

**Prácticas profesionales informe final**

**MACONDOLAB**

**Integrantes**

Olmar Canchano Salcedo

Nombre del Proyecto

Reconocimiento Facial OpenCv

Identificación del Grupo: creada por el profesor

**Fecha: 11/06/21**

**Historial de Cambios**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Revisión | Fecha | Ítem | Descripción de Cambio | Responsable |
| 1 | 03/03/21 | 1 | Iniciación de la Identificación y contexto del problema | Olmar canchano |
| 2 | 10/03/21 | 1,2,3 | Realización de la Introducción, entregables, cronograma y presupuesto | Olmar Canchano |
| 3 | 12/03/21 | 4 | Resumen del trabajo requerido | Olmar Canchano |
| 4 | 15/03/21 | 4 | Agregue los últimos componentes en el resumen del trabajo requerido | Olmar Canchano |
| 5 | 20/03/21 | 5 | Completar la organización y control del proyecto | Olmar Canchano |
| 6 | 1/04/21 | 6 | Realización del entorno y proceso de software | Olmar Canchano |
| 7 | 3/04/21 | 6 | Agregar los últimos complementos de entorno y proceso de software | Olmar Canchano |
| 8 | 12/04/21 | 6.2 | Inicialización del diseño de software | Olmar Canchano |
| 9 | 17/04/21 | 6.2.4 | Análisis para el requerimiento de software necesario | Olmar Canchano |
| 10 | 21/04/21 | 6 | Finalización del diseño de software necesario | Olmar Canchano |
| 11 | 02/05/21 | 6.3 | Finalización del modelado de software | Olmar Canchano |
| 12 | 15/05/21 | 6.5 | diseño de interfaz y diseño de datos | Olmar Canchano |
| 13 | 27/05/21 | 6.6 | Diseño de arquitectura | Olmar Canchano |
| 14 | 07/05/21 | 6.7 | Construcción y implementación | Olmar Canchano |
| 15 | 11/05/21 | 6.8 | Conclusión y recomendaciones | Olmar Canchano |

**Tabla de Contenido**

[*1*](#_heading=h.30j0zll) *Proyecto de Producto de Software 5*

[1.1](#_heading=h.1fob9te) Identificación y Contexto del Problema 5

[1.2](#_heading=h.3znysh7) Propósito, Alcance y Objetivo del Proyecto 5

[1.3](#_heading=h.2et92p0) Suposiciones y Restricciones 5

[1.4](#_heading=h.tyjcwt) Entregables del proyecto 5

[1.5](#_heading=h.3dy6vkm) Cronograma y Resumen del Presupuesto 5

[*2*](#_heading=h.1t3h5sf) *Documentos de Referencia 6*

[*3*](#_heading=h.4d34og8) *Definiciones y Acrónimos 7*

[*4*](#_heading=h.2s8eyo1) *Resumen del Trabajo Requerido 8*

[*5*](#_heading=h.17dp8vu) *Organización y Control del proyecto 9*

[*6*](#_heading=h.3rdcrjn) *Proceso de Desarrollo de Software 10*

[6.1](#_heading=h.26in1rg) Entorno del Proceso de Software 10

[6.2](#_heading=h.lnxbz9) Diseño de Software 10

[6.2.1](#_heading=h.35nkun2) Definición de Roles o Actores 10

[6.2.2](#_heading=h.1ksv4uv) Análisis de Requerimientos de Software 10

[6.2.3](#_heading=h.44sinio) Requerimientos Funcionales 10

[6.2.4](#_heading=h.2jxsxqh) Requerimientos No funcionales 11

[6.3](#_heading=h.z337ya) Modelado de Software 11

[6.3.1](#_heading=h.3j2qqm3) Diagramas de Casos de Uso 11

[6.3.2](#_heading=h.1y810tw) Diagramas de Secuencia 12

[6.3.3](#_heading=h.4i7ojhp) Diagramas de Actividad 12

[6.3.4](#_heading=h.2xcytpi) Diagrama de Clases 12

[6.4](#_heading=h.1ci93xb) Diseño de la Interfaz 13

[6.4.1](#_heading=h.3whwml4) Arquitectura de Información 13

[6.4.2](#_heading=h.2bn6wsx) Wireframe 13

[6.5](#_heading=h.qsh70q) Diseño de Datos 13

[6.5.1](#_heading=h.3as4poj) Método de acceso a la base de datos 13

[6.5.2](#_heading=h.1pxezwc) Definición de las Entidades de datos 13

[6.5.3](#_heading=h.49x2ik5) Diagrama Entidad Relación 13

[6.5.4](#_heading=h.2p2csry) Diccionario de Datos 14

[6.5.5](#_heading=h.147n2zr) Restricciones de seguridad e integridad 14

[6.6](#_heading=h.3o7alnk) Diseño de la Arquitectura de Software 14

[6.6.1](#_heading=h.23ckvvd) Patrón de Diseño 14

[6.6.2](#_heading=h.ihv636) Diagrama de Componentes 14

[6.6.3](#_heading=h.32hioqz) Diagrama de Despliegue 14

[6.7](#_heading=h.1hmsyys) Construcción e Implementación de Software 14

[6.7.1](#_heading=h.41mghml) Integración del sistema de software 14

[6.7.2](#_heading=h.2grqrue) Verificación y validación del sistema de software 15

[6.7.3](#_heading=h.vx1227) Integración de software en Hardware 15

[6.7.4](#_heading=h.3fwokq0) Transición de software y gestión de lanzamientos 15

[6.7.5](#_heading=h.1v1yuxt) Mantenimiento del software 15

[6.7.6](#_heading=h.4f1mdlm) Gestión de defectos y problemas de software 15

[*Anexos 16*](#_heading=h.2u6wntf)

# **Proyecto de Producto de Software**

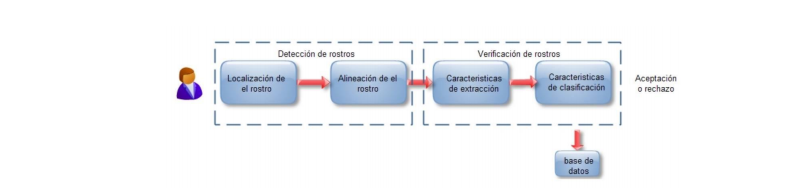
## **Identificación y Contexto del Problema**

**Introducción**

Los seres humanos desde tiempos remotos hasta la actualidad se han visto en la

obligación de reconocerse unos a otros mediante nombres, apodos, etc., pero es el rostro el que nos da una propia identidad a cada persona ya que por medio de estudios la cara es una de las cosas que imposible de olvidar, es por esto que, por medio de las nuevas tecnologías y algoritmos, se han implementado varias funcionalidades propias en respecto a esta identidad. En los inicios de esta tecnología llamada “Reconocimiento facial” se usaba algoritmos de reconocimientos muy simples el cual daba mayor oportunidad a que los errores se produjeran, ya que al ser así el mismo reconocimiento se podría dar para 2 personas diferentes. En la actualidad y con los avances logrados, además de los algoritmos que han sido exponencialmente mejorados, los errores son mínimos ya que se han afinado la forma en cómo se reconoce cada rostro.

Para entendernos mejor veremos cómo funciona cada uno de estos algoritmos además del funcionamiento y cada etapa en un reconocimiento facial, como se muestra en la siguiente figura a continuación, consta de varias secciones las cuales en complemento darán un buen funcionamiento del software que se implementará.



**Problema:**

En la actualidad el problema más grande referido al Reconocimiento Facial es la

disponibilidad de este software como también el costo de implementarlo y que funcione de manera óptima, confiable y segura. Esto no es un impedimento para países desarrollados como Estados Unidos o Rusia los cuales invierten millones de dólares en implementar esta herramienta en sus empresas para que así la autenticación del usuario en algún sistema sea lo más segura posible. En Chile la herramienta biométrica más utilizada es la huella dactilar ya que es menos costosa de implementar, fácil de utilizar y con un marco de error mínimo, pero a su vez se han encontrado métodos para clonar la mano completa de una persona cosa que con el Reconocimiento Facial no sucedería ya que clonar un rostro es mucho más complicado que una mano. Lo que limita implementar un software de Reconocimiento Facial en nuestro país es su alto valor.

Deben ser pocas las empresas en este país que utilicen el Reconocimiento Facial para

autenticar a algún usuario siendo que esta herramienta es una de las más segura y con los

años se ha ido disminuyendo la taza de error a través de nuevos métodos implementados en las técnicas y algoritmos.

Solución:

Son por estos problemas que se decidió dar el pie para empezar a desarrollar un

Software de reconocimiento facial, el cual en su primera versión tendrá las funcionalidades de detectar el rostro y verificarlo con la base de datos para así confirmar la identidad de la persona.

