Universidad de las Ciencias Informáticas

Facultad 1

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

Título:

Sistema de identificación de personas basado en características faciales.

Autores:

Claudia Durán Rodríguez.

David Fonseca Martínez.

Tutores:

Ing. Rafael Alberto Quiles Velázquez.

MSc. Martha Ambruster Crespo.

La Habana, Cuba. Junio 2015.

"El software libre construye una sociedad mejor"

Richard Stallman

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo de diploma y conferimos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de ____ del año ____.

_____	_____
Claudia Durán Rodríguez	David Fonseca Martínez
Autor	Autor
_____	_____
MSc. Martha Ambruster Crespo	Ing. Rafael Alberto Quiles Velázquez
Tutor	Tutor

DATOS DE CONTACTO

Tutor: MSc. Martha Ambruster Crespo.

Graduada de Ingeniería Informática en el Instituto Superior Politécnico José Antonio Echeverría en el 2003. Actualmente se desempeña como profesora del Departamento de Desarrollo de Aplicaciones del Centro de Identificación y Seguridad Digital (CISED). En el período comprendido entre el 2011-2013 perteneció al departamento de Biometría, en el cual atendió un proyecto relacionado con la identificación facial.
Correo electrónico: mambruster@uci.cu

Tutor: Ing. Rafael Alberto Quiles Velázquez

Ingeniero en Ciencias Informáticas. Profesor asociado al Centro de Identificación y Seguridad Digital (CISED), pertenece al Departamento de Desarrollo de Componentes, graduado en el curso 2012-2013. Tiene dos años de experiencia en el tema Procesamiento de imágenes y señales digitales.
Correo electrónico: raquiles@uci.cu

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) constituye un eslabón fundamental en el proceso de migración llevado a cabo en el país, ya que además de promoverlo también está inmersa en él. En el Centro de Identificación y Seguridad Digital (CISED) se desarrolló en el año 2013 el sistema de identificación de personas mediante rasgos faciales que permite el reconocimiento de rostros a través de imágenes digitales. Este sistema no permite ser desplegado sobre tecnologías libres siendo esta una limitación para su utilización. La presente investigación se centra en el desarrollo de un sistema de identificación de personas basado en características faciales utilizando software libre.

Como resultado del trabajo de diploma se obtuvo un sistema de identificación de personas basado en características faciales que permite realizar todos los pasos necesarios en el proceso de identificación. Este sistema tiene como fin garantizar la confiabilidad de acceso a las instalaciones del centro y que el mismo pueda ser desplegado sobre tecnologías libres. Además forma parte de los esfuerzos del CISED para desarrollar soluciones propias de identificación, representando un paso importante para encontrar alternativas de buena calidad a los costosos sistemas de identificación existentes en el mercado.

Para el desarrollo de la presente investigación se utilizaron herramientas y tecnologías libres. Se realizó además una valoración crítica de los métodos y algoritmos existentes que permiten realizar el reconocimiento de rostros en imágenes digitales.

Palabras claves: rasgos faciales, reconocimiento de rostros, tecnologías libres.

ÍNDICE

Índice de imágenes.

Índice de tablas.

Introducción

El reconocimiento facial en los últimos años se ha convertido en un área de investigación activa que abarca diversas disciplinas, entre las que se destacan el procesado de imágenes, el reconocimiento de patrones, la visión por ordenador, las redes neuronales y el reconocimiento de objetos, donde el rostro es un objeto tridimensional sujeto a variaciones (1). El objetivo de un sistema de reconocimiento facial es: dada una imagen de un rostro desconocido o imagen de prueba, encontrar una imagen del mismo rostro en un conjunto de imágenes conocidas o imágenes de entrenamiento. La gran dificultad añadida es la de conseguir que este proceso se pueda realizar en tiempo real. Estos sistemas son capaces de identificar los rostros presentes en imágenes o videos automáticamente (2).

Lo que se conoce actualmente por biometría facial nació en los años 60; siglo XX, cuando Woodrow Wilson Bledsoe, Helen Chan Wolf y Charles Bisson crearon el primer sistema semiautomático para reconocimiento facial. Con la supervisión de un administrador externo, los primeros sistemas reconocían rasgos como ojos, orejas, nariz o boca, para así tomar distancias de referencia y compararlas con un patrón dado. La automatización del reconocimiento facial no llegó hasta una década después, cuando se comenzaron a utilizar características como el grosor de los labios o el color del cabello. A partir de los años 90; siglo XX surge la biometría facial tal y como es conocida hoy en día, aunque su implementación práctica llegó en el año 2001, cuando se comenzaron a archivar fotografías de los sistemas de vigilancia y se compararon con bases de datos digitales (3).

Hoy la tecnología de reconocimiento facial se utiliza principalmente en sistemas de seguridad para el reconocimiento de usuarios. También se utiliza en aplicaciones de interacción persona-ordenador, en gestión multimedia, y en software tales como *Google's Picasa, Apple iPhoto, Sony's Picture Motion Browser (*PMB*), Facebook y Asus Smart Logon* (4). Debido a que la obtención de la imagen de una persona es una tarea sencilla hoy en día, se ha comenzado a implantar en aeropuertos y lugares públicos el reconocimiento facial para la identificación de personas (5).

El desarrollo de sistemas con herramientas y tecnologías privativas, las cuáles a nivel de mercado poseen un alto precio de uso, limita la adquisición de los mismos por países de bajos recursos económicos. Estos tienen que obtenerlos a través de terceros y en múltiples casos han sido utilizados sin tener presentes los problemas de derechos de autor y licencias que esto puede traer.

Cuba es uno de los países que cuenta con varios centros de desarrollo dedicados al estudio de la tecnología biométrica, entre ellos se encuentra el Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV). En este centro se desarrollan algoritmos propios de reconocimiento facial. Los mismos están integrados en varios sistemas del país utilizados mayormente para el control de las fronteras, la emisión de documentos de identidad, entre otros. Existen también instituciones docentes vinculadas, como es el caso de la Universidad de las Ciencias Informáticas (UCI), y en ella se destaca el Centro de Identificación y Seguridad Digital (CISED), en el que se desarrollan e integran soluciones en las áreas de identificación y control de fronteras, seguridad digital, tarjetas inteligentes y biometría. Esta última basándose en los procesos para el reconocimiento de personas a través de los rasgos faciales y las huellas dactilares. CISED cuenta con un sistema de identificación de personas basado en rasgos faciales, pero el mismo no puede ser utilizado en tecnologías libres. Lo anteriormente expuesto evidencia una dificultad ya que la universidad tiene hoy la necesidad de ejecutar acciones que garanticen la migración ordenada y progresiva hacia aplicaciones y plataformas de software libre.

Para dar solución a la problemática anteriormente planteada se define como **problema de investigación** ¿Cómo desarrollar un sistema de identificación de personas basado en características faciales que esté acorde con la política de migración a software libre definida por el país?

Definiéndose como **objeto de estudio** los procesos para la identificación de personas basados en características faciales.

Por lo que se establece como **objetivo general** desarrollar un sistema de identificación de personas basado en características faciales utilizando software libre.

Con el propósito de lograr el objetivo propuesto se han definido los siguientes **objetivos específicos**:

- Analizar los elementos teóricos relacionados con el proceso de identificación de personas mediante sus características faciales.
- Analizar las herramientas y tecnologías de software libre para el desarrollo de sistemas informáticos.
- Diseñar e implementar el sistema de identificación de personas basado en características faciales.
- Validar el sistema desarrollado a partir de la ejecución de pruebas de efectividad y funcionalidad.

Para lograr los objetivos propuestos se precisan las siguientes **tareas investigativas:**

- Análisis del sistema para la identificación de personas mediante rasgos faciales existente en el centro CISED, sus funcionalidades, diseño, arquitectura y limitaciones. (Responsable: David Fonseca Martínez)
- Análisis de soluciones existentes en el mercado para la identificación de personas mediante rasgos faciales. (Responsable: Claudia Durán Rodríguez)
- Análisis de algoritmos para el reconocimiento, normalización, extracción de características y comparación de esas características en imágenes faciales. (Responsable: Claudia Durán Rodríguez)
- Análisis de los estándares involucrados en el proceso de identificación de personas mediante imágenes faciales. (Responsable: Claudia Durán Rodríguez)
- Descripción de las tecnologías, herramientas y metodología seleccionada para el desarrollo del sistema de identificación de personas basado en características faciales. (Responsable: Claudia Durán Rodríguez)
- Definición de la arquitectura para el desarrollo del sistema de identificación de personas basado en características faciales. (Responsable: David Fonseca Martínez, Claudia Durán Rodríguez)
- Desarrollo del sistema de identificación de personas basado en características faciales utilizando herramientas de software libre. (Responsable: David Fonseca Martínez, Claudia Durán Rodríguez)
- Realización de las pruebas al sistema de identificación de personas basado en características faciales. (Responsable: David Fonseca Martínez)

Para el desarrollo de la investigación se utilizaron diferentes **métodos científicos** como son:

Métodos teóricos:

- **Análisis Histórico-Lógico:** Se realiza un estudio de la evolución de los sistemas de identificación de persona basado en rasgos faciales desarrollados hasta el momento y sus principales aportes.

- **Analítico-Sintético:** Se utiliza para consultar la bibliografía referente al tema de identificación de personas basado en rasgos faciales, así como las normas y estándares internacionales para la identificación de las personas de forma segura y se hace uso de la información más adecuada al presente trabajo. Este método permite definir elementos significativos como base para elaborar la propuesta de solución al problema planteado.

Método empírico

- **Observación:** Brinda las herramientas para la recopilación atenta, racional, planificada y sistemática del fenómeno relacionado con el objeto de investigación. Además facilita el análisis de las tecnologías y productos existentes para desarrollar el sistema deseado.
- **Modelación:** Ayuda a representar los procesos a través de diagramas y así facilitar la comprensión del sistema.
- **Experimental:** Posibilita la realización de las pruebas de efectividad bajo condiciones controladas.

La justificación de la investigación del presente trabajo se basa en el desarrollo de un sistema de identificación de personas basado en características faciales que esté acorde con la política de migración a software libre definida por el país. Surge como necesidad de la universidad de ejecutar acciones que garanticen la migración ordenada y progresiva hacia aplicaciones y plataformas de código abierto. De esta forma repercute en los esfuerzos del CISED para crear sistemas propios de identificación, representando un paso importante para encontrar soluciones alternativas que garanticen la seguridad en el control de acceso.

El contenido se encuentra estructurado en tres capítulos los cuáles se agrupan de la manera siguiente:

Capítulo 1. Fundamentación teórica: En este capítulo se describen los principales conceptos y definiciones sobre los sistemas de reconocimiento facial. Se analizan además las soluciones existentes tanto en el ámbito internacional como en el nacional, y se describen las herramientas a utilizar para el diseño e implementación de la solución.

Capítulo 2. Análisis y diseño: En este capítulo se identifican las historias de usuarios y los requisitos no funcionales a tener en cuenta durante el desarrollo de la propuesta de solución. Se realiza además la descripción de la arquitectura y se confecciona el plan de iteraciones y plan de entregas.

Capítulo 3. Implementación y validación de la solución: En este capítulo se describen los principales artefactos relacionados con la implementación del sistema, como los diagramas de componentes y de despliegue. Además se realizan pruebas funcionales y de efectividad para validar en correcto funcionamiento del sistema.

CAPÍTULO I: “FUNDAMENTACIÓN TEÓRICA”

1.1 Introducción

Este capítulo aborda los elementos teóricos que respaldan el objeto de estudio y el objetivo de la investigación. En él se relacionan todos los conceptos que desde el punto de vista teórico permiten un mejor entendimiento de lo que se plantea en la situación problemática. También se efectúa un estudio sobre las soluciones existentes en el mundo y en nuestro país, con el fin de facilitar la comprensión de la importancia de la investigación, su alcance y aporte práctico. Además se lleva a cabo un análisis de las principales metodologías existentes, se estudian el lenguaje, las tecnologías y herramientas que se utilizan a lo largo de la presente investigación.

1.2 Conceptos básicos asociados al dominio del problema

Software libre: El software libre es aquel que se encuentra disponible gratuita o comercialmente. Se basa en la cooperación, la transparencia y garantiza una serie de libertades a los usuarios (6).

Estas libertades son (7):

- La libertad de usar el programa, con cualquier propósito.
- La libertad de estudiar el funcionamiento del programa y adaptarlo a necesidades propias. El acceso al código fuente es un prerrequisito para esto.
- La libertad de distribuir copias para ayudar a los demás.
- La libertad de mejorar el programa y de publicar las mejoras, de modo que toda la comunidad se beneficie. El acceso al código fuente es un prerrequisito para esto**.**

Biometría: Biometría es la ciencia y la tecnología dedicada a medir y analizar datos biológicos. En el terreno de la tecnología de la información, la biometría hace referencia a las tecnologías que miden y analizan las características del cuerpo humano, como el ADN, las huellas dactilares, la retina y el iris de los ojos, los patrones faciales o de la voz y las medidas de las manos a efectos de autenticación de identidades (8).

Reconocimiento biométrico: El concepto de reconocimiento es un término genérico que no implica por defecto una verificación o identificación de un individuo. Todos los sistemas biométricos realizan reconocimiento para distinguir de nuevo una persona que se ha ingresado previamente a un sistema (9).

Sistemas biométricos: Un sistema biométrico es un método automático de identificación y verificación de la identidad de un individuo utilizando características físicas y de comportamiento precisas (10). Estos sistemas utilizan dispositivos biométricos para identificar atributos físicos tales como rasgos faciales, patrones oculares, huellas digitales, voz y escritura a mano.

1.3 Sistemas de reconocimiento facial

Un sistema de reconocimiento facial es una tecnología que permite identificar automáticamente a una persona a través de una imagen digital de su rostro. Estos sistemas permiten la comparación de determinadas características de un rostro con las informaciones de una persona en una base de datos facial, haciendo uso de un lector electrónico que identifica las características del rostro, pues todos los seres humanos tienen características morfológicas únicas que los diferencian (11). Los sistemas de reconocimiento facial tienen características que los hacen diferentes de los otros sistemas de identificación biométrica. En primer lugar, para la obtención de la imagen no es necesaria la participación de la persona a identificar. Otro factor que es importante destacar es la poca variabilidad que existe entre miembros de una misma familia. Sin embargo, la variabilidad entre imágenes de una misma persona puede ser muy grande: corte de pelo, gafas, barba, bigote, maquillaje, heridas o una sonrisa. El proceso de reconocimiento facial está compuesto por varias etapas. Estas etapas consisten en la detección del rostro una vez haya sido capturada la imagen facial de la persona a reconocer, la extracción de características y la confrontación con la base de datos (12).

1.3.1 Etapas del proceso de identificación

1.3.1.1 Detección del rostro

La detección del rostro supone el primer paso en cualquier sistema de reconocimiento facial. En este proceso existen dos objetivos principales: localización de la región facial (si existe) y segmentación de la misma. Para que cualquier sistema de reconocimiento facial funcione perfectamente se debe hacer una detección precisa del rostro. No solo se debe detectar el rostro para la identificación o verificación de

personas sino que también hay que tener en cuenta otros aspectos los cuáles podrían dificultar el proceso de detección del rostro tales como: (5)

- Estado de ánimo de la persona.
- Pose y orientación del rostro.
- Tamaño del rostro.
- Presencia de lentes, barba, gorros, etc.
- Expresión de la cara.
- Problemas de iluminación.
- Condiciones de la imagen.
- Cantidad desconocida de rostros en la imagen.

La detección de rostros por parte de una computadora es el proceso por el cual la computadora ubica los rostros presentes en una imagen o en un video. Existen diversos métodos para poder realizar detección de rostros por medio de una computadora. Una posible clasificación de los algoritmos para la detección de rostros es: (9)

- Técnicas basadas en rasgos faciales: buscan encontrar aquellas características presentes en cualquier rostro: ojos, cejas, labios, boca, mentón, líneas de contorno, etc.

Dentro de las técnicas basadas en rasgos faciales se pueden hacer diferentes tipos de análisis, como puede ser el análisis de bajo nivel, este no es más que técnicas que trabajan a nivel de píxel, ejemplo de este son los bordes, color y el video.

- Técnicas basadas en la imagen: aplican herramientas generales de reconocimiento de patrones para sintetizar un modelo a partir de un conjunto de imágenes de entrenamiento. Trabajan con la imagen completa o una región, sin buscar rasgos faciales de forma localizada.

Dentro de las técnicas basadas en la imagen se encuentran los siguientes métodos: (9)

- Redes neuronales: Es uno de los métodos más utilizado en la detección de rostros en una imagen, dado el alto porcentaje de aciertos que produce, siendo en algunos casos superiores al 95%. La red neuronal se entrena usando un conjunto de imágenes que representan rostros de todo tipo (de varias razas, tonos de piel, con y sin gafas, pendientes, posiciones de los labios y ojos, ligeras rotaciones, e incluso caras humanoides) y otro conjunto de imágenes que no representan rostros, de forma que

la red neuronal pueda establecer el criterio adecuado acerca de lo que es un rostro y lo que no lo es. La respuesta de la red neuronal ante una imagen de entrada es la de decidir si dicha imagen corresponde o no a un rostro, es decir, una respuesta binaria.

- Métodos estadísticos: Este tipo de algoritmo no asume ningún tipo de información previa de la tipología de un rostro; sino, que a partir de un conjunto de muestras (imágenes con y sin rostros) de entrenamiento extraen la información relevante que diferencia un objeto rostro de un objeto no rostro. En la **tabla 1** se comparan las técnicas para la detección de rostros anteriormente descritas (9).

Técnicas basadas en rasgos faciales	Análisis de bajo niel en el color	Ventajas	Efectividad %	Desventajas
		Minimizan los efectos que tienen las variaciones de iluminación o pose.	90%	Dificultad al localizar ciertos rasgos faciales por oclusión.
Técnicas basadas en la imagen	Redes neuronales	Alto porcentaje de aciertos que produce, superior al 95%.	95%	Se caracteriza por ser lento.
	Métodos estadísticos: AdaBoost desarrollado por Viola y Jones	Procesa imágenes extremadamente rápido. Eficaz y rápido en término de detección de rostro.	99%	Genera falsos positivos, al reconocer objetos que no son rostros dentro de una imagen.

Tabla 1: Comparación de los métodos de detección de rostros. (9)

Uno de los algoritmos para la detección de rostro basado en los métodos estadísticos de las técnicas basadas en la imagen es el algoritmo creado por Paul Viola y Michel Jones. Este algoritmo consta de dos etapas: en la primera se entrena al detector utilizando un algoritmo de *boosting* [1]para la obtención de una cascada de clasificadores y en la segunda etapa se usa esta cascada para la detección de los rostros. Para Paul Viola y Michel Jones hay tres contribuciones principales que caracterizan a este algoritmo para la detección de rostros en imágenes digitales. Se puede apreciar que estas contribuciones están enfocadas hacia la eficiencia computacional, en función de aumentar la velocidad y la eficiencia del algoritmo (14).

[1] Los algoritmos de *boosting* son meta algoritmos de máquinas de aprendizaje para el mejoramiento de aprendizajes supervisados. (9)

- La primera es la utilización de la imagen integral; esto permite evaluar de forma muy rápida las características utilizadas. Este algoritmo no trabaja con la intensidad de la imagen en sí, sino que trabaja con una representación de la misma. La gran ventaja es que se pueden calcular las características utilizadas en cualquier lugar de la imagen y en cualquier escala en el mismo lapso de tiempo y en forma sumamente eficiente desde el punto de vista del gasto computacional.
- La segunda contribución es el método de construcción del clasificador, que se lleva a cabo seleccionando un número reducido de significativas características, utilizando para ello una modificación del algoritmo AdaBoost. Este algoritmo se utiliza para seleccionar las características y entrenar los clasificadores.
- La tercera contribución es el método para combinar una sucesión de clasificadores cada vez más complejos en una estructura de cascada. En esta contribución los clasificadores más sencillos, de bajo costo computacional descartan una gran cantidad de sub-ventanas, dejando la mayor concentración en los clasificadores más complejos en las zonas donde es más probable que haya un rostro.

Viola y Jones utiliza un algoritmo clasificador que basa su funcionamiento en la detección de objetos nombrado AdaBoost. Este algoritmo consiste en un árbol de decisión compuesto por una serie de clasificadores débiles, los cuáles están dispuestos en forma de cascada formando una sucesión, arrojando como resultado un clasificador fuerte con alto grado de efectividad (15). El algoritmo AdaBoost utiliza una ventana que recorre la imagen comprobando en cada posición la existencia del objeto con el cual fue entrenado. El tamaño de la ventana está dado por el tamaño mínimo de los rostros que se desean detectar y una vez finalizado el recorrido de la imagen es aumentado sucesivamente hasta cubrir el total de la imagen. Finalmente, el clasificador devuelve una secuencia de rectángulos que indican las posiciones de los rostros detectados en la imagen.

