Universidad de las Ciencias Informáticas

Facultad 1

Título: Aplicación web para la validación y actualización de la imagen facial del ciudadano de la Universidad de las Ciencias Informáticas.

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

Autores: Alisbet Fernández Rojas
Daniel Vega Torres

Tutor: Lic. Yadier Perdomo Cuevas
Ing. Royli Hernández Delgado

La Habana, 2014

Declaración de autoría

Declaramos ser los autores del trabajo titulado “Aplicación web para la validación y actualización de la imagen facial del ciudadano de la Universidad de las Ciencias Informáticas” para el Centro de Identificación y Seguridad Digital y se le autoriza a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos el presente a los ____ días del mes de ____ del año 2014.

_____ _____

Alisbet Fernández Rojas Daniel Vega Torres

_____ _____

Lic. Yadier Perdomo Cuevas Ing. Royli Hernández Delgado

Datos del contacto

Lic. Yadier Perdomo Cuevas

Graduado de Ciencias en la computación en el año 2004, con categoría docente de Asistente, posee 10 años de experiencia laboral y nueve trabajando en el tema del Trabajo de Diploma. Actualmente se desempeña como subdirector de tecnología del Centro de Identificación y Seguridad Digital.

Correo electrónico: *ypc@uci.cu*

Ing. Royli Hernández Delgado

Ingeniero en Ciencias Informáticas. Recién graduado en adiestramiento asociado al Centro de Identificación y Seguridad Digital CISED específicamente al Dpto. de Componente, graduado en el curso 2012-2013.

Correo electrónico: *royli@uci.cu*

Agradecimientos

Primero agradecer a mis padres, con su apoyo, cariño y confianza, han logrado que realice dos de mis más grandes metas en la vida: la culminación de esta etapa de aprendizaje y el hacerlos sentirse orgullosos de mi persona. Ustedes han sido mi sostén, los autores de lo que soy hoy día y los que me han llevado hasta la orilla.

A mis hermanas y hermano por ser mis amigos, mi ejemplo a seguir, los que siempre están ahí junto a mis padres para que a mí no me falte nada, por quererme mucho, por apoyarme, defenderme y regañarme cuando me lo gane, gracias por ser mis segundos padres.

A mis sobrinos por alegrarme mis días.

A mis tíos, primos, a Marco, Migue, Cary, Papito, a mi familia en general por estar a mi lado y por hacerme saber que puedo contar con ellos.

A mi compañero de tesis por aguantar mi estrés, por emprender este camino conmigo, sin él no hubiera sido posible, muchas gracias.

A todos y cada uno de mis compañeros de universidad, a Ale por enseñarme las derivadas y a Adry, entre otras cosas por enseñarme las integrales, gracias por la inestimable ayuda, a las chicas del apartamento, por los grandes momentos que hemos pasado juntas; quiero agradecer especialmente a mis amigas Wendy, Tey, Lissy y Lisandra, por hacerme reír o por secarme las lágrimas cuando me hizo falta, espero sigamos juntas en esta etapa que comienza.

A Erne por permanecer a mi lado todo este tiempo, por su comprensión, amor y ternura.

Alisbet

A mi madre por todo su sacrificio, por su apoyo incondicional y por ser un ejemplo a seguir para mí durante toda mi vida.

A mi padre, que hoy no está aquí pero seguro está esperando saber cómo termina esta historia de 5 años en la que ha estado conmigo.

A mi familia, por todo el apoyo que me han brindado para lograr cada una de mis metas. A mi segunda madre Eva, que siempre está junto a mí en todo.

A mis amigos, que hicieron posible que estos años de universidad se convirtieran en un recuerdo inolvidable. Todos estarán en mi corazón siempre.

En especial a Yoilen y Reynier por toda su ayuda en el desarrollo de esta investigación que en un principio me parecía que se me iba un fusible en ella.

A Alisbet, por soportarme hasta en mis peores momentos y hacerme trabajar incluso cuando el vago me atacaba con mayor fuerza.

Daniel

A los profesores de la universidad por cada granito de arena que aportaron en nuestra formación como profesionales, a los tutores por su guía y sugerencias en la realización de este trabajo.

A todos los que de una forma u otra han contribuido a nuestro crecimiento profesional y humano en general.

Dedicatoria

A mis 2 madres y a mis 2 padres por creer siempre en mí, darme tanto amor y guiarme por el buen camino.

Alisbet

A mis padres por todo su sacrificio para lograr este sueño.

Daniel

Resumen

En la Universidad de las Ciencias Informáticas se desplegó a finales del curso 2012-2013, un nuevo sistema de emisión de credenciales desarrollado por el Centro de Identificación y Seguridad Digital. Durante el despliegue y utilización de este sistema, se detectaron dificultades que afectaron la calidad de la imagen captada, tales como: la incorrecta iluminación y la mala calidad de las cámaras web utilizadas, dañando algunos aspectos de calidad en la imagen como la nitidez, iluminación, balance de colores, entre otros; además de no verificarse las especificaciones de escena, presentes en la norma ISO/IEC 19794-5, como que el individuo estuviera mirando al frente y que la pose fuera la adecuada a la hora de capturar la imagen. Por otra parte, si un usuario tenía una imagen que cumpliera con los requisitos establecidos, este no podía actualizarla en el directorio de la universidad por no contar con ningún mecanismo que facilitara este proceso. Por este motivo el objetivo de este trabajo se centra en el desarrollo de una aplicación web para la actualización de la imagen facial del ciudadano de la Universidad de las Ciencias Informáticas cumpliendo con el estándar ISO/IEC 19794-5 para documentos de identificación y el reconocimiento facial. Como parte de la investigación se lleva a cabo una valoración crítica de los métodos y algoritmos existentes que permiten realizar el reconocimiento de rostros y la validación de parámetros de calidad en imágenes digitales; se seleccionan las herramientas necesarias para el desarrollo del sistema y se completan las fases de diseño, implementación y prueba de la solución, obteniéndose como resultado una aplicación lista para ser desplegada.

Palabras clave: aspectos de calidad en la imagen, estándar ISO/IEC 19794-5, reconocimiento facial.

Índice

Índice de figuras

Índice de tablas

Introducción

Con el transcurso de los años se han desarrollado y extendido numerosas vías para garantizar la seguridad en las diferentes entidades, sin embargo, son diversos los métodos delictivos que han surgido a través de los cuales personas malintencionadas tratan de aprovecharse de información vital de las mismas. Los gobiernos, instituciones y empresas tienen como objetivo que los procedimientos de control de acceso sean más confiables y que la autenticación se realice de forma sencilla y rápida, para esto deben adaptarse aceleradamente a los nuevos mecanismos de seguridad tales como el uso de contraseñas, números de identificación personal (PIN por sus siglas en inglés), nombres de usuario y la biometría. Esta última se encarga del reconocimiento automático de personas mediante sus rasgos físicos (cara, retina, iris, voz, huellas dactilares) o rasgos de conducta (forma de andar, de escribir), para verificar identidades o para identificar individuos (1).

El rostro es la característica biométrica utilizada por los seres humanos para la realización del proceso de reconocimiento facial, por lo que la tecnología de reconocimiento facial se basa en el concepto de biometría, hablándose de una tecnología basada en computadoras que determina la ubicación y el tamaño del rostro de las personas en imágenes digitales. Desde hace varias décadas, las imágenes de rostros han sido utilizadas para identificar a los individuos en los documentos personales, a raíz de esto se utiliza la tecnología de reconocimiento facial para combatir las falsificaciones de pasaportes, en la identificación de niños extraviados y para minimizar el fraude en las identificaciones (2).

Con el aumento de los sistemas biométricos basados en las imágenes de rostros, ha surgido la necesidad de definir estándares que certifiquen la calidad de estas imágenes, estableciendo los requisitos indispensables para que las imágenes que se utilizan tengan valor identificativo y se permita la interoperabilidad entre diferentes sistemas.

La norma internacional ISO/IEC[1] 19794-5 creada por el Comité Internacional para los Estándares de Información y Tecnología y adoptada por la Organización Internacional de la Aviación Civil (ICAO por sus siglas en inglés), es uno de los estándares de calidad de imágenes más efectivo utilizado hoy día, el cual

[1] ISO: Organización Internacional para la Estandarización.
IEC: Comisión Electrotécnica Internacional.

tiene como objetivo establecer los requisitos de las imágenes de rostros para aplicaciones de reconocimiento de personas y definir un formato para el almacenamiento e intercambio de las fotografías (3).

En la Universidad de las Ciencias Informáticas (UCI), no siempre se evalúan las imágenes de rostros siguiendo las normas establecidas en los estándares internacionales de calidad de imágenes faciales; ejemplo de esto lo constituye el despliegue a finales del curso 2012-2013 de un nuevo sistema de emisión de credenciales desarrollado por el Centro de Identificación y Seguridad Digital (CISED). Durante la utilización del mismo se detectaron dificultades que afectaron la calidad de la imagen captada, tales como: la incorrecta iluminación y la mala calidad de las cámaras web utilizadas, dañando algunos aspectos de calidad en la imagen como la nitidez, iluminación, balance de colores, entre otros, esto trajo como consecuencia que las personas se percibieran borrosas en las imágenes, que los rasgos faciales, principalmente en las personas de color, no se definieran bien, que en las personas que portaban espejuelos no se pudieran visualizar claramente los ojos por la existencia de reflejos en los cristales. Además no se verificaron las especificaciones de escena presentes en la norma ISO/IEC 19794-5, como que el individuo estuviera mirando al frente y que la pose fuera la adecuada a la hora de capturar la imagen. Los casos anteriormente planteados incidieron negativamente en la obtención de una imagen de calidad.

Los elementos dados permiten identificar el siguiente **problema de investigación**: ¿Cómo permitir la actualización de la imagen facial del ciudadano de la UCI garantizando que se cumpla el estándar ISO/IEC 19794-5 para documentos de identificación y el reconocimiento facial?

Para encontrar una solución al problema planteado se define como **objeto de estudio**: El proceso de validación de la calidad de una imagen facial para su utilización en algoritmos de reconocimiento facial.

Se formula como **objetivo general:** Desarrollar una aplicación web que automatice la actualización de la imagen facial del ciudadano de la UCI cumpliendo con el estándar ISO/IEC 19794-5 para documentos de identificación y el reconocimiento facial. Para dar cumplimiento al objetivo general, el mismo se ha desglosado en los siguientes **objetivos específicos**:

- ✓ Analizar los estándares, métodos y sistemas para la verificación de imágenes que garanticen el cumplimiento de estándares internacionales para el reconocimiento facial.

- ✓ Definir las tecnologías, herramientas y metodologías para el desarrollo de la solución.
- ✓ Diseñar una aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI.
- ✓ Implementar la solución propuesta para su posterior uso en la universidad.
- ✓ Validar el funcionamiento de la solución para la detección de posibles errores.

Para dar cumplimiento a los objetivos antes propuestos se definen las siguientes **tareas de la investigación**:

- ✓ Definición de los aspectos teóricos referentes al proceso de validación de imágenes faciales. Responsable: Alisbet Fernández Rojas.
- ✓ Análisis de las tendencias actuales relacionadas a los componentes de actualización en línea a nivel nacional e internacional. Responsable: Daniel Vega Torres.
- ✓ Análisis del estado del arte internacional sobre las diferentes alternativas, métodos o técnicas para la validación de imágenes faciales. Responsable: Daniel Vega Torres.
- ✓ Definición de las tecnologías y herramientas a utilizar para el desarrollo de la aplicación. Responsable: Alisbet Fernández Rojas.
- ✓ Definición de los requisitos del software. Responsables: Alisbet Fernández Rojas y Daniel Vega Torres.
- ✓ Definición de la arquitectura del software. Responsable: Daniel Vega Torres.
- ✓ Diseño de las interfaces de la aplicación web. Responsable: Alisbet Fernández Rojas.
- ✓ Definición de los artefactos ingenieriles derivados de la metodología de desarrollo de software seleccionada. Responsable: Alisbet Fernández Rojas.
- ✓ Implementación de algoritmos para la detección de rostro, fondo uniforme y detección de espejuelos. Responsable: Alisbet Fernández Rojas.
- ✓ Desarrollo de una aplicación web para la actualización de la imagen facial del ciudadano de la UCI cumpliendo con el estándar ISO/IEC 19794-5. Responsable: Alisbet Fernández Rojas y Daniel Vega Torres.
- ✓ Realización de pruebas a la aplicación. Responsables: Alisbet Fernández Rojas y Daniel Vega Torres.

Métodos Científicos a utilizar:

- Métodos Teóricos

Analítico–Sintético facilitó el procesamiento de la información obtenida en la investigación realizada sobre el objeto de estudio. Se realiza un estudio profundo de los conceptos, metodologías y herramientas a utilizar en el desarrollo de la investigación basándose en la revisión de documentos, artículos e informes.

Análisis histórico lógico posibilitó el conocimiento del estado del arte del objeto de estudio y de los algoritmos de reconocimiento facial analizados, así como su evolución y desarrollo. Permitiendo tener un mayor conocimiento para aplicarlo en la solución del problema presentado.

- Métodos Empíricos

Entrevista permitió obtener la información necesaria relacionada con los problemas presentes en los procesos de validación de imágenes, mediante entrevistas realizadas a distintos directivos del CISED y de la UCI en general (ver preguntas de la Entrevista en el Anexo 1).

Experimental se utilizó para registrar los tiempos de respuesta de la aplicación desarrollada.

Justificación de la investigación:

La relevancia de esta investigación se centra en la importancia que tiene la implementación de una aplicación web, que valide la calidad de una imagen facial de perfil de usuario y la actualice en el directorio UCI, aportándole a la universidad un producto reutilizable que automatice el proceso de validación de la calidad de la imagen facial.

Aporte práctico de la investigación:

Con el desarrollo de una aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI, se contará con un componente capaz de validar la calidad de una imagen facial siguiendo el estándar ISO/IEC 19794-5 el cual puede ser reutilizado en futuros desarrollos de software, el mismo será utilizado por la aplicación web permitiendo así renovar la imagen de perfil del usuario en el directorio UCI.

Estructura del documento:

Este documento está compuesto por tres capítulos los cuales recogen todo el proceso realizado:

Capítulo 1: "Fundamentación Teórica":

En este capítulo se abordarán conceptos que ayudan a un mejor entendimiento de la investigación, se realiza un estudio del estado del arte de los sistemas similares nacionales e internacionales que realizan validaciones de parámetros de calidad en imágenes faciales, se especifica la metodología a usar y la justificación de su uso, además de los lenguajes y herramientas de desarrollo que se utilizarán para implementar el sistema.

Capítulo 2: "Análisis y Diseño del sistema":

En este capítulo se presentan las fases de Planificación y Diseño definidas por la metodología Programación Extrema (XP) para dar solución al problema de investigación. Se define la propuesta del sistema por parte del equipo de desarrollo, se diseña el diagrama del modelo del dominio, las principales funcionalidades y los requisitos no funcionales con los que debe cumplir el sistema así como la arquitectura definida para la realización del mismo, además de los diferentes artefactos generados por XP como son: las historias de usuario, el plan de entregas, el de iteraciones y las tarjetas CRC.

Capítulo 3 "Implementación y Pruebas":

En este capítulo se muestran algunos de los artefactos relacionados con la implementación de la aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI, entre estos se encuentran las tareas de ingeniería, los diagramas de componentes y despliegue. Se describe la validación del sistema mediante pruebas unitarias y de aceptación. Además se muestran las principales interfaces.

Capítulo 1: Fundamentación teórica.

1.1 Introducción

Este capítulo aborda los elementos teóricos que sustentan el objeto de estudio y el objetivo de la investigación. Se relacionan todos los conceptos que desde el punto de vista teórico permiten un mejor entendimiento de la problemática planteada en sentido general.
También se realiza un estudio sobre las soluciones que existen en el mercado, con el fin de facilitar la comprensión de la importancia de la investigación, su alcance y su aporte científico.
Por otra parte, se realiza un análisis comparativo de las principales metodologías existentes, tecnologías y herramientas que se utilizan a lo largo de la presente investigación, conjuntamente con los algoritmos para la validación y el procesamiento de imágenes faciales que mejor se adapten a las necesidades del sistema.

1.2 Principales conceptos

- Imagen facial

Una imagen facial digital simboliza el aspecto de una persona a través de una representación de su zona superior. Permite verificar identidades a través de un análisis de las características faciales de la persona en la imagen. Las imágenes faciales digitales se pueden obtener por medio de dispositivos de conversión analógica-digital como los escáneres y las cámaras digitales (2).

- Calidad de imágenes faciales

¿Qué es calidad?

Definición de la norma ISO 9000: *"Calidad: grado en el que un conjunto de características inherentes cumple con los requisitos".*

Real Academia de la Lengua Española: *"Propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla como igual, mejor o peor que las restantes de su especie".*

Según los autores del trabajo: *"Calidad es cumplimiento de requisitos, satisfacción de las expectativas del cliente".*

La calidad de imágenes faciales se refiere al conjunto de normas y directrices de calidad, que se deben tener en cuenta en el proceso de validación de parámetros de calidad de las mismas. Las imágenes de rostros deben cumplir con los requisitos indispensables establecidos en los estándares internacionales para que tengan valor identificativo (4).

- Atributos para medir la calidad de imágenes faciales

Algunas de las medidas a evaluar en el proceso de verificación de calidad en imágenes faciales son:

- ✓ Nitidez
- ✓ Estado de los ojos.
- ✓ Pose.
- ✓ Expresión de la boca.
- ✓ Mirada al frente.
- ✓ Iluminación.
- ✓ Balance de colores.
- ✓ Rostro centrado
- ✓ Fondo uniforme
- ✓ Ojos rojos
- ✓ Presencia de espejuelos.

- Estándares de calidad de imágenes faciales

¿Qué es un estándar?

De acuerdo con la definición de la Real Academia Española, "*estándar es aquello que sirve como tipo, modelo, norma, patrón o referencia*".

¿Qué es un estándar de calidad?

Estándar de calidad es el que reúne los requisitos mínimos en busca de la excelencia (5).

¿Qué es un estándar de calidad de imágenes faciales?

Definen una estructura de datos estándar con los que las imágenes de rostros deben cumplir para tener valor identificativo y se permita así la interoperabilidad entre diferentes sistemas. Establecen los requisitos indispensables para que las imágenes puedan ser utilizadas en documentos oficiales, así como para el almacenamiento e intercambio de las fotografías (3). En la actualidad existen diferentes estándares de calidad de imágenes faciales, entre los que se encuentran: ISO/IEC 19794-5 e ICAO.

ISO/IEC 19794-5

La norma ISO/IEC 19794-5 es la quinta parte de un estándar de formato de intercambio de datos biométricos de varias partes. El estándar está organizado por modalidades, en otras partes cubre imágenes de huellas dactilares, iris, geometría de la mano, entre otros. La parte 5 de la norma es el estándar de la cara más ampliamente implementado, más desarrollado activamente, y el más moderno. La norma ISO/IEC 19794-5 ha sido utilizada por algunas de las principales aplicaciones de gestión de identidad. La más importante es la del pasaporte electrónico, la cual la Organización de Aviación Civil Internacional formalizó en su norma 9303 de la ICAO (6).

La norma ISO/IEC 19794-5 incluye más de 15 requisitos, que se dividen en tres grupos generales:

1. Especificaciones de escena (pose facial, expresión, uso de accesorios).
2. Características fotográficas (iluminación, nitidez, distancia focal, exposición, saturación).
3. Atributos propio de las imágenes digitales (resolución, nivel de compresión, formato de los archivos, formas de almacenamiento) (2).

ICAO

La Organización de la Aviación Civil Internacional establece normas y regulaciones internacionales necesarias para garantizar la seguridad, eficiencia y regularidad del transporte aéreo, sirve de catalizador para la cooperación en todas las esferas de la aviación civil entre sus 185 Estados contratantes.

En la tercera edición del Doc. 9303, Parte 3, incorpora la nueva forma de funcionamiento mundial opcional para la identificación biométrica del titular y para el almacenamiento de los datos conexos en un circuito integrado sin contacto (7).

Algunas de las directrices ilustrativas para los retratos en los DVLM[2] son:

- ✓ Pose

1.1 La fotografía tendrá menos de seis meses de captada a la fecha de la solicitud.

1.2 Deberá mostrar una toma de primer plano de la cabeza y los hombros.

1.3 La fotografía debe captarse de modo que una línea horizontal imaginaria entre los centros de los ojos resulte paralela al borde superior de la imagen.

1.4 El rostro debe ser enfocado con total nitidez y no deberán aparecer imperfecciones, como manchas de tinta o arrugas.

[2] DVLM: Documento de viaje oficial de lectura mecánica en forma de tarjeta

1.5 La fotografía deberá mostrar al sujeto en pose frontal y mirando directamente a la cámara con expresión neutra y boca cerrada (8).

- ✓ Iluminación, exposición y equilibrio cromático

2.1 La iluminación será uniforme sin sombras o reflejos sobre el rostro o en el trasfondo.
2.2 Los ojos del sujeto no deben aparecer rojos.
2.3 La fotografía debe tener brillo y contraste apropiados (8).

- ✓ Presentación del retrato a la autoridad expedidora

3.1 Cuando el retrato se presente a la autoridad expedidora en forma impresa, la fotografía producida, ya sea utilizando técnicas fotográficas convencionales o técnicas digitales, deberá estar sobre el papel fotográfico, de buena calidad y tener las dimensiones máximas específicas (8).

- ✓ Cumplimiento de las normas internacionales

4.1 La fotografía cumplirá con las definiciones apropiadas establecidas en ISO/IEC 19794-5 (8).

Como se puede observar el estándar de la ICAO se apoya en el ISO/IEC 19794-5, los autores del trabajo consideran que este último, según la investigación realizada, es el más completo y el más utilizado a nivel internacional, por lo que la aplicación web para la actualización de la imagen facial del ciudadano de la UCI se desarrollará siguiendo los parámetros establecidos en el estándar ISO/IEC 19794-5.

- Reconocimiento facial

Según el Diccionario Libre[3] es: "*La capacidad de reconocer a las personas por sus características faciales. La tecnología más avanzada se basa en el algoritmo Eigenfaces[4], que mapea las características de la cara de una persona en un espacio multidimensional de la cara. Las computadoras pueden realizar búsquedas en bases de datos faciales y / o realizar en vivo, uno-a-uno o uno-a-muchos verificaciones con una precisión sin precedentes y procesamiento de fracciones de segundo*".

[3] Diccionario Libre: Es un diccionario y una enciclopedia en línea estadounidense que recopila información de una variedad de fuentes (http://www.thefreedictionary.com).

[4] Eigenfaces: Sistema de vectores propios utilizado en la ciencia y las tecnologías de las computadoras, que obtiene información de una imagen mediante una aplicación informática para automáticamente identificar o verificar a una persona.

Según el Diccionario de Psicología[5] es: "*El proceso de identificar a un individuo utilizando sus características y expresiones faciales que típicamente se mantienen invariables durante más tiempo".*

Según el Diccionario Inglés Collins[6] es: "*La habilidad de una computadora para escanear, almacenar y reconocer rostros humanos para su uso en la identificación de personas*".

Definición asumida por los autores del trabajo: *El reconocimiento facial es el conjunto de técnicas capaces de asociar un set de datos (cualquier conjunto de características biológicas y que definen únicamente a alguien) con la identidad de una persona* (6).

1.3 Algoritmos de reconocimiento facial

El reconocimiento facial automatizado es un concepto relativamente nuevo, pues se introdujo en los años 60. Fue entonces cuando se desarrolló el primer sistema semiautomático para reconocimiento facial, el cual requería la imagen de una persona para localizar los rasgos (como ojos, nariz y boca) en las fotografías antes de que este calculara distancias a puntos de referencia en común, que posteriormente eran comparados con datos de referencia (1).

En los años 70 Goldstein, Harmon y Lesk, utilizaron 21 marcadores subjetivos específicos tales como el color del cabello y el grosor de labios para automatizar el reconocimiento facial. El problema con estas soluciones previas era que se seguía requiriendo un proceso manual. En 1988 Kirby y Sirobich aplicaron Análisis de Componentes Principales (PCA), una técnica estándar del álgebra lineal al problema del reconocimiento facial, para aumentar la exactitud de los resultados. Esto fue considerado un avance muy importante al mostrar que eran requeridos menos de 100 valores para codificar acertadamente la imagen de una cara convenientemente alineada y normalizada (9).

En 1991 Turk y Pentland, utilizando las técnicas de Eigenfaces, como se denominó al método de Kirby y Sirobich, demostraron que el error residual podía ser utilizado para detectar caras en las imágenes, un descubrimiento que permitió desarrollar sistemas automatizados fiables de reconocimiento facial en

[5] Diccionario de Psicología: Es una fuente de definiciones de psicología en línea, con más de 20 mil definiciones escritas por un equipo global de profesionales de psiquiatras y de psicología (http://psychologydictionary.org).

[6] Diccionario Inglés Collins: Diccionario impreso y en línea de inglés. Publicado por Harper Collins en Glasgow (http://www.collinsdictionary.com).

tiempo real. Si bien la aproximación era un tanto forzada por factores ambientales, creó sin embargo un interés significativo en posteriores desarrollos de éstos sistemas (10).

En la actualidad existen 3 técnicas de reconocimiento facial:

1. De rasgos locales: reconocen los ojos, la nariz, la boca, miden las distancias y los ángulos de la cara.
2. De rasgos globales: aportan información de toda la cara.
3. Mixtos: combinación de los anteriores (11).

