Universidad de las Ciencias Informáticas

Facultad 1

Título: *Componente para la reconstrucción de imágenes de huellas dactilares.*

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

Autor(es): *Laura Pelegrín Tumbeiro.*

Pedro Armando Rodríguez Fernández.

Tutor*: Ing. Ramón Santana Fernández.*

Co-Tutor*: Ing. Abel Ernesto Sánchez Alí.*

Declaración de autoría

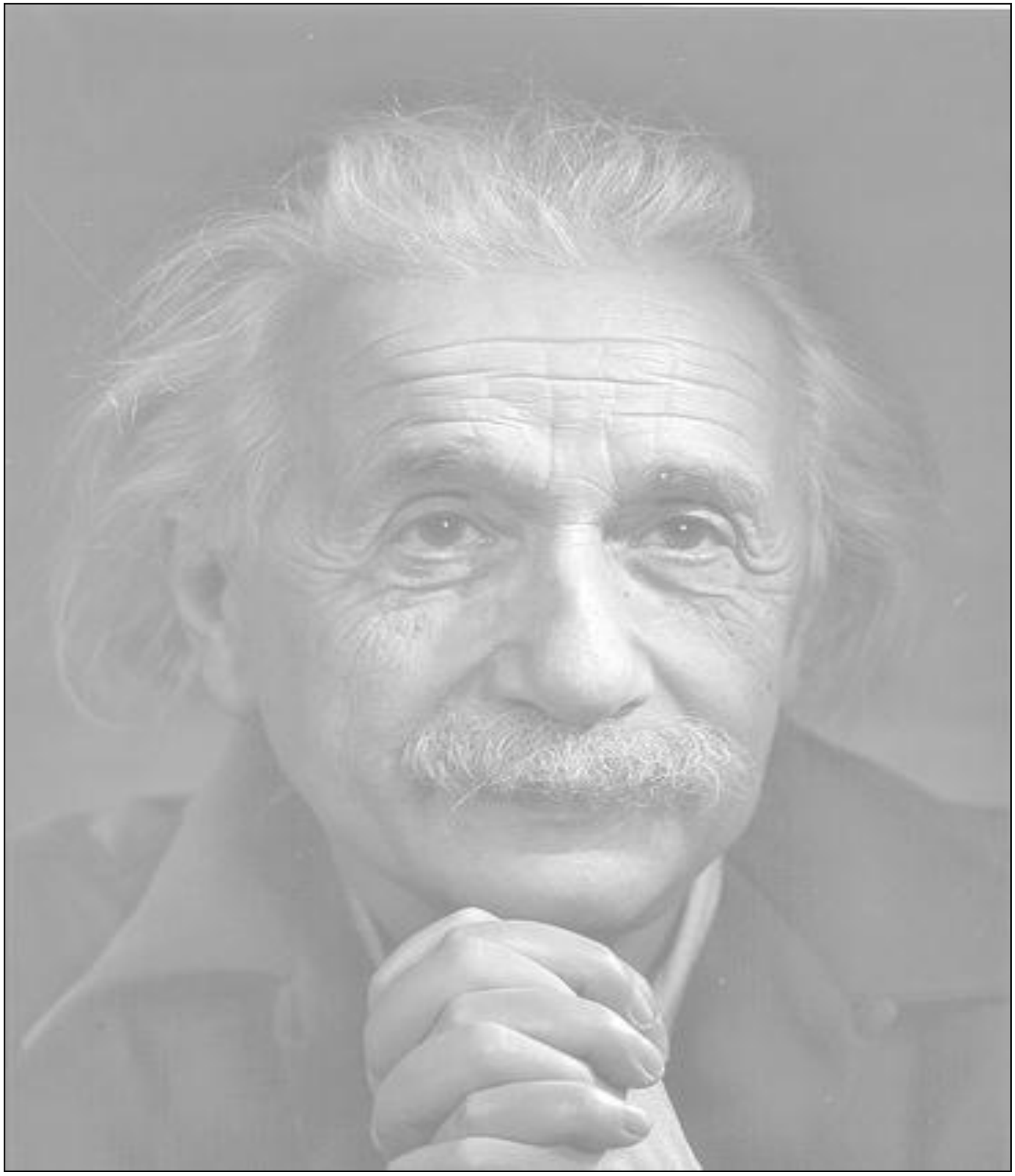
Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmamos la presente a los ____ días del mes de _____ del año 2012.

_____ _____

Autor: Laura Pelegrín Tumbeiro. **Autor:** Pedro Armando Rodríguez Fernández.

_____

Tutor: Ing. Ramón Santana Fernández.



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber"

Albert Einstein

Resumen

En el presente trabajo se desarrolla un componente de reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378, con el objetivo de obtener imágenes parciales de huellas reconstruidas que puedan ser utilizadas para realizar pruebas de seguridad de acceso a aplicaciones y detectar posibles vulnerabilidades en los sistemas automáticos de identificación por huella dactilar.

La reconstrucción del parcial de una imagen de huella dactilar se podrá realizar visualizando el proceso o de forma automática. La primera de ellas comienza por la selección de la plantilla de minucias, la formación de las tripletas, la estimación del campo de orientación y la reconstrucción de la imagen parcial de la huella dactilar. La segunda se inicia con la selección de las plantillas de minucias que se desean reconstruir y luego se procede a la generación automática, sin necesidad de observar el proceso de reconstrucción, todas las imágenes reconstruidas se almacenan en la dirección seleccionada por el usuario.

Con la realización de este componente se obtendrán un conjunto de datos de pruebas que podrán ser utilizados para burlar la seguridad de acceso de un sistema de reconocimiento de personas basados en huellas dactilares y su desarrollo contribuirá a mejorar la seguridad de los sistemas biométricos desarrollados en el departamento de Biometría del Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas.

Palabras claves: huella dactilar, minucia, plantilla de minucias, reconstrucción de huella.

Tabla de Contenidos

Índice de Figuras

Índice de tablas

Introducción.

Con el desarrollo de las tecnologías informáticas, se ha agilizado la automatización de los procesos de identificación de los individuos, y por tanto la Biometría se ha transformado en un área importante. El empleo de las técnicas biométricas en los sistemas informáticos, han permitido identificar y autenticar de forma segura a las personas, a partir de sus características fisiológicas o conductuales. Entre las características biométricas más comunes se encuentran el rostro, el iris, la geometría de la mano, la huella dactilar, la voz, la firma manuscrita y la manera de caminar.

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

La identificación de individuos mediante la huella dactilar se ha venido empleando desde finales del siglo XIX cuando Sir Francis Galton definió algunos de los puntos o características que podían ser identificadas a partir de una huella, conocidos también como minucias. A finales de la década de 1960, se utilizaron los puntos de Galton (minucias) para la automatización de las tecnologías de reconocimiento de huella dactilar. (2)

En 1969, el Buró Federal de Investigaciones (FBI), tuvo la necesidad de desarrollar un sistema que automatizara los procesos de identificación por huella dactilar, ya que el método que se empleaba era manual y se requería de muchas horas. Contrató al Instituto Nacional de Estándares y Tecnología (NIST) para estudiar todo el proceso de clasificación, búsqueda y comparación de las huellas dactilares.

Con el transcurso de los años los sistemas biométricos basados en huellas dactilares se han ido desarrollando, siendo utilizados en la actualidad en múltiples lugares para ampliar la seguridad y comodidad de la sociedad. Por ejemplo, son utilizados para el control de acceso a locaciones físicas o información lógica (cuentas personales en

computadoras, documentos electrónicos protegidos) y para aplicaciones de servicio social o de identificación nacional. A pesar de sus numerosas ventajas, los sistemas biométricos son vulnerables a ataques que pueden disminuir su seguridad.

Actualmente, el fraude de identidad es un problema ético que atenta contra la seguridad de los sistemas informáticos. Existe una serie de fuentes de ataques que a lo largo de los años han intentado burlar la seguridad de acceso de los sistemas biométricos.

En los últimos años se han realizado varias investigaciones sobre las vulnerabilidades de los sistemas biométricos basados en huellas dactilares frente a posibles ataques. Entre los que se encuentran: ataques directos dirigidos al sensor (llevados a cabo utilizando rasgos biométricos sintéticos como dedos de goma, moldes de plastilina, silicona, gelatina), y ataques indirectos (dirigidos contra alguno de los módulos internos del sistema, robo o modificación de la base de datos o plantilla).

En el proceso de captura de una huella dactilar, se extraen los puntos característicos de la imagen de la huella y la información biométrica extraída se almacena en plantillas biométricas. Varios desarrolladores afirman que de este modo se protegen los datos que podrían revelar la identidad de las personas, pero esta técnica puede ser burlada si se reconstruye un parcial de la huella dactilar a través de la plantilla de minucias. Esta imagen de la huella reconstruida puede ser similar a la imagen de la huella original y ser utilizada para acceder a sistemas de identificación basado en huellas dactilares.

Las pruebas de vulnerabilidad son una parte esencial para el buen funcionamiento de los sistemas biométricos, pues ofrecen información valiosa sobre el nivel de exposición que se tiene ante posibles amenazas.

En el departamento de Biometría del Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas se implementan algoritmos de extracción y macheo de características de huellas dactilares; también se desarrollan sistemas de control de acceso y sistemas de identificación a través de tarjetas inteligentes. Para el correcto funcionamiento de estos sistemas es necesario realizar pruebas con el objetivo de valorar la eficiencia de los mismos. Actualmente no se cuenta con una herramienta que brinde datos de pruebas (parciales de huellas reconstruidas), los cuales pueden ser

utilizados para realizar pruebas de acceso y detectar posibles vulnerabilidades en los sistemas de identificación dactilar desarrollados en el centro, constituyendo así la **situación problemática** de la investigación.

Una vez analizada la problemática anterior se formula el siguiente **problema de investigación:** ¿Cómo facilitar datos de prueba para detectar vulnerabilidades de acceso en los sistemas automáticos de identificación de huellas dactilares que almacenen minucias en estándar ANSI/INCITS 378?

Para dar solución al problema propuesto se define como **objeto de estudio** los procesos de reconstrucción de imágenes de huellas dactilares.

Se define como **objetivo general** desarrollar una aplicación de reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378, que permita almacenar imágenes parciales de huellas reconstruidas que puedan ser utilizadas para probar la seguridad de acceso de los sistemas de identificación basado en huellas dactilares desarrollados en el departamento de Biometría del Centro de Identificación y Seguridad Digital, de la Universidad de las Ciencias informáticas.

Se determina como **campo de acción** los procesos de reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378.

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

Tareas comunes:

- ✓ Análisis del estado del arte en Cuba y en el mundo referente a la reconstrucción de imágenes de huellas dactilares.
- ✓ Definición de las tecnologías, metodologías y herramientas a utilizar para el desarrollo del sistema.
- ✓ Definición de la arquitectura del software.
- ✓ Definición de los patrones del Diseño.
- ✓ Descripción de los algoritmos de reconstrucción de imágenes de huellas dactilares.

Tareas llevadas a cabo por Laura Pelegrín:

- ✓ Confección del modelo de dominio de la aplicación.
- ✓ Levantamiento de los requisitos funcionales y no funcionales de la aplicación.
- ✓ Análisis y Diseño de los Casos de Uso del sistema.
- ✓ Descripción de los Casos de Uso del sistema.
- ✓ Confección de los diagramas de clases del análisis.
- ✓ Confección de los diagramas de colaboración.
- ✓ Confección de los diagramas de secuencia.
- ✓ Confección del diagrama de clases del diseño de la aplicación.
- ✓ Confección del diagrama de componente de la aplicación.
- ✓ Validación de la aplicación.

Tareas llevadas a cabo por Pedro Armando Rodríguez:

- ✓ Análisis de los estándares para la formación y lectura de la plantilla de minucias.
- ✓ Implementación del algoritmo para la lectura de la plantilla de minucias.
- ✓ Análisis de las técnicas para la formación de tripletas de minucias.
- ✓ Implementación del algoritmo para la formación de las tripletas de minucias.
- ✓ Identificación de las técnicas para la estimación del campo de orientación de la huella dactilar.
- ✓ Implementación del algoritmo para la estimación del campo de orientación de la huella dactilar.
- ✓ Implementación del algoritmo para reconstruir un parcial de la imagen de la huella dactilar.
- ✓ Diseño de las interfaces gráficas de la aplicación.

Los **Métodos Científicos** empleados son:

- ❖ **Métodos Teóricos**.
  - ✓ **Analítico - Sintético:** Se utilizó en el análisis de las diferentes teorías de huellas dactilares, permitió descubrir las características generales de las mismas y establecer relaciones entre estas. Se utilizó además para extraer los elementos más importantes relacionados con el tema de reconstrucción de imágenes de huellas dactilares.

- ✓ **Análisis Histórico - Lógico:** Se utilizó en el estudio de los antecedentes, la evolución y el desarrollo que han tenido los sistemas biométricos de huellas dactilares, así como valorar las tendencias actuales de los algoritmos de reconstrucción de imágenes de huellas dactilares a través de la búsqueda de información en diferentes bibliografías.
- ✓ **Modelación:** Se utilizó en la modelación de los diagramas de clases del análisis y del diseño, lo que permitió un mejor entendimiento del sistema.

❖ **Métodos Empíricos**.

- ✓ **Método de Entrevista:** Se utilizó para realizar entrevistas a especialistas en huellas dactilares, con el objetivo de obtener información relacionada con el tema de reconstrucción de imágenes de huellas dactilares, precisar el problema a resolver y definir los procesos que se llevan a cabo.

Justificación de la investigación:

Los sistemas biométricos basados en huellas dactilares representan actualmente un área importante de investigación y desarrollo tecnológico, se emplean en varias aplicaciones y tienen gran aceptación en el mercado mundial.

A pesar de que existen en el mundo compañías expertas en desarrollo de aplicaciones biométricas con alta calidad, las herramientas biométricas implementadas son muy costosas y además existe poca documentación sobre los estudios relacionados con el tema de reconstrucción de imágenes de huellas dactilares. Cuba es un país subdesarrollado, por lo que es necesario contar con herramientas propias que realicen el proceso de reconstrucción de imágenes de huellas dactilares para desarrollar sistemas biométricos que cumplan con la seguridad y la calidad requerida.

Con este trabajo se pretende ofrecer el siguiente **aporte práctico.**

- Componente que permita reconstruir imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378, y de este modo obtener imágenes parciales de huellas reconstruidas que puedan ser utilizadas para probar la seguridad de acceso de los sistemas de identificación basados en huellas dactilares.

El presente trabajo de diploma consta de **3 capítulos estructurados** de la siguiente manera:

Capítulo 1: **Fundamentación Teórica**, se describen los principales conceptos abordados, se realiza un estudio del estado del arte del tema tratado a nivel internacional, nacional y en la Universidad y se hace un estudio de las principales tecnologías, metodologías y herramientas para darle solución al problema planteado.

Capítulo 2: **Características del Sistema**, se definen las características del sistema realizando un análisis del dominio de la aplicación, se modelan los diagramas de clases del análisis y del diseño. Se hace referencia a los patrones utilizados para la construcción del sistema y se propone la arquitectura del componente.

Capítulo 3: **Implementación y Prueba**, se tratan aspectos relacionados con la implementación de la solución propuesta, se modela el diagrama de componente y por último se realizan las pruebas del sistema, donde se valida el funcionamiento de los requisitos funcionales.

Capítulo 1: Fundamentación Teórica.

Introducción.

En este capítulo se describen los principales conceptos para comprender los procesos asociados al dominio del problema. Se realiza un estudio del estado del arte sobre las aplicaciones de reconstrucción de imágenes de huellas dactilares tratados a nivel internacional, nacional y en la Universidad. Se describen las principales tecnologías, metodologías y herramientas utilizadas en la actualidad, analizando sus características para seleccionar la más indicada en la implementación del componente a desarrollar.

1.1 Sistemas biométricos.

Son sistemas automatizados que permiten identificar y verificar la identidad de las personas basándose en algún rasgo físico medible o en algún patrón de comportamiento. Los sistemas biométricos se clasifican en fisiológicos y conductuales: (Ver figura 1).

- **Fisiológicos.**

- **Conductuales.**

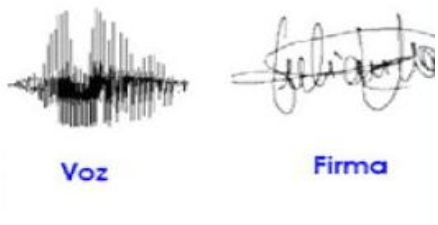


Figura 1: Clasificación de los sistemas biométricos.

1.2 Huellas dactilares.

Son patrones constituidos por las crestas papilares de los dedos de las manos que aparecen visibles en la epidermis, generando configuraciones diversas. (Ver figura 2). Las discontinuidades locales en el patrón del flujo de las crestas son conocidas como minucias. Las huellas dactilares 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. (3)

Figura 2: Huella dactilar.

1.3 Características de las huellas dactilares.

1.3.1 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). Una cresta está definida como un segmento de curva y un valle es la región entre dos crestas adyacentes. (Ver figura 3). (4)

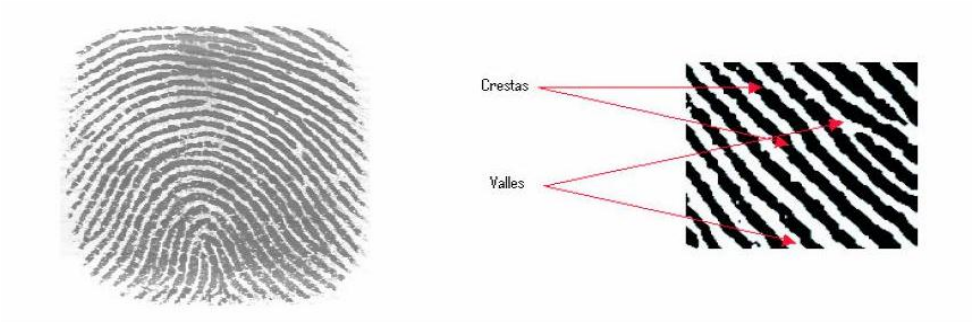


Figura 3: Crestas y Valles.

1.3.2 Minucias.

Las discontinuidades locales son conocidas como minucias. Son los puntos singulares encontrados en el trazo de las crestas. (Ver figura 4). En la identificación, las que tienen mayor valor significativo son las terminaciones y bifurcaciones. Cabe destacar que en una huella dactilar existen aproximadamente de 50 a 150 minucias. (1)

Figura 4: Minucias.

1.3.3 Núcleo y Delta.

El núcleo y el delta son singularidades de la huella. 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. (Ver figura 5). (1)

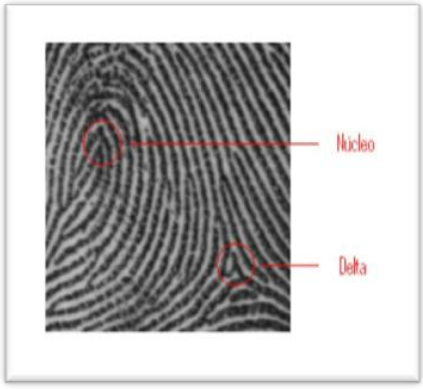


Figura 5: Núcleo y Delta.

1.3.4 Clasificación de las huellas dactilares.

Las huellas dactilares son únicas en cada individuo y estas se pueden clasificar en 5 tipos: Arco, Arco tendido, Lazo izquierdo, Lazo derecho y Espiral. (Ver figura 6).

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

- ✓ **Arco Tendido:** Semeja a un arco. Es un tipo de línea que rápidamente nace y se pierde en un paso de ángulo.

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

- ✓ **Lazo izquierdo:** Al igual que el lazo derecho, estas tienen un punto delta, pero éste se ubica del lado izquierdo de la huella. Las crestas papilares que forman el núcleo nacen a la derecha y su recorrido es a la izquierda para dar vuelta sobre sí mismas y regresar al mismo punto de partida.

- ✓ **Espiral (Anillo de cresta):** Los patrones en forma de anillos tienen muchos círculos que salen de un punto de la huella, las mismas cuentan con dos puntos deltas. (5)

Figura 6: Tipos de huellas dactilares.

1.4 Reconstrucción de imágenes de huellas dactilares.

Reconstruir una imagen de una huella dactilar, es el conjunto de procesos destinados a conectar las crestas de la huella a partir de las minucias, con el objetivo de obtener un parcial de la imagen de la huella dactilar.

1.4.1 Parcial dactilar.

El parcial es una parte de la huella dactilar que contiene menor información que la huella original.

1.4.2 Plantilla de minucias en estándar ANSI/INCITS 378.

Es una norma para formato de datos de plantillas de huellas dactilares. Este estándar define un método de representación de información de huellas dactilares usando el concepto de minucias. Define la ubicación de las minucias en una huella dactilar, la coordenada X, la coordenada Y, el ángulo de orientación de cada minucia, entre otros valores. (Ver tabla 1).