Todo esto se realizará en una interfaz sencilla en un comienzo para que así facilitar su

uso a los usuarios que lo utilicen por primera vez e integrando los algoritmos más utilizados y eficientes que se conozcan, y que se hayan implementado en otros proyectos de la misma naturaleza.

La solución más significativa que se podrá dar en un comienzo es referida a la

seguridad ya que a partir de este programa se podrá ir perfeccionando una y otra vez hasta que se complemente de la tal manera que sea seguro de usar, además de ser eficiente en lo que respecta a la identificación y reconocimiento de rostro y con esto hacer que más empresas la introduzcan en sus sistemas ya sea para un control de acceso o solo para hacer pruebas de algún tipo.

## **Propósito, Alcance y Objetivo del Proyecto**

1. **Objetivo General**

Implementación de un software de Reconocimiento Facial por medio de un lenguaje

de programación utilizando librerías de Open CV.

1. **Objetivos específicos**

* Conocer y comprender las ventajas de utilizar Open CV en el ámbito de reconocimiento de patrones y entender sus librerías para aplicarlas en el uso del reconocimiento facial.
* Realizar estudios en torno a los distintos algoritmos de reconocimiento facial e indicar cuál de éstos funcionar de mejor manera.
* Validar la detección de rostro, el guardado de imágenes y el reconocimiento de la imagen de entrada desarrollar un software de reconocimiento facial.

## **Suposiciones y Restricciones**

**2.1.1 Análisis de Riesgo:**

* **Riesgos de costo**
* Sobrepasar los costos de desarrollo previstos.
* Cambios en el alcance y los requerimientos de la
* parte del cliente.
* Mala estimación de los costos durante la fase de
* inicialización.
* Sobreestimación de los costos de desarrollos previstos.
* **Riesgos de calendario**
* Mala estimación del tiempo necesario.
* Mala asignación de recursos.
* Pérdida de recursos humanos no prevista.
* **Riesgos tecnológicos**
* Usar herramientas mal adaptadas.
* Usar herramientas no aprobadas o con fallas.
* Problemas de hardware/software.
* Problemas de integración de las diferentes partes del proyecto desarrolladas en paralelo.
* **Riesgos operacionales**
* Falta de liderazgo en el equipo.
* Falta de comunicación.
* Falta de motivación del equipo.
* **Riesgos externos**
* Desastres naturales (fuego, inundación, terremoto, entre otros).
* Enfermedades.
* Accidentes de los integrantes del grupo.

**2.1.2 Plan de Mitigación:**

* **Riesgos de calendario**
* Al sobrepasar el tiempo estimado, se buscará reajustar el calendario para
* las nuevas tareas a realizar; se buscará aumentar la rapidez de
* producción, para así recuperar el tiempo perdido.
* Realizar un buen análisis previo.
* En el caso de atrasos con tareas específicas, se debería darle prioridades a las tareas más importantes en el desarrollo del software.
* **Riesgos tecnológicos**
* Llevar a cabo un buen análisis de los requerimientos técnicos de lo que se necesita hacer y realizar una buena investigación de las tecnologías a utilizar.
* Procurar mantener una buena documentación y buen orden mientras se avanza en el proyecto.
* **Riesgos externos**
* Manejar dentro del presupuesto el uso de respaldo de información en caso de cualquier accidente del lugar físico.
* Contratar seguros.
* Cuidarse de las bajas temperaturas.

**2.1.3 Plan de atenuación**

* **Riesgos de calendario**
* Desarrollar versiones betas del programa.
* Cambiar la asignación de los recursos según prioridad.
* Investigar e ir adquiriendo conocimiento para mayor rapidez de las tareas planteadas.
* **Riesgos tecnológicos**
* Corrección en el trabajo existente y cambio a una tecnología más adecuada que este dentro del presupuesto.
* Pago de multas en caso de que la herramienta haya sido usada de una mala manera o corrección en las aplicaciones existentes.
* Estudio y corrección de fallas en la sinergia del proyecto. Hacer uso de documentación.
* **Riesgos externos**
* Hacer uso de respaldos, protecciones existentes y manejar la opción de ir cambiando el tiempo de las tareas según se vaya avanzando en el proyecto.

## Entregables del proyecto

**Detección de rostros**: La detección facial es una tecnología de visión computarizada que determina el lugar y el tamaño de rostros humanos en imágenes o videos. Es uno de los subtipos de detección de objeto/clase, cuya tarea es encontrar la localización y el tamaño de los objetos en una imagen perteneciente a la clase dada.

**Detector de caras: Algoritmo de Viola Jones**: Paul Viola y Michael Jones desarrollaron este algoritmo en 2001. Este sistema de detección de rostros representa un gran avance debido a su rapidez para identificar caras humanas, ya que realiza la clasificación mediante características extraídas en una escala de grises, a diferencia de sus predecesores que la realizaban pixel a pixel y en imágenes de color.

**Reconocimiento Facial**: Este sistema es usado cuando la identidad del individuo es desconocida, y es necesario identificarlo para diversas tareas, tales como: inicio de sesión y acceso a base de datos. Se realiza una búsqueda en la base de datos con imágenes y si existe coincidencia, el individuo es identificado.

**Estudio de viabilidad:** Los estudios de viabilidad son estudios técnico-económicos que se desarrollan para evaluar la pertinencia de ejecutar un proyecto determinado.

**Capacitaciones:** Proceso que ayudara a los usuarios que quieran implementar este software a adaptarse a las circunstancias que se presenten, permitiendo que puedan responder a ellas de manera eficiente, para ello, es necesario desarrollar habilidades por medio del aprendizaje.

## **Cronograma y Resumen del Presupuesto**

**Viene en PDF dentro de la carpeta**

# **Documentos de Referencia**

**Referencias**

|  |  |  |  |
| --- | --- | --- | --- |
| Referencia | Titulo | Autor | Año |
| [R1] | Diseño e implementación de una herramienta de detección facial | García, M. | 2009 |
| [R2] | Software Engineering: Principles and Practice | van Vliet, H | 2007 |
| [R3] | Machine Learning | Horsh W | 2005 |

**Estándares**

|  |  |  |  |
| --- | --- | --- | --- |
| Estandar | Titulo | Autor | Año |
| [E1] | Robust Real-Time Face Detection. International Journal of Computer Vision | Viola, P., Jones M. | 2001 |
| [E2] | Face Recognition with Local Binary Patterns, Spatial Pyramid Histograms and Naive Bayes Nearest Neighbor classification |  | 2010 |
|  |  |  |  |

# Definiciones y Acrónimos

**OpenCV:** OpenCV viene de las siglas Open Source Computer Vision Library [16], es una librería abierta desarrollada por Intel en el año 1999, contiene alrededor de 500 funciones. Esta librería proporciona un alto nivel de funciones para el procesado de imágenes. Algunas de las características que permite OpenCV son operaciones básicas, procesamiento de imágenes, análisis estructural, análisis de movimiento, reconocimiento del modelo, reconstrucción 3D, calibración de cámara, etc.

**EmguCV:** EmguCV es una plataforma cruzada .Net ligada a la librería de Intel OpenCV de

procesamiento de imágenes, permitiendo que las funciones de OpenCV sean llamadas desde .Net, compatible con lenguajes como C#, VB, VC ++, etc. EmguCV está escrito en C#, puede ser compilado en forma Mono (Monodevelop) por lo cual puede ejecutarse en cualquier plataforma que contenga la forma Mono, incluyendo Linux/Solaris y Mac.

Es necesario descargar todos los dlls que vienen incluidos en EmguCV para el uso de esta plataforma.

**Deep Learning:** Es uno de los métodos de aprendizaje de la inteligencia artificial, y a día de hoy pertenece a un subcampo del Machine Learning. El **Deep Learning o aprendizaje profundo** se define como un algoritmo automático estructurado o jerárquico que emula el aprendizaje humano con el fin de obtener ciertos conocimientos. Destaca porque no requiere de reglas programadas previamente, sino que el propio sistema es capaz de *«aprender»* por sí mismo para efectuar una tarea a través de una fase previa de entrenamiento.

**Reconocimiento Facial:** Este sistema es usado cuando la identidad del individuo es desconocida, y es necesario identificarlo para diversas tareas, tales como: inicio de sesión y acceso a base de datos. Se realiza una búsqueda en la base de datos con imágenes y si existe coincidencia, el individuo es identificado**.**

**Inteligencia artificial:** Las máquinas inteligentes imitan las funciones cognitivas de los humanos. La Inteligencia Artificial (IA) es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano.