Una de las principales ventajas del reconocimiento facial, es que se trata de un método no intrusivo, es decir, los datos pueden ser adquiridos incluso sin que el sujeto se percate de ello. Además, el aspecto facial es el método más utilizado de manera natural por los seres humanos para reconocerse unos a otros. Sin embargo, a la hora de identificar a una persona a partir de su aspecto facial, existen dificultades centradas, sobretodo, en el concepto de variabilidad. Se hace muy difícil el reconocimiento facial cuando la variabilidad entre individuos es muy pequeña (por ejemplo, el caso de familiares, especialmente el caso de los gemelos), o cuando la variabilidad entre distintas imágenes de un mismo individuo es muy amplia; esto puede ser debido a que dichas imágenes hayan sido adquiridas en diferentes condiciones de posición o iluminación (11).

El reconocimiento facial goza de gran importancia en el proceso de validación de las imágenes de rostros. La evaluación automática de la calidad de las imágenes de rostros es punto clave dentro del desarrollo de sistemas biométricos de reconocimiento de rostros. Los principales métodos de reconocimiento utilizados en la validación de la calidad de imágenes faciales son:

- ✓ Reconocimiento Facial Basado en la Apariencia.
- ✓ Reconocimiento Facial utilizando Análisis de Componentes Principales Kernel.
- ✓ Reconocimiento Facial Basado en Puntos Característicos de la Cara (11).

Los sistemas de reconocimiento facial modernos se basan en este último, de ahí que se haya hecho énfasis en él para el desarrollo de los algoritmos a utilizar en los procesos de la validación de imágenes faciales del presente trabajo. Para ello se utiliza un Modelo de Forma Activa (Active Shape Model, ASM), con el que se logra la ubicación de puntos en el rostro y a partir de estos la construcción de algoritmos para evaluar cada medida.

1.3.1 Propuesta de algoritmos a utilizar para la validación de imágenes en el desarrollo de la solución

A continuación se presenta un esquema para la validación de la calidad de imágenes de rostros, basado en la evaluación de los parámetros establecidos en la norma ISO/IEC 19794-5:2011[7]. Se describen brevemente los algoritmos utilizados para evaluar cada parámetro y algunos detalles de su implementación.

1.3.1.1 Evaluación de la calidad de imágenes de rostros

Entre los requisitos establecidos en la norma ISO/IEC 19794-5:2011, se seleccionaron para evaluar, los más importantes dentro de las especificaciones de la escena y las características fotográficas, que son los que determinan si la imagen del rostro posee valor identificativo, estos aparecen listados en la **Tabla 1**. Existe otro grupo de parámetros recomendables a la hora de utilizar las fotos en documentos de identidad, los cuales se especifican en la **Tabla 2** (12).

Las medidas seleccionadas son analizadas de manera individual, sin embargo, basta que la imagen no cumpla con uno de los parámetros que aparecen en la **Tabla 1**, para determinar que no tiene valor identificativo. Así mismo, si no se cumple alguno de los indicadores propuestos en la **Tabla 2**, se determina que la imagen no es recomendable para documentos de identidad. Por ese motivo se decidió proponer una estructura de grafo, como se muestra en la **Figura 1**, en la que se tienen en cuenta de manera conjunta los resultados de las evaluaciones individuales para determinar la calidad final de una imagen de rostro. En el grafo que se propone, el orden de evaluación de los distintos parámetros está determinado por la influencia que tiene cada uno de ellos en la evaluación de los restantes.

Tabla 1. Parámetros seleccionados que influyen en el valor identificativo de las imágenes de rostros.

Parámetros	Especificaciones
Resolución.	La distancia entre los centros de los ojos debe ser como mínimo de 60 píxeles.
Nitidez.	Las imágenes no deben estar borrosas o desenfocadas.

[7] ISO/IEC 19794-5:2011: Estándar ISO/IEC 19794-5 corregido y publicado en el 2011.

Estado de los ojos.	Los ojos deben estar abiertos de manera natural. El iris y la pupila deben estar claramente visibles.
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.	La luz debe estar distribuida uniformemente en el rostro, el cual no debe estar afectado por sombras, ni regiones brillantes.
Balance de colores.	Los colores deben ser naturales, no saturados y con un contraste adecuado.

Tabla 2. Parámetros adicionales seleccionados, recomendables para documentos de identidad.

Parámetros	Especificaciones
Dimensiones del rostro.	El largo de la cabeza debe comprender entre 70 y el 80% del alto de la imagen. La razón entre el ancho de la cabeza y el de la imagen debe encontrarse entre 4:7 y 1:2.
Rostro centrado.	El rostro debe estar centrado de manera horizontal y la posición vertical de los ojos debe ser entre el 50 y el 70% del alto de la imagen.
Fondo uniforme.	El fondo tras la silueta de la persona debe ser uniforme o tener una transición suave en una dirección.
Ojos rojos.	No se admiten fotos con ojos rojos. Los colores deben ser naturales.
Presencia de espejuelos.	Solo se permite el uso de espejuelos si la persona normalmente los usa. No puede haber luces reflejadas en los espejuelos.

La existencia de problemas en determinados parámetros, causa que otros no puedan ser bien evaluados. Por ejemplo, si la imagen está desenfocada o no es nítida, no hay seguridad en la evaluación con respecto al estado de los ojos y de la boca, porque los píxeles en esas regiones no estarían bien

definidos. Por otra parte, esta estructura permite que se mejore la eficiencia, en cuanto al tiempo computacional necesario cuando alguna de las medidas falla y se decide, sin disminuir la eficacia, que la imagen no tiene calidad. Si se falla en la evaluación de alguno de los parámetros del grafo, la respuesta general de la calidad de la imagen y la de ese parámetro en específico, es negativa. En ese caso, el resto de los parámetros que queden por debajo de ese nodo del grafo no se evalúan.

Como puede observarse en la **Figura 1**, el primer paso en el esquema que se propone, es la detección del rostro. Este es el primer paso en cualquier sistema que trabaje con imágenes de rostros. Este paso permite trabajar únicamente con la región del rostro. La detección en el sistema se realiza utilizando el algoritmo de Viola y Jones[8], basado en el clasificador Adaboosting[9], el cual reporta los mejores resultados en la literatura con un 93.9% de efectividad (13), utilizando el entrenamiento del clasificador Haar[10] propuesto por Lienhart (14). Como es lógico y se puede observar en la figura, si en la imagen que se intenta evaluar no se detecta un rostro, es imposible realizar alguna evaluación.

Una vez detectado el rostro, se procede a la obtención de los puntos característicos que se sitúan mediante un modelo probabilístico utilizando un Modelo de Forma Activa. Solamente al comparar las coordenadas de los puntos situados en el centro de los ojos, es posible determinar si estas cumplen con el parámetro indispensable de resolución y en caso de no cumplirlo, se evita analizar innecesariamente el resto de los requisitos.

[8] Viola y Jones: Es un algoritmo de detección de objetos en imágenes basado en la técnica de aprendizaje de máquina. Primero se calcula una representación de la imagen llamado Imagen Integral. A continuación se utiliza un algoritmo de aprendizaje basado en el AdaBoost que selecciona las características visuales más importantes para crear clasificadores. Por último, estos clasificadores se combinan en una estructura de cascada que permite reducir el tiempo de cómputo empleado para la detección de objetos (13).

[9] AdaBoost: Es un algoritmo de aprendizaje de máquina formulado por Yoav Freund y Robert Schapire. Permite combinar el resultado de algoritmos de aprendizaje débiles en una suma de pesos que representa el resultado final del clasificador mejorado (17).

[10] Clasificador Haar: Es un método desarrollado por Viola y Jones y es una versión del algoritmo "AdaBoost". Es un clasificador basado en árboles de decisión con entrenamiento supervisado. El entrenamiento se realiza determinando una serie de características basadas en sumas y restas de los niveles de intensidad de la imagen. Basándose en estas características locales se puede obtener un detector de objetos robusto. También se denominan estos clasificadores mediante el nombre de cascada, ya que el resultado del clasificador es el fruto de varios clasificadores más simples o etapas (15).

El siguiente elemento a analizar es la nitidez, la cual es decisiva en el resto de las evaluaciones, puesto que en una imagen distorsionada los píxeles están alterados y esto conlleva a errores en el análisis de las distintas regiones del rostro.

Luego se evalúa la pose, la cual, es de los parámetros más importantes en el reconocimiento de rostros y además, si no es correcta, puede influir negativamente en el resultado de la clasificación de medidas como la iluminación, la expresión de la boca y la mirada al frente, por lo que una pose no frontal provocaría que estos métodos no dieran una respuesta confiable.

Seguidamente se evalúa la iluminación, parámetro que influye en la eficiencia del algoritmo utilizado para evaluar la mirada al frente.

El balance de colores, la detección de los ojos, el estado de los ojos, el rostro centrado, el tamaño del rostro, la detección de la boca, y la presencia de espejuelos son medidas independientes que no influyen unas sobre las otras, por lo que se analizan todas en el mismo nivel del grafo. Para la detección de los ojos y la boca se utiliza el algoritmo de Viola y Jones, basado en el clasificador Adaboosting (13), utilizando el entrenamiento del clasificador Haar propuesto por Castrillón (16), el cual tiene el inconveniente de no poder detectar las bocas que se encuentran ocultas total o parcialmente por causa de bigotes o barbas. Si no es posible realizar algunas de estas detecciones no se validan los parámetros que dependen de la misma y se deja el criterio al experto humano.

Si la detección de los ojos se realiza de forma satisfactoria se procede a evaluar los indicadores de mirada frontal y ojos rojos. Así mismo, si la detección de la boca arroja un resultado favorable, el sistema verifica el estado de la boca.

Como salida del grafo se determina si la imagen tiene valor identificativo y es posible su uso en el perfil del usuario del directorio UCI.

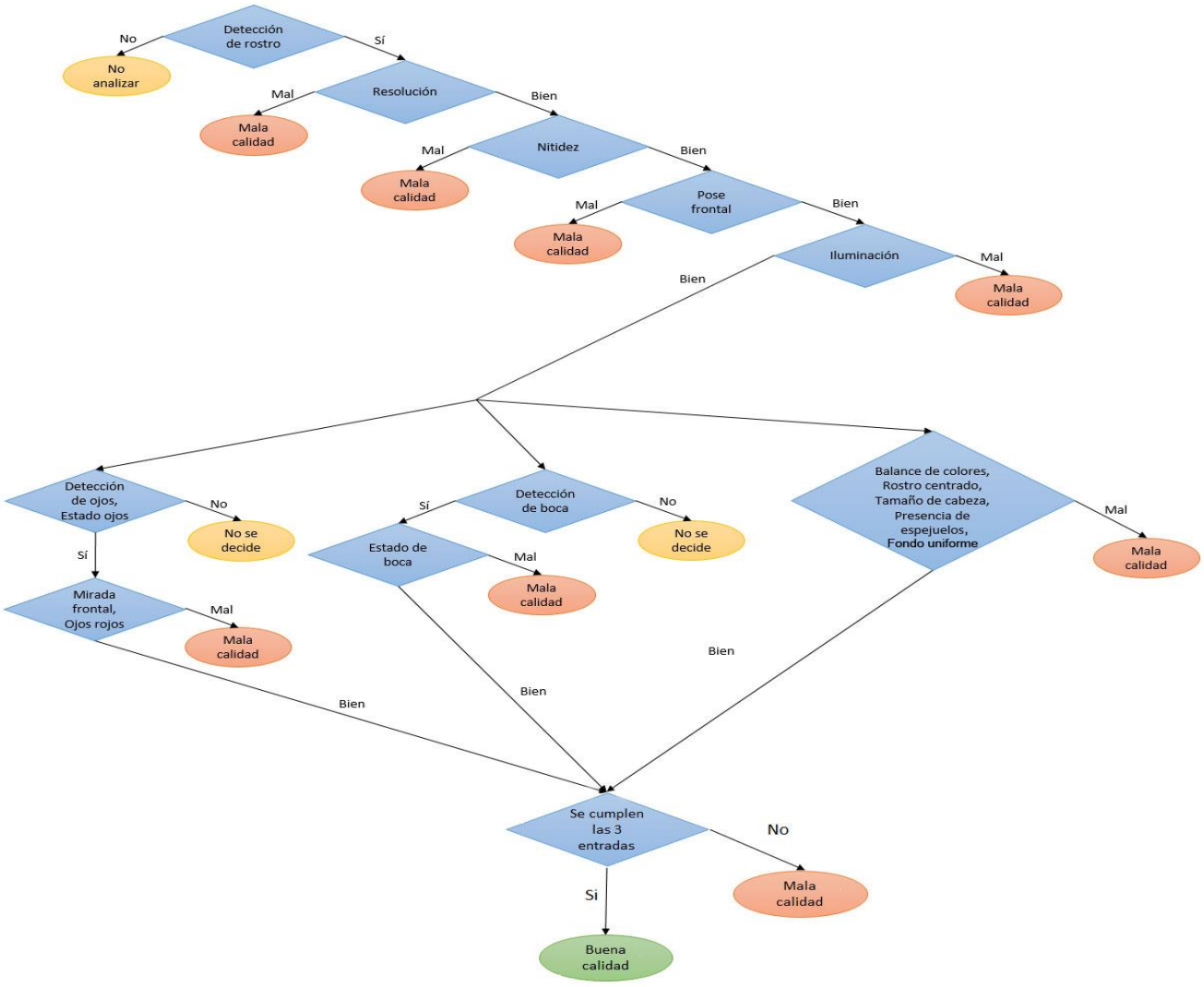


Figura 1: Estructura de grafo para la evaluación de la calidad de una imagen de rostro.

A continuación se describen brevemente los algoritmos utilizados para evaluar cada uno de los parámetros.

1.3.1.2 Descripción de la evaluación de cada una de las medidas seleccionadas

Los parámetros de resolución, el rostro centrado y el tamaño de la cabeza, son muy sencillos de evaluar a partir de la información sobre la región de la imagen que pertenece al rostro y los ojos, es decir, la región devuelta por el algoritmo de Viola y Jones y las coordenadas determinadas como el centro de los ojos por el modelo probabilístico de la biblioteca ASM. A su vez, el estado de los ojos se determina a partir de la

detección de los ojos por el método de Viola y Jones, ya que el entrenamiento utilizado no detecta los ojos si estos no se encuentran abiertos de forma correcta.

1.3.1.2.1 Nitidez

Este parámetro evalúa el grado de distorsión de la imagen. En la **Figura 2** se presenta una imagen distorsionada (a) y otra que no tiene este problema (b). Como puede observarse, cuando una imagen está borrosa o desenfocada, pierde los detalles de los bordes.

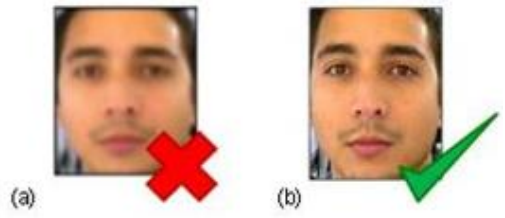


Figura 2: Representación de la nitidez en la imagen de rostro: a) mala b) buena (2)

En una imagen en la que los bordes estén bien definidos existe un gran número de píxeles, pertenecientes a estos bordes, en los que sus píxeles vecinos son muy diferentes. Teniendo esto en cuenta, si se remplaza cada píxel con el valor medio de las intensidades de los píxeles circundantes, la imagen obtenida, que es una versión suavizada de la imagen original, es muy diferente a esta. Por otra parte, si la imagen es borrosa, las intensidades de los píxeles en pequeñas vecindades es muy similar y la imagen suavizada que se obtiene es muy parecida a la original. Luego, para evaluar la nitidez se seleccionó un método que ha sido utilizado en otras aplicaciones de procesamiento digital de imágenes, el cual consiste en aplicar un filtro de media a la imagen que se está evaluando (en este trabajo se usó un filtro de 3x3 píxeles) para obtener la imagen suavizada, que luego se resta de la imagen original para saber cuánto se diferencian (18).

1.3.1.2.2 Expresión de la boca

Para el caso de la expresión de la boca, se decidió utilizar un algoritmo basado en los puntos interiores de la misma obtenidos por la biblioteca ASM. Primero se verifica la simetría de la boca utilizando el mismo algoritmo propuesto para la validación de la simetría del rostro tomando los triángulos pertenecientes a la zona de la boca. Después se obtiene una distancia euclidiana entre los puntos interiores y los exteriores a la boca y se comparan con un umbral obtenido mediante la evaluación de dicha medida en imágenes con una correcta expresión de la boca.

1.3.1.2.3 Pose

Este parámetro se utiliza para determinar si el rostro se encuentra en una pose frontal. Este es uno de los factores que más influye en la efectividad de los algoritmos de reconocimiento de rostros, por lo que es muy importante su evaluación. El método seleccionado es el propuesto por Gao[11] (19), en el mismo, para evaluar la pose, se analiza la simetría del rostro. Como puede verse en la **Figura 3**, las desviaciones de la pose del rostro provocan afectaciones en su simetría. Para el análisis de la simetría, se utilizó una malla triangular diseñada por los autores de la investigación, representada en la **Figura 4**, sobre la cual fueron aplicados los Patrones Binarios Locales[12] (LBP) con la medida de similitud Chi-Cuadrado para comparar las dos mitades del rostro (20).

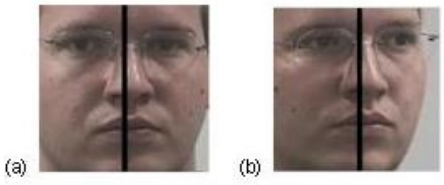


Figura 3: Relación entre la pose del rostro y la simetría vertical: a) rostro frontal y simétrico, b) rostro no frontal y asimétrico (2).

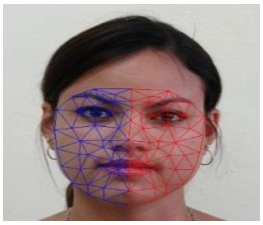


Figura 4: Malla triangular utilizada en la evaluación de la pose.

[11] Método de Gao: Se basa en el análisis de la simetría utilizando características locales de la imagen, como los valores de los píxeles, o valores de los píxeles filtrados localmente. Las diferencias entre las características de la imagen en los correspondientes píxeles de derecha e izquierda, proporcionan medidas locales de asimetría causada por una pose no frontal. Si la imagen es estrictamente simétrica entre derecha e izquierda, dichas diferencias deben ser igual a 0. Propone el uso de las diferencias entre los histogramas locales como medidas de asimetría en la imagen. Los histogramas se calculan mediante el uso de Patrones Binarios Locales y luego se calcula la distancia entre los correspondientes histogramas de derecha e izquierda (19).

[12] Operador de Patrones Binarios Locales: Es utilizado para transformar una imagen en un arreglo, o imagen con etiquetas enteras describiendo características de pequeñas proporciones de la imagen. Estas etiquetas o sus estadísticas, mayormente el histograma, se utilizan para el análisis posterior de la imagen. Las versiones más utilizadas comúnmente están diseñadas para imágenes en escala de grises, aunque el operador ha sido extendido para su uso en imágenes a color (20).

1.3.1.2.4 Iluminación

La iluminación es otro de los factores que más influye en el desempeño de los algoritmos de reconocimiento de rostros. A pesar de que el tema del reconocimiento de rostros bajo condiciones variables de iluminación ha sido ampliamente abordado en la literatura, muy poco se encuentra sobre la evaluación de este parámetro en las imágenes de rostros (2).

Se desarrolló para este propósito una versión de un algoritmo basado en el análisis local del comportamiento de la iluminación sobre el rostro, debido a que esta afecta de manera diferente las distintas partes de la cara (2). Primeramente, se ubica de manera automática una malla triangular sobre la imagen a analizar cómo puede observarse en la **Figura 5**, en la que cada triángulo define una región con una luminancia aproximadamente constante.

De cada región, se determina ese valor promedio de luminancia y se compara con el valor esperado para una imagen de buena calidad.

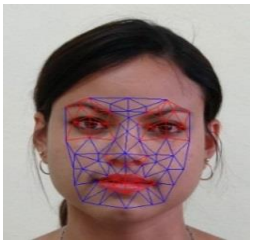


Figura 5: Regiones del rostro utilizadas para analizar la influencia de la iluminación.

1.3.1.2.5 Balance de Colores

Este parámetro se define para evaluar el balance de los colores en la imagen, por tanto está relacionado con características fotográficas de la misma. En esta evaluación se tienen en cuenta el Contraste, la Saturación y la Exposición como indicadores que se incluyen dentro de la estimación del comportamiento de los colores de la imagen.

El contraste es la diferencia relativa en intensidades, entre un punto de una imagen y sus alrededores. La medida seleccionada para evaluar el contraste se describe en la **Figura 6**, esta se basa en el histograma de las intensidades de los píxeles de la imagen en el modelo RGB[13] (20). Inicialmente se localiza el centro de masa del histograma y se calcula el valor de la amplitud del 98% de la masa partiendo de dicho centro.

[13] RGB: Modelo aditivo de colores rojo, verde, azul.

En este caso se considera que un 2% representa el ruido en la imagen. El contraste está dado entonces por la amplitud del 98% de la masa de los píxeles, como se ejemplifica a continuación**.**

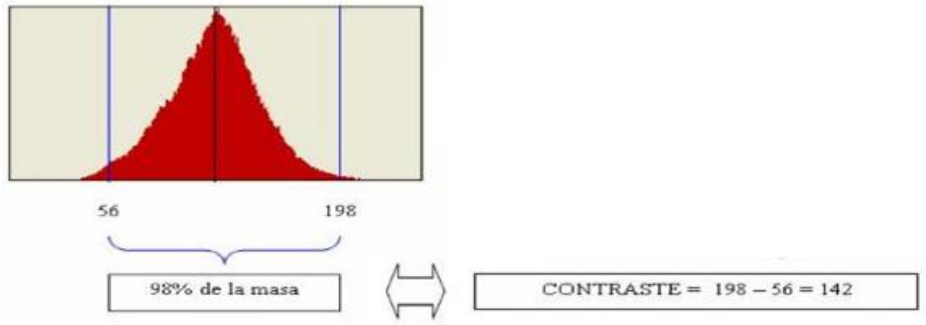


Figura 6: Esquema del cálculo del contraste (2).

Para la evaluación de la saturación y la exposición se hace la conversión de la imagen del espacio de colores RGB al espacio HSL[14]. En el espacio HSL se trabaja con las componentes saturación y luminancia, esta última se utiliza para la evaluación de la exposición. En ambos casos se analiza el valor promedio de estos parámetros en la imagen (20).

Por cada uno de estos tres indicadores se definen dos umbrales, uno mínimo y uno máximo. El intervalo definido por estos umbrales en cada uno de los indicadores, representa los valores de comportamiento para imágenes de buena calidad, en este caso con colores naturales. Para determinar estos umbrales, se formó un conjunto de 100 imágenes capturadas en buenas condiciones para dichos parámetros y se calcularon los valores mínimo y máximo obtenidos para cada uno de los tres factores. Luego, para validar y refinar estos valores, fue necesario crear 6 subconjuntos de imágenes, representando cada una de las posibles afectaciones: sobre-contraste, sub-contraste, sobre-saturación, sub-saturación, sobre-exposición, sub-exposición. En la **Figura 7** se muestra un ejemplo de las imágenes en cada uno de los subconjuntos.

Para determinar finalmente la calidad respecto al balance de colores de una imagen, se evalúan estos tres factores y se emite una estimación final, donde la buena calidad está representada por la evaluación positiva de los tres, siendo la imagen de mala calidad en caso contrario.

[14] HSL: Espacio de color de saturación, tono y luminancia.

Figura 7: Ejemplo de cada uno de los 7 subconjuntos de saturación.

1.3.1.2.6 Fondo uniforme

El algoritmo para evaluar la uniformidad del fondo, se diseñó a partir de un grupo de resultados encontrados en la literatura sobre segmentación de imágenes de rostros (2). El proceso seguido se resume en la **Figura 8.**

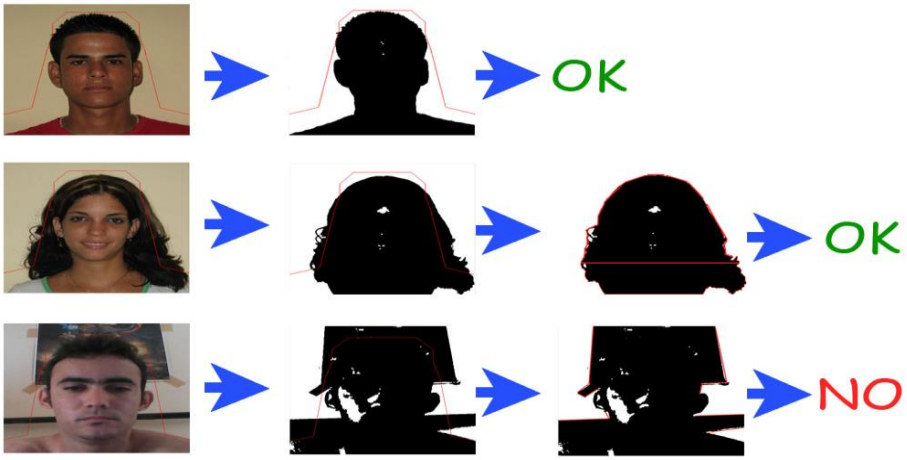


Figura 8: Pasos del proceso para evaluar la uniformidad del fondo.