Tabla 1: Formato de datos del estándar ANSI/INCITS 378 para la representación de las minucias.

Campo	Tamaño
Tipo	**2 bits**
Posición X	**14 bits**
Reservado	**2 bits**
Posición Y	**14 bits**
Ángulo de orientación σ	**1 byte**
Calidad	**1 byte**

1.4.3 Tripletas.

Es un triángulo formado por la unión de tres puntos de minucias. Una tripleta válida tiene que cumplir una serie de restricciones relacionadas con los lados del triángulo, los ángulos interiores y las orientaciones referentes a cada punto de minucia.

1.5 Estado del arte.

1.5.1 Nivel Internacional.

En la actualidad en el ámbito internacional existen muchas empresas que desarrollan sistemas biométricos. Los principales vendedores en la industria de la tecnología biométrica de huellas dactilares son: SAGEM, Cogent, SecuGen, Identix y Biometric, los cuales desarrollan diferentes sistemas que comprenden los procesos de extracción y macheo de características de huellas dactilares, sin embargo no se encuentran evidencias de que realicen el proceso de reconstrucción dactilar. Existen otras instituciones que investigan y desarrollan algoritmos relacionados con las huellas dactilares.

BioLab es un laboratorio de la Universidad de Bologna donde se desarrollan sistemas biométricos. Los algoritmos propuestos por esta institución han sido reconocidos por su calidad y eficiencia. En el 2007 publicó el primer método eficaz capaz de reconstruir imágenes de huellas dactilares a partir de plantillas de minucias (Ver figura 7), pero hasta el momento no se conoce públicamente una aplicación que realice dicho proceso. (6)

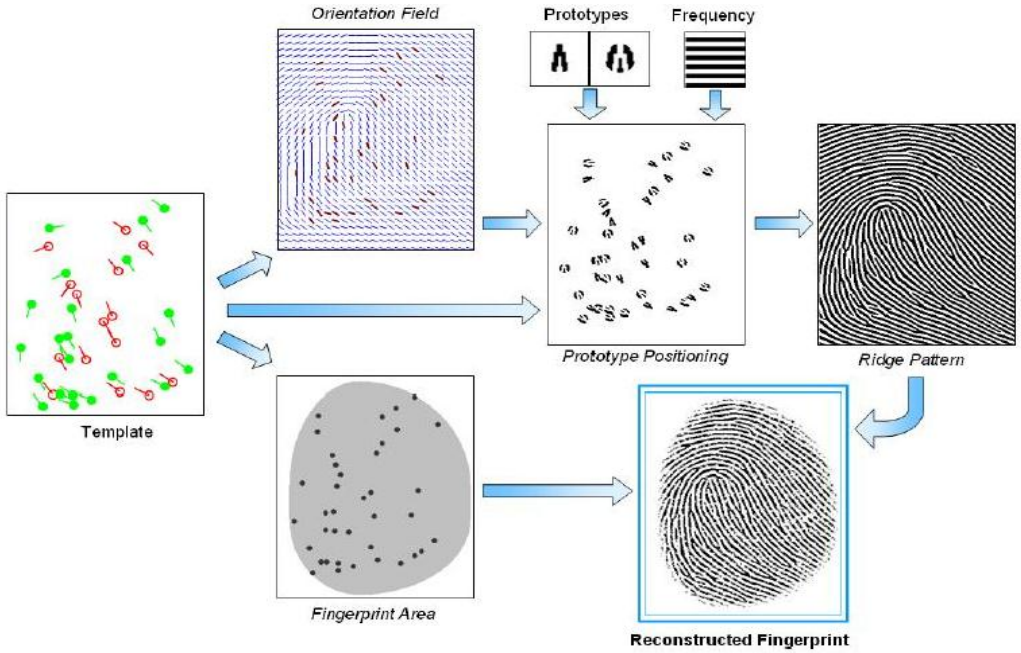


Figura 7: Algoritmo de reconstrucción propuesto por BioLab.

Existen sistemas biométricos que permiten darle tratamiento a las huellas dactilares, que consiste en la eliminación del ruido, adelgazamiento de la imagen entre otras técnicas para mejorar la calidad de la imagen, un ejemplo de estos sistemas es el **Fingerprint Recover** que es un sistema para la recuperación y enrolamiento de huellas dactilares, pero no realiza el proceso de reconstrucción de imágenes parciales de huellas dactilares.

Actualmente existe una aplicación llamada **SFINGE** que genera huellas dactilares de manera artificial, pero no realiza el proceso de reconstrucción. Se utiliza para crear grandes bases de datos de huellas dactilares artificiales, lo que permite a los algoritmos de reconocimiento ser simplemente probados y optimizados. La aplicación completa tiene un costo de 4900 euros.

1.5.2 Nivel Nacional.

En Cuba, la Biometría es un campo joven, la empresa cubana **Datys** es líder en soluciones biométricas. Desde el año 2006 viene desarrollando **Biomesys AFIS**, que es un sistema automatizado de identificación Dactiloscópica, permite la identificación y autenticación de personas de forma rápida y segura. Este sistema obtuvo el premio de "Calidad Informática" en la Feria de Informática 2009 en Ciudad de la Habana. Hasta el momento no se conoce ninguna empresa en el país que haya desarrollado algún software que permita reconstruir imágenes parciales de huellas dactilares.

1.5.3 En la Universidad de las Ciencias Informáticas.

En la Universidad de las Ciencias Informáticas (UCI), radica un departamento de Biometría perteneciente al Centro de Identificación y Seguridad Digital (CISED), en este departamento se desarrollan investigaciones y sistemas biométricos. Se han desarrollado varias aplicaciones que utilizan algoritmos para la extracción de minucias, comparación de huellas y generación de huellas dactilares artificiales, un ejemplo de este último es el sistema **Sintetizador de huellas dactilares**, el cual permite generar un conjunto de huellas artificiales a partir de una serie de parámetros definidos por el usuario, logrando construir huellas dactilares lo más reales posibles.

Actualmente CISED no cuenta con un componente para la reconstrucción de imágenes de huellas dactilares, herramienta que serviría para obtener parciales de imágenes de

huellas dactilares reconstruidas y con estos datos probar la seguridad de acceso de los sistemas de reconocimiento de personas mediante la huella dactilar.

En la investigación realizada no se encontró ninguna aplicación que realizara el proceso de reconstrucción de imágenes de huellas dactilares, sin embargo varios científicos han propuestos diferentes algoritmos para realizar el proceso de reconstrucción de imágenes de huellas dactilares.

1.6 Descripción de los principales algoritmos de reconstrucción de imágenes de huellas dactilares.

1.6.1 Algoritmo de reconstrucción a partir del modelo de Frecuencia Modular. (7)

Hill y Ross proponen:

- ✓ Reconstruir una imagen esqueletizada a partir de las minucias, y convertir una imagen a escala de grises.
- ✓ Generar el campo de orientación basado en las minucias y las singularidades de la huella.
- ✓ Generar una serie de curvas splines[1] que pasan a través de las minucias.
- ✓ Conformar las tripletas de minucias y estimar el campo de orientación a partir de las mismas.
- ✓ Aplicar texturas a partir de una integral de convolución lineal.
- ✓ Aplicar suavizado a la imagen.

Desventajas y Resultados.

- Tiende a generar falsas minucias.
- El algoritmo se prueba realizando el macheo de 2000 imágenes reconstruidas con las originales en la base de datos NIST-4, reportándose el 23% de ellas. (7)

[1] Curva diferenciable definida en porciones mediante polinomios.

Capelli propone:

- ✓ Realizar una variación del algoritmo de Hill y Ross en el cual el campo de orientación se calcula mediante el modelo no lineal propuesto por Vizcaya y Gerhaldt.
- ✓ Reconstruir la imagen a escala de grises a partir de las minucias.
- ✓ Calcular el campo de orientación a partir del modelo no lineal propuesto por Vizcaya y Gerhardt usando la dirección de las minucias.
- ✓ Aplicar textura a partir de filtros de Gabor de manera iterativa comenzando por las minucias.

Desventajas y Resultados.

- Se machean 120 imágenes reconstruidas contra 120 imágenes originales.
- Este algoritmo genera muchas falsas minucias. (7)

Jianjiang Feng y Anil K. Jain proponen: (Ver figura 8).

- ✓ Realizar una estimación de la representación de la frecuencia modular de la huella digital original
- ✓ Realizar una obtención de la fase en espiral.
  - ❖ Reconstruir el campo de orientación.
  - ❖ Estimar el gradiente de la fase continua.
  - ❖ Reconstruir la fase continua.
  - ❖ Combinar la fase en espiral con la fase continua. (7)

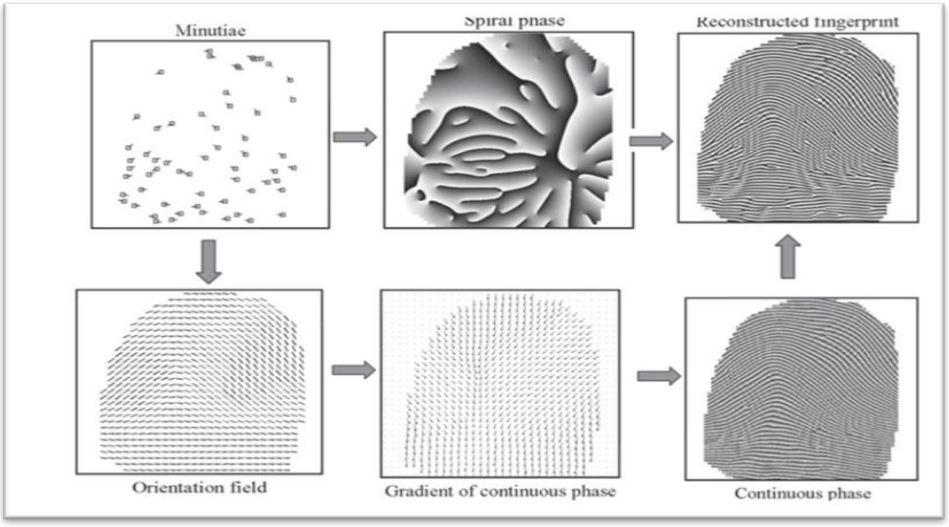


Figura 8: Proceso de reconstrucción a partir del modelo de Frecuencia Modular.

1.6.2 Algoritmo de reconstrucción a partir de la plantilla de minucias. (8)

Arun Ross (miembro de IEEE), Jidnya Shah y Anil K. Jain (miembro IEEE) proponen:

- ✓ Leer las plantillas de minucias (Ver figura 9).
- ✓ Formar las tripletas de minucias (Ver figura 10).
- ✓ Estimar el campo de orientación (Ver figura 11).
- ✓ Reconstruir la imagen de la huella dactilar (Ver figura 12).

Pasos para la reconstrucción a partir de plantillas de minucias:

1).

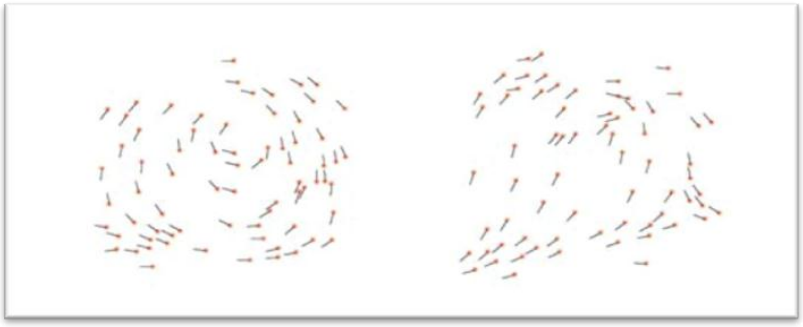


Figura 9: Cargar plantilla de minucias.

2).

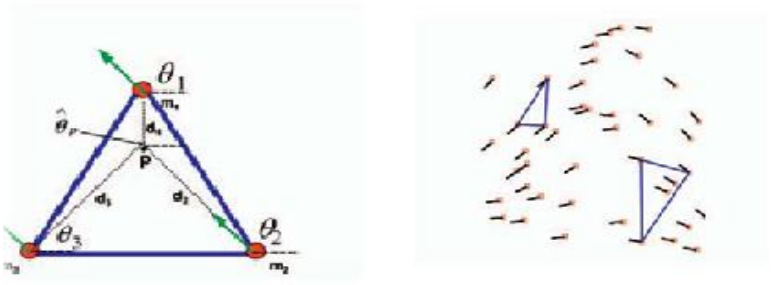


Figura 10: Formar Tripletas.

3).

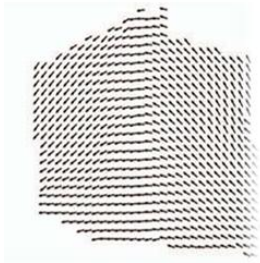


Figura 11: Estimar Campo de orientación.

4).

Figura 12: Reconstruir parcial dactilar.

Ventajas y resultados:

- ✓ El algoritmo se prueba realizando el macheo de 2000 imágenes reconstruidas con las originales en la base de datos NIST-4.
- ✓ Se toma una muestra de 500 huellas por clase, reportándose que las de tipo arco, lazo izquierdo y lazo derecho presentan mayor similitud en el proceso de macheo. (8)

Para el desarrollo del componente se seleccionó el algoritmo de reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias, basándose en que las validaciones estadísticas arrojadas por los especialistas (miembros de la IEEE), son precisas y confiables, y se descarta el primer algoritmo propuesto porque genera muchas minucias falsas lo cual dificulta el proceso de macheo.

1.7 Tecnología, metodologías y herramientas.

1.7.1 Metodología de desarrollo de software.

Una metodología para el desarrollo de software es un modo sistemático para realizar, gestionar y administrar un proyecto. Seleccionar una buena metodología será transcendental para el éxito de un producto. El papel principal de una metodología es guiar y organizar actividades que conlleven a las metas trazadas.

Para elegir una metodología de desarrollo de software se debe tener en cuenta dos factores fundamentales: el tipo de proyecto que se desea desarrollar y el tiempo que se dispone para desarrollar el mismo.

En la actualidad no se puede afirmar que existe una metodología que funcione de manera universal, son más bien concebidas como marcos metodológicos que deben ajustarse a cada organización y tipo de proyecto a desarrollar.

Existen dos grandes enfoques de metodologías: las tradicionales y las ágiles. (9)

1.7.1.1 Las metodologías tradicionales llevan un control estricto del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán. Centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto. Entre las principales metodologías tradicionales se encuentran RUP y MSF. (9)

- ✓ **Rational Unified Process (RUP)** es una metodología tradicional. Es un proceso de desarrollo de software formal, preparado para desarrollar grandes y complejos proyectos en largos plazos, y tiene la capacidad de adaptarse a las variaciones de complejidad de los proyectos. Fue desarrollado por Rational Software y utiliza el Lenguaje Unificado de Modelado UML como lenguaje de representación visual orientado a objetos. Este proceso de desarrollo cuenta con cuatro elementos principales: los trabajadores (quién), las actividades (cómo), los artefactos (qué) y el flujo de actividades (cuándo). RUP tiene tres características esenciales: está dirigido por los Casos de Uso, está centrado en la arquitectura, y es iterativo e incremental. (9)

El ciclo de vida de **RUP** se caracteriza por:

- **Dirigido por Casos de Uso:** Los Casos de Uso representan los requisitos funcionales del sistema, guían su diseño, implementación y prueba. No sólo inician el proceso de desarrollo sino que proporcionan un hilo conductor, permitiendo establecer trazabilidad entre los artefactos que son generados en las diferentes actividades del proceso de desarrollo. Basándose en los Casos de Uso se crean los modelos de análisis y diseño, luego la implementación que los lleva a cabo, y se verifica que efectivamente el producto implemente adecuadamente cada Caso de Uso. Constituyen un elemento integrador y una guía del trabajo. (Ver figura 13). (10)

Figura 13: Los Casos de Usos integran el trabajo.

- **Centrado en la arquitectura:** La arquitectura de un sistema es la organización o estructura de sus partes más relevantes, lo que permite tener una visión común entre todos los involucrados y una perspectiva clara del sistema completo, necesaria para controlar el desarrollo. En el caso de RUP se establece temprano una buena arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento.

- **Iterativo e Incremental:** La estrategia que se propone en RUP es tener un proceso iterativo e incremental el cual consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Una iteración puede realizarse por medio de una cascada. Se pasa por los flujos fundamentales (Requisitos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores. (10)

RUP divide el proceso de desarrollo del software en cuatro fases (Ver figura 14):

- I**nicio**: Su objetivo fundamental es determinar la visión inicial del proyecto y su alcance. Se define el modelo del negocio, se identifican todas las entidades externas que interactúen con el sistema (actores) y todos los casos de uso y se diseñan los Casos de uso más esenciales.
- **Elaboración**: El propósito de esta fase es analizar el problema, definir la línea base de la arquitectura, elaborar el plan del proyecto y eliminar los elementos de mayor riesgo para el sistema. Comprende la planificación de las actividades y del equipo necesario. La especificación de las necesidades y el diseño de la arquitectura. Al terminar esta fase se debe tener un modelo de los requerimientos del sistema, una descripción arquitectónica y un plan de desarrollo de software.
- **Construcción**: Durante esta fase todos los componentes, características y requisitos del sistema deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto. Comprende el diseño del sistema, la implementación y las pruebas**.** Al terminar esta fase se

debe tener un producto listo para su utilización que esté documentado y que posea un manual de usuario.

- **Transición**: Se ocupa de mover el sistema desde una comunidad de desarrollo hacia la comunidad del usuario y hacerlo trabajar en un entorno real. Al final de esta fase se debe tener un sistema de software documentado, que funcione correctamente y que satisfaga al usuario. (10)

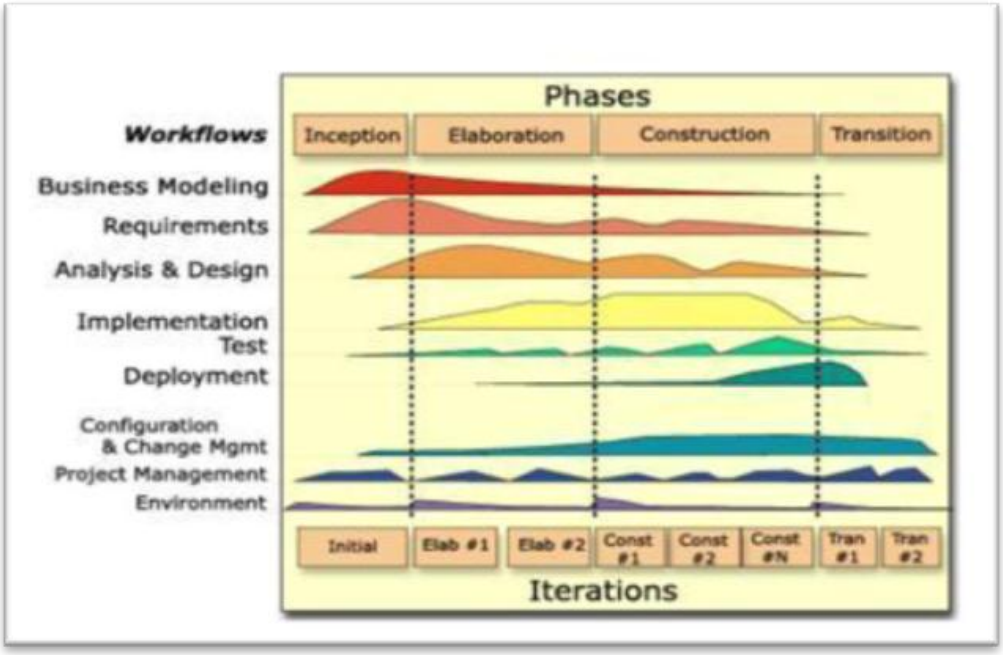


Figura 14: Esfuerzo de actividades según fase de proyecto.

- **Microsoft Solution Framework (MSF):** Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. Más que una metodología rígida de administración de proyectos, MSF es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información.

  MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

  Todo proyecto es separado en cinco principales fases:

  - Visión y Alcances.
  - Planificación.