1.3.1.2 Pre-procesado de la imagen facial

El pre-procesado se encarga de aplicar todas las operaciones que se crean necesarias sobre la información de entrada, con el objetivo siempre de mejorar los resultados del sistema. Por lo tanto permite solucionar posibles problemas con la información de entrada lo que mejora la capacidad de adaptación del programa a variaciones en los datos con los que se trabaja. Algunas de las operaciones que se realizan para lograr el pre-procesado de la imagen son las siguientes: (16)

- Normalización de las imágenes: Para llevar a cabo este proceso se necesita una serie de imágenes de tamaño MxN píxeles. Con esto se pretende ajustar las dimensiones de todas las imágenes de manera que adopten el mismo tamaño, obteniendo así uniformidad en sus dimensiones (16).
- Corrección de la iluminación: Este es uno de los grandes problemas a los que se enfrenta este tipo de sistemas. Un cambio en el nivel de iluminación de las imágenes, ya sea local o global, afectará enormemente al proceso de clasificación de las imágenes. Se pueden corregir estos problemas mediante la ecualización del histograma, que no es más que una transformación donde se pretende obtener para una imagen un histograma con el mismo número de pixeles para cada nivel de gris (16).
- Corrección de la inclinación del rostro: Este es un caso muy común en sistemas de reconocimiento facial sobre todo si se realizan las capturas de las imágenes desde distintos puntos de vista. Se pueden encontrar situaciones en que la cabeza del individuo en cuestión se encuentra inclinada con respecto al centro de la imagen, lo cual puede perjudicar al sistema (16).

1.3.1.3 Extracción de las características

La extracción de características consiste en localizar de forma precisa las regiones faciales y esta se puede tratar de dos formas: (5)

- Métodos basados en plantillas: Los métodos basados en plantillas utilizan imágenes predefinidas de las regiones faciales para realizar su localización.
- Métodos estructurales: Los métodos estructurales usan características del contorno y de la textura de la región facial para construir un modelo estadístico de la misma.

Enfocados en las características que cumplen los sistemas de identificación de personas, existen dos enfoques predominantes, el geométrico; basado en rasgos y el fotométrico; basado en lo visual (5). Los métodos basados en rasgos faciales buscan encontrar aquellas características presentes en cualquier rostro: ojos, cejas, labios, boca, mentón, líneas de contorno, etc. Se pueden definir tres ramas dentro del conjunto de métodos basados en rasgos faciales: (17)

- Análisis de bajo nivel: son técnicas que trabajan directamente con los píxeles.

- Análisis de rasgos faciales: son técnicas que hacen énfasis a las relaciones geométricas que cumplen los diferentes rasgos presentes en un rostro.
- Análisis mediante modelos de contornos activos: son técnicas que buscan adaptar un modelo genérico de un rasgo a la imagen. Iteran deformando el modelo hasta adaptarlo al rasgo buscado. Se basan fuertemente en la información local de la imagen.

Los métodos basados en la imagen aplican herramientas generales de reconocimiento de patrones para sintetizar un modelo a partir de un conjunto de imágenes de entrenamiento. Trabajan con la imagen completa o una región de esta sin buscar rasgos faciales de forma localizada (17).

- Sub-espacios lineales: Esta técnica se fundamenta en representar las imágenes de los rostros en espacios lineales buscando a qué espacio lineal pertenece mediante un análisis estadístico, entre los cuáles se destacan: Análisis de componentes principales (PCA), Análisis de discriminante lineal (LDA), Análisis de componentes independientes (ICA).
- Redes neuronales: Es una técnica de mayor uso para el reconocimiento de patrones ya que se puede verificar si una imagen contiene un rostro. Esto se logra entrenando las redes neuronales con imágenes que contienen rostros y otras imágenes que no los contienen (18).
- Métodos estadísticos: Este tipo de algoritmo no asume ningún tipo de información previa de la tipología de un rostro; sino, que a partir de un conjunto de muestras de entrenamiento extraen la información relevante que diferencia un objeto rostro de un objeto no rostro.

A continuación se describen algunos métodos existentes para realizar la extracción de las características:

PCA (Análisis de componentes principales): Es una técnica tradicional en el reconocimiento de rostros y probablemente la más utilizada. Esta técnica consiste en extraer de un conjunto de imágenes de entrenamiento un sub-espacio que maximice la varianza del espacio original, a estos vectores que se obtienen de estos cálculos se les denomina Eigenfaces. De esta manera se logra reducir de forma considerable la dimensión del problema, tomando las características únicas y propias de las imágenes originales. Luego se podrán fijar métricas, y con estas hallar la distancia del vector de características de entrada con la distancia de los vectores almacenados en la base de datos (17).

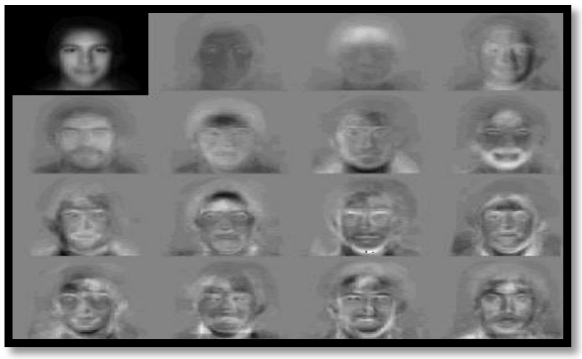

Figura 1: Eigenfaces estándar. (19)

ICA (Análisis de componentes independientes): Este método intenta representar el espacio de rostros en un sub-espacio que minimice la dependencia de segundo y de mayor orden entre sus componentes. Se asume que las señales de entrada son combinaciones de fuentes no observables estadísticamente independientes. Si la combinación es lineal, se puede definir una matriz de combinación cuyos coeficientes son los que definen la combinación lineal. De esta forma ICA estima la matriz inversa de la matriz de combinación. Vale acotar que para aplicar ICA se utiliza previamente PCA para reducir la dimensión del espacio original de rostros con el objetivo de disminuir el costo computacional (20).

LDA (Análisis de discriminante lineal): Esta técnica consiste en encontrar combinaciones lineales para poder reducir la dimensión del problema, de tal manera que se mantenga la habilidad de separar dos o más clases de objetos. La idea del algoritmo es encontrar la base de vectores en un sub-espacio que mejor discrimine entre las diferentes clases, en el caso del reconocimiento las identidades. Se utilizan todas las muestras de todas las clases y se calcula la matriz de dispersión entre clases distintas y la matriz de dispersión en la misma clase. Se busca maximizar la relación entre el determinante de la matriz entre clases distintas y el determinante de la matriz entre la misma clase. Los elementos de la base que maximiza la relación anterior, se denominan Fisherfaces (21).

Figura 2: Ejemplo de seis clases utilizando LDA. (19)

EP (*Evolutionary pursuit*): Esta técnica utiliza algoritmos genéticos para proveer la proyección de los rostros en un espacio no ortonormal resaltando sus diferencias. La dimensión del problema se ve reducida gracias al PCA. El sub-espacio resultante es rotado para maximizar las diferencias entre las proyecciones de cada imagen. Esta técnica puede tener más ventajas que la PCA siempre y cuando el entrenamiento de las imágenes se haga de forma balanceada (22).

DCT (Transformada discreta del coseno): Esta técnica depende solo del orden de la transformada seleccionado, y no de las propiedades estadísticas de los datos de entrada. Posee la capacidad de cuantificar los coeficientes utilizando valores de cuantificación que se eligen de forma visual. Es una transformada real, debido a que los vectores base se componen exclusivamente de funciones coseno muestreadas. Además la DCT minimiza algunos de los problemas que surgen con la aplicación de la DFT a series de datos (23).

1.3.1.4 Reconocimiento

El reconocimiento en un sistema de identificación de rostros parte de la base que rostros de distintos individuos tienen características similares, por lo que se pueden agrupar imágenes de individuos distintos en una misma clase. Un sistema de comparación debe ser capaz de asignar a cada nueva imagen de entrada una de estas clases. Para realizar la comparación se calcula la distancia entre imágenes, dentro de un espacio multidimensional. Por lo tanto el resultado de la decisión de este sistema dependerá en gran medida del método utilizado para calcular las distancias entre las diferentes muestras, algunas de las cuáles se citan a continuación: (5)

Distancia euclídea: Esta es la distancia más común que se suele definir como la distancia entre dos puntos cualesquiera y viene definida por el teorema de Pitágoras. Utilizando este teorema para el cálculo de distancias, el espacio euclidiano (o cualquier espacio con producto interior) se convierte en un espacio métrico. La fórmula en este caso es:

$$d = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

Ecuación 1: Distancia euclídea. (16)

Distancia Manhattan: Esta distancia suma todas las distancias de cada dimensión que forma el espacio. Se puede calcular de la siguiente forma:

$$d = \sum_{i=1}^{n} |p_i - q_i|$$

Ecuación 2: Distancia Manhattan. (16)

1.4 Algoritmos para obtener función resumen

Los sistemas de identificación por rostros al igual que otros sistemas realizan una gestión de usuarios y por lo tanto las contraseñas de estos usuarios son guardadas en base de datos. Estas contraseñas por cuestiones de seguridad no se guardan en texto plano, sino que se aplica una función resumen al texto de la contraseña. En el siguiente epígrafe se caracterizan algunos de los algoritmos más populares.

1.4.1 MD5

MD5 es uno de los algoritmos de reducción criptográficos que permite obtener un resumen de cualquier secuencia de datos y en particular de un archivo o documento, mediante ciertas funciones matemáticas. Es un algoritmo resumen de mensajes y autentificación de documentos por lo que solo se obtiene como resultado de la encriptación una serie de n números de los cuáles no se pueden descifrar el mensaje. La

única forma para poder falsear el *hash* [2] es por colisión o fuerza bruta, por lo que cuando se comiencen a encontrar muchas colisiones el algoritmo no será tan eficaz. A continuación se detallan las principales características del algoritmo: (24)

- Si se aplica a una misma entrada siempre se obtiene la misma salida.
- Genera pequeñas diferencias en la salida, lo que hace que cualquier modificación en el documento original se vea reflejada en el resumen.
- La probabilidad de que dos documentos distintos generen la misma salida es muy baja.
- A pesar de haber sido considerado criptográficamente seguro en un principio, ciertas investigaciones han revelado vulnerabilidades que hacen cuestionable el uso futuro del MD5 (25).

1.4.2 SHA

El **SHA** (Algoritmo de *Hash* Seguro) es una familia de funciones *hash* de cifrado. La primera versión del algoritmo fue creada en 1993 con el nombre de SHA, aunque en la actualidad se le conoce como **SHA-0** para evitar confusiones con las versiones posteriores. La segunda versión del sistema, publicada con el nombre de **SHA-1** fue publicada dos años más tarde (26). Este algoritmo trata bloques de 512 bits de mensaje con un total de 80 vueltas, el vector inicial tiene una palabra de 32 bits permitiendo que el resumen sea de 160 bits.

Luego de realizar un análisis de los algoritmos expuestos anteriormente se decide utilizar SHA-1 para obtener la función resumen de las contraseñas de los usuarios registrados en el sistema a desarrollar. Se escogió el algoritmo SHA-1 ya que el mismo aún no presenta ataques registrados y al convertir el texto en una cadena hexadecimal provoca que sea muy difícil de corromper y a su vez no se puede desencriptar. La salida que genera la encriptación mediante SHA-1 tiene un tamaño de 160 bits superior a la salida con longitud de 128 bits generada por el algoritmo MD5, esto aporta mayor seguridad en el algoritmo.

1.5 Propuesta de los algoritmos a utilizar en el sistema de reconocimiento facial

1.5.1 Algoritmo a utilizar para la detección de rostros

Como se ha mencionado, la primera etapa en el proceso de reconocimiento facial es la detección del rostro una vez que se haya capturado de forma correcta la imagen facial de la persona. Para el presente trabajo

[2] Es el resultado de aplicar una función resumen.

se utiliza en la etapa de detección del rostro las técnicas basadas en la imagen, ya mencionadas anteriormente como una de las metodologías a utilizar en el proceso de detección, las cuáles establecen dos métodos principales; los basados en redes neuronales y los estadísticos siendo este último el que se elige para la implementación de la fase de detección de rostro. Como método estadístico se decide utilizar el algoritmo de Viola y Jones por la elevada efectividad del mismo. Además el algoritmo no utiliza directamente la imagen facial sino que utiliza una representación de la imagen llamada imagen integral y en lugar de ampliar la propia imagen, el algoritmo escala las características de la misma.

1.5.2 Extracción de características faciales

La extracción de características (28) es una parte fundamental de un sistema de reconocimiento facial. La aplicación de los algoritmos para la extracción de características en imágenes faciales ha aumentado considerablemente en los últimos años. Con la necesidad de describir el contenido de la información de una imagen se hace uso de los descriptores de imagen. Un descriptor de una imagen es una forma de representar a una imagen por sus características, con fines de almacenamiento y recuperación, en donde el descriptor es una formulación matemática (28). La extracción de características que se realiza en el sistema de identificación se basa en el descriptor de distribución de color (DDC). El proceso de extracción de este descriptor de color consta de cuatro etapas:

- División de la imagen: La imagen de entrada se divide mediante una rejilla en diferentes bloques o regiones.
- Selección del color más representativo: Para cada uno de los bloques de la cuadrícula se selecciona un único color como representante de cada bloque.
- Transformada DCT: Una vez se obtiene el icono de la imagen y tras efectuar una conversión del espacio de color de la imagen original al espacio de color YCbCr, se realiza el cálculo de la DCT de cada una de las tres componentes de color, obteniendo así los llamados coeficientes de la DCT en una matriz.
- Exploración en zigzag: En esta última etapa se realiza una exploración en zigzag de los coeficientes de la matriz, con el objetivo de ponderar en mayor medida aquellos relacionados con las bajas frecuencias.

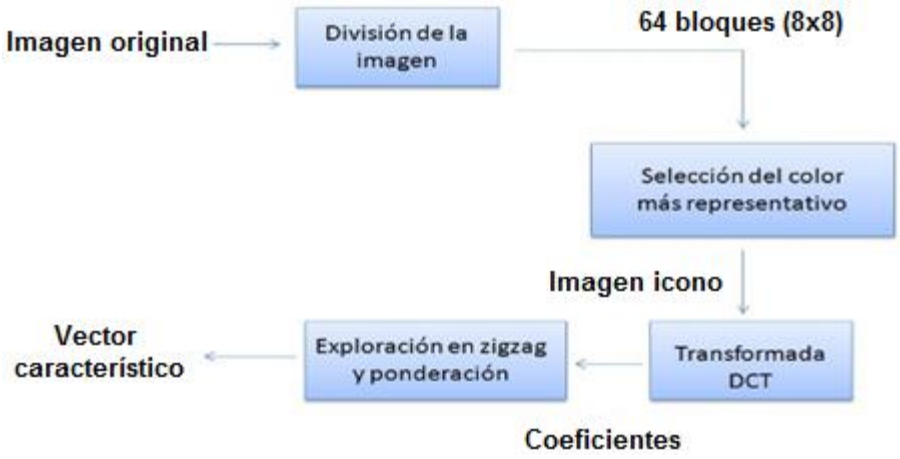

Figura 3: Diagrama de DCT. (28)

1.5.3 Confrontación con la base de datos

Luego de terminar todo el proceso descrito anteriormente se encuentra la etapa de confrontación con la base de datos que compara los descriptores de rostros para el reconocimiento. Este proceso ayuda a evaluar si dos vectores característicos son semejantes entre ellos. La confrontación con la base de datos se puede realizar una vez que sean comparados los vectores característicos a partir de la fórmula de distancia entre dos vectores. Su procedimiento es el siguiente: Dada una imagen de entrada la aplicación intenta encontrar otra imagen con un vector similar en la base de datos.

$$D = \sqrt{\sum_i w_{yi}(DY_i - DY'_i)^2} + \sqrt{\sum_i w_{bi}(DCb_i - DCb'_i)^2} + \sqrt{\sum_i w_{ri}(DCr_i - DCr'_i)^2}$$

Ecuación 3: Distancia entre vectores. (28)

El sub-índice (i) representa el orden de los coeficientes del vector característico, conjuntamente se utiliza el peso de los coeficientes (w) con el fin de ajustar el rendimiento del proceso del emparejamiento, en este caso (w) toma un valor 1. Analizando la fórmula, se puede inferir que:

- Dos imágenes son idénticas si la distancia es 0.

- Dos imágenes son similares si la distancia es cercana a 0.

El proceso de emparejamiento permite identificar imágenes similares. El objetivo es identificar personas basado en sus rasgos faciales, de modo que una vez obtenido el vector característico para cada persona, compara el vector obtenido con los almacenados en la base de datos basado en la técnica de emparejamiento. Luego se identifica la persona y se muestran los datos de la misma.

1.6 Sistemas similares

La revolución científica y tecnológica en que se encuentra el mundo, está implícita en todos los campos de la sociedad a nivel internacional, provocando que las soluciones se diversifiquen cada vez más aparejadas al desarrollo alcanzado. Existen diferentes soluciones en el ámbito nacional e internacional que permiten realizar el proceso de identificación de personas basados en sus características faciales.

- ***Picasa***: Es una herramienta de búsqueda de reconocimiento de rostros desarrollada por la empresa *Google Inc*.[3], capaz de identificar y buscar automáticamente el rostro de cualquier persona que se le indique y muestra todas las imágenes donde crea que se encuentre (29).
- ***Biomesys Face***: Es un sistema de identificación biométrica por rostro desarrollado en la empresa *Datys*[4], diseñado para reducir los tiempos y aumentar la efectividad en el reconocimiento de las personas a partir de las características faciales (30).
- **Sistema de identificación de personas basado en rasgos faciales:** Es un sistema de identificación de las personas mediante los rasgos faciales desarrollado en la UCI por el centro CISED, que permite realizar el reconocimiento del rostro de una persona a través de una imagen digital (9).

[3] Google Inc. es una empresa multinacional estadounidense especializada en productos y servicios relacionados con Internet, software, dispositivos electrónicos y otras tecnologías.
[4] DATYS es una empresa de alta tecnología, especializada en el desarrollo de aplicaciones informáticas, ofrece soluciones propias a problemas tecnológicos complejos. Fundada en el año 2005.

- **Sistema de próxima generación de identificación:** Es un sistema de reconocimiento para el control de criminales desarrollado por especialistas del FBI[5], capaz de identificar cualquier rostro aunque posea cicatrices, tatuajes, entre otros (31).
- ***DeepFace*:** Es un sistema de reconocimiento facial utilizado por *Facebook* [6] para etiquetar las personas presentes en cualquier imagen que haya sido subida a esta red social (32).

A continuación se muestra una tabla comparativa donde se detallan las principales características de los sistemas mencionados anteriormente:

Producto	Multiplataforma	Privativo	Lenguaje	Nacional
Picasa	Si	Si	C++	No
Biomesys Face	Si	Si	C++	Si
Sistema de identificación de personas basado en rasgos faciales	No	Si	C# &C++	Si
Sistema de próxima generación de identificación	No	Si	C++	No
DeepFace	No	Si	Python	No

Tabla 2: Tabla comparativa entre sistemas similares.

Se estudiaron las distintas soluciones antes mencionadas llegando a la conclusión, de que Cuba al ser un país bloqueado, necesita contar con un software de producción nacional que permita la sustitución de importaciones, aporte a la soberanía tecnológica del país y que el mismo no se encuentre desarrollado sobre tecnologías y herramientas privativas, puesto que estas tienen un alto costo en el mercado dificultando su adquisición. Se propone entonces desarrollar un sistema de identificación de personas basado en características faciales que esté acorde con la política de migración a software libre definida por el país y permita llevar el control de las personas a través de sus datos identificativos y características faciales.

[5] El FBI es una agencia federal de investigación e inteligencia con jurisdicción sobre una gran variedad de delitos federales, incluyendo asuntos de seguridad nacional como terrorismo y espionaje, secuestro o extravío de menores, crimen organizado, corrupción pública, y delitos cibernéticos e informáticos.
[6] Facebook es una red social creado por Mark Zuckerberg y fundada junto a Eduardo Saverin, Chris Hughes y Dustin Moskovitz en el 2004.

1.7 Lenguajes, metodologías y herramientas de desarrollo

La creciente informatización de los procesos productivos y sociales, ha traído consigo que las organizaciones y empresas requieran cada vez más de software confiable y de alta calidad. Por esta razón se debe hacer un profundo análisis para poder seleccionar correctamente todas las herramientas que se van a utilizar en el desarrollo de la solución, siempre pensando en las necesidades del cliente y tratando de utilizar las tecnologías más novedosas.

1.7.1 Metodología para el desarrollo del software

Una metodología es un proceso que engloba procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software. En ella se va indicando paso a paso lo que se debe hacer para lograr un producto informático. Además en esta se especifica las personas que van a participar en el proceso así como el papel que van jugar en él (33).

Existen dos tipos de metodologías: las robustas y las ágiles, entre los tipos de metodologías robustas se encuentran: *Rational Unified Process* (RUP), *Microsoft Solutions Framework* (MSF) y Métrica 3.0; dentro de las metodologías ágiles se encuentran: *Extreme Programming* (XP), *Scrum* y *Feature Driven Development* (33).

A continuación se detallan las características de dos metodologías ágiles, para de ellas seleccionar la más adecuada que guíe el desarrollo de la aplicación.

1.7.1.1 Programación extrema

XP guía el desarrollo de software haciendo énfasis en las relaciones interpersonales, fomentando el trabajo en equipo y la estrecha comunicación con el cliente. El producto de software se construye teniendo en cuenta que la solución más simple es la mejor. Está orientada a la adaptación paulatina de los requisitos y de sus cambios en cualquier punto de la vida del proyecto. Define historias de usuario para describir las funciones del sistema, las cuáles son escritas por el cliente. El ciclo de desarrollo en XP es iterativo e incremental y la propia metodología está regida por varias fases, las cuáles se enuncian a continuación (9).