Primeramente, se utiliza una plantilla que tiene un área de fondo preestablecida. Cuando esta plantilla se adapta a la imagen de rostro, el área que es considerada como fondo se evalúa según la desviación estándar de la intensidad de los píxeles que la conforman. Si el resultado de esta técnica es que el fondo no es uniforme, se entra en más detalles para asegurar que la plantilla no haya incluido como fondo parte de la persona (por ejemplo, hombros o pelo). En este caso se utiliza una función de inversión binaria para segmentar los objetos que no pertenecen al fondo de la imagen.

El resultado de este paso es analizado en cuanto a sus características geométricas para determinar si la segmentación fue exitosa. Si la segmentación es mala, se determina que el fondo no es uniforme ya que objetos del fondo fueron segmentados junto con la persona. Si es buena, se analiza el área exterior de la silueta (fondo), para determinar si la desviación estándar del color de los píxeles se encuentra bajo el umbral designado para identificar uniformidad de color.

1.3.1.2.7 Presencia de espejuelos

Para determinar si el individuo representado en la imagen porta espejuelos o no, se desarrolló un algoritmo basado en la propuesta de Jing y Mariani (21) para detectar y extraer de forma automática los espejuelos en imágenes de rostros. Para esto se segmenta la imagen obteniendo la región ubicada entre los ojos, para lo cual se aplica un operador de Sobel[15] a la imagen (ver **Figura 9**) y se obtiene para cada punto (x,y) la magnitud y la orientación del gradiente:

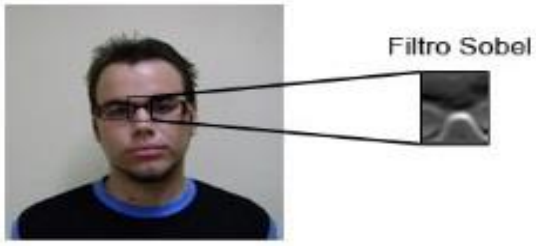


Figura 9: Proceso de detección de espejuelos.

Donde Gx y Gy representan las derivadas horizontal y vertical obtenidas en el píxel (x,y) (ver **Figura 10**) . Si el número de píxeles que tienen una dirección vertical excede un umbral establecido, se determina la presencia de espejuelos (21).

[15] El operador de Sobel realiza una medición del gradiente en un espacio de dos dimensiones de una imagen. Además, enfatiza las regiones con una alta frecuencia espacial correspondientes a los bordes. El operador consiste en un par de kernels de convolución de tamaño 3x3. Un kernel representa al otro, rotado en un ángulo de 90 grados. Estos kernels están diseñados para responder a los bordes posicionados de forma vertical y horizontal. Dichos kernels pueden ser aplicados de forma individual para obtener las medidas del gradiente en ambas orientaciones para cada punto. Los valores de gradiente obtenidos permiten calcular la magnitud y orientación del gradiente en cada punto de la imagen (22).

$$G = \sqrt{G_x^2 + G_y^2} \ , \ \theta = \arctan(G_y / G_x)$$

Figura 10: Cálculo de la magnitud y la orientación del gradiente (21).

1.3.1.2.8 Mirada frontal

Para determinar si el individuo se encuentra mirando de frente a la cámara, se desarrolló un algoritmo mediante el cual se segmenta la imagen obteniendo la región de cada uno de los ojos. A continuación, mediante el uso de puntos ubicados mediante un modelo de forma activa en el centro de cada ojo, se define un rectángulo donde se espera la presencia de la pupila del ojo. Como paso siguiente se utiliza una función de inversión binaria y luego, se ajusta la posición del rectángulo propuesto. Por último, se calcula el por ciento de oscuridad (por ciento de puntos negros) del rectángulo y se compara con un umbral preestablecido. Si el valor excede dicho umbral se determina que el individuo se encuentra mirando hacia el frente (ver **Figura 11**).

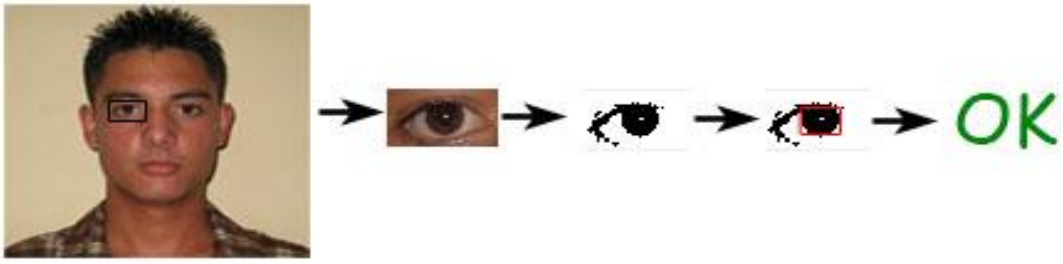


Figura 11: Proceso de verificación de mirada al frente.

1.3.1.2.9 Ojos rojos

Uno de los problemas que se puede presentar al capturar imágenes de rostros es la presencia del efecto conocido como “ojos rojos”, consistente en la aparición de pupilas rojas en fotos tomadas con flash en ambientes de poca luz, y con un flash situado cerca del lente de la cámara (23).

La evaluación de este parámetro parte de la posición detectada como el centro de los ojos. Teniendo una región rectangular G, que delimita el área donde se espera esté la pupila del ojo representada en un espacio de colores RGB (ver **Figura 13**), y se calcula el grado de coloración roja presente en dicha región (rednessG), utilizando la siguiente ecuación (ver **Figura 12**) (2):

$$redness_G = \sum_x \sum_y \frac{R(x,y)^2}{G(x,y)^2 + B(x,y)^2 + K}$$

Figura 12: Cálculo del grado de coloración roja.

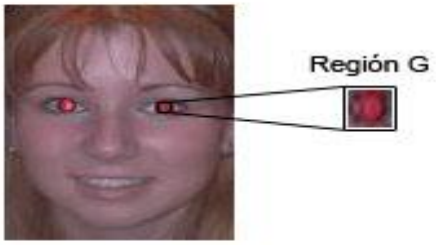


Figura 13: Región definida para determinar la presencia de ojos rojos.

1.4 Análisis de sistemas similares

Actualmente en el mundo existen varios componentes biométricos que efectúan validaciones en imágenes de rostros, entre estos se encuentran:

- **Internacionales:**

Face Component:

Face Component es un componente desarrollado por la firma AWARE (ver **Figura 14**) usado para capturar imágenes faciales biométricas de forma automática que estén acordes a los estándares biométricos de los Estados Unidos así como al estándar internacional ISO/IEC 19794-5 (23). El análisis de la calidad de las imágenes faciales se realiza en tiempo real en su auto-captura, y del mismo modo otro análisis de calidad en la post-captura. También se efectúa un procesamiento post-captura de dichas imágenes (rotar, escalar, cortar, optimizar). Se realizan operaciones con la cámara como el zoom, brillo, balance de color, además de tener soporte para cámaras personales, webcams y cámaras industriales de diferentes productores. Face Component es un producto web, de código propietario, siendo imposible de reutilizar al no brindar información alguna de los algoritmos utilizados en el proceso de validación de las imágenes de rostros (24).

Morpheus ICAO:

Morpheus ICAO es un producto desarrollado por la compañía Kee Square (ver **Figura 15**) el cual detecta la cara y las características faciales (ojos, boca, cejas, etc.) en imágenes de escala de grises de 8 bits o

imágenes RGB de 24 bits y automáticamente chequea los requerimientos fotográficos y geométricos acorde al estándar ISO/IEC 19794-5. Morpheus ICAO es un componente web, de código propietario, siendo imposible de reutilizar al no brindar información alguna de los algoritmos utilizados en el proceso de validación de las imágenes de rostros (25).

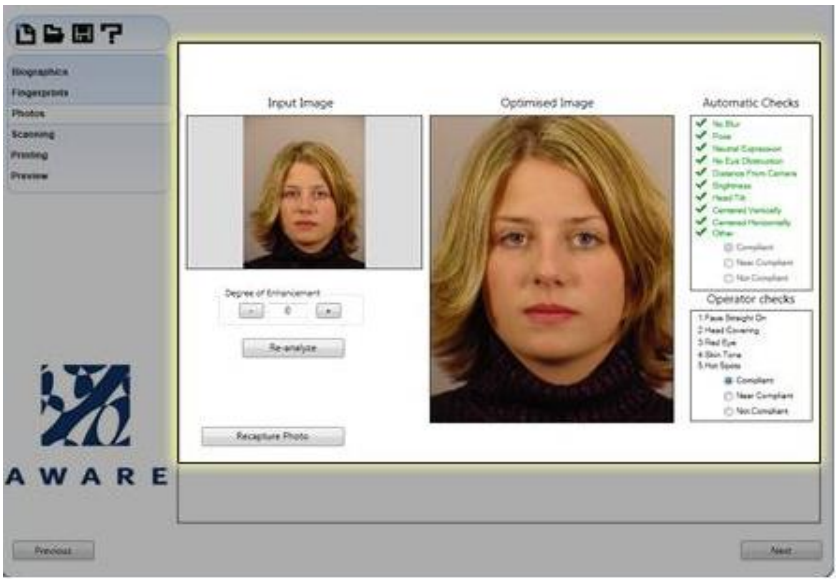


Figura 14: Face Component (23).

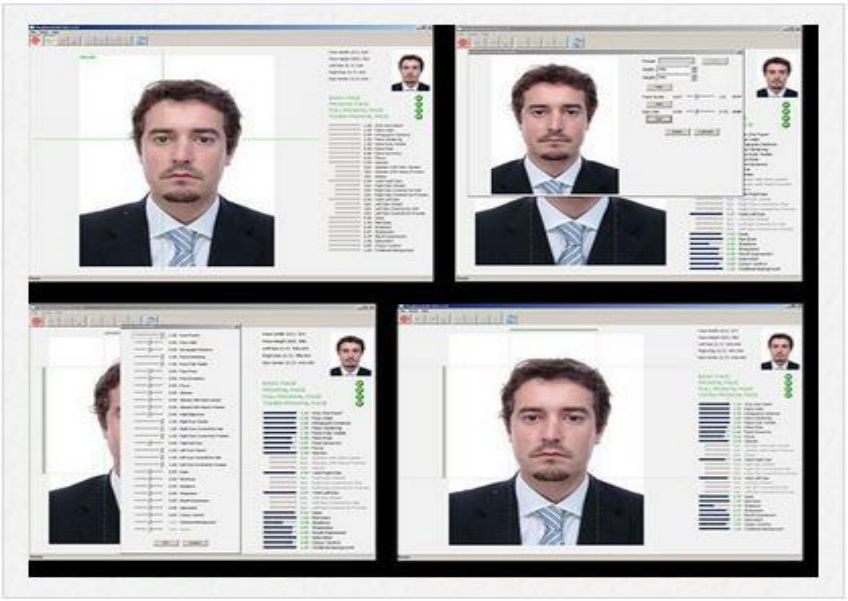


Figura 15: Morpheus ICAO (24).

- **Nacionales:**

Face Quality:

Face Quality es un componente desarrollado por las empresas cubanas CENATAV (Centro Nacional de Tecnologías Avanzadas) y DATYS (Desarrollo de Aplicaciones, Tecnologías y Sistemas), usado para evaluar si las imágenes de rostros cumplen los parámetros establecidos en la norma ISO/IEC 19794-5 y determinar si las mismas tienen valor identificativo y pueden ser utilizadas en documentos de identificación personal (2). Cabe destacar que este sistema no siempre presenta un funcionamiento óptimo en sus validaciones.

El Sistema Único de Identificación Nacional (SUIN), sistema que tiene como objetivo el control estricto de la identificación y registro de la población nacional o extranjera que reside temporal o permanentemente en el país, cuenta con una adaptación de Face Quality para la validación de la calidad de imágenes de rostros en su entidad, el mismo se muestra en la **Figura 16**:

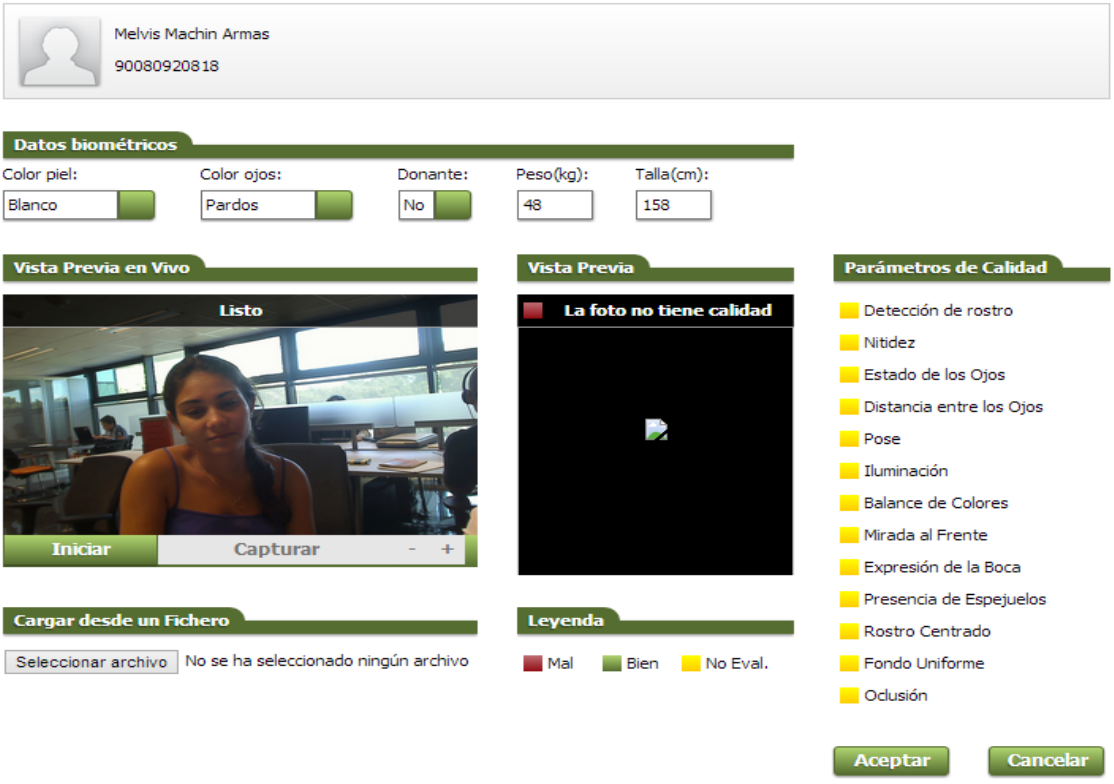


Figura 16: Versión de Face Quality utilizada por el SUIN.

Componente de validación y post-procesamiento de imágenes faciales para su uso en documentos oficiales*:* Componente desarrollado en el CISED con el objetivo de validar la calidad de las imágenes faciales a ser utilizadas en los documentos oficiales, tales como pasaportes e identificaciones, siguiendo las normas y regulaciones establecidas por el estándar de la ICAO (8). Cabe resaltar que este sistema no presenta un funcionamiento óptimo en sus validaciones.

Tomando en cuenta que la mayoría de los gobiernos exigen cada día, con mayor fuerza, el cumplimiento obligatorio de muchos de los parámetros de calidad de imágenes a la hora de tomar fotografías para lograr darle un uso real y eficaz al ser plasmadas en documentos de identificación personal, como las tarjetas de identidad, los pasaportes y las licencias de conducción (26), y en consecuencia, la comprobación automática del cumplimiento de estos requisitos se ha convertido en un tema de gran importancia dentro del desarrollo de sistemas biométricos de reconocimiento de rostros. Son varias las empresas importantes en el campo del reconocimiento de rostros que cuentan con sistemas comerciales para la captura y evaluación de las fotografías siguiendo estándares de calidad de imágenes internacionales (27). Sin embargo, la mayoría de estos sistemas evalúan sólo algunos de los parámetros en dependencia del propósito específico para el que hayan sido diseñados, algunos no presentan un funcionamiento óptimo, tienen elevados costos y no brindan información de los algoritmos utilizados para evaluar cada medida, por lo que son poco flexibles y no permiten una configuración variable de los parámetros utilizados en conjunto para determinar la calidad final de una imagen de rostro, por lo que se propone desarrollar una aplicación web que automatice el proceso de actualización de la imagen facial del ciudadano de la UCI, utilizando para esto un componente que efectúe el proceso de validación de la calidad de la imagen facial siguiendo el estándar ISO/IEC 19794-5.

1.5 Metodología de desarrollo de software a utilizar

Las metodologías de desarrollo de software surgen ante la necesidad de utilizar una serie de procedimientos, técnicas, herramientas y soporte documental a la hora de desarrollar un producto de software. Mientras se desarrolla se va indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben tener. Se detalla la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla.

Dichas metodologías pretenden guiar a los desarrolladores, pero los requisitos de un software a otro son tan variados y cambiantes, que ha dado lugar a que exista una gran variedad de metodologías para la creación de los mismos. Se podrían clasificar en dos grandes grupos (28):

- ✓ Las metodologías orientadas al control de los procesos, estableciendo rigurosamente las actividades a desarrollar, herramientas a utilizar y notaciones que se usarán. Estas metodologías son llamadas ***Metodologías Pesadas o Robustas.***
- ✓ Las metodologías orientadas a la interacción con el cliente y el desarrollo incremental del software, mostrando versiones parcialmente funcionales del software al cliente en intervalos cortos de tiempo, para que pueda evaluar y sugerir cambios en el producto según se va desarrollando. Estas son llamadas ***Metodologías ligeras/ágiles.***

La aplicación web a desarrollar pertenece al Departamento de Componentes del CISED. Este centro declara a la metodología ágil Programación Extrema (XP) como la metodología a utilizar en el desarrollo de sus productos informáticos, teniendo en cuenta el estado actual y el creciente avance de las mismas. Seguidamente se describe y fundamenta la selección de la metodología a seguir en el desarrollo de la aplicación.

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

1.5.1 Metodología XP:

El ciclo de desarrollo en XP es iterativo e incremental (ver **Figura 17**), en todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración (29).

Proceso XP

El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1 (29).

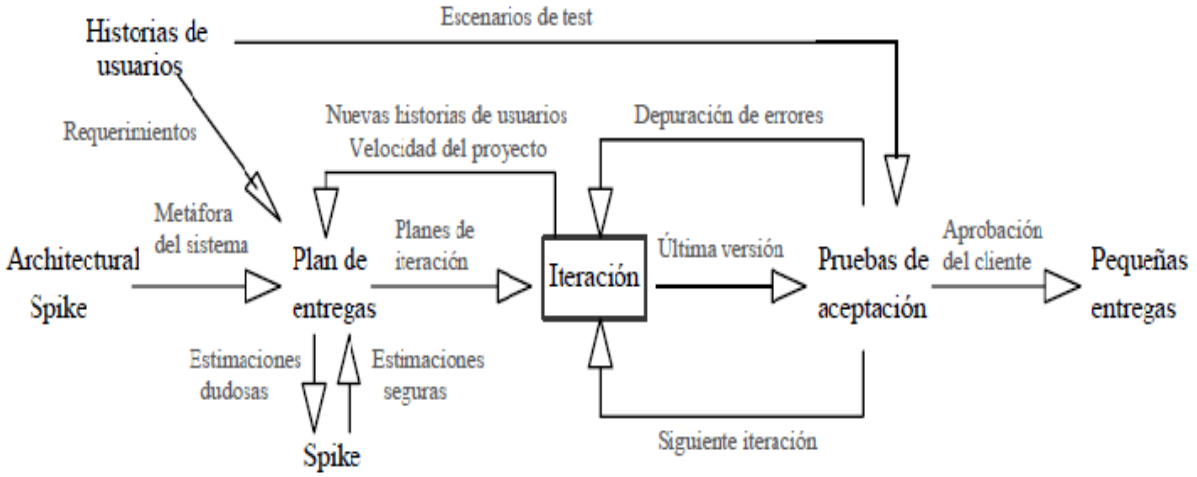


Figura 17: Flujo de trabajo en XP (29).

El ciclo de vida de XP consta de 6 fases:

- ✓ Fase I (Exploración): El equipo se familiariza con el proyecto y las tecnologías necesarias. Al mismo tiempo, el cliente desarrolla la mayor cantidad posible de historias de usuarios. La duración de esta fase depende del nivel de conocimientos del equipo sobre las tecnologías a utilizar.
- ✓ Fase II (Planificación de entrega): El cliente prioriza las historias de usuarios y el equipo hace una estimación de duración para las mismas. Cada entrega debe realizarse en unos pocos meses.
- ✓ Fase III (Iteraciones): Cada iteración incluye historias de usuarios que no hayan sido incluidas en la iteración anterior, corrige defectos encontrados y asume tareas anteriormente sin terminar.
- ✓ Fase IV (Producción): Al sistema se le realiza una serie de pruebas más rigurosas para ser trasladado al entorno del cliente.
- ✓ Fase V (Mantenimiento): Cada última versión se mantiene funcionando mientras el equipo desarrolla la siguiente. Esto requiere realizar tareas de soporte al cliente.
- ✓ Fase VI (Muerte del proyecto): Cuando el cliente no tiene más historias de usuarios que agregar y se comienza a refinar el rendimiento y otros aspectos de interés. Otra causa puede ser la cancelación del proyecto (29).

Algunas prácticas de desarrollo de XP

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas (29):

El juego de la planificación*:* Hay una comunicación frecuente entre el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.

Metáfora*:* El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).

Entregas pequeñas: Se desarrollan versiones incompletas del producto, pero con funcionalidades totalmente operativas que añaden valor al mismo.

Pruebas*:* La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.

Integración continua: Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

Diseño simple: Reduce al mínimo posible el código y la cantidad de funciones. Además trata de no duplicar lógica de negocio alguna.

Refactorización: Consiste en la revisión constante de código para reestructurarlo y hacerlo más entendible, eliminar duplicación de algoritmos y funciones.

Programación en parejas: Las principales ventajas de esta práctica son el poco tamaño del código, menor tasa de errores, solución rápida de los problemas y mejor diseño.

Propiedad colectiva del código: Cada programador puede cambiar cualquier fragmento de código en cualquier momento. Esto estimula la creatividad del equipo (29).

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí. La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya

habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

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

1.6 Herramientas y tecnologías a utilizar en el desarrollo de la aplicación.

1.6.1 Lenguaje para el modelado de objetos.

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

Toda metodología de desarrollo de software utiliza un lenguaje para el modelado de objetos, para la representación de sus diagramas y artefactos (30).

- ***Lenguaje Unificado de Modelado (UML)***

Para el desarrollo de la aplicación se tiene como propuesta, siguiendo la política del centro al cual pertenece el presente trabajo, utilizar como lenguaje de modelado el UML, el mismo es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software (31). Está consolidado como el lenguaje estándar en el análisis y diseño de sistemas de cómputo. Mediante UML es posible establecer una serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

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

De forma general las principales ventajas que brinda UML son:

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones).
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

Algunos beneficios de UML:

- Mejores tiempos totales de desarrollo (de 50% o más).
- Modelar sistemas (y no solo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y mínimos costos (31).

1.6.2 Herramienta de modelado.

Las herramientas de modelado de objetos, son fundamentales para el análisis del sistema. Hay varias herramientas creadas para el desarrollo de la Ingeniería de Software. Estas existen con el fin de desarrollar programas, utilizando técnicas de diseño y metodologías bien definidas, soportadas por herramientas automáticas.

Dentro de las herramientas claves en el desarrollo de aplicaciones informáticas se encuentran las herramientas de Ingeniería de Software Asistida por Ordenador (CASE por sus siglas en inglés), las cuales son las encargadas de ayudar en el ciclo de desarrollo, con el fin de aumentar la productividad y reducir el costo en términos de tiempo y dinero. Las herramientas CASE se han venido ampliando y desarrollando, algunos de sus ejemplos son: Microsoft Project, Rational Rose, JDeveloper, MagicDraw, Visual Paradigm, Microsoft Visio, Enterprise Architect, entre otros (31).

En la presente investigación se analizaron solo dos de ellas las cuales fueron:

- ***Visual Paradigm***

Visual Paradigm es una herramienta CASE multiplataforma que incluye UML y soporta el ciclo de vida completo del desarrollo de software. Permite el modelado visual del software con el paradigma orientado a objetos. Entre sus características más significativas se encuentran la ingeniería directa e inversa, la modelación de todos los tipos de diagramas de clases y la generación de documentación en varios formatos. Sus soluciones se enfocan en eliminar la complejidad, aumentando así la productividad y disminuyendo el tiempo de desarrollo de las aplicaciones informáticas lo cual posibilita desarrollar aplicaciones de calidad, rápidas y baratas (32).

- ***Rational Rose Enterprise***

Rational Rose es una herramienta de desarrollo basada en modelos, que se integra con las bases de datos y los IDE de las principales plataformas del sector. IBM Rational Rose Enterprise es uno de los productos más completos de la familia Rational Rose. Todos los productos de Rational Rose dan soporte a UML, pero no son compatibles con las mismas tecnologías de implementación. Rational Rose Enterprise es un entorno de modelado que permite generar código a partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic. Al igual que todos los productos de Rational Rose, ofrece un lenguaje de modelado común que agiliza la creación del software (33).

1.6.3 Lenguaje de programación.

Los lenguajes de programación son herramientas que permiten crear software. Los lenguajes de programación facilitan la tarea de programación, ya que disponen de formas adecuadas que permiten ser leídas y escritas por personas, a su vez resultan independientes del modelo de computador a utilizar (34).

- ***Visual 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 (35).