# Resumen del Trabajo Requerido

El reconocimiento facial es una herramienta que nos permite identificar a una persona automáticamente por medio de una imagen digital. Es una forma de seguridad biométrica que se ha desarrollado crecientemente desde las primeras pruebas en 1995 , en la actualidad los algoritmos poseen hasta 100% más de exactitud que en aquella época en un comienzo solo se utilizaban imágenes en 2D las cuales solos permite ver el rostro de la persona a la cual queremos identificar , pero ha habido actualizaciones de esta herramienta que permiten , en este momento y aun en desarrollo , verlas en 3D , esta dará una mayor precisión y eficacia al momento de que la imagen de la persona coincida con la imagen que esta almacenada en la base de datos. Para realizar un reconocimiento facial se deben analizar las características faciales de la persona, las cuales se pueden extraer ya sea de una fotografía o desde un fotograma en una fuente de video, esta se convierte en una plantilla y luego se compara con las imágenes en una base de datos para verificar la identidad de la persona. El objetivo de este software es el de encontrar, dada una cara desconocida, una conocida que posea las mismas características. Existen dos formas de operar:

1. Verificación o autentificación de caras: En donde se compara una imagen del rostro de una persona con otra imagen a la cual se le quiere hacer coincidir. El software aceptara si las imágenes coinciden o las rechazara si no.
2. Identificación o reconocimiento: Se compara una imagen de una persona desconocida con la base de datos para encontrar la identidad de la persona.

**Detección de rostros:**

La detección facial es una tecnología de visión computarizada que determina el lugar y el tamaño de rostros humanos en imágenes o videos. Es uno de los subtipos de detección de objeto/clase, cuya tarea es encontrar la localización y el tamaño de los objetos en una imagen perteneciente a la clase dada. Mientras que para el ser humano es trivial esta tarea, una computadora presenta dificultades debido a diversos factores como: variabilidad en la posición del rostro, presencia o ausencia de componentes estructurales (ejemplo: bigote), expresión facial, oclusión (uso de lentes, gorro, etc.) y condiciones del ambiente. Detectar el rostro humano es el primer paso en un sistema de reconocimiento facial, sin embargo, influye de manera significante en el resultado del proceso, ya sea dado un conjunto de imágenes o video en tiempo real. Por lo tanto, debe ser capaz de identificar los rostros independientemente de los factores que anteriormente se mencionaron. Los métodos de detección facial se dividen en cuatro categorías (no excluyentes):

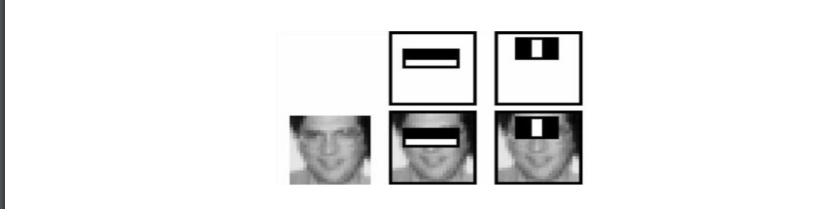
* Métodos basados en conocimiento: Codifican el conocimiento humano mediante distancias y posiciones entre las características humanas (ojos, nariz, labios).
* Métodos basados en características invariantes: Las características invariantes son aquellas que no se modifican a eventuales cambio de luz, pose o ubicación de la cámara, tales como la ceja, nariz, textura de la piel y línea de pelo. Este método 9 funciona detectando uno de estos componentes, construyendo un modelo estadístico y con los resultados, verificar la existencia de un rostro.
* Métodos basados en moldes (patrones): Es la relación entre una imagen de entrada y un patrón o molde previamente definido, cuyo objetivo es capturar características del rostro.
* Métodos basados en apariencia: Utilizan modelos obtenidos mediante entrenamiento de imágenes, tomando la imagen como un vector de características, es decir, es visto como una variable aleatoria. A diferencia de los métodos basados en moldes, donde el patrón es definido por un “experto”, los patrones en este modelo son determinados por el aprendizaje obtenido en el entrenamiento de imágenes.

Existen varios métodos para detectar rostros, sin embargo, el algoritmo de Viola-Jones, es el más eficaz, obteniendo un mayor porcentaje de aciertos respecto a sus pares, además de más rapidez. Este algoritmo integra un nuevo concepto, la imagen integral, que junto con el algoritmo de boost como método de entrenamiento, forman un clasificador complejo y preciso.

**AdaBoost:** AdaBoost (AdaptiveBoost) es un algoritmo de aprendizaje máquina que consiste en la extracción de características por medio de clasificadores. Basado en la mejor forma de ejecutar los clasificadores para detectar de forma favorable el rostro humano. En el algoritmo de Viola-Jones, AdaBoost elige un gran número de características de Haar, para seleccionar cuál de ellas se ajusta mejor para clasificar (clasificador) los distintos elementos, en este caso si es rostro o no. Para seleccionar las características y entrenar al clasificador, se combina una selección de funciones débiles de clasificación para formar un clasificador fuerte. Las funciones débiles están diseñadas para seleccionar una característica de Haar que separa de mejor forma los ejemplos positivos y negativos. Los pasos del algoritmo son:

* Se tienen las imágenes de ejemplo (X1, Y1), (Xn, Yn) donde Yi = 0,1 para ejemplos negativos y positivos respectivamente.
* Inicializar los pesos, dados por 1/2m y 1/2l donde m y l son el número de ejemplos positivos y negativos, respectivamente.
* Normalizar los pesos.
* Para cada característica, se evalúa el error con respecto a una ventana de 24x24 pixeles y ejemplos positivos.
* De esta forma se escoge un clasificador, con el menor error.
* Se actualizan los pesos.
* Después de algunas iteraciones, se obtiene un clasificador fuerte, que separa de manera eficiente ejemplos positivos de ejemplos negativos.

Para detectar bien el rostro, las características elegidas por AdaBoost son significativas y de fácil interpretación. La elección de la primera característica se basa en la propiedad basada en que la región de los ojos es más oscura que la región de la nariz y las mejillas. La segunda característica se basa en que los ojos son más oscuros que el puente de la nariz (comienzo de la nariz).



**FisherFaces:**

Esta técnica considera las imágenes de entrenamiento de un mismo individuo como clases, por lo tanto, existen el mismo número de clases que personas. Una vez definida las clases se procede a calcular dos matrices: la matriz de dispersión entre clases y la matriz de dispersión dentro de clases. Una vez calculada estas matrices se obtiene una matriz de proyección donde cada columna será la base del nuevo subespacio, denominada Fisherfaces.

****

Para realizar el cálculo de la matriz de proyección, requiere que las matrices de dispersión sean no-singulares (que posean inversa), esto no siempre es posible debido a que el número de imágenes casi siempre es menos al número de pixeles de cada imagen. Para solucionar esto, se utilizan Componentes Principales en las matrices de dispersión para reducir su dimensión. Las FisherFaces también permiten una reconstrucción de la imagen, por lo tanto, también se utiliza la comparación de imágenes mediante la distancia euclidiana.

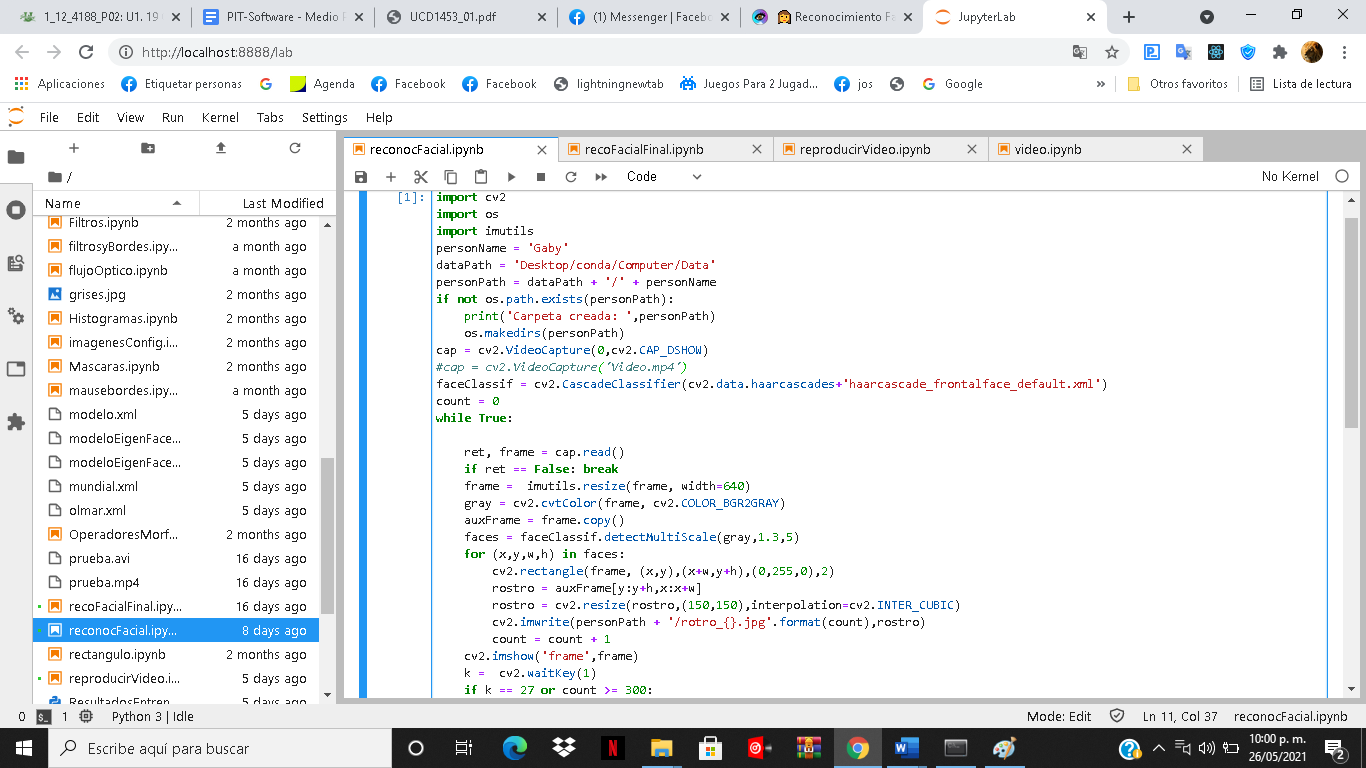
# **Organización y Control del proyecto**

