valles. Esto mejora el contraste entre las crestas y los valles en la imagen de la huella dactilar, y por consiguiente facilita la extracción de minucias. [16]

## Adelgazamiento

Se aplica sobre la imagen binarizada obtenida de la etapa anterior y como resultado todas las crestas tienen el grosor de un píxel, haciendo que los puntos característicos de la huella dactilar se puedan identificar con más facilidad. Este proceso se implementa normalmente a través de operaciones morfológicas, que permiten reducir el ancho de las crestas a un solo píxel. Después de esta etapa, la imagen está lista para la extracción de características. [17]

## Método basado en la transformada rápida de Fourier

La mejora de la imagen basada en el análisis de la Transformada Rápida de Fourier (*FFT por sus siglas en inglés Fast Fourier Transform*), divide la imagen en pequeños bloques de procesamiento con un tamaño de treinta y dos por treinta y dos píxeles. Con el fin de mejorar un bloque en específico por sus frecuencias dominantes, se multiplica la magnitud de la trasformada de Fourier del bloque por la magnitud de un conjunto de tiempos que permiten mejorarlos. Para ello se toma un valor umbral “k” y se considera que a mayor valor de “k” se puede mejorar la apariencia de las crestas, pero un “k” demasiado alto puede dar lugar a resultados falsos en la unión de las crestas, provocando una terminación para convertirse en una bifurcación o viceversa. [16]

## Método basado en la transformada en tiempo corto de Fourier

Debido a las propiedades de regularidad y continuidad de la imagen de la huella dactilar, la región ocluida y dañada puede ser recuperada utilizando la información contextual de la vecindad circundante. La eficiencia de un algoritmo de mejora automatizado depende de la medida en que se utilizará la información contextual. Uno de estos algoritmos utilizados para el proceso de mejora de la imagen de la huella dactilar es el análisis mediante la transformada en tiempo corto de Fourier (*STFT por sus siglas en inglés Short Time Fourier Transform*).

Durante el análisis STFT la imagen se divide en pequeños bloques solapados, que son estacionarios y pueden ser modelados aproximadamente como una onda de superficie. El espectro de Fourier de esta pequeña región se analiza y se obtienen estimaciones probabilísticas de la frecuencia y orientación de la cresta. Este análisis también da lugar a un mapa de energía que puede ser utilizado como una máscara de región para distinguir entre la huella dactilar y la región del fondo. Por lo tanto, los resultados del análisis STFT son el cálculo simultáneo de la orientación y la frecuencia de las crestas, además de la máscara de la región de interés. [17]

## Etapa de extracción

Por lo general el método más empleado para la extracción de minucias es el basado en el concepto de número de cruce (*CN por sus siglas en inglés*). Este método detecta las minucias mediante la localización de los puntos finales y los puntos de bifurcación en el esqueleto de cresta adelgazado, basado en el número de píxeles vecinos. La extracción se realiza mediante el recorrido de la vecindad local de cada píxel de la cresta en la imagen utilizando bloques de tres por tres. El valor del CN se calcula como la mitad de la suma de las diferencias entre pares de píxeles adyacentes dentro de los ocho vecinos, para esto se usan las propiedades del CN, como se muestra en la Tabla 2. [18]

El número de cruce para un píxel P es:

| | | |
|---|---|---|
| P4 | P3 | P2 |
| P5 | **P** | P1 |
| P6 | P7 | P8 |

**Tabla 1: Representación de la vecindad de ocho respecto al píxel central**

| CN | Propiedades |
|---|---|
| 0 | Propiedad aislada del punto |
| 1 | Punto de terminación |
| 2 | Punto de conexión |
| 3 | Punto de bifurcación |
| 4 | Punto de cruce |

**Tabla 2: Propiedades del número de cruce o crossing number**

## Etapa de post-procesamiento

Al realizar el proceso de extracción de características pueden ser introducidas falsas minucias en la imagen de la huella dactilar, pues generalmente las imágenes contienen ruido y defectos creados por el proceso de captura; por lo tanto, después de que los puntos característicos se extraen, es necesario ejecutar una etapa de post-procesamiento con el fin de validar las minucias. [19]

Para eliminar las falsas minucias se tiene en cuenta el conjunto de condiciones que se describe a continuación:

- Si la distancia entre una terminación y una bifurcación es menor que D y las dos minucias son la misma cresta, entonces se eliminan ambas. D se determina como la media entre el ancho de la cresta que representa el promedio de la distancia entre dos crestas paralelas y que son vecinas.
- Si la distancia entre dos bifurcaciones es menor que D y ellas están en la misma cresta, se eliminan ambas.
- Si dos terminaciones están a una distancia D y sus direcciones son coincidentes con una variación pequeña del ángulo, y cumplen la condición de que ninguna otra terminación se encuentra entre las dos terminaciones, entonces son consideradas como minucias falsas derivadas de una cresta rota, por ende son eliminadas.
- Si dos bifurcaciones son halladas en una cresta pequeña con longitud inferior a D, son eliminadas.
- Si un punto de ramificación tiene al menos dos puntos en la rama vecina, y están cada uno no más lejos que el máximo valor umbral de distancia y estos son puntos de ramificación estrechamente relacionados en el segmento de línea común, entonces son eliminados los puntos de ramificación.

## 1.5 Estado del arte

Actualmente con el desarrollo de las tecnologías se cuenta con herramientas y sistemas de alta potencia que permiten realizar procesos de manera automatizada. Ejemplo de ello son los sistemas de identificación basados en huellas dactilares, cuya aplicación abarca diversas esferas de la sociedad. A continuación se describen algunos sistemas que realizan el proceso de extracción de características en huellas dactilares.

A nivel internacional existe una extensa lista de compañías de software que aplican la biometría en el desarrollo de sus productos. Ejemplos de estos productos son el SourceAFIS y el VeriFinger SDK:

**SourceAFIS** es una biblioteca de clases para el reconocimiento de las huellas dactilares, creado por Robert Vazan, que incluye una interfaz de programación de aplicaciones (*API por sus siglas en inglés Application Programming Interface*) fácil de usar soportada sobre ".Net" y una versión alfa no funcional para Java que no cuenta con el módulo de extracción de características, además de una variedad de aplicaciones y herramientas para la visualización y el análisis de las huellas dactilares. 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. [20]

**VeriFinger SDK** es una tecnología de identificación de las huellas dactilares desarrollada desde 1998 por la compañía Neurotechnology que puede ser utilizada por los desarrolladores e integradores de sistemas biométricos. La última versión de VeriFinger es compatible con NIST MINEX, basada en el motor de identificación de la huella dactilar del MegaMatcher; y sigue el esquema de identificación con más aceptación a nivel mundial. Aquel esquema que utiliza un conjunto de puntos característicos específicos (*minucias*) de las huellas dactilares, junto a un número de soluciones algorítmicas propietarias que mejoran el rendimiento y la fiabilidad del sistema. Es capaz de asegurar que se almacene en la base de datos la plantilla de huellas dactilares de mejor calidad, además, permite mejorar la imagen a través de un filtro adaptativo. 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. Se encuentra disponible como un conjunto de herramientas para el desarrollo de software (*SDK por sus siglas en inglés Software Development Kit)* para MS Windows, Windows CE 3.0 y Linux. [21]

En el ámbito nacional Cuba ha desarrollado productos de software que aplican la biometría, formando parte del desarrollo tecnológico del país. Ejemplos de estos productos son BIOMESYS Control de Asistencia y Biomesys AFIS:

**BIOMESYS Control de Asistencia** es un sistema de reconocimiento basado en huellas dactilares altamente eficaz y preciso. Permite registrar los eventos de asistencia de los empleados en una empresa a partir de la identificación y autenticación del personal mediante un censor biométrico de huellas dactilares. La infraestructura general del sistema está compuesta por varios módulos, entre los que 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. [22]

**Biomesys AFIS** es un sistema automatizado de identificación dactiloscópica, diseñado para reducir los tiempos y aumentar la efectividad en la identificación y autenticación de individuos, a partir de las impresiones dactilares. En su versión civil permite la identificación y autenticación de individuos de forma rápida y segura detectando los intentos de suplantación de identidad. 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. [23]

Los sistemas antes mencionados no son de utilidad para el desarrollo del componente, debido a que están soportados sobre tecnologías propietarias y no contribuyen al cumplimiento del objetivo planteado. Por su parte, la biblioteca de clases SourceAFIS aunque está soportada sobre ".Net", es de código abierto, y

cuenta con una versión alfa no funcional en Java que se utilizará como base para la implementación del proceso de extracción de características en huellas dactilares.

## 1.6 Ambiente de desarrollo

En este apartado se realiza un análisis relacionado con varias metodologías, herramientas y tecnologías existentes, y se establece la definición de aquellas que en conjunto, conforman el ambiente de desarrollo.

### Metodologías de desarrollo de software

Una metodología de desarrollo de software es un conjunto de técnicas, herramientas, procedimientos y soporte documental, que permite a los desarrolladores definir los elementos necesarios para la construcción de un nuevo producto de software. Estas indican paso a paso todas las actividades que se deben realizar para obtener con éxito el producto final, qué personas deben participar en el desarrollo de las actividades y el rol que deben cumplir. También detallan la información necesaria para comenzar una actividad y el resultado que se debe obtener. [24]

Las metodologías de desarrollo de software se clasifican en dos tipos: ágiles y robustas. Las primeras se enfocan fundamentalmente en dar mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas, mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. En cambio, las robustas son conocidas por la forma tradicional de desarrollar software y se basan en un proceso unificado, con un gran flujo de trabajo, obteniéndose una amplia documentación y una detallada planificación inicial que debe seguirse estrictamente; además, son adaptables para proyectos a largo plazo.

**Proceso unificado de software** (*RUP por sus siglas en inglés Rational Unified Process*) es un proceso de desarrollo de software que junto con el lenguaje unificado de modelado (*UML*, *por sus siglas en inglés*, *Unified Modeling Language*), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. Describe la gestión de proyectos como el arte de lograr un balance al gestionar objetivos, riesgos y restricciones para desarrollar un producto que se adapte a las necesidades de los clientes y los usuarios. Además, permite la mitigación temprana de posibles riesgos y gestionar la complejidad; abordando cuestiones de alto riesgo y valorándolas en las primeras iteraciones, donde incluye a los usuarios, principalmente en su fase inicial. Es de utilidad en grandes proyectos y de mediana envergadura con requisitos precisos durante todo el ciclo de vida del proyecto. También verifica constantemente la calidad, asigna tareas y responsabilidades

de forma organizada; gestiona cuidadosamente los requisitos y permite el modelado visual del software. [25]

Dentro de sus principales características se encuentran:

- Es un proceso iterativo e incremental.
- Dirigido por casos de uso.
- Centrado en la arquitectura.
- A través de la gestión de riesgos se pueden reconocer previamente problemas y fallos de forma temprana y prevenirlos o corregirlos.
- Adecuado control de cambios.
- Verificación continúa de la calidad del software.
- Recomendada para proyectos grandes y de muchos recursos (tanto humanos como materiales).
- Requiere la elaboración de una documentación excesiva, lo que aumenta el esfuerzo en el ciclo de vida del proyecto.

**SCRUM** es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software de forma iterativa e incremental donde cada ciclo o iteración, con una duración entre dos y cuatro semanas, termina con una pieza de software ejecutable que incorpora una nueva funcionalidad. Posee la mayor aplicación en la gestión empresarial y se focaliza en priorizar el trabajo en función del valor que tenga para el negocio, maximizando la utilidad de lo que se construye y el retorno de la inversión. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares, de esta forma el producto se puede adaptar en tiempo real, a las necesidades del cliente aumentando su satisfacción. [26]

**Programación extrema** (*XP por sus siglas en inglés Extreme Programing*) es una metodología ágil diseñada para proporcionar el software que el cliente solicita para cuando lo necesite y tiene como principal objetivo la satisfacción del cliente. Debe responder de forma rápida a los cambios, incluso cuando estos se produzcan al final del ciclo de vida del software. También potencia al máximo el trabajo colectivo y tanto el jefe de proyecto, como el cliente y los desarrolladores, son parte del equipo encargado de la implementación del software. Para de esta forma participar en el proceso creativo mediante aportes y sugerencias. Esto implica que los diseños sean claros y sencillos, y que los clientes dispongan de versiones operativas a corto plazo. Las iteraciones son radicalmente más cortas de lo que es usual en otros métodos, de manera que se pueda beneficiar de la retroalimentación tan a menudo como sea

posible. 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 artefacto; es flexible y orientada a la entrega rápida de resultados tangibles. [27]

Está definida especialmente para proyectos pequeños, con requisitos imprecisos, muy cambiantes y se basa en cinco valores fundamentales:

- Comunicación: todos son parte del equipo, existiendo una comunicación cara a cara todos los días.
- Simplicidad: se hace lo que sea necesario y solicitado, pero no más. Esto maximizará el valor creado para la inversión realizada hasta la fecha. Se toman medidas simples para lograr los objetivos y mitigar los fracasos en el momento que ocurren.
- Retroalimentación: existencia de una retroalimentación entre el equipo de desarrollo y el cliente. Se muestran con frecuencia los adelantos del software y se escucha con atención, tanto al equipo de desarrollo, como al cliente.
- Respeto: todo el proyecto da y siente el respeto que merece como miembro del equipo aportando un valor, incluso si es simplemente el entusiasmo.
- Valor: decir la verdad sobre los avances y estimaciones con un plan para tener éxito sin excusas para un posible fracaso.

Entre las metodologías descritas con anterioridad se decidió utilizar XP, pues independientemente de que es la metodología utilizada en el proyecto es adaptable a las necesidades del producto, y no sigue un régimen estricto, lo que facilita que no se añada un trabajo excesivo por la cantidad de roles o documentación que se genera con el uso de la metodología RUP. Además, al tener un enfoque de desarrollo en grupo permite plena interacción con el cliente haciendo viable las contribuciones, por su parte, SCRUM se basa en los principios de inspección continua, adaptación, auto-gestión e innovación, lo que permite la adaptación en tiempo de las necesidades del cliente al producto; pero sólo es posible cambiar el curso de una iteración (*sprint*), abortándolo, mientras que XP añade valor a la respuesta ante el cambio, que es más importante que el seguimiento de un plan.

## Lenguaje de programación a utilizar

Los lenguajes de programación son idiomas artificiales diseñados para expresar cálculos y procesos que serán llevados a cabo por ordenadores, y están formados por un conjunto de palabras reservadas, símbolos, reglas sintácticas y semánticas que definen su estructura, así como el significado de sus

elementos y expresiones. Teniendo en cuenta el objetivo planteado se decide utilizar Java como lenguaje de programación.

Java fue creado por la compañía Sun Microsystems en 1991 y no fue hasta 1995 que apareció como lenguaje de programación, desde entonces se ha convertido en uno de los más populares y se distingue por su portabilidad. Es un lenguaje de desarrollo de propósito general, muy valorado y multiplataforma. En realidad, no es sólo un lenguaje de programación, constituye además, un conjunto de herramientas de desarrollo (*JDK por sus siglas en inglés Java Development Kit*), un entorno de ejecución (*JRE por sus siglas en inglés Java Runtime Environment*) y un conjunto de librerías para el desarrollo de programas sofisticados. El JDK puede conseguirse gratis, es fácil de aprender, está bien estructurado y permite escribir y ejecutar Applets en el ordenador local. Puede considerarse como una evolución del C++, y es un lenguaje orientado a objetos puro, limpio y práctico. [28]

## Entornos de desarrollo

Un entorno de desarrollo es un software informático compuesto por un conjunto de herramientas que permiten al programador desarrollar o implementar códigos de una forma más fácil. Para llevar a cabo la etapa de implementación de la presente investigación se realizó un estudio de las herramientas que pudieran proporcionar un entorno amigable, dentro de los candidatos se encuentran los entornos integrados de desarrollo (*IDE por sus siglas en inglés Integrated Development Environment*) Eclipse y Netbeans.