- ***Lenguaje Java***

Java es un lenguaje de programación portable sobre diferentes plataformas que fue desarrollado originalmente por la compañía *Sun Microsystems* (la cual fue adquirida por la compañía Oracle en 2010). Es un lenguaje de propósito general, orientado a objetos, basado en clases y concurrente, cuyo diseño está guiado a tener pocas dependencias de implementación y su sintaxis se deriva de C++. Integra librerías estándar para interfaces de usuario, objetos distribuidos, hilos de ejecución, XML, web, móviles, tv, entre otras.

Las aplicaciones escritas en Java hacen un uso intensivo de los recursos del ordenador, como son memoria y procesador. Además, la ejecución de estas aplicaciones es lenta cuando se realizan cálculos matemáticos complejos o el diseño contiene gran cantidad de elementos visuales (36).

- ***Lenguaje CSharp (C#)***

C# es un lenguaje de programación orientado al desarrollo y estandarizado por Microsoft como parte de su plataforma .NET, que más tarde fue aprobado como un estándar por la ECMA (Asociación Europea de Fabricantes de Computadoras) e ISO (37).

Aunque C# forma parte de la plataforma.NET, ésta es una interfaz de programación de aplicaciones (API), mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma.

Algunas de las características que presenta:

- Sencillez.
- Modernidad.
- Orientación a objetos.
- Orientación a componentes.
- Gestión automática de memoria.
- Seguridad de tipos.
- Instrucciones seguras.
- Sistema de tipos unificado.
- Extensibilidad de tipos básicos.
- Extensibilidad de operadores.
- Extensibilidad de modificadores.
- Es versionable.
- Eficiencia.
- Compatibilidad (37).

1.6.4 Entorno Integrado de Desarrollo.

Un Entorno Integrado de Desarrollo (IDE por sus siglas en inglés) es un sistema que facilita el trabajo del desarrollador de software, integrando sólidamente la edición orientada al lenguaje, la compilación o interpretación, la depuración, etc.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk. Algunos IDE's soportan múltiples lenguajes, tales como Eclipse o NetBeans, ambas basadas en Java o MonoDevelop, basado en C#. El soporte para lenguajes alternativos es a menudo proporcionado por plugins, que permiten ser instalados al mismo tiempo en el mismo IDE. Por ejemplo, Eclipse y NetBeans tiene plugins para C / C + +, Ada, Perl, Python, Ruby y PHP, entre otros lenguajes (38).

- ***Visual Studio .NET***

Visual Studio .NET es un entorno de desarrollo integrado para sistemas operativos Windows. Soporta varios lenguajes de programación tales como C++, C#, ASP.NET y Visual Basic .NET, aunque actualmente se han desarrollado las extensiones necesarias para muchos otros. Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET. Otra de las funcionalidades del entorno de desarrollo es que permite la ejecución de los programas paso a paso e incluir puntos de interrupción diversos, además de poder analizar el contenido de las variables a medida que se está ejecutando la aplicación (39).

- ***MonoDevelop***

MonoDevelop es un proyecto separado del proyecto Mono, pero muy unido a este. Es un entorno de desarrollo bastante completo, muy parecido al SharpDevelop y que solo funciona en entornos Unix/Linux. Permite la creación de proyectos diversos, ya sean simples o complejos, incluye plantillas y permite compilar diversos lenguajes, C#, C++, Visual Basic. Al igual que Visual Studio o SharpDevelop, dispone de una función de autocompletado que es útil a la hora de escribir el código, aunque no dispone de funciones de depuración como la de establecer puntos de ruptura o ejecutar paso a paso, por lo que en algunos casos encontrar un fallo específico puede ser tedioso (40).

1.6.5 Plataforma de desarrollo

Durante el desarrollo de la presente investigación se utilizará el **Framework.net en su versión 4.0** de Microsoft, con independencia de plataforma de hardware. El mismo permite un rápido desarrollo de aplicaciones. Ofrece una manera rápida y económica, que a la vez es segura y robusta, a la hora de desarrollar aplicaciones. Provee un extenso conjunto de soluciones predefinidas para necesidades generales de la programación de aplicaciones, y administra la ejecución de programas escritos específicamente con la plataforma. En términos simples un framework es un conjunto de clases base que pueden ser reutilizadas para la construcción de un nuevo software (41). A continuación se muestran algunas características que lo identifican:

- Completamente orientado a objetos.
- Multilenguaje.
- Modelo de programación único para todo tipo de aplicaciones y dispositivos de hardware.
- Se integra fácilmente con aplicaciones desarrolladas en plataformas Microsoft o en otras plataformas (41).

Dentro del Framework.net se utilizará **ASP.NET MVC 4 Framework** para el desarrollo de la aplicación web, el mismo es un framework de aplicaciones web que implementa el patrón modelo-vista-controlador (MVC). Basado en ASP.NET, permite a los desarrolladores de software construir una aplicación web como una composición de tres funciones: Modelo, Vista y Controlador. MVC es un patrón de arquitectura que ayuda a crear una separación lógica entre el modelo (la lógica de acceso a datos), la vista (la lógica de presentación) y el controlador (la lógica de negocio). Uno de los pilares básicos de ASP.NET MVC 4 es el concepto de enrutamiento (routing), lo que permite a las aplicaciones aceptar peticiones a URLs (Localizador de Recursos Uniforme) que no se corresponden con ficheros físicos en el servidor. De forma predeterminada, ASP.NET MVC 4 enruta las peticiones al controlador y a la vista adecuada en función de la URL (42).

1.6.6 Biblioteca de clases

Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos, permitiendo que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables. Ejecutables y bibliotecas hacen

referencias (llamadas enlaces) entre sí a través de un proceso conocido como enlace, que por lo general es realizado por un software denominado enlazador (43).

- ***OpenCV***

OpenCV es una biblioteca libre de versión artificial originalmente desarrollada por Intel. Desde que apareció su primera versión alfa en el mes de enero de 1999, se ha utilizado en infinidad de aplicaciones. Desde sistemas de seguridad con detección de movimiento, hasta aplicaciones de control de procesos donde se requiere un reconocimiento de objetos (43).

OpenCV es multiplataforma, existiendo versiones para GNU/Linux, Mac OS y Windows. Contiene más de 500 funciones que abarcan una gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras, visión estéreo y visión robótica. Una de las características de OpenCV es que sólo es compatible con los lenguajes de programación C++, C, y Python. Con el fin de usarlo en el lenguaje C #, debe utilizarse un contenedor (43).

- ***EmguCV***

EmguCV esencialmente proporciona una interfaz entre OpenCV y C#. Las llamadas a OpenCV se realizan a través EmguCV. EmguCV es una biblioteca de enlace dinámico, puede ser cargada y ejecutada en cualquier proceso en ejecución. EmguCV es una plataforma cruzada, una envoltura .Net para la biblioteca de procesamiento de imágenes OpenCV. Permite a las funciones de OpenCV ser llamadas desde lenguajes compatibles con .NET como C#, VB, VC++, Iron Python, etc. EmguCV está escrita enteramente en C# por lo que puede ser compilado sin problemas en la plataforma Mono, lo que quiere decir que el código generado es multiplataforma, puede ser usado tanto en Windows, como en Linux o en Mac OS X (44).

- ***ASM***

Los Modelos de Forma Activa (ASM por sus siglas en inglés) son modelos paramétricos deformables donde un modelo estadístico de la variación global de la forma del objeto es generado a partir de un conjunto de entrenamiento consistente en imágenes anotadas. Dicho modelo, conocido como Modelo de Distribución de Puntos (MDP), es utilizado posteriormente para ajustar una plantilla a instancias del objeto no presentes en el conjunto de entrenamiento. La forma del objeto es representada como un conjunto de

puntos (controlado por el modelo de la forma). Fue presentado por Tim Cootes y Chris Taylor en 1995 (45).

1.6.7 Gestor de base de datos

Un Sistema Gestor de Base de Datos (SGBD) permite procesar, describir, administrar y recuperar los datos almacenados en una base de datos. Estos sistemas brindan un conjunto de programas, procedimientos y lenguajes para consultar los datos y realizar acciones sobre ellos, garantizando además, la seguridad de los mismos (46).

- ***PostgreSQL***

PostgreSQL es un SGBD objeto-relacional disponible en un amplio rango de plataformas y que se distribuye bajo la licencia BSD (Berkeley Software Distribution) de software libre. Es mantenido por la organización PostgreSQL Global Development Team y por la colaboración de una amplia comunidad de usuarios y programadores denominada PGDG (PostgreSQL Global Development Group). Es en la actualidad el SGBD de código abierto más potente del mercado. Disímiles son las empresas y compañías que construyen sus productos de software utilizando este gestor, entre ellas Debian, Apple, Red Hat y Sun Microsystems (47).

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

Entre las principales características de PostgreSQL se encuentran el uso de procedimientos almacenados, el soporte de integridad referencial y manejo de distintos tipos de datos. Brinda una API flexible para programar en C/C++, Java y Python, entre otros (47).

- ***MySQL***

MySQL es un SGBD relacional que posee una licencia GNU GPL y una licencia comercial para aquellas empresas que quieran incorporarlo en productos privativos. Es ampliamente conocido y utilizado por su simplicidad y notable rendimiento y es además, muy confiable en términos de estabilidad. La empresa MySQL AB, además de ser la encargada de la venta de licencias privativas y poseer el copyright de la mayor parte del código, es la que ofrece el soporte. Según cifras del fabricante, en la actualidad existen

más de seis millones de copias de MySQL funcionando y cuenta con usuarios de renombre como Google, Yahoo, Nokia y Wikipedia (48).

Este gestor es multiplataforma y tiene un diseño multihilo, lo que le permite soportar una carga considerable de manera eficiente. Entre sus principales características se encuentran que es multiusuario, cuenta con APIs que posibilitan a aplicaciones escritas en diversos lenguajes de programación acceder a los bancos de datos MySQL, incluyendo, entre otros, C++, Java y Python; uso de procedimientos almacenados y vistas actualizables, gestión de usuarios y contraseñas. Soporta una gran cantidad de tipos de datos (48).

1.6.8 Sistema de Mapeo Objeto-Relacional

El mapeo objeto-relacional (más conocido por su nombre en inglés, Object-Relational mapping, o sus siglas O/RM, ORM, y O/R mapping) es una técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y el utilizado en una base de datos relacional, utilizando un motor de persistencia. En la práctica esto crea una base de datos orientada a objetos virtual, sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo) (49).

- ***NHibernate***

NHibernate es la conversión de Hibernate de lenguaje Java a C# para su integración en la plataforma .NET. Al usar NHibernate para el acceso a datos el desarrollador se asegura de que su aplicación es independiente en cuanto al motor de base de datos a utilizar en producción, pues NHibernate soporta los más habituales en el mercado: MySQL, PostgreSQL, Oracle, MS SQL Server, etc. Sólo se necesita cambiar una línea en el fichero de configuración para que se pueda utilizar una base de datos distinta.

NHibernate es software libre, distribuido bajo los términos de la LGPL (Licencia Pública General Menor de GNU) (50).

- ***Entity Framework***

Entity Framework es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos. Permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicas del dominio, como clientes y direcciones de cliente, sin tener que preocuparse por las tablas y columnas de la base de datos subyacente donde se almacenan estos datos. Con Entity

Framework, los desarrolladores pueden trabajar en un nivel mayor de abstracción cuando tratan con datos, y pueden crear y mantener aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales. Dado que Entity Framework es un componente de .NET Framework, las aplicaciones de Entity Framework se pueden ejecutar en cualquier equipo en el que esté instalado .NET Framework a partir de la versión 3.5 (51).

1.7 Selección del marco de trabajo.

La aplicación web a desarrollar pertenece al Departamento de Componentes del CISED. Este centro especifica algunas de las herramientas, metodologías y tecnologías a utilizar en el desarrollo de sus productos informáticos, teniendo en cuenta el estado actual y el creciente avance de estos elementos. Seguidamente se describe y fundamenta la selección del marco de trabajo para el desarrollo del sistema, según las líneas definidas por el centro y las decisiones propias del equipo de proyecto después del análisis de los mejores candidatos en cuanto a metodologías, herramientas y tecnologías.

No existen muchas diferencias entre Visual Paradigm y Rational Rose en cuanto a las funcionalidades básicas necesarias para la modelación del sistema. Sin embargo se seleccionó **Visual Paradigm en su versión 8.0** debido a que el equipo de proyecto está familiarizado con su empleo y ya se tiene una experiencia previa positiva, además por su sencillez y por ser una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software, es multiplataforma, ayudando a una rápida construcción de aplicaciones de calidad, mejores y a un menor coste ya que tiene la ventaja de ser software libre.

La implementación se realizará mediante el lenguaje de programación **C#**, el mismo es de propósito general y ofrece economía sintáctica, control de flujo y estructuras sencillas y un buen conjunto de operadores. Con C# se ahorra tiempo en la programación ya que presenta bibliotecas de clases muy completas y bien diseñadas y sobre todo se alinean a las necesidades del cliente de una aplicación que presente gran velocidad en la manipulación de los datos así como una interfaz que sea lo más rápida posible y que permita una fácil navegación.

Dentro de las bibliotecas de clases que existen se decidió trabajar con aquellas que facilitarían la detección del rostro en una imagen digital. OpenCV sólo es compatible con C++, C y Phyton. Con el fin de utilizar C# como lenguaje de programación se debe emplear un contenedor disponible y fácil de aplicar, al cual se le conoce como **EmguCV**, este esencialmente proporciona una interfaz entre OpenCV y C#.

EmguCV es una biblioteca libre, multiplataforma y brinda el algoritmo de AdaBoost creado por Viola y Jones con un 93.9 % de efectividad en la detección de rostro como vía para poder desarrollar esta fase del proceso de reconocimiento facial en el sistema. Además se escogió la biblioteca **ASM** para establecer puntos característicos en la imagen.

Debe señalarse que a pesar de ser Netbeans un fuerte candidato por su modularidad y extensibilidad, además de su condición de ser libre y gratuito, el sistema a desarrollar será ejecutado sobre la plataforma .Net, por lo que se necesita compatibilidad en este aspecto, por lo que el IDE seleccionado para el desarrollo es **Visual Studio 2012** el cual permite a los desarrolladores crear aplicaciones de alta calidad con gran rapidez, más seguras, confiables y administrables.

Se utilizará la plataforma de desarrollo **Framework.net en su versión 4.0**. Dentro del Framework.net se utilizará **ASP.NET MVC 4 Framework** para el desarrollo de la aplicación web, el mismo es un framework de aplicaciones web que implementa el patrón modelo-vista-controlador (MVC) permitiendo a los desarrolladores de software construir una aplicación web como una composición de tres funciones que ayuda a crear una separación lógica entre el modelo (la lógica de acceso a datos), la vista (la lógica de presentación) y el controlador (la lógica de negocio).

Entre MySQL y PostgreSQL, se decidió que este último era el que con mayor completitud ofrecía un balance entre la integridad, la eficiencia y las funcionalidades para la consulta de los datos. **PostgreSQL en su versión 9.2** presenta un buen funcionamiento con grandes cantidades de datos y una alta concurrencia de usuarios sin presentar incoherencias en la escritura de los datos. Además, es muy tolerante a fallos debido al uso de multiprocesos y no de multihilos. Estas son características únicas de PostgreSQL que lo hacen la opción más idónea para un sistema de tiempo de respuesta rápido como es el caso del sistema a desarrollar.

En el desarrollo de la aplicación se empleará el **ORM NHibernate en su versión 3.1** para el acceso a datos ya que con NHibernate se puede usar cualquier base de datos, incluyendo la que se usará en el desarrollo de la aplicación: PostgreSQL, mientras que Entity Framework sólo soporta de forma nativa SQL Server, por lo que se necesitan para trabajar con este ORM, el uso de componentes desarrollados por terceros.

1.8 Conclusiones parciales

Una vez analizados los diferentes sistemas con características afines que existen en Cuba y en el resto del mundo, se ha llegado a la conclusión de que la adquisición de algún sistema para la realización automática del proceso de validación de la imagen facial del ciudadano de la UCI implicaría un gran gasto de dinero para el país puesto que la biometría es una tecnología altamente costosa, la mayoría de estos sistemas son propietarios y no brindan información alguna de los algoritmos utilizados en el proceso de validación de las imágenes de rostros, siendo poco flexibles, además, los que se encuentran disponibles para su reutilización no presentan un funcionamiento óptimo en sus validaciones, por lo que se propone desarrollar una aplicación que permita facilitar el proceso de validación de las imágenes de perfiles de los ciudadanos de la UCI para su posterior actualización en el directorio. Los conceptos y otras temáticas tratadas en el capítulo, resultan fundamentales para lograr una mejor comprensión de la problemática y posibilitar el avance de la investigación. Además se definió que la metodología, herramientas y tecnologías que se utilizarán en el desarrollo de la aplicación, serán: la metodología XP, como lenguaje de modelado el Lenguaje Unificado de Modelado (UML), como herramienta de modelado Visual Paradigm en su versión 8.0, el lenguaje de programación será C# aparejado de la biblioteca de clases EmguCV y ASM, como entorno integrado de desarrollo Visual Studio 2012 conjuntamente con la plataforma de desarrollo Framework.net en su versión 4.0 y ASP.NET MVC 4, como gestor de base de datos se empleará PostgreSQL 9.2 y para el acceso a datos el ORM NHibernate 3.1.

Capítulo 2: Propuesta de solución

2.1 Introducción

En el presente capítulo se presenta la propuesta de solución al problema planteado utilizando la metodología de desarrollo de software Programación Extrema (XP). Se muestra la evolución de la solución durante las fases iniciales de Planificación y Diseño, y se presentan los diferentes artefactos generados en las mismas, los cuales constituyen punto de partida para la entrega final de la aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI.

2.2 Modelo del dominio

En la **Figura 18** se muestra el modelo del dominio[16] que conceptualiza los elementos principales del entorno en que se desarrolla la aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI, así como las relaciones entre ellos:

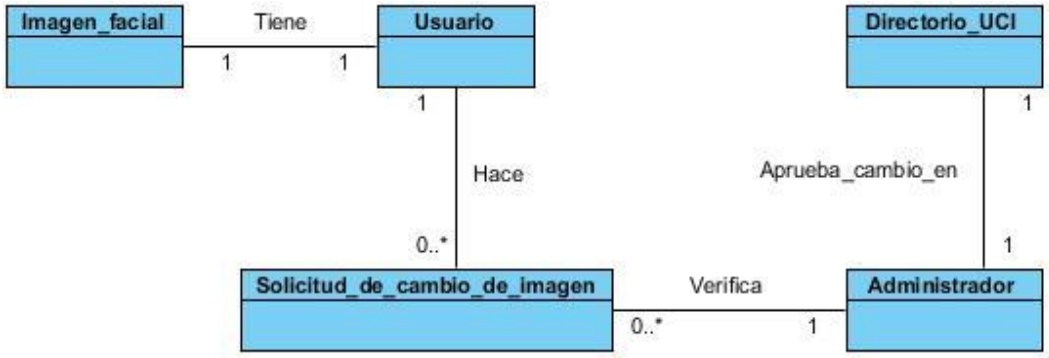


Figura 18: Modelo del dominio.

2.2.1 Glosario de conceptos del modelo del dominio

- **Imagen facial:** Retrato que identifica a cada persona con sus respectivos parámetros los cuales se verán inmersos en el principal proceso del sistema: la validación de la calidad de la imagen.
- **Usuario**: Persona que interactúa con el sistema.

[16] Modelo del dominio: Se les denomina además modelos conceptuales, modelo de objetos del dominio y modelos de objetos de análisis.

- **Solicitud de cambio de imagen:** Solicitud hecha por el usuario al sistema, para cambiar su foto de perfil en el directorio UCI. La solicitud solo se almacena si la imagen cumple con los parámetros de calidad requeridos.
- **Administrador:** Se encarga de determinar si la imagen cargada por el usuario le pertenece verdaderamente al mismo.
- **Directorio UCI:** Aplicación web donde se pueden visualizar los perfiles de usuarios.

2.3 Metáfora

Para describir el entorno del sistema a desarrollar es preciso crear una metáfora que detalle cómo debe funcionar el mismo. La elección de una metáfora permite mantener la coherencia de todos los componentes a implementar y elimina la selección rígida de una arquitectura inicial.
A continuación se define la metáfora global que guía el desarrollo del presente proyecto:

El sistema estará basado en una aplicación web mediante la cual los usuarios podrán iniciar las solicitudes de cambio de imagen de perfil en el directorio UCI (ver **Figura 19**). La aplicación contendrá un componente (dll)[17] de validación de imágenes y contará con una base de datos que almacenará las solicitudes.

El primer paso en el proceso de cambio de imagen es la interacción del usuario con la interfaz de la aplicación web, el mismo se autentica e inicia la solicitud cargando una imagen facial en el sistema. El servidor principal inicia el proceso de validación de la calidad de la imagen mediante el estándar ISO/IEC 19794-5:2011 a través del componente, el cual se encargará de procesar la imagen, extrayendo los rasgos faciales de la misma para realizar la verificación de los parámetros de calidad. Si la imagen facial cumple satisfactoriamente con los parámetros, se almacena la solicitud y pasa a esperar la aprobación por parte del administrador del sistema, una vez realizada dicha aprobación se procede a la actualización de la imagen de perfil de usuario en el directorio UCI.

[17] DLL: Biblioteca de enlace dinámico.

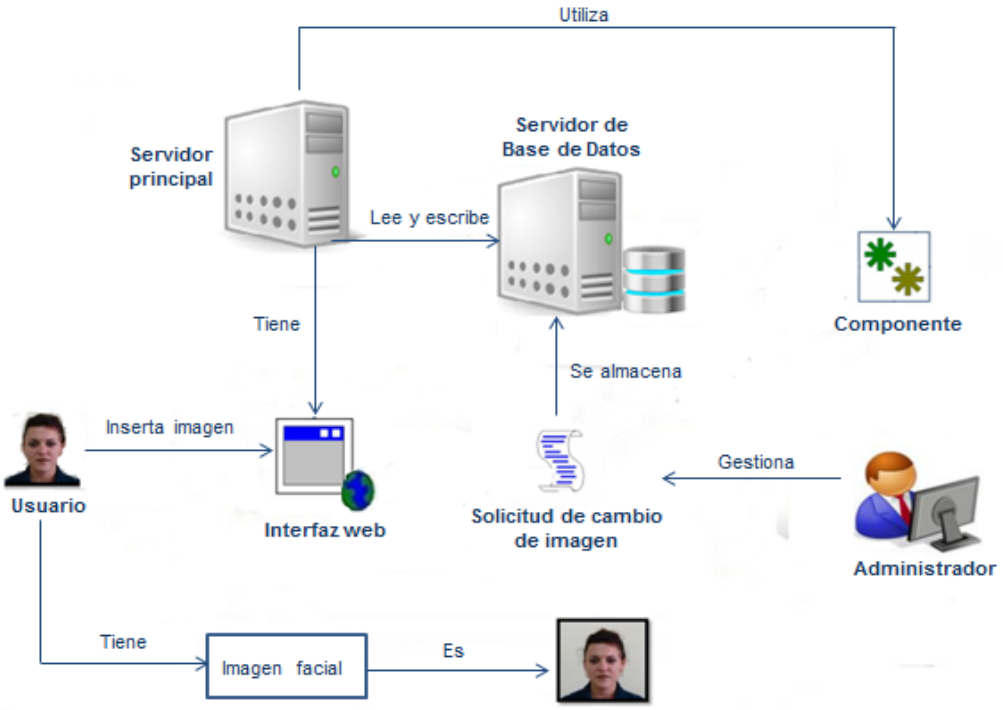


Figura 19: Caracterización del entorno.

2.4 Requisitos funcionales (RF)

Los requisitos funcionales son características del sistema que expresan funcionalidades, es decir, describen lo que el sistema debe hacer (52). A continuación se enumeran los requisitos funcionales de la aplicación a desarrollar, los cuales serán descritos con mayor detalle en las historias de usuario.
A continuación, se muestran las funcionalidades que el sistema debe cumplir:

RF1. Permitir cargar imagen.

RF2. Detectar rostro.

RF3. Verificar rostro centrado.

RF4. Verificar fondo uniforme.

RF5. Verificar ojos no rojos.

RF6. Verificar la no presencia de espejuelos.

RF7. Verificar balance de colores.

RF8. Validar nitidez.

RF9. Validar iluminación.

RF10. Verificar expresión de la boca.

RF11. Verificar mirada al frente.

RF12. Validar pose.

RF13. Cambiar imagen.

RF14. Gestionar control de acceso.

RF14.1. Permitir el acceso mediante LDAP UCI.

RF14.2. Permitir el acceso del administrador del sistema.

2.5 Requisitos no funcionales (RNF)

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido y confiable. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con toda la funcionalidad requerida, las propiedades no funcionales, como cuan usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación (52).

Para el desarrollo de la aplicación los requisitos no funcionales planteados son:

RNF1.Software

- Servidor: Se requiere que tenga instalado el sistema operativo Windows en cualquiera de sus versiones y el Framework .NET en su versión 4.0.
- Cliente: Se requiere que tenga algún navegador instalado.

RNF2. Requisitos de Hardware

Para los servidores:

- Se requiere tarjeta de red.
- 2 GB de RAM como mínimo.
- 3 GB de disco duro como mínimo.

- Procesador 2.66 GHz como mínimo.

Para las computadoras del cliente:

- Se requiere tengan tarjeta de red.
- 256 MB de memoria RAM como mínimo.
- Procesador 256 MHz como mínimo.