|  |  |  |
| --- | --- | --- |
| Nombre y Apellidos | Rol en el Proyecto | Responsabilidades |
| Olmar canchano | Machine learning Developer | Implementar el entrenamiento de redes neuronales para el aprendizaje profundo, entrenando el software para que este pueda predecir con precisión y exactitud los resultados esperados. |
|  |  |  |
| Jhon Torres | Desarrollador de software | Analista y programador. |

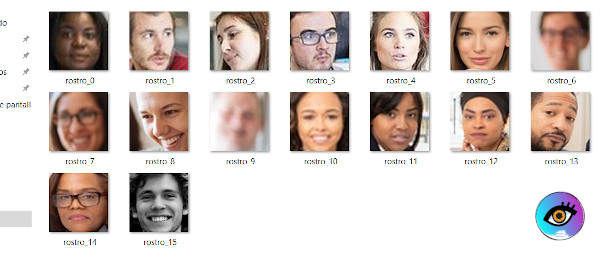
# **Proceso de Desarrollo de Software**

## **Entorno del Proceso de Software**

Como en un principio se mencionó el sistema de Reconocimiento Facial no está sujeto a servidores, ni requiere una conexión a Internet, solo se gestiona a través de una base de datos local situada en el Disco Duro del computador o en la dirección de escritorio que uno escriba en el código de programación. Una Base de datos Local trabaja de la misma manera que una distribuida, o sea que se puede acceder a ella por medio de la red, solo que la primera se ve más restringida a los usuarios que puedan utilizarla, sin embargo, sus datos se pueden visualizar, actualizar, ingresar o eliminar dependiendo de la función que se quiera realizar. En el presente proyecto se ejecutará el funcionamiento de una Base de Datos local. Como se puede ver en la línea de código la dirección en donde se almacenarán las imágenes y nombre es: /Datapath donde se guardarán todas las carpetas de entrada y se obtiene el nombre escrito “NOMBRE\_PERSONA”, en este caso será Gaby, aquí se almacenará todas las imágenes de los rostros de la persona, donde se validará si la carpeta esta repetida, y si la carpeta de guardado no existe se crea una con el mismo nombre.



Banco de imágenes de ejemplo



Por otra parte, dentro de la carpeta se irá almacenando cada imagen entrenada de la persona a la que se le capturó el rostro, en este caso usamos distintos modelos para reconocimiento facial usando open cv, como lo vemos en la imagen a continuación.

# Métodos para entrenar el reconocedor

face\_recognizer = cv2.face.EigenFaceRecognizer\_create()

face\_recognizer = cv2.face.FisherFaceRecognizer\_create()

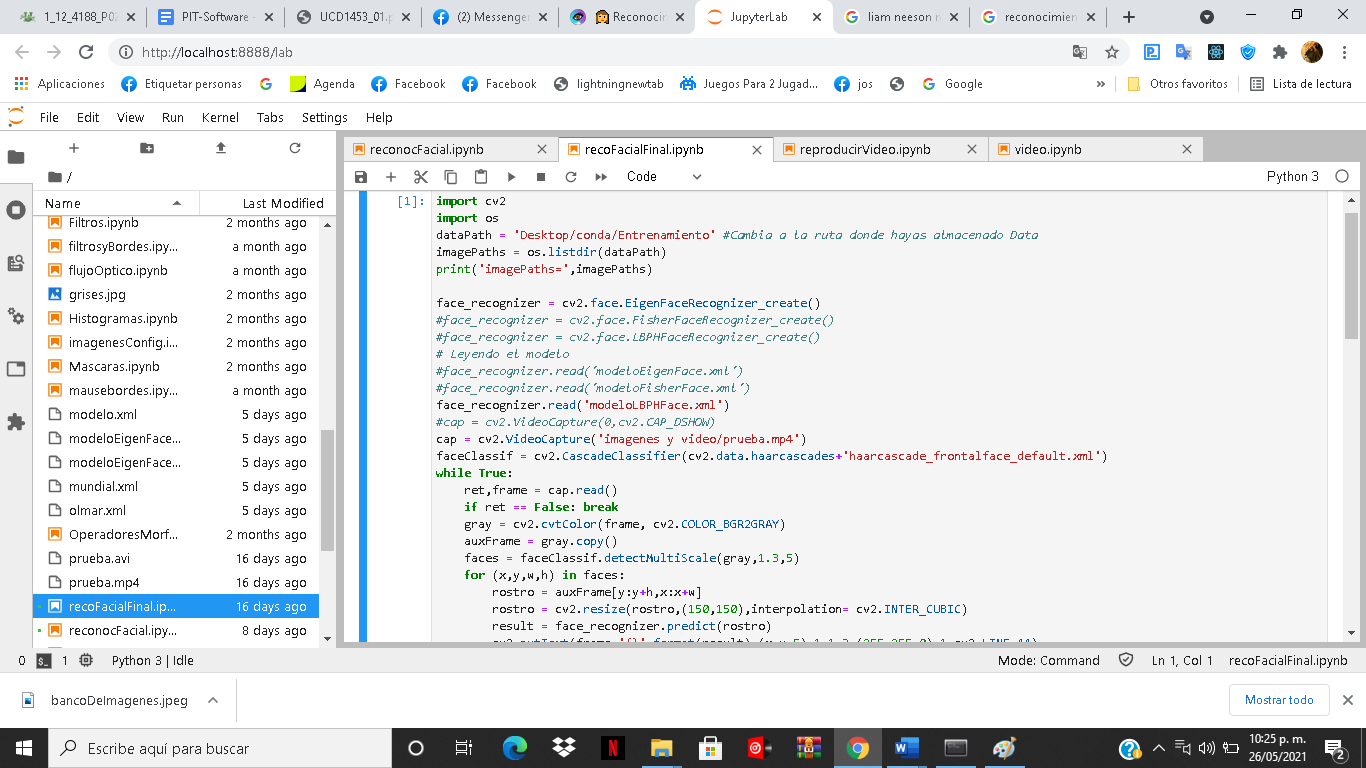
face\_recognizer = cv2.face.LBPHFaceRecognizer\_create()

Una vez que se haya elegido uno de estos métodos es hora de entrenar el reconocedor.

print("Entrenando...")

face\_recognizer.train(facesData, np.array(labels))

y ya tenemos nuestro reconocedor entrenado con toda la base de datos anteriormente mencionada y listo para el reconocimiento facial de imágenes.



## **Diseño de Software**

### **Definición de Roles o Actores**

|  |  |
| --- | --- |
| Nombre del Rol | Descripción |
| Data Scientist | Este tipo de usuario es el que tiene los privilegios para cargar los datasets, generar predicciones y testear los resultados que arroje la red neuronal en el sistema. |

### **Análisis de Requerimientos de Software**

La finalidad del proyecto a realizar es el de la implementación de un software que permita, a través de algoritmos y librerías, poder reconocer el rostro de una persona por medio de una fotografía tomada por una cámara web, hacerle modificaciones a dicha imagen y poder compararla con una base de datos de para verificar la identidad de la persona.