- Desarrollo.
- Estabilización.
- Implantación. (9)

1.7.1.2 Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Se basa en la filosofía de que es más importante desarrollar software que funcione, que conseguir una buena documentación y es más importante responder ante un cambio, que seguir estrictamente un plan. (11)

- ✓ **Programación Extrema (XP)** es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo y propiciando un buen clima de trabajo. Es utilizada para proyectos de corto plazo, equipo pequeño y cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. El objetivo fundamental es la satisfacción del cliente. Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP define cuatro fases fundamentales: exploración, planificación, implementación y pruebas. Es especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. (11)

- ✓ **Scrum** no es propiamente un método o metodología de desarrollo, e implantarlo como tal resulta insuficiente, se utiliza como marco para otras prácticas de ingeniería de software como RUP o XP. Es un proceso ágil y liviano que sirve para administrar y controlar el desarrollo de software. El desarrollo se realiza en forma iterativa e incremental (una iteración es un ciclo corto de construcción repetitivo). Las iteraciones en general tienen una duración entre 2 y 4 semanas. Está diseñado especialmente para adaptarse a los cambios en los requerimientos. Los requerimientos y las prioridades se revisan y ajustan durante el proyecto en intervalos muy cortos y regulares. Se busca entregar software que realmente resuelva las necesidades, aumentando la satisfacción

del cliente. El equipo se focaliza en una única cosa: construir software de calidad. (9)

1.7.2 Lenguaje de Modelación.

Un lenguaje para el modelado de objetos es un conjunto estandarizado de símbolos y de modos de disponerlos para modelar un diseño de software orientado a objetos.

- ✓ **Unified Modeling Language (UML):** Se define como un lenguaje de modelado visual que permite especificar, visualizar y construir los artefactos de los sistemas de software. Es importante resaltar que UML es un "lenguaje" para especificar y no para describir métodos o procesos. UML está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Permite modelar, construir y documentar los elementos que forman un sistema de software orientado a objetos. (12)

1.7.3 Herramientas para el diseño de la aplicación.

Herramienta CASE (Ingeniería del software asistida por computadoras): Es un conjunto de aplicaciones informáticas que tienen el objetivo de aumentar la productividad en el desarrollo de un software mitigando los costes en términos de tiempo. 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 que hacen de ellas una fuente importante para la creación de un producto con calidad. (13)

Entre los beneficios más significativos de las herramientas CASE se encuentran los siguientes:

- Facilidad para la revisión de aplicaciones.
- Soporte para el desarrollo de prototipos de sistemas.
- Generación de código.
- Mejora en la habilidad para satisfacer los requerimientos del usuario.
- Soporte interactivo para el proceso de desarrollo. (13)

Las herramientas CASE se han venido ampliando y desarrollando, algunos de sus ejemplos son: Microsoft Project, Rational Rose, JDeveloper, MagicDraw, Visual Paradigm, Microsoft Visio, Enterprise Architect, entre otros.

1.7.3.1 Visual Paradigm.

Visual Paradigm es un potente conjunto de herramientas CASE, que utiliza UML como lenguaje de modelado, soporta el ciclo de vida completo del desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Su diseño está centrado en casos de uso y permite modelar todos los tipos de diagramas de clases. Produce documentación del sistema en varios formatos como PDF y HTML. Presenta licencia gratuita y comercial, además que es fácil de instalar, actualizar y compatible entre ediciones. (14)

1.7.3.2 Rational Rose.

Rational Rose es una herramienta de producción y comercialización establecida por Rational Software Corporation. Utiliza el Lenguaje Unificado (UML) para facilitar la captura de dominio de la semántica, la arquitectura y el diseño. Este software tiene la capacidad de crear, ver, modificar y manipular los componentes de un modelo. No es gratuito, admite la integración con otras herramientas de desarrollo. Habilita asistentes para crear clases y provee plantillas de código que pueden aumentar significativamente la cantidad de código fuente generada. (15)

1.7.4 Lenguajes de Programación.

Los lenguajes de programación se usan para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Están formados por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura, el significado de sus elementos y expresiones.

Los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar. (16)

- ✓ **Lenguaje C++**

C++ es un lenguaje imperativo orientado a objetos, es un súper conjunto de C, que nació para añadirle cualidades y características de las que carecía. El resultado es que como su ancestro, sigue muy ligado al hardware subyacente, manteniendo una considerable potencia para programación a bajo nivel, pero se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción.

C++ no es un lenguaje orientado a objetos puro (en el sentido en que puede serlo Java por ejemplo), se trata simplemente del sucesor de un lenguaje de programación hecho por programadores (de alto nivel) para programadores, al cual le han ido añadiendo todos los elementos que la práctica aconsejaba como necesarios, con independencia de su belleza o purismo conceptual. Es versátil, flexible, conciso y muy eficiente. Por muchos años fue el preferido en el desarrollo de aplicaciones. Se ha utilizado para implementar el núcleo de sistemas como Windows. (17)

- ✓ **Lenguaje C#**

C# es el nuevo lenguaje de propósito general diseñado por Microsoft para su plataforma .NET. Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado en ella, por lo que desarrollar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes. Por esta razón, se suele decir que C# es el lenguaje nativo de .NET. (18)

Presenta las siguientes características:

- Sencillez.
- Modernidad.
- Orientación a objetos.
- Orientación a componentes.
- Gestión automática de memoria.
- Seguridad de tipos.
- Instrucciones seguras.
- Sistema de tipos unificado.
- Eficiencia.
- Compatibilidad.

- ✓ **Lenguaje Java**

Java está inspirado en C++ y se proyectó con la finalidad de obtener un producto de pequeñas dimensiones, simple y portátil sobre diferentes plataformas y sistemas operativos ya sea a nivel de código fuente como a nivel de código binario. Es un lenguaje orientado a objetos, eso implica que su concepción es muy próxima a la forma de pensar humana, genera ficheros de clases compiladas, pero estas son en realidad interpretadas por la Máquina Virtual de Java, quien mantiene el control sobre las clases que se estén ejecutando. El mismo es multiplataforma y seguro: La máquina virtual, al ejecutar el código java, realiza comprobaciones de seguridad, además el propio lenguaje carece de características inseguras, como por ejemplo los punteros.

Su sintaxis ha sido trabajada mejorando la de C++ logrando mayor sencillez y legibilidad. Presenta mayor robustez al simplificar la gestión de memoria y eliminar las complejidades del manejo explícito de punteros. Se puede compilar y ejecutar en cualquier plataforma de sistema operativo por ejemplo en Windows, Solaris o Linux gracias a su máquina virtual. Su desarrollo ha sido rápido y exitoso debido a la gran cantidad de grandes empresas colaboradoras que han dado su aporte para enriquecerlo. La ejecución de programas escritos en Java suele comportarse más lenta que la de aplicaciones de otro lenguaje haciendo un uso voraz de recursos como memoria y procesador. Esto se hace más notorio si la ejecución se basa en cálculos matemáticos complejos o si la aplicación presenta un diseño cargado de componentes visuales. (19)

1.7.5 Entornos integrados de desarrollo.

Un entorno integrado de desarrollo (IDE), es un programa informático compuesto por un conjunto de herramientas de programación, utilizadas por los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos.

Las herramientas que normalmente componen un entorno de desarrollo integrado son las siguientes: un editor de texto, un compilador, un intérprete, unas herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y, opcionalmente, un sistema de control de versiones.

Hoy día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado como es el caso de C++, C#, Java, Python y Visual Basic. (20)

Entre los entornos integrados de desarrollo se encuentran los siguientes:

- Eclipse
- NetBeans
- Visual Studio.NET

✓ **Eclipse**

Eclipse fue desarrollado originalmente por IBM. Es ahora desarrollado por la Fundación Eclipse, una organización independiente que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-Liviano" basadas en navegadores, es una potente y completa plataforma de Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Además contiene una atractiva interfaz que lo hace fácil y agradable de usar. (21)

✓ **NetBeans**

NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados de manera independiente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. NetBeans es un proyecto de código abierto de gran éxito y con una comunidad en constante crecimiento. (22)

✓ **Visual Studio .NET**

Es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como C++, C#, J#, ASP.NET y Visual Basic .NET. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y

aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. (23)

1.8 Propuesta de las herramientas, metodologías y tecnologías a utilizar.

Después de haber analizado las diferentes características de las metodologías, tecnologías y herramientas más usadas en la actualidad se propone utilizar RUP como metodología de desarrollo de software, ya que posee grandes ventajas para el análisis, implementación y documentación de sistemas orientados a objetos. Esta metodología está preparada para desarrollar todo tipo de proyectos a largos plazos y es muy organizada ya que genera una documentación donde se detalla todo el proceso en general. Para modelar el sistema se propone la herramienta Case (versión 6.0) de Visual Paradigm Enterprise Edition para UML, ya que utiliza un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación. Visual Paradigm tiene la ventaja de ser software libre. Permite realizar ingeniería tanto directa como inversa y además es una herramienta colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como web o PDF y es además una herramienta multiplataforma. Para la implementación del componente se utiliza el lenguaje de programación C#, es un lenguaje orientado a objetos, tiene una sintaxis muy parecida a Java y posee la potencia de C++. Las soluciones desarrolladas en el departamento de Biometría son implementadas en C#, teniendo en cuenta la posibilidad de integración con otra herramienta se decidió seguir una misma línea de desarrollo para un mejor entendimiento del grupo de trabajo. Como entorno integrado de desarrollo se usará Visual Studio porque permite a los desarrolladores crear aplicaciones de alta calidad con gran rapidez, más seguras, confiables y administrables en cualquier entorno que soporte la plataforma .NET.

1.9 Conclusiones parciales.

En el estudio realizado a los diferentes sistemas biométricos basados en huellas dactilares no se encontró evidencias de que existan sistemas que realicen el proceso de reconstrucción de parciales de huellas dactilares a partir de plantillas de minucias. Los sistemas encontrados se ajustan a las necesidades y características de la institución para las cuales fueron desarrollados, por lo que no son aplicables al campo de acción, además no contemplan el proceso de reconstrucción de una huella dactilar. Después de

haber analizado las ventajas y desventajas de los diferentes algoritmos de reconstrucción de imágenes de huellas dactilares, se definió para implementar el algoritmo de reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378.

El análisis realizado a las diferentes herramientas, lenguajes y tecnologías permitió profundizar los conocimientos necesarios para el desarrollo del componente de reconstrucción de huellas dactilares, y ayudó a determinar según las necesidades de la solución, a utilizar RUP como metodología de desarrollo, C# como lenguaje de programación, Visual Studio como entorno de desarrollo, y Visual Paradigm como herramienta de modelado.

Capítulo 2: Características del Sistema.

2.1 Introducción.

En este capítulo se describen las principales características del sistema propuesto. Se ofrece el modelo de dominio de la aplicación, se especifica el levantamiento de los requisitos funcionales y no funcionales, y a partir de ellos, los casos de uso del sistema con sus respectivas descripciones. Se realizan los diagramas de clases del análisis, diagramas de colaboración, diagramas de secuencia y el diagrama de clases del diseño. Además se define la arquitectura y se especifican los patrones utilizados, y mediante ellos se logra un mejor entendimiento para el posterior desarrollo del sistema.

2.2 Descripción del problema.

En el departamento de Biometría, perteneciente al Centro de Identificación y Seguridad Digital de la Universidad de las Ciencias Informáticas se desarrollan sistemas biométricos basados en iris, rostro y huella dactilar. Actualmente se desarrolla un sistema que pretende integrar funcionalidades para trabajar con las huellas dactilares. El componente a desarrollar permite reconstruir imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378 y aporta al departamento imágenes parciales de huellas reconstruidas que pudieran ser utilizadas para detectar posibles vulnerabilidades de acceso en los sistemas de reconocimiento por huella dactilar. Desarrollar este componente permitirá probar la seguridad de acceso de los sistemas biométricos desarrollados en el centro.

2.2.1 Modelo de Dominio.

En el modelo de dominio se muestran los principales conceptos con los que trabaja la aplicación en desarrollo, lo que ayuda a comprender el entorno del sistema. Este modelo contribuirá a describir las clases más importantes dentro del contexto de la aplicación.

A través de este modelo se observa:

- **Usuario:** Es el encargado de seleccionar la plantilla de minucias para iniciar el proceso de reconstrucción. Carga una o varias plantillas de minucias y a partir de estas el usuario inicia el proceso para reconstruir una o varias huellas dactilares.

- **Huella dactilar:** Es la representación superficial de la epidermis de un dedo, posee un conjunto de líneas conocidas como crestas y los puntos donde estas se terminan o se bifurcan son tipos de minucias.
- **Minucias:** Son los puntos singulares en el trazo de las crestas de una huella dactilar. Cada minucia tiene una coordenada X, coordenada Y, y el ángulo de orientación. En una huella dactilar existen aproximadamente de 50 a 150 minucias. Las minucias que tienen mayor valor significativo en la identificación, son las terminaciones y bifurcaciones.
- **Plantilla:** Almacena información de la huella dactilar, está compuesta por un conjunto de minucias en estándar ANSI/INCITS 378.

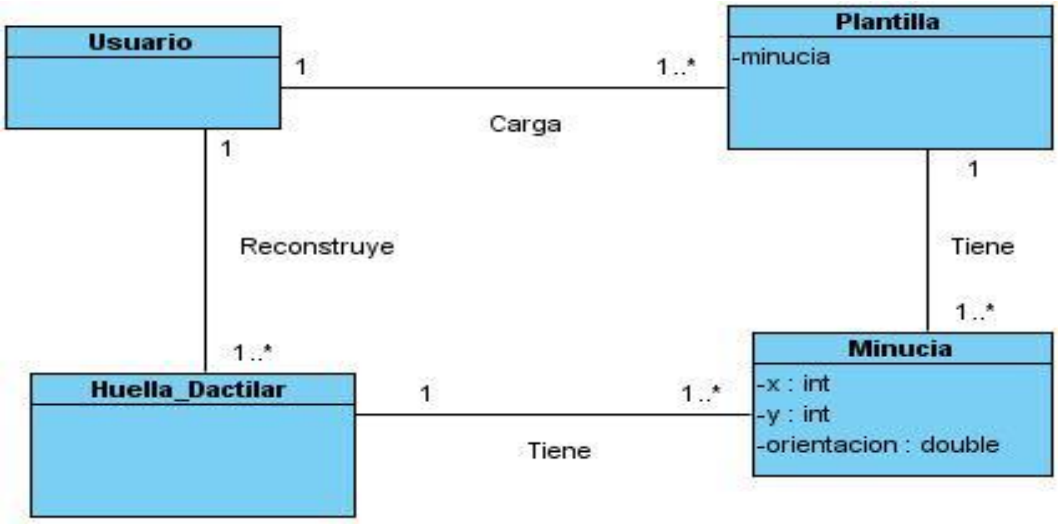


Figura 15: Modelo de dominio.

2.3 Modelo del sistema.

Con el conocimiento adquirido hasta el momento sobre los conceptos que rodean al campo de acción, se pueden analizar las características que debe tener el sistema para que se cumplan los objetivos planteados al inicio. Para ello se identifican los requisitos funcionales y no funcionales, modelando los requisitos funcionales en términos de casos de uso.

2.3.1 Propuesta del sistema.

El presente trabajo pretende desarrollar un componente para la reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI/INCITS 378, con el objetivo de obtener imágenes parciales de huellas dactilares

reconstruidas que puedan ser utilizadas para probar la seguridad de acceso de los sistemas de identificación biométrica basados en huellas dactilares. El componente está basado en dos módulos que permiten la reconstrucción de la huella dactilar.

En el primer módulo las imágenes de huellas dactilares son reconstruidas de forma visual a partir de una serie de pasos seleccionados por el usuario, comenzando por la selección de la plantilla de minucias, la formación de las tripletas de minucias, estimación del campo de orientación y por último la reconstrucción de la imagen parcial de la huella dactilar. El usuario podrá reconstruir la cantidad de plantillas de minucias que desee, pero sólo puede reconstruir una huella dactilar por plantilla y podrá almacenarlas en un fichero en formato BMP, JPG, PNG o TIFF.

En el segundo módulo las imágenes de huellas dactilares son reconstruidas de forma automática, se comienza por la selección de las plantillas de minucias que se desean reconstruir y luego se procede a la opción de generación automática, sin necesidad de observar el proceso de reconstrucción todas las imágenes reconstruidas se almacenan en una dirección seleccionada por el usuario en formato PNG.

2.3.2 Requisitos funcionales.

Los requisitos funcionales describen lo que el sistema debe hacer.

- ✓ **RF-1 Reconstruir Huella Dactilar**. (Permite reconstruir una huella dactilar de manera visible para el usuario y muestra todo el proceso de reconstrucción).
  - 1.1 Leer las plantillas de minucias en estándar ANSI/INCITS 378.
  - 1.2 Formar las tripletas de minucias.
  - 1.3 Estimar campo de orientación.
  - 1.4 Reconstruir un parcial de la imagen de la huella.
  - 1.5 Guardar la imagen de la huella en formato BMP, JPG, PNG o TIFF.

- ✓ **RF-2 Generación automática** (Permite reconstruir una o varias plantillas de minucias de huellas dactilares de manera automática, sin necesidad de observar el proceso de reconstrucción todas las imágenes reconstruidas se almacenan en formato PNG).
  - 2.1 Leer la(s) plantilla(s) de minucias en estándar ANSI/INCITS 378.
  - 2.2 Seleccionar una carpeta de destino para almacenar las huellas reconstruidas.
  - 2.3 Seleccionar generar parciales.

2.3.3 Requisitos no funcionales.

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

- ✓ **RNF-1. Apariencia o interfaz externa:**
  - El diseño de la interfaz debe ser sencillo y amigable.
  - Los colores deben ser refrescantes a la vista del usuario.

- ✓ **RNF-2. Usabilidad:**
  - El sistema podrá ser usado por cualquier persona con conocimientos básicos sobre el manejo de la computadora.

- ✓ **RNF-3. Software.**
  - Sistema operativo: Windows.
  - Framework .NET 4.0.

2.3.4 Modelo de Caso de Uso del sistema.

Una vez definidos los requisitos funcionales del sistema, estos se representarán mediante un diagrama de Casos de Uso.

Los Casos de Uso representan fragmentos de funcionalidad del sistema que proporcionan un valor observable al usuario. Es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. (24)

Un actor representa un conjunto coherente de roles que juegan los usuarios de los Casos de Uso cuando interactúan con éstos. Los actores pueden ser personas (roles que desempeñan las personas) u otros sistemas de cómputo.

Primero se debe definir claramente cuáles son los actores que van a interactuar con el sistema, y los Casos de Uso que representarán todas las funcionalidades del sistema. (Anexo 1).

2.3.5 Diagrama de Caso de Uso del sistema.

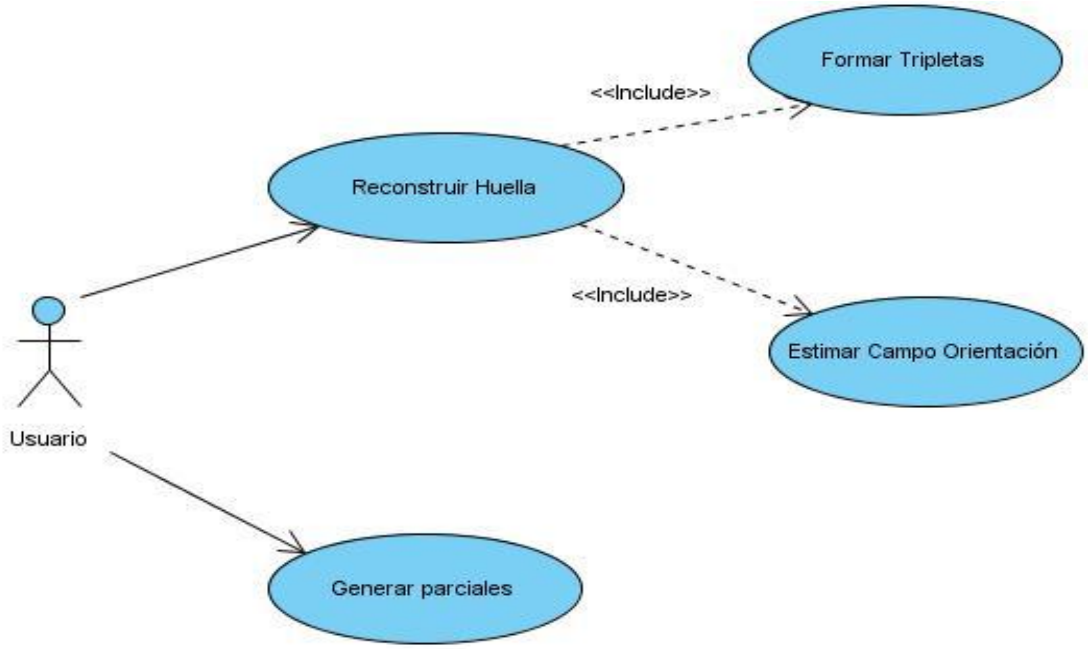


Figura 16: Diagrama de Caso de Uso.

2.3.6 Descripción de los Casos de Uso del sistema.

Tabla 2: Descripción del Caso de Uso: Reconstruir huella.