- Fase I: Exploración: En esta fase el cliente plantea a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán para el desarrollo del proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.
- Fase II: Planificación de la Entrega: En esta fase el cliente establece la prioridad de cada historia de usuario y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.
- Fase III: Iteraciones: Esta fase incluye varias iteraciones antes de realizada la entrega del sistema. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede establecer una arquitectura del sistema que se utilice durante el resto del desarrollo del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración; para maximizar el valor de sistema. Al final de la última iteración el sistema estará listo para entrar en la fase de producción.
- Fase IV: Producción: Esta fase requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.
- Fase V: Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.
- Fase VI: Muerte del Proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo (34).

Características principales de XP:

- Desarrollo iterativo e incremental.
- Pruebas unitarias continuas.
- Frecuente interacción del equipo de programación con el cliente o usuario.
- Corrección de todos los errores antes de añadir una nueva funcionalidad.

Ventajas que aporta XP:

- Apropiado para entornos volátiles.
- Planificación más transparente para los clientes, conocen las fechas de entrega de funcionalidades.
- Permite definir en cada iteración cuáles son los objetivos de la siguiente.
- Permite tener realimentación concreta y frecuente del cliente, del equipo y de los usuarios finales.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

1.7.1.2 Scrum

Scrum es una metodología ágil enfocada a la gestión de proyectos. Sus principales características se pueden resumir en: el desarrollo de sprint o iteraciones y reuniones a lo largo del desarrollo. Las iteraciones en Scrum tienen una duración máxima de 30 días y el resultado de cada una de ellas define un incremento del producto a desarrollar. La evolución del proyecto por la metodología se define a través de reuniones diarias donde el trabajo del día anterior es revisado por el equipo, previendo además la labor a realizar el día siguiente. Dentro de las prácticas definidas por la metodología Scrum se encuentran (35):

- Planificación de la iteración o sprint.
- Reunión diaria.
- Revisión de la iteración o sprint.

Para la selección de la metodología se debe tener en cuenta que la comunicación entre el equipo de desarrollo y los clientes en el CISED será maximizada y efectuada de forma directa e interpersonal con el propósito de evitar los problemas y errores causados por una mala comunicación. Se realizará un trabajo

que favorezca la comunicación garantizando la calidad. Se estará constantemente midiendo el sistema para conocer cuánto se acerca a las funcionalidades necesarias mediante pruebas por parte de los usuarios. Los requisitos asociados a la aplicación pueden sufrir cambios debido a que se pueden arrojar nuevas funcionalidades a incluir durante el proceso de desarrollo del sistema de identificación. Vale destacar que el equipo de desarrollo cuenta con poco personal lo cual dificulta la adopción de nuevos roles en cada etapa así como la generación de múltiples artefactos. Por lo antes expuesto se selecciona la metodología XP como rectora del proceso para el desarrollo del sistema de identificación por lo apropiado de sus prácticas y valores para el desarrollo de software.

1.7.2 Lenguaje de modelado

Para el desarrollo del sistema de identificación de personas basado en características faciales se tiene como propuesta utilizar el lenguaje unificado de modelado (*Unified Modeling Language*, UML) en su versión 8.0, que se define como un lenguaje que permite especificar, visualizar y construir los artefactos de los sistemas de software. Este lenguaje de modelado se usa para entender, diseñar, configurar, mantener, y controlar la información. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. UML incluye conceptos semánticos, notación, y principios generales. Se puede aplicar en el desarrollo de software permitiendo gran variedad de formas para dar soporte a una metodología de desarrollo, pero no especifica en sí mismo qué metodología o proceso usar (36).

Un modelo UML está compuesto por tres clases de bloques de construcción:

- Elementos: los elementos son abstracciones de cosas reales o ficticias (objetos, acciones).
- Relaciones: relacionan los elementos entre sí.
- Diagramas: son colecciones de elementos con sus relaciones (37).

1.7.3 Herramienta de modelado

Las aplicaciones informáticas que facilitan el trabajo dentro del ciclo de desarrollo del software son conocidas como herramientas CASE (Ingeniería de Software Asistida por Computadora) y se emplean para aumentar la productividad del desarrollo del software, disminuyendo los tiempos de construcción y el costo de los

mismos. La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad en el desarrollo de sistemas de información. Para el desarrollo del ciclo de software se utilizará como herramientas CASE Visual Paradigm en su versión 8.0. Visual Paradigm es una herramienta CASE que soporta el ciclo de vida completo en el desarrollo de software: análisis y desarrollos orientados a objetos, construcción, prueba y despliegue. Permite diseñar todo tipo de diagrama de clases, ingeniería inversa, generación de código a partir de diagramas y generar documentación (38).

1.7.4 Lenguajes de desarrollo

Un lenguaje de programación es una herramienta que permite comunicar e instruir a la computadora para que realice una tarea específica. Los lenguajes de programación se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas. A continuación se detallan las características de dos lenguajes de programación, con el objetivo de seleccionar el más adecuado para el desarrollo de la aplicación.

1.7.4.1 Java

Java es un lenguaje de programación orientado a objetos. Desarrollado por la empresa Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de sus sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria. Con respecto a la memoria, su gestión no es un problema ya que ésta es gestionada por el propio lenguaje y no por el programador. Toda la programación en java en su mayoría está orientada a objeto, ya que al estar agrupados en estructuras encapsuladas es más fácil su manipulación. Permite abrir *sockets*, establecer y aceptar conexiones con los servidores o clientes remotos. Además facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red (39).

1.7.4.2 C++

C++ es un lenguaje de programación de propósito general que ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. No es un lenguaje de muy alto nivel y más bien

un lenguaje pequeño, sencillo y no está especializado en ningún tipo de aplicación. Esto lo hace un lenguaje potente, con un campo de aplicación ilimitado y sobre todo, se aprende rápidamente. En poco tiempo, un programador puede utilizar la totalidad del lenguaje (40).

Después de haber realizado un estudio entre estos dos lenguajes de programación se llegó a la conclusión que se utilizará el lenguaje Java ya que es un lenguaje de programación libre. Cuenta con la ventaja de poder ser usado en cualquier plataforma que cuente con la máquina virtual de Java instalada. Presenta una amplia documentación disponible para la comunidad de manera gratuita. Es muy importante mencionar que los desarrolladores del sistema cuentan con una amplia experiencia en el lenguaje Java y en el IDE de desarrollo para dicho lenguaje.

1.7.5 Entorno de desarrollo integrado (IDE)

El entorno de desarrollo es el conjunto de herramientas y procesos de programación utilizados para crear el programa o producto de software. Es aquel en el que los procesos y las herramientas se coordinan para ofrecer a los desarrolladores una interfaz ordenada y de cómoda visión del proceso de desarrollo, o al menos los procesos de la escritura de código, probándolo y empaquetándolo para su uso (41).

Para el desarrollo del sistema de identificación a desarrollar se decide utilizar el Entorno de Desarrollo Integrado Netbeans en su versión 8.0 ya que es una herramienta libre y gratuita sin restricciones de uso. Está escrito en Java pero puede soportar cualquier otro lenguaje de programación. A continuación se detallan características más precisas de dicha herramienta.

1.7.5.1 Netbeans

Es una herramienta que permite que los desarrolladores puedan escribir, depurar, compilar y ejecutar programas. Está escrito en Java pero soporta otros lenguajes de programación como C, C++, Python, CSS, PHP, HTML y Ruby. Es libre y gratuito sin restricciones de uso, tiene una interfaz amigable e intuitiva, posee todas las herramientas para crear aplicaciones profesionales ya sean de escritorio, empresariales, web y móviles. Es multiplataforma y posee una creciente comunidad de usuarios, permite la depuración y ejecución de programas escritos. Posee integración con bases de datos, permitiendo escribir instrucciones de eliminar

y actualizar. Sus funciones están previstas por módulos (42). Por las características que se describen se selecciona esta herramienta para el desarrollo del sistema.

1.7.6 Gestores de base de datos

Un sistema de gestión de base de datos (SGBD), es un software que permite introducir, organizar y recuperar la información de las bases de datos. Proporciona un control de la seguridad y privacidad de los datos, así como la mejora de su manipulación, facilita el proceso de diseño de aplicaciones, posibilita que los tratamientos sean más eficientes y rápidos, dando la mayor flexibilidad posible a los usuarios (43).

Sus objetivos fundamentales son:

- Independencia de los datos y los programas de aplicación.
- Minimización de la redundancia.
- Integración y sincronización de las bases de datos.
- Integridad de los datos.
- Seguridad y protección de los datos.
- Facilidad de manipulación de la información.
- Control centralizado.

1.7.6.1 PostgreSQL

Para el desarrollo del sistema de identificación de personas basado en características faciales se utilizará PostgreSQL en su versión 9.4 ya que su uso es gratuito bajo la licencia BSD de código abierto. Además es un gestor de bases de datos capaz de manejar una enorme cantidad de datos permitiendo el acceso simultáneo de un conjunto de usuarios. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo todos a la misma vez (44).

Las características más representativas del PostgreSQL se enuncian a continuación:

- Estabilidad.
- Potencia.

- Robustez.
- Facilidad de administración e implementación de estándares.

Soporta distintos tipos de datos como fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de datos de tipos propios. Incluye herencia entre tablas, por lo que a este gestor de bases de datos se le incluye entre los gestores objeto-relacionales. Cuenta con una regionalización por columna y múltiples métodos de autentificación.

1.7.7 Mapeo Objeto Relacional

Hibernate es una herramienta de Mapeo Objeto-Relacional (ORM por sus siglas en inglés) para la plataforma Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación. Para el desarrollo del sistema de identificación de personas basado en características faciales se utilizará Hibernate como herramienta ORM ya que permite transformar de forma automática los datos entre el lenguaje de programación orientado a objetos y el sistema de base de datos relacional, a través de un motor de persistencia. Esto surge debido a que la mayoría de las aplicaciones están diseñadas para usar la Programación Orientación a Objetos (POO), mientras que las bases de datos más extendidas son del tipo relacional. Estas bases de datos solo permiten guardar tipos de datos primitivos (enteros, cadenas de texto) impidiendo guardar de forma directa los objetos de la aplicación en las tablas, sino que estos se deben de convertir antes en registros. En el momento de volver a recuperar los datos, hay que hacer el proceso inverso de convertir los registros en objetos. En la transformación se obtiene como resultado una base de datos orientada a objetos virtual sobre la base de datos relacional (45).

Ventajas:

- Reutilización: Utiliza los métodos de un objeto desde distintas zonas de la aplicación como puede ser dentro o fuera de esta.
- Seguridad: Suelen implementar sistemas para evitar tipos de ataques como pueden ser las inyecciones de SQL.
- Mantenimiento del código: Facilita el mantenimiento del código debido al correcto orden de la capa de datos, haciendo que el mantenimiento del código sea mucho más sencillo.

- Encapsulación: Encapsula la lógica de los datos permitiendo hacer cambios que afectan a toda la aplicación.

1.7.8 Bibliotecas de clases utilizadas en el sistema

1.7.8.1 OpenCV

Para el desarrollo del sistema de identificación se utiliza la biblioteca de clases de visión artificial y aprendizaje automático Open CV en su versión 2.4.4. Estos algoritmos permiten identificar objetos, rostros, clasificar acciones humanas en vídeo, hacer tracking de movimientos de objetos, extraer modelos 3D, encontrar imágenes similares, eliminar ojos rojos, seguir el movimiento de los ojos, reconocer escenarios, etc. Su uso es gratuito bajo la licencia BSD [7] de código abierto (46). OpenCV ha sido desarrollada para aliviar la carga en la programación de aplicaciones para la visión artificial, aportando un gran número de funciones para el procesamiento de imágenes, y algoritmos numéricos de propósito general (47).

Esta biblioteca de clases actualmente se utiliza para el procesamiento de imágenes. Provee rutinas fundamentales para el tratamiento de imágenes estáticas o en movimiento y basa su funcionamiento en las técnicas estadísticas y de inteligencia artificial. Dentro de los diversos métodos que contiene la biblioteca de clases OpenCV, existe uno para la detección de rostros, que se basa en el algoritmo de Viola y Jones".

El reconocimiento de patrones es un área importante dentro de la visión artificial, donde se intenta descubrir si existen patrones conocidos en una imagen dada. El proceso que comprueba en la imagen la posible ubicación de un patrón se llama búsqueda de coincidencias de imágenes. Las técnicas de coincidencia constituyen la forma más sencilla de hacer el reconocimiento de patrones. Debido a que la visión y el aprendizaje por computadora están muy unidos, OpenCV incluye una biblioteca de clases completa llamada *Machine Learning Library* (MLL). Algunas de sus principales funciones son: (48)

- Clasificadores bayesianos.
- Algoritmos de clasificación por vecindad.

[7] La Licencia de Distribución de Software de Berkeley (Berkeley Software Distribution) no impone ninguna restricción a los desarrolladores de software en lo referente a la utilización posterior del código en derivados y licencias de estos programas. Este tipo de licencia permite a los programadores utilizar, modificar y distribuir a terceros el código fuente y el código binario del programa de software original con o sin modificaciones. Los trabajos derivados pueden optar a licencias de código abierto o comercial.

- Máquinas de soporte vectorial.
- Árboles de clasificación.
- Redes de neuronas.
- *Boosting*.
- *Random trees*.

1.7.8.2 WebCamCapture

WebCamCapture en su versión 0.3.10 es la biblioteca de clases que permite el uso de la cámara directamente desde Java. Está diseñada para funciones de la cámara de uso generalmente abstracto y para apoyar diversos marcos de captura. No requiere software adicional. Soporta múltiples plataformas como Windows, Linux, Mac OS y diversas arquitecturas (49).

1.8 Conclusiones parciales

En el presente capítulo se plantearon conceptos fundamentales que posibilitaron una mejor comprensión de la problemática. Se realizó un estudio de los diferentes algoritmos y técnicas para realizar el proceso de identificación, así como de sistemas similares a nivel nacional e internacional lo que permitió definir elementos significativos que sirvieran de base para elaborar la propuesta de solución. También se efectuó un estudio de las metodologías, lenguajes, tecnologías y herramientas. Este estudio ayudó a determinar y fundamentar, según las necesidades de la solución, la utilización de XP como metodología de desarrollo, Visual Paradigm como herramienta de modelado, Java como lenguaje de programación, Netbeans como entorno de desarrollo y PostgreSQL como gestor de base de datos. La investigación realizada permitió adquirir y fomentar conocimientos necesarios para el desarrollo de la propuesta de solución.

CAPÍTULO 2: ANÁLISIS Y DISEÑO

2.1 Introducción

El objetivo que se persigue con la elaboración de este capítulo es mostrar la evolución del sistema a desarrollar, primeramente se realiza un análisis de todos los elementos que se relacionan con el tema de identificación de personas basado en rasgos faciales. Luego se listan los requisitos funcionales y no funcionales. Además se define la arquitectura y el diseño a utilizar, se muestra el diagrama lógico de datos o diagrama de clases y al mismo tiempo se desarrollan las fases iniciales de la metodología de desarrollo XP, planificación y diseño.

2.2 Propuesta de solución del sistema

El sistema desarrollado es una aplicación de escritorio que permitirá identificar a una persona basándose en sus rasgos faciales, proceso que será posible una vez que se realice la captura de la imagen facial del usuario a identificar o se haya cargado la imagen desde un directorio fijo.

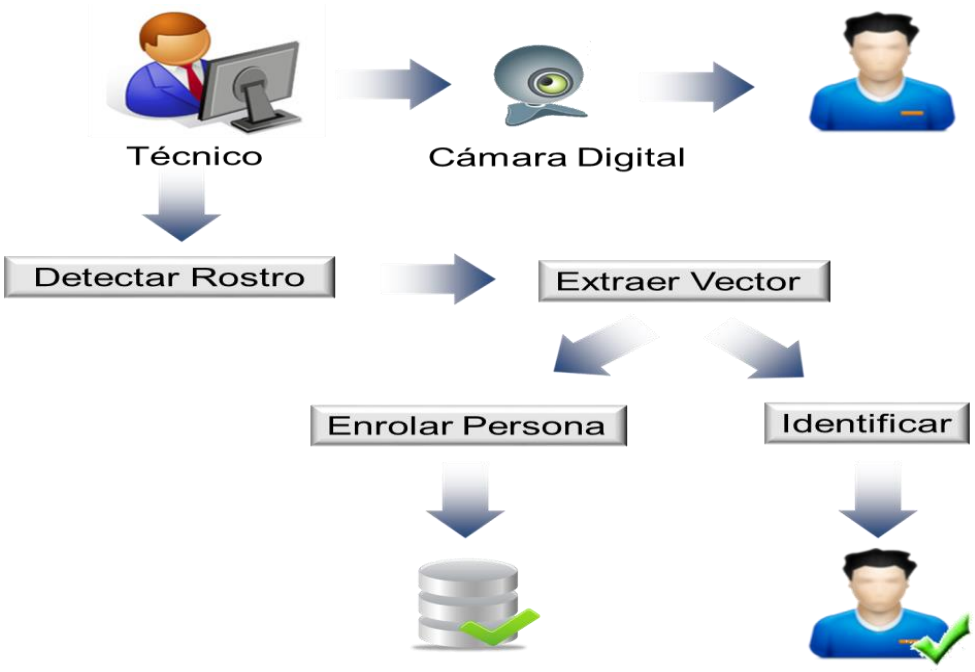

Figura 4: Etapas del sistema de identificación de personas.

Primeramente se debe capturar la imagen facial de la persona a identificar, este proceso en el sistema a desarrollar se realizará desde la cámara digital integrada a la computadora donde se encuentre instalado el sistema o desde una cámara digital externa. La misma debe poseer 1.3 megapíxeles de resolución o superior, vale destacar que a mayor resolución de la cámara la imagen tendrá mejor calidad. Para lograr que las imágenes capturadas posean valor identificativo se deben adoptar un conjunto de requisitos, con el fin de mejorar la precisión y efectividad del reconocimiento facial. En la **tabla 3** se ofrecen aquellos requisitos que se consideran importantes dentro de las especificaciones de la escena y las características fotográficas.

Parámetro	Especificaciones
Nitidez	Las imágenes no deben estar borrosas o desenfocadas.
Estado de los ojos	Los ojos deben estar abiertos de manera natural.
Pose	La imagen debe ser lo más frontal posible.
Expresión de la boca	La boca debe estar cerrada y tener una expresión neutral.
Mirada al frente	Los ojos deben estar mirando al frente.
Iluminación	El rostro no debe estar afectado por sombras, ni regiones brillantes.
Oclusión	Los peinados y otros accesorios no deben obstruir la imagen del rostro o los ojos.
Fondo uniforme	El fondo debe ser de colores claros.
No presencia de espejuelos	Solo se permite el uso de espejuelos si la persona normalmente los usa. Estos no deben oscurecer los ojos y deben ser cristales claros y transparentes de manera que las pupilas y el iris queden visibles.

Tabla 3: Parámetros seleccionados que influyen en el valor identificativo de las imágenes de rostros.

Luego de capturar la imagen de la persona, la misma deberá ser enrolada en el sistema. Durante este proceso de enrolamiento se hace necesario la obtención de los datos identificativos de la persona tales como: nombre, apellidos y carne de identidad. En caso de que la persona sea un usuario del sistema deberá especificar un nombre de usuario y una contraseña para acceder a las funcionalidades del sistema de identificación. De igual forma también se debe definir el rol a desempeñar dentro del sistema los cuáles pueden ser:

- Usuario básico: Solamente tiene acceso al proceso de identificación y modificar su contraseña.
- Administrador: Tiene acceso a todas las funcionalidades del sistema.

Una vez terminado el enrolamiento se procede a la detección del rostro. La detección se lleva a cabo utilizando el algoritmo AdaBoost de la biblioteca de clase OpenCV explicado anteriormente. La **figura 5** muestra cómo se realiza el proceso de detección en el sistema de identificación, mientras que la **figura 6** representa la imagen una vez recortada.

Figura 5: Detección del rostro de la persona.

Figura 6: Imagen recortada.

Culminado este proceso la imagen está lista para aplicarle la extracción de las características con el filtro correspondiente, la transformada discreta del coseno (DCT). De este proceso se obtiene como resultado final un vector característico asociado a la imagen procesada. En la primera etapa se realiza la división de la imagen de entrada en 64 bloques garantizando la invariabilidad de la resolución y el escalado.

Figura 7: División de la imagen en bloques.

Después de finalizada la etapa de división de la imagen, se selecciona un color como el más representativo de cada uno de estos 64 bloques. El resultado de la selección se conoce como imagen icono de tamaño 8x8 bloques.

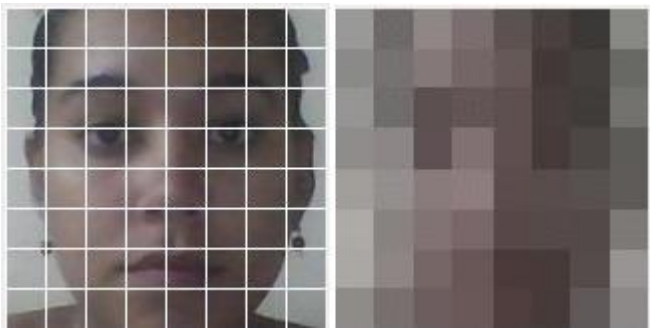

Figura 8: Selección del color más representativo de cada región.