### **Requerimientos Funcionales**

|  |  |  |
| --- | --- | --- |
| Referencia | Requerimiento |  |
| RF01 | Implementación de algoritmo para la detección de rostro |  |
| RF02 | Implementación de algoritmo para el reconocimiento facial |  |
| RF03 | Añadir y eliminar imágenes desde la base de datos |  |
| RF04 | Mostrar en una pantalla de Windows la cámara web para la prueba de software. |  |
| RF05 | Añadir opción para elegir el algoritmo de Reconocimiento facial que se desee ocupar. |  |
| RF06 | Mostrar imagen ya entrenada (blanco y negro). |  |
| RF07 | Crear base de datos local en la cual se almacenará cada imagen |  |
| RF08 | Añadir figura geométrica la cual se posicionará en el rostro de la persona |  |
| RF09 | Implementar segundo algoritmo de reconocimiento facial |  |
| RF10 | Generar un manual de usuario para el uso posterior del sistema |  |
| RF11 | Realizar más de un entrenamiento de imágenes por toma, por ejemplo, tomar 10 fotografías, almacenarlas y entrenarlas con un solo click en el botón de entrenar |  |
| RF12 | Realizar informe sobre los aciertos o no del reconocimiento facial, tanto en el reconocimiento como en la detección, abordando los 2 algoritmos implementados en el sistema |  |

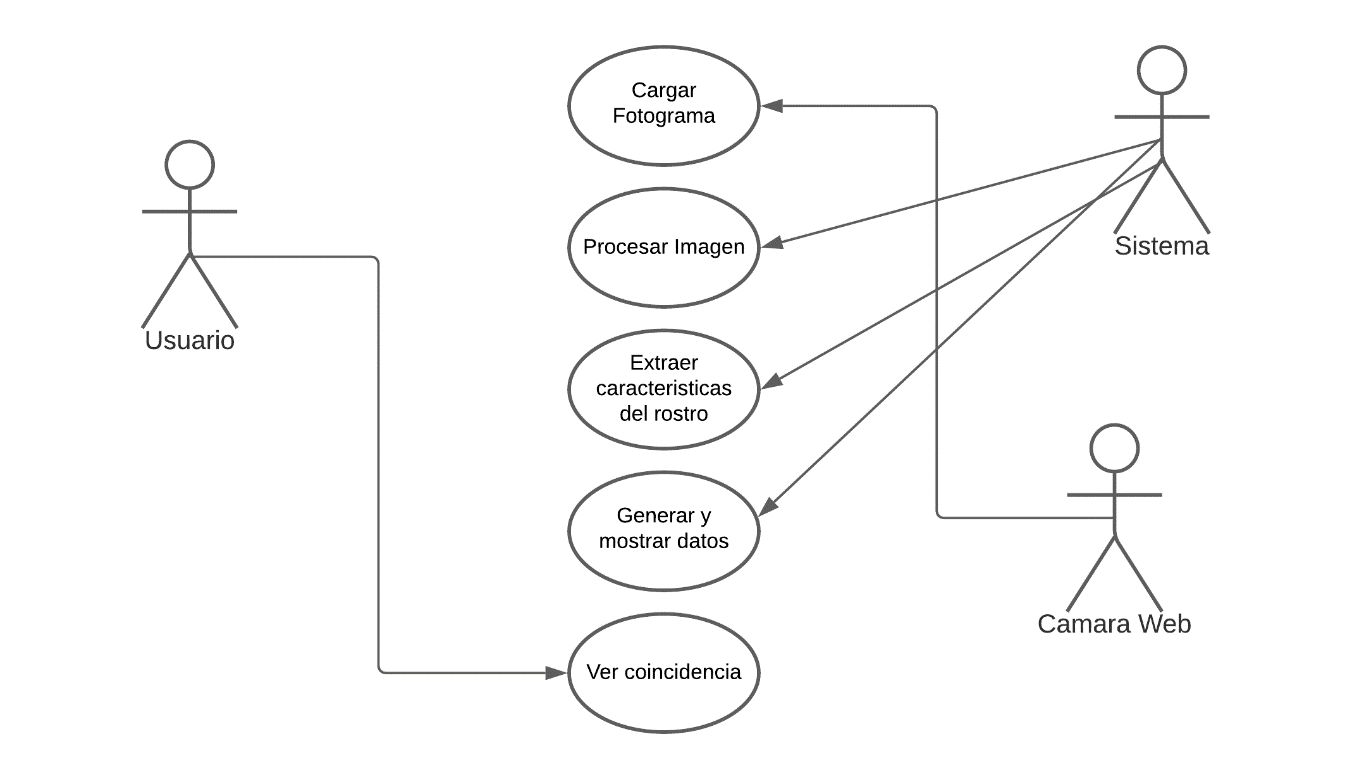
### **Requerimientos No funcionales**

|  |  |  |
| --- | --- | --- |
| Referencia | Tipo | Descripción |
| RNF-01 | Eficiencia | Diseño de una interfaz sencilla para el buen uso de un usuario principiante. |
| RNF-02 | Seguridad | Completa documentación del código |
| RNF-03 | Rendimiento | Rendimiento de software será optimo en donde tomará muy poco tiempo en realizar sus funciones. |
| RNF-04 | Eficiencia | La Base de datos como se implementará de forma local siempre estará disponible. |
| RNF-05 | seguridad | Integridad de las imágenes en la base de datos |
| RNF-06 | Eficiencia | Fácil interacción don el usuario |

## Modelado de Software

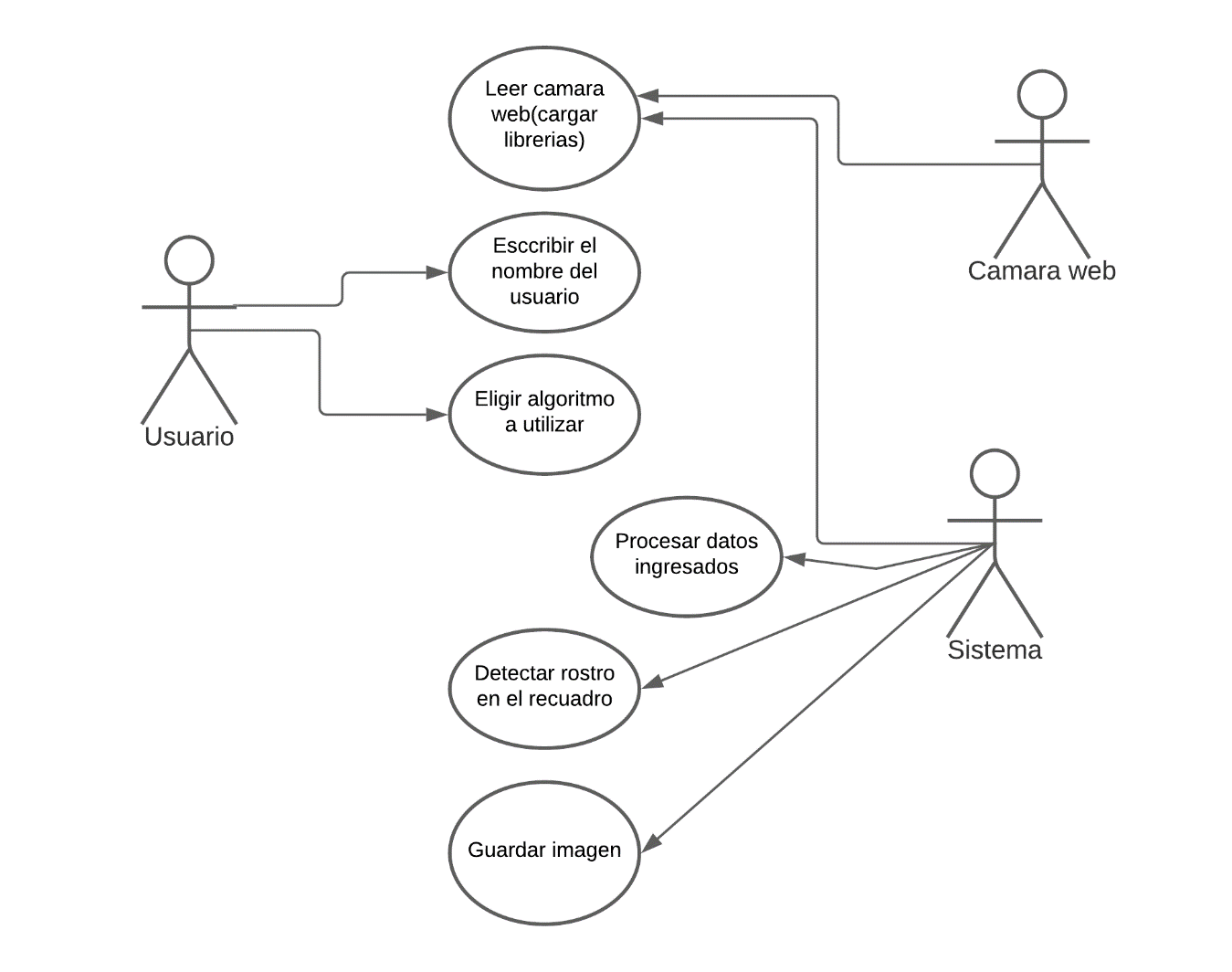
### Diagramas de Casos de Uso

**Diagrama de caso de uso general**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Uso | General | | | |
| Versión | V1 | | Referencia | CU1 |
| Autor | Olmar Canchano | | Fecha | 01/06/2021 |
| Actores | Usuario, Sistema, Cámara Web | | | |
| Referencias | RF04, RF06, RF08 | | | |
| Precondiciones | Haber entrenado el reconocedor. | | | |
| Pos condición | El usuario podrá reconocer a la persona por medio del algoritmo de detección de rostro a través de un recuadro de referencia | | | |
| Flujo Normal | | | | |
| Paso | Actor | Descripción | | |
| 1 | Sistema | Carga los fotogramas, procesa la imagen para extraer las características del rostro en cuestión | | |
| 2 | Cámara Web | Procesa los fotogramas, carga la fotografía para mostrarla al usuario. | | |
| 3 | Usuario | Ve las coincidencias en lo que muestra la cámara sobre el reconocimiento facial. | | |
| Flujos alternos | Si el sistema o la cámara web no procesan los datos, no se podrá aplicar el algoritmo de reconocimiento. | | | |
| Excepciones | Elegir el método de entrenamiento | | | |