Nombre del caso de uso: Reconstruir huella.	
Actores	Usuario
Propósito	Reconstruir una imagen parcial de una huella dactilar de manera visual y luego almacenarla en una carpeta local seleccionada por el usuario.
Resumen	El Caso de Uso inicia cuando el usuario selecciona la opción Reconstruir Huella.
Referencia	RF-1,RF-1.1, RF-1.2, RF-1.3, RF-1.4 RF-1.5 CUS: Formar tripletas (include). CUS: Estimar campo orientación (include).
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción	2. El sistema muestra una lista con las

Cargar plantilla.	plantillas de minucias de las huellas dactilares en estándar ANSI/INCITS 378.
3. El usuario escoge la plantilla de minucias de la huella dactilar que desea reconstruir.	4. El sistema muestra una interfaz con la representación de la plantilla de minucias seleccionada.
5. El usuario selecciona la opción formar tripletas.	6. El sistema muestra las tripletas de minucias. (Ver CUS: Formar tripletas)
7. El usuario selecciona la opción Estimar campo orientación.	8. El sistema visualiza el campo de orientación de la huella dactilar. (Ver CUS: Estimar campo orientación).
9. El usuario selecciona la opción Reconstruir parcial dactilar.	10. El sistema muestra la imagen parcial de la huella dactilar.
11. El usuario selecciona la opción Guardar parcial.	12. El sistema muestra una interfaz para escoger la ruta destino donde se desea almacenar la huella y muestra los diferentes formatos BMP, JPG, PNG o TIFF en que serán guardados.
13. El usuario escoge la ruta destino y el formato de la huella y selecciona la opción guardar.	14. El sistema muestra un mensaje de confirmación.
Pos condiciones	Se obtuvo una imagen parcial de la huella.

Tabla 3: Descripción del Caso de Uso: Formar tripletas.

Nombre del caso de uso: Formar tripletas.	
Actores	Usuario
Propósito	Formar las tripletas de minucias.
Resumen	El caso de uso inicia cuando el usuario selecciona la opción Formar tripletas.
Referencia	RF-1,RF-1.1, RF-1.2
Precondiciones	Se tiene que haber iniciado la opción Cargar plantilla y haber seleccionado la plantilla de minucias en estándar ANSI/INCITS 378.
Flujo normal de eventos	

Acción del actor	Respuesta del sistema
1. El usuario selecciona la opción Formar tripletas.	2. El sistema busca tres minucias que cumplan con las condiciones de formar tripletas.
	3. El sistema calcula la distancia entre dos puntos (minucias).
	4. El sistema verifica que Lmin<= Li<=Lmax donde Li (cada lado de la tripleta); Lmin=20; Lmax=30.
	5. El sistema verifica que el máximo valor de la resta de cada una de las orientaciones, menos la mediana de dichas orientaciones tiene que ser menor o igual que 30 grados.
	6. El sistema verifica que los ángulos interiores de una tripleta deben ser mayor que 20 grados.
	7. El sistema muestra una interfaz gráfica con la representación de las tripletas.
Flujo Alterno	
Acción del actor.	Respuesta del Sistema.
	2.1 Si las tres minucias escogidas no cumplen con algunas de las condiciones para formar tripletas, el sistema buscará otro trío de minucias que cumplan con todas las condiciones.
Pos condiciones	Se obtuvo las tripletas de minucias.

Tabla 4: Descripción del Caso de Uso: Estimar campo orientación.

Nombre del caso de uso: Estimar campo orientación.	
Actores	Usuario
Propósito	Representar gráficamente la orientación de las minucias de la huella.
Resumen	El caso de uso inicia cuando el usuario selecciona la opción

	Estimar campo orientación.
Referencia	RF-1, RF-1.1, RF-1.2, RF-1.3.
Precondiciones	Se tiene que haber cargado la plantilla de minucias en estándar ANSI/INCITS 378 y haber formado las tripletas.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción Estimar campo orientación.	2. El sistema calcula la orientación de cada pixel de la imagen.
	3. El sistema estima la matriz de orientación.
	4. El sistema realiza un suavizado a la matriz de orientación.
	5. El sistema realiza un promedio de orientación de los pixeles de la imagen.
	6. El sistema reduce la matriz en bloques de 13 x13 pixeles.
	7. El sistema representa gráficamente el campo de orientación.
Pos condiciones	Se obtuvo el campo de orientación de la huella.

Tabla 5: Descripción del Caso de Uso: Generar parciales.

Nombre del caso de uso: Generación automática.	
Actores	Usuario
Propósito	Obtener un conjunto de imágenes parciales de huellas reconstruidas de forma automática.
Resumen	El caso de uso inicia cuando el usuario selecciona la opción Generación automática.
Referencia	RF-2
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción Generación automática.	2. El sistema muestra una interfaz donde el usuario debe escoger la opción de cargar plantillas.

3. El usuario selecciona la opción cargar plantillas y selecciona las plantillas que desea reconstruir en estándar ANSI/INCITS 378.	4. El sistema muestra una interfaz donde el usuario debe seleccionar la ruta de destino para almacenar las huellas reconstruidas.
5. El usuario escoge la ruta de destino y selecciona la opción Generar.	6. El sistema muestra una interfaz donde se muestra una barra indicando el estado del proceso y guarda los parciales de huellas en formato PNG.
	7. El sistema notifica un mensaje indicando que el proceso ha terminado.
	8. El sistema notifica un mensaje indicando si desea abrir la carpeta donde fueron guardadas los parciales reconstruidos.
Pos condiciones	Se obtuvo un conjunto de imágenes parciales de huellas reconstruidas en formato PNG.

2.4 Análisis.

El análisis es la etapa del flujo de trabajo Análisis y Diseño, que se encarga de refinar y estructurar los requisitos identificados, con el objetivo de lograr una mejor comprensión y descripción de los mismos, que facilite estructurar el sistema en su totalidad. Se procura ante todo identificar y describir los objetos (o conceptos) dentro del dominio del problema.

En la construcción del modelo de análisis se identifican las clases que describen la realización de los casos de uso, los atributos y las relaciones entre ellas. Con esta información se construye el diagrama de clases del análisis. El modelo de análisis es el resultado de la actividad de analizar los casos de uso y su realización ayuda la transición al diseño y se utiliza para tener una visión general de la propuesta del sistema. (25)

2.4.1 Diagrama de Clases del Análisis.

Un diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa el funcionamiento del mundo real, no de la implementación automatizada del mismo. Las clases del análisis se centran en

los requisitos funcionales y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos.

RUP propone clasificar a las clases en:

- ✓ **Clase Interfaz:** Se utilizan para modelar la interacción entre el sistema y los actores.

- ✓ **Clase Entidad:** Se utilizan para modelar información que a menudo es persistente. Modelan la información y el comportamiento asociado de algún fenómeno o concepto, como una persona, un objeto, o un suceso del mundo real.

- ✓ **Clase Controladora:** Coordinan la realización de uno o unos pocos casos de uso coordinando las actividades de los objetos que implementan la funcionalidad del caso de uso.

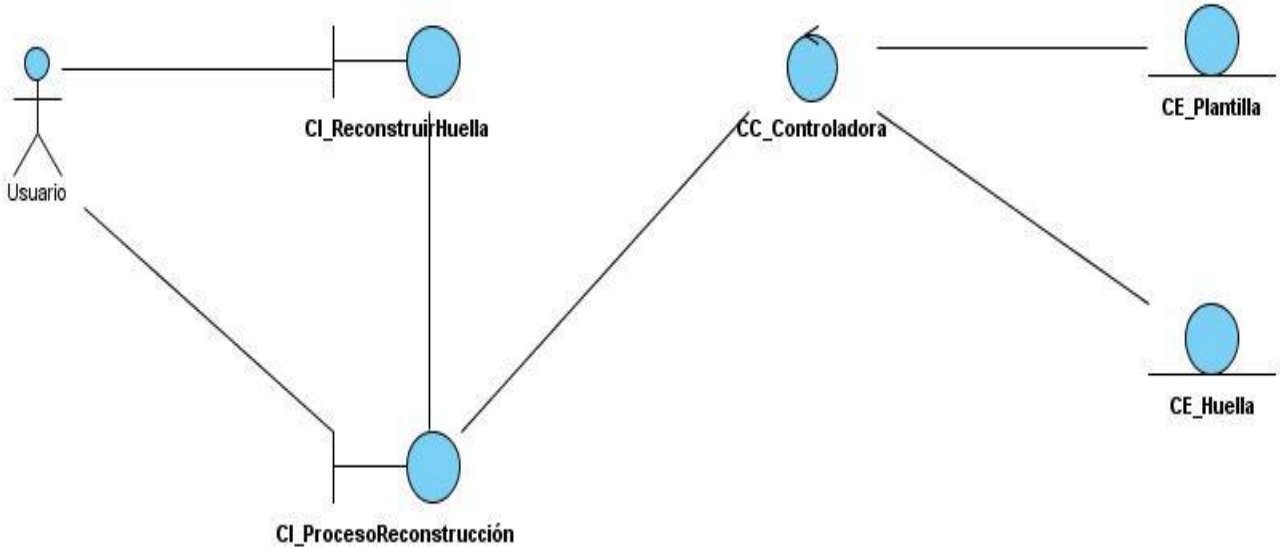


Figura 17: Diagrama de clases del análisis del CUS: Reconstruir Huella Dactilar.

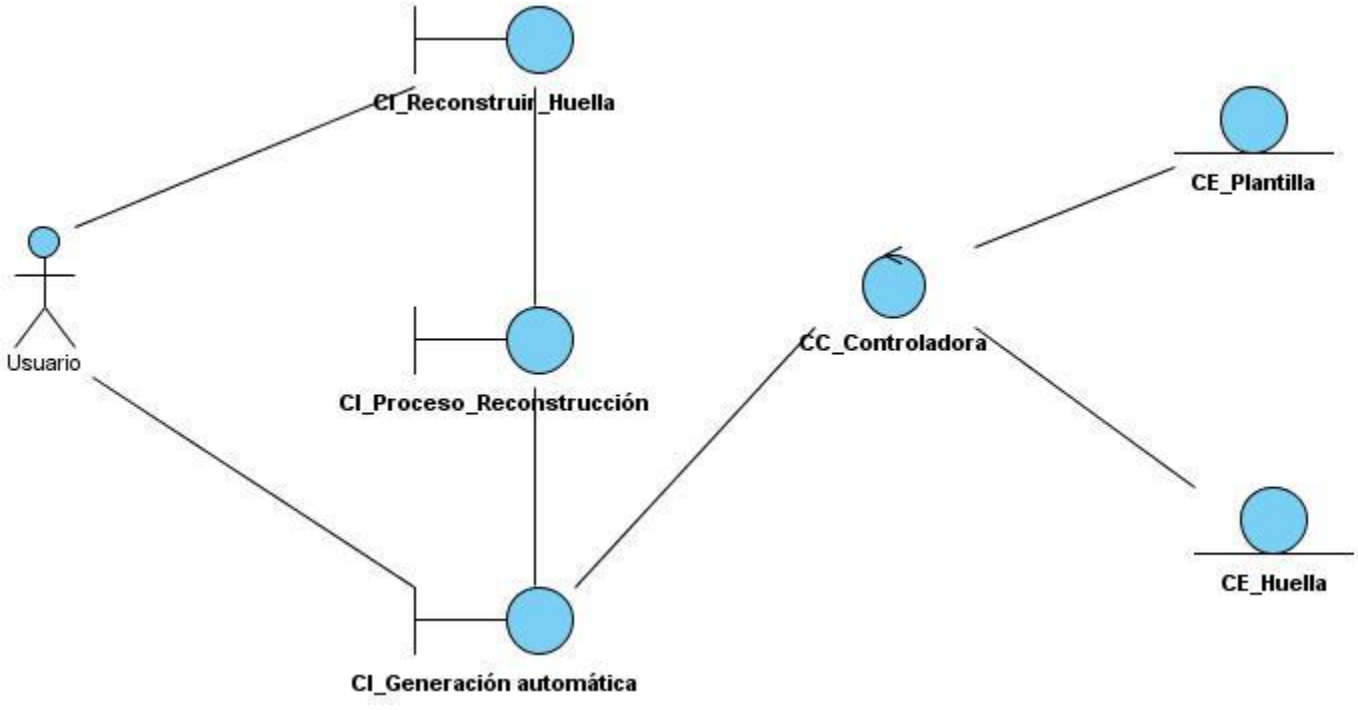


Figura 18: Diagrama de clases del análisis del CUS: Generar parciales.

2.4.2 Diagramas de interacción.

Los diagramas de colaboración o diagramas de secuencia explican gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas. (Anexo 2)

2.5 Diseño.

El propósito fundamental del diseño es indicar como se llevará a cabo el sistema y encontrar su forma (incluida la arquitectura) para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen. Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. Se procura definir los objetos lógicos del software que finalmente serán implementados en un lenguaje de programación orientado a objetos. Los objetos tienen atributos y métodos. (25)

2.5.1 Patrones del diseño.

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Se consideran como procedimientos para solucionar diversas veces un problema del mismo tipo y son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software. Expresan esquemas

para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

Un patrón impone una regla sobre la arquitectura, pues describe la manera en que el software manejará algún aspecto de su funcionalidad al nivel de la infraestructura.

Los patrones son una pareja de problema/solución con un nombre, que codifican buenos principios y sugerencias relacionados frecuentemente con la asignación de responsabilidades. (25)

Los desarrolladores lo usan como una forma de reutilizar la experiencia, clasificando las soluciones con términos de común denominación. Van formando un amplio repertorio de principios generales y de expresiones que los guían al crear software.

En el diseño de la aplicación se usarán los patrones **GRASP** (General Responsibility Asignment Software Patterns) que significa patrones generales de software para asignar responsabilidades. Describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (25)

Dentro de los patrones de diseño GRASP se utilizó:

- ✓ **Controlador:** Responsable de gestionar un evento de entrada al sistema. Es un intermediario entre la capa de presentación y el núcleo de las clases donde reside la lógica del sistema. El controlador coordina la actividad de otros objetos. Por ejemplo en la aplicación se hace uso del patrón controlador definiendo la clase Controladora la cual gestiona todo el flujo de datos de la aplicación, y maneja varias instancias de objetos.

- ✓ **Experto:** La solución factible es asignar la responsabilidad a la clase contenedora de la información necesaria para cumplir la responsabilidad. El uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide, favorece la existencia de mínimas relaciones entre las clases, lo que permite contar con un sistema sólido y fácil de mantener. Por ejemplo este patrón se pone de manifiesto en la clase OrientacionPorBloques, la cuál es la encargada de estimar la orientación de la matriz.

- ✓ **Creador:** Responsable de crear una nueva instancia de alguna clase. Guía la asignación de responsabilidades relacionadas con la creación de objetos. Se

hace uso de este patrón cuando es necesario que las clases creen instancias con el nivel necesario de información para acceder a los datos almacenados, de acuerdo a la ejecución de una determinada acción. Se considera para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto solo pueda ser creada por el objeto que contiene la información necesaria para ello. El uso de este patrón admite crear las dependencias mínimas necesarias entre las clases, beneficiando el mantenimiento del sistema y brindando mejores oportunidades de reutilización. Por ejemplo, en la aplicación se pone en práctica en la clase Controladora, donde se crea un objeto de la clase Tripleta, en la cual para poder crear un instancia de ella hay que pasarle la información necesaria, o sea, tres puntos de minucias.

También se utilizó para el diseño de la aplicación el patrón **Acción** de tipo **Comportamiento**, el cual pertenece a los patrones **GoF (*Gang Of Four*).**

- ✓ **Comportamiento**: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos. (32)

- **Acción/ Comando**: Es un patrón de tipo comportamiento, a nivel de objetos. Se pone de manifiesto cuando se está en presencia de una interfaz gráfica de usuario que consta con un menú desplegable y una barra de botones, donde en esta barra se anclan los elementos más usados de la aplicación. El problema de esta solución es que se duplica la implementación de la operación que ejecutan tanto la entrada de menú como el botón. El hecho de que existan dos o más alternativas para acceder a una misma función de la aplicación, no debe traducirse en redundancia a la hora de implantar dicha función. Una solución que nos permite evitar esa redundancia consiste en definir la acción que se quiere ejecutar como un objeto, al cual pueden referirse todos aquellos elementos gráficos que lo deseen. Por ejemplo, en la aplicación se pone de manifiesto este patrón, cuando se permite cargar la plantilla de minucias, a esta funcionalidad se puede acceder de dos maneras: a través de un menú desplegable, o a través de un botón Abrir. Se implementa una acción nombrada Abrir, la cual es utilizada

por todos aquellos elementos visuales que lo deseen, esta forma permite acceder de dos maneras a una misma funcionalidad y evita que exista redundancia a la hora de la implementación. (Ver anexo 4).

2.5.2 Arquitectura.

La arquitectura constituye un modelo relativamente pequeño y comprensible de cómo está estructurado el sistema y cómo trabajan juntos sus componentes.
La arquitectura del software es el diseño de más alto nivel de la estructura de un sistema. También es denominada Arquitectura Lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema informático. Establece los fundamentos para que analistas, diseñadores, programadores y otros roles, trabajen en una línea común que permita alcanzar los objetivos del sistema, cubriendo todas las necesidades. (26)

Las arquitecturas y estilos arquitectónicos más universales son:

- ✓ **Cliente-servidor:** Esta arquitectura se divide en dos partes claramente diferenciadas, la primera es la parte del servidor y la segunda la de un conjunto de clientes. Normalmente el servidor es una máquina bastante potente que actúa de depósito de datos y funciona como un sistema gestor de base de datos (SGBD). Por otro lado los clientes suelen ser estaciones de trabajo que solicitan varios servicios al servidor. Ambas partes deben estar conectadas entre sí mediante una red.

- ✓ **Arquitectura n-capas:** Se basa en una distribución jerárquica de roles y responsabilidades para proporcionar una división efectiva de los problemas a resolver. Los roles indican el tipo y la forma de interacción con otras capas, y las responsabilidades la funcionalidad que implementan. El sistema se divide en subsistemas (capas) donde cada capa descansa sobre la inferior, utilizando servicios de esta. El flujo de datos en este estilo está concebido unidireccionalmente, es decir los datos viajarán en una dirección y desde la capa presentación no se puede acceder a la base de datos, y viceversa, lo que asegura la integridad de los datos.

- ✓ **Tuberías y Filtros:** Se descompone el sistema en módulos funcionales. Interacción sucesiva, transformación de flujos de datos. Los datos llegan a un filtro, se transforman y son pasados a través de tubos al siguiente filtro. Es apropiada para sistemas que implementan transformaciones de datos en pasos sucesivos. Se aplica cuando los datos de entrada son transformados a través de una serie de componentes computacionales o manipulativos en los datos de salida. 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 datos de una forma específica.

- ✓ **Arquitectura Orientada a Servicios (SOA):** Establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma.

Se utilizó la **arquitectura n-capas** como línea base para la organización estructural de la aplicación Reconstrucción de imágenes de huellas dactilares. El sistema se divide en las siguientes capas: (Ver figura 19).

- **Capa Presentación:** Representa la forma de interactuar el usuario con la aplicación. Refleja las interfaces visuales.
- **Capa de Dominio:** Representa la organización de las clases y relaciones entre estas, reflejando la lógica del sistema.
- **Capa de Persistencia de datos:** Representa las clases persistentes del sistema (plantillas de minucias en estándar ANSI/INCITS 378, a partir de estas se inicia el proceso de reconstrucción)**.**

Figura 19: Arquitectura N-Capas.

El estilo arquitectónico propuesto es **Tuberías y Filtros** para la infraestructura de comunicación, el mismo provee una estructura para procesar flujos de datos. El sistema se percibe como una sucesión de transformaciones que sufre una serie de datos de entrada. Los datos ingresan al sistema y fluyen a través de los componentes uno a uno hasta que se asignan a un destino final: salida o almacenamiento. (Ver figura 20).

Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Cada paso de procesamiento se encapsula en un filtro, cada filtro consume y procesa sus datos en forma incremental. (27)

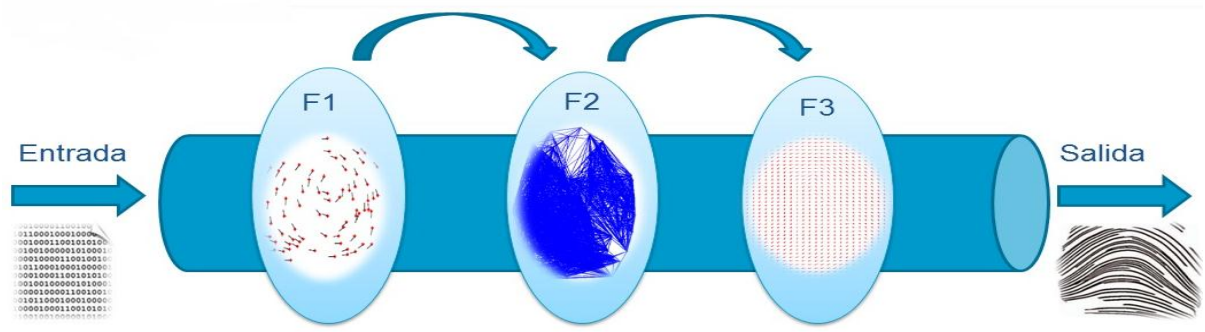


Figura 20: Estilo arquitectónico Tuberías y Filtros.