Una vez que se ha obtenido la imagen icono, se realiza una conversión del espacio de color de la imagen original al espacio de color YCbCr formado por la crominancia Y, las crominancias azul y roja Cb y Cr respectivamente. La característica más básica del contenido visual es el color, es por eso que se pueden utilizar los colores para representar o describir una imagen. Una de las herramientas que se emplean para describir los colores es el Descriptor de la Distribución del Color (DDC), que permite describir la relación que existe entre secuencias o grupos de imágenes. Este descriptor se obtiene aplicando la transformada discreta del coseno (DCT) obteniendo de esta forma 3 grupos de 64 coeficientes. Para calcular la DCT de una matriz 2D se emplea la siguiente ecuación:

$$B_{pq} = a_p a_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

$$0 \leq p \leq M-1, 0 \leq q \leq N-1$$

$$a_{p=} \begin{cases} \frac{1}{\sqrt{M}} \\ \sqrt{\frac{2}{M}} \end{cases} \quad p = 0; 1 \leq p \leq M-1$$

$$a_q = \begin{cases} \frac{1}{\sqrt{N}} \\ \sqrt{\frac{2}{N}} \end{cases} \quad q = 0; 1 \leq q \leq N-1$$

Ecuación 4: Fórmula de la Transformada Discreta del Coseno. (28)

Luego de ser aplicada la fórmula de DCT se realiza la exploración en zigzag con los tres grupos de 64 coeficientes DCT dando como respuesta el vector característico. El objetivo de realizar el recorrido en zigzag guarda relación con la ubicación de las componentes de baja frecuencia de la imagen localizadas en la parte superior izquierda de la matriz transformada, es decir; la energía de la imagen se concentra en dicha localización. El trazado comienza por la esquina superior izquierda y termina en la esquina inferior derecha.

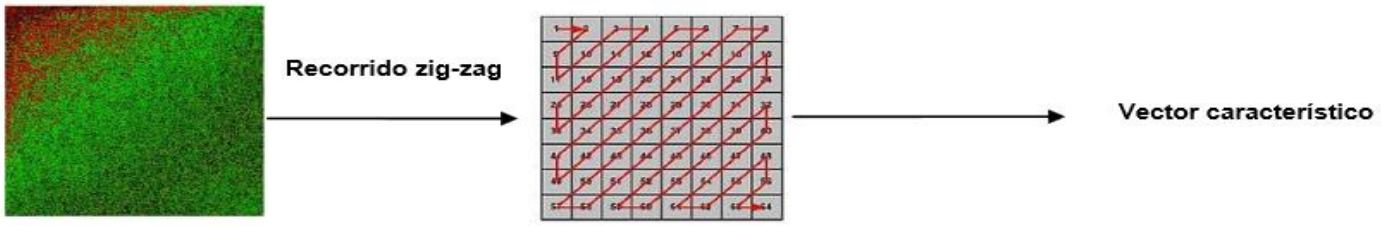

Figura 9: Recorrido en zigzag.

El sistema contará con una base de datos que contendrá las imágenes faciales de diferentes personas. En esta base de datos se almacena el vector característico que se desea identificar, este es comparado con cada uno de los vectores ya anteriormente calculados y almacenados en la base de datos utilizando la fórmula para hallar la distancia entre dos vectores, arrojando como resultado la aceptación o rechazo. En caso de ser identificada la persona se mostrarán los datos identificativos de la misma como son: nombre, apellidos, carné de identidad y sexo.

2.3 Modelo de dominio

El modelo de dominio es una representación visual de clases conceptuales o de objetos reales en un marco de interés. Consiste en un conjunto de diagramas de clases, sin definición de operaciones, donde se pueden representar los atributos de las clases, las relaciones entre las entidades que interactúan en el dominio del problema así como su cardinalidad (51).

Para la mejor comprensión de la propuesta de solución que se plantea, se realiza un modelo de clases del dominio, donde están representadas las clases conceptuales identificadas. Con este modelo se pretende contribuir a la comprensión de la solución, esquematizando las entidades dejando claro el entorno y el alcance de la propuesta de solución para el sistema de identificación de personas basado en características faciales.

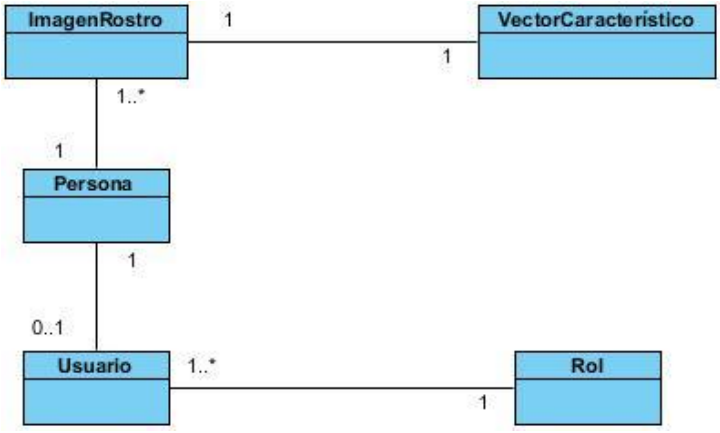

Figura 10: Diagrama del modelo de dominio.

2.4 Fase de planificación

La planificación es una etapa importante de todo proyecto en XP. En este punto se comienza a interactuar con el cliente y el resto del grupo de desarrollo para descubrir los requisitos del sistema. Se identifican el número y tamaño de las iteraciones al igual que se plantean ajustes necesarios a la metodología según las características del proyecto.

2.4.1 Actores del sistema

Un actor del sistema representa el rol que juega una o varias personas, un equipo o un sistema automatizado que interactúa con el sistema y que es externo a él. En la **tabla 4** se muestran los actores identificados en el sistema de identificación de personas.

Actores del sistema	Descripción
Administrador	Es el encargado de gestionar los usuarios y las personas del sistema y conjuntamente puede realizar el proceso de la captura del rostro de la persona a identificar.
Técnico	Es el encargado de realizar la captura del rostro de la persona a identificar.

Tabla 4: Actores del sistema.

2.4.2 Especificación de requisitos

La especificación de requisitos es la base que permite verificar si se alcanzaron o no los objetivos establecidos en el proyecto, ya que estos son un reflejo detallado de las necesidades de los clientes o usuarios del sistema. La especificación de requisitos es una descripción completa del comportamiento del sistema que se va a desarrollar. En esta etapa se debe generar una información clara y precisa de los aspectos más relevantes del producto, ya que esta actividad es el hilo conductor de todo el desarrollo del software. A continuación se describen los requisitos funcionales y no funcionales que fueron definidos para el sistema de identificación de personas basado en características faciales (52).

2.4.2.1 Requisitos funcionales (RF)

A continuación se muestran los requisitos funcionales definidos para la propuesta de solución.

Nombre	Requisito funcional	Complejidad
RF1	Autenticar Usuario.	Media
RF2	Capturar imagen con dispositivo de video.	Alta
RF3	Cargar imagen.	Baja
RF4	Guardar imagen.	Baja
RF5	Procesar imagen.	Alta
RF5.1	Detectar rostro en imagen.	Alta
RF5.2	Extraer características faciales.	Alta
RF5.3	Comparar características faciales	Alta
RF6	Gestionar usuarios.	Alta
RF6.1	Registrar usuario.	Media
RF6.2	Modificar usuario.	Media
RF6.3	Eliminar usuario.	Media
RF6.4	Listar usuarios.	Media
RF7	Gestionar personas.	Alta
RF7.1	Registrar persona.	Media
RF7.2	Modificar persona.	Media
RF7.3	Eliminar persona.	Media
RF7.4	Listar personas enroladas.	Media

Tabla 5: Requisitos funcionales.

2.4.2.2 Requisitos no funcionales (RNF)

Los requisitos no funcionales son aquellos que se refieren a las propiedades o cualidades que el producto debe tener. Son características que hacen al producto atractivo, usable, rápido o confiable (51).

1. **Requisitos de software**

   Las computadoras en las que se utilizará el software, deben tener instalado Linux o Windows XP/Vista/7/8 como sistema operativo, los controladores de la cámara digital, el JRE (*Java Runtime Environment*) versión 7 o superior el cual permite la ejecución de los programas de Java y la biblioteca de clases OpenCV versión 2.0 o superior.

2. **Requisitos de hardware**

   Las computadoras en las que se utilice el sistema deben poseer un microprocesador de 1.0 GHZ o superior y 1GB de Memoria RAM o superior. Debe contar con una cámara digital integrada o externa con una resolución mínima de 1.3 megapíxeles o superior.

3. **Requisitos de diseño e implementación**

   Se usará como entorno de desarrollo integrado Netbeans 8.0, lenguaje de programación Java, herramienta de modelado UML Visual Paradigm 8.0, biblioteca para el reconocimiento facial OpenCV 2.4.4 y como gestor de bases de datos PostgreSQL 9.4.

4. **Requisitos de apariencia e interfaz**

   La interfaz debe utilizar colores frescos y ofrecer suficiente contraste entre texto y fondo para no dificultar la lectura. La aplicación debe utilizar como idioma el español.

5. **Requisitos de seguridad**

   Se asignarán los permisos de acceso y de edición en dependencia del rol que desempeñe cada usuario del sistema. Se permitirá realizar copias de seguridad de la base de datos hacia otro dispositivo de almacenamiento externo. La información manejada por el sistema estará protegida de acceso no

autorizado y divulgación. Las contraseñas de los usuarios del sistema serán encritpadas utilizando el algoritmo SHA-1.

6. Requisitos de soporte

El sistema debe tener una arquitectura flexible, capaz de permitir cambios sin que afecten la estructura de la solución.

7. Requisitos de portabilidad

El sistema será independiente de aplicaciones ajenas para su funcionamiento.

2.4.3 Historias de Usuarios (HU)

Las historias de usuario (HU) son aquellas que representan las funcionalidades que el cliente desea que estén presentes en el sistema; por lo que todo el trabajo realizado debe estar enfocado a satisfacer las expectativas del cliente. A continuación se describe la historia de usuario de autenticar usuario. El resto de las historias de usuario se encuentran descritas en los anexos. (Ver Anexos 4.1)

Historia de Usuario	
Número: 1	**Usuario:** Técnico, Administrador
Nombre historia: Autenticar Usuario.	
Prioridad en el negocio: Media	**Riesgo en desarrollo:** Media
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: David Fonseca Martínez	
Descripción: El técnico o el administrador tendrán la posibilidad de autenticarse en el sistema.	
Observaciones: Si el sistema no identifica el usuario este mostrará un mensaje comunicando que sus datos no son válidos.	

Tabla 6: Descripción de la HU-Autenticar Usuario.

2.4.4 Plan de iteraciones

El plan de iteraciones tiene como entrada las especificaciones de requisitos descritas anteriormente. En el mismo se establecen las iteraciones necesarias para desarrollar el producto y se define qué requisitos se deben implementar en cada una de estas. El objetivo de cada iteración es obtener una versión funcional del sistema que, aunque no cuente con todos los requisitos definidos, constituya un resultado de valor para el cliente (52).

Iteración	Código	Historias de usuario	Estimación (Semanas)
Iteración 1	HU2	Capturar imagen con dispositivo de video.	05/01/2015-09/01/2015
	HU3	Cargar imagen.	10/01/2015-14/01/2015
	HU4	Guardar imagen.	15/01/2015-19/01/2015
Iteración 2	HU1	Autenticar Usuario.	26/01/2015-02/02/2015
	HU6	Gestionar usuarios.	03/02/2015-16/02/2015
	HU7	Gestionar personas.	03/02/2015-16/02/2015
Iteración 3	HU5	Procesar imagen.	24/02/2015-17/03/2015

Tabla 7: Plan de iteraciones.

- ✓ La iteración 1 tendrá un total de 2 semanas de duración.
- ✓ La iteración 2 tendrá un total de 3 semanas de duración.
- ✓ La iteración 3 tendrá un total de 3 semanas de duración.

2.4.5 Plan de entregas.

Una vez que se concluye la tarea por parte del cliente de elaborar las distintas HU, se comienza con la creación del plan de entrega, para estimar el tiempo de desarrollo de las mismas. Este artefacto se elabora con la intención de fijar qué período de tiempo puede tardar la implementación de cada una de las historias en cada iteración, definiéndose las fechas en que serán liberadas las versiones funcionales del producto.

1ra Entrega (Iteración 1)	**2da Entrega (Iteración 2)**	**3ra Entrega (Iteración 3)**

19 de enero de 2015	16 de febrero de 2015	17 de marzo de 2017

Tabla 8: Plan de entrega.

2.5 Fase de diseño

La metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando el Lenguaje Unificado de Modelado (UML); en su lugar usa una técnica para representar clases nombrada tarjeta Clase-Responsabilidad-Colaboración (CRC).

2.5.1 Tarjetas CRC

La técnica CRC propone una forma de trabajo, preferentemente grupal, para encontrar los objetos del dominio de la aplicación, sus responsabilidades y cómo colaboran con otros para realizar tareas. Esta técnica utiliza las llamadas ***tarjetas CRC***, las cuáles registran el nombre de las clases, sus responsabilidades y las otras clases con la que colaboran (53).

La **tabla 9** muestra la tarjeta CRC de la clase VectorCaracteristico perteneciente al paquete del Motor de Reconocimiento Facial, citar que las clases que no se describen en el epígrafe se encuentran en los anexos.(Ver Anexos 4.2).

Nombre de la Clase :VectorCaracteristico	
Responsabilidades	**Colaboradores**
1. getIdvector **2.** getImagen **3.** getSuma **4.** getPersona **5.** getVector	**1.** Persona **2.** SistemaControler

Tabla 9: Tarjeta CRC correspondiente a la clase VectorCaracteristico.

2.5.2 Diagramas de clases del diseño

Para una mejor comprensión del funcionamiento del sistema a desarrollar se realizan los diagramas de clases del diseño. Para tener una mejor organización de las clases que conforman el sistema de identificación de personas basado en características faciales, se decidió diseñar el siguiente diagrama de paquetes (**ver figura 11**), el cual está compuesto por tres sub-paquetes, el primero es Gestión de usuarios, en él se encuentran las clases que permiten la gestión de los usuarios del sistema (**ver figura 12**), el segundo es nombrado Motor de Reconocimiento Facial donde se encuentran las clases que rigen el proceso de identificación de personas basado en rasgos faciales y el tercero se llama Seguridad (**ver figura 13**) y éste contiene otro paquete nombrado Excepciones donde se produce el tratamiento de las mismas (**ver figura 14**).

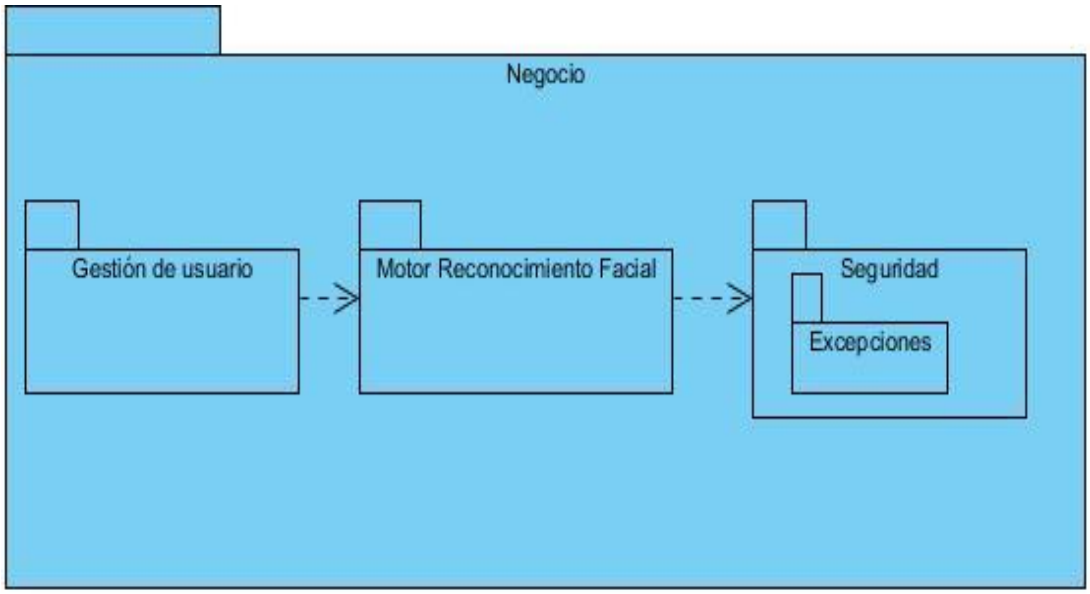

Figura 11. Diagrama de paquetes.

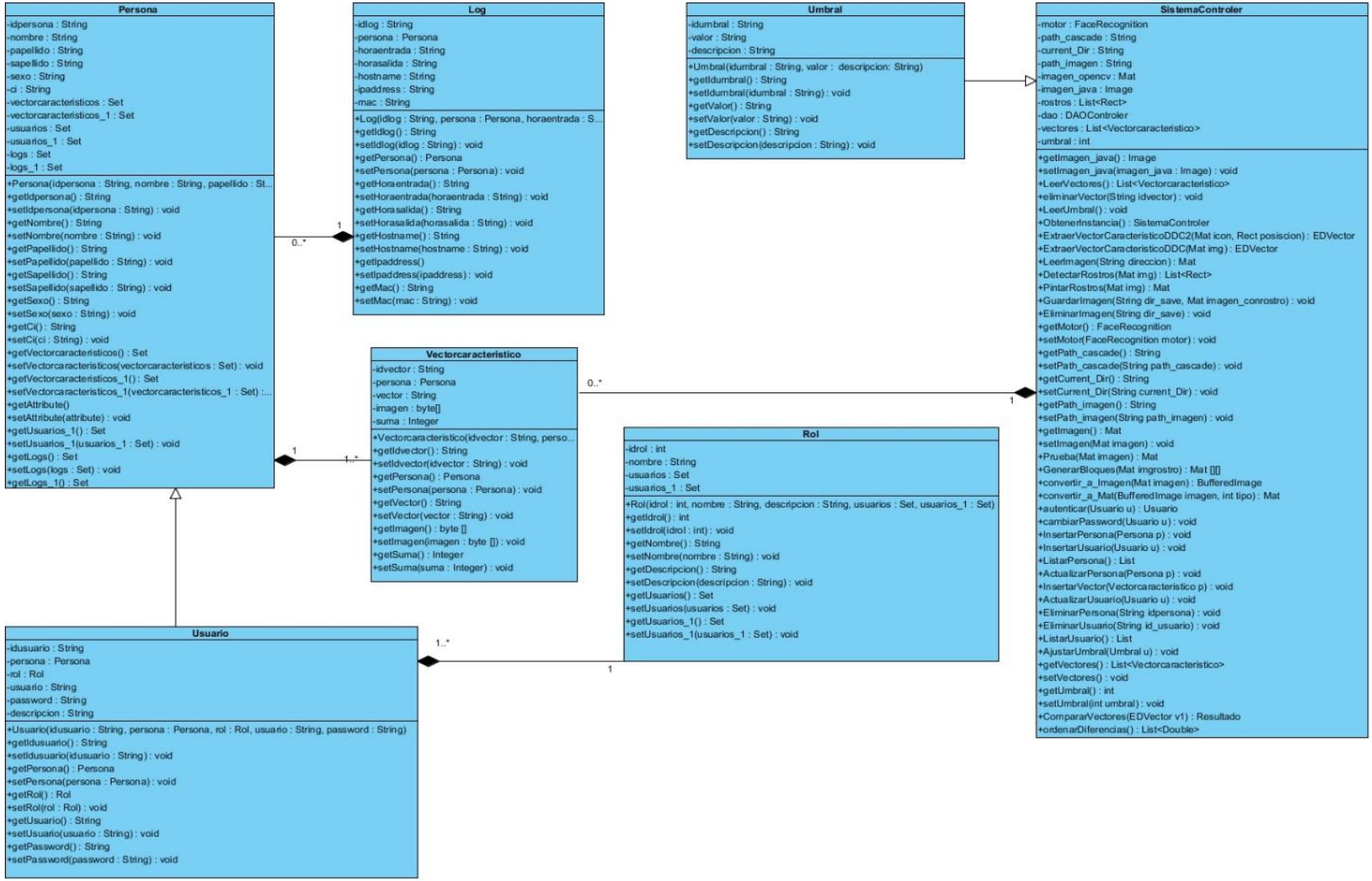

Figura 12: Diagrama de clases pertenecientes al sub-paquete Gestión de usuarios.