RNF3. Restricciones en el diseño y la implementación

- Estándares requeridos por la ISO/IEC 19794-5.
- Metodología de desarrollo de software: XP.
- Lenguaje para el modelado de objetos: UML.
- Plataforma de desarrollo: Framework.net 4.0.

RNF4. Rendimiento

- La eficiencia de la aplicación debe ser óptima en cuanto a la velocidad de procesamiento y tiempo de respuesta, para esto debe permitir validar las imágenes relacionadas a las solicitudes de cambio en un tiempo menor de 80 segundos para un máximo de 100 conexiones concurrentes.

RNF5. Requisitos de seguridad

- Proteger la información manejada por el sistema de accesos no autorizados.
- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel de usuario que esté activo.

RNF6. Apariencia o Interfaz interna

- Se utilizarán colores claros que ofrezcan suficiente contraste entre texto y fondo para no dificultar la lectura.
- Empleo de animaciones sencillas, permitiendo la rapidez de la aplicación.

RNF7. Portabilidad

- Se podrá acceder al sistema desde cualquier navegador.

2.6 Historias de usuario

En la metodología de desarrollo de software XP los requisitos que debe cumplir el mismo son especificados por los clientes en las denominadas historias de usuario. Estas historias son descompuestas en tareas de programación y asignadas a los programadores. A continuación se muestra una de las historias de usuario del sistema (ver Anexo 3 para consultar el resto de las historias de usuario):

Tabla 3. HU Permitir cargar imagen.

Historia de Usuario	
Número: 1	**Nombre:** Permitir cargar imagen
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Programador responsable: Alisbet Fernández Rojas	
Iteración Asignada: 3	
Descripción: La presente historia de usuario tiene como objetivo permitirle al usuario escoger la imagen que desea para su perfil en alguna carpeta de la pc y cargarla en el sistema.	
Observaciones: El usuario debe estar autenticado satisfactoriamente en el sistema. La imagen debe tener extensión JPEG, BMP, PNG, TIFF.	

2.7 Planificación

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

2.7.1 Plan de entrega

El plan de entrega constituye una herramienta fundamental, pues proporciona una estimación del tiempo de desarrollo que requieren las historias de usuario, fijándose de esta manera el tiempo que tardaría la implementación de cada una de ellas y definiéndose así finalmente las fechas exactas por iteración en que serán liberadas las versiones funcionales del producto (ver **Tabla 4**).

Tabla 4. Plan de entrega.

Entregable	Iteración	Fin Iteración
- Detección del rostro. - Validación de parámetros de documentos de identificación personal.	1	Marzo/2014
- Validación de parámetros biométricos.	2	Abril/2014
- Gestión de solicitudes de cambio de imagen. - Gestión del control de acceso al sistema.	3	Mayo/2014

2.7.2 Plan de iteraciones

El plan de iteraciones determina la duración de cada una de las iteraciones previstas para el desarrollo del sistema, lo que permite la estimación del tiempo requerido hasta la obtención del producto final del proyecto. Con este plan se logra una mayor organización y estructuración del trabajo al mostrar el orden en que serán implementadas cada una de las historias de usuario dentro de cada iteración

Una vez identificadas y descritas las HU las mismas serán seccionadas en tres iteraciones (ver **Tabla 6**).

Iteración 1: En esta iteración se realizará la detección del rostro de la persona en la imagen como punto de partida, y luego las historias de usuario relacionadas a los parámetros necesarios establecidos para los de documentos de identificación personal.

Iteración 2: En esta iteración se realizarán las historias de usuario relacionadas con la validación de los parámetros biométricos en las imágenes de rostros. Una vez concluida esta iteración el sistema se encontrará en un estado plenamente funcional.

Iteración 3: En esta iteración se realizarán las historias de usuario que son menos complejas y que no inciden críticamente en la lógica de la aplicación, en esta etapa se realizará la gestión de solicitudes de cambio de imagen y la gestión del control de acceso. Una vez concluida esta iteración la aplicación se encontrará completamente concluida y lista para su explotación.

2.7.2.1 Plan de duración de las iteraciones

El plan de duración de las iteraciones (ver **Tabla 6**) se encarga de mostrar las HU en el orden en que se implementarán en cada iteración así como la duración estimada de las mismas. La **tabla 5** muestra la estimación de esfuerzo por cada historia de usuario valorada en semanas[18].

Tabla 5. Estimación de esfuerzo por Historia de Usuario.

No.	Historia de usuario	Estimación(semanas)
1	Permitir cargar imagen.	0.4
2	Detectar rostro.	0.6
3	Verificar rostro centrado.	0.4
4	Verificar fondo uniforme.	1
5	Verificar ojos no rojos.	1
6	Verificar la no presencia de espejuelos.	1
7	Verificar balance de colores.	1
8	Validar nitidez.	1
9	Validar iluminación.	0.4
10	Verificar expresión de la boca.	1
11	Verificar mirada al frente.	1
12	Validar pose.	0.6

[18] Una semana de desarrollo está compuesta por 5 días (1 día de trabajo equivale a 0.2 semanas).

13	Cambiar imagen.	1
14	Gestionar control de acceso.	0.6

La **tabla 6** muestra el plan de duración de las iteraciones que permite el desarrollo del sistema:

Tabla 6. Plan de iteraciones.

Iteración	No. HU	Historia de Usuario	Semanas estimadas
Iteración 1	2 3 4 5 6	• Detectar rostro. • Verificar rostro centrado • Verificar fondo uniforme • Verificar ojos no rojos • Verificar la no presencia de espejuelos	4
Iteración 2	7 8 9 10 11 12	• Verificar balance de colores • Validar nitidez • Validar iluminación • Verificar expresión de la boca • Verificar mirada al frente • Validar pose	5
Iteración 3	1 13 14	• Permitir cargar imagen. • Cambiar imagen. • Gestionar control de acceso.	2

2.8 Diseño

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

2.8.1 Tarjetas CRC

Las tarjetas CRC (Clases, Responsabilidades y Colaboradores) son una técnica utilizada en XP, el uso de las mismas permite un diseño de software orientado a objetos. Estas tarjetas proponen una forma de trabajo para encontrar los objetos del dominio del sistema, sus responsabilidades y cómo colaboran con otros para realizar tareas. De esta manera el algoritmo para utilizar las tarjetas CRC es: primero la identificación de clases y asociaciones que participan del diseño del sistema, luego la obtención de las responsabilidades que debe cumplir cada clase y por último el establecimiento de cómo una clase colabora con otras para cumplir con sus responsabilidades.

La **Tabla 7** muestra la tarjeta CRC de la clase Checker (ver Anexo 4 para consultar el resto de tarjetas CRC).

Tabla 7. CRC. Checker.

Clase Checker	
Responsabilidades	**Colaboradores**
Extraer los principales rasgos faciales a tener en cuenta para las validaciones.	Extractor
Realizar las validaciones de los parámetros biométricos de la imagen facial.	BiometricReqValidator
Realizar las validaciones de los parámetros para documentos de identidad de la imagen facial.	IdDocsReqValidator

2.8.2 Diagrama de clases del diseño

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

En la **Figura 20** se muestra el diagrama de clases del diseño perteneciente al componente de validación de imágenes, que sería el corazón del sistema, ya que es aquí donde se encuentran las clases que rigen el proceso de validación de los parámetros de calidad en las imágenes.

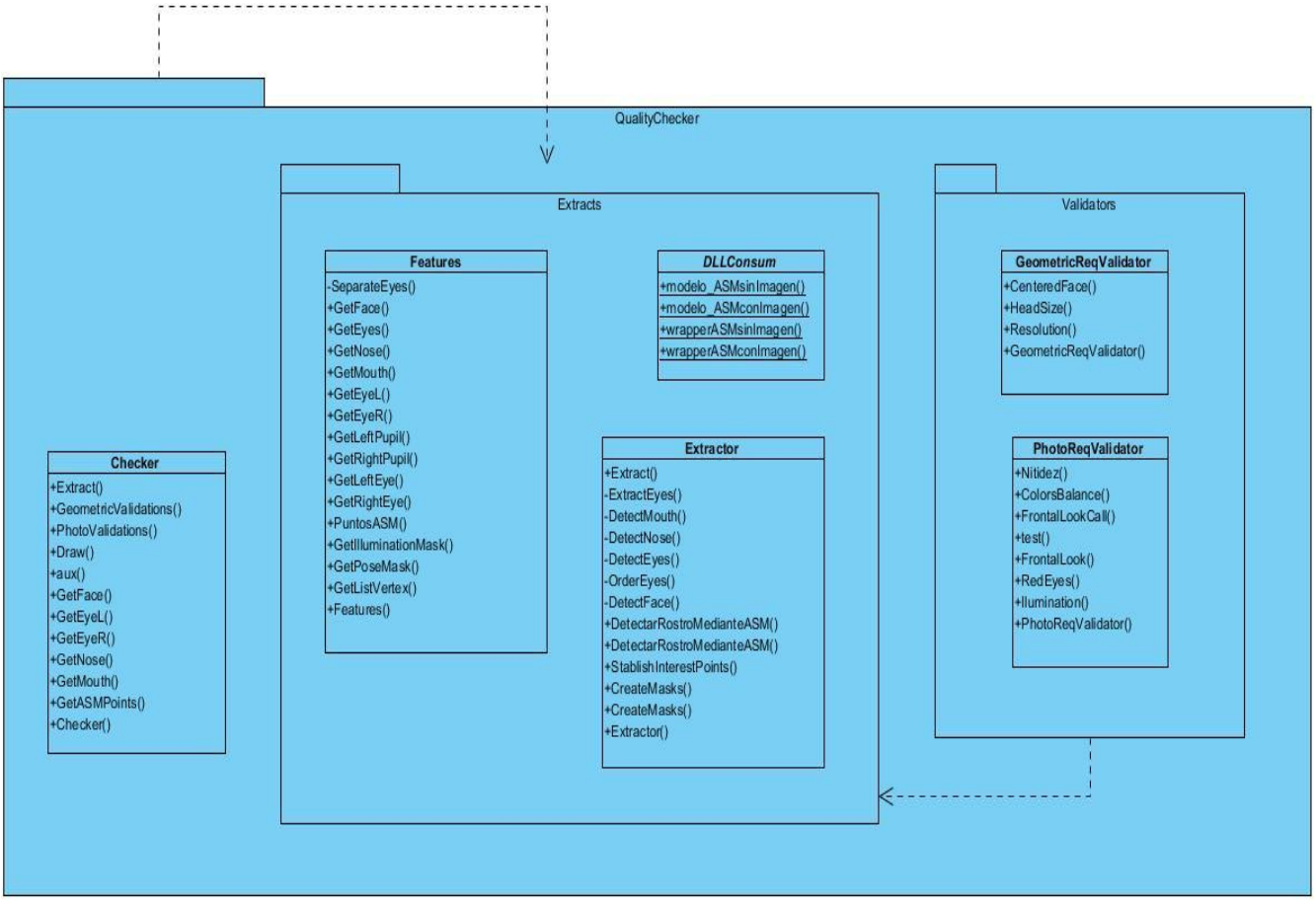


Figura 20: Diagrama de clases del diseño perteneciente al componente de validación de imágenes.

2.8.3 Modelo de datos

La persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo. Las clases entidad son las que contienen toda la información persistente manejada por el sistema, cuya estructura física y lógica describe el modelo de datos. El modelo de datos de la aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI (ver **Figura 21**) está diseñado por diferentes tablas, las cuales son:

1. *request*: Guarda las solicitudes de cambio de imagen.
2. *admin*: Guarda el usuario del administrador para identificarlo.
3. *validations:* Guarda los parámetros de validación de imágenes.
4. *image*: De esta tabla se puede obtener la imagen que identifica a cada persona.

A continuación se muestra el modelo de datos del sistema a desarrollar:

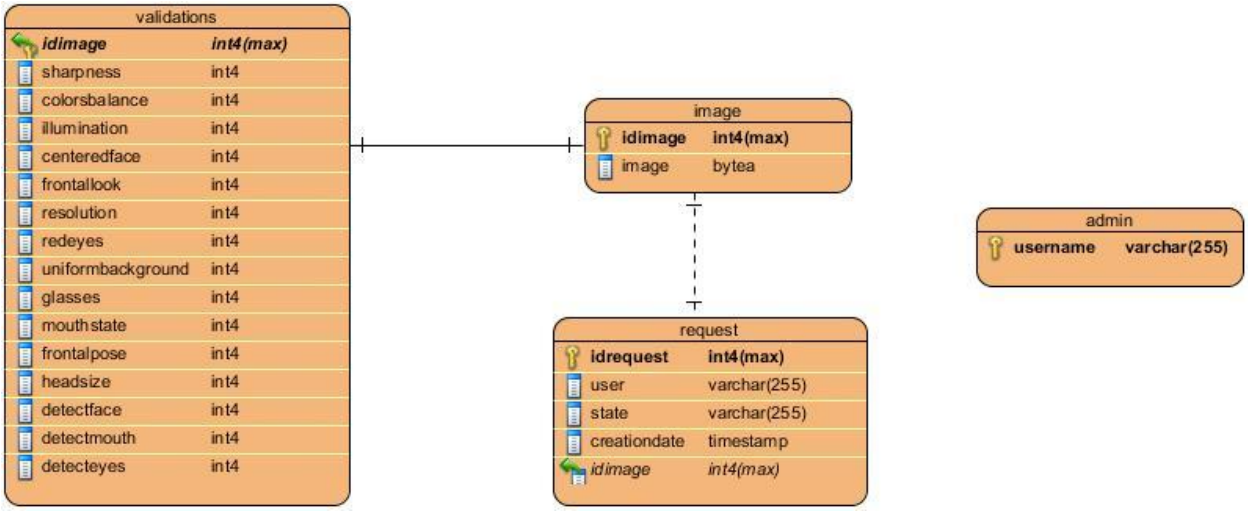


Figura 21: Diagrama Entidad-Relación del sistema.

2.8.4 Patrones de diseño

Los patrones de diseño expresan esquemas para solucionar problemas del mismo tipo que comúnmente se presentan en el diseño del software (31).

2.8.4.1 Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns, Patrones Generales de Software para Asignación de Responsabilidades) describen los principios fundamentales de la asignación de responsabilidades a objetos (31). Los patrones de este tipo usados son:

Experto en información es el principio básico de asignación de responsabilidades. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento). Este patrón se evidencia en la clase *Extractor* la cual es la encargada de extraer las características faciales de la imagen a procesar.

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. Este patrón se evidencia en la clase *Checker* la cual crea instancias de varias clases relacionadas con ella.

Alta Cohesión: Asignar una responsabilidad 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 mucha cohesión es

útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Ejemplo de clase de este tipo es *Features*.

Bajo Acoplamiento: Asignar una responsabilidad para mantener bajo acoplamiento. Las clases deben comunicarse con un número pequeño de clases tanto como sea posible. El empleo del patrón Experto favorece el bajo acoplamiento, por lo que un ejemplo de clase donde existe bajo acoplamiento es la clase *Extractor*.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa. Este patrón se evidencia en la clase *Checker* la cual es la encargada de controlar todo el flujo de procesamiento del componente de validación.

2.8.4.2 Patrones GoF

Los patrones GoF (Gang of Four, Banda de los Cuatro) describen las formas en las que pueden ser organizados los objetos para trabajar unos con otros, formando estructuras de mayor complejidad (31). Los patrones de este tipo usados son:

Creación: El objetivo de este patrón es el de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

1. Prototype (Prototipo): permite crear nuevos objetos clonándolos de una instancia ya existente, copiando el prototipo de esta clase.

Ejemplo:

```
private Extracts.Features features;
this.features = features;
```

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

Ejemplo:

```
private static ISessionFactory sessionFactory;
public void Initialize()
   {
      if (sessionFactory == null)
```

```
        sessionFactory = RepositoryFactory.InitializeSessionFactory;
}
```

2.9 Arquitectura del sistema

La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los principales componentes del mismo, la conducta de esos componentes según se percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema (53).

El patrón de arquitectura propuesto para estructurar la aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI es Modelo Vista Controlador (MVC) (ver **Figura 22**), el mismo es un patrón de diseño de arquitectura de software que ayuda a crear una separación lógica entre el modelo (la lógica de acceso a datos), la vista (la lógica de presentación) y el controlador (la lógica de negocio). El patrón MVC se ve frecuentemente en aplicaciones Web, donde la vista es la página HTML, el código que provee de datos dinámicos a la página es el controlador y el acceso a datos se realiza en el modelo. MVC permite reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos. Al incorporar el modelo de arquitectura MVC a un diseño, las piezas de un programa se pueden construir por separado y luego unirlas en tiempo de ejecución. Si se observa posteriormente que uno de los componentes funciona mal, puede reemplazarse sin que las otras piezas se vean afectadas (53).

El Modelo es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. Se encuentran las clases relacionadas con la persistencia final de los datos, en este caso las solicitudes de cambio de foto de perfil, así como las clases utilizadas para la construcción dinámica de vistas, ejemplo de clases: RequestModel, LoginModel.

La Vista es el objeto que maneja la presentación visual de los datos representados por el Modelo. Contiene las interfaces de la aplicación web. Ejemplo de clases: Account, Admin.

El Controlador contiene toda la lógica de funcionamiento de la aplicación. Se encarga de controlar el flujo de los datos dentro de la aplicación, realizar solicitudes, validar solicitudes, etc. Ejemplo de clases: AccountController, AdminController, GuestController, DllCaller.

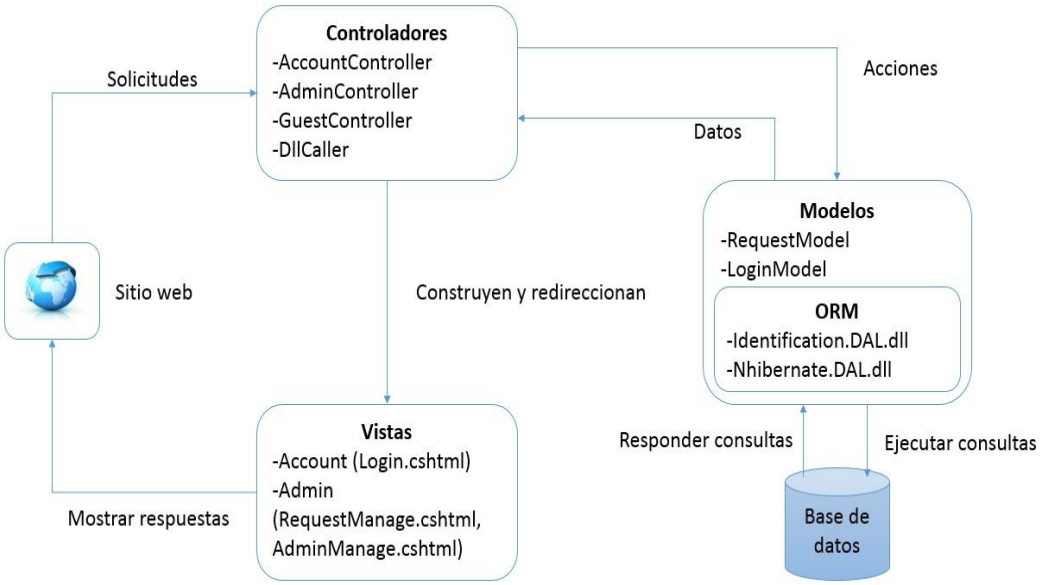


Figura 22: Arquitectura del sistema.

No obstante, aunque se utilice este patrón arquitectónico, se propone el uso de forma secundaria del estilo tubería y filtros, con el propósito de organizar la infraestructura de comunicación entre los sub-procesos que componen la etapa de extracción de características de la imagen facial y la validación de los parámetros de calidad en el componente a desarrollar.

Una tubería (pipeline) es una popular arquitectura que conecta componentes computacionales (filtros) a través de conectores (pipes), de modo que las computaciones se ejecutan a la manera de un flujo. Los datos se transportan a través de las tuberías entre los filtros, transformando gradualmente las entradas en salidas. Debido a su simplicidad y su facilidad para captar una funcionalidad, es una arquitectura mascota cada vez que se trata de demostrar ideas sobre la formalización del espacio de diseño arquitectónico, igual que el tipo de datos pila lo fue en las especificaciones algebraicas o en los tipos de datos abstractos (54).

Se utiliza dicho patrón puesto que proporciona una estructura para sistemas que procesan un flujo de datos como es el caso. El proceso puede ser descompuesto en etapas sucesivas, donde cada una es incremental respecto a la anterior. Cada paso del procesamiento está encapsulado en un filtro. El dato pasa a través de la tubería entre los filtros adyacentes.

Las ventajas que ofrece este estilo arquitectónico se enmarcan en:

- Es simple de entender e implementar, es posible implementar procesos complejos con editores gráficos de líneas de tuberías o con comandos de línea.
- Fuerza un procesamiento secuencial.
- Es fácil de envolver en una transacción atómica.
- Los filtros se pueden empaquetar y hacer paralelos o distribuidos (54).

El flujo propuesto tiene como entrada la imagen facial que se desea validar, y como salida la imagen validada, lista para su actualización en el directorio UCI. Los sub-procesos o filtros identificados son:

1. Extracción de rasgos.
2. Validación de parámetros.

En la **Figura 23** se muestran los resultados visuales de la aplicación de este estilo arquitectónico.

Figura 23: Estilo arquitectónico tuberías y filtros del sistema.

2.10 Conclusiones parciales

El análisis de los procesos involucrados en la validación de la calidad de imágenes permitió especificar las características del sistema, así como definir el modelo del dominio. A partir de este se identificaron los requisitos funcionales y no funcionales que fueron descritos en las historias de usuario. La solución propuesta implementa una arquitectura Modelo Vista Controlador para la aplicación web y Tuberías y Filtros para el desarrollo del componente de validación. El sistema cumple con los patrones de diseño GRASP y GoF que garantizan la organización de las clases y de la estructura de trabajo.

Capítulo 3: Implementación y pruebas

3.1 Introducción

En el presente capítulo se describe la implementación del software, fase donde finalmente se materializa el producto final y se cumple con los requisitos identificados al inicio de la investigación. Para lograr esto se generan los diagramas de componentes y de despliegue, donde se observan las dependencias lógicas entre los elementos de software y los nodos necesarios para el despliegue del sistema respectivamente. Además se describen las pruebas realizadas para validar el correcto funcionamiento de la solución.

3.2 Implementación

Una vez definidas las historias de usuario y concluido el diseño se pasa a la etapa de codificación de la solución propuesta. Los objetivos de la misma van destinados a desarrollar de forma iterativa e incremental un producto completo que esté preparado para la transición a su comunidad de usuarios, consiguiendo versiones útiles de forma rápida y práctica, que paulatinamente completen la planeación, diseño, desarrollo y prueba de toda la funcionalidad necesaria.

Durante la codificación la programación debe ser por parejas, con el objeto de crear código para cada historia de usuario en dependencia de lo concebido en el plan de iteraciones y las tareas ingenieriles. Esto da un mecanismo para la solución de problemas en tiempo real y también para el aseguramiento de la calidad en tiempo real. A medida que concluya cada iteración, el código desarrollado se integrará con el resto, puesto que esta estrategia de integración continua ayuda a evitar los problemas de compatibilidad e interfaces, y a descubrir a tiempo los errores.

3.2.1 Tareas de ingeniería

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

Las tareas de programación o ingenieriles a realizarse en la primera iteración se especifican en la **Tabla 8.** Las de las segunda y tercera iteraciones se pueden consultar en los anexos (ver Anexo 5 donde se detallan los elementos y descripción de cada una de estas tareas).

Tabla 8. Tareas de ingeniería de la primera iteración.

Iteración	Historia de usuario	Tareas
1	2. Detectar rostro.	• Detectar puntos característicos en la imagen. • Notificar al usuario los resultados de la validación.
	3. Verificar rostro centrado.	• Verificar que el rostro esté centrado de manera horizontal en la imagen. • Notificar al usuario los resultados de la validación.
	4. Verificar fondo uniforme.	• Verificar que el fondo tras la silueta de la persona sea uniforme. • Notificar al usuario los resultados de la validación.
	5. Verificar ojos no rojos.	• Verificar que los ojos de la persona en la imagen no estén rojos, permitiéndose solamente colores naturales. • Notificar al usuario los resultados de la validación.
	6. Verificar la no presencia de espejuelos.	• Detectar si el individuo porta espejuelos. • Notificar al usuario los resultados de la validación.

3.2.2 Estándar de codificación

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

3.2.2.1. Estilos para la capitalización de los identificadores

Para la capitalización de los identificadores se utilizaron los convenios:

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

La convención Pascal se empleó en los identificadores de las clases, métodos y nombres de ficheros. Mientras que la convención Camel es el estilo de los identificadores de las variables, atributos y parámetros.
En los identificadores de las variables se tuvo en cuenta emplear su significado y obviar las abreviaturas. Los identificadores de los métodos están en correspondencia a la función que realizan.

3.2.2.2. Comprensión y legibilidad del código

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

3.2.3 Diagrama de componentes

El diagrama de componentes representa, como indica su nombre, la división del software en partes más pequeñas denominadas componentes. Muestra la organización y las dependencias entre los distintos elementos de software, sean estos: código fuente, binario o ejecutable. Las dependencias en este diagrama indican que un componente utiliza los servicios ofrecidos por otro componente. A continuación se describe el diagrama de componentes del sistema (ver **Figura 24**).