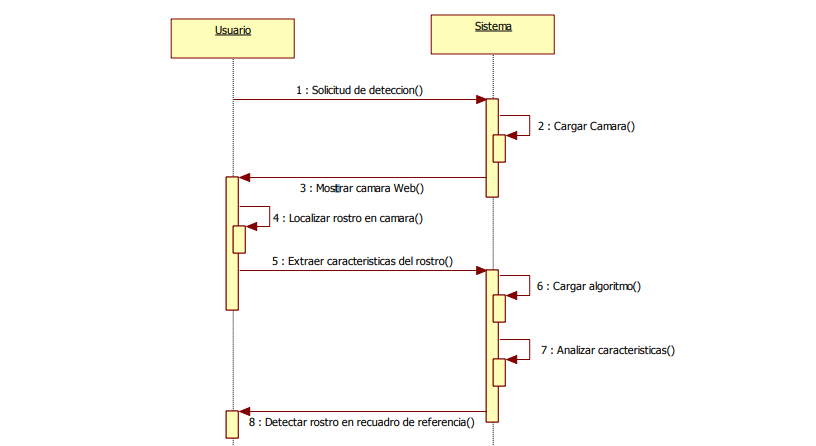
**Cargar Fotograma**



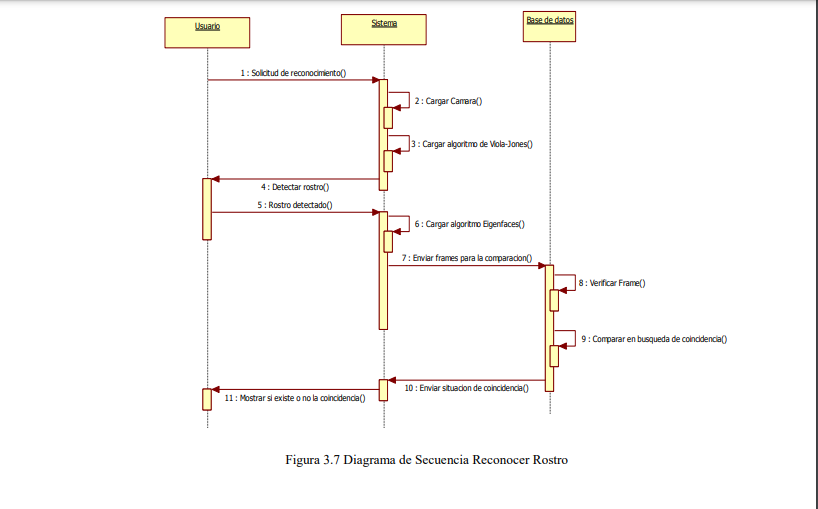
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de Uso | Cargar Fotograma | | | |
| Versión | V1 | | Referencia | CU2 |
| Autor | Olmar Canchano | | Fecha | 01/06/2021 |
| Actores | Usuario, Sistema, Cámara Web | | | |
| Referencias | RF01, RF02, RF03, RF05, RF04, RF06, RF08 | | | |
| Precondiciones | Haber guardado las imágenes en la base de datos, Haber entrenado el reconocedor. | | | |
| Pos condición | Se podrá cargar el video en el cual se reconocerá a la persona en medio de un recuadro de referencia. | | | |
| Flujo Normal | | | | |
| Paso | Actor | Descripción | | |
| 1 | Usuario | Erigirá el nombre de la persona a reconocer. más el método con que se entrenara el reconocedor. para aplicar el algoritmo de reconocimiento facial. | | |
| 2 | Cámara Web | Procesa los fotogramas, carga la fotografía para mostrarla al usuario. | | |
| 3 | Sistema | Procesa los datos ingresados mientras carga las librerías que se utilizaran para detectar el rostro y el entrenamiento del reconocedor. | | |
| Flujos alternos | Sino se especifica el algoritmo y método a utilizar para el entrenamiento del reconocedor el sistema no podrá cargar los fotogramas necesarios para su procesamiento. | | | |
| Excepciones | Imagen entrenada se almacena en una base de datos local | | | |

### Diagramas de Secuencia

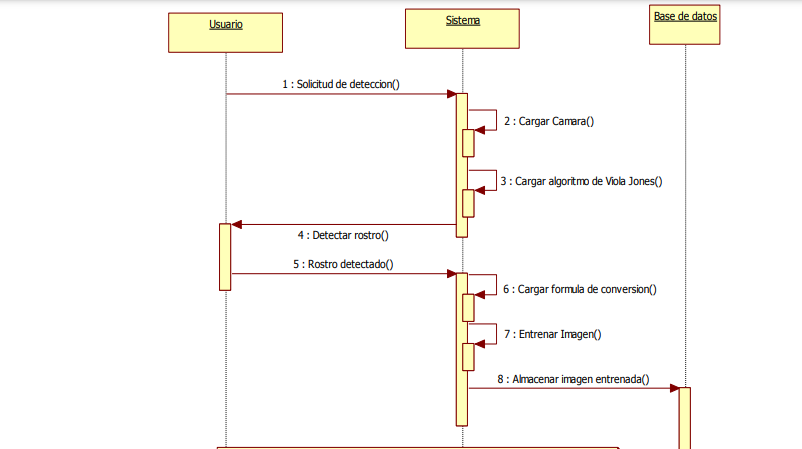
**Diagrama de secuencia detectar rostro**