**Eclipse** facilita enormemente las tareas de edición, compilación y ejecución de programas durante su fase de desarrollo, y constituye una aplicación gratuita y de código abierto, que establece un ambiente ideal para los productores de Applets. Es multiplataforma y aunque pretende ser un entorno versátil soportando varios lenguajes de programación, es con el lenguaje Java con el que mejor se integra y el que ha tributado a ganar su popularidad. Una particularidad de Eclipse es que no necesita instalación, pues una vez descargado el fichero comprimido lo único que se debe hacer para utilizarlo es descomprimirlo en algún directorio y seguidamente ejecutar el binario correspondiente. [29]

**Netbeans** es de código abierto diseñado para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. Es un proyecto de software libre creado en el 2000 por la compañía Sun MicroSystems y tiene una arquitectura modular que permite se le añadan con posterioridad complementos (*plugins*). Soportado sobre la plataforma Netbeans, lo que permite que las aplicaciones sean desarrolladas a partir de un conjunto de bibliotecas de clases y componentes de software predefinidos. Con Netbeans es posible desarrollar tanto aplicaciones de escritorio como

aplicaciones empresariales en varias capas, o programas para todo tipo de dispositivos móviles; además, es un producto libre, gratuito y sin restricciones de uso. [30]

Eclipse y Netbeans se benefician de la capacidad de aceptar plugins de fuentes abiertas o comerciales escritos por los propios desarrolladores de Java, lo que permite tener una plataforma flexible y potente, ya que sus capacidades se extienden hasta donde llegue la imaginación y la destreza de los desarrolladores. Además permite que los IDEs sean neutrales con respecto a la plataforma y el lenguaje, ya que es posible desarrollar plugins para crear un IDE para un lenguaje específico. Sin embargo esto trae como consecuencia que el proceso de desarrollo sea más lento y complicado debido a la búsqueda y configuración del plugins adecuado.

Con Netbeans no se necesitan configurar tantos plugins después de la instalación porque varios de ellos son integrados como parte del sistema base. Con Eclipse es necesario buscar y configurar plugins, pues sin ellos no se proporciona la funcionalidad requerida.

Para la selección del entorno de desarrollo se consideraron aspectos tales como: el lenguaje de programación a utilizar, el objetivo específico que se persigue y el dominio o familiarización del equipo de desarrollo con el IDE sobre el cual se implementará el componente. Como resultado del análisis se decidió utilizar el IDE Netbeans en su versión 7.1.

## Herramientas de modelado

La utilidad de estas herramientas consiste principalmente en realizar un buen diseño del proyecto y a partir de este, implementar parte del código automáticamente, documentar o detectar errores, entre otras características, que hacen de ellas una fuente importante para la creación de un producto con calidad. [31]

Ejemplos de herramientas de ingeniería del software asistida por computadoras (*CASE por sus siglas en inglés Computer Aided Software Engineering*) son: Microsoft Project, Rational Rose, Visual Paradigm, Microsoft Visio y Enterprise Architect.

**Visual Paradigm** es un potente conjunto de herramientas CASE que utiliza UML como lenguaje de modelado y soporta el ciclo de vida completo del desarrollo de software. Su diseño está centrado en casos de uso y enfocado al negocio, permite realizar ingeniería directa e inversa, dibujar todos los tipos de diagramas de clases, y generar documentación. Proporciona tutoriales, demostraciones interactivas y proyectos UML. Presenta licencia gratuita y comercial, además de ser compatible entre ediciones, fácil de instalar y de actualizar. [2]

**Rational Rose** es una herramienta de producción y comercialización establecida por Rational Software Corporation, que utiliza UML para facilitar la captura de dominio de la semántica, la arquitectura y el

diseño del software. Tiene la capacidad de crear, visualizar, modificar y manipular los componentes de un modelo, y aunque 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. [2]

Para modelar el componente se propone Visual Paradigm para UML en su versión 8.0, ya que utiliza un lenguaje estándar para todo el equipo de desarrollo, facilitando así la comunicación. Permite realizar ingeniería tanto directa como inversa, es de software libre, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos y es también multiplataforma. Además, permite llevar la trazabilidad de los requisitos de un proyecto desde que son creados hasta el final.

### Lenguaje de Modelado

UML permite especificar, visualizar y construir los artefactos de los sistemas de software. Es importante resaltar que es un "lenguaje" para especificar y no para describir métodos o procesos. Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. [32]

## 1.7 Propuesta de solución

En la investigación realizada no se encuentran evidencias de sistemas que realicen el proceso de extracción de características 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 extracción y el sistema de verificación dactilar sea un proceso difícil y costoso. Por tales razones se propone realizar la implementación del componente de extracción de características del SourceAFIS en Java, el cual tiene como principal finalidad ser integrado al Applet del sistema de verificación dactilar. Facilitando así, tanto la integración, como la ejecución del proceso de extracción directamente desde el navegador.

## Conclusiones del capítulo 1

El análisis realizado sobre los diferentes aspectos y conceptos involucrados en el proceso de extracción de características en huellas dactilares, posibilitó un mejor entendimiento del contexto en el que se desarrollará la investigación y de la problemática a resolver.

Del estudio realizado a diferentes sistemas que realizan el proceso de extracción de características en huellas dactilares, se concluyó que la versión alfa no funcional del SourceAFIS será la base para la implementación del componente.

Como metodología para el desarrollo del componente se definió XP, la cual está clasificada dentro de las metodologías ágiles y se caracteriza por centrar el desarrollo del producto en la plena interacción con el cliente, quien forma parte del grupo de desarrollo.

Se seleccionó el Netbeans en su versión 7.1 como entorno de desarrollo para llevar a cabo la implementación del componente, entorno que soporta e integra completamente el lenguaje de programación Java.

Por último se decidió utilizar como herramienta CASE Visual Paradigm para UML en su versión 8.0 que utiliza un lenguaje común para todo el equipo de desarrollo.

# Características del componente

Este capítulo describe las características principales del componente, especificando condiciones y cualidades con las que debe cumplir, haciendo énfasis en aspectos relacionados con la metodología de software que guía el proceso de desarrollo, la definición del estilo arquitectónico y patrones de diseño sobre los cuales se implementará el componente.

## 2.1 Descripción del problema

En el departamento de Biometría del CISED se lleva a cabo el desarrollo de un sistema de verificación de individuos mediante huellas dactilares utilizando tecnologías libres; el cual debe contar, entre otros, con un componente que realice el procesamiento de la huella dactilar, con el objetivo de extraer sus características. Los desarrollos previos de algoritmos que realizan este proceso fueron implementados utilizando tecnologías propietarias, ejecutándose de forma centralizada en el servidor en el que está publicado el sistema, provocando sobre-carga del mismo, además de ralentizar la respuesta ofrecida al cliente y que la integración sea un proceso difícil y costoso. Para que el sistema cumpla su objetivo es necesario realizar del lado del cliente el proceso de extracción de características en huellas dactilares.

## 2.2 Modelo de dominio

En el modelo de dominio se muestran los principales conceptos que ayudan a comprender el entorno del componente, y contribuye a describir las clases más importantes dentro de su contexto.

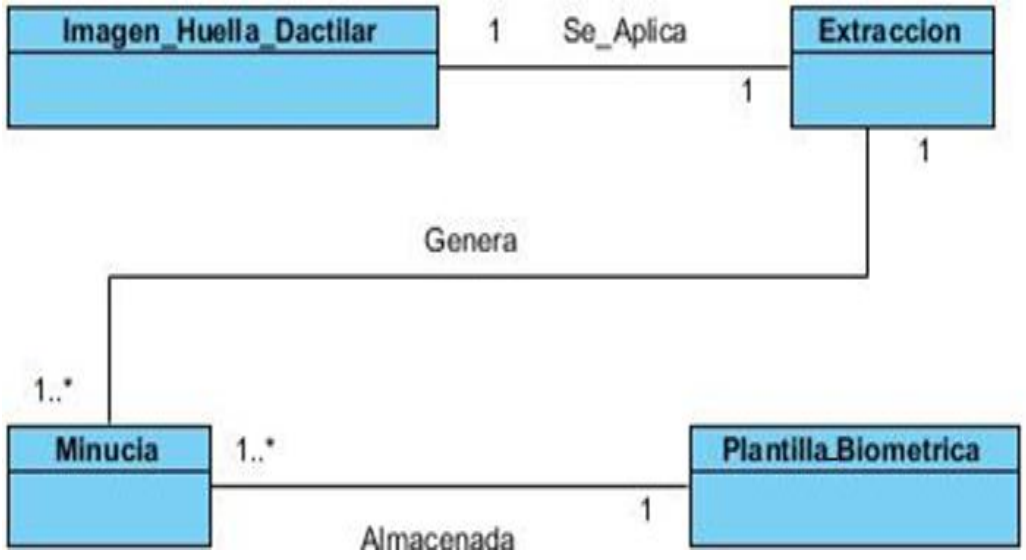


**Figura 5: Modelo de dominio del componente**

**Imagen de la huella dactilar** (*Imagen_Huella_Dactilar*): representación digital de la huella dactilar de un

individuo, obtenida del sistema de verificación dactilar.

**Extracción** (*Extraccion*)**:** es realizado con el fin de extraer todas las minucias presentes en la imagen de la huella dactilar, que puedan ser útiles en la etapa de comparación.

**Minucia:** punto característico de una huella dactilar.

**Plantilla biométrica** (*Plantilla_biometrica*)**:** almacena información de la huella dactilar y está compuesta por un conjunto de minucias y cada una de ellas tiene asociada un tipo, el ángulo de orientación y una coordenada (X,Y) que define el punto donde se encuentra ubicada.

## 2.3 Exploración

La primera fase de la metodología de desarrollo 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 exploran las posibilidades de la arquitectura del sistema construyendo un prototipo.

### Propuesta del componente

El componente propuesto consta de tres etapas fundamentales: pre-procesamiento, extracción de características y post-procesamiento. La entrada del pre-procesamiento es una imagen de la huella dactilar, sobre la cual se aplicará un proceso de mejora para que la fase posterior sea más simple y eficiente. Como parte de esta etapa, inicialmente se definirá la región de interés en la imagen de la huella dactilar mediante la segmentación; seguidamente se normalizará la imagen a través de la ecualización del histograma previamente calculado, para luego obtener la orientación de cada una de las crestas que conforman la huella dactilar. Con el objetivo de facilitar la búsqueda de información útil, la imagen será binarizada y adelgazada; como resultado, estará lista para realizar la extracción de características. En esta etapa se deben detectar y extraer todas las minucias de la huella dactilar, y una vez concluida se aplicará una etapa de post-procesamiento con el fin de validar las minucias y eliminar aquellas que no aportan información de interés. Como resultado de aplicar cada una de las etapas que intervienen en el proceso de extracción de características, debe generarse una plantilla biométrica en formato ISO/IEC 19794-2:2011, que poseerá información útil para ser utilizada en la etapa de comparación.

### 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, además de establecer 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 la red, que cuenta con los recursos, medios y aplicaciones definidos modularmente en los servidores, los que administran, ejecutan y atienden las solicitudes de los clientes; estableciendo un enlace de comunicación transparente entre los elementos que conforman la estructura. [33]

Por otra parte, el patrón arquitectónico **modelo vista controlador** (*MVC por sus siglas en inglés* Model View Controller) separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos: el modelo contiene la información central y los datos, las vistas despliegan información al usuario y los controladores capturan la entrada del usuario. Las vistas y los controladores constituyen la interfaz del usuario. [34]

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 estilo tuberías y filtros 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 otra. [35]

El estilo arquitectónico propuesto para estructurar el componente de extracción de características en huellas dactilares es tuberías y filtros, que por lo general es apropiado cuando se implementan transformaciones de datos sucesivos; encapsulando cada paso del procesamiento en un filtro, donde los datos se pasan usando los tubos entre filtros adyacentes.

El flujo de datos en el componente se comporta de la siguiente manera con la entrada de una imagen de la huella dactilar: quien es trasladada a través de una tubería hacia el filtro de pre-procesamiento, para mejorar su calidad; el resultado obtenido vuelve a una tubería y de ahí pasa al filtro de extracción de características. Al culminar pasa a otra tubería para entrar al filtro de post-procesamiento donde se validan las minucias encontradas, y se genera la plantilla biométrica asociada a la huella dactilar analizada.

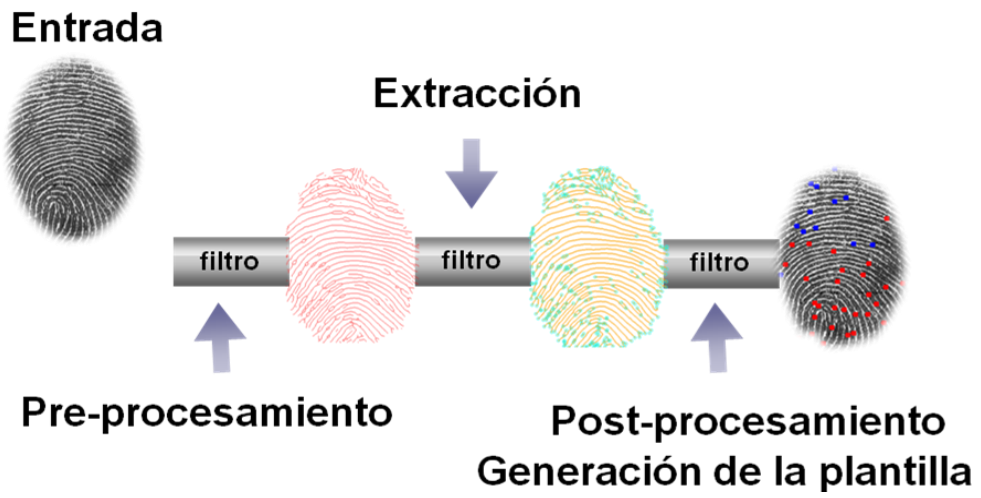


**Figura 6: Flujo tuberías y filtros del componente**

## Requisitos funcionales

Una vez definida la arquitectura los esfuerzos se enfocaron en la definición de los requisitos del componente. Como resultado se obtuvieron los requisitos funcionales ($RF$) que se muestran a continuación, los cuales son capacidades o condiciones que el producto debe cumplir.

- RF 1: Realizar pre-procesamiento de la imagen de la huella dactilar
  - ✓ RF 1.1: Segmentar la imagen de la huella dactilar
  - ✓ RF 1.2: Normalizar la imagen de la huella dactilar
  - ✓ RF 1.3: Calcular la orientación de la huella dactilar
  - ✓ RF 1.4: Binarizar la imagen de la huella dactilar
  - ✓ RF 1.5: Adelgazar la imagen de la huella dactilar
- RF 2: Realizar extracción de características de la imagen de la huella dactilar
- RF 3: Realizar post-procesamiento de la imagen de la huella dactilar
- RF 4: Generar plantilla biométrica de la huella dactilar

## Historias de usuario

Las HU son el artefacto más importante que genera la metodología XP y su propósito, es representar los requisitos funcionales del componente. A continuación se describe la HU asociada a la funcionalidad Realizar pre-procesamiento de la imagen de la huella dactilar. Para consultar el resto de las HU dirigirse al **Anexo_Historias de usuario**.

| **Historia de usuario** | |
|---|---|
| **Número:** 1 | **Usuario:** Sistema |
| **Nombre historia:** Realizar pre-procesamiento de la imagen de la huella dactilar | |
| **Prioridad en negocio:** alta | **Riesgo en desarrollo:** alto |
| **Puntos estimados:** 8 | **Iteración asignada:** 1 y 2 |
| **Programador responsable:** Edelsys Vivian Carrazana Paneque y Leandro Fortún Luna | |
| **Descripción:** con el objetivo de mejorar la imagen de la huella dactilar es necesario realizar los siguientes escenarios que dan paso al cumplimiento de esta funcionalidad:<br>**Escenario Segmentación:** consiste en la separación de la región del primer plano en la imagen de la región del fondo.<br>**Escenario Normalización:** es usada para estandarizar las variaciones de los valores de niveles de gris que constituyen la imagen dentro de cierto rango, considerado lo suficientemente bueno para mejorar su contraste.<br>**Escenario Cálculo de la orientación:** no es más que el cálculo del flujo de una cresta en particular durante un intervalo de píxeles en la imagen.<br>**Escenario Binarización:** es el proceso que convierte una imagen en escalas de grises a una imagen binaria donde cada píxel toma valor de 0 o 1.<br>**Escenario Adelgazamiento:** como resultado de este proceso todas las crestas tienen el grosor de 1 píxel haciendo que los puntos característicos de la huella dactilar se puedan identificar con más facilidad. | |
| **Observaciones:** se debe disponer de la imagen de la huella dactilar | |
| **Prototipo de interfaces:** sin prototipo de interfaz | |

**Tabla 3: HU_ Realizar pre-procesamiento de la imagen de la huella dactilar**

## Requisitos no funcionales

Los requisitos no funcionales (*RNF*) son propiedades o cualidades que el producto debe tener. Son características que hacen al producto usable, rápido o confiable.

| Número | Requisito | Clasificación |
|---|---|---|
| RNF-1 | El componente está concebido para un ambiente donde se integre a un sistema de verificación, por tanto los tiempos de respuestas deben ser rápidos, así como la velocidad de procesamiento de la información | Rendimiento |
| RNF-2 | PC (*computadora personal del inglés Personal Computer*) Intel Pentium 4 o superior<br>CPU (*unidad central de procesamiento del inglés Central Processing Unit*) 1.6 GHZ o superior | Hardware |

| | | |
|---|---|---|
| | 512 MB de Memoria RAM (memoria de acceso aleatorio del inglés *Random Access Memory*) o superior | |
| RNF-3 | JRE versión 7.5 | Software |

**Tabla 4: Requisitos no funcionales del componente**

## 2.4 Planificación

La planificación 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 HU, así como su entrega. Esta planificación debe ser flexible para que pueda adaptarse a los posibles cambios que puedan surgir.