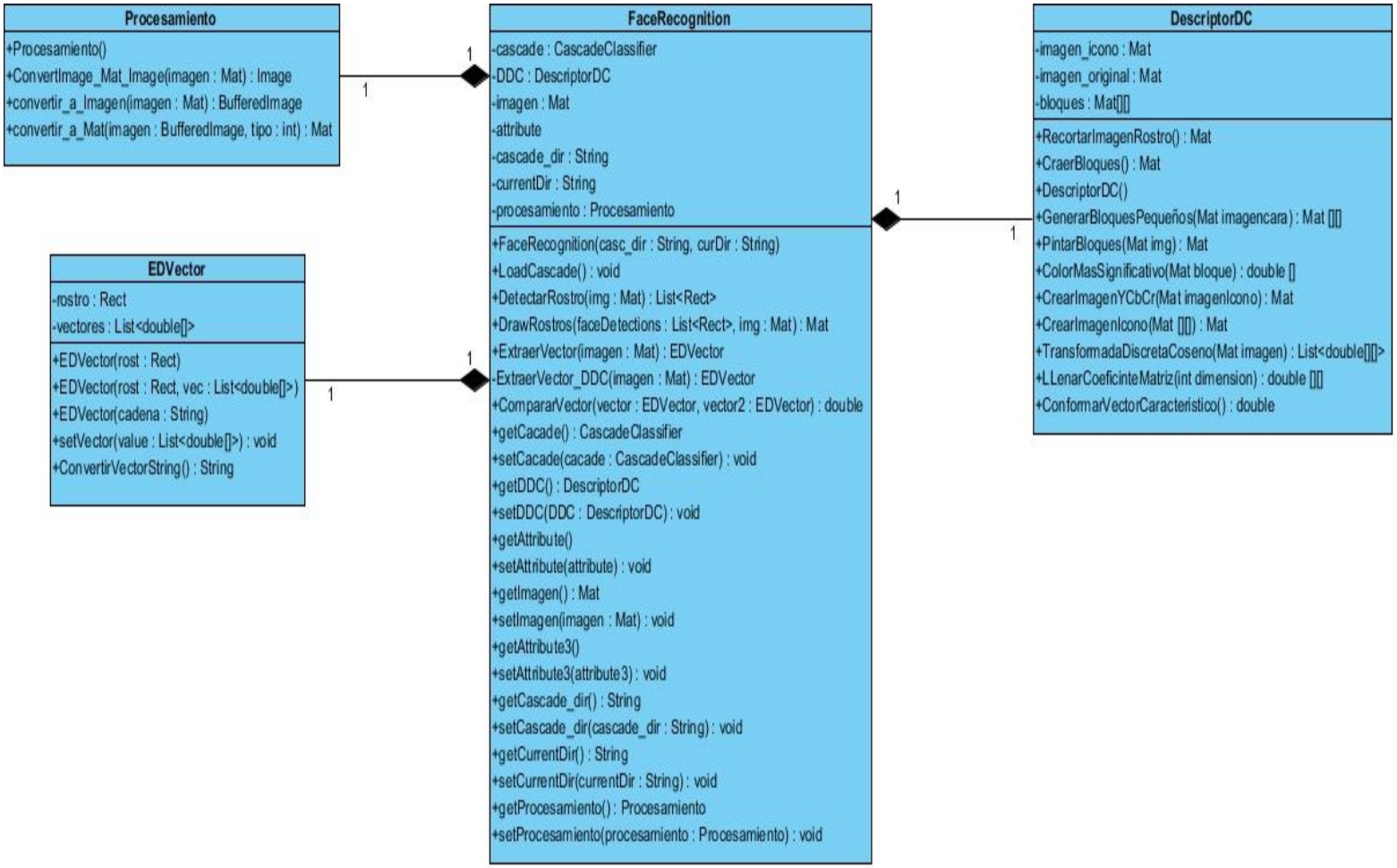

Figura 13: Diagrama de clases del Motor de reconocimiento facial.

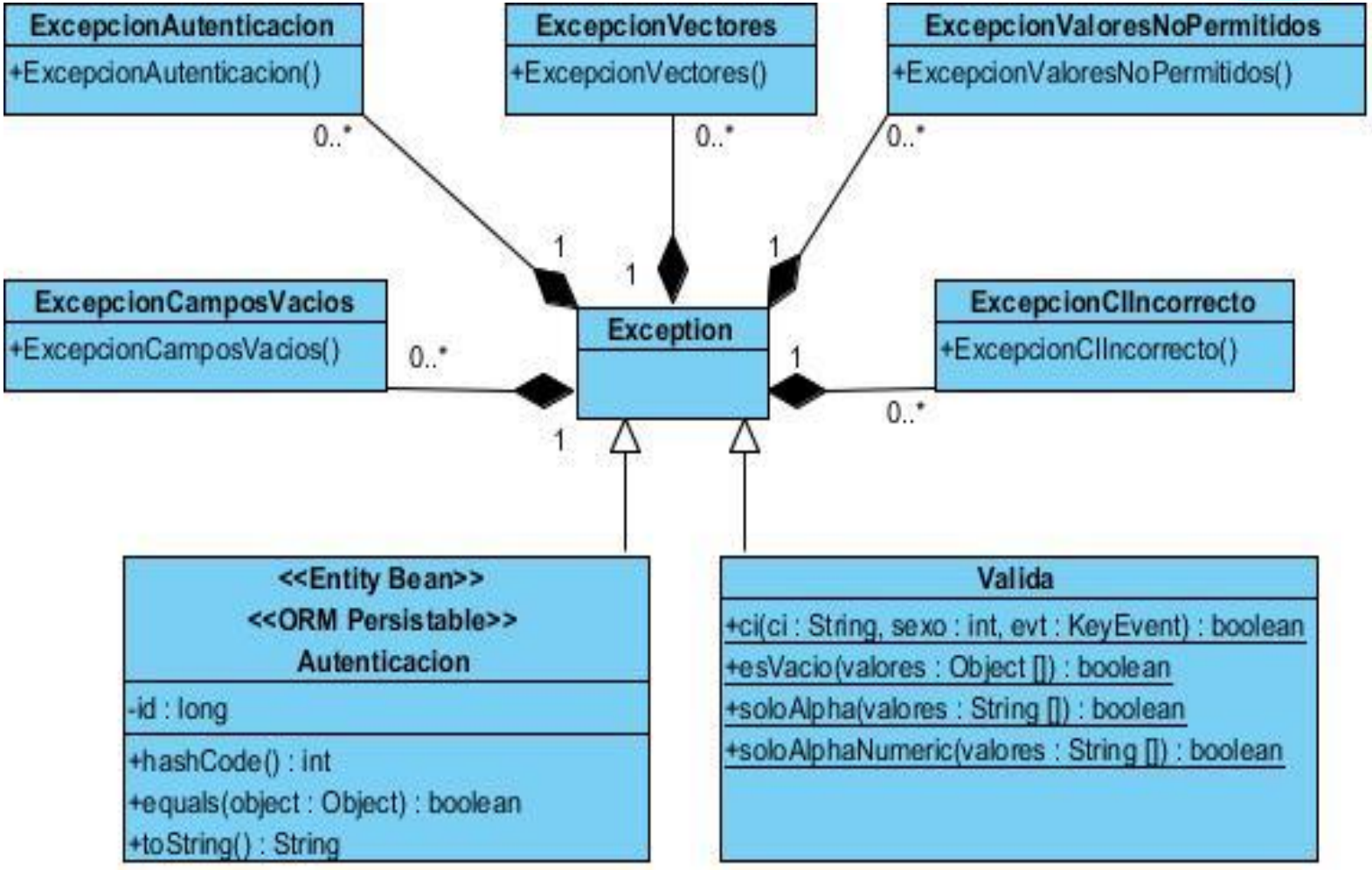

Figura 14: Diagrama de clases perteneciente al sub-paquete Seguridad.

2.5.3 Modelo de datos del sistema

El modelo de datos del sistema de identificación de personas basado en rasgos faciales está compuesto por diferentes tablas como son:

1. Persona: Esta tabla almacena la información relacionada con las personas registradas en el sistema.
2. Log: En esta tabla se registran los datos de la PC mediante la cual el usuario accede al sistema.
3. Usuario: Esta tabla almacena la información del usuario del sistema asociada a una persona.
4. Vector característico: En esta tabla se almacena el vector característico que distingue a cada persona en el sistema, con la imagen que la identifica.
5. Rol: Esta tabla almacena los tipos de roles y la descripción de cada uno de ellos.

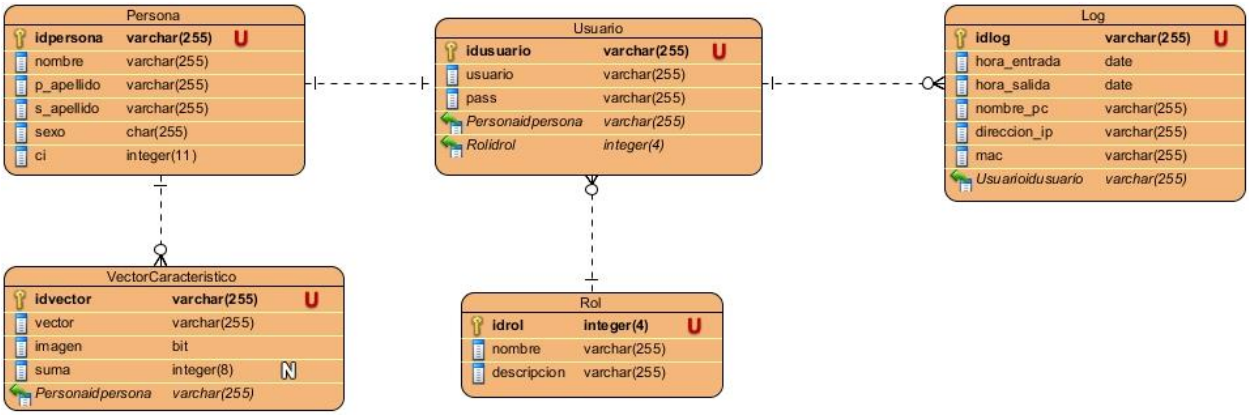

Figura 15. Diagrama de Entidad-Relación.

2.5.4 Arquitectura del sistema

La arquitectura de software es un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del sistema. Las arquitecturas y estilos arquitectónicos más universales se abordan a continuación: (51)

2.5.4.1 Cliente-servidor.

La arquitectura Cliente-Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones al servidor y este emite la respuesta. Normalmente, el servidor es una máquina bastante potente que puede actuar como servidor de aplicación, de depósito de datos, o sencillamente brindar determinados servicios (53). Por otro lado, los clientes suelen ser estaciones de trabajo que realizan varias solicitudes o peticiones al servidor. Ambas partes deben estar conectadas entre sí mediante una red.

2.5.4.2 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 cuáles se ofrecen como servicios. La forma más habitual de implementarla es mediante Servicios Web, una tecnología basada en estándares e independiente de la plataforma, con la que SOA puede descomponer aplicaciones monolíticas en un conjunto de servicios e implementar esta funcionalidad en forma modular (55).

2.5.4.3 Modelo Vista Controlador (MVC)

El Modelo-Vista-Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario; y la lógica de control en tres componentes distintos, permitiendo realizar una programación multicapa (37).

2.5.4.4 Arquitectura n-capas.

Se basa en una distribución jerárquica de roles y responsabilidades tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. Las interacciones entre las capas ocurren generalmente por invocación de métodos; por definición, los niveles más bajos no deben poder utilizar funcionalidades ofrecidas por las capas superiores. Este estilo facilita la modularidad del sistema, la localización de errores y mejora considerablemente el soporte del sistema.

Para el desarrollo del sistema de identificación de personas basado en características faciales se utilizó la arquitectura n-capas como línea base para la organización estructural de la aplicación. El sistema se divide en las siguientes capas:

Capa de presentación**:** La capa de presentación es la que ve el usuario. La misma presenta el sistema al usuario, le comunica la información y captura la información del usuario. Esta capa se comunica únicamente con la capa de negocio. En ella se encuentra contenida la interfaz visual del sistema de identificación a desarrollar.

Capa de negocio**:** En la capa de negocio es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de acceso a datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él. Esta capa contiene todas las funcionalidades del sistema de identificación, las clases pertenecientes al motor de reconocimiento facial así como las bibliotecas de clases OpenCV y WebCamCapture.

Capa de acceso a datos**:** En la capa de acceso a datos es donde residen los datos y se encarga de acceder a ellos. Además se reciben solicitudes de almacenamiento o recuperación de información desde la capa de negocio. En el sistema de identificación se utiliza en esta capa PostgreSQL como gestor de bases de datos para realizar el almacenamiento de datos. También se utiliza Hibernate como herramienta de Mapeo objeto-relacional (ORM) que facilita el mapeo de atributos entre la base de datos y el modelo de objetos del sistema.

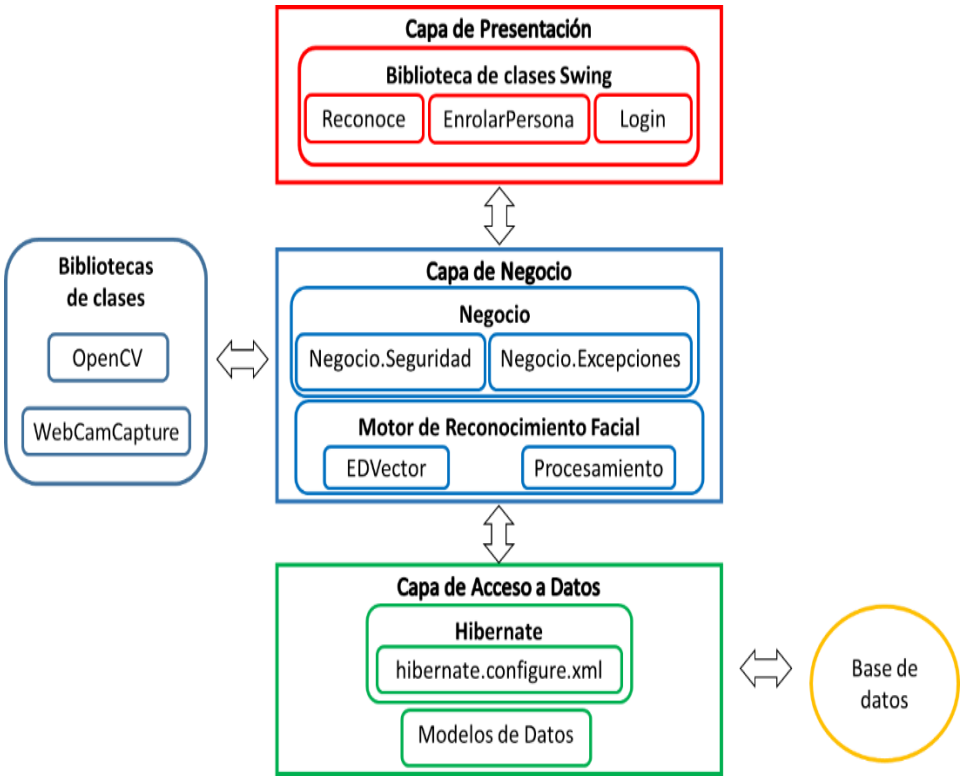

Figura 16. Arquitectura del sistema de identificación de personas.

El estilo arquitectónico propuesto para el desarrollo del sistema de identificación es Tuberías y filtros ya que provee una estructura para procesar flujos de datos. El sistema se muestra como una sucesión de transformaciones que sufre una serie de datos de entrada. Los datos ingresan al sistema y fluyen a través de los filtros uno a uno hasta que se asignan a un destino final: salida o almacenamiento. El sistema se divide en varios pasos secuenciales de procesamiento; los pasos se conectan con flujos de datos; cada filtro consume y procesa sus datos en forma incremental; el destino y los filtros se conectan con tubos que implementan el flujo de datos entre los pasos de procesamiento. Estos pasos se muestran en la **figura 17**:

- Entrada: Imagen del rostro capturada por medio de una cámara digital.
- Filtro #1: Detección del rostro en la imagen de entrada.
- Filtro #2: Extracción de las características.
- Salida: Vector característico.

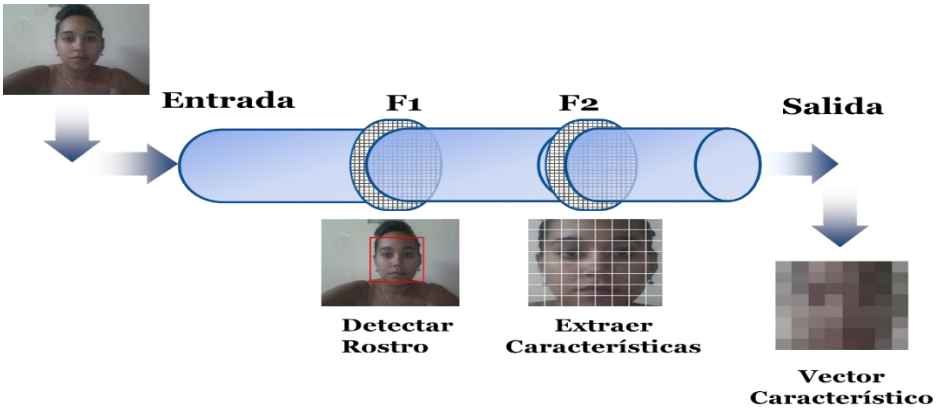

Figura 17: Estilo arquitectónico tuberías y filtros del sistema.

2.5.5 Patrones de diseño utilizados

Los patrones de diseño son los principios generales de soluciones que aplican ciertos estilos que ayudan a la creación de software. Un patrón de diseño resulta ser una solución a un problema de diseño. Para que una solución sea considerada un patrón debe poseer ciertas características. Una de ellas es que debe haber comprobado su efectividad resolviendo problemas similares en ocasiones anteriores. Otra es que debe ser reutilizable, lo que significa que es aplicable a diferentes problemas de diseño en distintas circunstancias (54).

2.5.5.1 Patrones para asignar responsabilidades (GRASP).

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. El nombre se eligió para indicar la importancia de captar estos principios, si se quiere diseñar eficazmente el software orientado a objetos (55). A continuación se mencionan los patrones GRASP utilizados en el desarrollo de la solución:

- Experto en información: Se utiliza porque conserva el encapsulamiento, pues los objetos se valen de su propia información para realizar la tarea encomendada. Esto soporta un bajo acoplamiento, lo que favorece al hecho de tener sistemas más robustos y de fácil mantenimiento. Además, el comportamiento se distribuye entre las clases que cuentan con la información requerida, haciéndolas más fáciles de comprender y de mantener, garantizando una alta cohesión (58). Este patrón se utiliza

en la clase controladora ***SistemaControler***, que posee la información necesaria para cumplir con cada una de las responsabilidades que le corresponden.

- Creador: El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El patrón se pone de manifiesto en la clase ***FaceRecognition*****,** encargada de crear y guiar la asignación de responsabilidades relacionadas con la creación de instancias de las clases **Procesamiento**, **EDVector** y **DescriptorDC**.
- Alta Cohesión: Este patrón permite asignar responsabilidades de modo que la cohesión siga siendo alta. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Una clase con alta cohesión es útil porque es fácil darle mantenimiento, entenderla y reutilizarla (58). En las clases **Persona**, **Usuario** y **Rol** del sistema de identificación se puede encontrar alta cohesión de modo que cada clase solo se encarga de los eventos a los cuales se le ha asignado la responsabilidad.
- Bajo Acoplamiento: Cuándo una clase de objetos puede realizar sus tareas, sin depender de ninguna otra clase de objetos entonces se dice que hay bajo acoplamiento (58). Las clases deben comunicarse con un número pequeño de clases tanto como sea posible. Este patrón se utiliza en la clase **Rol** de tal forma que en caso de producirse una modificación en esta clase, se tenga la mínima repercusión posible en el resto de las clases.
- Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa (58). Este patrón se utiliza en la clase controladora del sistema llamada ***SistemaControler*** que se encargan de obtener los datos y enviarlos a las bibliotecas de clases y las capas.

2.5.5.2 Patrones GOF

Los patrones GoF, describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación entre clases, la combinación de clases y la formación de estructuras de mayor complejidad. Permiten crear grupos de objetos para ayudar a realizar tareas complejas (59).

- *Singleton*: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia (59).

Ejemplo:

```
public class SistemaControler {

 private static SistemaControler sistema;

private SistemaControler(){
        current_Dir = System.getProperty("user.dir");
        path_cascade = current_Dir + "/Cascade/haarcascade_frontalface_alt.xml";
        path_imagen = current_Dir + "/Imagen/4.jpg";
        motor = new FaceRecognition(getPath_cascade(), getCurrent_Dir());
        rostros = new ArrayList<Rect>();
        dao = DAOControler.ObtenerInstanciaDAO();
        LeerUmbral();
        LeerVectores();
}

 public static SistemaControler ObtenerInstancia() {
 if (sistema == null) {
            sistema = new SistemaControler();
        }
 return sistema;
    }
  }
```

- *Facade* (Fachada): Provee una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema (59). Este patrón se evidencia en la clase controladora del sistema *SistemaControler*.
- *Interpreter* (Intérprete): Se emplea para definir un lenguaje para definir expresiones regulares que representen cadenas a buscar dentro de otras cadenas (59). Ejemplo de este patrón se encuentra la clase Valida.

2.6 Conclusiones parciales

Luego de definir las características que debe cumplir el sistema y de realizar el análisis y diseño correspondiente, se concluye que durante el desarrollo del capítulo se precisaron las diferentes historias de usuario definiendo así un plan de duración y un plan de entrega para la realización del sistema, permitiendo que el mismo cumpla con las funcionalidades requeridas y con el tiempo estimado. Además se seleccionó la arquitectura n-capas que garantiza el desarrollo de la aplicación con el uso apropiado de los patrones de diseño para lograr un software con calidad.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS

3.1 Introducción

Luego de realizar el diseño del sistema propuesto y de definir su arquitectura, se procede a implementar un software que cumpla con los requisitos que se necesitan. El presente capítulo recoge los aspectos relacionados con la implementación y las pruebas a realizar al sistema de identificación de personas basado en características faciales. Se detallan las iteraciones llevadas a cabo durante la etapa de construcción de la aplicación, mostrando los principales artefactos de la implementación del mismo. Además se muestran los resultados de las pruebas de funcionalidad y efectividad al sistema implementado.

3.2 Estándar de codificación

Un estándar de codificación puede asegurar que los programadores del proyecto trabajen de forma coordinada y permite la reutilización del código (60). Para la realización del sistema es necesario definir un criterio único e invariable que permita seguir un conjunto de reglas para la creación de nombres para las variables y los métodos en la fase de implementación, que proporcione un mejor entendimiento del código y una mejor lectura del software. A continuación se describe el estándar de codificación utilizado en la implementación de la propuesta de solución.