**Diagrama de secuencia reconocer rostros**

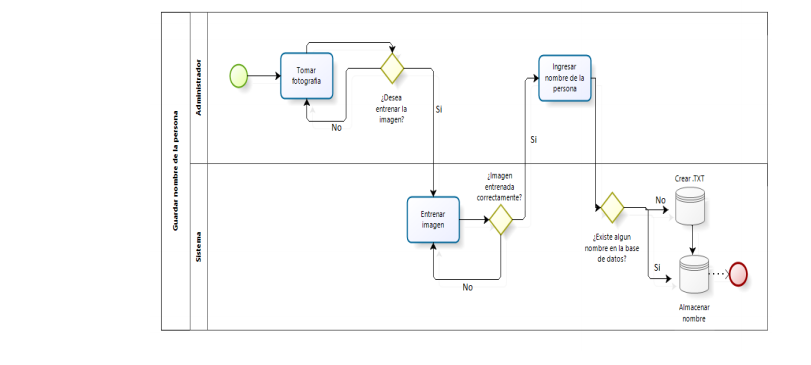
****

**Diagrama de secuencia convertir imágenes en grises**

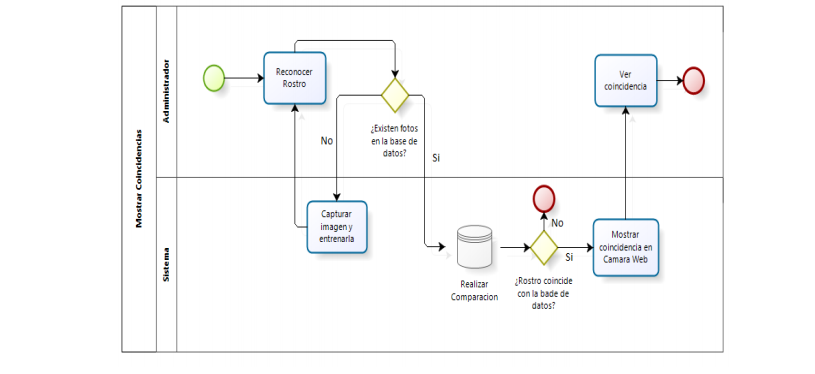
****

### Diagramas de Actividad

**Diagrama BPMN Guardar el nombre de la persona**



**Diagrama BPMN Mostrar Coincidencia**

****

## Diseño de la Interfaz

### Wireframe

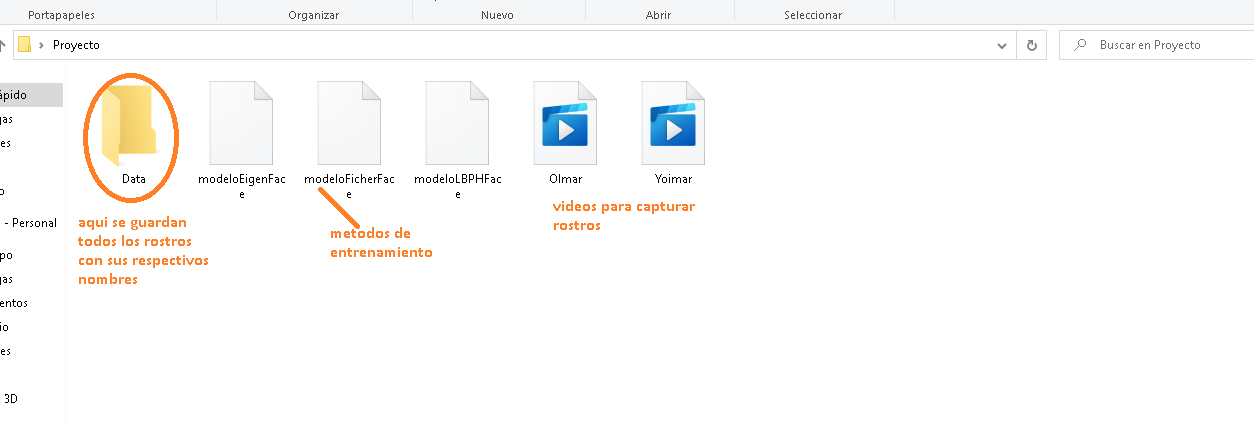
No tenemos un Wireframe especifico ya que no tenemos una página web, mi software solo es un reconocedor de rostro que quedaría de la siguiente manera con su respectivo rostro.



## Diseño de Datos

### Método de acceso a la base de datos

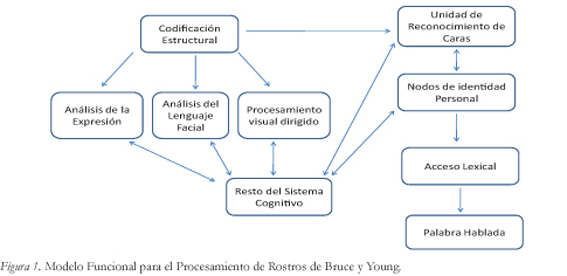
Como en un principio se mencionó el sistema de Reconocimiento Facial no está sujeto a servidores, ni requiere una conexión a Internet, solo se gestiona a través de una base de datos local situada en el Disco Duro del computador o en la dirección de escritorio que uno escriba en el código de programación. Una Base de datos Local trabaja de la misma manera que una distribuida, o sea que se puede acceder a ella por medio de la red, solo que la primera se ve más restringida a los usuarios que puedan utilizarla, sin embargo, sus datos se pueden visualizar, actualizar, ingresar o eliminar dependiendo de la función que se quiera realizar. Como se puede ver a continuación donde se hospedan todos los datos de los rostros y sus respetivos métodos de entrenamiento



### Definición de las Entidades de datos

|  |  |
| --- | --- |
| Entidad | Descripción |
| Data | Aquí se guardarán todos los rostros de las personas, en sus respectivas carpetas con sus nombres. |
| Métodos de entrenamiento | Son Modelos para entrenar el reconocedor, cada uno de estos nos arrojara un margen de error habiendo entrenado la data y se escoge el de mejor resultado. |
|  |  |

### Diagrama Entidad Relación



.

## Diseño de la Arquitectura de Software

### Patrón de Diseño

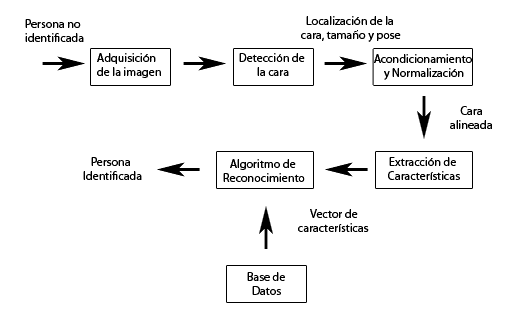
Detectar el rostro humano es el primer paso en un sistema de reconocimiento facial, sin embargo, influye de manera significante en el resultado del proceso, ya sea dado un conjunto de imágenes o video en tiempo real. Por lo tanto, debe ser capaz de identificar los rostros independientemente de los factores que anteriormente se mencionaron.

Los métodos de detección facial se dividen en cuatro categorías (no excluyentes):

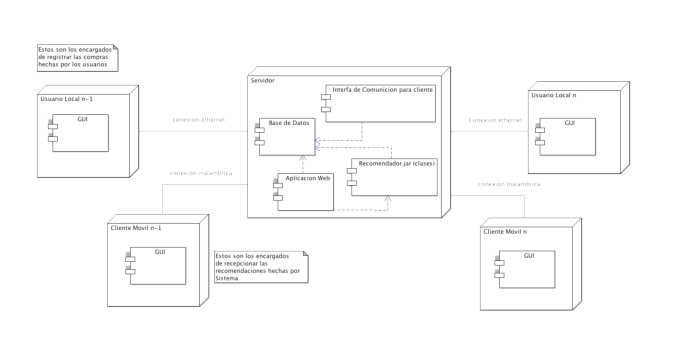
* **Métodos basados en conocimiento**: Codifican el conocimiento humano mediante distancias y posiciones entre las características humanas (ojos, nariz, labios).
* **Métodos basados en características invariantes**: Las características invariantes son aquellas que no se modifican a eventuales cambio de luz, pose o ubicación de la cámara, tales como la ceja, nariz, textura de la piel y línea de pelo. Este método 9 funciona detectando uno de estos componentes, construyendo un modelo estadístico y con los resultados, verificar la existencia de un rostro.
* **Métodos basados en moldes (patrones):** Es la relación entre una imagen de entrada y un patrón o molde previamente definido, cuyo objetivo es capturar características del rostro.
* **Métodos basados en apariencia:** Utilizan modelos obtenidos mediante entrenamiento de imágenes, tomando la imagen como un vector de características, es decir, es visto como una variable aleatoria. A diferencia de los métodos basados en moldes, donde el patrón es definido por un “experto”, los patrones en este modelo son determinados por el aprendizaje obtenido en el entrenamiento de imágenes

Existen varios métodos para detectar rostros, sin embargo, el algoritmo de Viola-Jones, es el más eficaz, obteniendo un mayor porcentaje de aciertos respecto a sus pares, además demás rapidez. Este algoritmo integra un nuevo concepto, la imagen integral, que junto con el algoritmo de boost como método de entrenamiento, forman un clasificador complejo y preciso.

### Diagrama de Componentes



### Diagrama de Despliegue



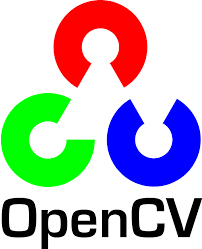
## Construcción e Implementación de Software

**Desarrollo del Sistema:** Para el desarrollo del sistema de reconocimiento facial, se han estado estudiando los lenguajes de programación y las herramientas necesarias para el reconocimiento de rostro. A continuación, se irán nombrando los diferentes elementos con los que se estará desarrollando el software de reconocimiento facial.

**Jupyter-lab**: JupyterLab es una interfaz de usuario basada en web para proyectos Jupyter está estrechamente integrada en Adobe Experience Platform. Proporciona un entorno de desarrollo interactivo para que los científicos de datos trabajen con Jupyter Notebooks, código y datos.



**Open Cv:** OpenCV viene de las siglas Open Source Computer Vision Library [16], es una librería abierta desarrollada por Intel en el año 1999, contiene alrededor de 500 funciones. Esta librería proporciona un alto nivel de funciones para el procesado de imágenes. Algunas de las características que permite OpenCV son operaciones básicas, procesamiento de imágenes, análisis estructural, análisis de movimiento, reconocimiento del modelo, reconstrucción 3D, calibración de cámara, etc.

****

**EmguCV:** es una plataforma cruzada .Net ligada a la librería de Intel OpenCV de procesamiento de imágenes, permitiendo que las funciones de OpenCV sean llamadas desde .Net, compatible con lenguajes como C#, VB, VC ++, etc. EmguCV está escrito en C#, puede ser compilado en forma Mono (Monodevelop) por lo cual puede ejecutarse en cualquier plataforma que contenga la forma Mono, incluyendo Linux/Solaris y Mac. Es necesario descargar todos los dlls que vienen incluidos en EmguCV para el uso de esta plataforma.