### Estimación del esfuerzo por historias de usuario

Las estimaciones del esfuerzo asociado a la implementación de las HU la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Los resultados obtenidos en esta estimación de acuerdo a las HU se exponen en la siguiente tabla:

| No. | Historias de usuario | Estimación (semanas) |
|---|---|---|
| 1 | Realizar el pre-procesamiento de la imagen de la huella dactilar | 8 |
| 2 | Realizar la extracción de minucias de la imagen de la huella dactilar | 4 |
| 3 | Realizar el post-procesamiento de la imagen de la huella dactilar | 4 |
| 4 | Generar la plantilla biométrica de la huella dactilar | 3 |

**Tabla 5: Estimación del esfuerzo por HU**

### Plan de iteraciones

Una vez identificadas y descritas las HU, y estimado el esfuerzo dedicado a la realización de cada una, 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 HU son traducidas como tareas de programación, que serán desarrolladas y probadas a través del orden que se defina en el siguiente ciclo de iteraciones.

**Iteración 1 y 2**

En estas iteraciones se implementa la HU_Realizar pre-procesamiento de la imagen de la huella dactilar que se considera tiene la mayor prioridad y trascendencia, pues de ella depende el resto.

**Iteración 3**

En la tercera iteración se procede con la implementación de la HU_Realizar extracción de características de la huella dactilar, además, se corregirán los errores e inconformidades del cliente respecto a la HU implementada en la iteración anterior.

**Iteración 4**

En la cuarta iteración se realizará la implementación de la HU_Realizar post-procesamiento de la imagen de la huella dactilar y se corregirán los errores e inconformidades del cliente respecto a la HU implementada en la iteración anterior.

**Iteración 5**

Esta es la última iteración que se efectuará, en ella se implementa la HU_Generar plantilla biométrica. Corrigiendo los errores e inconformidades de la iteración anterior. De esta manera se obtendrá la primera versión funcional del producto final.

## Plan de duración de las iteraciones

El plan de duración de las iteraciones persigue el objetivo de mostrar en qué orden serán implementadas las HU en cada iteración, así como el tiempo destinado a cada una de ellas, permitiendo al equipo de desarrollo y al cliente, conocer cuánto durará la obtención del componente.

| NO. | Historias de usuario | Duración de las iteraciones |
|---|---|---|
| Iteración#1 y 2 | Realizar el pre-procesamiento de la imagen de la huella dactilar | 8 |
| Iteración#3 | Realizar la extracción de características de la imagen de la huella dactilar | 4 |
| Iteración#4 | Realizar el post-procesamiento de la imagen de la huella dactilar | 4 |
| Iteración#5 | Generar la plantilla biométrica de la huella dactilar | 3 |

**Tabla 6: Plan de duración de las iteraciones**

## 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 cada iteración.

| No. de iteración | Fecha límite |
|---|---|
| Iteración#1 y 2 | 24/02/2013 |
| Iteración#3 | 24/03/2013 |
| Iteración#4 | 21/04/2013 |
| Iteración#5 | 12/05/2013 |

**Tabla 7: Plan de entrega por iteraciones**

## 2.5 Diseño

Cuando los requisitos, tanto funcionales como no funcionales, se traducen en una representación del software se está en presencia de la fase de diseño. Su objetivo, es producir un modelo de instrumentación o implantación basado en los modelos conceptuales desarrollados durante el análisis, para distribuir los datos y los procesos del software en entidades que se van a construir con posterioridad. [36]

### Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema de diseño general en un contexto particular, y brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones son una pareja de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. [36] Entre los patrones que se utilizarán se encuentran los patrones generales de software para la asignación de responsabilidades (*GRASP por sus siglas en inglés General Responsibility Asignment Software Patterns*) y los patrones GoF (*Gang Of Four*).

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos expresados en formas de patrones y los usados en el componente se describen a continuación:

**Controlador:** es un objeto de interfaz no destinada al usuario que se encarga de efectuar las asignaciones en cuanto al manejo de los eventos del sistema y definir sus operaciones. Se evidencia en la clase “Extraction.java” que gestiona todo el flujo de datos del componente, además, posee la información necesaria para instanciar los objetos que serán utilizados durante todo el proceso. Ejemplo:

```
public class Extraction {
    public LocalHistogram Histogram = new LocalHistogram();
    public SegmentationMask Mask = new SegmentationMask(); (…)
```

```
        public byte[] Extract(final byte[][] invertedImage, final int dpi) {…}
}
```

**Experto:** el uso de este patrón permite a los objetos valerse de su propia información para hacer lo que se les pide. Por ejemplo, el objeto "Histogram" definido en la clase principal no puede cumplir con la responsabilidad de adelgazar la huella dactilar porque no cuenta con la información necesaria para realizar este proceso. Ejemplo de la clase experta en cuestión:

```
public final class Thinner {(…)
        public BinaryMap Thin(final BinaryMap input) {…}
}
```

**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 sólo pueda ser creada por la clase que contiene la información necesaria para ello. Por ejemplo, el objeto "Histogram" solamente puede ser creado por la clase "LocalHistogram".

```
public class Extraction {
        public LocalHistogram Histogram = new LocalHistogram(); (…)
}
```

**Bajo acoplamiento:** el siguiente diagrama muestra como se delegan las responsabilidades a las diferentes clases del componente, permitiendo visualizar que la relación entre ellas es mínima reduciendo así el impacto de posibles cambios y haciéndolas reutilizables.

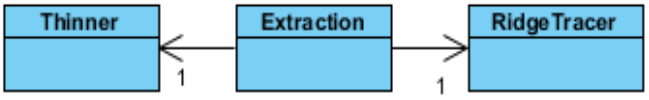


**Figura 7: Ejemplo que evidencia el patrón bajo acoplamiento en el componente**

**Alta cohesión:** en el componente este patrón se ve presente en la clase "Extraction" pues con ella colaboran para realizar el proceso de extracción, las clases "Thinner" y "ThresholdBinarizer" las cuales tienen responsabilidades moderadas con propósitos muy específicos, permitiendo que sean fácil de mantener, entender y reutilizar. Diagrama de ejemplo:


**Figura 8: Ejemplo que evidencia el patrón alta cohesión en el componente**

Los patrones GoF son también conocidos como los patrones de la pandilla de los cuatro y se agrupan en tres grandes categorías: creacionales, estructurales y de comportamiento. [37]

Los creacionales muestran la guía de cómo crear instancias de objetos, su objetivo es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados. Dentro de esta categoría se encuentran Fábrica abstracta, Constructor, Método de fabricación, Prototipo, y Singleton.

**Singlenton o instancia única:** este patrón se evidencia en la clase "Ridge.java", donde se crea un objeto privado de la clase, y para acceder a él, se utiliza un método de acceso global, permitiendo que siempre se utilice la misma instancia del objeto. Ejemplo:

```
public final class Ridge {
        Ridge Reversed;
        public Ridge getReversed() {…}
}
```

Otra de las categorías son los estructurales, que describen cómo las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Ejemplos de patrones estructurales son: Adaptador, Puente, Compuesto, Decorador, Facade y Peso ligero.

**Facade o fachada:** este patrón se pone de manifiesto en la clase "JAppletExtraction.java", la cual crea un objeto de la clase principal que actúa como intermediario para realizar el proceso de extracción; permitiendo, a quien lo utilice, abstraerse de cómo se realiza. Ejemplo:

```
public String Extract(String path) throws Exception {
        Extraction obj = new Extraction
        return convertToString(obj.Extract(invertedImage, dpi));
 }
```

Por otra parte los patrones de comportamiento ayudan a definir la comunicación entre los objetos de un sistema y su objetivo es reducir el acoplamiento entre los objetos; y se clasifican en: Cadena de responsabilidad, Orden, Intérprete, Iterador, Mediador, Observador, Estado, Estrategia, Método plantilla y Visitante.

**Iterator o Iterador:** este patrón se evidencia en la clase "ReversedList.java", la cual declara métodos para acceder secuencialmente a los objetos de una colección. Ejemplo:

```
public final class ReversedList<T> implements List<T> { (…)
```

```
        class InnerIterator<E> implements Iterator<E> {(…) }
}
public final class Ridge { (…)
        Ridge(Ridge reversed) { (…)
        Points = new ReversedList<>(reversed.Points); }
}
```

## Diagrama de clases

La metodología XP no establece una representación del producto mediante diagramas de clases, paquetes u otros; en su lugar, define la elaboración de las llamadas tarjetas de clase, responsabilidad y colaboración (*CRC por sus siglas en inglés Class, Responsabilities and Collaboration*) que permiten establecer relaciones y colaboraciones entre clases. El uso de diagramas puede ser útil siempre y cuando influyan en un mejor entendimiento de la comunicación y se enfoquen en los aspectos más relevantes del producto.

En la presente investigación, los diagramas de clases que se definieron representan los procesos principales del componente, y cada uno está contenido dentro de un paquete; pues al crear el diagrama de clases se hace muy extensa su representación. Ver **Anexo_Diagramas de clases**.

## Diagrama de paquetes

El presente diagrama de paquetes describe la relación que se establece entre los paquetes que integran al componente.

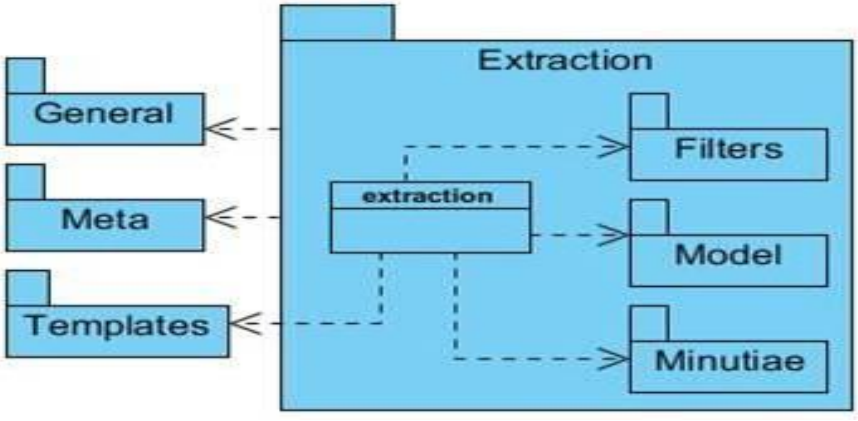


**Figura 9: Diagrama de paquetes del componente**

En el paquete “Extraction” se encuentra la clase principal “extraction”, encargada de ejecutar todo el proceso de forma secuencial y para ello se apoya de los paquetes “Filters” para el pre- procesamiento de la imagen, “Model” para la extracción y el post-procesamiento, “Minutiae” y “Templates” para la generación de la plantilla biométrica y como apoyo para todo el proceso se utilizan los paquetes “General” y “Meta”.

## Tarjetas CRC

El uso de las tarjetas CRC permite al programador centrarse y apreciar el desarrollo orientado a objetos olvidándose de los malos hábitos de la programación procedural clásica. Cada tarjeta representa una clase con su nombre en la parte superior, en la sección inferior izquierda se describen las responsabilidades y a la derecha las clases que le sirven de soporte. [38]

Una clase es cualquier persona, evento, concepto, pantalla o reporte y sus responsabilidades son las que conoce y las que realizan sus atributos y métodos. Los colaboradores son las demás clases con las que trabaja en conjunto para llevar a cabo sus responsabilidades. A continuación se muestra la tarjeta CRC asociada a la clase “Thinner.java”, para consultar el resto dirigirse al siguiente enlace: **Anexo_ Tarjetas CRC.**

| **Clase: Thinner.java** | | |
|---|---|---|
| **Responsabilidad** | **Colaboran** | |
| Llevar las crestas de la huella dactilar al grosor de un píxel | package extraction.filters<br>➢ BinaryMap.java<br>➢ Calc.java<br>➢ Neighborhood.java<br>➢ Point.java<br>➢ RectangleC.java<br>➢ Parallel.java | ➢ ParallelForDelegate.java<br>➢ ParallelForEachDelegate.java<br>package meta<br>➢ DpiAdjusted.java<br>➢ Parameter.java |

**Tabla 8: Tarjeta CRC asociada a la clase Thinner del pre-procesamiento de la imagen**

## Diseño del prototipo de la interfaz de prueba

Los prototipos son una de las mejores herramientas a la hora de diseñar interfaces gráficas y su elaboración proporciona a los desarrolladores una ayuda de cómo debe quedar la interfaz que se utilizará en un proyecto determinado. Al realizar un prototipo siempre se debe tener bien definido qué es lo que se trata de reflejar, y generalmente son aquellos aspectos de la interfaz del usuario referidos a las interacciones que proporciona el sistema. A continuación se muestra el diseño de la interfaz de prueba del componente de extracción de características en huellas dactilares.

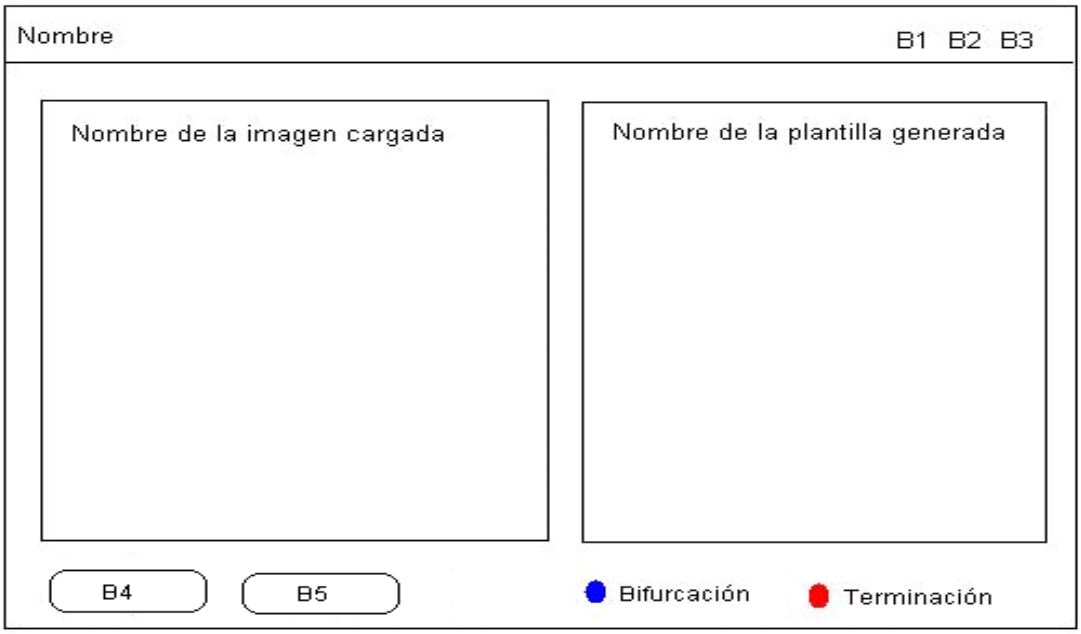


**Figura 10: Prototipo de la interfaz de prueba del componente**

- Nombre: se refiere al nombre que identifica al componente.
- B1: botón para minimizar el componente, B2: botón para maximizar el componente, B3: botón para cerrar el componente.
- Nombre de la imagen cargada: representa la imagen de la huella dactilar que se carga con el objetivo de aplicarle el proceso de extracción de características.
- Nombre de la plantilla generada: representa la plantilla que se obtiene como resultado de aplicar sobre la imagen original el proceso de extracción de características.
- B4 y B5: botón para indicar la ejecución del proceso de extracción a la imagen de prueba que se encuentra en la posición anterior o posterior a la actual. Las imágenes se cargan automáticamente desde el directorio “img/” y el resultado es almacenado en el directorio “template/”.
- Los puntos en azul y rojo representan los tipos de minucias bifurcaciones y terminaciones respectivamente en la plantilla biométrica generada.