Estructura:

- **Filtros:**
  - ✓ Enriquece, refina y/o transforma datos.
  - ✓ El procesamiento se inicia cuando :
    - el elemento siguiente solicita datos.
    - el elemento anterior envía datos.

- el filtro tiene un ciclo interno que solicita datos del filtro anterior y envía datos al siguiente.

Tubos:

✓ Conecta origen - filtro, filtro - filtro, y filtro - destino.

2.5.3 Diagrama de clases del diseño

Un diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. Los diagramas de clases del diseño contienen la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Información sobre los tipos de los atributos.
- Navegabilidad.
- Dependencias.

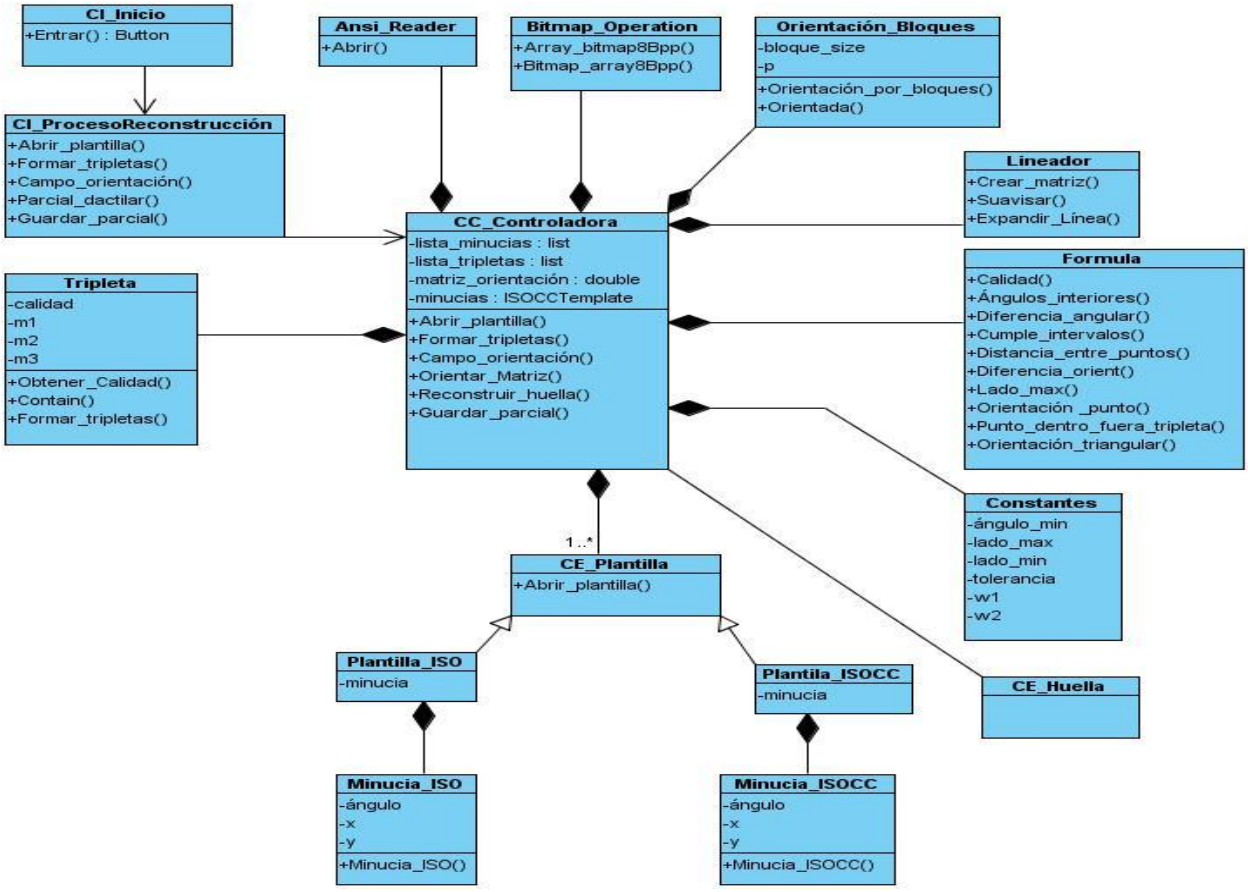


Figura 21: Diagrama de clases del diseño.

2.6 Conclusiones parciales.

El análisis realizado a los procesos involucrados en la reconstrucción de una huella dactilar, provee un mejor entendimiento de la problemática planteada, y facilita el mecanismo apropiado para comprender las características del sistema. El modelo de dominio permitió identificar los principales conceptos relacionados con el dominio de la aplicación y ayudó a comprender el entorno del sistema. A través de la captura de los requisitos funcionales se definieron las funcionalidades que debe cumplir el sistema.

La utilización de los patrones GRASP y GOF permitió realizar una programación organizada. El estudio de las diferentes arquitecturas y estilos arquitectónicos, permitió definir para la estructura del sistema, la arquitectura N-Capas, y como estilo arquitectónico Tuberías y Filtros, ya que el flujo de procesos descrito y la estructura se adecuan al sistema a desarrollar.

A través del modelado del análisis se mostró una visión general del sistema, lo que fue un punto de partida para obtener el diagrama de clases del diseño que constituye una parte fundamental del modelo de diseño, dejando todo listo para la implementación del componente.

Capítulo 3: Implementación y Prueba.

3.1 Introducción.

En este capítulo se muestra la implementación del sistema dándole cumplimiento a los requerimientos planteados al inicio de la investigación, se refleja el diagrama de componente. Se visualizan las interfaces de usuario y se da una explicación de cada una de ellas para un mayor entendimiento acerca del funcionamiento de la aplicación. Se realizan las pruebas donde se valida el funcionamiento de los requisitos funcionales.

3.2 Modelo de despliegue.

Un modelo de despliegue muestra el despliegue físico del sistema en un ambiente de producción (o de prueba). Muestra dónde se ubican los componentes, en qué servidores, máquinas o hardware. (28)

3.2.1 Diagrama de componentes.

El diagrama de componentes ilustra los componentes de software que se usarán para construir el sistema. Muestra la relación entre componentes de software, sus dependencias, su comunicación, su ubicación y otras condiciones. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente, incluyendo componentes de código fuente, componentes del código binario, y componentes ejecutables. (29)

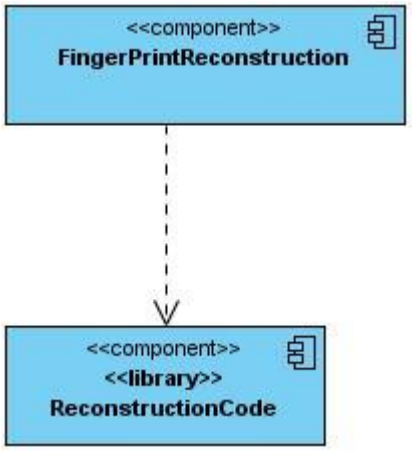


Figura 22: Diagrama de componentes.

3.3 Interfaz de usuario.

La interfaz gráfica es el medio por el cual el usuario interactúa con el sistema, por lo que su diseño debe ser amigable, consistente y que no requiera usuarios con un alto nivel informático. Una aplicación con una interfaz bien diseñada, además de un buen diseño gráfico, debe tener una buena navegabilidad, usabilidad y distribución de los contenidos. Este trabajo utilizará en el diseño los principios antes mencionados.
Después de una breve reseña de la aplicación, se muestran las interfaces para que se tenga un mayor entendimiento de lo que hace el sistema.

En la interfaz principal el usuario selecciona la opción de iniciar el proceso de reconstrucción de huellas. (Ver Fig.23).

Figura 23: Interfaz principal.

Luego se muestra una interfaz donde se puede observar el área de trabajo y desde ella se puede acceder a las diferentes funcionalidades que brinda la aplicación donde el usuario comienza seleccionando la opción de cargar plantilla. (Ver Fig. 24).

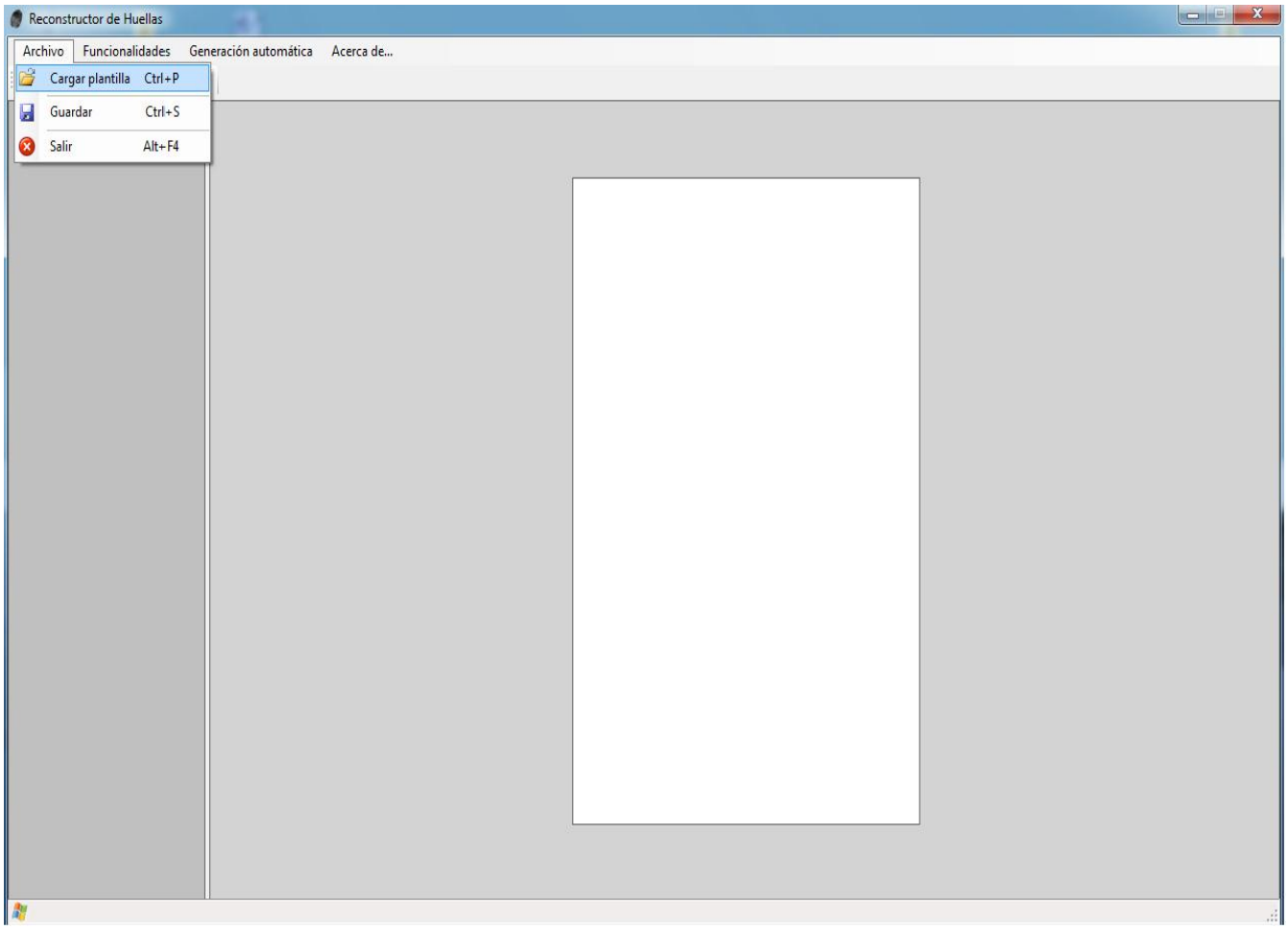


Figura 24: Interfaz área de trabajo.

Se muestra la plantilla de minucias seleccionada y el usuario puede seleccionar la opción de formar tripletas. (Ver Fig. 25)

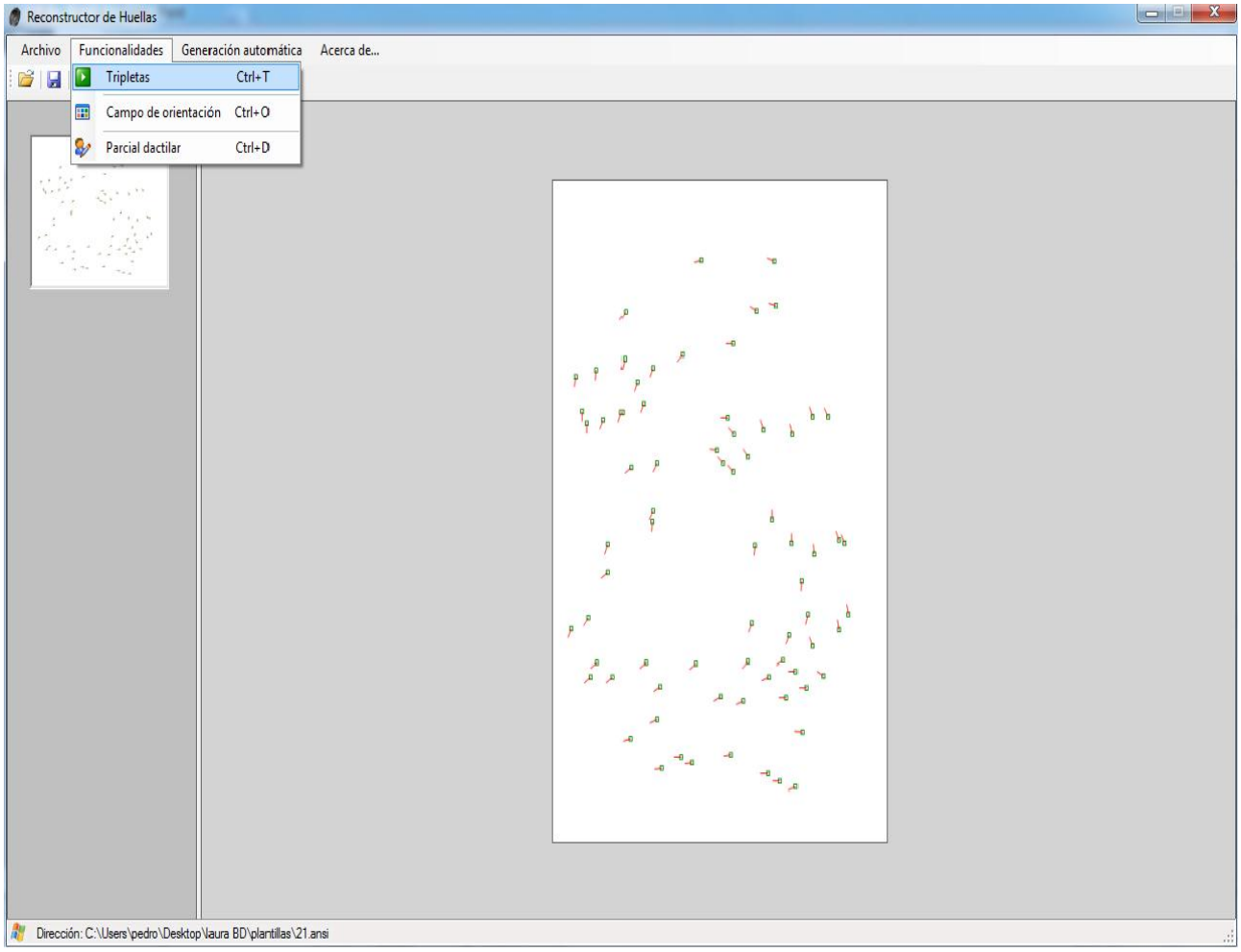


Figura 25: Interfaz mostrar minucias.

Se muestran las tripletas de la plantilla de minucias y el usuario puede seleccionar la opción campo de orientación. (Ver Fig. 26)

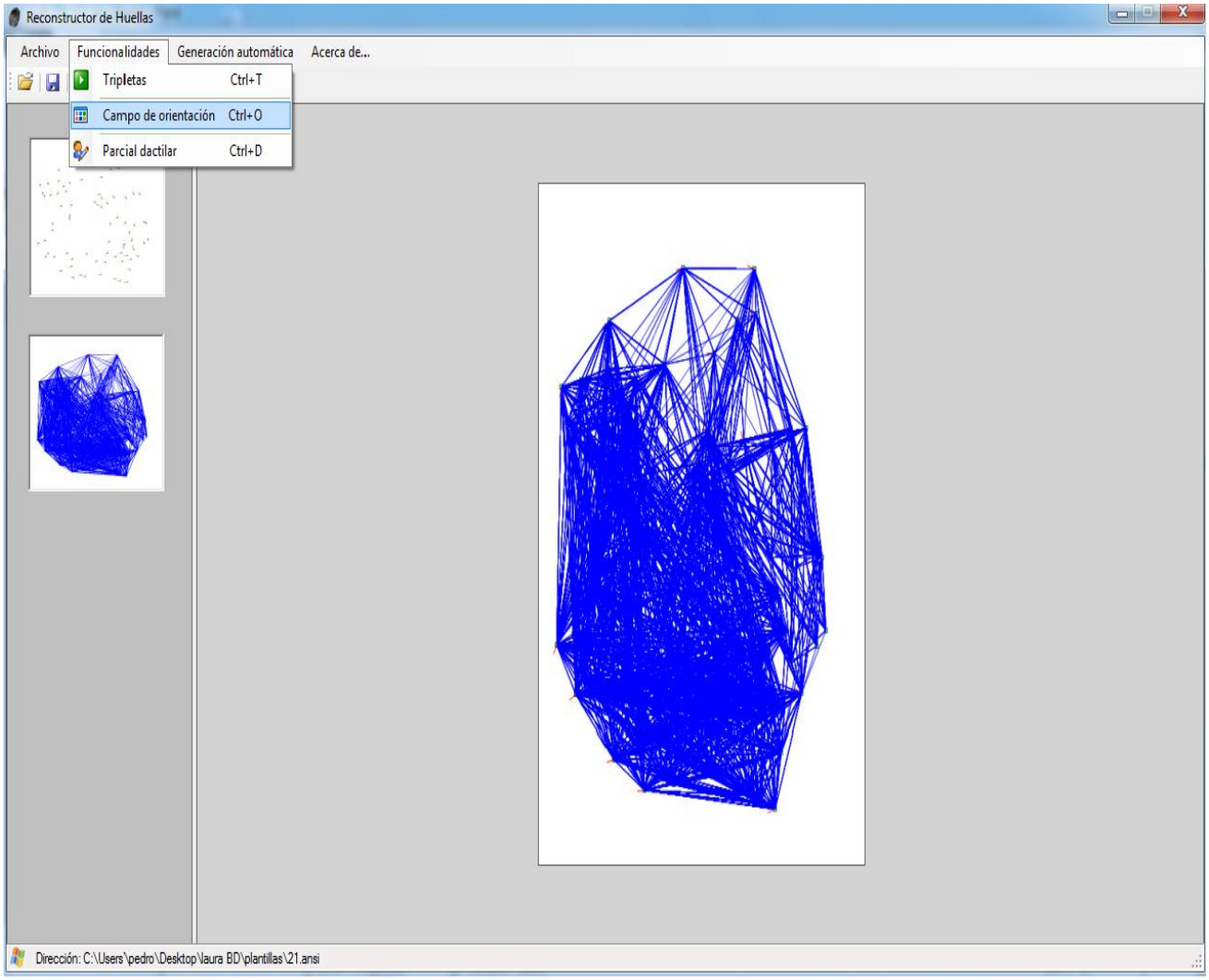


Figura 26: Interfaz mostrar tripletas.