Reglas para la codificación

1. La longitud de las líneas de código no debe ser mayor de 80 caracteres aproximadamente, para que puedan ser manejadas de la mejor manera por las herramientas.
2. Cada línea de código debe contener solo una sentencia.
3. Cada funcionalidad debe contener un comentario que explique su funcionamiento.
4. Se deben seguir las convenciones de nombres que se muestran en la **tabla 13.**

Tipos de identificadores	Reglas de nombres	Ejemplos
Clases o interfaces.	La primera letra del nombre de las clases o las interfaces debe ser en mayúscula.	Persona, SistemaControler

Métodos.	Los nombres de los métodos deben reflejar la acción a realizar y deben comenzar con letra mayúscula. En caso de ser un nombre compuesto ambas palabras deben comenzar en mayúscula.	ListarPersona, AbrirSesion, EliminarUsuario
Variables.	Los nombres de las variables deben comenzar con letra minúscula. En caso de ser un nombre compuesto ambas palabras deben comenzar en minúscula.	idpersona, papellido.
Constantes	Cada carácter que pertenezca al nombre de la constante se escribirá en mayúscula y en caso de ser un nombre compuesto, cada palabra se separará por un guion bajo "_".	PI.

Tabla 10: Convenciones de nombres.

3.3 Tareas de ingeniería

La metodología de desarrollo de software XP define que la implementación de cualquier sistema debe hacerse a través de iteraciones, donde se obtiene al finalizar cada iteración un producto funcional, que debe ser probado y mostrado al cliente. Durante el curso de estas iteraciones se realiza la implementación de las HU definidas por el cliente y descritas en la etapa de planificación por el equipo de desarrollo. Estas HU se descomponen en tareas de ingeniería que luego son asignadas a los programadores para que sean implementadas en la iteración correspondiente (9). En la **tabla 11** se muestran las tareas correspondientes a las HU en la primera iteración del proceso de desarrollo del sistema de identificación de personas basado en características faciales. En la **tabla 12** se muestran las tareas correspondientes a las HU en la segunda iteración del proceso de desarrollo de la propuesta de solución.

Iteración 1	
Historias de Usuario	**Tareas**
Capturar imagen con dispositivo de video	**1.** Verificar que existe dispositivo de video conectado **2.** Obtener la imagen para el procesado
Cargar imagen	**1.** Buscar directorio donde se encuentra la imagen

Guardar imagen	**1.** Especificar directorio donde desea guardar la imagen capturada
Procesar imagen	**1.** Detectar el rostro en la imagen obtenida. **2.** Realizar el pre-procesado de la imagen captura. **3.** Extracción del vector característico. **4.** Comparación con la base de datos.

Tabla 11: Tareas de ingeniería en la primera iteración.

Iteración 2	
Historias de Usuario	**Tareas**
Autenticar usuario	**1.** Validar campos de entrada. **2.** Encriptar combinación de contraseña. **3.** Verificar autenticación del usuario. **4.** Mostrar mensaje de autenticación correcta o incorrecta en el sistema.
Gestionar persona	**Insertar persona** **1.** Validar campos de entrada. **2.** Procesado de las imágenes capturadas. **3.** Mostrar mensaje de enrolamiento **Eliminar persona** **1.** Seleccionar la persona que desea eliminar. **2.** Eliminar la persona de la base de datos. **3.** Mostrar mensaje de persona eliminada correctamente.

	Modificar persona **1.** Seleccionar la persona a modificar. **2.** Validar campos de entrada. **3.** Modificar los datos de la persona en la base de datos. **4.** Mostrar mensaje de modificación efectuada con éxito. **Listar persona** **1.** Listar personas enroladas.
Gestionar usuario	**Insertar usuario** **1.** Validar campos de entrada. **2.** Procesado de las imágenes capturadas. **3.** Mostrar mensaje de enrolamiento. **Eliminar usuario** **1.** Seleccionar el usuario que desea eliminar. **2.** Eliminar el usuario de la base de datos. **3.** Mostrar mensaje de usuario eliminado correctamente. **Modificar usuario** **1.** Seleccionar el usuario a modificar. **2.** Validar campos de entrada. **3.** Modificar los datos del usuario en la base de datos. **4.** Mostrar mensaje de modificación efectuada con éxito. **Listar usuario** **1.** Listar usuarios enrolados.

Tabla 12: Tareas de ingeniería en la segunda iteración.

3.4 Diagrama de componentes

Los diagramas de componentes son usados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Muestran las opciones de realización incluyendo código fuente, binario y ejecutable. Los componentes representan todos los tipos de elementos que entran en la fabricación de aplicaciones informáticas permitiendo conocer a los desarrolladores y clientes la estructura física que tiene el sistema y cómo se relacionan sus partes. Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente (60).

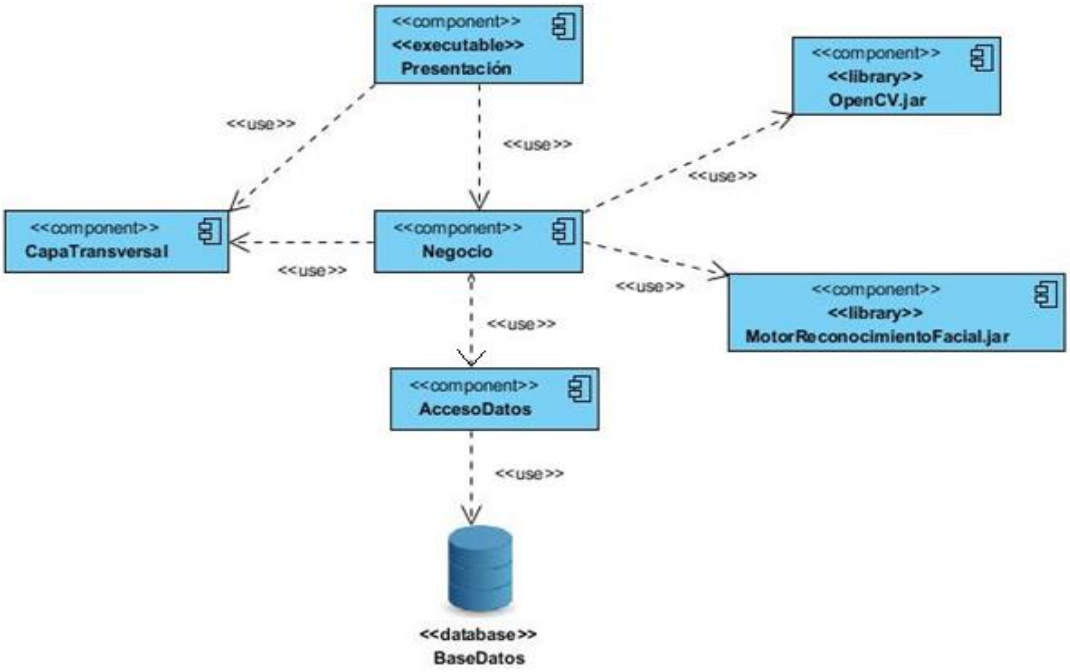

Figura 18: Diagrama de componentes del sistema de reconocimiento facial.

Componente	Propósito
Capa Transversal	Componente que contiene operaciones de autenticación, gestión de excepciones, registros (*logging*), trazas y validaciones de datos entrada.

Presentación	Componente que contiene la información y realiza la captura de los datos que el usuario introduce en la aplicación y además se comunica con el componente de negocio enviando peticiones al mismo.
Negocio	Componente que se comunica con el componente Presentación, para recibir las peticiones del usuario y presentar los resultados al mismo. Además se comunica con el componente AccesoDatos para el almacenamiento, recuperación y transformación de los datos.
AccesoDatos	Componente encargado de acceder y modificar los datos de la base de datos del sistema.
OpenCV	Componente encargado de la detección del rostro y procesamiento de imágenes.
MotorReconocimientoFacial	Componente en el que se encuentran las funcionalidades principales del software encargadas de realizar todo el proceso de identificación de la persona, el trabajo con la biblioteca de clases OpenCV y la extracción de características del rostro.

Tabla 13: Descripción de los componentes del Sistema de reconocimiento facial.

3.5 Diagrama de despliegue

El diagrama de despliegue muestra las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como una computadora, un dispositivo o memoria (56).

La siguiente figura muestra el diagrama de despliegue correspondiente a la propuesta de solución. Este está compuesto por una cámara digital mediante la cual será posible realizar la captura del rostro de la persona que será identificada en el sistema. En el caso de que la computadora donde se encuentre el sistema no posea una cámara integrada, la cámara se conectará a través del protocolo USB con la computadora que

tendrá instalado el sistema y mediante el protocolo TCP se realizará el intercambio de información con el servidor de base de datos.

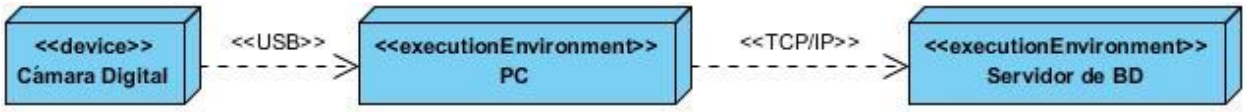

Figura 19: Diagrama de despliegue del Sistema de reconocimiento facial.

3.6 Pruebas

Luego de implementado el sistema se hace necesario realizar un conjunto de pruebas que permitan certificar la calidad de una aplicación informática, estas aumentan la calidad del sistema final, reducen los costos de avance y acortan el tiempo necesario para el mismo. Para determinar el nivel de calidad se deben realizar pruebas que permitan comprobar el grado de cumplimiento respecto a las especificaciones principales del sistema (60). A continuación se describen las pruebas realizadas.

3.6.1 Pruebas de funcionalidad

Las pruebas de funcionalidad se enfocan en las acciones por parte del usuario y las respuestas por parte del sistema. Se llevan a cabo para comprobar los requisitos y se considera que una funcionalidad tiene éxito cuando se comporta de la manera esperada por el cliente (60).

3.6.1.1 Pruebas de caja negra

El método de caja negra está basado en la ejecución, revisión y retroalimentación de las funcionalidades previamente diseñadas para el software. Se basa en el análisis de los datos de entrada y en los de salida. Verifican las especificaciones funcionales y no consideran la estructura interna del programa. El método de caja negra puede encontrar errores de las siguientes categorías: función incorrecta o ausente, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas (9). Se realizaron pruebas a diferentes funcionalidades del sistema una de ellas se muestra a continuación y las restantes se pueden observar en los anexos. (Ver anexos 4.3).

	Casos de Prueba de Funcionalidad
Código Caso de Prueba: 1	**Nombre Historia de Usuario:** Comparar características faciales

Nombre de la persona que realiza la prueba: Claudia Durán Rodríguez
Descripción de la prueba: Dada la captura de la imagen de un rostro y luego de la realización de los procesos de detección del rostro en la imagen y extracción de características, se realiza la comparación con la base de datos que permitirá la identificación de la persona.
Condiciones de ejecución: La imagen que se utilice debe poseer buenas condiciones de iluminación y la resolución debe ser mayor de 250x250, para obtener resultados satisfactorios. Para poder identificar una persona, esta debe haber pasado por el proceso de enrolamiento de forma previa.
Entrada/Pasos de ejecución: **1.** Imagen capturada por un dispositivo de video.
Resultado Esperado: Devuelve los datos identificativos de la persona, nombre, apellidos, sexo, carné de identidad y la diferencia entre los vectores, conjuntamente con la imagen por la cual fue identificada en la base de datos.
Evaluación: Satisfactoria

Tabla 14: Caso de prueba de la HU: Comparar características faciales.

3.6.1.2 Pruebas de caja blanca

El método de caja blanca comprueba los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos 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. Mediante la realización de las pruebas de caja blanca el ingeniero del software puede obtener casos de prueba que: (51)

1. Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
2. Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
3. Ejecuten todos los bucles en sus límites operacionales.
4. Ejerciten las estructuras internas de datos para asegurar su validez.

Por las razones expuestas anteriormente se considera que las pruebas de caja blanca son uno de los tipos de pruebas más importantes que se le aplican a cualquier software. La prueba del camino básico (62) es una técnica de prueba de caja blanca propuesta inicialmente por Tom McCabe. Esta técnica 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. Los casos de prueba derivados del conjunto básico garantizan que durante la prueba se ejecuta por lo menos una vez cada sentencia del programa. Para obtener dicho conjunto básico de caminos de ejecución se construye el grafo de flujo asociado y se calcula su complejidad ciclomática.

La complejidad ciclomática define el número de caminos independientes del conjunto básico de un programa y da un límite inferior para el número de pruebas que se deben realizar para asegurar que se ejecuta cada sentencia al menos una vez.

Un **camino independiente** es cualquier camino del programa que introduce por lo menos un nuevo conjunto de sentencias de procesamiento o una nueva condición. En términos del grafo de flujo, un camino independiente se debe mover por lo menos por una arista que no haya sido recorrida anteriormente en la definición de un camino.

La **complejidad ciclomática V (G)** se puede calcular de tres formas:

1. El número de regiones del grafo de flujo coincide con la complejidad ciclomática.
   V (G) = A - N + 2

Dónde: A es el número de aristas del grafo y N es el número de nodos.

2. La complejidad ciclomática V(G), también se define como:
   V (G) = P + 1

Dónde: P es el número de nodos predicados[8] contenido en el grafo G.

3. Números de regiones del grafo.
   V (G) = R

Dónde R es el número de regiones.

Realizar prueba de caja blanca a la funcionalidad Comparar Descriptores.

- ✓ **Funcionalidad: CompararDescriptores.**

```
public double CompararDescriptores(List<double[]> descrip1, List<double[]> descrip2)
{
```

[8] Un nodo predicado es el que representa una condicional if o case, es decir, que de él salen varios caminos.

```
        double valor = 0;//1
          int TotalY = 0; //1
          double ValorY = 0;//1
          for (int i = 0; i < 12; i++)//2
          {
             double[] Y = descrip1.get(0);//3
             double[] Yx = descrip2.get(0);//3
             double aux = Y[i] - Yx[i];//
             TotalY += (int)Math.pow(aux, 2);//3
          }
          ValorY = Math.sqrt(TotalY);//4
          int TotalCb = 0;//4
          double ValorCb = 0;//4
          for (int i = 0; i < 6; i++)//5
          {
             double[] Cb = descrip1.get(1);//6
             double[] Cbx = descrip2.get(1);//6
             double aux = Cb[i] - Cbx[i];//6
             TotalCb += (int)Math.pow(aux, 2);//6
}
          ValorCb = Math.sqrt(TotalCb);//7
int TotalCr = 0;//7
          double ValorCr = 0;//7
          for (int i = 0; i < 6; i++)//8
          {
             double[] Cr = descrip1.get(2);//9
             double[] Crx = descrip2.get(2);//9
             double aux = Cr[i] - Crx[i];//9
             TotalCr += (int)Math.pow(aux, 2);//9
}
          ValorCr = Math.sqrt(TotalCr);//10
          valor = ValorY + ValorCb + ValorCr;//10
         return Math.round(valor);//10
       }
```

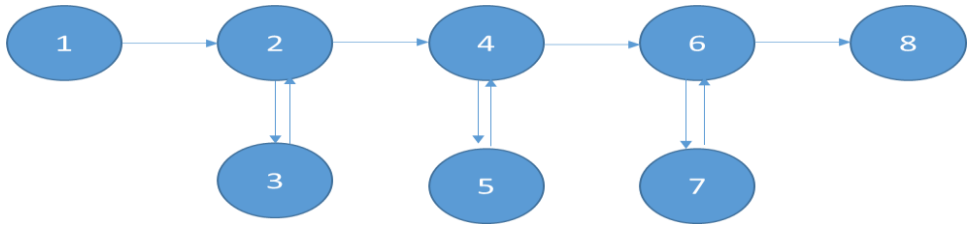

Figura 20: Grafo de flujo: Funcionalidad Comparar Descriptores.

- ✓ Complejidad ciclomática para la funcionalidad CompararDescriptores.:

- **Fórmula 1**

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = (10 – 8) + 2 = 4

- **Fórmula 2**

V (G) = P (Nodos Predicados) + 1

V (G) = 3 + 1 = 4

- **Fórmula 3**

V (G) = R = 4

Caminos independientes obtenidos:

- 1-2-3
- 1-2-4-5
- 1-2-4-6-7
- 1-2-4-6-8

Al aplicar las pruebas funcionales se realizaron tres iteraciones donde se detectaron varias no conformidades que en la mayoría de los casos estuvieron dadas por errores de interfaz, de validación y ortografía. En la primera iteración se encontraron un total de 16 no conformidades, las cuáles fueron analizadas y corregidas satisfactoriamente. En el transcurso de las pruebas se fueron mitigando las 8 no conformidades encontradas en la segunda iteración. En la tercera iteración no se obtuvo ninguna no conformidad para dar así por concluidas las pruebas funcionales. A continuación se muestra una gráfica con un resumen estadístico del número de no conformidades encontradas en cada iteración.

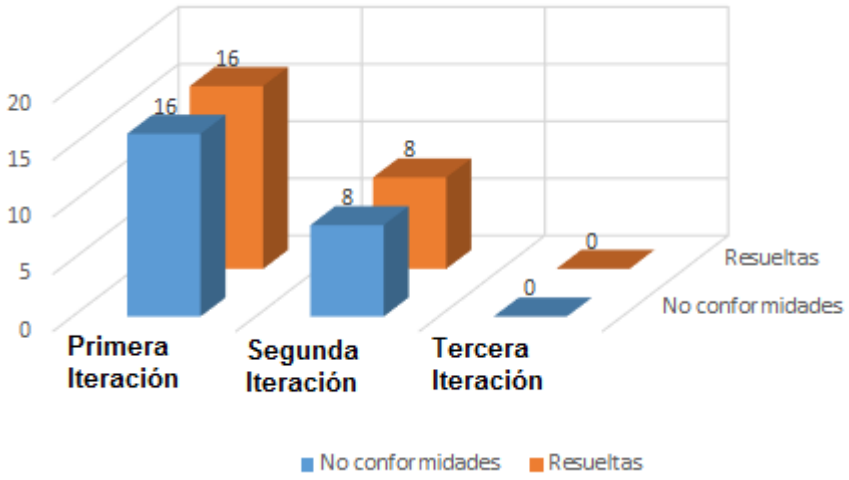

Figura 21: Resultado de las pruebas funcionales.

3.6.2 Pruebas de rendimiento

3.6.2.1 Evaluación del rendimiento del sistema

Para evaluar el rendimiento del sistema, se deben analizar y valorar los siguientes parámetros estándares:

- **FAR** (*False Acceptances Rate* o falsos positivos): Se produce cuando un sistema valida la identidad de una persona que no ha suministrado una muestra con anterioridad. En esta tasa se evidencia el porcentaje de veces que el sistema produce una falsa aceptación, es decir cuando un individuo es identificado de manera incorrecta (57).
- **FRR** (*False Reject Rate* o falsos negativos): Se produce cuando un sistema no valida la identidad de una persona que si ha suministrado anteriormente una muestra (57).
- **EER** (*Equal Error Rate* o tasa de error igual): El punto de intersección entre la FAR y el FRR se conoce como la tasa de error igual. Entre más bajo sea el valor de la tasa de error igual más alta será la precisión del sistema (57).

La FRR es una función estrictamente decreciente y la FAR una función estrictamente creciente. La FAR y la FRR al ser modeladas como función del umbral de aceptación tienen por dominio al intervalo real [0,1]. El umbral de aceptación se define en el punto donde la FRR y la FAR toman el mismo valor y puede ser utilizado como medida única para caracterizar el grado de seguridad del sistema. Usualmente se elige un

umbral de aceptación por debajo del punto donde se igualan estos dos factores con el objetivo de reducir la FAR y aumentar los FRR.

En la **figura 22** se puede observar la prueba realizada al sistema de reconocimiento facial para determinar el umbral correspondiente. Esta prueba se hizo con una base de datos que contenía un total de 100 imágenes faciales. Las imágenes fueron capturadas en entornos controlados como se explicó anteriormente.

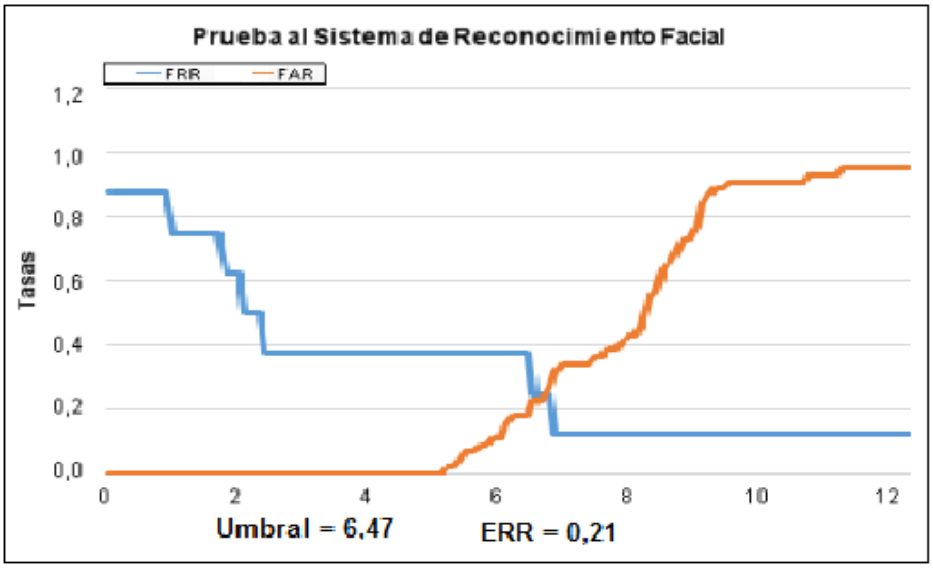

Figura 22: Determinación del umbral de aceptación.