## Conclusiones del capítulo 2

La descripción de las principales características del componente, permitió construir el modelo de dominio para un mejor entendimiento del problema.

Se identificaron RF y RNF brindando al equipo de desarrollo una mayor visión del producto deseado por el cliente, lo que posibilitó generar el principal artefacto de la metodología XP, las HU, que describen y se corresponden con los RF identificados.

Se definió como base estructural del componente el estilo “Tuberías y Filtros” que brinda la posibilidad de ir transformando un flujo de datos en un proceso comprendido por varias fases secuenciales, siendo la entrada de cada una la salida de la anterior.

Se describieron los patrones de diseño utilizados y se brindó una breve representación organizacional del componente a través del diagrama de paquetes propuesto.

La confeccionaron de las tarjetas CRC, permitió comprender a grandes rasgos la relación que se establece entre las clases, para realizar la implementación del componente.

La elaboración del prototipo de la interfaz de prueba posibilitó un acercamiento a cómo se manifestará la interacción con el componente.

# Implementación y prueba

Con anterioridad se expusieron aspectos relacionados con las características que debe poseer la solución propuesta, de vital importancia para dar continuidad a la investigación. A continuación se aborda la implementación del componente referenciando los estándares de programación a utilizar, así como la traducción de las HU en tareas de ingeniería, permitiendo dar cumplimiento a los requerimientos planteados. Se describen además, los problemas encontrados durante el proceso de implementación y las pruebas realizadas para validar el correcto funcionamiento del componente.

## 3.1 Estado de implementación de la versión alfa no funcional del SourceAFIS en Java

Al iniciar la implementación del componente, la versión alfa no funcional del SourceAFIS en Java contaba con un 33,8 % de clases implementadas; de ellas, el 25,35 % representaban clases totalmente implementadas y el 8,45 % clases parcialmente implementadas. Con respecto al 100 %, el 66,2 % constituían clases no implementadas y tenían correspondencia con el por ciento de clases necesarias para realizar el proceso de extracción de características.

El análisis anterior se representa en la siguiente gráfica:

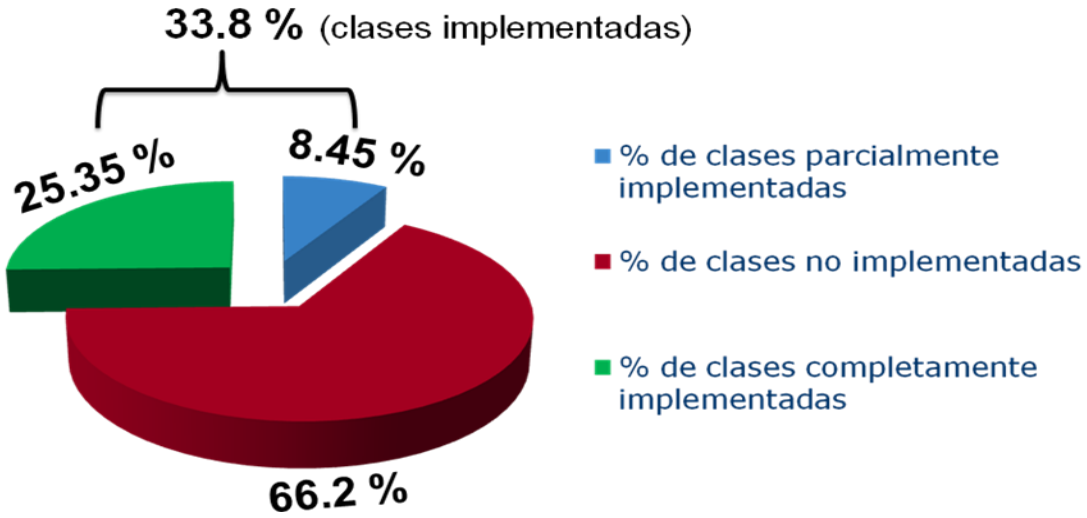


**Figura 11: Representación gráfica del estado de implementación de la versión alfa no funcional del SourceAFIS en Java**

En la versión en C#, para realizar el proceso de extracción se contaba con un total de cuatrocientos métodos implementados en dos mil sesenta y cinco líneas de código; un análisis de complejidad ciclomática (*CC del inglés Cyclomatic Complexity*) realizado arrojó, que de ellos, cuatrocientos siete tenían una CC normal, tres eran muy complejos y uno extremadamente complejo. El análisis realizado con

anterioridad se muestra en la siguiente gráfica:

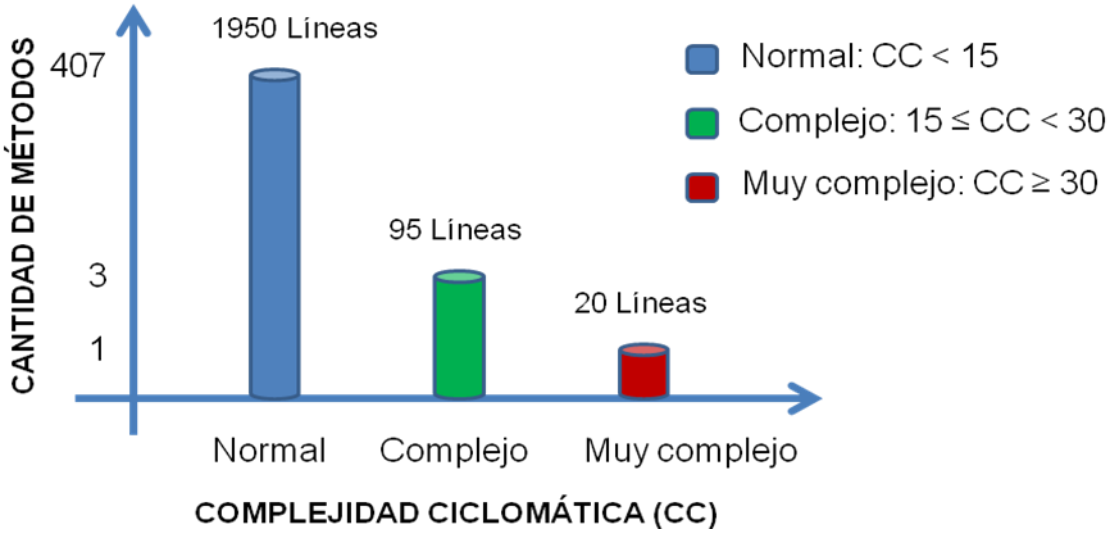


**Figura 12: Complejidad ciclomática del proceso de extracción en C#**

La CC es una métrica del software que proporciona una medida cuantitativa de la complejidad lógica de un programa y se basa en el recuento del número de caminos lógicos individuales contenidos en un programa. Para los métodos se define como: 1 + {el número de expresiones que se encuentran en el cuerpo del método}, pero también puede ser calculada como:

1. CC = e - n + 2 donde “e” representa el número de aristas y “n” el número de nodos.
2. CC = número de regiones cerradas en el grafo + 1

Expresiones a tener en cuenta: if | while | for | foreach | case | default | continue | goto | && | || | catch | ternary operator ? : | ??

Para el cálculo de la CC se utilizó la herramienta NDepend en su versión 4 para Visual Studio.Net. Para el método de ejemplo “GetMaskLineRange” la CC calculada mediante el NDepend resultó ser igual a 5, además se utilizó la fórmula 2 para comparar resultados, y se obtuvo el mismo valor para ambos casos. Ver **Anexo_Complejidad mediante la fórmula 2**.

## 3.2 Estándar de programación

Para la implementación del componente es necesario establecer nomenclaturas o estándares que sean entendibles por cada uno de los programadores que integran el equipo de desarrollo, de forma tal que se siga un mismo criterio para desarrollar el código.

Un estándar de programación es una forma de "normalizar" la programación, de manera tal, que al trabajar en un proyecto, cualquier persona involucrada tenga acceso y comprenda el código. [39] Permite definir la escritura y organización del código fuente de un programa, facilitando a un programador la modificación de su propio código fuente, aunque no esté trabajando en el equipo; además de definir la forma en que deben ser declaradas las variables, las clases, los comentarios y especificar qué datos deben incluirse acerca del programador y de los cambios realizados al código fuente. [40]

Los estándares de programación son decisivos para poder plantear con éxito la propiedad colectiva del código, y para lograrlo, se debe contar con una codificación basada en estándares que haga que todos se sientan cómodos con el código escrito por otro miembro del equipo. A continuación se define el estándar de codificación para el componente de extracción de características en huellas dactilares:

## Ficheros

Como primer elemento se tienen los nombres de ficheros o sufijos en Java para el código fuente y para cuando sea compilado:

Código fuente: Applet.java

Código compilado: JavaAppletExtraction.jar

Un fichero consta de secciones separadas por líneas en blanco y líneas de código y no deben superar las dos mil líneas de código. Considerando este criterio, la organización de los ficheros se enuncia de la siguiente manera:

- Cada fichero del código fuente contiene una única clase, interface o enumerativo y en ellos se podrán tener clases anidadas. Ejemplo:

  public final class HillOrientation {…

  class NeighborInfo {…} }

- Las secciones en un fichero del código fuente deben ir separadas por una línea en blanco:

Sentencia: package extraction;

Sentencias de importación: import extraction.filters.*; No hay líneas en blanco entre ellas.

Código de la clase: precedido por comentarios con la siguiente información: autor, versión y prólogo explicativo de la clase. Esto únicamente se hará para las clases que lo requieran por el nivel de complejidad. Ejemplo:

/**

```
* @author Leandro Fortún Luna
* @author Edelsys Vivian Carrazana Paneque
* @version 1.0
* @descripción de la clase InnerMask: su responsabilidad es delimitar el área
* de interés que será utilizada para eliminar aquellos puntos
* característicos que no se encuentren dentro de esta área.
*/
```

- Las líneas de código largas que sobrepasen los ochenta caracteres se deben fraccionar atendiendo a las siguientes especificaciones:

Fraccionar después de una coma con espaciado de ocho caracteres, se evidencia dentro de (……). Ejemplo:

```
double[][] equalized = Equalizer.Equalize(blocks, image,
(……)smoothHistogram, mask);
```

Fraccionar después de un operador con espaciado de 8 caracteres. Ejemplo:

```
result.Map[y][Map[0].length - 1] &= ~(long)0
(……) >> (int)((long)WordSize - (Width & WordMask));
```

## Declaraciones

Otro aspecto es cómo se declaran las variables, clases o interfaces, y el número de declaraciones que se realizan sobre una misma línea.

- Por ello cada variable será declarada en líneas distintas. Ejemplo: private final byte value;
- Los arreglos se deben declarar: private static boolean[] IsRemovable;
- La declaración de las listas deben tener la siguiente estructura: private List<Point> Points;

La inicialización de las variables cuando se declaran, sólo se efectuará si su valor inicial depende de algún cálculo. Ejemplo:

- El valor inicial no depende de un cálculo: private final byte value;
- El valor inicial depende de un cálculo: public static final int WordBytes = WordSize / 8;

Las declaraciones de clases, interfaces o métodos se regirán por las siguientes reglas:

- Se abre el cuerpo de la clase, interfaz o métodos con la llave "{" al final su declaración y se cierra con llave "}"en una línea aparte al mismo nivel que el método o clase que cierra.
- Hay una línea entre la declaración de la clase, interfaz o método (*no es obligatorio, depende de la complejidad de la línea*) y el cuerpo.
- No hay espacio entre el nombre del método y el paréntesis, pero sí hay un espacio entre la lista de parámetros.

Ejemplo clase:

public final class HillOrientation {…}

Ejemplo interfaz:

public @interface DpiAdjusted { …}

Ejemplo método:

public byte[][] Detect(double[][] image, BinaryMap mask, BlockMap blocks) {…}

## Sentencias

Sentencias simples: cada línea debe contener una sola sentencia.

Sentencias compuestas: son las que están entre las llaves "{" y "}" de forma anidada. Estas deben estar a cuatro espacios de la instrucción que la contiene.

Todas las sentencias del tipo if, for, while, do... while deben tener llaves, aunque sólo contengan una línea de código, de esta forma se evita la introducción accidental de errores, si se añaden posteriormente otras instrucciones.

Las normas genéricas para la asignación de nombres son:

1. Se usan descriptores en inglés que precisan el cometido de la variable, método, clase o interfaz.
2. Se usan terminologías aplicables al dominio.
3. Se debe evitar en lo posible los nombres largos (menos de quise letras sería lo ideal), a menos que esto traiga consigo que no se cumpla la norma 1.
4. Evitar nombres que difieran en una letra o en el uso de mayúsculas.
5. Un nombre no debería contar con más de tres palabras.
6. No usar siglas en los nombres a menos que queden muy largos o sean siglas conocidas por todos.

7. Cada tipo de elemento debe nombrarse con una serie de reglas determinadas.
8. Paquetes: todo en minúsculas.
9. Clases, interfaces y enumerativos:
    - Clases: la inicial en mayúscula excepto la clase principal.
    - Métodos: la primera letra en mayúscula, si no es un método contenido dentro de otro, en este caso debe ser con minúscula y el resto de las palabras empiezan con mayúsculas.
    - Atributos: cada letra inicial de las palabras que conformen el nombre del atributo será en mayúsculas.
    - Variables: deben comenzar con minúscula.

Nota: no se utilizará en ningún caso el caracter "_" delante o entre dos palabras que conformen un nombre.

## 3.3 Tareas de ingeniería

Con el objetivo de obtener una versión funcional del producto que pueda ser probado por el cliente al culminar cada iteración, se descomponen las HU en tareas de ingeniería, las que son asignadas a los programadores para ser implementadas durante la iteración correspondiente. Las tareas de ingeniería se corresponden con las HU identificadas, pero una HU puede tener asociada varias tareas, lo que posibilita dar cumplimiento a los requerimientos establecidos por el cliente. A continuación se describen las tareas de ingeniería correspondientes a las HU identificadas.

| **Tareas de ingeniería** | | | | |
|---|---|---|---|---|
| **Id** | **No_HU** | **Nombre_HU** | **Nombre_Tarea** | **Responsable** |
| 1 | 1 | Realizar pre-procesamiento de la imagen de la huella dactilar | Implementar el escenario segmentación de la imagen | Leandro Fortún Luna |
| 2 | 1 | Realizar pre-procesamiento de la imagen de la huella dactilar | Implementar el escenario normalización de la imagen | Leandro Fortún Luna |
| 3 | 1 | Realizar pre-procesamiento de la imagen de la huella dactilar | Implementar el escenario cálculo de la orientación | Leandro Fortún Luna |
| 4 | 1 | Realizar pre-procesamiento de la imagen de la huella dactilar | Implementar el escenario binarización de la imagen | Edelsys Vivian Carrazana Paneque |

| | | | | |
|---|---|---|---|---|
| 5 | 1 | Realizar pre-procesamiento de la imagen de la huella dactilar | Implementar el escenario adelgazamiento de la imagen | Edelsys Vivian Carrazana Paneque |
| 6 | 2 | Realizar extracción de minucias de la imagen de la huella dactilar | Implementar la extracción de características de la imagen | Leandro Fortún Luna |
| 7 | 3 | Realizar post-procesamiento de la imagen de la huella dactilar | Implementar el post-procesamiento de la imagen | Edelsys Vivian Carrazana Paneque |
| 8 | 4 | Generar plantilla biométrica de la huella dactilar | Implementar el estándar para generar la plantilla biométrica de la huella dactilar | Leandro Fortún Luna |

**Tabla 9: Tareas de ingeniería asociadas a las HU**

## 3.4 Principales problemas en el proceso de implementación

En la medida en que fueron implementadas cada una de las tareas de ingeniería, se encontraron incompatibilidades asociadas a la diferencia entre los lenguajes C# y Java. A continuación se presentan ejemplos de los problemas y la solución propuesta.