Se muestra el campo de orientación de la huella y luego el usuario puede seleccionar la opción de dibujar el parcial dactilar. (Ver Fig. 27)

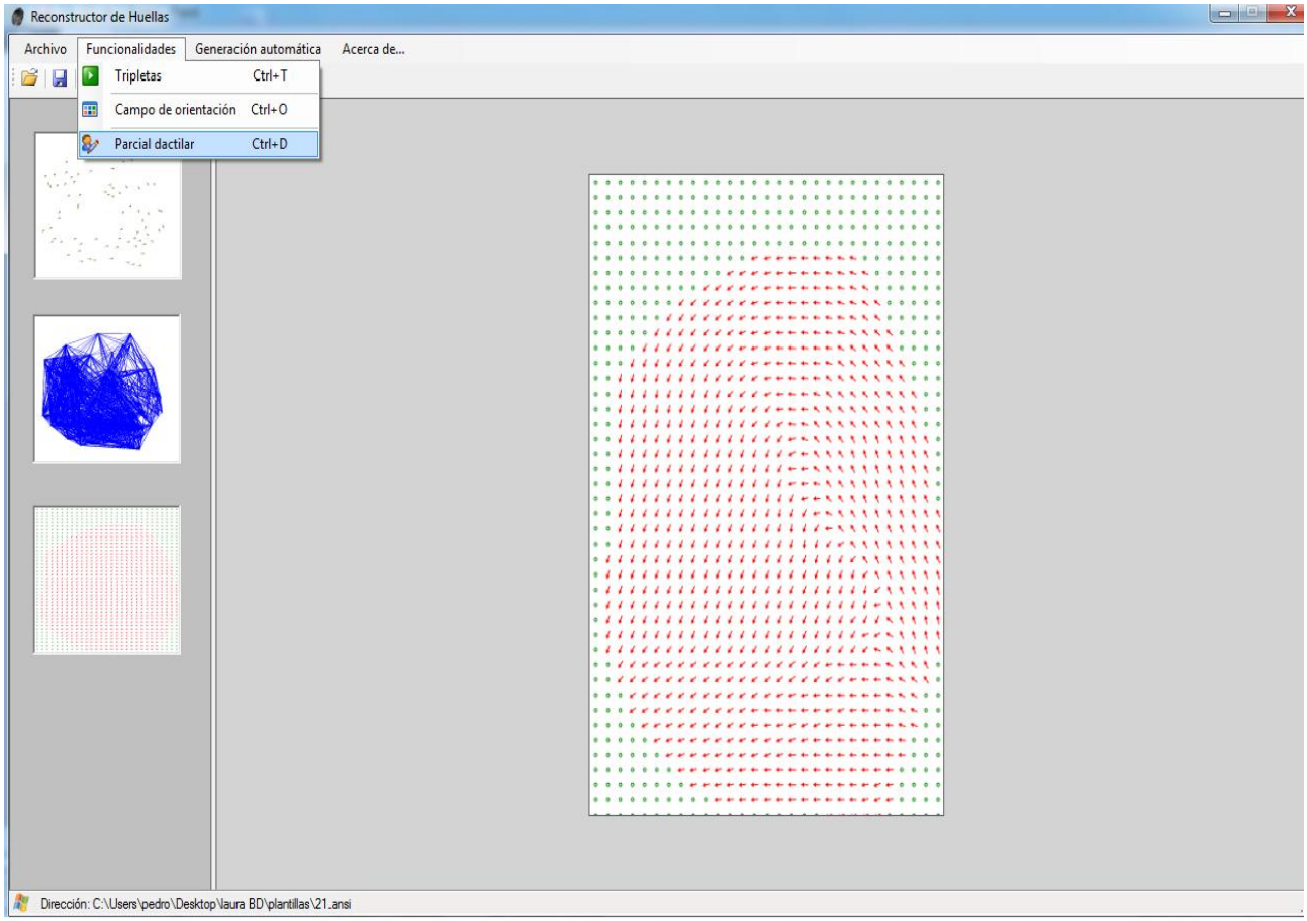


Figura 27: Interfaz campo de orientación.

Se obtiene el parcial de la huella, y el usuario puede seleccionar la opción guardar parcial dactilar. (Ver Fig. 28)

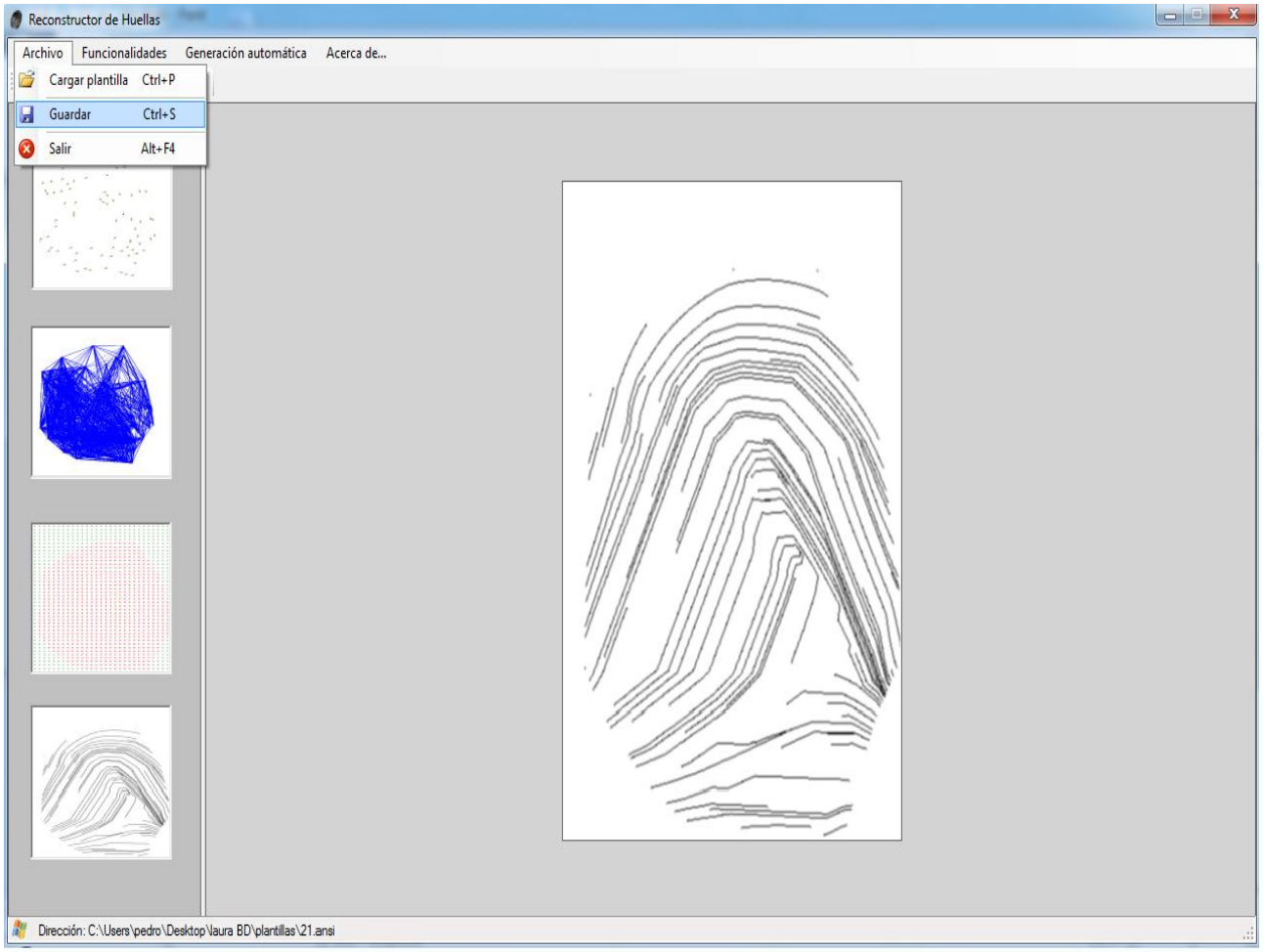


Figura 28: Interfaz parcial dactilar.

Luego se selecciona la ruta destino donde se desea almacenar el parcial de la huella reconstruida y se selecciona el formato en que se desea guardar, y por último se notifica un mensaje confirmando que el parcial ha sido guardado.

A través de la funcionalidad generación automática el usuario podrá reconstruir la cantidad de plantillas de minucias que desee sin necesidad de observar el proceso de reconstrucción, automáticamente se guardarán todos los parciales reconstruidos en una carpeta seleccionada por el usuario. (Ver Fig. 29).

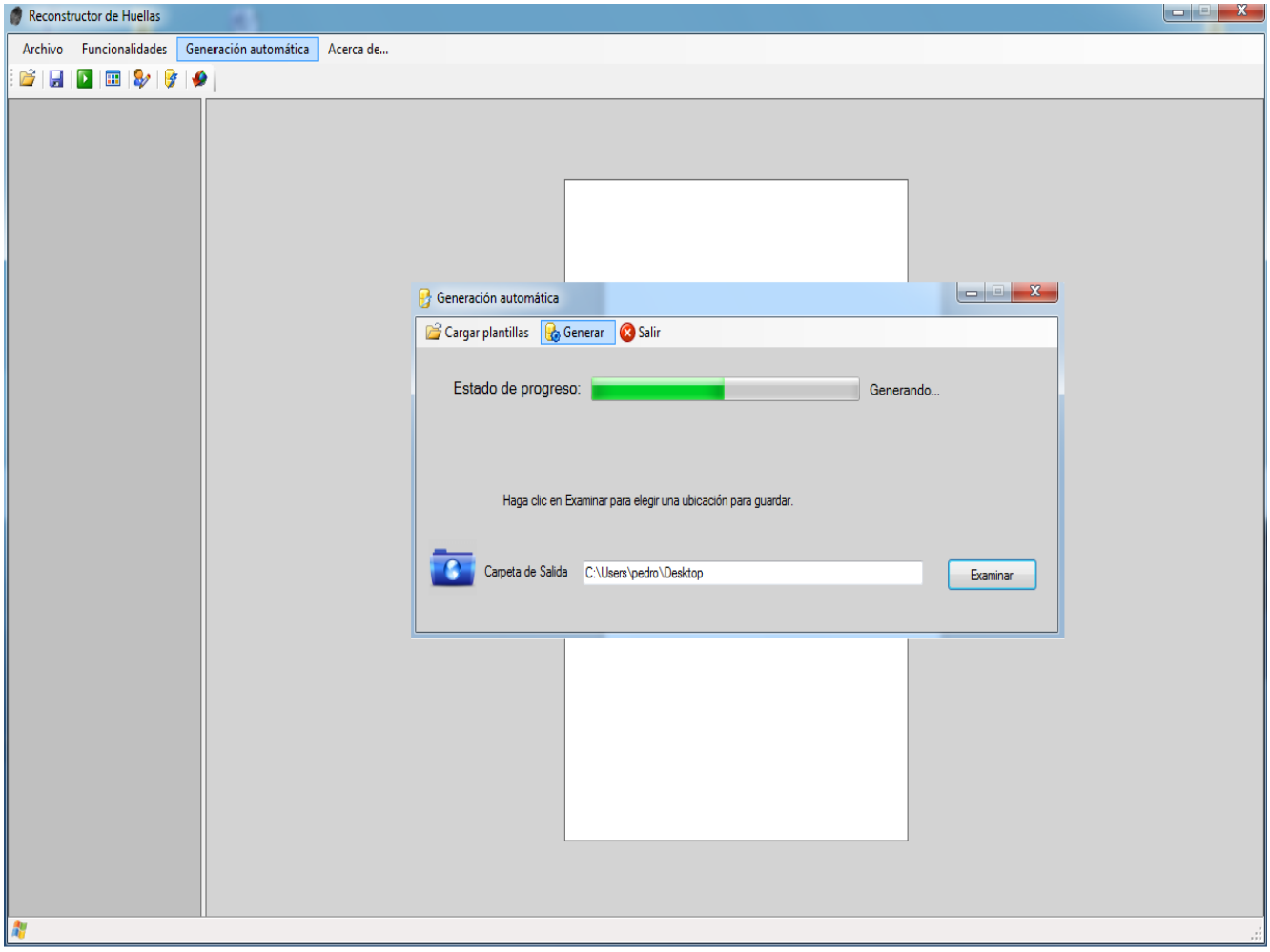


Figura 29: Interfaz generación automática.

3.4 Pruebas y Validación.

La prueba de software es un elemento crítico para la garantía de la calidad del software, constituyen aquellas actividades mediante la cual un sistema es ejecutado bajo determinadas condiciones específicas, enfocada principalmente a la evaluación y determinación de la calidad del producto. (30)

- ✓ Una prueba es un proceso de ejecución de un programa, que conlleva la intención de descubrir un error.
- ✓ Un caso de prueba es bueno cuando su ejecución conlleva una alta probabilidad de encontrar un error.
- ✓ El éxito de la prueba se mide en función de la capacidad de detectar un error no descubierto hasta entonces. (30)

Dentro de las pruebas se encuentran dos fundamentales: Pruebas de Caja Blanca y Pruebas de Caja Negra.

3.4.1 Pruebas de Caja Blanca:

Las pruebas de caja blanca requieren del conocimiento de la estructura interna del programa y son derivadas a partir de las especificaciones internas de diseño o el código. Se examinan los caminos lógicos del software exponiendo que los casos de prueba se realizan juntos, definidos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o mencionado.

La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes. El ingeniero de software puede obtener casos de prueba donde garanticen que.

- ✓ Se ejerciten por lo menos una vez todos los caminos independientes para cada módulo.
- ✓ Se ejerciten las estructuras internas de datos para asegurar su validez.
- ✓ Se ejerciten todas las decisiones lógicas en sus vertientes verdaderas y falsa.
- ✓ Ejecuten todos los bucles en sus límites y con sus límites operacionales. (30)

Prueba del camino básico: Permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. (30)

Los pasos que se siguen para aplicar esta técnica son:

1. A partir del diseño o del código fuente, se dibuja el grafo de flujo asociado.
2. Se calcula la complejidad ciclomática del grafo.
3. Se determina un conjunto básico de caminos independientes.
4. Se preparan los casos de prueba que obliguen a la ejecución de cada camino del conjunto básico. (30)

La complejidad ciclomática de un grafo de flujo V (G) establece el número de caminos independientes. Puede calcularse de tres formas alternativas:

1. V (G) = (A (Aristas) – N (Nodos)) + 2.
2. V (G) = P (Nodos Predicados) + 1.
3. V (G) = R (El número de regiones del grafo de flujo). (30)

- **Funcionalidad: Diferencia angular.**

```
public static double Diferencia_Angular(double a, double b)
    {
        double valor = a - b;
        double c = Math.Abs(valor); //1
        if (c > Math.PI) //2
            c = (2 * Math.PI - c); //3
        return c; //4
    }
```

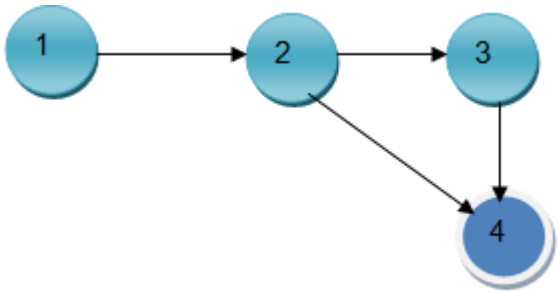


Figura 30: Grafo de flujo funcionalidad Diferencia Angular.

✓ Complejidad ciclomática:

Tabla 6: Complejidad ciclomática para la funcionalidad Diferencia Angular.

Fórmula 1	Fórmula 2	Fórmula 3
V (G) = (A (Aristas) – N (Nodos)) + 2 V (G) = (4 – 4) + 2 = 2	V (G) = P (Nodos Predicados) + 1 V (G) = 1+1=2	V (G) = R = 2

✓ Caminos independientes:

- 1-2-3-4
  - Que el valor de la variable c sea mayor que 3.14.
- 1-2-4
  - Que el valor de la variable c sea menor que 3.14.

➢ **Funcionalidad: Ejecutar acción Formar tripletas.**

```
public static void EjecutarAcción(Form_Principal fp, Controladora c, ISOCCTemplate
template)
{
Graphics graficar;
List<Tripleta> lista_trip = new List<Tripleta>();
lista_trip = c.FormarTripletas();
Bitmap bmp1 = (Bitmap)fp.pb_minucias.Image.Clone();
graficar = Graphics.FromImage(bmp1);
graficar.SmoothingMode = SmoothingMode.AntiAlias;
int longitud = lista_trip.Count; //1
    for (int i = 0; i < longitud; i++) //2
            {
              graficar.DrawLine(Pens.Blue, new Point(lista_trip[i].M1.X, lista_trip[i].M1.Y),
                  new Point(lista_trip[i].M2.X, lista_trip[i].M2.Y));
               graficar.DrawLine(Pens.Blue, new Point(lista_trip[i].M2.X, lista_trip[i].M2.Y),
                  new Point(lista_trip[i].M3.X, lista_trip[i].M3.Y));
               graficar.DrawLine(Pens.Blue, new Point(lista_trip[i].M3.X, lista_trip[i].M3.Y),
                  new Point(lista_trip[i].M1.X, lista_trip[i].M1.Y)); //3
             }
```

```
        fp.pb_tripletas.Image = bmp1; //4
}
```

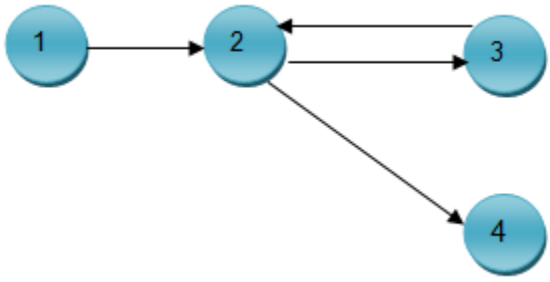


Figura 31: Grafo de flujo funcionalidad formar tripletas.

✓ Complejidad ciclomática:

Tabla 7: Complejidad ciclomática para la funcionalidad Ejecutar Formar tripletas.

Fórmula 1	Fórmula 2	Fórmula 3
V (G) = (A (Aristas) – N (Nodos)) + 2 V (G) = (4 – 4) + 2 = 2	V (G) = P (Nodos Predicados) + 1 V (G) = 1 + 1 = 2	V (G) = R = 2

✓ Caminos independientes:

- 1-2-3-2-4
  - Que el valor de la variable i sea menor que el valor de la variable longitud.
- 1-2-4
  - Que el valor de la variable i sea mayor que el valor de la variable longitud.

➢ **Funcionalidad: Ejecutar Acción Guardar Imagen.**

```
public static void EjecutarAcción(Form_Proceso fp)
    {
        SaveFileDialog sf = new SaveFileDialog();
        string text = "Parcial";
        sf.FileName = text;
        sf.Filter = "Imagen BMP(*.bmp)|*.bmp|Imagen JPG(*.jpg)|*.jpg|Imagen
TIFF(*.tif)|*.tif|Imagen PNG(*.png)|*.png"; //1
```

```
            if (sf.ShowDialog() == DialogResult.OK) //2
            {
                ImageFormat formato = ImageFormat.Bmp; //3
                if (Path.GetExtension(sf.FileName) == ".png") //4
                    formato = ImageFormat.Png; //5
                if (Path.GetExtension(sf.FileName) == ".tif") //6
                    formato = ImageFormat.Tiff; //7
                if (Path.GetExtension(sf.FileName) == ".jpg") //8
                    formato = ImageFormat.Jpeg; //9
                fp.dibujarOrientacion2.Image.Save(sf.FileName, formato); //10
            }
}
```

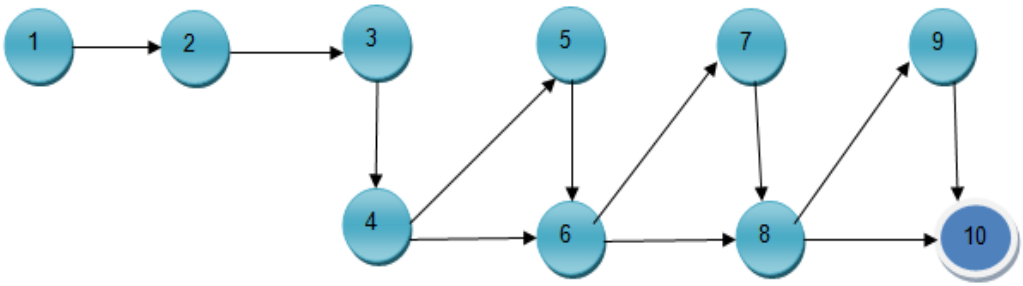


Figura 32: Grafo de flujo funcionalidad guardar imagen.

- ✓ Complejidad ciclomática:

Tabla 8: Complejidad ciclomática para la funcionalidad Guardar Imagen.

Fórmula 1	Fórmula 2	Fórmula 3
V (G) = (A (Aristas) – N (Nodos)) + 2 V (G) = (12 – 10) + 2 = 4	V (G) = P (Nodos Predicados) + 1 V (G) = 3 + 1 = 4	V (G) = R = 4

- ✓ Caminos independientes:

- 1-2-3-4-5-6-8-10
  - Que se seleccione algún archivo.
  - Que la extensión del archivo seleccionado sea igual a png.
  - Que la extensión del archivo seleccionado sea distinto de tif.

- Que la extensión del archivo seleccionado sea distinto de jpg.

- 1-2-3-4-6-7-8-10
  - Que se seleccione algún archivo.
  - Que la extensión del archivo seleccionado sea distinto de png.
  - Que la extensión del archivo seleccionado sea igual a tif.
  - Que la extensión del archivo seleccionado sea distinto de jpg.

- 1-2-3-4-6-8-9-10
  - Que se seleccione algún archivo.
  - Que la extensión del archivo seleccionado sea distinto de png.
  - Que la extensión del archivo seleccionado sea distinto de tif.
  - Que la extensión del archivo seleccionado sea igual a jpg.

- 1-2
  - Que no se seleccione ningún archivo.