Luego de analizar los resultados del experimento se concluye que para un umbral de 0,65 la efectividad es 2%. Comparando este resultado con los obtenidos de las pruebas realizadas al sistema de identificación de personas basado en rasgos faciales desarrollado en el año 2013 en el CISED el cual arrojó que para un umbral de 0,88 su efectividad era de 1% (9); se puede inferir que el sistema desarrollado en el CISED en el año 2013 posee mejor eficiencia con un umbral superior.

La tabla 17 muestra los resultados de las pruebas realizadas al sistema de identificación que indican el tiempo promedio en identificar una persona, este experimento se realizó en una computadora AMD C-50, a 1.0GHz, 3GB de Memoria RAM DDR3, con sistema operativo Ubuntu 14.04.

Prueba 1

Total Imágenes en la base de datos	Tiempo en Segundo
1	0,202
10	0,405
100	0,667
1000	0,779
Prueba 2	
Total Imágenes en la base de datos	**Tiempo en Segundo**
1	0,357
10	0,477
100	0,695
1000	0,768

Tabla 15: Tiempos de extracción de características conjuntamente con la comparación con la BD.

En la **figura 23** se muestran los resultados de las pruebas realizadas al sistema de identificación.

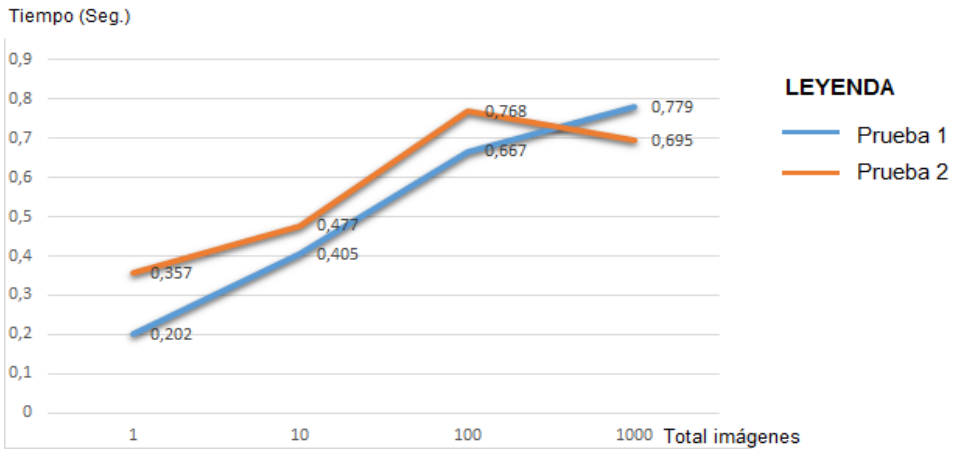

Figura 23: Representación de los tiempos de identificación de una persona.

Luego de analizar los resultados del experimento se concluye que el tiempo promedio de aceptación o rechazo de una persona bajo las condiciones antes mencionadas es aproximadamente 0,54 segundos.

3.7 Conclusiones parciales

En este capítulo se definió el estilo de codificación utilizado en la implementación del sistema, lo que permitió mantener una uniformidad en la codificación y mayor claridad a la hora de leer o agregar líneas de código. Se realizaron los diagramas de componentes y de despliegue, los cuáles posibilitaron un mejor

entendimiento del software desarrollado, ayudando así a su implementación. Las pruebas de rendimiento permitieron comprobar el buen funcionamiento del algoritmo de reconocimiento facial obteniendo una tasa de error igual de 2% con un umbral óptimo de aceptación de 0.65. El diseño de pruebas de caja blanca y negra contribuyó a establecer los parámetros para maximizar la efectividad del sistema.

Conclusiones generales

- El estudio de soluciones informáticas existentes permitió identificar elementos significativos que sirvieron de base para el desarrollo de una propuesta de solución al problema planteado.
- Como resultado del proceso de diseño se lograron definir las tecnologías y herramientas de software libre, así como la metodología a utilizar, lo que permitió crear una solución multiplataforma y con el menor costo de construcción posible.
- Se logró el diseño y la implementación de un sistema de identificación de personas basado en características faciales desarrollado sobre tecnologías libres como parte del proceso de migración hacia aplicaciones y plataformas de software libre que se lleva a cabo en la universidad.
- Para validar el sistema se realizó una estrategia de pruebas de rendimiento, efectividad y funcionalidad. Las pruebas de efectividad permitieron comprobar el buen funcionamiento del algoritmo de reconocimiento facial obteniendo una tasa de error igual de un 2% con un umbral óptimo de aceptación de 0,65. Además se calculó el tiempo promedio de identificación de una persona siendo este aproximadamente de 0,54 segundos. El diseño de pruebas de caja blanca y negra permitieron verificar el correcto funcionamiento del sistema de identificación.

Recomendaciones

Para mejorar el sistema y seguir profundizando en el campo del reconocimiento facial, se recomienda:

- Integrar el sistema al componente para la clasificación de rasgos biométricos faciales utilizando características geométricas; desarrollado en el CISED para optimizar el proceso de comparación en la base de datos.

Bibliografía referenciada

1. Rearte, Manuela y Martin Sanchez, Gonzalo. *Introducción a la Inteligencia Artificial.* 2011.

2. IBIX. [En línea] 2013. [Citado el: 30 de mayo de 2015.] http://www.ibix.com/reconocimiento-facial.

3. Kirby, M. y Sirovich, L. *A Low-Dimensional Procedure for the Chatecterization of Human Faces.* 1987.

4. Johnson, Ryan y Bonsor, Kevin. How Facial Recognition Systems Work. *howstuffworks.* [En línea] http://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm.

5. Interfaces-SB-2011. [En línea] https://sites.google.com/site/interfacessb2011/sistema-biometrico-de-cara.

6. Stallman, Richard M. *Software libre para una sociedad libre.* s.l. : Free Software Foundation, 1995-1999.

7. Malcolm, Bain. *Aspectos legales y de explotación del software libre.* 2000. Vol. Vol 1.

8. Rouse, Margaret. SearchDataCenter. [En línea] octubre de 2008. [Citado el: 30 de mayo de 2015.] http://searchdatacenter.techtarget.com/es/definicion/Biometria.

9. Quiles Velázquez, Rafael Alberto y Aguilera Reyes, Amelia. *Sistema de identificación de personas basado en rasgos faciales.* La Habana : s.n., 2013.

10. UNAM-Facultad de Ingeniería Biométrica Informática. [En línea] http://redyseguridad.fi-p.unam.mx/proyectos/biometria/basesteoricas/caracteristicassistema.html.

11. Weissman, Kimberly. *Face Recognition.* 2010.

12. Berchesi, Sebastián, Di Martino, Luis y Lema, Gabriel. Instituto de Ingeniería Electrica. *Proyecto Faceval.* [En línea] noviembre de 2012. [Citado el: 30 de mayo de 2015.] http://iie.fing.edu.uy/publicaciones/2012/BDL12/Faceval.pdf.

13. Vovk. *Proceedings of the Eighth Annual Conference on Computational Learning Theory.*

14. Viola, Paul y Jones, Michel. *Rapid object detection using a boosted cascade of simple features.* 2001.

15. Peláez Fernández, Borja. Reconocimiento de caras en entornos no controlados. [En línea] 20 de junio de 2012. [Citado el: 30 de mayo de 2015.] http://upcommons.upc.edu/pfc/bitstream/2099.1/15642/4/Reconocimiento_de_caras_en_entornos_no_controlados,_Borja_Pelaez_Fernandez.pdf.

16. López Pérez, Nicolás y Toro Agudelo, Juán José. *Técnicas de biometría basadas en patrones faciales del ser humano.* 2012.

17. Aguerrebere, C. *Proyecto Aguará: Reconocimiento Automático de Caras.*

18. Biometría. *Reconocimiento facial.* [En línea] 24 de mayo de 2015. http://www.biometria.gov.ar/metodos-biometricos/facial.aspx.

19. Aguerrebere Otegui, Celia. Proyecto Aguará. *Proyecto de grado Ingeniería Eléctrica.* [En línea] http://iie.fing.edu.uy/investigacion/grupos/biometria/proyectos/aguara/descargas/documenta_aguara_v1.0.pdf.

20. Delbracio, Mauricio y Mateu, Matias. *Trabajo Final de Reconocimiento de Patrones: Identificación utilizando PCA,ICA y LDA.* 2006.

21. Breno Santos Araujo, Alexi Manso Correa. *Towards a better comprehension of the role of image registration in face recognition altorithms.* Brasil : s.n., 2011.

22. Martin Marcos, Alfonso L. *Transformada discreta del coseno DCT.*

23. MD5. [En línea] 5 de junio de 2012. [Citado el: 30 de mayo de 2015.] http://es.slideshare.net/crissss13/equipo-1-13208407.

24. Xie, Taxo, Fanbao, Liu y Dengguo, Feng. *Fast Collision Attack on MD5.* 2013.

25. Aguirre, Jorge Ramió. *Libro electrónico de seguridad informática y criptografía. Versión 4.1.* Madrid : Universidad Politécnica de Madrid, marzo 2006.

26. Monografías. *Funciones Hash.* [En línea] 2008. http://www.monografias.com/trabajos76/funciones-hash-criptografia/funciones-hash-criptografia2.shtml.

27. García Boullosa, Óscar. *Estudio comparativo de descriptores visuales para la detección de escenas casi duplicadas.* Madrid : s.n., 2011.

28. Recuperación de información multimodal. *Descriptores de imagen.* [En línea] 21 de mayo de 2011. [Citado el: 30 de mayo de 2015.] http://recuperaciondeinformacionmultimodal.blogspot.com/2011/05/descriptores-de-imagenes.html.

29. Picasa. [En línea] [Citado el: 30 de mayo de 2015.] https://picasa.google.com/.

30. Sistema., Datys: Tecnologías y. *BiomesysFace.*

31. CNN. *El FBI lanza un poderoso sistema de reconocimiento facial.* [En línea] http://cnnespanol.cnn.com/2014/09/17/el-fbi-lanza-un-poderoso-sistema-de-reconocimiento-facial/.

32. *DeepFace: Closing the Gap to Human-Level Performance in Face Verification.* [En línea] 24 de junio de 2014. https://research.facebook.com/publications/480567225376225/deepface-closing-the-gap-to-human-level-performance-in-face-verification/.

33. Canós, José H., Letelier, Patricio y Penadés, María Carmen. *Metodologías ágiles en el desarrollo de software.*

34. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). [En línea] 15 de enero de 2006. [Citado el: 30 de mayo de 2015.] http://www.cyta.com.ar/ta0502/v5n2a1.htm. ISSN 1666-1680.

35. Proyectos Ágiles.org. *Qué es SCRUM.* [En línea] [Citado el: 30 de mayo de 2015.] http://www.proyectosagiles.org/que-es-scrum.

36. Schmuller, Joseph. *UML en 24 Horas.*

37. Mastermagazine. *Definición de UML.* [En línea] 2008. [Citado el: 30 de mayo de 2015.] http://www.mastermagazine.info/termino/7006.php..

38. Perissé, C.M. *Herramientas case.* [En línea] 2010. [Citado el: 30 de mayo de 2015.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro/c5/c5.htm.

39. Marañón, G.Á. *Características del lenguaje Java.* [En línea] 2006. http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html..

40. Escribano, G F. *Extreme Programming.* 2002.

41. Fergarciac. *Entorno de desarrollo integrado (IDE).* [En línea] 25 de enero de 2013. [Citado el: 30 de mayo de 2015.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

42. Téllez Escalona, Addiel Alejandro. *Sistema para la gestión de las imágenes que se utilizan en la publicación de noticias en el departamento de Comunicación Audiovisual de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2013.

43. Luján Mora, Sergio. *C++ Paso a Paso.* Alicante : s.n., 2006. 84-7908-888-5.

44. PostgreSQL. *Características, limitaciones y ventajas.* [En línea] noviembre de 2012. [Citado el: 30 de mayo de 2015.] http://postgresql-dbms.blogspot.com/p/limitaciones-puntos-de-recuperacion.html.

45. TuProgramación.com. *¿Qué es un ORM?* [En línea] [Citado el: 30 de mayo de 2015.] http://www.tuprogramacion.com/glosario/que-es-un-orm/.

46. OpenCV. [En línea] 21 de 08 de 2014. [Citado el: 30 de mayo de 2015.] http://opencv.org/opencv-3-0-alpha.html.

47. Chaczko, Yeoh. *A preliminary investigation on computer vision for telemedicine systems using OpenCV,IEEE.* 2010.

48. G. Bradski, A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library.* 2008.

49. Webcam Capture. *Webcam Capture API.* [En línea] [Citado el: 30 de mayo de 2015.] http://webcam-capture.sarxos.pl/.

50. "Tecnología y synergix. Modelo de dominio.". [En línea] 5 de julio de 2008. [Citado el: 30 de mayo de 2015.] http://synergix.wordpress.com/?s=modelo+dominio..

51. Sommerville, Ian. *Ingeniería del Software. Séptima Edición.* 2005.

52. Joskowicz, José. Reglas y prácticas en eXtreme Programming. [En línea] 10 de febrero de 2008. [Citado el: 31 de mayo de 2015.] http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf.

53. Cortes, Carlos, y otros. Metodologías ágiles: Metodología XP. [En línea] 6 de marzo de 2013. [Citado el: 31 de mayo de 2015.] http://es.slideshare.net/LisPater1/metodologias-agiles-xp.

54. Reynoso, Carlos Billy. *Introducción a la Arquitectura de Software.* 2004.

55. Entorno virtual de aprendizaje. *Arquitectura orientada a servicios.* [En línea] [Citado el: 31 de mayo de 2015.] http://eva.uci.cu.

56. Pressman, Roger. *Ingeniería del Software un enfoque práctico. Estrategia de pruebas. Quinta Edición.* 2007. ISBN: 97-0105-473-3.

57. Lago, Ramiro. *Patrones de diseño software.* 2007.

58. Taringa. *Patrones de diseño GRASP.* [En línea] 9 de diciembre de 2014. [Citado el: 31 de mayo de 2015.] http://www.taringa.net/posts/apuntes-y-monografias/18339620/Patrones-de-Diseno-GRASP.html.

59. Mundo informático. *Patrones de diseño GoF.* [En línea] 2013. [Citado el: 31 de mayo de 2015.] https://infow.wordpress.com/category/patrones-de-disenogof/.

60. Anias Santos, Javier y Cabeza Matos, Kirenia. *Solución para la gestión de información de los procesos de Extensión Universitaria en el área de Extensión Cultural de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2013.

61. Sarmiento, Johana. UML: Diagrama de Despliegue. *Visión general de los diagramas de despliegue.* [En línea] 12 de abril de 2013. [Citado el: 30 de mayo de 2015.] http://umldiagramadespliegue.blogspot.com/2013/04/vision-general-de-los-diagramas-de.html.

62. EcuRed. *Pruebas de caja blanca.* [En línea] [Citado el: 30 de mayo de 2015.] http://www.ecured.cu/index.php/Pruebas_de_caja_blanca.

63. Ciudad digital. [En línea] 3 de julio de 2010. [Citado el: 30 de mayo de 2015.] http://ciudadaniasdigitales.blogspot.com/2010/07/glosario-digital-parametros-estandares.html.

Bibliografía consultada

1) Alba, José Luis, Cid, Jesús and Mora, Inmaculada. Extracción de características. 2006.
2) Hernández, Roger. Estudio de técnicas de reconocimiento facial. Barcelona: s.n., 2010.
3) Beck, Kent. Extreme Programming Explained.2004.
4) Méndez Vázquez, Heydi. Algoritmo de reconocimiento de rostros basados en la apariencia local para aplicaciones reales en condiciones variables de iluminación. Ciudad de la Habana .2012.
5) González, C and Woods, R. Digital Image Processing. New Jersey. EEUU Prentice Hall, 2007.
6) Niu, Z. Enhance ASMs Based on AdaBoost-Based Salient Landmarks Localization and Confidence-Constraint Shape Modeling. Berlin: 2005.
7) R.S. Pressman. Software Engineering a Practitioner's Approach. McGraw Hill Publication, 2010.
8) N. Degtyarev and O. Seredin. Comparative testing of face detection algorithms. Image and Signal
9) Processing, 2010.
10) R. Gimeno Hernández. Estudio de técnicas de reconocimiento facial. 2011.

Glosario de términos

Escala de grises: Modelo de representación de una imagen digital mediante 256 tonalidades de gris.

Ecualización del histograma: Técnica que consiste en ajustar los niveles de gris de una imagen para obtener una nueva imagen con un histograma uniforme.

Framework: Estructura de artefactos o módulos concretos con base en la que otro proyecto de software puede ser desarrollado.

Normalización: Acción de transformar una distribución cualquiera a una distribución normal manteniendo la proporción de los datos.

Segmentación: Proceso mediante el cual se divide una imagen en múltiples partes para lograr una mejor representación y obtener así información relevante.

Umbral de aceptación: Valor mínimo a partir del cual se considera que un elemento pertenece a una clase.

Acrónimos

C	**CISED:** Centro de Identificación y Seguridad Digital. **CENATAV:** Centro de Aplicaciones de Tecnologías de Avanzadas. **CASE:** Ingeniería de Software Asistida por Ordenador. **CRC:** Tarjeta Clase-Responsabilidad-Colaboración.
D	**DATYS:** Empresa de Desarrollo de Aplicaciones, Tecnologías y Sistemas. **DCT:** Transformada Discreta del Coseno. **DDC:** Descriptor de la Distribución del Color.
E	**EER:** Taza de Error Igual. **EP:** Evolutionary Pursuit.
F	**FDD:** Feature Driven Development. **FAR:** Falso Positivo.Ocurre cuando un elemento es enmarcado dentro de una clase conocida y este no pertenece a dicha clase. **FRR:** Falso Negativo. Cuando el sistema devuelve que un elemento es desconocido y sin embargo pertenece una clase conocida.
G	**GRASP:** Patrones de asignación de responsabilidades. **GOF:** Banda de los Cuatro.
H	**HU:** Historias de usuario.
I	**ICA:** Análisis de Componentes Independientes. **IDE:** Entorno de Desarrollo Integrado, también conocido como entorno de diseño integrado o entorno de depuración integrada.
L	**LDA:** Análisis de Discriminante Leal.
M	**MLL:** Machine Learning Library. **MVC:** Modelo Vista Controlador.
O	**OpenCV:** Visión por Computadora de Código Abierto. **ORM:** Mapeo Objeto-Relacional.

P	**PCA:** Análisis de Componentes Principales. **PDM:** Modelos de Puntos Distribuidos. **PIN:** Número de Identificación Personal, es una contraseña o clave numérica que se utiliza para acceder a cajeros automáticos o servicios de telefonía. **POO:** Programación Orientada a Objetos.
S	**SGBD:** Sistema de Gestor de Base de Datos. **SOA:** Arquitectura Orientada a Servicios.
U	**UCI:** Universidad de las Ciencias Informáticas. **UML:** Lenguaje Unificado de Modelado.
X	**XP:** Programación Extrema.

Anexos

4.1 Descripción de las historias de usuarios del sistema de identificación de personas basado en características facial.

Historia de Usuario	
Número: 2	**Usuario:** Técnico, Administrador
Nombre historia: Capturar imagen con dispositivo de video.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: David Fonseca Martínez	

Tabla 16: Descripción de la HU: Capturar imagen con dispositivo de video.

Historia de Usuario	
Número: 3	**Usuario:** Administrador
Nombre historia: Cargar imagen	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: David Fonseca Martínez	
Descripción: El administrador cargará la imagen de una dirección física de la computadora.	
Observaciones:	

Tabla 17: Descripción de la HU: Cargar imagen.

Historia de Usuario	
Número: 4	**Usuario:** Administrador
Nombre historia: Guardar imagen.	
Prioridad en el negocio: Baja	**Riesgo en desarrollo:** Baja

Puntos estimados: 1	**Iteración asignada:** 1
Programador responsable: David Fonseca Martínez	
Descripción: El administrador guardara la imagen procesada.	
Observaciones:	

Tabla 18: Descripción de la HU: Detectar rostro en imagen.

Historia de Usuario	
Número: 5	**Usuario:** Técnico, Administrador.
Nombre historia: Procesar imagen	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 1
Programador responsable: David Fonseca Martínez	
Descripción: 1) Se debe detectar un rostro en la imagen digital. 2) Se extraerán las características faciales presentes en la imagen procesada, para obtener el vector característico asociado. 3) Se compara el vector obtenido con los que se encuentran almacenados en la base de datos y se procede a la identificación de la persona.	
Observaciones: En caso de no detectarse ningún rostro en la imagen no se podrá realizar el proceso de identificación.	

Tabla 19: Descripción de la HU: Procesar imagen.

Historia de Usuario	
Número: 6	**Usuario:** Administrador.
Nombre historia: Gestionar usuario	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media

Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: David Fonseca Martínez	
Descripción: El administrador tendrá la posibilidad de gestionar los usuarios que operarán en el sistema, lo cual incluyen el rol y los datos personales del mismo.	
Observaciones: Gestionar usuario incluye crear, modificar, eliminar y listar usuarios.	

Tabla 20: Descripción de la HU: Gestionar usuario.