1) Al implementar varios métodos asociados a la etapa de pre-procesamiento, se pudo apreciar que la representación del rango de valores del tipo de dato “byte” tanto en C**#** (el rango de valores va de 0 a 255) como en Java (el rango de valores va de -128 a 127) es diferente. Tal argumento impedía que la salida del histograma en Java coincidiera con los valores que arrojaba la versión en C#. Entre las clases en las que se evidencia se encuentra “LocalHistogram.java”, y en el cuerpo de esta, en el método “Analyze”.

Código ejemplo del problema: ++histogram[block.Y][block.X][image[y][x]];

Para C#:

block.Y = 0 y block.X = 0

image[0][0] = -11 (posición no válida en la matriz del histograma)

++histogram [0][0][**-11**];

Propuesta de solución: se realizó un “AND o &” lógico con “255 o 0xFF” con el objetivo de llevar el número deseado a un rango de 0 a 255.

Código ejemplo de la solución: ++histogram[block.Y][block.X][image[y][x] **& 0xFF**];

Para Java:

block.Y = 0 y block.X = 0

((image[0][0] = -11) **& 0xFF**) = 245 (posición válida en la matriz del histograma)

++histogram [0][0][**245**];

2) Al realizar el proceso de segmentación los resultados obtenidos no eran los esperados, debido a que el mapa binario que utiliza la biblioteca de clases SourceAFIS solamente trabaja con números positivos, para ello utilizan el tipo de dato unsigned integer o uint de C# (entero sin signo que almacena valores de 0 a 429 496 729 5). Java no permite el trabajo solamente con números positivos, por lo que se hace necesario utilizar un tipo de dato mucho más grande donde pueda ser almacenado hasta el mayor número del uint.

Código ejemplo del problema: Map[y][x] = ~Map[y][x];

Para C#:

Map[0][0] = 268435455

Para Java:

~Map[0][0] = **-268 435 456** (el resultado que se almacena en el mapa binario es un número negativo)

Propuesta de solución: se decide utilizar el tipo de dato long, que permite almacenar todos los valores del uint. Cuando se realiza una operación a nivel de bit, por ejemplo una negación (NOT o ~), si el número es positivo el resultado sería negativo, por lo que si se hace un "And/&" lógico con "& 0xffffffffL" (representa el máximo valor que alcanza el uint), se obtiene una representación de este número negativo en los positivos, obteniéndose así los resultados esperados.

Código ejemplo de la solución: Map[y][x] = ~Map[y][x] **& 0xffffffffL**;

Para Java:

Map[0][0] = 268 435 455

~Map[0][0] = -268 435 456

Map[y][x] = -268 435 456 & 0xffffffffL = 402 653 184 0 (el resultado que se almacena en el mapa binario es una representación del número negativo en los positivos)

3) Durante la implementación de la normalización y el post-procesamiento se presentaron problemas de incompatibilidad de tipos de datos asociados a los float, debido a que el redondeo luego de la coma, en C# y en Java no son interpretados de igual forma.

Ejemplo del problema para la normalización en el método "ComputeEqualization" de la clase

"Equalizer.java":

Operación: float widthMax = RangeSize / 256f * MaxScaling;

Para C#:

RangeSize = 2.0 y MaxScaling = 3.99

float widthMax = 2.0 / 256f * 3.99 = 0.031 171 875 **1**

Para Java:

RangeSize = 2.0 y MaxScaling = 3.99

float widthMax = 2.0 / 256f * 3.99 = 0.031 171 875

Estas pequeñas variaciones traen consigo que los valores de salida en la ecualización del histograma sean diferentes, influyendo este resultado solamente en el cálculo de la orientación, no así en los demás procesos que se realizan.

Ejemplo del problema para el post-procesamiento en el método ConstructLine de la clase Calc:

Operación: result[i] = new Point(from.X + i, from.Y + Convert.ToInt32(i * (relative.Y / (float) relative.X)));

Para C#:

(float) relative.X = 2.0, relative.Y = 1 y i = 1

Convert.toInt32(1 * (1 / (float) 2.0) = **0**

new Point(213 + 1, 146 + **0**)

result[1] = [214, 14**6**]

Para Java:

(float) relative.X = 2.0, relative.Y = 1 y i = 1

Math.round (1 * (1 / (float) 2.0) = **1**

new Point(213 + 1, 146 + **1**)

result[1] = [214, 14**7**]

La clase "Convert.ToInt32 (Método)", convierte el valor que se le especifica en un entero de 32 bits con signo. Para representar esta conversión Java cuenta con la clase "Math.round (Método)", pero estas clases no trabajan el redondeo por exceso y por defecto de igual forma. La siguiente figura representa el comportamiento de las clases "Convert.ToInt32 (Método)" y "Math.round (Método)".

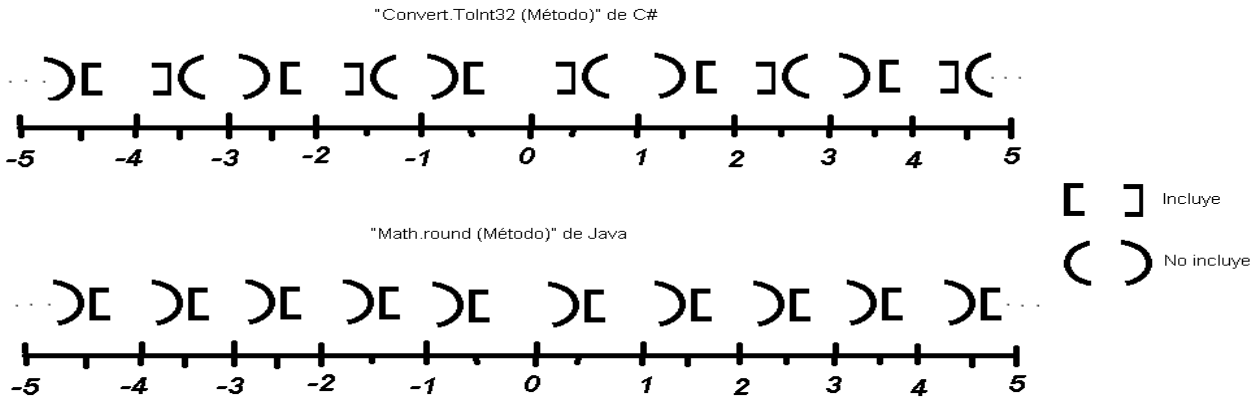


**Figura 13: Comportamiento de las clases "Convert.ToInt32" de C# y "Math.round" de Java**

Para solucionar este problema se estudió el comportamiento de la clase "Convert.ToInt32 (Método)", y se implementó ese comportamiento.

Propuesta de solución:

- Paso 1: se obtiene la parte entera del número especificado.
- Paso 2: se almacena el signo del número, si es negativo se corresponde con el valor -1, si es positivo el valor es 1.
- Paso 3: se obtiene el valor absoluto del número.
- Paso 4: si el resto de la división entera del Paso 1 entre dos es igual a cero, se comprueba que Paso 3 sea menor o igual a 0.5; si se cumple la condición, el resultado es igual a Paso 1, sino, el resultado es igual a Paso 2 por el valor absoluto de Paso 1 más uno.
- Paso 5: si Paso 4 no se cumple, se comprueba que Paso 3 sea mayor o igual a 0.5, si se cumple el resultado es igual a Paso 2 por el valor absoluto de Paso 1 más 1, en caso contrario el resultado es igual a Paso 1.

4) Otro de los problemas encontrados es el asociado al uso del lenguaje integrado de consultas (*LINQ por sus siglas en inglés Language-Integrated Query*), que es usado en C# para el trabajo con colecciones de datos y que no es soportado por el lenguaje Java.

Representación del problema en C#:

```
public void Filter(TemplateBuilder template){
        var radiusSq = Calc.Sq((float)NeighborhoodRadius);
        template.Minutiae = template.Minutiae.Except(
```

```
            (from minutia in template.Minutiae
            where template.Minutiae.Count(neighbor => Calc.DistanceSq(neighbor.Position,
            minutia.Position) <= radiusSq) - 1 > MaxNeighbors
            select minutia).ToList()).ToList();
    }
```

Nota: La CC para este método es igual a 1 y fue calculada con la herramienta "NDepend en su versión cuatro".

Solución al problema en Java:

```
            public void Filter(TemplateBuilder template) {(…)}
            private List<Minutia> getSortedMinutiaList(TemplateBuilder template) {(…)}
            private Integer getNeighborDistance(Minutia minutia, TemplateBuilder template) {(…)}
```

Nota: La CC para estos métodos es igual a 2, 3 y 2 respectivamente.

## 3.5 Descripción de la interfaz de prueba

La interfaz es el medio mediante el cual el usuario interactúa con un sistema e incluye elementos intuitivos, normalmente suelen ser fáciles de entender y fáciles de accionar. Como se hizo referencia en el capítulo anterior, el componente no tiene interfaz, excepto la de prueba, y representa un diseño sencillo, amigable y entendible.

Para ver y procesar las imágenes de las huellas dactilares se debe dar clic en los botones Anterior o Siguiente, como resultado, se obtendrá a la izquierda la imagen de la huella dactilar original y a la derecha la plantilla biométrica correspondiente; conjuntamente se muestra un mensaje informando el tiempo que se demora en procesar la imagen. Para una muestra de 900 imágenes el tiempo promedio de procesamiento es de 149 milisegundos. Ver **Anexo_Vista inicial**.

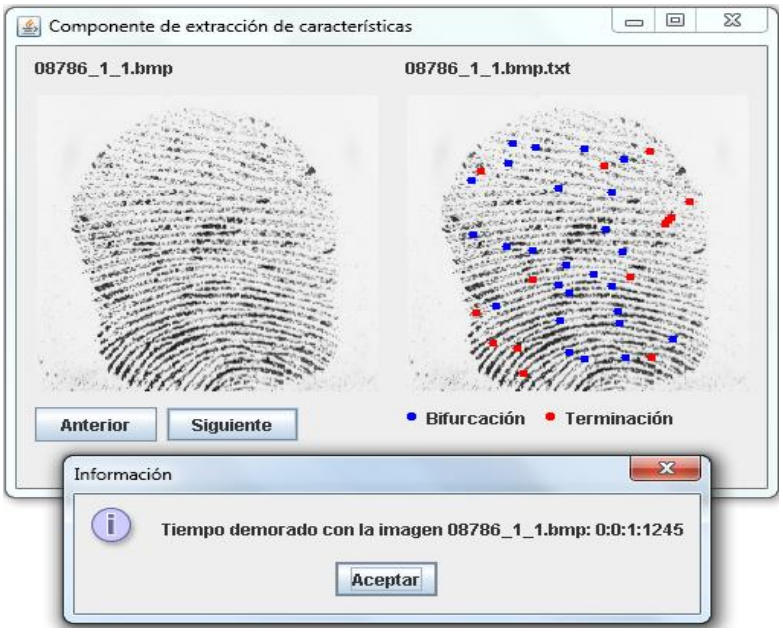


**Figura 14: Vista que se obtiene al ejecutar el proceso de extracción de características**

## 3.6 Pruebas

Uno de los pilares fundamentales de la metodología XP lo constituye el proceso de pruebas, que impulsa a los desarrolladores a probar constantemente. Mediante esta filosofía se minimiza la cantidad de errores no detectados y también el tiempo transcurrido entre la introducción de éste en la aplicación y su detección. Todo esto contribuye a elevar la calidad del producto y a la seguridad de los desarrolladores a la hora de realizar cambios. Las pruebas de software constituyen una actividad en la cual un sistema o componente es ejecutado bajo condiciones específicas, donde se observan o almacenan los resultados y se ejecuta una evaluación de algún aspecto del sistema o componente. Además, hacen la excelencia del desempeño de un programa y se estima que representan más de la mitad del costo de desarrollo. [41]

La metodología XP divide las pruebas en dos grupos:

- Pruebas unitarias: son pruebas desarrolladas por los programadores durante todo el ciclo de implementación y su función es verificar el código de forma automática.
- Pruebas de aceptación: están destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida, además de comprobar que dicha funcionalidad sea la esperada por el cliente.

En la evaluación del componente se aplicaron las pruebas antes descritas, las unitarias para asegurar el correcto funcionamiento de cada clase y las pruebas de aceptación, evidenciadas mediante la elaboración de un caso de prueba por cada requisito funcional identificado.

A continuación se muestra el caso de prueba de aceptación correspondiente a la prueba realizada al culminar la primera iteración planificada, y se asocia con la HU Realizar pre-procesamiento de la imagen de la huella dactilar. Para ver el resto siga este enlace **Anexo_Casos de prueba de aceptación**.

| **Caso de prueba de aceptación** | |
|---|---|
| **Código de caso de prueba: 1** | **Nombre de la historia de usuario:** Realizar pre-procesamiento de la imagen de la huella dactilar |
| **Responsable de la prueba:** Leandro Fortún Luna | |
| **Descripción de la prueba:** prueba para verificar que el componente realiza el pre-procesamiento de la imagen de la huella dactilar | |
| **Condición de ejecución:** el proceso debe recibir una imagen de la huella dactilar en escala de grises y el DPI de la imagen | |
| **Entrada/Pasos de ejecución:** luego de cargarse la imagen de la huella dactilar se ejecutan automáticamente los siguientes procesos: segmentación, normalización, orientación, binarización y adelgazamiento | |
| **Resultado esperado:** se debe obtener la imagen de la huella dactilar adelgazada | |
| **Evaluación de la prueba:** prueba satisfactoria | |

**Tabla 10: Prueba de aceptación realizada a la HU_ Realizar pre-procesamiento de la imagen de la huella dactilar**

## 3.7 Resultados obtenidos

Con el objetivo de realizar las pruebas de aceptación al componente, se generaron 900 plantillas biométricas, tanto en la biblioteca de clases SourceAFIs en C# como en el componente desarrollado en Java. Para cotejar los resultados obtenidos se implementó una clase que permite almacenar información en un fichero, siendo comparados mediante el uso del plugin de comparación del editor de texto Notepad++.

Las pruebas realizadas a tres versiones funcionales del componente arrojaron los resultados que se muestran en la siguiente tabla.

| **Versiones** | **Total de plantillas biométricas generadas** | **Cantidad de plantillas con errores** | **Por ciento de error respecto al 100 %** |
|---|---|---|---|
| **Versión 1** | 900 | 66 | 7.3 % |
| **Versión 2** | 900 | 12 | 1.3 % |
| **Versión 3** | 900 | 0 | 0.0 % |

**Tabla 11: Resultados de las pruebas de aceptación realizadas al componente**

En la gráfica se muestran los resultados obtenidos en las pruebas de aceptación:

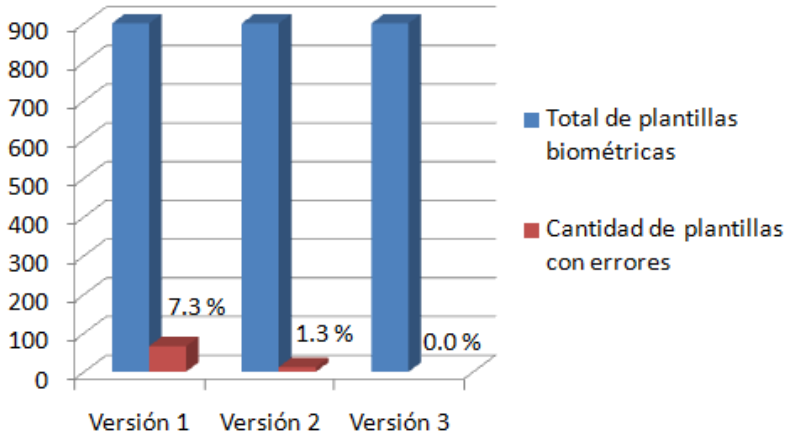


**Figura 15: Representación gráfica de los resultados finales de las pruebas**

Los problemas asociados a las plantillas con errores que se muestran en la tabla 11 son los siguientes:

| Versiones | No. | Tipo de error |
|---|---|---|
| **Versión 1** | 1 | Incorrecta implementación de la clase MinutiaCloudRemover, esta no eliminaba, de la plantilla biométrica, todas las minucias donde la cantidad de vecinos fueran mayores que el máximo de vecinos permitidos |
| | 2 | Imprecisiones del tipo de dato float en la etapa de post-procesamiento, debido a que en la clase "Calc.java" de la biblioteca SourceAFIS en C# se utiliza la clase System.Convert.ToInt32 y Java no cuenta con ella, lo que también influía en la dirección de algunas de las minucias de las 66 plantillas con errores |
| **Versión 2** | 2 | Continúa el error #2, aunque se reduce la cantidad de plantillas con errores de 66 a 12 |
| **Versión 3** | - | No se detectaron errores con la muestra analizada |

**Tabla 12: Representación de los tipos de errores detectados en las pruebas de aceptación**

## Conclusiones del capítulo 3

El análisis realizado a la versión alfa no funcional del SourceAFIS en Java permitió conocer el estado de implementación con el que contaba.