➢ **Funcionalidad: Ejecutar Acción Abrir.**

```
public static void EjecutarAcción(Form_Principal fp,ref Controladora c, AnsiReader ar,
ref ISOCCTemplate template ,ref ISOCCTemplate original, Bitmap bmp)
    {
        fp.pb_tripletas.Image = null;
       Graphics graficar;
       OpenFileDialog abrir = new OpenFileDialog();
       abrir.Filter = "Archivos de Plantillas | *.ansi"; //1
       if (abrir.ShowDialog() == DialogResult.OK) //2
      {
          ar.Open(abrir.OpenFile());
          template = ar.GetFinger(0, true);
          original = ar.GetFinger(0, false);
          fp.lb_Dirección.Text = "Dirección:" + " " + abrir.FileName;
          fp.lb_Dirección.Visible = true;
          List<int> cordenadas = Formula.Valores_Coordenadas(template.Minutias);
          bmp = new Bitmap(cordenadas[1], cordenadas[3]);
```

```
            c = new Controladora(template, bmp);
            bmp = new Bitmap(cordenadas[1] + 50, cordenadas[3] + 50);
            graficar = Graphics.FromImage(bmp);
            graficar.SmoothingMode = SmoothingMode.AntiAlias;
            int longitud = template.Minutias.Length; //3
            for (int i = 0; i < longitud; i++) //4
            {
                graficar.DrawRectangle(Pens.Green, template.Minutias[i].X - 2,
template.Minutias[i].Y - 2, 4, 4);
                graficar.DrawLine(Pens.Red, new Point(template.Minutias[i].X,
template.Minutias[i].Y),
                Angulo(template.Minutias[i].Angle, new Point(template.Minutias[i].X,
template.Minutias[i].Y))); //5
            }
            fp.pb_minucias.Image = bmp; //6
        }       }
```

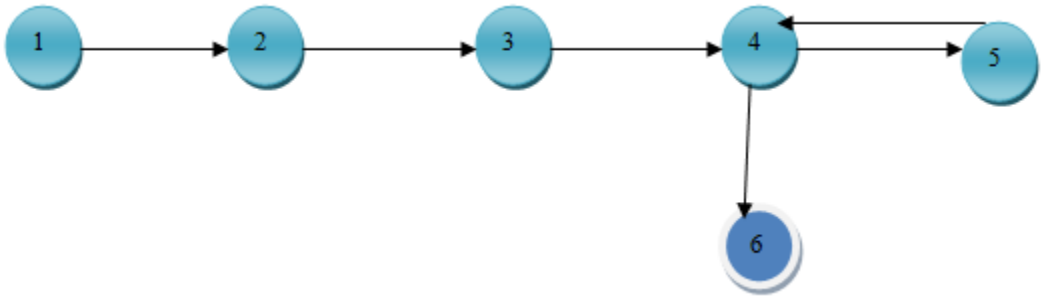


Figura 33: Grafo de flujo funcionalidad EjecutarAbrir.

- ✓ Complejidad ciclomática:

Tabla 9: Complejidad ciclomática para la funcionalidad EjecutarAbrir.

Fórmula 1	Fórmula 2	Fórmula 3
V (G) = (A (Aristas) – N (Nodos)) + 2 V (G) = (6 – 6) + 2 = 2	V (G) = P (Nodos Predicados) + 1 V (G) = 1 + 1 = 2	V (G) = R = 2

- ✓ Caminos independientes:

- 1-2-3-4-5-4-6
    - Que se seleccione algún archivo.
    - Que el valor de la variable i sea menor que la variable longitud.

- 1-2-3-4-6
    - Que se seleccione algún archivo.
    - Que el valor de la variable i sea mayor que la variable longitud.

Para cada funcionalidad en particular se realizó el cálculo de la complejidad ciclomática mediante las fórmulas descritas, por lo que ha dado el mismo valor, lo que significa que existen esos posibles caminos por donde el flujo puede circular.

3.4.2 Pruebas de Caja Negra.

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre las funcionalidades del software. A través de los casos de prueba se demuestra que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene.

Tabla 10: Pruebas de Caja Negra. SC<Reconstruir huella dactilar>

Escenario	Descripción	Var válidas	Var inv.	Respuesta del sistema	Flujo central
EC1.1 Cargar plantilla	Permite cargar una plantilla de minucias en estándar ANSI, y posteriormente se visualizará la plantilla de minucias seleccionada.	Plantilla de minucias en estándar ANSI.		Se muestra la plantilla de minucias seleccionada.	1. Seleccionar la opción Cargar plantilla. 2. Seleccionar la plantilla que se desea reconstruir y ejecutar la opción Abrir.
EC 1.2 Formar tripletas	Permite formar y visualizar las tripletas de	Plantilla de minucias en estándar		Se visualizan las tripletas de minucias.	1. Se tiene que haber cargado la plantilla de minucias.

	minucias correspondientes a la plantilla de minucias cargada.	ANSI.			2. Seleccionar la opción Formar tripletas.
EC 1.2 Formar tripletas	Permite formar y visualizar las tripletas de minucias correspondientes a la plantilla de minucias cargada.		Tripleta	Se muestra un mensaje de error ´´Se debe cargar las plantilla de minucias antes de formar tripletas´´	1. Seleccionar la opción Formar tripletas.
EC 1.3 Campo de orientación.	Permite estimar y visualizar el campo de orientación correspondiente a la plantilla de minucias cargada.	Tripletas de minucias.		Se visualiza el campo de orientación de la plantilla de minucias.	1. Se tiene que haber cargado la plantilla de minucias y haber formado las tripletas. 2. Seleccionar la opción Estimar campo de orientación.
EC 1.4 Reconstruir Parcial	Permite reconstruir y visualizar el parcial dactilar correspondiente a la plantilla de minucias.	Campo de orientación de la plantilla de minucias.		Se muestra el parcial dactilar.	1. Se tiene que haber cargado la plantilla de minucias, haber formado las tripletas y estimado el campo de orientación. 2. Seleccionar la opción Reconstruir parcial.

EC 1.5 Guardar parcial dactilar.	Permite almacenar el parcial reconstruido en el formato BMP, JPG, PNG o TIFF.	Parcial dactilar		Se almacena el parcial reconstruido y se muestra un mensaje de confirmación “El parcial ha sido guardado satisfactoria mente ”	1. Se tiene que haber reconstruido el parcial dactilar. 2. Se selecciona la opción guardar parcial. 3. Se selecciona la ruta destino donde se desea almacenar la huella reconstruida y el formato BMP, JPG, PNG o TIFF del parcial reconstruido.

3.4.3 Validación

Para la realización de las validaciones se obtuvo una huella original y a partir de la plantilla de minucias correspondiente a la huella original se reconstruyó el parcial dactilar usando la propuesta de solución desarrollada, donde se observó gran similitud en el trazo de las crestas de las mismas.

Figura 34: Huella original y parcial reconstruido.

Se utilizó el software Image Processing Lab, realizado por Andrew Kirillov, basado en Aforge.NET framework, para superponer la huella original con el parcial reconstruido.

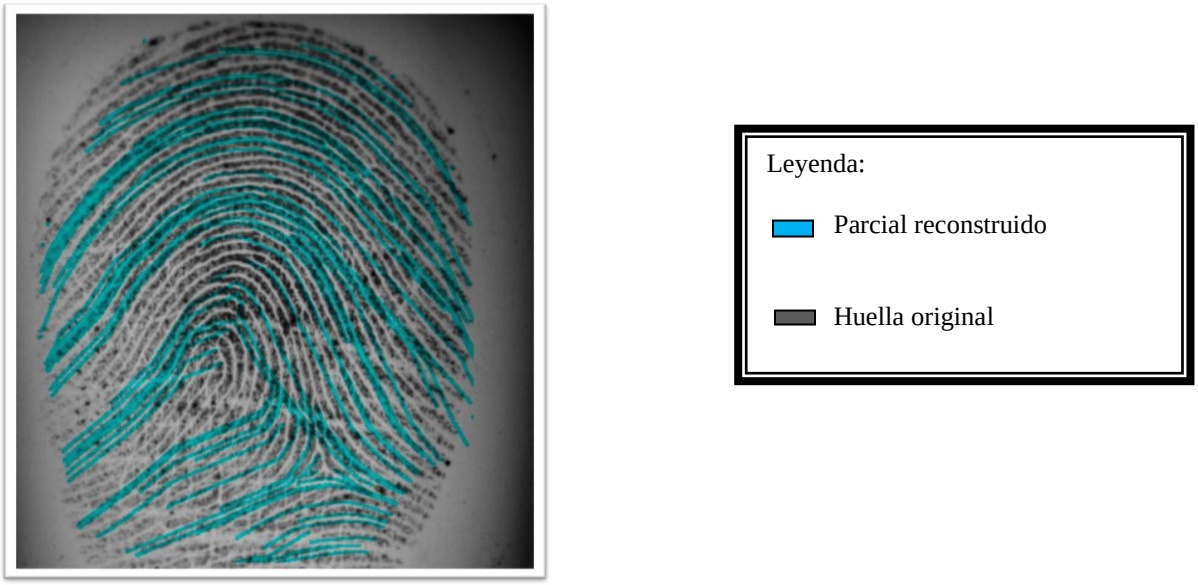


Figura 35: Huellas intersectadas.

Resultados arrojados:

- ✓ Se reconstruyeron 100 imágenes de huellas a partir de plantillas ANSI, de la base de datos FVC2006 Innovatrics y Dermalog.
- ✓ Se observó que de las 100 huellas originales, 82 presentaron similitud con las huellas reconstruidas.

Además se tomó una población total de 40 huellas originales y 40 huellas reconstruidas, exactamente 10 huellas por cada tipo de huella dactilar, correspondientes con los cuatro tipos de huellas principales: Espiral (E), Arco (A), Lazo izquierdo (LI) y Lazo derecho (LD) de la base de datos FVC 2006 Dermalog.

Para la realización de las pruebas se utilizó la librería **SourceAfis** en su versión 1.6.0, haciendo uso de los algoritmos de extracción y macheo presentes en dicha librería, se pudo obtener los scores (valor de similitud) de las huellas reconstruidas respecto a las huellas originales. Se realizó un promedio de los scores arrojados por las 10 huellas seleccionadas por cada tipo de huella dactilar obteniendo los siguientes resultados (E: 6.75, A: 9.34, LI: 7.29 y LD: 8.09). (Ver Fig. 36).

Un análisis realizado por especialistas a los algoritmos de extracción y macheo presentes en la librería SourceAFis arrojó que el umbral mínimo de macheo para burlar un sistema de identificación dactilar de baja seguridad es 6,666667. (31)

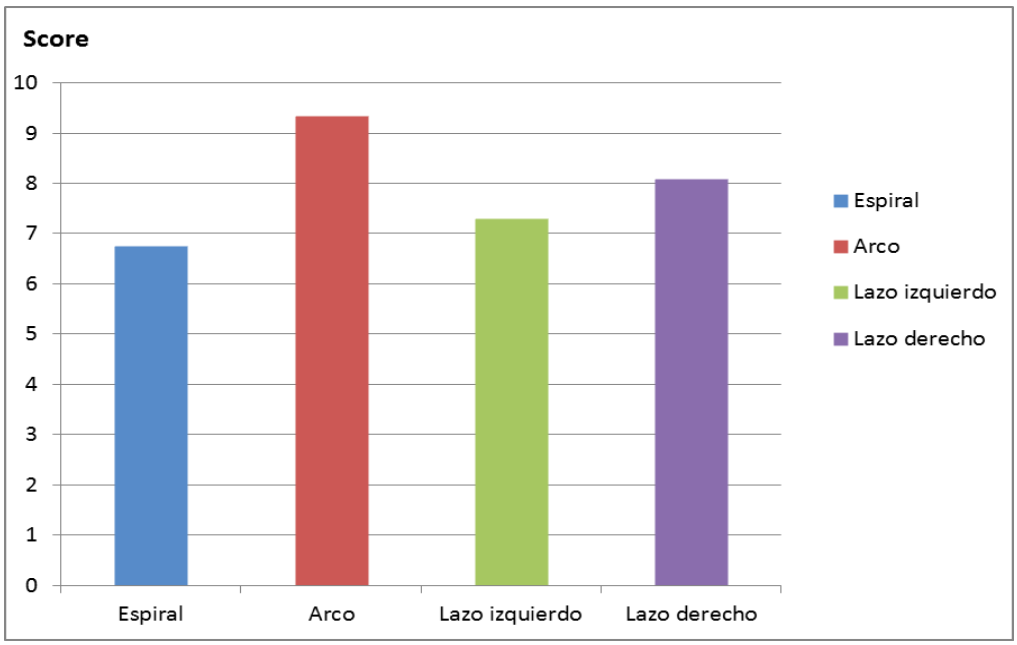


Figura 36: Gráfica de relación entre score y tipo de huella dactilar.

En la gráfica se muestra que todas las huellas reconstruidas sobrepasan el umbral mínimo, arrojando mejores resultados las de tipo Arco con un 9.34 score generado.

De esta manera queda evidenciado que todos los parciales reconstruidos son capaces de burlar la seguridad de acceso de un sistema de identificación dactilar de baja seguridad, contribuyendo este trabajo a probar y mejorar la seguridad de acceso de dichos sistemas.

3.5 Conclusiones parciales.

En el presente capítulo se muestra el proceso de implementación del sistema mediante las interfaces visuales. Las pruebas realizadas al software permitieron evaluar el funcionamiento interno del sistema y detectar errores no encontrados. La intersección de una huella original con el parcial reconstruido permitió observar la similitud de ambas y haciendo uso de los algoritmos de extracción y macheo presentes en la librería **SourceAfis**, fue posible la validación del software quedando evidenciado que los parciales reconstruidos son capaces de burlar la seguridad de acceso de los sistemas de identificación dactilar de baja seguridad.

Conclusiones Generales.

En el presente trabajo se realizó el estudio de los sistemas biométricos basados en huellas dactilares existentes en el mundo y en nuestro país, obteniendo como resultado un conocimiento profundo de las tendencias actuales en este campo. En la investigación realizada no se encontraron evidencias de sistemas que realicen el proceso de reconstrucción de imágenes de huellas dactilares a partir de plantillas de minucias en estándar ANSI, por lo cual surge la necesidad de desarrollar una aplicación que realice dicho proceso y de esta manera contar con datos de prueba para probar sistemas biométricos basados en huellas dactilares y aumentar los niveles de seguridad de los mismos.

- El análisis de las diferentes metodologías, herramientas y tecnologías hizo posible la selección adecuada de las mismas para el desarrollo del sistema.
- La realización de los diferentes diagramas permitió obtener una visión detallada del sistema.
- La selección correcta de la arquitectura y los patrones de diseño permitió implementar un componente organizado.
- La realización de las pruebas de caja blanca y caja negra demostraron el correcto funcionamiento del sistema.
- La validación realizada al parcial reconstruido arrojó scores que indican que el parcial es capaz de burlar la seguridad de acceso de sistemas de identificación dactilar de baja seguridad.

Por todo lo anterior se concluye que los objetivos propuestos para el presente trabajo han sido cumplidos satisfactoriamente, logrando un componente de reconstrucción de imágenes parciales de huellas dactilares a partir de plantillas de minucias en formato ANSI/INCITS 378 para el beneficio de las instituciones nacionales que precisen de este tipo de herramienta, lo que ahorraría cuantiosos gastos al país.

Además se sentaron las bases para trabajos futuros, que evolucionen en el tema de reconstrucción de huellas dactilares.

Recomendaciones.

- ✓ Realizar optimizaciones al proceso según avance el desarrollo de este campo.
- ✓ Realizar la implementación del algoritmo de reconstrucción en el lenguaje C++ para aumentar su eficiencia.

Referencias bibliográficas.

1. **Fernando Martín Rodríguez, Fransico Javier Suárez López.** Identificación Dactilar basada en filtros Gabor.

2. **Subcomité de Biometría del NSTC**. Biometría. [En línea] Agosto de 2006. [Citado el: 21 de Noviembre de 2011.] http://www.biometria.gov.ar/metodos-biometricos/dactilar.aspx

3. **Victoria Alexandra Hidalgo Jácome**. Implementación de un sistema de autenticación biométrica basado en huellas dactilares. Riobamba, Ecuador, 2010.

4. **Gerardo E. Canedo, Ma. de Guadalupe García, Heriberto Gutiérrez, Noé Mosqueda**. Aplicación del NFIS para la extracción de características de huellas dactilares. Guanajato, México : s.n., 2005. 001.

5**. Aradí Rosales Cruz.** Clasificación de huellas digitales mediante minucias. Instituto Nacional de Astrofísica, Óptica y Electrónica, 2009.

6. Biometric System Laboratory. Universidad de Bologna. [En línea] [Citado el: 8 de Diciembre de 2011.] http://biolab.csr.unibo.it/research.asp

7. **Jianjiang Feng, Anil K. Jain.** FM Model Based Fingerprint Reconstruction from Minutia Template. Michigan State : s.n., 2009.

8. **Arun Ross , Jidnya Shah, Anil K. Jain**. From Template to Image: Reconstructing fingerprints from Minutia Point. 4, 2007, Vol. 29.

9. **Roberth F. Figueroa, camilo J. Solís, Armando A. Cabrera**. Metodologías tradicionales vs metodlogías ágiles.

10. **Patricio Letelier.** Rational Unified Process (RUP). Departamento Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.

11. **José H. Canós, Patricio Letelier, María del Carmen Penadés**. Metodologías ágiles en el proceso en el desarrollo de software. Valencia.

12. **James Rumbauhg, Ivar Jacobson, Grady Booch**. El lenguaje Unificado de Modelado. Manual de referencia. Addison Wesley.

13. **Pearson Educación S.A, Addison Wesley**. Ingeniería del Software. Ian Sommerville: séptima edición, Madrid,España, 2005.

14. Visual Paradigm for UML. [En línea] [Citado el: 2 de Febrero de 2012.] http://www.visual-paradigm.com/product/vpuml/editions/modeler.jsp.

15. Ingeniería del Software. Facilitador de Ingeniería de software. [En línea] [Citado el: 15 de Febrero de 2012.] http://ingenieriasoftware2011.wordpress.com/2011/07/08/herramientas-case-ejemplos-para-evaluarlas/

16. Lenguajes de programación. [En línea] [Citado el: 27 de Febrero de 2012.] http://www.lenguajes-de-programacion.com/lenguajes-de-programacion.shtml

17. Curso C++. [En línea] [Citado el: 3 de Marzo de 2012.] http://www.zator.com/Cpp/E1_2.htm

18. **Armando Sánchez.** Tutorial visual C#. Instituto Tecnológico Superior de Tepeaca. Vol 1, 2010

19. **Jesús Pando Barrón.** Manual de Java. [En línea] [Citado el: 8 de Marzo de 2012.] http://uploading.com/files/QXAR1BU6/ManualJava.rar.html

20. Programación Desarrollo.es. [En línea] [Citado el: 8 de Marzo de 2012.] http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollo-integrado-ide/

21. **Comunidad de desarrollo de Eclipse IDE.** Plataforma Eclipse. [En línea] [Citado el: 8 de Marzo de 2012.] http://plataformaeclipse.com/

22. **Javier Antonio Ballesteros Ricaurter.** Java, 2009. [En línea] [Citado el: 8 de Marzo de 2012.] http://javierballesteros-uptc.weebly.com/uploads/1/9/6/8/1968265/ java.pdf

23. **Tomás Gregorio Serrano Escamilla**. El entorno de desarrollo Visual Studio.Net. Ciudad Universitaria, Coyoacán, México. [En línea] [Citado el: 8 de Marzo de 2012.] http://es.scribd.com/doc/52374583/18/El-entorno-de-desarrollo-de-Visual-Studio-NET

24. **James Rumbauhg, Ivar Jacobson, Grady Booch.** El Proceso Unificado de Desarrollo de Software.

25. **Larman, Craig.** UML y Patrones. Introducción al análisis y diseño orientado a objetos. Prentice Hall, México: Mexicana, 1999. 1524.

26. **César de la Torre Llorente, Unai Zorrilla Castro, Miguel Angel Ramos, Javier Calvarro**. Guía de Arquitectura N-Capas orientada al Dominio con NET 4.0.

27. Estilos y Patrones en la Estrategia de Arquitectura de Microsoft. [En línea] [Citado el: 8 de Marzo de 2012.] http://carlosreynoso.com.ar/archivos/arquitectura/Estilos.PDF

28. **Departamento de Sistemas Informáticos.** Modelo de Implementación: Diagramas de Componentes y Despliegue. [En línea] [Citado el: 5 de Abril de 2012.] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf.

29. **Geoffrey Sparks.** El Modelo de Componentes. Una Introducción al UML. Sparx Systems, Australia.