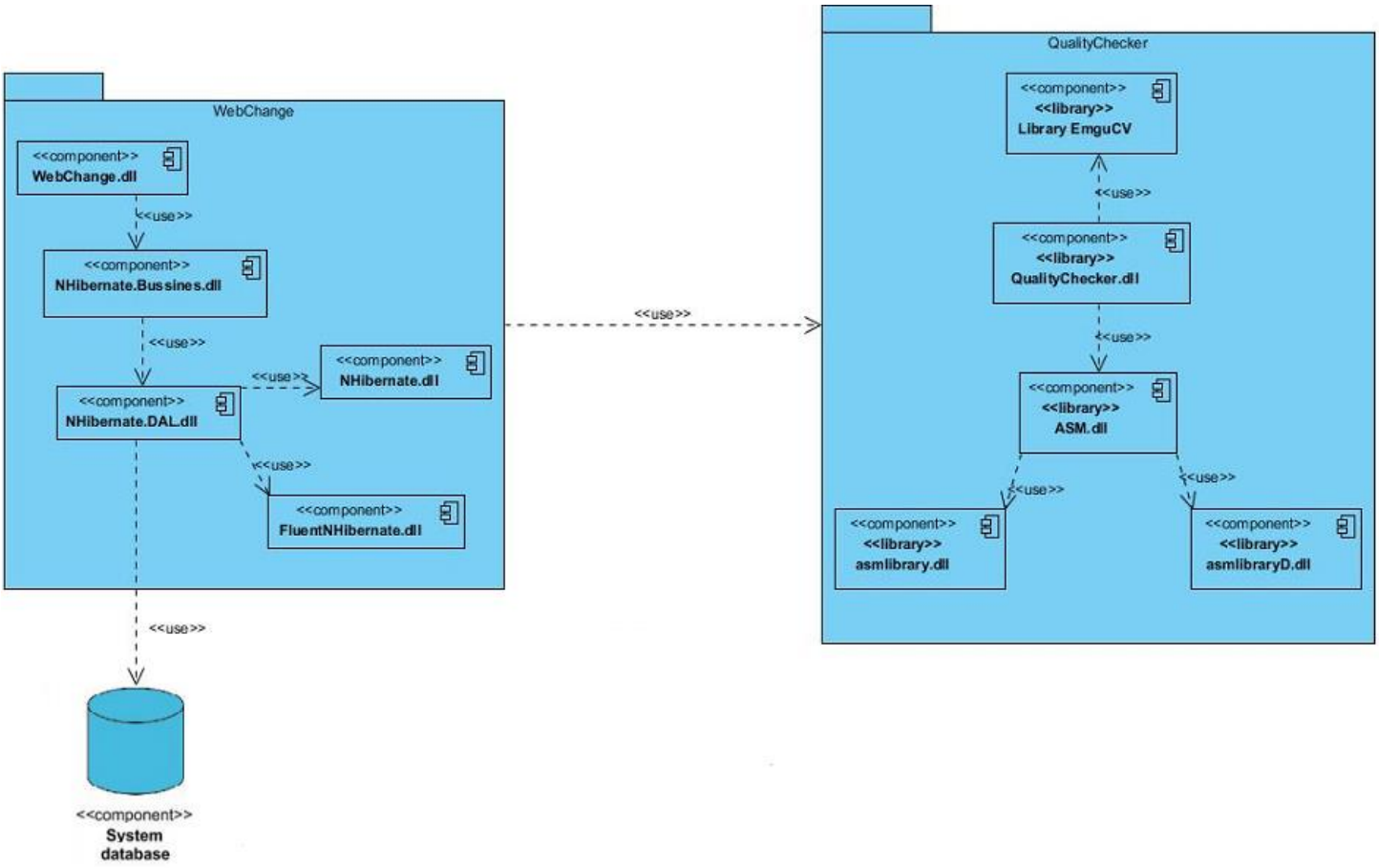


Figura 24: Diagrama de componentes del sistema.

3.2.3.1 Descripción del diagrama de componentes

Componente	Propósito
Library EmguCV	Componente encargado de la detección del rostro y procesamiento de imágenes.
QualityChecker	Componente en el que se encuentran las funcionalidades principales del software encargadas de realizar todo el proceso de validación de las imágenes faciales.
ASM	Componente responsable de la localización de los puntos característicos en el rostro.

asmlibrary	Componente responsable del procesamiento de la imagen para la localización de los puntos característicos en el rostro.
asmlibraryD	Componente responsable del procesamiento de la imagen para la localización de los puntos característicos en el rostro.
WebChange	Aplicación web que brinda la interfaz de usuario para interactuar con el componente de validación de la calidad de imágenes.
NHibernate.Bussines	Controla toda la lógica relacionada con el acceso y la persistencia de la información en la base de datos.
NHibernate.DAL	Componente encargado de proporcionar las clases necesarias para el mapeo de las entidades, así como clases intermedias del acceso a datos.
NHibernate	Sistema de Mapeo Objeto-Relacional (ORM) utilizado para la conversión de tipos de datos entre el lenguaje C# y PostgreSQL.
FluentNHibernate	Encargado de realizar el mapeo de las entidades a la base de datos.

3.2.4 Diagrama de despliegue

El diagrama de despliegue describe la distribución física de un software en un ambiente de producción o prueba. La **Figura 25** muestra el diagrama de despliegue correspondiente al sistema de validación de imágenes faciales a desarrollar. Este está compuesto por una computadora en la cual el usuario podrá realizar la búsqueda de la nueva imagen que desea y cargarla en el sistema para iniciar la solicitud de cambio de imagen de perfil. La computadora se conectará a través del protocolo HTTPS con la aplicación web, la cual contiene el componente de validación de imágenes faciales donde se realizará el procesamiento de la imagen, y mediante el protocolo TCP se realizará el intercambio de información con el servidor de base de datos.

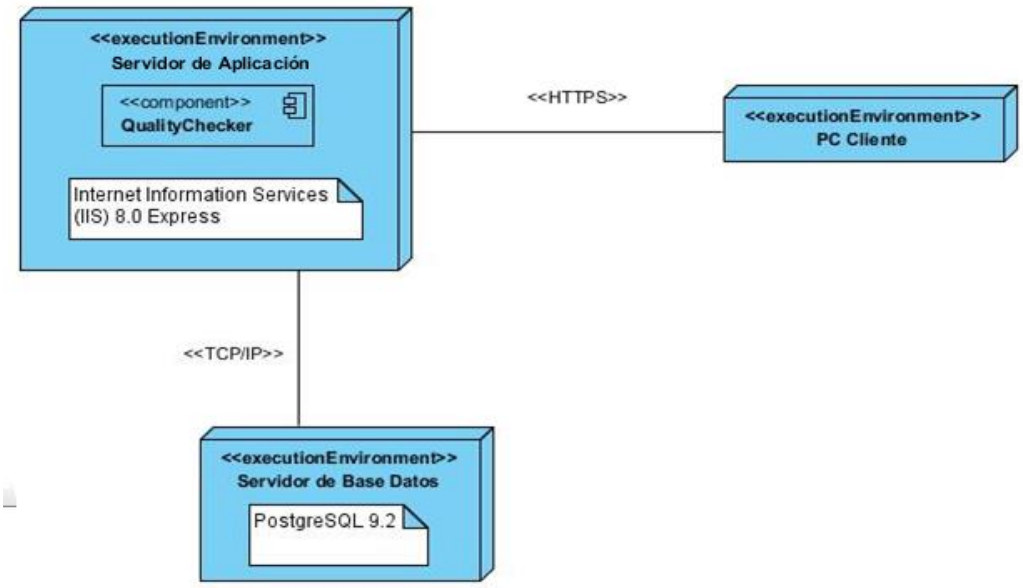


Figura 25: Diagrama de despliegue del sistema.

3.2.5 Interfaz gráfica

La interfaz de usuario es el medio a través del cual el usuario puede comunicarse con una máquina, un equipo o una computadora, por lo cual su diseño debe ser amigable y consistente, de esta forma pueden llegar a 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.

En el Anexo 6 se muestran las interfaces de la aplicación para que se tenga un mayor entendimiento del trabajo con la misma.

3.3 Pruebas

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

3.3.1 Pruebas unitarias

Las pruebas unitarias son escritas por los programadores antes de comenzar la codificación. El objetivo de estas pruebas es aislar pequeñas e individuales porciones del código para verificar que no tengan

errores. Para la realización de estas pruebas se utilizó un Add-in de Visual Studio llamado ReScharper (R#) que permite detectar errores y brinda soluciones instantáneas para corregirlos. Los resultados de estas pruebas fueron satisfactorios, verificándose de esta manera que con todas las funcionalidades se obtiene en cada caso la respuesta esperada, los mismos pueden consultarse en el Anexo 7: Resultados de pruebas unitarias. Al mismo tiempo las pruebas unitarias van enfocadas a los elementos más pequeños del software, siendo aplicables a funcionalidades para verificar que los flujos de control y de datos están cubiertos, y funcionan como se espera. La prueba de unidad siempre está orientada a caja blanca (52).

3.3.1.1 Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos, logrando como resultado que disminuya en un gran por ciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que (52):

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.

En la aplicación creada, se usó la técnica de pruebas de caja blanca del camino básico, puesto que permite obtener una medida de la complejidad lógica de un diseño y usarla como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de prueba derivados del camino básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

Existen 3 formas fundamentales de calcular la complejidad:

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

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

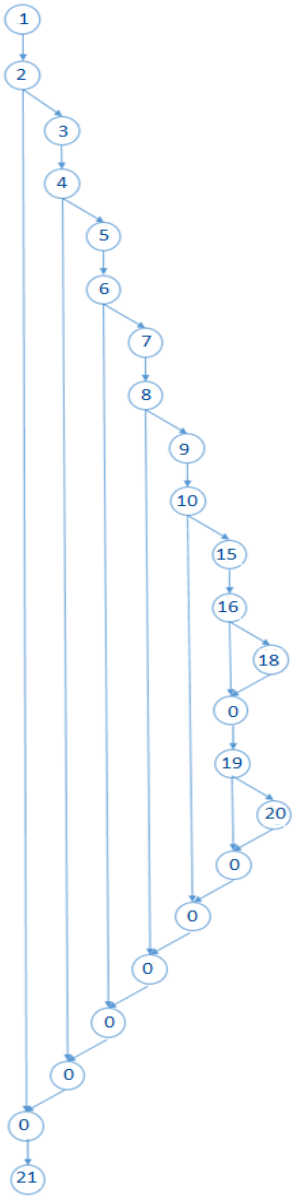
2. La complejidad ciclomática, V (G), también se define como: V (G) = P + 1

Dónde: P es el número de nodos predicado contenido en el grafo G.

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

Dónde: R es el número de regiones.

En el caso del algoritmo diseñado para la validación de los parámetros de calidad en las imágenes de rostros, cuyo código se muestra en los anexos (ver Anexo 8), la aplicación de la prueba del camino básico quedaría de la siguiente manera:



Cálculo de la complejidad ciclomática para la funcionalidad:

- Fórmula 1

V (G) = (A (Aristas) – N (Nodos)) + 2

V (G) = (29 – 23) + 2 = 8

- Fórmula 2

P {2, 4, 6, 8, 10, 16, 19}

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

V (G) = 7 + 1 = 8

- Fórmula 3

V (G) = R = 8

Caminos independientes obtenidos:

1. 1,2,21
2. 1,2,3,4,21
3. 1,2,3,4,5,6,21
4. 1,2,3,4,5,6,7,8,21
5. 1,2,3,4,5,6,7,8,9,10,21
6. 1,2,3,4,5,6,7,8,9,10,15,16,19,21
7. 1,2,3,4,5,6,7,8,9,10,15,16,18,19,21
8. 1,2,3,4,5,6,7,8,9,10,15,16,18,19,20,21

Figura 26: Grafo de flujo asociado.

3.3.2 Pruebas de aceptación

Las pruebas de aceptación se realizan para determinar el nivel de satisfacción del cliente final con el producto informático. Para que estas pruebas sean lo más objetivas posible, no deben ser diseñadas por los ingenieros de software que desarrollan el producto, sino por el cliente. Las pruebas de aceptación se realizan a partir de las historias de usuario, cada historia de usuario se convierte en un caso de prueba en el que el cliente especifica los puntos a probar, aunque una historia de usuario pudiera tener más de una prueba de aceptación, las que sean necesarias para garantizar la calidad del producto final con el cumplimiento de los requisitos especificados por el cliente. Una historia de usuario no se puede considerar terminada hasta tanto pase correctamente todas las pruebas de aceptación (55).

A continuación se muestra el caso de prueba de aceptación correspondiente a la historia de usuario "*Cambiar imagen*", por ser codificada en la última iteración y englobar el resultado de las implementadas en las iteraciones anteriores. Los casos de prueba de aceptación definidos por el cliente para el resto de las historias de usuario se especifican en los anexos (ver Anexo 9: Casos de prueba).

Tabla 9. Caso de prueba de aceptación 13.

	Caso de prueba de aceptación
Código de caso de prueba: CP13_ HU13	Nombre de la historia de usuario: Cambiar imagen.
Responsable de la prueba: Alisbet Fernández Rojas	
Descripción de la prueba: Prueba de funcionalidad para cambiar una imagen de perfil de usuario, validada previamente, en el directorio UCI.	
Condiciones de ejecución: La imagen debe haber sido validada satisfactoriamente.	
Entrada/Pasos de ejecución: • El administrador permite que la imagen se cambie.	
Resultado esperado: Notificación de cambio de imagen en el directorio UCI.	

Evaluación de la prueba: Prueba satisfactoria.

Los casos de prueba fueron aplicados por separado a cada imagen facial de una base de datos de 100 imágenes seleccionadas arbitrariamente, obteniendo los siguientes resultados:

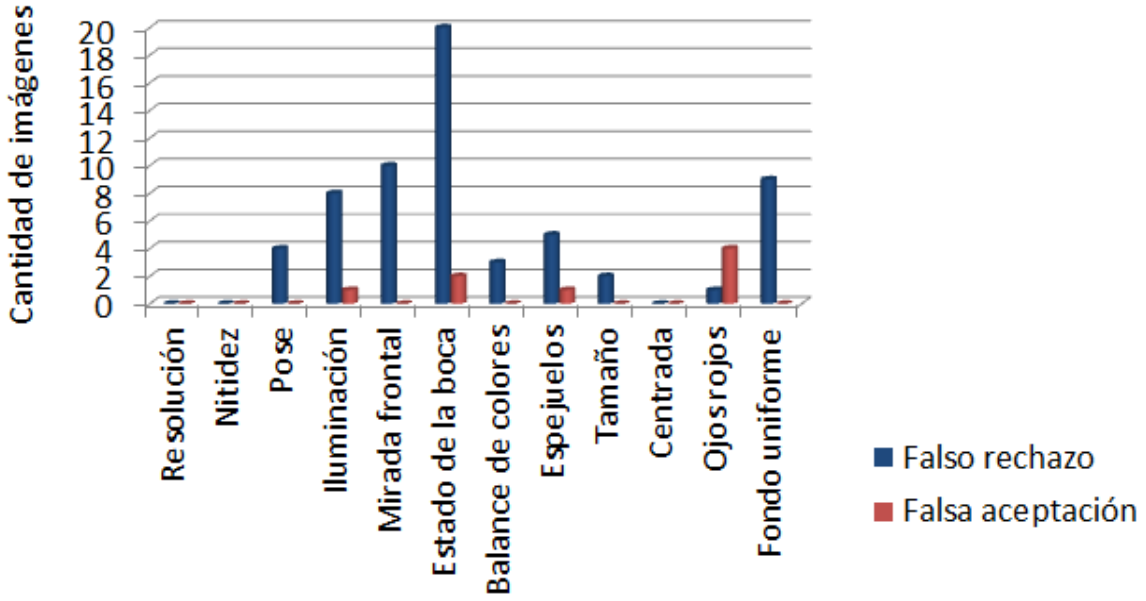


Figura 27: Resultados del proceso de validación.

Como se puede apreciar en la gráfica anterior, los algoritmos que presentan un buen funcionamiento representan el 75%, siendo el más crítico, en cuanto a falso rechazo, el del estado de la boca, y por la parte de falsa aceptación, el menos óptimo fue el de ojos rojos. De las 100 imágenes analizadas se reflejó que solamente 14 de éstas cumplían satisfactoriamente con todos los parámetros de calidad requeridos, representando un por ciento bajo, evidenciándose la mala calidad de las imágenes almacenadas hoy día en la BD de la UCI.

3.3.3 Pruebas de rendimiento

La aplicación desarrollada fue sometida a pruebas de rendimiento para medir el funcionamiento de la misma. Estas pruebas estuvieron enfocadas al análisis del comportamiento de los tiempos de respuesta en dependencia de condiciones variables como el crecimiento de los datos y la escalabilidad, esta última expresada en la cantidad de usuarios involucrados en las solicitudes.

Para medir el rendimiento de la aplicación se utilizó una base de datos con 100 peticiones, y como servidor de pruebas una PC con 2 GB de memoria RAM, procesador Core 2 Quad a 2.66 GHz y 3 GB libre de disco duro. En la **Figura 28** se pueden observar los tiempos de respuesta (en segundos) arrojados en el proceso de atención de las solicitudes para la actualización de la imagen, mostrando que al aumentar la cantidad de solicitudes, la aplicación ralentiza su funcionamiento con un incremento promedio de 7.3 segundos por cada 10 peticiones.

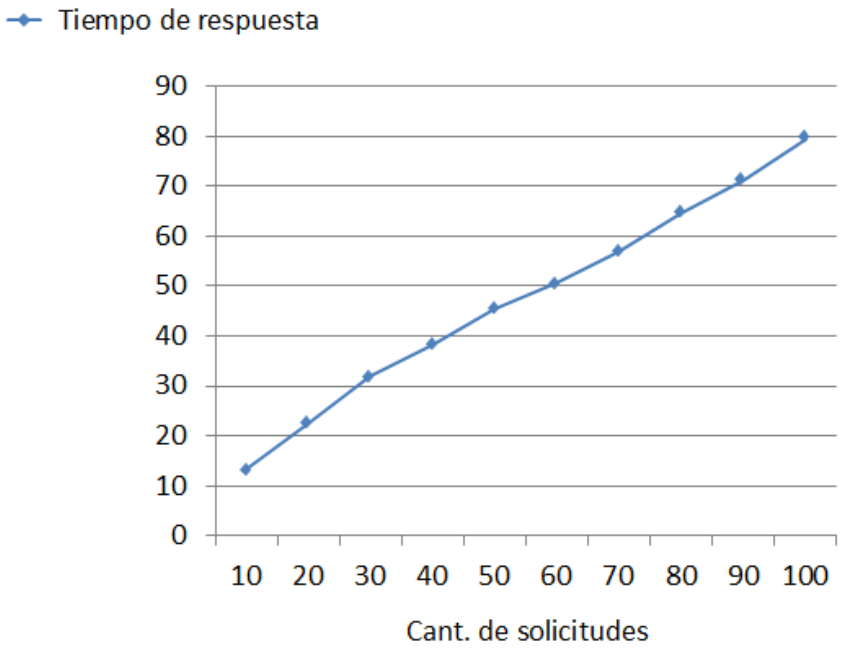


Figura 28: Tiempos de respuesta de la aplicación.

3.3.4 Resultados de las pruebas

Después de haber realizado las pruebas de aceptación de la aplicación a partir de los 14 casos de pruebas definidos por el cliente durante las tres iteraciones de codificaciones planificadas, se detectaron en cada una de éstas 9, 12 y 4 no conformidades respectivamente, lo que equivale a un total de 25 no conformidades, que revelaban errores en la codificación que en la mayoría de los casos no incidían significativamente en las respuestas esperadas. Estas dificultades fueron solucionadas en un corto plazo, antes de pasar a la iteración posterior. Una vez obtenido el producto final, se verificó el correcto funcionamiento de todo el sistema, en este momento ya no se detectaron nuevas no conformidades,

demostrando que, aunque exista una tasa de falso rechazo y una de falsa aceptación para algunas imágenes, se cumplieron correctamente todos los requisitos solicitados por el cliente.

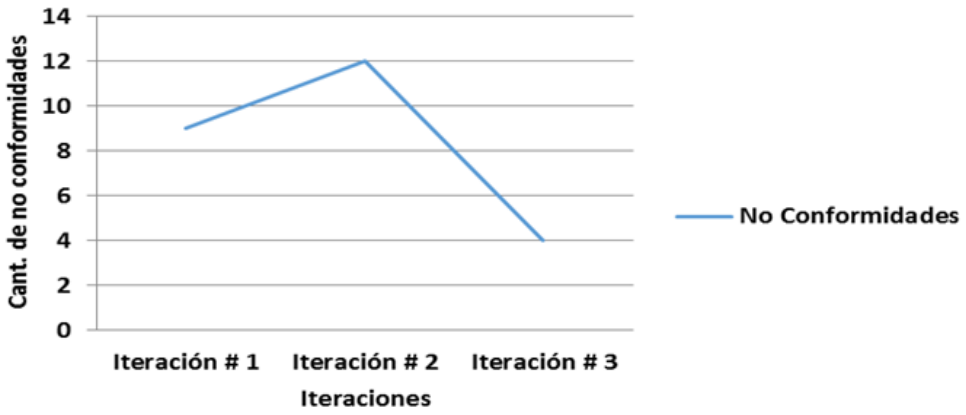


Figura 29: No conformidades detectadas durante las 3 iteraciones.

3.4 Conclusiones parciales

El estilo de codificación utilizado en la implementación del sistema permitió mantener una uniformidad en la codificación y mayor claridad a la hora de leer o agregar líneas de código. Al realizar los diagramas de componentes y despliegue se facilitó un mejor entendimiento del software desarrollado, ayudando así a su implementación. Las pruebas unitarias permitieron comprobar el buen funcionamiento de la aplicación asegurando una codificación legible y sin errores. Aunque tras la ejecución de varias iteraciones, cuyas entregas eran inmediatamente probadas, se detectaron varias no conformidades, éstas luego fueron corregidas en su totalidad y se validó el correcto funcionamiento del sistema en relación a los requisitos definidos por el cliente.

Conclusiones generales

Durante el desarrollo de la presente investigación se consultó bibliografía referente al tema de validación de imágenes, para tener un mayor conocimiento sobre la verificación de parámetros de calidad a través de estándares internacionales que garantizaran la misma. Con el estudio de los sistemas de validación de imágenes existentes alrededor del mundo, se alcanzó una mejor comprensión de la problemática planteada, aunque demostró que ninguna de las soluciones analizadas resolvía el problema de investigación planteado. El análisis de las diferentes metodologías y herramientas posibilitó la selección adecuada de la base tecnológica para el desarrollo del sistema. El establecimiento de los requisitos funcionales y los no funcionales permitió definir desde el principio las metas a alcanzar con la implementación del sistema para que el cliente quedara satisfecho. La definición de un estilo arquitectónico de tuberías y filtros posibilitó definir correctamente el flujo de datos en el proceso de validación. La realización de planes de iteración y de entrega ayudó a la estructuración y organización del trabajo en aras de lograr la obtención del producto final en tiempo. Definir las tareas de ingeniería y los patrones de diseño para la codificación permitió organizar la etapa de implementación ahorrando tiempo de desarrollo. El uso y escritura de pruebas unitarias sobre partes y funciones sensibles del sistema aseguró una codificación legible y sin errores. La realización de pruebas de aceptación permitió satisfacer las expectativas del cliente, verificando que todas las funcionalidades previstas para el sistema funcionaran. Finalmente, el sistema desarrollado soluciona la problemática que lo originó mediante el cumplimiento de los objetivos trazados.

Recomendaciones

Luego de concluida la investigación y de haberse obtenido finalmente la aplicación web para la validación y actualización de la imagen facial del ciudadano de la UCI en su primera versión, se recomienda para futuras versiones:

1. Agregar más funcionalidades en la validación de las imágenes para mayor rigor y efectividad en el proceso de reconocimiento facial.
2. Utilización del Sistema de Identificación de Personas[19] para que el proceso de aprobación se realice de forma automática, este sistema verificaría que la imagen a actualizar pertenece al individuo que realiza la solicitud.

[19] Sistema de Identificación de Personas: Tesis desarrollada en el CISED en el año 2013 por los autores Rafael Alberto Quiles Velázquez y Amelia Aguilera Reyes.

Bibliografía referenciada

1. **Sirovich, L. y Kirby, M.** *A Low-Dimensional Procedure for the Characterization of Human Faces.* s.l. : Journal of the Optical Society of America, 1987. 519/524.

2. **Méndez-Vázquez, H., Chang, L., Rizo-Rodríguez, D., y Morales-González, Annette.** *Evaluación de la calidad de las imágenes de rostros utilizadas para la identificación de las personas*: s.n., 2012.

3. **InterNational Committee for Information Technology Standards (INCITS) Secretariat**. *Face Recognition Format Data Interchange*: s.n., 2004.

4. **Arnauda Sequera, Luis Andres.** *Normas ISO 9000*: s.n., 2010.

5. **Pulido, Hernán Javier**. *Documento para la comunidad de la UCET, como soporte a su labor, en busca de la Excelencia*: s.n., *2004.*

6. **Stan Z. Li.** *Encyclopedia of Biometrics: I - Z., Volumen 1*: s.n., 2013.

7. **ICAO Pack**. [En línea] 18 de Junio de 2012. [Citado el: 12 de Enero de 2014.] http://www.aware.com/biometrics/icaopack.htm.

8. **Armada Viera, Daily**. *Componente de validación y post- procesamiento de imágenes faciales para su uso en documentos oficiales.* La Habana, UCI: s.n., 2013

9. **Goldestein, A.J., Harmon, L.D. y Lesk, A.B.** *Identification of human faces.* 1971. 748/760.