El establecimiento de un estándar de codificación permitió a los desarrolladores integrarse como un solo equipo, al trabajar sobre la misma base de codificación.

El desglose de las HU en tareas de ingeniería resultó ser una buena práctica, donde los programadores mostraron las funcionalidades específicas a implementar.

El desarrollo guiado por pruebas, aseguró la ejecución correcta de la solución durante todo el ciclo de desarrollo, igualmente las pruebas de aceptación concluyeron con resultados exitosos demostrando la satisfacción del cliente, elemento de vital importancia.

# Conclusiones

- El estudio realizado a sistemas que realizan el proceso de extracción demostró que, aunque son sistemas reconocidos, constituyen herramientas propietarias que dificultan la integración con el sistema de verificación dactilar.
- El desarrollo del componente para la extracción de características en huellas dactilares teniendo en cuenta las tecnologías, metodologías y herramientas analizadas, representa una solución con un alto nivel de independencia, que puede ser integrado a cualquier sistema que requiera del servicio que brinda. Además, puede ser ejecutado en un entorno multiplataforma, dando cumplimiento al objetivo planteado.
- Las pruebas realizadas al componente arrojaron los resultados esperados, por tanto, se logró realizar el proceso de extracción de características directamente desde el navegador, aprovechando así, los recursos disponibles del cliente.

# Recomendaciones

Aunque se cumplió el objetivo de la investigación, se recomienda implementar una solución para no utilizar el tipo de dato “Long”, ya sea a nivel de bits o con algún tipo de dato que simule el trabajo con enteros sin signo.

# Referencias bibliográficas

1. *Strategy for digital image segmentation of latent fingerprints.* **Ruíz, María E, Morales S, Miguel and Hernández M, Yahir.** 1, Ciudad Obregón, Sonora, México : Instituto Tecnológico de Sonora, 2011, Vol. 9. ISSN 1879-9532.

2. **Pelegrin Tumbeiro, Laura and Armando, Rodríguez Fernández Pedro.** *Componente para la reconstrucción de imágenes de huellas dactilares.* La Habana : Universidad de las Ciencias Informáticas, 2013.

3. **Bohigas, Xavier and Montse Novell, Jaén.** *Applets en la enseñanza de la física.* España : Universitat Politècnica de Catalunya, 2003, 2003.

4. **Microsoft.** Centro de seguridad de Microsoft. [Online] [Cited: 11 5, 2012.] http://www.microsoft.com/es-es/security/resources/activex-whatis.aspx.

5. **World Wide Web Consortium.** W3C. [Online] [Cited: 12 10, 2012.] http://www.w3.org.

6. **International Organization forStandardization.** *SO/IEC 19794-2:2011 - Information technology -- Biometric data interchange formats -- Part 2: Finger minutiae data.* s.l. : ISO/IEC, 2011.

7. **Márquez Álvarez, Yaimara and Salazar Reyes, Licel.** *Propuesta de un filtro para la eliminación del ruido causado por la iluminación en el sistema de visión de un robot móvil.* La Habana : Universidad de las Ciencias Informáticas, 2009.

8. **Artola Santiago, Nancy Roxana.** *La dactiloscopia como prueba eficaz dentro del proceso penal Guatemalteco, para la identificacion de personas que intervienenen un hecho delictivo.* Guatemala : Universidad de San Carlos de Guatemala, Facultad de ciencias jurídicas y sociales, 2009.

9. **Campo Buzón, José.** *Evaluación de conformidad con ISO/IEC 19784-1 para algoritmos de identificación biométrica.* Madrid : niversidad Carlos III de Madrid. Departamento de Tecnología Electrónica, 2009.

10. *Fingerprint Recognition Using Local Features.* **Aguilar, Gualberto, et al.** 46, México : Revista de la facultad de Ingeniería de la Universidad de Antioquía, 2008.

11. **Feng, Jianjiang and K. Jain, Anil.** *FM Model Based Fingerprint Reconstruction from Minutiae Template.* Michigan State : Departmentof Coputer Science and Engineering Michigan State University, 2009.

12. **Megha Kulshrestha, Pooja and K Banga, V.** *Selection of an Optimal Algorithm for Fingerprint Matching.* s.l. : World Academy of Science, Engineering and Technology, 2011.

13. *Age Classification System Anchored in Fingerprint Minutiae Extraction.* **Sudha Ponnarasi, S and Rajaram, M.** 2, India : European Journal of Scientific Research, 2012, Vol. 73. ISSN 1450-216X.

14. *A Multi-Level Model for Fingerprint Image Enhancement.* **G Babatunde, Iwasokun, et al.** Hong Kong : IMECS, 2012, Vol. 1. ISSN 978-988-19251-1-4.

15. *Modificación del histograma de una imagen.* **Felip León, Xavier, et al.** s.l. : Revista de los Estudios de Informática de la UJI, 2005.

16. *Estimating the Impact of Fingerprint Image Enhancement Algorithms for Better Minutia Detection.* **Rajinikannan, M, Ashok Kumar, D and Muthuraj, R.** 1, s.l. : International Journal of Computer Applications, 2010, Vol. 2.

17. *Fingerprint Enhancement using Improved Orientation and Circular Gabor Filter.* **Pokhriyal, Avinash, Sengar, Praveen and Lehri, Sushma.** 2, India : International Journal of Computer Applications, 2012, Vol. 43.

18. *Reliable Fingerprint Feature Extraction Algorithm.* **Karuna kumar, B and Satya Prasad, K.** 1, India : ICGST-DSP Journal, 2010, Vol. 10.

19. *Estimating the Impact of Minutia Extraction Algorithms in Fingerprint Recognition.* **Sudha Ponnarasi, S and Rajaram, M.** 6, India : International Journal of Research and Reviews in Computer Science, 2011, Vol. 2. ISSN: 2079-2557.

20. *Finger Print Analysis Using Termination and Bifurcation Minutiae.* **Awasthi, Vipul, Awasthi, Vanchha and Kumar Tiwari, Krishna.** 2, India : International Journal of Emerging Technology and Advanced Engineering, 2012, Vol. 2. ISSN 2250-2459.

21. **Vazan, Robert.** SourceForge. [Online] [Cited: Febrero 19, 2013.] http://sourceforge.net/projects/sourceafis/.

22. **NEUROTECHNOLOGY.** *VeriFinger SDK.* Lithuania : Neurotechnology, 2012.

23. **DATYS Tecnologías y Sistemas.** *BIOMESYS Control de Asistencia.Epecifaciones Técnicas.* La Habana : DATYS Tecnologías y Sistemas, 2012.

24. **DATYS Tecnologías y Sistemas.** *Especificaciones Técnicas del BIOMESYS AFIS Civil.* La Habana : DATYS Tecnologías y Sistemas, 2010.

25. **Álvarez Rey, Juan Manuel and Arzola Febles, Mairobys.** *Módulo de procesamiento de datos mediante algoritmos de paralelización para Airesweb.* La Habana : Universidad de las Ciencias Informáticas, 2011.

26. **Carrillo Pérez, Isaías, Pérez González, Rodrigo and Rodríguez Martín, Aureliano David.** M*etodologia de desarrollo del software.* 2008.

27. **Mousqués, Gastón.** *Metodología SCRUM.* Uruguay : Cátedra de Ingeniería de Software de Universidad ORT de Uruguay, 2003.

28. **Kniberg, Henrik.** *Scrum y XP desde las trincheras.* Estados Unidos : C4Media, 2007. ISBN: 978-1-4303-2264-1.

29. **Martínez Ladrón de Guevara, Jorge.** *Fundamentos de programación en Java.* s.l. : EME. ISBN 978-84-96285-36-2.

30. **Chapela Martíınez, Jairo.** *Introducción al entorno de desarrollo Eclipse.* 2007.

31. **Padrón Puyol, Jesús and Bolaños Solana, Daniel Alejandro.** *Plugin para el desarrollo de aplicaciones T-arenal.* La Habana : Universidad de las Ciencias Informáticas, 2010.

32. **Salazar Ramírez, Jose Ramón.** *Herramienta para el modelado y configuración de modelos de características.* Málaga : Escuela técnica superior de ingeniería informática, 2009.

33. **Cueva Lovelle, Juan Manuel.** *Introducción a UML.* España : Departamento de Informática Universidad de Oviedo, 1999.

34. **Cisnero Almésiga, Pavel Armando.** *Propuesta de un cliente web para el motor de base de datos apache derby.* Holguín : Universidad “Oscar Lucero Moya”, 2009.

35. **CAMACHO, ERIKA, CARDESO, FABIO and NUÑEZ, GABRIEL.** *Arquitecturas de software.* 2004.

36. **Allen, Robert and Garlan, David.** *“A formal approach to software architectures”.* s.l. : Proceedings of the IFIP Congress ’92, 1992.

37. **S Pressman, Roger.** *Software engineering practitioner's approach.* New York : The McGraw-Hill Companies, 2010. ISBN 978-0-07-337597-7.

38. **Javier Martínez, Juan Francisco.** *Guía de construcción de software en Java con patrones de diseño.* s.l. : Escuela universitaria de ingeniería técnica en informática de OVIEDO.

39. **Echeverry Tobóm, Luis Miguel and Delgado Carmona, Luz Elena.** *Caso práctico de la metodología ágil XP al desarrollo de software.* Pereira : Facultad de Ingeniería de la Universidad tecnológica de Pereira, 2007.

40. **González Jarros, Anet and Barnet Díaz, Eider.** *Componente para la extracción de características en imágenes faciales.* Ciudad Habana : Universidad de las Ciencias Informáticas, 2012.

41. **G F, Escribano.** *Extreme Programing.* London : Springer-Verlag, 2002. ISSN 3-540-44024-0.

# Bibliografía


I. **Hernández León, Rolando Alfredo y Coello González, Sayda.** *El proceso de investigación científica.* Ciudad de La Habana : Editorial Universitaria del Ministerio de Educación Superior, 2011. ISBN 978-959-16-1307-3.

II. *Reconocimiento de Huellas Dactilares Usando Características Locales.* **Aguilar, Gualberto, y otros.** México DF : Revista Facultad de Ingeniería de la Universidad de Antioquía, 2008, Vol. 46.

III. **Molpeceres, Alberto.** *Conveciones de código para el lenguaje de programación JavaTM.* 2001.

IV. **Posada Gómez, Oscar y Vargas Peña, Perfecto.** *Algoritmo de Extracción de minucias para identificación de personas por medio de la huella dactilar.* s.l. : Universidad Autónoma Metropolitana, 2005.

V. *Local Features for Enhancement and Minutiae Extraction in Fingerprints.* **Fronthaler, Hartwig, Kollreider, Klaus y Bigun, Josef.** 3, s.l. : IEEE TRANSACTIONS ON IMAGE PROCESSING, 2008, Vol. 17.

VI. *Design of a Wireless Fingerprint Authentication Service Platform Using the SOPC Technology.* **Liang, Xiaoying.** 8, s.l. : Journal of Convergence Information Technology, 2012, Vol. 7.

VII. *A New Approach To Fingerprint Recognition.* **Panda, Ipsha, y otros.** 5, s.l. : International Journal on Computer Science and Engineering, 2012, Vol. 4. ISSN : 0975-3397.

VIII. *Fingerprint Verification System using Minutiae Extraction Technique.* **Kaur, Manvjeet, y otros.** s.l. : World Academy of Science, Engineering and Technology, 2008, Vol. 46.

IX. *A Mathematical Modeling Method for Fingerprint Ridge Segmentation and Normalization.* **Babatunde, Iwasokun Gabriel, Oluwole Charles, Akinyokun y Olatunbosun, Olabode.** 2, Nigeria : International Journal of Computer Science and Information Technology & Security, 2012, Vol. 2. ISSN: 2249-9555.

X. *A Novel Approach for Human Identification through Fingerprints.* **Ramaswamy, G, y otros.** 3, s.l. : International Journal of Computer Applications, 2010, Vol. 4. ISSN 0975 – 8887.

XI. *Minutiae Extraction Based on Propriety of Curvature.* **Khalid, Abbad, Hamid, Tairi y Abdellah, Aarab.** 3, MORROCO : International Journal of Computer Theory and Engineering, 2011, Vol. 3.

XII. *Fingerprint Matching Using Two Methods.* **Patel, Hemlata y Asrodia, Pallavi.** 3, India : International Journal of Engineering Research and Applications, 2012, Vol. 2. ISSN: 2248-9622.

XIII. *Fingerprint Image Enhancement and its Feature Extraction for Recognition.* **Bhowmik, Pankaj, y otros.** 5, s.l. : INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH,, 2012, Vol. 1. ISSN 2277-8616.

XIV. *Singular points detection using fingerprint orientation field reliability.* **Sayim Khalil, Mohammed, y otros.** 4, Saudi Arabia : International Journal of Physical Sciences, 2010, Vol. 5. ISSN 1992 - 1950.

XV. *A Fingerprint and Palmprint Recognition Approach Based on Multiple Feature Extraction.* **Gayathri, R y Ramamoorthy, P.** 4, India : European Journal of Scientific Research, 2012, Vol. 76. ISSN 1450-216X.

XVI. *Cross-matching Algorithm Study of Fingerprint Images Based on Multi-type Fingerprint Sensors.* **WANG, Feng, y otros.** 9, China : Journal of Computational Information Systems, 2012, Vol. 8. ISSN 3613–3621.

XVII. *Fingerprint Features Extraction and matching.* s.l. : International Journal of Computer Applications, 2012. ISSN: 0975 - 8887.

XVIII. *Fingerprint Image Enhancement using Wavelet over Gabor filters.* 3, s.l. : Internacional Journal Computer Technology & Applications, 2012, Vol. 3. ISSN:2229-6093.

XIX. *Automated Fingerprint Recognition Using the DECOC Classifier.* **Ben Ayed, Mossaad, Bouchhima, Faouzi y Abid, Mohamed.** s.l. : International Journal of Computer Information Systems and Industrial Management Applications, 2012, Vol. 4. ISSN 2150-7988.

XX. *High-Resolution Fingerprint Matching using Level 3 Incipient Ridges and Scars.* **Latha Jothi, V y Arumugam, S.** 8, India : International Journal of Computer Applications, 2012, Vol. 48. ISSN 0975 – 888.

XXI. *Singularity Points Detection in Fingerprint Images.* **P Chaudhari, Jitendra, M Patil, Pradeep y P Kosta, Y.** 5, India : International Journal of Computer Applications, 2012, Vol. 4. ISSN 0975 – 8887.

XXII. *SVM-BDT Based Intelligent Fingerprint Authentication System using Geometry Approach.* 6, s.l. : International Journal of Advanced Research in Computer Science and Software Engineering, 2012, Vol. 2. ISSN: 0975-8283.

XXIII. *Increasing The Accuracy Of An Existing Fingerprint Recognition System Using Adaptive Technique.* 6, s.l. : International Journal of Advanced Research in Computer Science and Software Engineering, 2012, Vol. 2. ISSN: 2277 128X.