Historia de Usuario	
Número: 7	**Usuario:** Administrador.
Nombre historia: Gestionar persona	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 2	**Iteración asignada:** 2
Programador responsable: David Fonseca Martínez	
Descripción: El administrador tendrá la posibilidad de gestionar las personas que serán identificadas, lo cual incluyen los datos personales del mismo.	
Observaciones: Gestionar persona incluye crear, modificar, eliminar y listar personas.	

Tabla 21: Descripción de la HU: Gestionar persona.

4.2 Descripción de las tarjetas CRC pertenecientes a las clases del negocio del sistema de identificación de persona basado en características faciales.

Nombre de la Clase :Persona	
Responsabilidades	**Colaboradores**
1. getIdPersona **2.** getNombre **3.** getPapellido **4.** gerSapellido **5.** getSexo	**1.** Usuario **2.** VectorCaracterisitco **3.** Log

6. getCi **7.** getVectorcaracteristicos **8.** getVectorcaracteristicos_1 **9.** getAttribute **10.** getUsuarios_1 **11.** getLogs **12.** getLogs_1	

Tabla 22: Tarjeta CRC correspondiente a la clase Persona.

Nombre de la Clase :Rol	
Responsabilidades	**Colaboradores**
1. getIdrol **2.** getNombre **3.** getDescripcion **4.** getUsuarios **5.** getUsuarios_1	**1.** Usuario

Tabla 23: Tarjeta CRC correspondiente a la clase Rol.

Nombre de la Clase :Log	
Responsabilidades	**Colaboradores**
1. getIdlog **2.** getPersona **3.** getHoraentrada **4.** getHorasalida **5.** getHostname **6.** getIpaddress **7.** getMac	**1.** Persona

Tabla 24: Tarjeta CRC correspondiente a la clase Log.

Nombre de la Clase :SistemaControler	
Responsabilidades	**Colaboradores**
1. getImagen_java	**1.** Umbral

2. LeerVectores **3.** eliminarVector **4.** LeerUmbral **5.** ObtenerInstancia **6.** ExtraerVectorcaracteristicoDDC2 **7.** ExtraerVectorcaracteristicoDDC **8.** LeerImagen **9.** DetectarRostros **10.** PintarRostros **11.** GuardarImagen **12.** EliminarImagent **13.** getMotor **14.** getPath **15.** getCurrent **16.** getPath_imagen **17.** getImagen **18.** GenerarBloques **19.** Convertir_a_Imagen **20.** Convertir_a_Mat **21.** Autenticar **22.** CambiarPassword **23.** InsertarPersona **24.** InsertarUsuario **25.** ListarPersona **26.** ActualizarPersona **27.** InsectarVector **28.** ActualizarUsuario **29.** EliminarPersona **30.** EliminarUsuario **31.** ListarUsuario **32.** AjustarUmbral **33.** getVectores **34.** getUmbral	**2.** VectorCaracteristico

35. ComparарVectores **36.** ordenarDiferencias	

Tabla 25: Tarjeta CRC correspondiente a la clase SistemaControler.

Nombre de la Clase :Umbral	
Responsabilidades	**Colaboradores**
1. getIdumbral **2.** getValor **3.** getDescripcion	**1.** Persona

Tabla 26: Tarjeta CRC correspondiente a la clase Umbral.

Nombre de la Clase :Usuario	
Responsabilidades	**Colaboradores**
1. getIdusuario **2.** getPersona **3.** getRol **4.** getUsuario **5.** getPasssword	**1.** Rol **2.** Persona

Tabla 27: Tarjeta CRC correspondiente a la clase Usuario.

4.3 Descripción de los casos de prueba de las funcionalidades

Casos de Prueba de Funcionalidad	
Código Caso de Prueba: 2	**Nombre Historia de Usuario:** Detectar rostro en imagen.
Nombre de la persona que realiza la prueba: Claudia Durán Rodríguez	
Descripción de la prueba: Dada la captura de la imagen de un rostro se realizará el proceso de detección del rostro que permitirá detectar la presencia del rostro dentro de una imagen digital.	
Condiciones de ejecución: Las imágenes que se utilicen deben poseer buenas condiciones de iluminación y la resolución debe ser mayor de 250x250, para obtener resultados satisfactorios en el proceso de identificación	
Entrada/Pasos de ejecución:	

1. Imagen capturada por un dispositivo de video.
Resultado Esperado: Rostro detectado.
Evaluación: Satisfactoria

Tabla 28: Caso de prueba de la funcionalidad Detectar rostro en imagen.

Casos de Prueba de Funcionalidad	
Código Caso de Prueba: 4	**Nombre Historia de Usuario:** Extraer características faciales.
Nombre de la persona que realiza la prueba: Claudia Durán Rodríguez	
Descripción de la prueba: Luego de la captura de la imagen de un rostro y realizado el proceso de detección del rostro en la imagen se procede a extraer las características faciales para así obtener el vector característico.	
Condiciones de ejecución: La imagen que se utilice debe poseer buenas condiciones de iluminación y la resolución debe ser mayor de 250x250, para obtener resultados satisfactorios.	
Entrada/Pasos de ejecución: **1.** Capturar imagen con un dispositivo de video. **2.** Detectar rostro dentro de la imagen capturada.	
Resultado Esperado: Vector característico.	
Evaluación: Satisfactoria	

Tabla 29: Caso de prueba de la funcionalidad Extraer características faciales.

4.4 Pruebas de caja blanca

Funcionalidad: ExtraerVectorCaracteristicoDDC.

```
public EDVector ExtraerVectorCaracteristicoDDC (Mat img)
   {
      try //1
{
      List<Rect> pos_rostro=DetectarRostros(img); //2
```

```
        Rect posiscion=pos_rostro.get(0); //2
       Mat img_rostro = motor.getDDC().RecortarImagenRostro(img, posiscion.x, posiscion.y, posiscion.width,
posiscion.height); //2
       Mat[][] bloques = motor.getDDC().GenerarBloquesPequños(img_rostro); //2
          Mat icon=motor.getDDC().CrearImagenIcono(bloques); //2
             List<double[][]> vectortransformado= motor.getDDC().TransformadaDiscretaCoseno(icon); //2
      double [][] matrizR=vectortransformado.get(0); //2
      double [][] matrizG=vectortransformado.get(1); //2
      double [][] matrizB=vectortransformado.get(2); //2
      double [] arrayR=motor.getDDC().ConformarVectorCaracteristico(matrizR); //2
      double [] arrayG=motor.getDDC().ConformarVectorCaracteristico(matrizG); //2
double [] arrayB=motor.getDDC().ConformarVectorCaracteristico(matrizB); //2
       List<double[]> aux=new ArrayList<>(); //2
       aux.add(arrayR); //2
       aux.add(arrayG); //2
       aux.add(arrayB); //2
       EDVector vector=new EDVector(posiscion, aux); //3
       return vector; //3
        }
        catch (Exception e) //4
        {
            return null; //5
}
   }
```

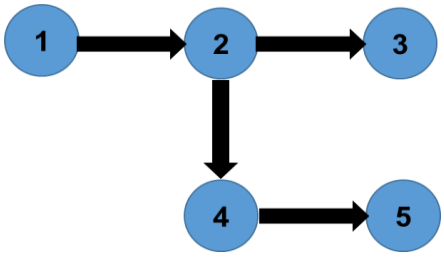

Figura 24: Grafo de flujo: Funcionalidad ExtraerVectorCaracterísticoDDC.

- ✓ Complejidad ciclomática para la funcionalidad ExtraerVectorCaracteristicoDDC.:

- **Fórmula 1**

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = (4 – 5) + 2 = 1

- **Fórmula 2**

V (G) = P (Nodos Predicados) + 1

V (G) = 1 + 1 = 2

- **Fórmula 3**

V (G) = R = 2

Caminos independientes obtenidos:

- ✓ 1-2-3
- ✓ 1-4-5

Funcionalidad:DetectarRostro.

```
public List<Rect> DetectarRostro(Mat img)
{
    if(!img.empty()) //1
     {
MatOfRect faceDetections = new MatOfRect();//2
      cascade.detectMultiScale(img, faceDetections); //2
      imagen=img; //2
      return faceDetections.toList();//3
    }
```

```
    Else //4
    {
        //throw new Exception("Errror en imagen"); //5
return null; //6
    }
  }
```

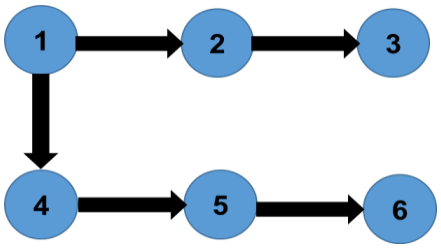

Figura 25: Grafo de flujo: Funcionalidad Detectar Rostro.

- ✓ Complejidad ciclomática para la funcionalidad DetectarRostro.:

- **Fórmula 1**

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = (5 – 6) + 2 = 1

- **Fórmula 2**

V (G) = P (Nodos Predicados) + 1

V (G) = 1 + 1 = 2

- **Fórmula 3**

V (G) = R = 2

Caminos independientes obtenidos:

- ✓ 1-2-3
- ✓ 1-4-5-6

4.5 Interfaces de usuario

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, lo cual su diseño debe ser amigable y consistente por lo que suelen ser fáciles de

entender y fáciles de accionar. Una aplicación con una interfaz bien diseñada, además de un buen diseño gráfico, debe tener una buena navegabilidad, usabilidad y distribución de los contenidos.

A continuación se muestran las interfaces para que se tenga un mayor entendimiento de lo que realiza el sistema.

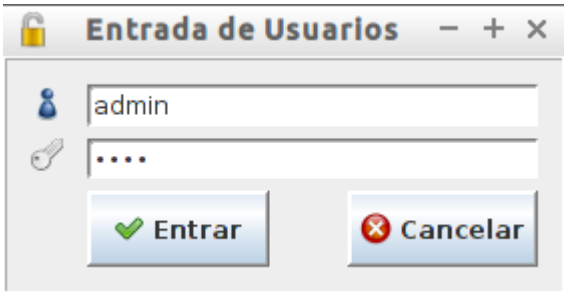

Figura 26: Interfaz Autenticar Usuario.

Figura 27: Interfaz de Administración.

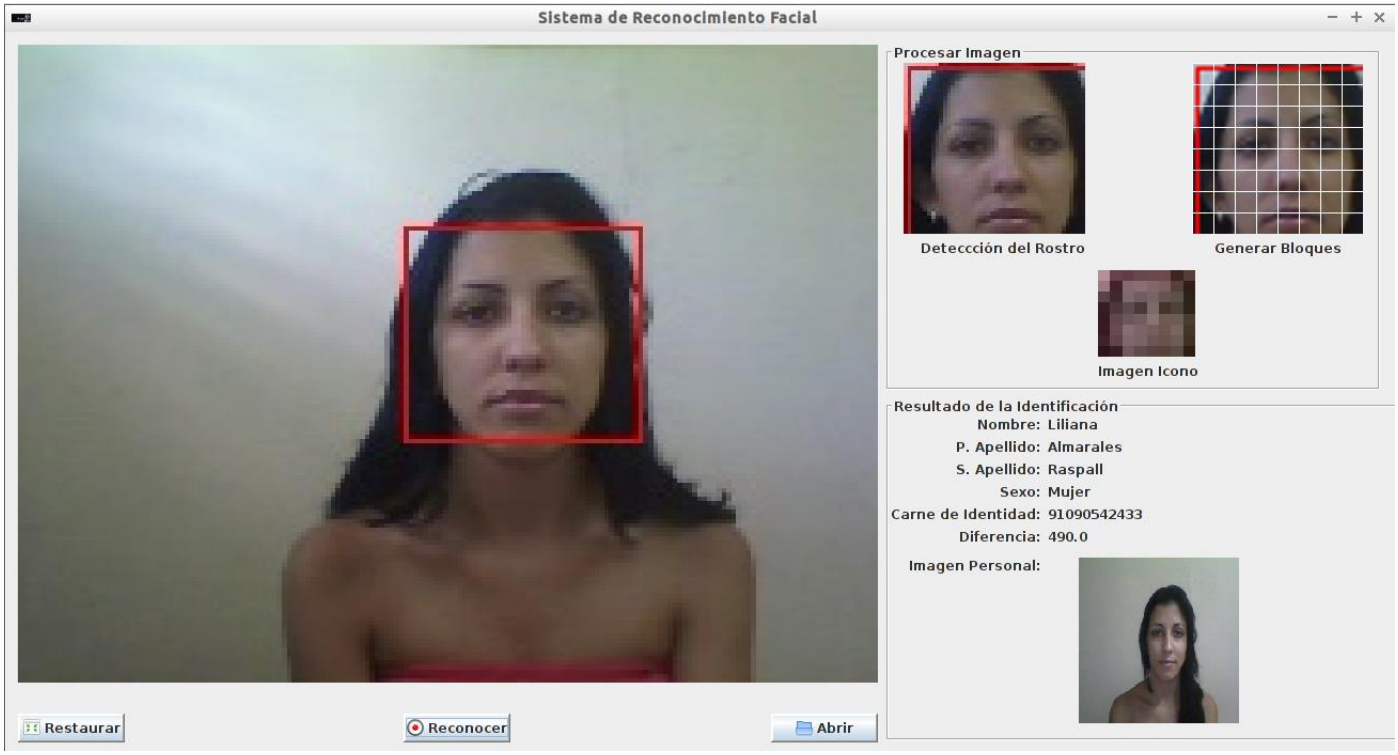

Figura 28: Interfaz para procesar el rostro.

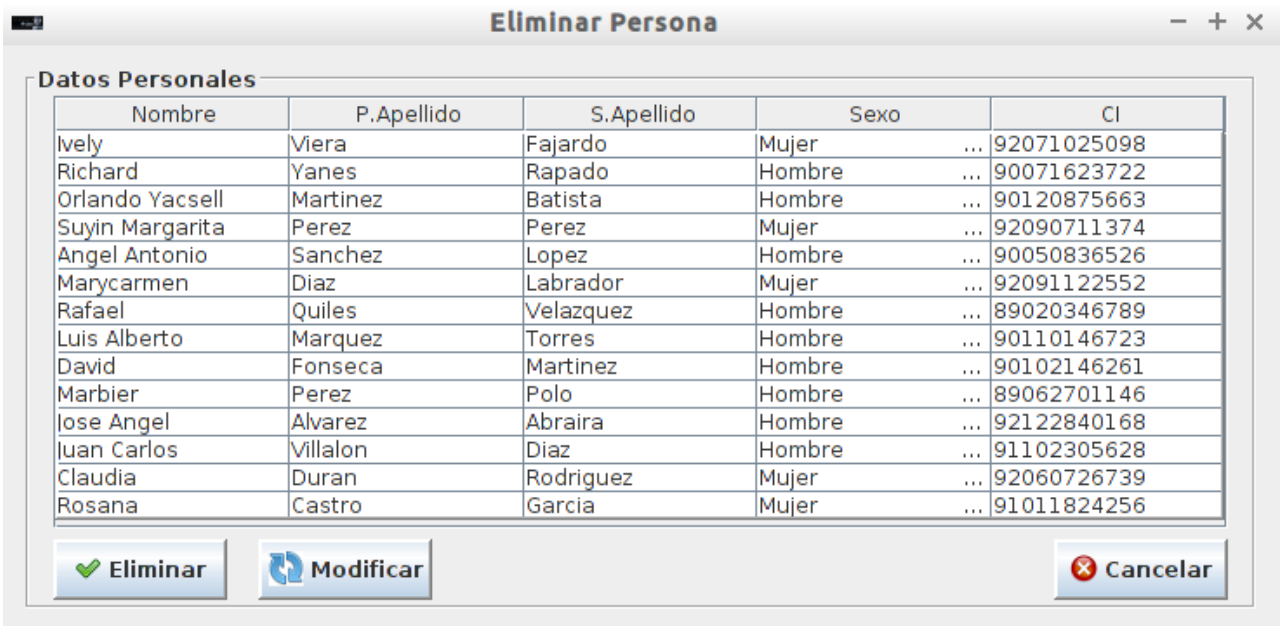

Figura 29: Formulario para eliminar una persona.

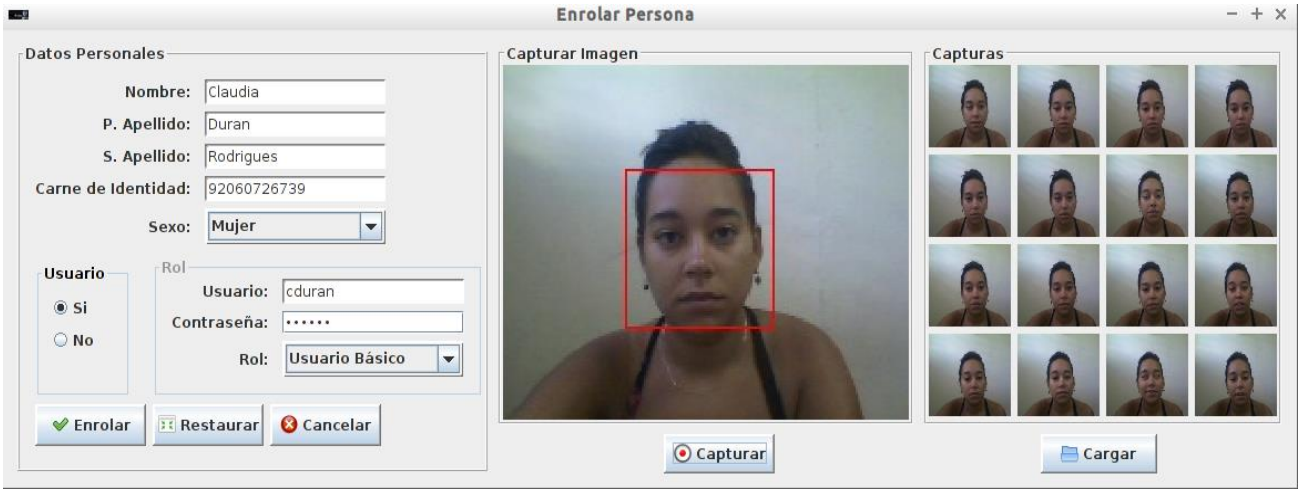

Figura 30: Formulario para enrolar una persona.

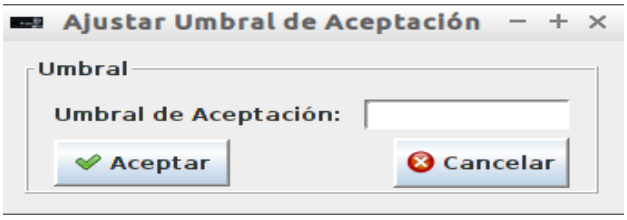

Figura 31: Formulario para ajustar el umbral de aceptación.

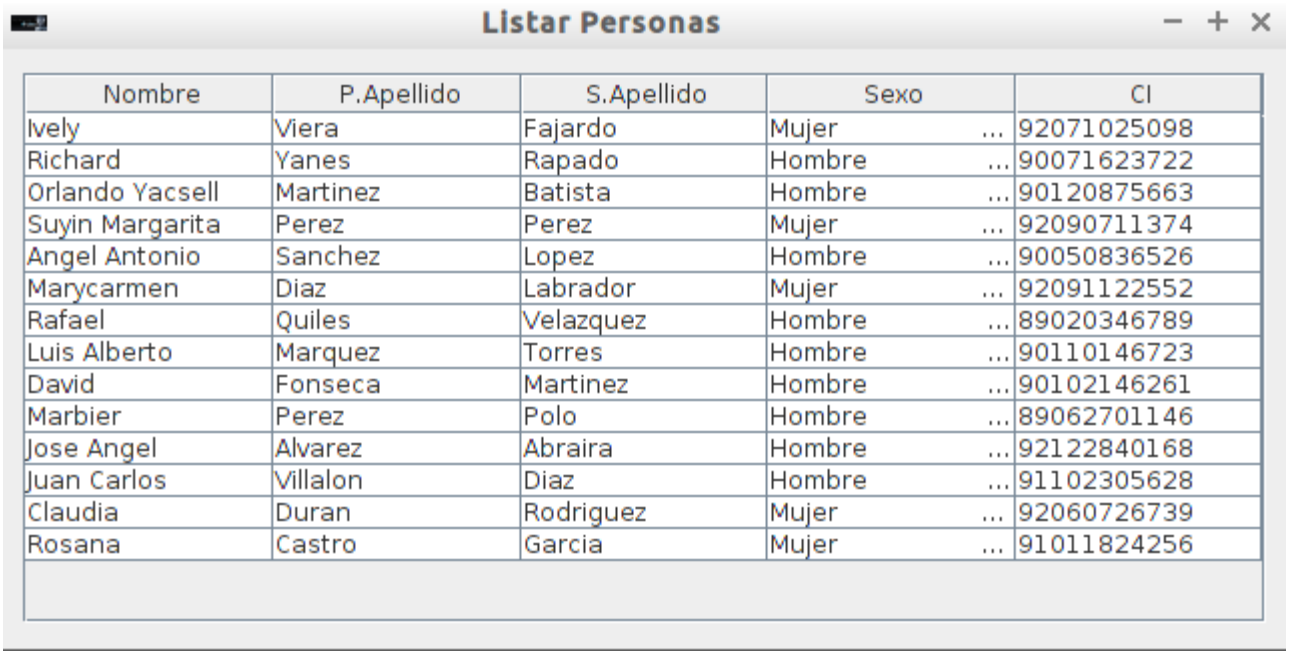

Figura 32: Formulario para listar las personas enroladas en la base de datos.