10. **Turkand, M. A. y Pentland, A. P.** *Face Recognition Using Eigenfaces.* s.l. : IEEE, 1991.

11. **Blazquez Pérez, Luis**. *Reconocimiento facial basado en puntos característicos: s.n.,* 2013.

12. **Ferrara, M., Franco, A., & Maltoni, D.** *Evaluating systems assessing face-image compliance with ICAO/ISO standards. Biometrics and Identity Management*: s.n., 2008.

13. **Viola, P. & Jones, M**. *Rapid object detection using a boosted cascade of simple features*. s.l. : IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Hawaii, USA, TR2004-043, 2004.

14. **Lienhart, R., Kuranov, A., & Pisarevsky, V.** *Empirical analysis of detection cascades of boosted classifiers for rapid object detection.* s.l. : Pattern Recognition. Lecture Notes in Computer Science, 2781, 297–304, 2003.

15. **D. Eduardo Navas Torres.** *Interfaz accesible persona-robot para un brazo articulado portátil*: s.n., Junio de 2013.

16. **Castrillón-Santana, M., Lorenzo-Navarro, J., Déniz-Suárez, O., Isern-González, J., & Falcón-Martel, A**.). *Multiple face detection at different resolutions for perceptual user interfaces*. s.l. : Pattern Recognition and Image Analysis. Lecture Notes in Computer Science, 3522, 767–808, 2005.

17. **Šochman, Jan; Matas, Jiří**. *Adaboost with Totally Corrective Updates for Fast Face Detection*. s.l. : ISBN 0-7695-2122-3, 2004.

18. **Chang, L., Rodés, I., Méndez, H., & Del Toro, E**. *Best-Shot Selection for Video Face Recognition Using FPGA.* s.l. : Progress in Pattern Recognition, Image Analysis and Applications. Lecture Notes in Computer Science, 5197, 543–550, 2008.

19. **Gao, X., Li, S.Z., Liu, R., & Zhang, P**. *Standardization of Face Image Sample Quality. Advances in Biometrics.* s.l. : Lecture Notes in Computer Science, 4642, 242–251, 2007.

20. **Ke, Y., Tang, X., & Jing, F**. *The Design of High-Level Features for Photo Quality Assessment*. s.l. : IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, USA, 419–426., 2006.

21. **Jing,Zhong & Mariani,Robert.** *Glasses Detection and Extraction by Deformable Contour*. s.l. : Multi-Modal Function Laboratory, Kent Ridge Digital Laboratory 21,Heng Mui Keng Terrace Singapore 119613, 2010.

22. **R. Fisher, S. Perkins, A. Walker y E. Wolfart.** [En línea] 2012. [Citado el: 20 de Marzo de 2014.] http://www.homepages.inf.ed.ac.uk.

23. **Gaubatz, M. & Ulichney, R**. *Automatic Red-Eye Detection and Correction*. s.l. : International Conference on Image Processing (ICIP 2002I), Rochester, New York, USA, I-804–I-807, 2002.

24. **Aware, Inc.** [En línea]. [Citado el: 12 de Febrero de 2014.] http://www.aware.com/biometrics/

25. **Kee Square, "Morpheus ICAO"**. [En línea] 10 de Mayo de 2013. [Citado el: 20 de Febrero de 2014.] http://www.keesquare.com/morpheus_icao_en.html

26. **Face Recognition Homepage: Vendors**. [En línea] 4 de Junio de 2012. [Citado el: 20 de Febrero de 2014.] http://www.face-rec.org/vendors/.

27. **Arévalo, V., González, J., & Ambrosio, G.** *OpenCV: La Librería Open Source de Visión Artificial, Linux Free Magazine*: s.n., 2005.

28. **Sommerville, Ian.** *Ingeniería del Software 7ma Edición.* Madrid : s.n., 2005.

29. **Beck, Kent.** *Extreme Programming Explained*: s.n., 2004.

30. **Falgueras, Benet Campderrich**. *Ingeniería de Software*: s.n., 2010.

31. **Larmn, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos*: s.n., 2004.

32. **Paradigm, Visual.** [En línea] [Citado el: 12 de Noviembre de 2014.] http://www.visual-paradigm.com..

33. **Mundo Libre.** mundolibre10. [En línea] Marzo de 2010. [Citado el: 10 de Noviembre de 2013.] http://mundolibre10.blogspot.com/2010/03/rational-rose-enterprise-edition-suite.html.

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

35. **Benítez, Ingrid.** Lenguaje C++. [En línea] 2012. [Citado el: 18 de Enero de 2014.] http://prezi.com/8llmfjk35wkq/lenguaje-dev-c/

36. **Java**. [En línea] [Citado el: 15 de Noviembre de 2013.] http://arantxa.ii.uam.es/~castells/docencia/poo/2-java-esp.pdf.

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

38. **González Barahona, Jesús M.** *Introducción al software libre.* s.l. : Madrid : GSyC/LibreSoft, 2003.

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

40. **MonoDevelop**. [En línea] 10 de Junio de 2012. [Citado el: 20 de Noviembre de 2013.] http://monodevelop.com/.

41. **García, Rey.** Slideshare. [En línea] 2 de Febrero de 2009. [Citado el: 20 de Octubre de 2013.] http://www.slideshare.net/soreygarcia/net-framework-981946.

42. **Otegem, Michiel Van**. *Interview with Scott Guthrie, creator of ASP.NET*: s.n., 2008

43. **Bradski, Gary y Kaebler, Arian.** *Learning OpenCV*: s.n., 2008.

44. **Agam, Gady.** *Introduction to programming with OpenCV:* s.n., 2006.

45. **Cootes, T.F, Taylor, C.J., Cooper, D.H. y J. Graham.** *Active shape models - their training and application*. s.l. : Computer Vision and Image Understanding: pp. 38–59. 1991.

46. **Asenjo, Jorge Sánchez.** *Apuntes Completos sobre Sistemas gestores de Bases de Datos*: s.n., 2009.

47. **The PostgreSQL Global Development Group**. *PostgreSQL: PostgreSQL Featured Users*: s.n., 2013.

48. **Oracle**. About MySQL. [En línea] 2013. [Citado el: 20 de Octubre de 2013.] http://www.mysql.com/about/

49. **Dentler, Jason**. *NHibernate Cookbook (1st ed.)*: s.n., 4 de Octubre de 2010.

50. **Kuaté, Pierre Henri; Harris, Tobin; Bauer, Christian y King, Gavin**. *NHibernate in Action*. : s.n., Febrero de 2009.

51. **Entity Framework**. *Información general de Entity Framework*. [En línea] 2014. [Citado el: 2 de Marzo de 2014.] http://msdn.microsoft.com/es-es/library/bb399567%28v=vs.110%29.aspx

52. **Pressman, Roger S.** *Software Engineering: A Practitioner's Approach.* s.l. : McGraw-Hill, 2010. ISBN 978-0-07-337-597-7.

53. **UNIVERSIDAD DE BUENOS AIRES.** *Introducción a la Arquitectura de Software.* s.l. : Buenos Aires, 2014.

54. **Reynoso , Carlos y Kicillof, Nicolás .** *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* s.l. : UNIVERSIDAD DE BUENOS AIRES, 2004.

55. **Joskowicz, Ing. José**. *Reglas y Prácticas en eXtreme Programming*. España: Doctorado de Ingeniería Telemática de la Universidad, 2008.

Bibliografía consultada

1. **Talavera Bustamante, Isneri y Rodríguez Hierrezuelo, Jorge Luis.** *Reconocimiento de patrones.* La Habana : s.n., 2008.

2. **Blázquez Pérez, Luis.** *Reconocimiento facial basado en puntos característicos de la cara en entornos no controlados.* Madrid : Universidad Autónoma de Madrid, 2013.

3. **Alvarez, J. & Chang, L**. *Análisis de la simetría facial como medida de estimación de la pose del rostro*. s.l. : Memorias del evento COMPUMAT, ISSN 1728-6042, Noviembre 2009.

4. **Delgado, Yisel y Pomar, Claudia.** *Algoritmo de detección facial para sistemas de autenticación biométrica.* La Habana : UCI, 2010.

5. **Niu, Z.** *Enhance ASMs Based on AdaBoost-Based Salient Landmarks Localization and Confidence-Constraint Shape Modeling.* Berlin: s.n., 2005.

6. **NetBeans**. [En línea]. [Citado el: 15 de Noviembre de 2013.] http://netbeans.org/index_es.html

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

8. **OACI.** [En línea]. [Citado el: 10 de Noviembre de 2013.] http://es.wikipedia.org/wiki/Organizaci%C3%B3n_de_Aviaci%C3%B3n_Civil_Internacional

9. **Sección de intereses de los Estados Unidos.** Solicitud de visa de no inmigrante. [En línea]. [Citado el: 15 de Enero de 2014.] http://spanish.havana.usint.gov/ds_160inf.html

10. **Zamuriano Sotés, Roberto F**. Desarrollo de aplicaciones Web en MicroSoft C# modeladas en UML. 2008.

11. **Rousset, C. & Coulon, P.Y.** *Frequential and color analysis for hair mask segmentation*. s.l. : 15th IEEE International Conference on Image Processing (ICIP 2008), San Diego, CA, USA, 2276–2279, 2008.

12. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson Addisson-Wesley, 2000.

13. **Hotelling, H.** *Analysis of a complex of statistical variables into principal components.* s.l. : Journal of Educational Psychology, 24(6), 417–441, 1933.

14. **Gonzalez, R.C. & Woods, R.E**. *Intensity Transformations and Spatial Filtering*. s.l. : Digital Image Processing 3rd Edition, (104–198), Upper Saddle River, N.J.: Prentice Hall, 2008.

Glosario de términos

Convolución: Operación matemática sobre dos funciones f y g, para producir una tercera función que representa una versión modificada de una de las funciones originales, dando el área donde se superponen las dos funciones como una función de la cantidad que es transformada una de las funciones originales.

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

GHz: Frecuencia de vibraciones eléctricas (ciclos) por segundo. Abreviado un Hz es igual a un ciclo por un segundo.1 GHz equivale 109 Hz.

Kernel: Núcleo, software que constituye la parte más importante del sistema operativo. Es el principal responsable de facilitar a los distintos programas acceso seguro al hardware de la computadora, o en forma básica, es el encargado de gestionar recursos, a través de servicios de llamada al sistema.

MB: El Megabyte es una unidad de cantidad de datos informáticos. Es múltiplo del byte u octeto, que equivale a 1024 bytes.

Multiplataforma: Se refiere a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas.

Píxel: Un píxel, plural píxeles (acrónimo del inglés picture element, "elemento de imagen") es la menor unidad homogénea en color que forma parte de una imagen digital.

Plataforma: Es un término de carácter genérico que designa normalmente una arquitectura de hardware, aunque también se usa en ocasiones para sistemas operativos o para el conjunto de ambos.

RAM: Es un tipo de memoria de ordenador a la que se puede acceder aleatoriamente; es decir, se puede acceder a cualquier byte de memoria sin acceder a los bytes precedentes. La memoria RAM es el tipo de memoria más común en ordenadores y otros dispositivos como impresoras.

Rasgos conductuales: son aquellos que se soportan sobre características de la conducta del ser humano.

Rasgos físicos: huella dactilar, geometría de la mano, características del iris, patrones vasculares de la retina, mano, etc.

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

Acrónimos	
A	**API:** Interfaz de programación de aplicaciones. **ASM:** Modelo de Forma Activa.
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. **DLL:** Biblioteca de enlace dinámico.
G	**GRASP:** Patrones generales de software para asignación de responsabilidades. **GoF:** Banda de los Cuatro.
H	**HSL:** Espacio de color de saturación, tono y luminancia. **HTML:** Lenguaje de marcado de hipertexto. **HU:** Historia de usuario.
I	**ICAO**: Organización de Aviación Civil Internacional. **IEC**: Comisión Electrotécnica Internacional. **IDE:** Entorno de desarrollo integrado, también conocido como entorno de diseño integrado o entorno de depuración integrada. **ISO:** Organización para la Estandarización Internacional.
L	**LDAP:** Protocolo Ligero de Acceso a Directorios. **LBP:** Patrones Binarios Locales.
M	**MVC:** Modelo Vista Controlador.
O	**ORM:** Mapeo Objeto-Relacional.
	PIN: Número de identificación personal, es una contraseña o clave numérica que se

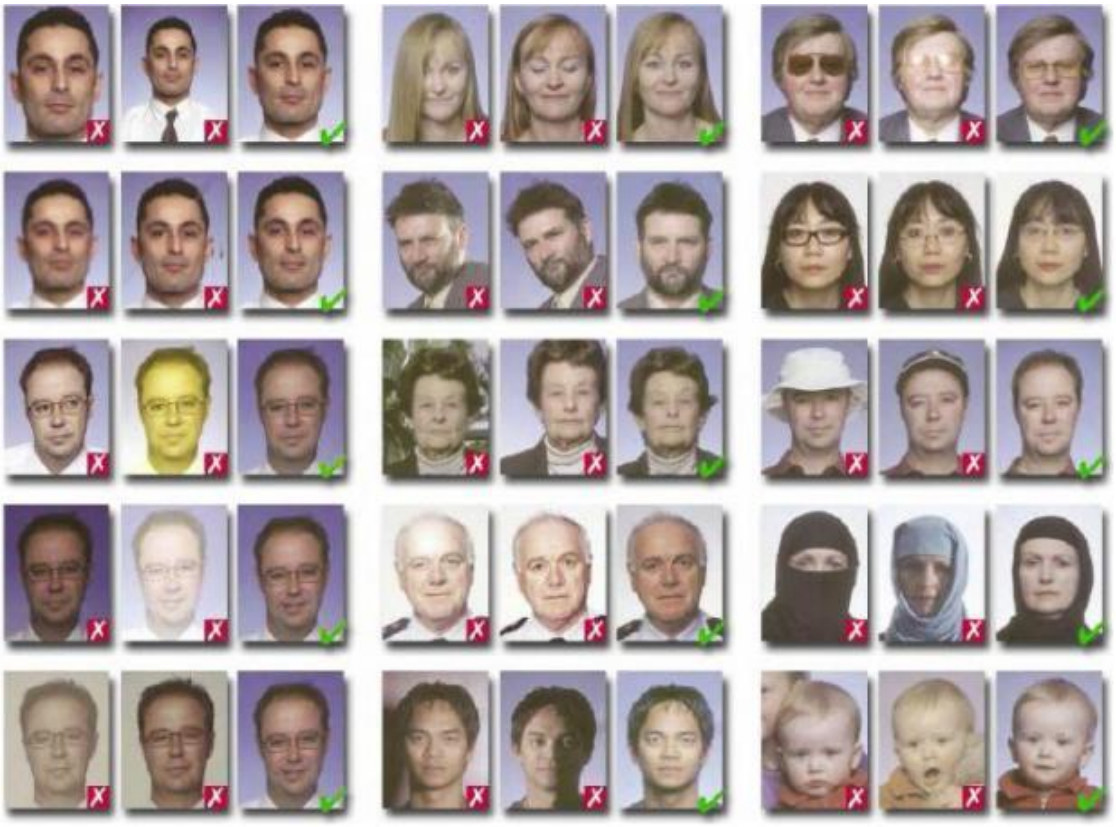
P	utiliza para acceder a cajeros automáticos o servicios de telefonía.
R	**RGB:** Modelo aditivo de colores rojo, verde, azul.
S	**SUIN:** Sistema Único de Identificación Nacional.
U	**UCI:** Universidad de las Ciencias Informáticas. **UML:** Lenguaje Unificado de Modelado.
X	**XP:** Programación Extrema.

Anexos

Anexo 1: Preguntas de la entrevista.

1. ¿Cuáles son los procedimientos fundamentales inmersos en la validación de imágenes?
2. ¿En qué consisten?
3. ¿Tiene conocimiento de algún software que realice dicho procedimiento?
4. ¿En qué lenguaje de programación está implementado el mismo?
5. ¿Qué tecnologías utiliza?
6. ¿Se apoya en alguna biblioteca de clases externa?
7. ¿Tiene conocimiento de alguna biblioteca de clases que facilite el trabajo con estos procedimientos?
8. ¿Cuál estándar considera más eficaz?
9. ¿Qué parámetros se toman en cuenta para verificar que las imágenes de rostros tienen calidad?
10. ¿De los parámetros a evaluar cuáles son los que se presentan deficientes con mayor reiteración?
11. ¿Cuáles son los dispositivos de captura de imágenes más utilizados?
12. ¿Cuáles de estos dispositivos generan imágenes de mayor calidad?
13. ¿Tiene algunas consideraciones a ser tomadas en cuenta a la hora de la implementación de este software?

Anexo 2: Ejemplo del cumplimiento de los requisitos especificados de la norma ISO/IEC 19794-5.



Anexo 3: Historias de usuario

Historia de Usuario	
Número: 13	**Nombre:** Cambiar imagen.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Alisbet Fernández Rojas	
Iteración Asignada: 3	
Descripción: La presente historia de usuario tiene como objetivo cambiar la imagen de perfil de usuario escogida en el directorio UCI.	

Observaciones: Esto se hace siempre y cuando la validación de los parámetros de calidad concebida por el sistema resulte satisfactoria.

Historia de Usuario	
Número: 14	**Nombre:** Gestionar control de acceso.
Usuario: Daniel Vega Torres	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 3	
Descripción: La presente historia de usuario tiene como objetivo gestionar el control de acceso a la aplicación según el rol del usuario que desea acceder.	
Observaciones: Si el sistema no identifica el usuario este mostrará un mensaje comunicando que sus identificadores no son válidos.	

- **Historias de Usuario de los parámetros de validación:**

Historia de Usuario	
Número: 2	**Nombre:** Detectar rostro.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 1	

Descripción: La presente historia de usuario tiene como objetivo la detección de los puntos característicos del rostro en la imagen.
Observaciones: Este se paso es previo a la validación de parámetros de calidad de la imagen.

Historia de Usuario	
Número: 3	**Nombre:** Verificar rostro centrado.
Usuario: Todos	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Bajo
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 1	
Descripción: La presente historia de usuario tiene como objetivo verificar que el rostro esté centrado de manera horizontal.	
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.	

Historia de Usuario	
Número: 4	**Nombre:** Verificar fondo uniforme.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 1	

Descripción: La presente historia de usuario tiene como objetivo verificar que el fondo tras la silueta de la persona sea uniforme.
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.

Historia de Usuario	
Número: 5	**Nombre:** Verificar ojos no rojos.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 1	
Descripción: La presente historia de usuario tiene como objetivo verificar que no se admitan fotos con ojos rojos. Los colores deben ser naturales.	
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.	

Historia de Usuario	
Número: 6	**Nombre:** Verificar la no presencia de espejuelos.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 1	

Descripción: La presente historia de usuario tiene como objetivo detectar si el usuario porta espejuelos o no en la imagen facial cargada en el sistema.
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.

Historia de Usuario	
Número: 7	**Nombre:** Verificar balance de colores.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 2	
Descripción: La presente historia de usuario tiene como objetivo verificar que los colores sean naturales, no saturados y con un contraste adecuado.	
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.	

Historia de Usuario	
Número: 8	**Nombre:** Validar nitidez.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 2	

Descripción: La presente historia de usuario tiene como objetivo verificar que las imágenes no estén borrosas o desenfocadas.
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.

Historia de Usuario	
Número: 9	**Nombre:** Validar iluminación.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Bajo
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 2	
Descripción: La presente historia de usuario tiene como objetivo verificar que la luz esté distribuida uniformemente en el rostro, el cual no debe estar afectado por sombras, ni regiones brillantes.	
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.	

Historia de Usuario	
Número: 10	**Nombre:** Verificar expresión de la boca.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 2	

Descripción: La presente historia de usuario tiene como objetivo verificar que la boca esté cerrada y con una expresión neutral.
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.

Historia de Usuario	
Número: 11	**Nombre:** Verificar mirada al frente.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 2	
Descripción: La presente historia de usuario tiene como objetivo verificar que los ojos estén mirando de frente a la cámara.	
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.	

Historia de Usuario	
Número: 12	**Nombre:** Validar pose.
Usuario: Todos	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Medio
Programador responsable: Daniel Vega Torres	
Iteración Asignada: 2	

Descripción: La presente historia de usuario tiene como objetivo verificar que la imagen esté lo más frontal posible, o sea que al trazar una línea horizontal imaginaria al centro de los ojos resulte paralela al borde superior de la imagen, verificando así de esta manera una buena posición del titular con respecto a la cámara.
Observaciones: Esto se hace como un paso en la validación de los parámetros de calidad de la imagen.

Anexo 4: Tarjetas CRC

Clase BiometricReqValidator	
Responsabilidades	**Colaboradores**
Validar la calidad de la nitidez de la imagen.	Features
Validar que la imagen cumpla con el tamaño de resolución mínima.	Features
Validar el balance de colores en la imagen.	Features
Validar que la mirada se encuentre orientada de manera frontal.	Features
Validar que la iluminación se encuentre distribuida de manera uniforme en el rostro.	Features
Validar que el rostro presente una pose frontal.	Features
Validar que el fondo de la imagen sea uniforme.	Features
Validar que la boca esté cerrada y con una expresión neutral.	Features

Clase IdDocsReqValidator	
Responsabilidades	**Colaboradores**
Validar el rostro cumpla con el tamaño mínimo requerido.	Features
Validar que el rostro se encuentre centrado en la imagen.	Features

Validar que la imagen no presente ojos rojos.	Features
Detectar la presencia de espejuelos en la imagen.	Features

Clase Extractor	
Responsabilidades	**Colaboradores**
Detectar rostro.	---
Detectar ojos.	---
Detectar boca.	---
Extraer puntos característicos del rostro.	ASMConsum
Crear máscaras para la validación de la pose frontal y la iluminación.	---

Clase ASMConsum	
Responsabilidades	**Colaboradores**
Obtener puntos característicos del rostro.	---

Clase AccountController	
Responsabilidades	**Colaboradores**
Autenticar usuarios en el sistema.	UserModel, BussinesImpl

Clase AdminController

Responsabilidades	Colaboradores
Gestionar los administradores de la aplicación.	BussinesImpl
Gestionar solicitudes.	BussinesImpl

Clase GuestController	
Responsabilidades	**Colaboradores**
Permitir el inicio de las solicitudes de cambio de imagen de perfil de usuario.	BussinesImpl
Comprobar el estado de la solicitud.	BussinesImpl

Clase BussinesImpl	
Responsabilidades	**Colaboradores**
Permitir la persistencia de los datos de la aplicación.	IBussines

Anexo 5: Tareas de ingeniería

Iteración	Historia de usuario	Tareas
	7. Verificar balance de colores.	• Verificar que los colores sean naturales. • Notificar al usuario los resultados de la validación.
	8. Validar nitidez.	• Verificar que la imagen no esté borrosa o desenfocada. • Notificar al usuario los resultados de la validación.
	9. Validar iluminación.	• Verificar que la luz esté distribuida uniformemente en el rostro.

2		• Notificar al usuario los resultados de la validación.
	10. Verificar expresión de la boca.	• Verificar que la boca esté cerrada y con una expresión neutral. • Notificar al usuario los resultados de la validación
	11. Verificar mirada al frente.	• Verificar que la mirada se encuentre orientada de manera frontal. • Notificar al usuario los resultados de la validación.
	12. Validar pose.	• Verificar que el rostro presente una pose frontal. • Notificar al usuario los resultados de la validación.
3	1. Permitir cargar imagen	• Definir interfaz que permita al usuario cargar una imagen en el sistema. • Cargar la imagen facial del usuario, la misma debe estar ubicada en una de las carpetas de la computadora.
	13. Cambiar imagen	• Enviar un mensaje identificado como petición de validación al administrador. • En caso de que la validación de la imagen sea satisfactoria, actualizar la imagen de perfil de usuario en el directorio UCI.
	14. Gestionar control de acceso	• Definir interfaz de autenticación de usuario. • Validar los datos introducidos por el usuario (nombre de usuario y contraseña). • Permitir el acceso al sistema según el rol del usuario (Acceso mediante LDAP, Acceso del administrador del sistema). • Notificar al usuario si los datos son correctos o no.

- **Tareas ingenieriles para codificar cada una de las historias de usuario.**

Tarea de ingeniería	
Número: TI_1_HU_2	**Historia de usuario:** Detectar rostro.
Nombre de tarea: Detectar puntos característicos en la imagen.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/02/2014	**Fecha fin:** 19/02/14
Programador responsable: Daniel Vega Torres.	
Descripción: Detectar los puntos característicos en la imagen mediante el uso de la biblioteca de clases ASM.	

Tarea de ingeniería	
Número: TI_2_HU_3	**Historia de usuario:** Verificar rostro centrado.
Nombre de tarea: Verificar que el rostro esté centrado de manera horizontal en la imagen.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 20/02/2014	**Fecha fin:** 21/02/14
Programador responsable: Daniel Vega Torres.	
Descripción: El rostro debe estar centrado de manera horizontal y la posición vertical de los ojos estar entre el 30% y el 50% del alto de la imagen.	

Tarea de ingeniería	
Número: TI_3_HU_4	**Historia de usuario:** Verificar fondo uniforme.

Nombre de tarea: Verificar que el fondo tras la silueta de la persona sea uniforme.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 24/02/2014	**Fecha fin:** 28/02/14
Programador responsable: Daniel Vega Torres.	
Descripción: Utilizar plantilla con área de fondo preestablecida. Adaptar la planilla a la imagen de rostro, el área que es considerada como fondo se evalúa según la desviación estándar de la intensidad de los píxeles que la conforman.	