# Glosario de términos

*A*

*AFIS (Automated Fingerprint Identification System): sistema de identificación automatizado de huellas dactilares que compara un registro de huellas dactilares con los registros de una base de datos para determinar la identidad de un individuo.*

*B*

*API (Programación de Interfaces de Aplicación): conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones.*

*C*

*CISED (Centro de Identificación y Seguridad Digital): centro de desarrollo de soluciones integrales, productos y servicios en el campo de la identificación y la seguridad digital. Se encuentra estrechamente vinculado a la empresa comercializadora Albet SA y a la Universidad de las Ciencias Informáticas, lo que le ofrece una amplia posibilidad a su capital humano para la investigación, desarrollo e innovación. Cuenta además, con un equipo de especialistas de alto nivel, con gran experiencia y reconocimiento a nivel nacional e internacional.*

*I*

*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.*

*M*

*MINEX (Minutiae Interoperability Exchange Test): el propósito de la prueba de intercambio de interoperabilidad de las minucias es determinar la viabilidad de la utilización de datos de minucias como medio de intercambio de información de la huella dactilar entre diferentes sistemas de huellas dactilares coincidentes.*

*N*

*NIST (Instituto Nacional de Estándares y Tecnología): agencia fundada en 1901 para promocionar la innovación y competitividad industrial de los Estados Unidos. Entre sus labores, está establecer patrones de medida y estándares de tecnologías*

*NIST MINEX: es una evaluación rigurosa de algoritmos biométricos dactilares, cuyo propósito es proveer medidas sobre el rendimiento e interoperabilidad en la extracción y comparación de la información dactilar, a través de los algoritmos biométricos automáticos; establecer estándares de codificación y comparación en el marco del programa de control biométrico gubernamental y comprobar que la tecnología biométrica sea realmente confiable para la verificación de identidad de personas.*

*T*

*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.*

*W*

*WSQ (algoritmo de cuantificación escalar Wavelet): es un algoritmo de compresión utilizado para imágenes de huellas dactilares en escala de grises. Se basa en la teoría de la onda y se ha convertido en un estándar para el intercambio y almacenamiento de las imágenes de las huellas dactilares.*

# Anexos

## Anexo_Clasificación de la huella dactilar

**Figura 16: Huella dactilar de tipo arco, arco tendido, lazo derecho, lazo izquierdo y espiral**

## Anexo_Vista inicial del componente

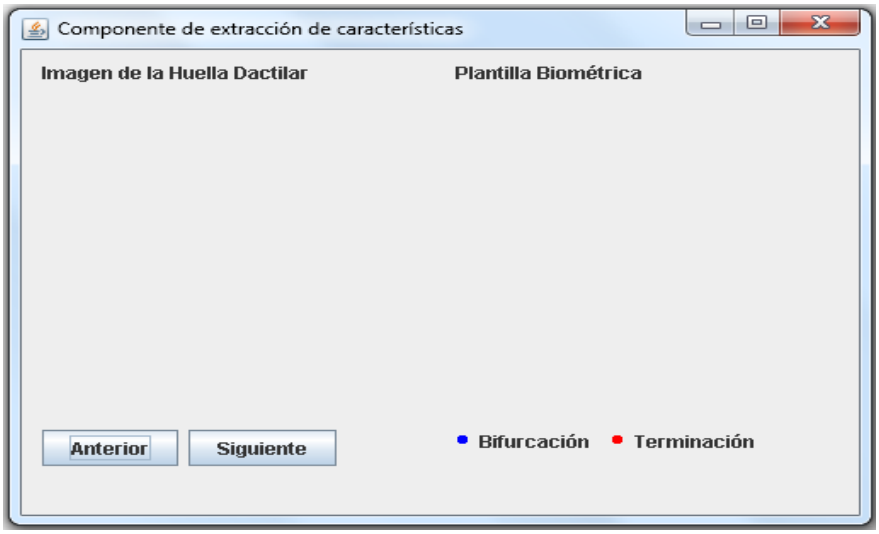


**Figura 17: Vista inicial al ejecutar el componente**

## Anexo_Historias de usuario

| **Historia de usuario** | |
|---|---|
| **Número:** 2 | **Usuario:** Sistema |
| **Nombre historia:** Realizar extracción de características de la imagen de la huella dactilar | |
| **Prioridad en negocio:** alta | **Riesgo en desarrollo:** alto |
| **Puntos estimados:** 4 | **Iteración asignada:** 2 |
| **Programador responsable:** Leandro Fortún Luna | |
| **Descripción:** este proceso consiste en extraer todas las minucias de la huella dactilar | |

| |
|---|
| **Observaciones:** para realizar este proceso son de vital importancia las mejoras realizadas a la imagen de la huella dactilar en la etapa de pre-procesamiento. |
| **Prototipo de interfaces:** sin prototipo de interfaz |

**Tabla 13: HU_ Realizar extracción de características de la imagen de la huella dactilar**

| **Historia de usuario** | |
|---|---|
| **Número:** 3 | **Usuario:** Sistema |
| **Nombre historia:** Realizar post-procesamiento de la imagen de la huella dactilar | |
| **Prioridad en negocio:** alta | **Riesgo en desarrollo:** alto |
| **Puntos estimados:** 4 | **Iteración asignada:** 3 |
| **Programador responsable:** Edelsys Vivian Carrazana Paneque | |
| **Descripción:** en esta etapa se validan las minucias, quedando sólo aquellas que aportan información útil para la etapa de comparación | |
| **Observaciones:** este proceso debe recibir el esqueleto de la imagen de la huella dactilar | |
| **Prototipo de interfaces:** sin prototipo de interfaz | |

**Tabla 14: HU_ Realizar post-procesamiento de la imagen de la huella dactilar**

| **Historia de usuario** | |
|---|---|
| **Número:** 4 | **Usuario:** Sistema |
| **Nombre historia:** Generar plantilla biométrica de la huella dactilar | |
| **Prioridad en negocio:** alta | **Riesgo en desarrollo:** alto |
| **Puntos estimados:** 3 | **Iteración asignada:** 4 |
| **Programador responsable:** Leandro Fortún Luna | |
| **Descripción:** en esta etapa son almacenadas las minucias determinadas como no espurias en una plantilla biométrica con formato ISO/IEC 19794-2:2011 | |
| **Observaciones:** este proceso debe recibir el esqueleto de la imagen de la huella dactilar | |
| **Prototipo de interfaces:** prototipo de interfaz | |

**Tabla 15: HU_ Generar plantilla biométrica de la huella dactilar**

# Anexo_Diagramas de clases

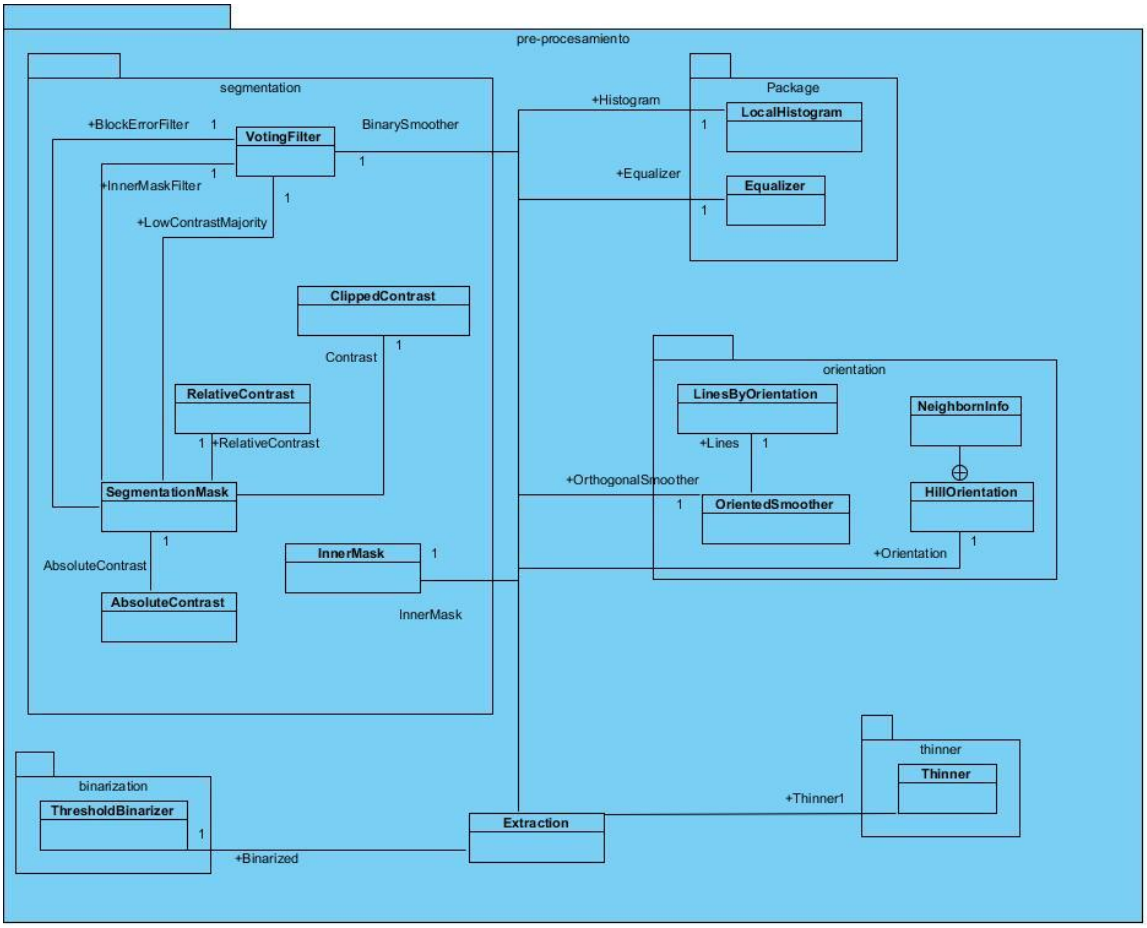


**Figura 18: Diagrama de clases asociado a la etapa de pre-procesamiento de la huella dactilar**

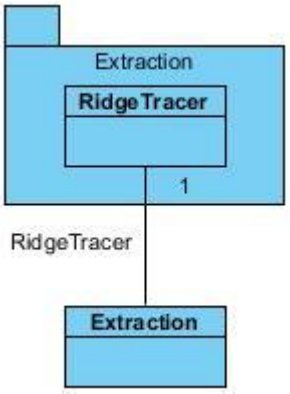


**Figura 19: Diagrama de clases asociado a la etapa de extracción de características**

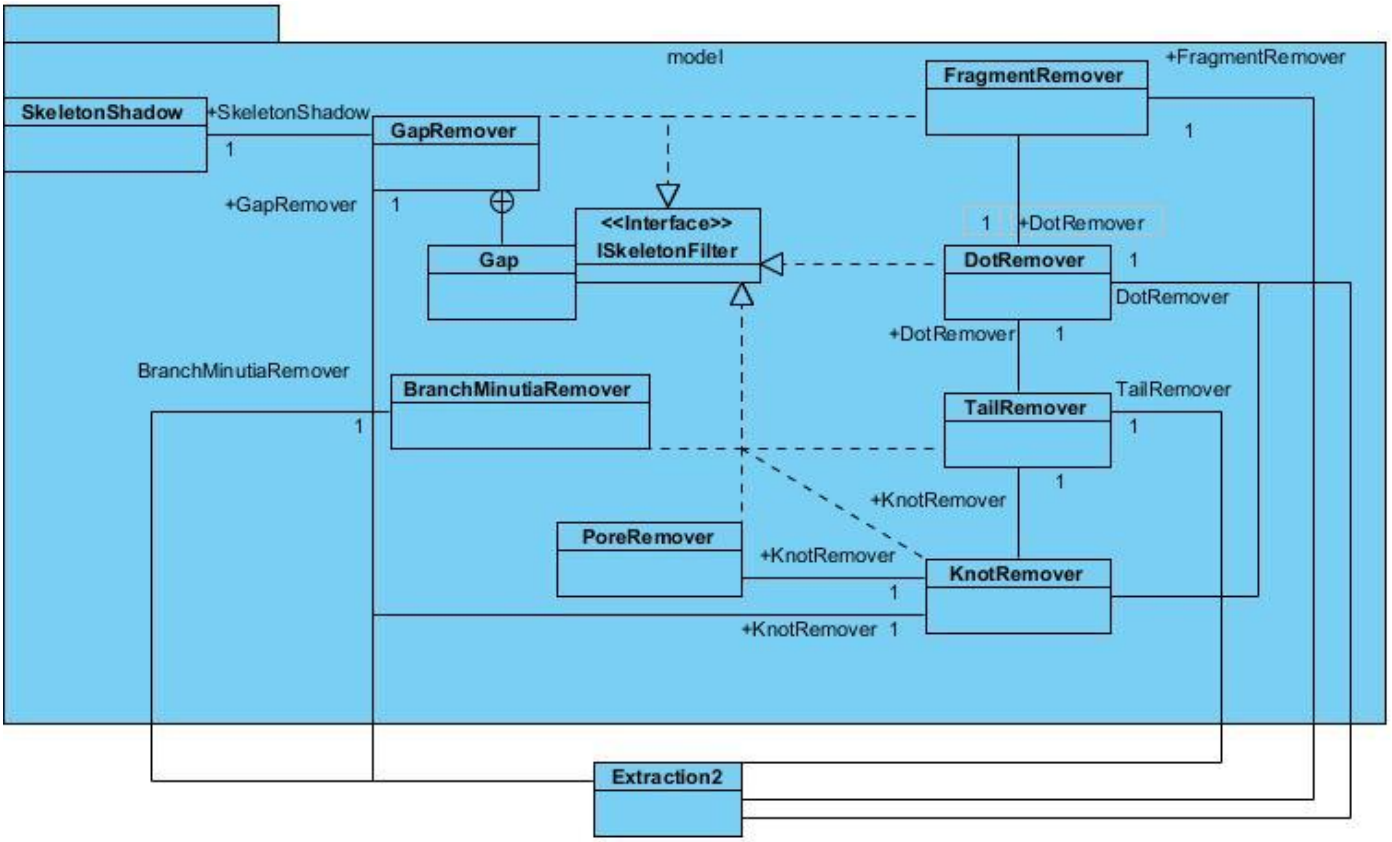


**Figura 20: Diagrama de clases asociado a la etapa de post-procesamiento de la huella dactilar**

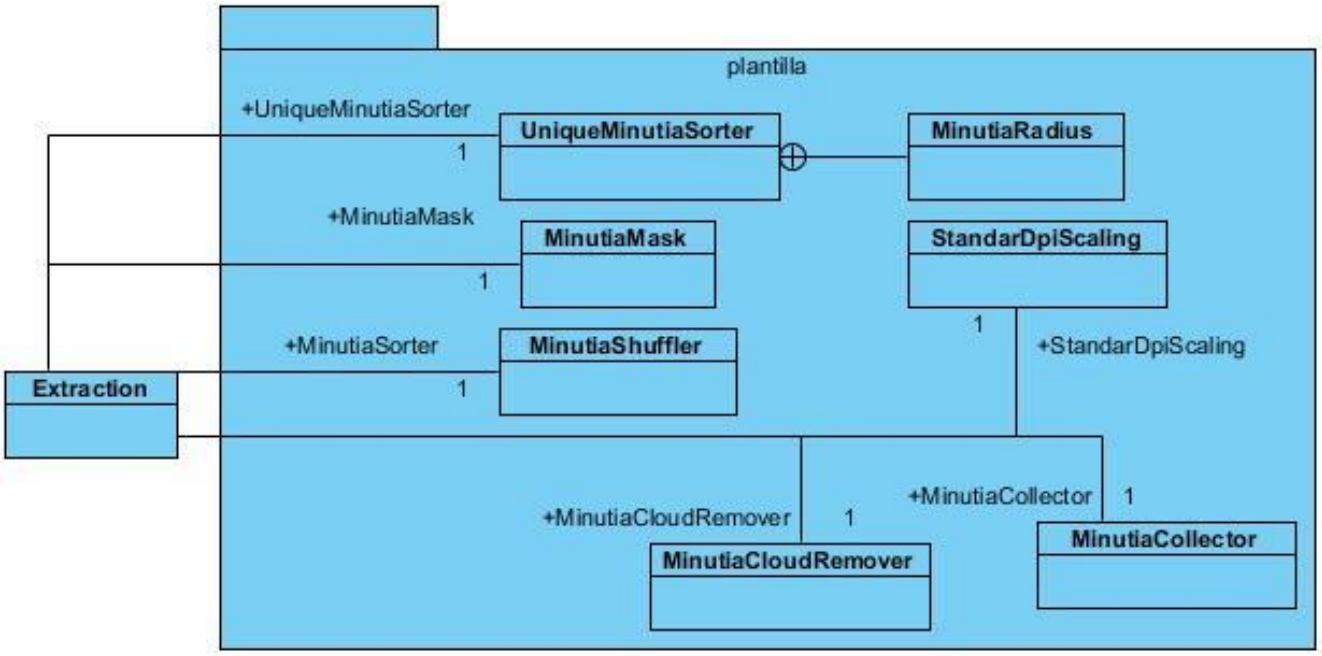


**Figura 21: Diagrama de clases asociado a la etapa de generación de la plantilla biométrica de la huella dactilar**

# Anexo_Tarjetas CRC

| Clase: DpiAdjuster.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Inicializar y ajustar los valores del DPI para cada clase | package meta<br>➢ Action.java<br>➢ DpiAdjusted.java<br>➢ ObjectTree.java<br>➢ ParameterValue.java |