30. **Patricio Letelier**. Pruebas del Software. Departamento Sistemas Informáticos y Computación, Universidad Politécnica de Valencia.
31. **Departamento de Biometría**. Pruebas realizadas a la librería de clases: SourceAfis. Universidad de las Ciencias Informáticas. C. Habana, Cuba.
32. **Unidad Docente de Ingeniería del Software.** Patrones del "Gang of Four".Facultad de informática, Universidad Politécnica de Madrid.

Bibliografías consultadas.

- **Arun Ross, Jidnya Shaha , Anil K. Jain.** Towards Reconstructing Fingerprints From Minutiae Points, Orlando, USA, 2005.
- **Jianjiang Feng, Anil K. Jain.** Fingerprint Reconstruction: From Minutiae to Phase. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 33, No. 2, 2011.
- **Marcos Martínez Días.** Vulnerabilidades en sistemas de reconocimiento basado en huellas dactilar: Ataques Hill-Climbing. Universidad autónoma de Madrid, España, 2006.
- **Colunga Ruíz, A. Padilla Vivanco.** Reconstrucción de Imágenes digitales mediante Momentos de Legendre y Tchebyshev.
- Fingerprint Recover 3.0. Sistema de Recuperación y Enrolamiento de Huellas Dactilares. Buenos Aires, Argentina, 2002.
- **César Tolosa Borja, Álvaro Giz Bueno.** Sistemas biométricos.
- **Jorge Ormachea.** Seguridad en los sistemas biométricos. 2008
- **Raffaele Cappelli.** SFinGe: an Approach to Synthetic Fingerprint Generation.
- **Pressman.** Software e Ingeniería del Software. http://www.mhhe.com/pressman.
- **Abel Ernesto Sánchez Alí, Raúl Angel Ballester Mena**. Componente para la Comparación de Huellas Dactilares en Tarjetas Inteligentes. Universidad de las Ciencias Informáticas, La Habana, 2010
- **Fabián Humberto Herrera**. Detección de detalles en huellas dactilares usando redes neuronales. Bogotá, 2005
- **Ramón Santana, Yusnieira Reyes**. Sintetizador de Huellas Dactilares. Universidad de las Ciencias Informáticas, La Habana, 2011
- **Alicia Hortensia Beisner Muñoz**. Ataques tipo Side-Chanel a sistemas biométricos de huella dactilar. Universidad autónoma de Madrid, España, 2011

- **Marcos Faúndez Zanuy, Sergio Osuna Silvestre.** Experimentos sobre la vulnerabilidad de sistemas biométricos, Barcelona, España

Glosario de siglas y términos.

RUP: Proceso Unificado de Rational (en inglés Rational Unified Process). Es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

UML: Lenguaje Unificado de Modelado (en inglés Unified Modeling Language). Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.

Case: Ingeniería de Software Asistida por Ordenador (en inglés Computer Aided Software Engineering.

GRASP: Patrones de asignación de responsabilidades. Acrónimo de General Responsibility Assignment Software Patterns.

GoF: Banda de los Cuatro (Gang of Four). Es el nombre con el que se conoce comúnmente a los autores del libro Patrones de Diseño, el cual hace referencia específicamente al diseño orientado a objetos.

Plantilla: Es un medio que permite guiar, portar o construir un diseño o esquema predefinido. Las plantillas, como norma general, pueden ser utilizadas por personas o por sistemas automatizados. Una plantilla puede servir como muestra base de una diversidad sobre la que comparten elementos comunes (patrón) y que en sí es lo que constituye la plantilla.

ANSI: Instituto Nacional Estadounidense de Estándares (viene de las siglas en inglés de American National Standards Institute), el cual es una organización encargada de

supervisar el desarrollo de normas para los servicios, productos, procesos y sistemas en los Estados Unidos. El ANSI forma parte de la ISO y de la IEC, Organización Internacional para la Estandarización y de la Comisión Electrotécnica Internacional.

INCITS: Comité Internacional para estándares de Tecnología de Información (viene de las siglas en inglés InterNational Comittee for Information Technology Standards).

FVC: Competencia Internacional de Algoritmos de verificación de huellas dactilares. (Fingerprint Verification Competition).

Algoritmo: Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema. Un conjunto prescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien lo ejecute.

Anexos.

Anexo1. Actores y resumen de los Casos de Uso del sistema.

Actores del sistema.

Actores del Sistema	Justificación
Usuario	Representa la persona que es la encargada de hacer todos los procesos que se llevan a cabo para la reconstrucción de huellas dactilares.

Resumen del Caso de Uso: Reconstruir huella dactilar.

CU1	Reconstruir huella dactilar.
Actor	Usuario
Descripción	El usuario selecciona la opción reconstruir huella dactilar para obtener una imagen parcial de la huella.
Referencia	RF-1, RF-1.1, RF-1.2, RF-1.3, RF-1.4 RF-1.5.

Resumen del Caso de Uso: Generar parciales.

CU1	Generar parciales.
Actor	Usuario
Descripción	El usuario selecciona la opción de generación automática para obtener un conjunto de imágenes parciales de huellas reconstruidas.
Referencia	RF-2

Anexo 2. Diagramas de interacción.

Diagrama de colaboración del CUS: Reconstruir Huella Dactilar.

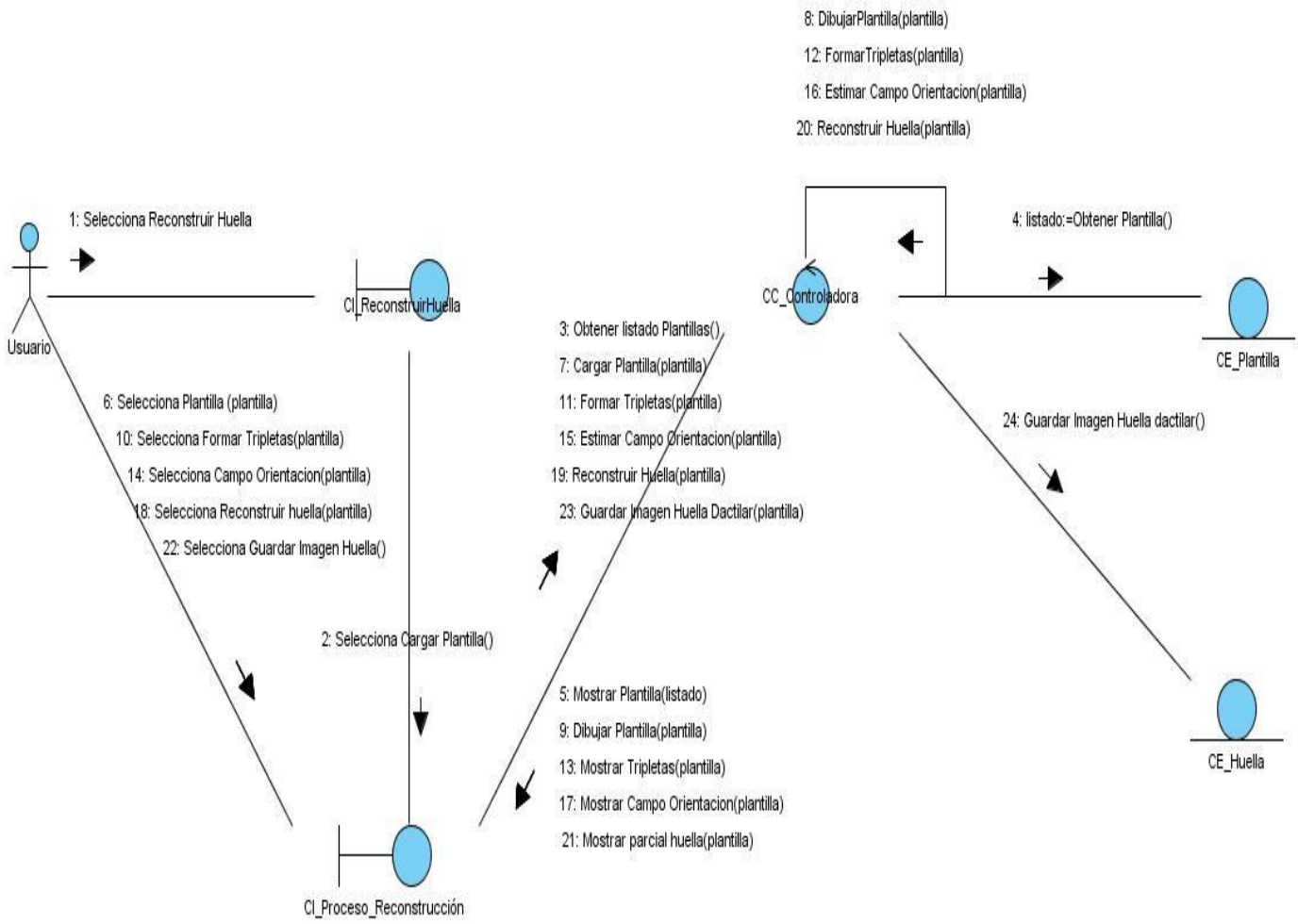


Diagrama de colaboración del CUS: Generar parciales.

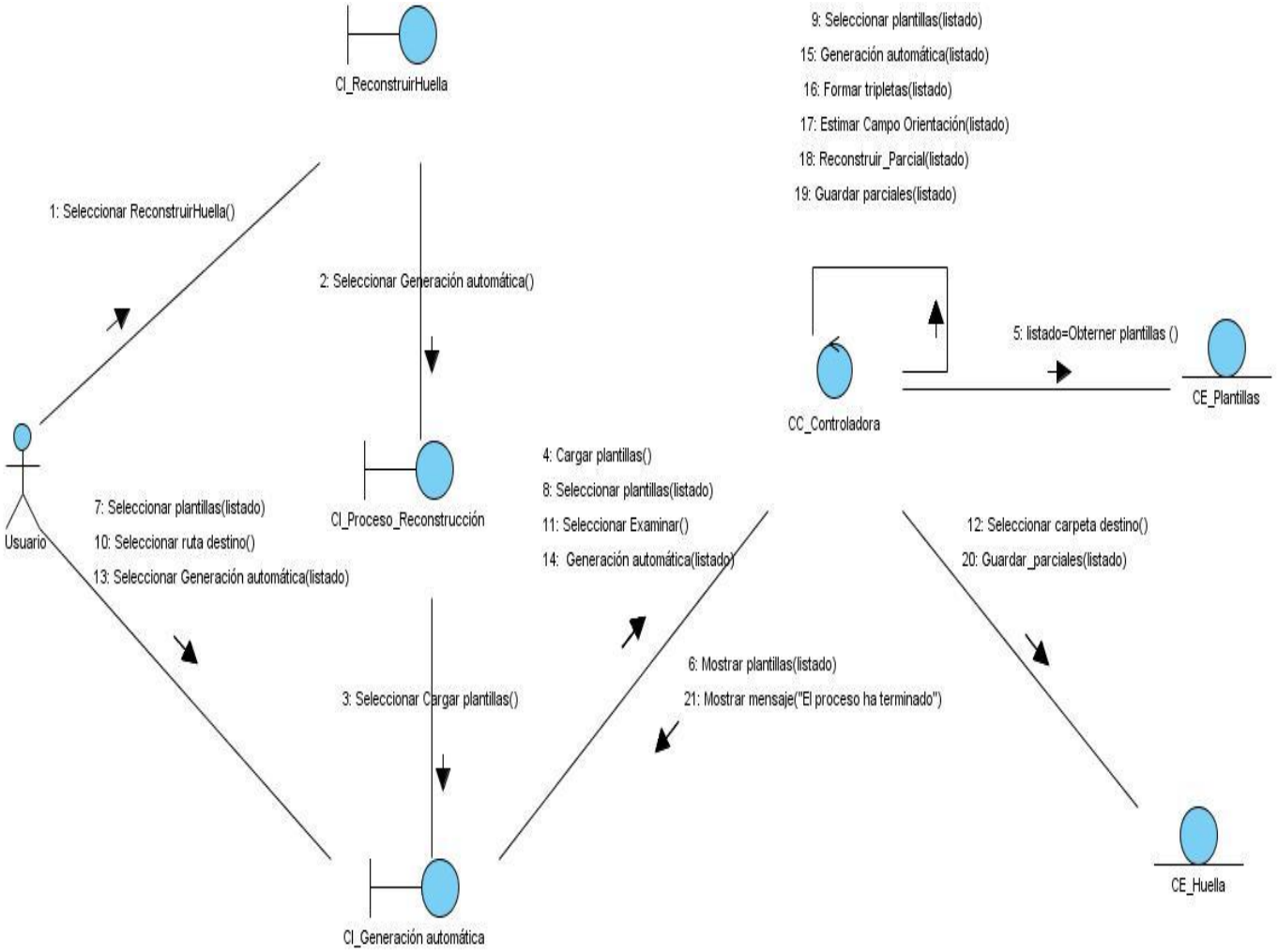


Diagrama de Secuencia del CUS: Reconstruir huella dactilar.

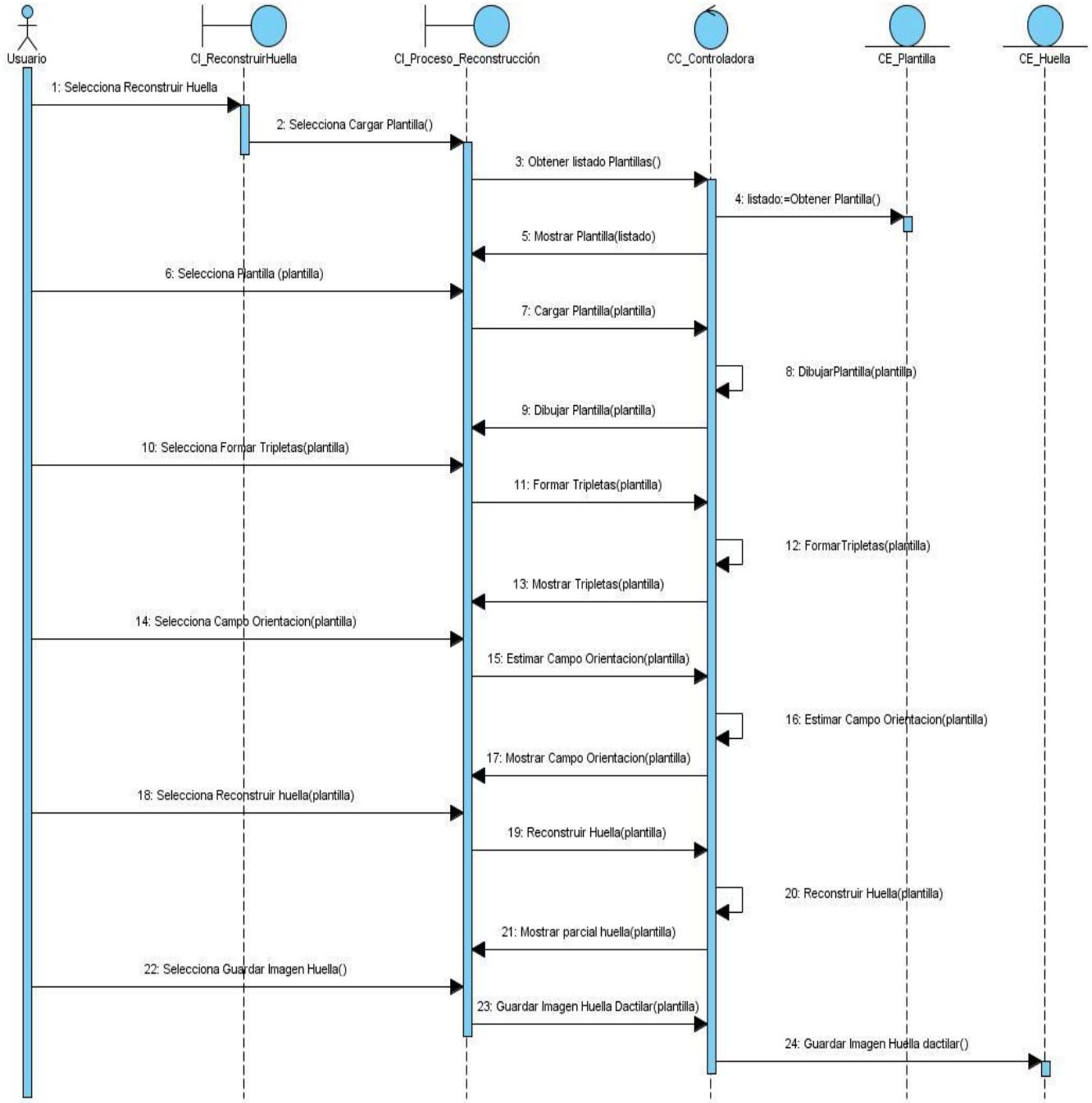
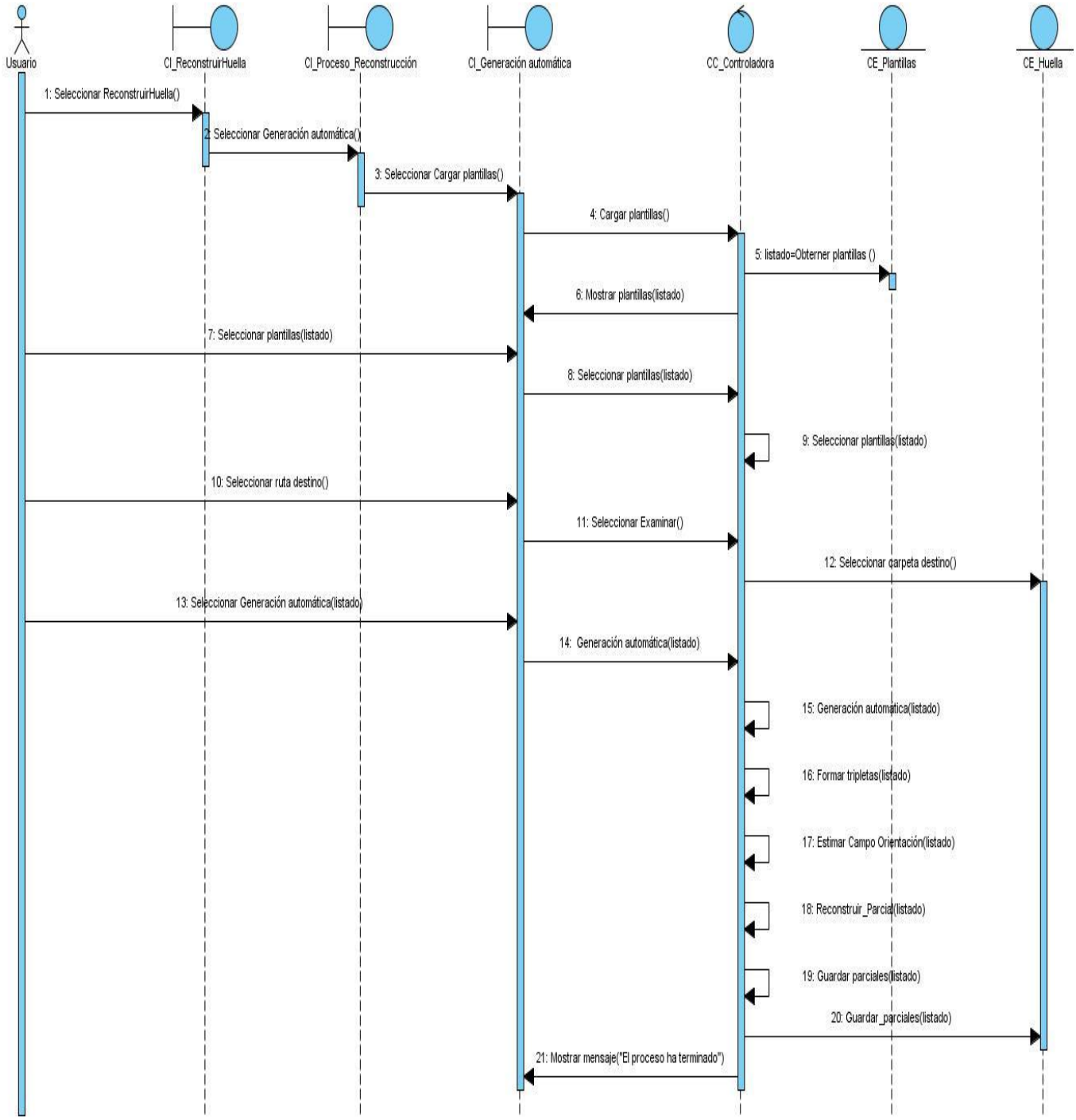
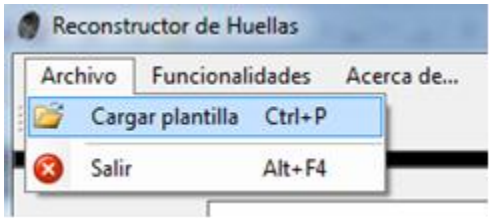


Diagrama de Secuencia del CUS: Generar parciales.



Anexo 3:

Menú desplegable



Barra de botones.