Tarea de ingeniería	
Número: TI_4_HU_5	**Historia de usuario:** Verificar ojos no rojos.
Nombre de tarea: Verificar que los ojos de la persona en la imagen no estén rojos, permitiéndose solamente colores naturales.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 3/03/2014	**Fecha fin:** 7/03/14
Programador responsable: Daniel Vega Torres.	
Descripción: Definir región rectangular que delimite el área donde se espera esté la pupila del ojo representada en un espacio de colores RGB, calcular el grado de coloración roja presente en dicha región.	

Tarea de ingeniería	
Número: TI_5_HU_6	**Historia de usuario:** Verificar la no presencia de espejuelos.
Nombre de tarea: Detectar si el individuo porta espejuelos.	
Tipo: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 10/03/2014	**Fecha fin:** 14/03/14
Programador responsable: Daniel Vega Torres.	
Descripción: Determinar si el individuo representado en la imagen porta espejuelos o no, desarrollar para esto un algoritmo basado en la propuesta de Jing y Mariani para detectar y extraer de forma automática los espejuelos en imágenes de rostros.	

Tarea de ingeniería	
Número: TI_6_HU_7	**Historia de usuario:** Verificar balance de colores.
Nombre de tarea: Verificar que los colores sean naturales.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 17/03/2014	**Fecha fin:** 21/03/14
Programador responsable: Daniel Vega Torres.	
Descripción: Evaluar el balance de colores en la imagen teniendo en cuenta el contraste, la saturación y la exposición como indicadores que se incluyen dentro de la estimación del comportamiento de los colores de la imagen. Por cada uno de estos tres indicadores se definen dos umbrales, uno mínimo y uno máximo. El intervalo definido por estos umbrales en cada uno de los indicadores, representa los valores de comportamiento para imágenes de buena calidad, en este caso con colores naturales	

Tarea de ingeniería	
Número: TI_7_HU_8	**Historia de usuario:** Validar nitidez.
Nombre de tarea: Verificar que la imagen no esté borrosa o desenfocada.	
Tipo: Desarrollo	**Puntos estimados:** 1

Fecha inicio: 24/03/2014	**Fecha fin:** 28/03/14
Programador responsable: Daniel Vega Torres.	
Descripción: Evaluar el grado de distorsión de la imagen aplicando un filtro de media de 3x3 píxeles a la imagen que se está evaluando para obtener la imagen suavizada, que luego se resta de la imagen original para saber en cuánto se diferencian.	

Tarea de ingeniería	
Número: TI_8_HU_9	**Historia de usuario:** Validar iluminación.
Nombre de tarea: Verificar que la luz esté distribuida uniformemente en el rostro.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 31/03/2014	**Fecha fin:** 1/04/14
Programador responsable: Daniel Vega Torres.	
Descripción: Desarrollar algoritmo basado en el análisis local del comportamiento de la iluminación sobre el rostro, para esto diseñar malla triangular a aplicar sobre la imagen, en la que cada triángulo define una región con una luminancia aproximadamente constante. De cada región, se determina el valor promedio de luminancia y se compara con el valor esperado para una imagen de buena calidad.	

Tarea de ingeniería	
Número: TI_9_HU_10	**Historia de usuario:** Verificar expresión de la boca.
Nombre de tarea: Verificar que la boca esté cerrada y con una expresión neutral.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 2/04/2014	**Fecha fin:** 8/04/14

Programador responsable: Daniel Vega Torres.
Descripción: Verificar la simetría de la boca utilizando el algoritmo para la validación de la simetría del rostro tomando los triángulos pertenecientes a la zona de la boca. Después obtener distancia euclidiana entre los puntos interiores y los exteriores a la boca y comparar con el umbral obtenido mediante la evaluación de dicha medida en imágenes con una correcta expresión de la boca.

Tarea de ingeniería	
Número: TI_10_HU_11	**Historia de usuario:** Verificar mirada al frente.
Nombre de tarea: Verificar que la mirada se encuentre orientada de manera frontal.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 9/04/2014	**Fecha fin:** 15/04/14
Programador responsable: Daniel Vega Torres.	
Descripción: Desarrollar un algoritmo de segmentación de la imagen para obtener la región de cada ojo. Luego con el uso de puntos ubicados mediante un modelo de forma activa en el centro de cada ojo, definir rectángulo donde se espere la presencia de la pupila del ojo.	

Tarea de ingeniería	
Número: TI_11_HU_12	**Historia de usuario:** Validar pose.
Nombre de tarea: Verificar que el rostro presente una pose frontal.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 16/04/2014	**Fecha fin:** 18/04/14
Programador responsable: Daniel Vega Torres.	

Descripción: Analizar la simetría del rostro utilizando el método propuesto por Gao, diseñar para esto una malla triangular sobre la cual deben ser aplicados los Patrones Binarios Locales (LBP) con la medida de similitud Chi-Cuadrado para comparar las dos mitades del rostro.

Tarea de ingeniería	
Número: TI_12_HU_1	**Historia de usuario:** Permitir cargar imagen.
Nombre de tarea: Cargar desde una dirección especificada por el usuario, la imagen facial.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 21/04/2014	**Fecha fin:** 22/04/14
Programador responsable: Alisbet Fernández Rojas.	
Descripción: La aplicación debe cargar una imagen almacenada en un archivo específico.	

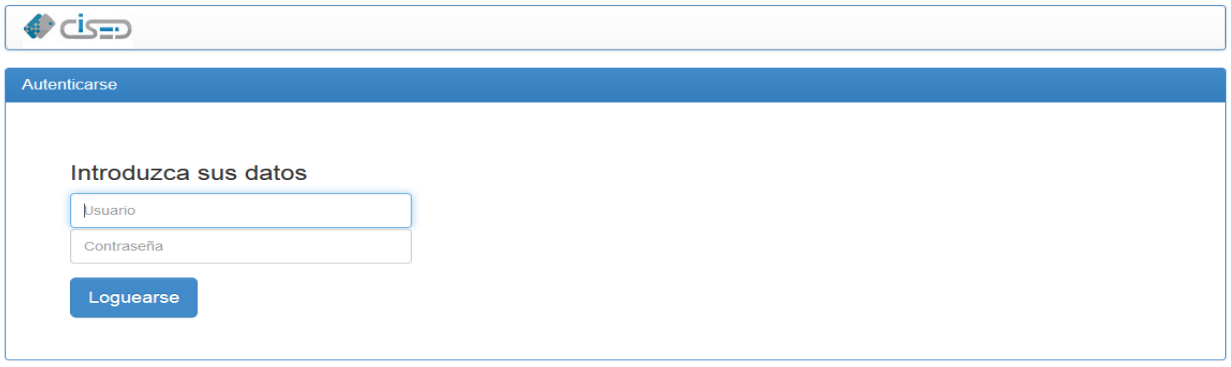
Tarea de ingeniería	
Número: TI_13_HU_13	**Historia de usuario:** Cambiar imagen.
Nombre de tarea: En caso de que la validación de la imagen sea satisfactoria, actualizar la imagen de perfil de usuario en el directorio UCI.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 23/04/2014	**Fecha fin:** 29/04/14
Programador responsable: Alisbet Fernández Rojas.	
Descripción: Cuando el administrador del sistema de su aprobación respecto a la validación de la imagen cargada por el usuario en el sistema, acceder a la BD y cambiar la imagen en el perfil del usuario.	

Tarea de ingeniería	
Número: TI_14_HU_14	**Historia de usuario:** Gestionar control de acceso.
Nombre de tarea: Permitir el acceso al sistema.	
Tipo: Desarrollo	**Puntos estimados:** 1
Fecha inicio: 31/04/2014	**Fecha fin:** 2/05/14
Programador responsable: Alisbet Fernández Rojas.	
Descripción: La aplicación debe ser capaz de permitir el acceso al sistema a los usuarios autorizados, consumir para esto el servicio LDAP UCI.	

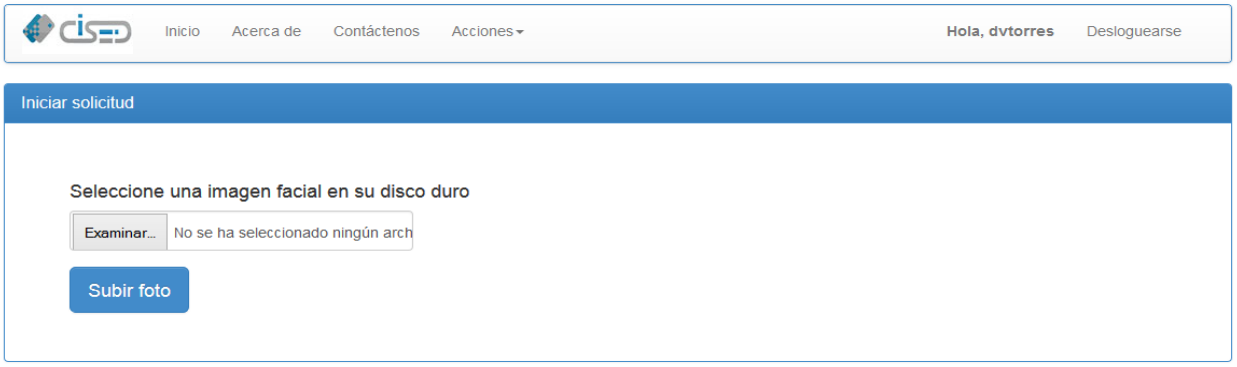
Anexo 6: Interfaces de usuario.

A continuación se explica cada interfaz de usuario de la aplicación para que se tenga un mejor entendimiento en el trabajo con la misma.

Para el usuario poder solicitar el cambio de imagen en su perfil del directorio UCI debe primeramente autenticarse en el sistema a través de la siguiente interfaz, para lo cual debe introducir su correspondiente usuario y contraseña.

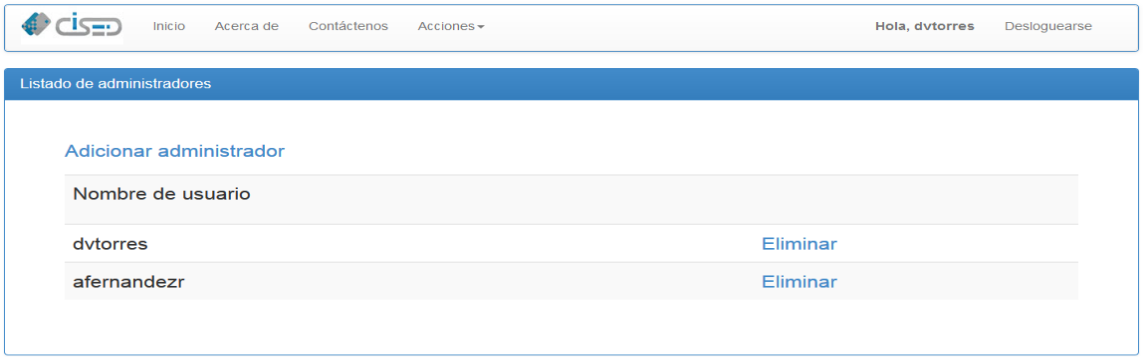


Una vez que se ejecute la correcta autenticación por parte del usuario en el sistema, el mismo procede a cargar la nueva imagen de perfil que desea a través de la siguiente interfaz.

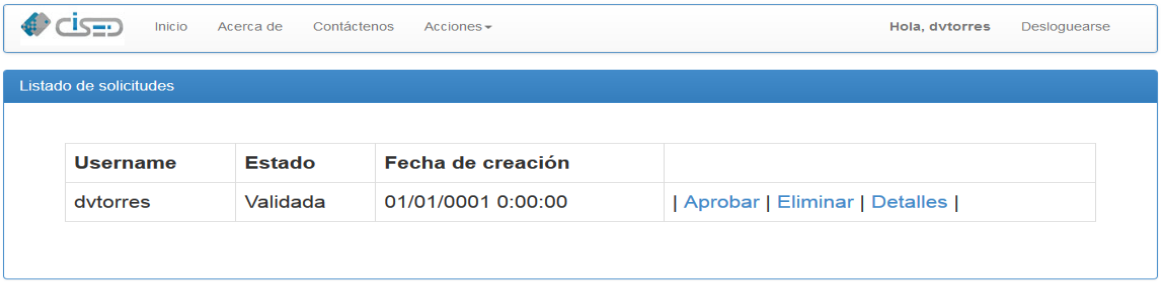


La siguiente interfaz se obtiene luego de que el usuario haya cargado la imagen facial que desea actualizar, en la misma se mostrará el resultado de la petición realizada.

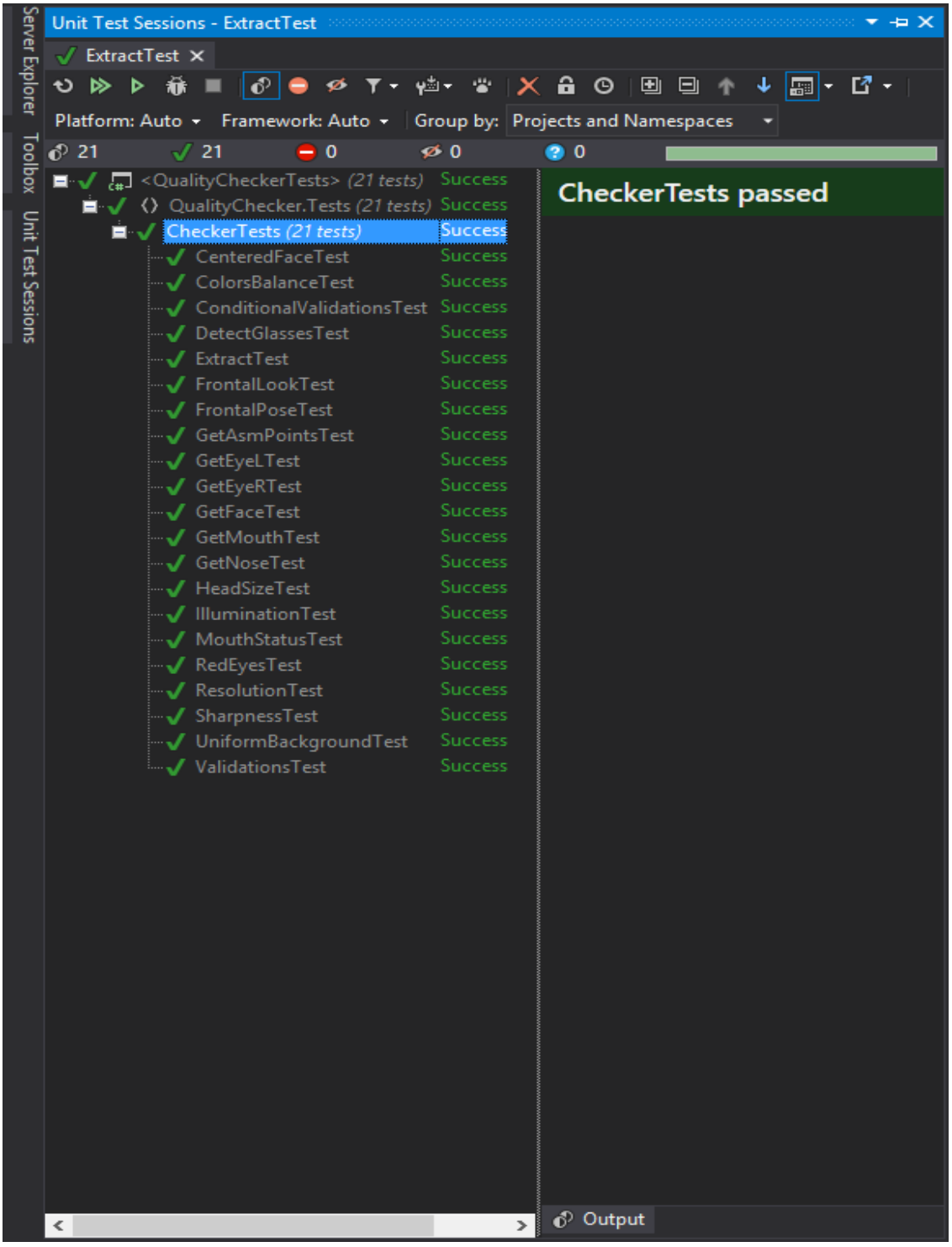
A través de la siguiente interfaz se puede visualizar el listado de administradores del sistema:



A través de la siguiente interfaz se puede visualizar el listado de solicitudes aceptadas:



Anexo 7: Resultado de pruebas unitarias



```
[TestMethod()]
0 references
public void ConditionalValidationsTest()
{
    Bitmap image = new Bitmap("image.jpg");
    Checker checker = new Checker(image, "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug",
        "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug");
    checker.Extract();
    int[] result = checker.ConditionalValidations();
    Assert.IsNotNull(result);
}
```

```
[TestMethod()]
0 references
public void ConditionalValidationsTest()
{
    Bitmap image = new Bitmap("image.jpg");
    Checker checker = new Checker(image, "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug",
        "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug");
    checker.Extract();
    int[] expected = {1, 1, 1, -1, 0, 0, 0, 0, 0, 0, 0, 0};
    int[] result = checker.ConditionalValidations();
    Assert.AreEqual(expected,result);
}
```

```
[TestMethod()]
0 references
public void ExtractTest()
{
    Bitmap image = new Bitmap("image.jpg");
    Checker checker = new Checker(image, "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug",
        "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug");
    int[] result = checker.Extract();
    Assert.IsNotNull(result);
}
```

```
[TestMethod()]
0 references
public void FrontalPoseTest()
{
    Bitmap image = new Bitmap("image.jpg");
    Checker checker = new Checker(image, "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug",
        "D:/Repositorio/QualityChecker/QualityCheckerTests/bin/Debug");
    checker.Extract();
    int expected = 1;
    int result = checker.FrontalPose();
    Assert.AreEqual(expected,result);
}
```

Anexo 8: Código del algoritmo para la validación de los parámetros de calidad en las imágenes.

```
public int[] ConditionalValidations()
{
    var result = new int[12];
    if (extractedFeatures.GetDetection()[0] == 1)
    {
        result[0] = bioValidator.Resolution();
        if (result[0] == 1)
        {
            result[1] = bioValidator.Sharpness();
            if (result[1] == 1)
            {
                result[2] = bioValidator.FrontalPose();
                if (result[2] == 1)
                {
                    result[3] = bioValidator.Ilumination();
                    if (result[3] == 1)
                    {
                        result[6] = bioValidator.ColorsBalance();
                        result[7] = idValidator.DetecGlasses();
                        result[9] = idValidator.CenteredFace();
                        result[10] = idValidator.HeadSize();
                        result[11] = idValidator.UniformBackground();
                        if (extractedFeatures.GetDetection()[1] == 1)
                        {
                            result[4] = bioValidator.FrontalLookCall();
                            result[8] = idValidator.RedEyes();
                        }
                        if (extractedFeatures.GetDetection()[2] == 1)
                        {
                            result[5] = bioValidator.MouthStatus();
                        }
                    }
                }
            }
        }
    }
    return result;
}
```

Anexo 9: Casos de prueba

Caso de prueba de aceptación

Código de caso de prueba: CP1_ HU2	Nombre de la historia de usuario: Detectar rostro.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para detectar el rostro en una imagen.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema trata de detectar el rostro en la imagen.	
Resultado esperado: Si el rostro en la imagen se pudo detectar de manera satisfactoria se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	
Evaluación de la prueba: Prueba satisfactoria.	

	Caso de prueba de aceptación
Código de caso de prueba: CP2_ HU3	Nombre de la historia de usuario: Verificar rostro centrado.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar que el rostro esté centrado en una imagen facial.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema verifica que el rostro esté centrado en la imagen.	
Resultado esperado: Si el rostro está centrado se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	

Evaluación de la prueba: Prueba satisfactoria.

Caso de prueba de aceptación	
Código de caso de prueba: CP3_ HU4	Nombre de la historia de usuario: Verificar fondo uniforme.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar que el fondo sea uniforme en una imagen facial.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema verifica que el fondo sea uniforme.	
Resultado esperado: Si el fondo es uniforme se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	
Evaluación de la prueba: Prueba satisfactoria.	

Caso de prueba de aceptación	
Código de caso de prueba: CP4_ HU5	Nombre de la historia de usuario: Verificar ojos no rojos.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar que los ojos de la persona no estén rojos.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema verifica que los ojos de la persona no estén rojos.	

Resultado esperado: Si el sistema verifica que los ojos de la persona no están rojos se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).
Evaluación de la prueba: Prueba satisfactoria.

Caso de prueba de aceptación	
Código de caso de prueba: CP5_ HU6	Nombre de la historia de usuario: Verificar la no presencia de espejuelos.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para detectar si la persona porta espejuelos.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema detecta la presencia de espejuelos o gafas oscuras.	
Resultado esperado: Si la presencia de espejuelos es nula se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	
Evaluación de la prueba: Prueba satisfactoria.	

Caso de prueba de aceptación	
Código de caso de prueba: CP6_ HU7	Nombre de la historia de usuario: Verificar balance de colores.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar el balance de colores en una imagen facial.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	

Entrada/Pasos de ejecución: • El sistema verifica el balance de colores en la imagen.
Resultado esperado: Si el balance de colores es adecuado se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).
Evaluación de la prueba: Prueba satisfactoria.

	Caso de prueba de aceptación
Código de caso de prueba: CP7_ HU8	Nombre de la historia de usuario: Validar nitidez.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para validar la nitidez en una imagen facial.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema verifica que la imagen es nítida.	
Resultado esperado: Si la nitidez en la imagen es favorable se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	
Evaluación de la prueba: Prueba satisfactoria.	

	Caso de prueba de aceptación
Código de caso de prueba: CP8_ HU9	Nombre de la historia de usuario: Validar iluminación.
Responsable de la prueba: Daniel Vega Torres	

Descripción de la prueba: Prueba de funcionalidad para validar la iluminación en una imagen facial.
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.
Entrada/Pasos de ejecución: • El sistema valida que la imagen tenga buena iluminación.
Resultado esperado: Si la iluminación en la imagen es favorable se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).
Evaluación de la prueba: Prueba satisfactoria.

Caso de prueba de aceptación	
Código de caso de prueba: CP9_ HU10	Nombre de la historia de usuario: Verificar expresión de la boca.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar la expresión de la boca de la persona en una imagen facial.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema verifica que la boca esté correctamente cerrada, sin ningún tipo de inclinación en la misma.	
Resultado esperado: Si la expresión de la boca es la apropiada se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	
Evaluación de la prueba: Prueba satisfactoria.	

Caso de prueba de aceptación	
Código de caso de prueba: CP10_ HU11	Nombre de la historia de usuario: Verificar mirada al frente.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar que la mirada de la persona en una imagen facial sea al frente.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema verifica que la mirada de la persona sea al frente de manera horizontal.	
Resultado esperado: Si la mirada de la persona es la apropiada (al frente de manera horizontal) se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio en la leyenda).	
Evaluación de la prueba: Prueba satisfactoria.	

Caso de prueba de aceptación	
Código de caso de prueba: CP11_ HU12	Nombre de la historia de usuario: Validar pose.
Responsable de la prueba: Daniel Vega Torres	
Descripción de la prueba: Prueba de funcionalidad para verificar la buena postura de la persona en una imagen facial.	
Condiciones de ejecución: La imagen debe haber sido cargada en el sistema.	
Entrada/Pasos de ejecución: • El sistema valida que la persona tenga una buena pose.	
Resultado esperado: Si la pose es la adecuada se muestra el indicador en verde, en caso contrario el sistema muestra el indicador en rojo (el color verde significa bien/satisfactorio, el color rojo significa mal/insatisfactorio	

en la leyenda).
Evaluación de la prueba: Prueba satisfactoria.

Caso de prueba de aceptación	
Código de caso de prueba: CP12_ HU1	Nombre de la historia de usuario: Permitir cargar imagen.
Responsable de la prueba: Alisbet Fernández Rojas	
Descripción de la prueba: Prueba de funcionalidad para permitirle al usuario cargar una imagen en el sistema.	
Condiciones de ejecución: El usuario debe estar autenticado en el sistema.	
Entrada/Pasos de ejecución: • El usuario carga una imagen en el sistema desde un directorio local de la pc.	
Resultado esperado: Iniciar el proceso de validación de la imagen cargada por el usuario.	
Evaluación de la prueba: Prueba satisfactoria.	

Caso de prueba de aceptación	
Código de caso de prueba: CP14_ HU14	Nombre de la historia de usuario: Gestionar control de acceso.
Responsable de la prueba: Alisbet Fernández Rojas	
Descripción de la prueba: Prueba de funcionalidad para permitir el acceso de los usuarios al sistema.	
Condiciones de ejecución: El usuario accede a la interfaz de autenticación del sistema.	
Entrada/Pasos de ejecución: • El sistema muestra un formulario que espera los siguientes datos. ✓ Identificador de usuario (cadena de texto no vacía de 30 caracteres como máximo que admite solo	

letras).

- ✓ Contraseña (cadena de texto no vacía de 128 caracteres como máximo).

- Si el identificador de usuario existe y la contraseña es correcta, el sistema muestra una interfaz de bienvenida, si no, se muestra un mensaje de error de identificador de usuario o contraseña incorrecta.

Resultado esperado: El usuario queda autenticado en el sistema.

Evaluación de la prueba: Prueba satisfactoria.