**Tabla 16: Tarjeta CRC asociada a la clase DpiAdjuster**

| Clase: LocalHistogram.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Calcular las frecuencias relativas de los niveles de gris en la imagen | package extraction.filters<br>➢ BlockMap.java<br>➢ Point.java<br>➢ RectangleC.java<br>➢ Parallel.java<br>➢ ParallelForEachDelegate.java<br>➢ Calc.java |

**Tabla 17: Tarjeta CRC asociada a la clase LocalHistogram**

| Clase: SegmentationMask.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Separar la región del primer plano de la huella dactilar de la región del fondo | package extraction.filters<br>➢ BinaryMap.java<br>➢ BlockMap.java<br>package meta:<br>➢ Nested.java |

**Tabla 18: Tarjeta CRC asociada a la clase SegmentationMask**

| Clase: HillOrientation.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |

| | | |
|---|---|---|
| Calcular el ángulo de orientación de cada línea o cresta en la huella dactilar | package extraction.filters<br>➢ Angle.java<br>➢ BinaryMap.java<br>➢ BlockMap.java<br>➢ Calc.java<br>➢ Point.java<br>➢ PointF.java<br>➢ Range.java<br>➢ RectangleC.java | ➢ Parallel.java<br>➢ ParallelForDelegate.java<br>➢ ParallelForEachDelegate.java<br>package meta<br>➢ DpiAdjusted.java<br>➢ Parameter.java |

**Tabla 19: Tarjeta CRC asociada a la clase HillOrientation**

| **Clase: ThresholdBinarizer.java** | | |
|---|---|---|
| **Responsabilidad** | **Colaboran** | |
| Calcular el umbral de binarización<br>Convertir la imagen en escalas de grises en una imagen binaria | package extraction.filters<br>➢ Angle.java<br>➢ BinaryMap.java<br>➢ BlockMap.java<br>➢ Calc.java<br>➢ Point.java<br>➢ PointF.java<br>➢ Range.java<br>➢ RectangleC.java | ➢ Parallel.java<br>➢ ParallelForDelegate.java<br>➢ ParallelForEachDelegate.java<br>package meta<br>➢ DpiAdjusted.java<br>➢ Parameter.java |

**Tabla 20: Tarjeta CRC asociada a la clase ThresholdBinarizer**

| **Clase: Equalizer.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Ajustar los valores de niveles de gris de los píxeles de la imagen en busca de uniformidad | package extraction.filters<br>➢ BinaryMap.java<br>➢ BlockMap.java<br>➢ Calc.java<br>➢ Point.java<br>➢ PointF.java<br>➢ RectangleC.java<br>➢ Parallel.java<br>➢ ParallelForEachDelegate.java |

| | |
|---|---|
| | ➢ Parameter.java |

**Tabla 21: Tarjeta CRC asociada a la clase Equalizer**

| **Clase: OrientedSmoother.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Corregir el ángulo de orientación de cada cresta en la huella dactilar<br>. | package extraction.filters<br>➢ Angle.java<br>➢ BinaryMap.java<br>➢ BlockMap.java<br>➢ Calc.java<br>➢ Point.java<br>➢ RectanglC.java<br>➢ Parallel.java<br>➢ ParallelForEachDelegate.java<br>➢ Nested.java |

**Tabla 22: Tarjeta CRC asociada a la clase OrientedSmoother**

| **Clase: VotingFilter.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Mejorar la binarización | package extraction.filters<br>➢ BinaryMap.java<br>➢ Point.java<br>➢ RectangleC.java<br>➢ Size.java<br>➢ Parallel.java<br>➢ ParallelForDelegate.java<br>➢ Parameter.java |

**Tabla 23: Tarjeta CRC asociada a la clase VotingFilter**

| **Clase: CrossRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |

| | |
|---|---|
| Buscar y eliminar las minucias que se corresponden con el tipo de minucias cruces | package extraction.filters<br>➢ BinaryMap.java<br>➢ Point.java<br>➢ RectangleC.java |

**Tabla 24: Tarjeta CRC asociada a la clase CrossRemover**

| **Clase: InnerMask.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Delimitar el área de interés en la imagen de la huella dactilar | package extraction.filters<br>➢ BinaryMap.java<br>➢ Point.java<br>➢ RectangleC.java<br>➢ Parameter.java<br>➢ DpiAdjusted.java |

**Tabla 25: Tarjeta CRC asociada a la clase InnerMask**

| **Clase: MinutiaMask.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Eliminar las minucias que no se encuentran dentro del área de interés | package extraction.minutiae<br>➢ Angle.java<br>➢ BinaryMap.java<br>➢ Calc.java<br>➢ Point.java<br>➢ PointF.java<br>➢ Size.java<br>➢ DpiAdjusted.java<br>➢ Parameter.java<br>➢ Minutia.java<br>➢ TemplateBuilder.java |

**Tabla 26: Tarjeta CRC asociada a la clase MinutiaMask**

| **Clase: PoreRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |

| | |
|---|---|
| Buscar y eliminar lagunas en el esqueleto de la huella dactilar | package extraction.model<br>➢ Calc.java<br>➢ Point.java<br>➢ DpiAdjusted.java<br>➢ Nested.java<br>➢ Parameter.java |

**Tabla 27: Tarjeta CRC asociada a la clase PoreRemover**

| **Clase: GapRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Buscar y eliminar las crestas superpuestas que se encuentran en el esqueleto de la huella dactilar | package extraction.model<br>➢ Calc.java<br>➢ Point.java<br>➢ DpiAdjusted.java<br>➢ Nested.java<br>➢ Parameter.java<br>➢ Angle.java<br>➢ BinaryMap.java<br>➢ PriorityQueueF.java |

**Tabla 28: Tarjeta CRC asociada a la clase GapRemover**

| **Clase: TailRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Despegar del esqueleto aquellas crestas donde la cantidad de puntos es menor que el mínimo permitido | package extraction.model<br>➢ DpiAdjusted.java<br>➢ Nested.java<br>➢ Parameter.java |

**Tabla 29: Tarjeta CRC asociada a la clase TailRemover**

| **Clase: FragmentRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Eliminar pequeños fragmentos que no cumplen con | package extraction.model |

| | |
|---|---|
| la cantidad de puntos especificados | ➢ DpiAdjusted.java<br>➢ Nested.java<br>➢ Parameter.java |

**Tabla 30: Tarjeta CRC asociada a la clase FragmentRemover**

| **Clase: BranchMinutiaRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Eliminar del esqueleto las ramas que tienen más de dos crestas | package extraction.model |

**Tabla 31: Tarjeta CRC asociada a la clase BranchMinutiaRemover**

| **Clase: MinutiaCollector.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Almacenar para cada minucia la dirección, tipo y posición | package extraction.minutiae<br>➢ Ridge.java<br>➢ SkeletonBuilder.java<br>➢ SkeletonBuilderMinutia.java<br>➢ Angle.java<br>➢ DpiAdjusted.java<br>➢ Parameter.java<br>➢ Minutia.java<br>➢ MinutiaType.java<br>➢ TemplateBuilder.java |

**Tabla 32: Tarjeta CRC asociada a la clase MinutiaCollector**

| **Clase: MinutiaShuffler.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Mezclar todas las minucias y adicionarlas a la plantilla biométrica | package extraction.minutiae<br>➢ Calc.java<br>➢ Minutia.java<br>➢ TemplateBuilder.java |

**Tabla 33: Tarjeta CRC asociada a la clase MinutiaShuffler**

| **Clase: StandarDdpiScaling.java** |
|---|

| Responsabilidad | Colaboran |
|---|---|
| Calcular la posición de cada minucia almacenada en la plantilla biométrica | package extraction.minutiae<br>➢ Calc.java<br>➢ Minutia.java<br>➢ TemplateBuilder.java<br>➢ DpiAdjusted.java<br>➢ Parameter.java |

**Tabla 34: Tarjeta CRC asociada a la clase StandardDpiScaling**

| Clase: MinutiaCloudRemover.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Eliminar de la plantilla biométrica aquellas minucias donde la cantidad de vecinos es mayor que el máximo de vecinos especificados | package extraction.minutiae<br>➢ Calc.java<br>➢ Minutia.java<br>➢ TemplateBuilder.java<br>➢ Parameter.java |

**Tabla 35: Tarjeta CRC asociada a la clase MinutiaCloudRemover**

| Clase: UniqueMinutiaSorter.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Ordenar las minucias dentro de la plantilla biométrica en función de la cantidad de vecinos | package extraction.minutiae<br>➢ Calc.java<br>➢ Minutia.java<br>➢ TemplateBuilder.java<br>➢ Parameter.java |

**Tabla 36: Tarjeta CRC asociada a la clase UniqueMinutiaSorter**

| Clase: RidgeTracer.java | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Obtener las minucias asociadas a la huella dactilar | package extraction.model<br>➢ AssertException.java<br>➢ BinaryMap.java<br>➢ Calc.java<br>➢ Neighborhood.java |

| | |
|---|---|
| | ➢ Point.java |

**Tabla 37: Tarjeta CRC asociada a la clase RidgeTracer**

| **Clase: DotRemover.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Buscar las minucias que no pertenecen a ninguna cresta y eliminarlas | package extraction.model |

**Tabla 38: Tarjeta CRC asociada a la clase DotRemover**

| **Clase: MinutiaCollector.java** | |
|---|---|
| **Responsabilidad** | **Colaboran** |
| Almacenar en la plantilla biométrica la dirección, tipo y posición de las minucias obtenidas | package extraction.minutae<br>➢ Ridge.java<br>➢ SkeletonBuilder.java<br>➢ SkeletonBuilderMinutia.java<br>➢ Angle.java<br>➢ DpiAdjusted.java<br>➢ Parameter.java<br>➢ Minutia.java<br>➢ MinutiaType.java<br>➢ TemplateBuilder.java |

**Tabla 39: Tarjeta CRC asociada a la clase MinutiaCollector**

## Anexo_Casos de prueba de aceptación

| **Caso de prueba de aceptación** | |
|---|---|
| **Código de caso de prueba: 2** | **Nombre de la historia de usuario:** Realizar extracción de características de la huella dactilar |
| **Responsable de la prueba:** Leandro Fortún Luna | |
| **Descripción de la prueba:** prueba para verificar que el componente realiza el proceso de extracción de características de la huella dactilar | |
| **Condición de ejecución:** el proceso recibe una imagen de la huella dactilar adelgazada | |

| |
|---|
| **Entrada/Pasos de ejecución:** el proceso se realiza automáticamente después de haberse ejecutado la etapa de pre-procesamiento de la imagen de la huella dactilar |
| **Resultado esperado:** se deben obtener las minucias que caracterizan a la imagen de la huella dactilar procesada |
| **Evaluación de la prueba:** prueba satisfactoria |

**Tabla 40: Prueba de aceptación realizada a la HU_Realizar extracción de características de la huella dactilar**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código de caso de prueba: 3** | **Nombre de la historia de usuario:** Realizar post-procesamiento de la imagen de la huella dactilar |
| **Responsable de la prueba:** Edelsys Vivian Carrazana Paneque | |
| **Descripción de la prueba:** prueba para verificar que el componente elimina las falsas minucias extraídas de la imagen de la huella dactilar | |
| **Condición de ejecución:** el proceso recibe el esqueleto de la imagen de la huella dactilar | |
| **Entrada/Pasos de ejecución:** el proceso se realiza automáticamente después de haberse ejecutado el proceso de extracción de características de la huella dactilar | |
| **Resultado esperado:** se debe eliminar las falsas minucias que no se corresponden con terminaciones y bifurcaciones en la huella dactilar | |
| **Evaluación de la prueba:** prueba satisfactoria | |

**Tabla 41: Prueba de aceptación realizada a la HU_ Realizar post-procesamiento de la imagen de la huella dactilar**

| **Caso de prueba de aceptación** | |
|---|---|
| **Código de caso de prueba: 4** | **Nombre de la historia de usuario:** Generar plantilla biométrica de la huella dactilar |
| **Responsable de la prueba:** Leandro Fortún Luna | |
| **Descripción de la prueba:** prueba para verificar que el componente genera la plantilla biométrica que caracteriza la huella dactilar procesada | |
| **Condición de ejecución:** el proceso recibe el esqueleto de la imagen de la huella dactilar | |
| **Entrada/Pasos de ejecución:** el proceso se realiza automáticamente después de haberse ejecutado el proceso de post-procesamiento de la imagen de la huella dactilar | |
| **Resultado esperado:** se debe generar la plantilla biométrica asociada a la huella dactilar procesada en formato ISO/IEC 19794-2:2011. | |

| **Evaluación de la prueba:** prueba satisfactoria |
|---|

**Tabla 42: Prueba de aceptación realizada a la HU_ Generar plantilla biométrica de la huella dactilar**

## Anexo_Complejidad ciclomática del código ejemplo

```
Range GetMaskLineRange (BinaryMap mask, int y) {
1.  int first = -1;
        int last = -1;
2.  for (int x = 0; x < mask.getWidth(); ++x) {
        a.  if (mask.GetBit(x, y)) {
        b.  last = x;
        c.  if (first < 0) {
        d.  first = x;
            }
3.  }
4.  }
5.  if (first >= 0) {
6.  return new Range (first, last + 1);
    } else {
7.  return new Range (0, 0);
8.  }
9.  }
```

CC = número de regiones cerradas en el grafo + 1
CC = 4+1
CC = 5

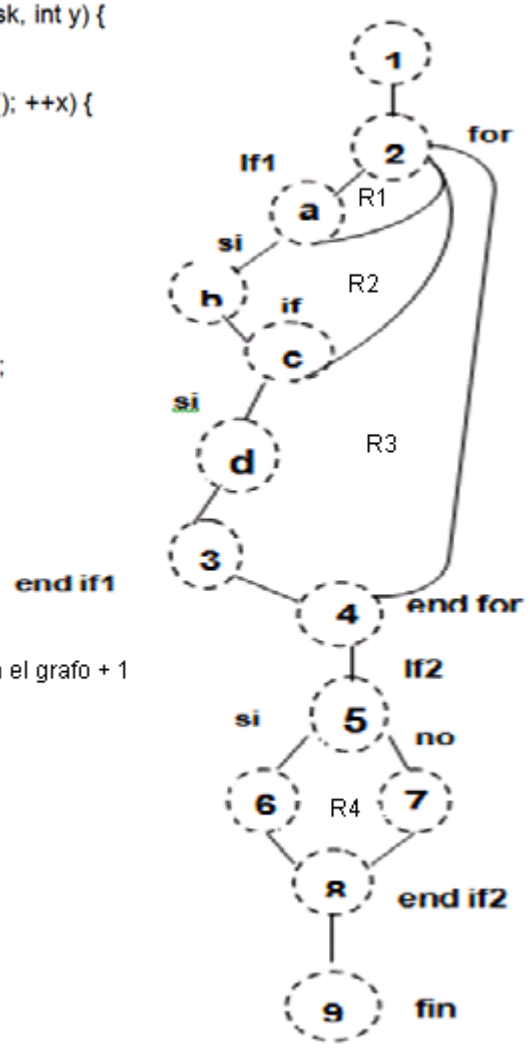


**Figura 22: Complejidad ciclomática del código ejemplo**