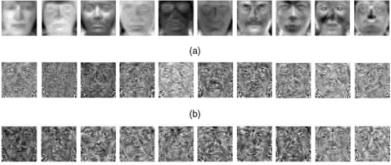
**Entrenamiento de rostros:** Existen bases de datos universales en las cuales están almacenado los rostros de personas comunes y corrientes grabadas en condiciones naturales y con distintos tipos de gestos faciales como, por ejemplo, como se muestra en la figura. Estas bases de datos están liberadas para poder ser descargadas desde la Web para fines de investigación

****

Estas imágenes permiten ahorrarnos el paso del cambio de color a gris por medio de un algoritmo en el software y a su vez poder entrenarlas inmediatamente. Como se muestra en la figura a cada imagen original se le ha aplicado un algoritmo en particular ya sea:

* Eigenfaces.
* FisherFaces.
* Laplacianfaces

Esto permite saber qué imágenes son aptas para poder reconocerlas y cuales se deben rechazar por no ser aptas para entrenarlas y así se podrán realizar estudios sobre qué algoritmo posee menor taza de error y cuál es el que permite mayor tolerancia a la luz o a los movimientos de la cabeza.



Las bases de datos más conocidas en torno a imágenes para reconocimiento facial son YALE, BioId, The AR-FACE, ORL, entre otras. Todas se descargan de manera gratuita y poseen características diferentes unas de otras como la iluminación, incluyen puntos en la cara etc.

**Plan y diseño de pruebas:** Las pruebas en un sistema sirven para verificar que el sistema de información cumple con las necesidades establecidas por el usuario, con las debidas garantías de calidad, para ello se realiza un sistema de pruebas que implican la operación o aplicación del mismo a través de condiciones controladas y por tanto la consiguiente evaluación. Las condiciones controladas deben incluir tanto situaciones normales como anormales. El objetivo de las pruebas de un sistema es encontrar un error para determinar qué hacer en el momento que ocurre algo que no debe pasar y viceversa, es decir, un sistema de pruebas está orientado a detectar errores del sistema de información. Para poder realizar el sistema que deseamos veremos a continuación los distintos tipos de prueba que existen.

* **Pruebas de Caja Negra**: El sistema de pruebas de caja negra no considera la codificación dentro de sus parámetros a evaluar, es decir, que no están basadas en el conocimiento del diseño interno del programa. Estas pruebas se enfocan en los requerimientos establecidos y en la funcionalidad del sistema
* **Pruebas de Caja Blanca**: Al contrario de las pruebas de caja negra, éstas se basan en el conocimiento de la lógica interna del código del sistema. Las pruebas contemplan los distintos caminos que se pueden generar a través de las estructuras condicionales, a los distintos estados del mismo y otros
* **Pruebas de Integración**: Las pruebas de integración buscan probar la combinación de las distintas partes de la aplicación para determinar si funcionan correctamente en conjunto.
* **Pruebas del sistema:** Son similares a las pruebas de caja negra, solo que éstas buscan probar al sistema como un todo. Están basadas en los requerimientos generales y abarca todas las partes combinadas del sistema.
* **Pruebas de implantación**: Incluyen las verificaciones necesarias para asegurar que el sistema funcionará correctamente en el entorno de operación al responder satisfactoriamente a los requisitos de rendimiento, seguridad, operación y coexistencia con el resto de los sistemas de la instalación, y conseguir la aceptación del sistema por parte del usuario de operación
* **Pruebas de aceptación**. Van dirigidas a validar que el sistema cumple con los requisitos de funcionamiento esperados.

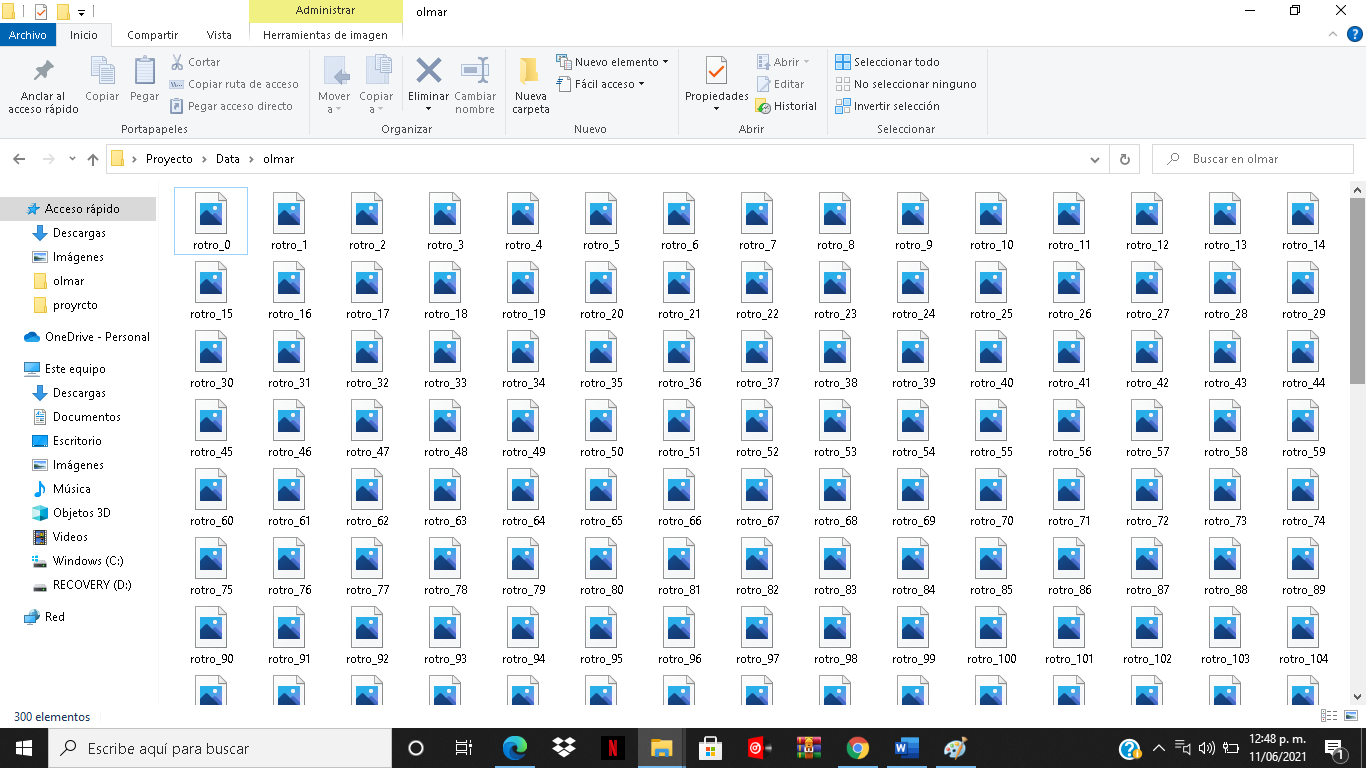
**Planificación de las pruebas:** Para realizar la planificación de pruebas en nuestro sistema se utilizó las pruebas de caja negra y los puntos a evaluar se definen a continuación:

* **Pruebas de funcionalidad**: Este tipo de pruebas examina si el sistema cubre sus necesidades de funcionamiento, acorde a las especificaciones de diseño. En ellas se debe verificar si el sistema lleva a cabo correctamente todas las funciones requeridas, se debe verificar la validación de los datos y se debe realizar pruebas de comportamiento ante distintos escenarios. Estas pruebas deben estar enfocadas a tareas, a límites del sistema, a condiciones planeadas de error y de exploración. Para estas pruebas usamos los esquemas de pruebas de caja negra ya que nos interesa saber si funciona correctamente independientemente de la forma en que se haga.
* **Pruebas de usabilidad**: Las pruebas realizadas en este rubro tienen la finalidad de verificar que tan fácil de usar es el software. Las pruebas de usabilidad deben verificar aprendizaje. Eficiencia, manejo de errores y grado de satisfacción.

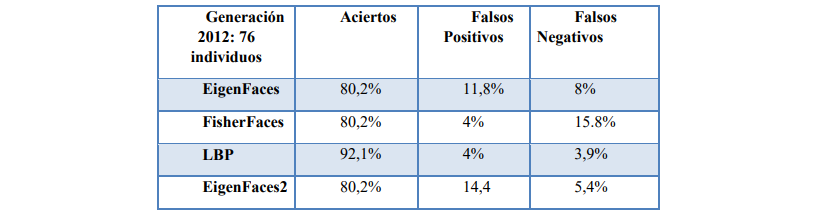
**Pruebas y Resultados:** En este capítulo, se entregan los resultados de las pruebas realizadas para evaluar el rendimiento de los métodos de reconocimiento facial que se han implementado en este proyecto. Cada una de estas pruebas utilizó una o más de una base de datos, con características diferentes que serán útiles para tener una idea más clara del funcionamiento de los algoritmos frente a diferentes condiciones.

**Estudio N°1**

El primer set de pruebas está compuesto por 300 clases (300 fotos de personas diferentes), con una foto 150x150 entrenada desde una foto tamaño carnet 100x125



En este estudio todos los rostros tienen la misma expresión, y están situados de manera frontal, además no hay cambio de entorno. Los resultados son:

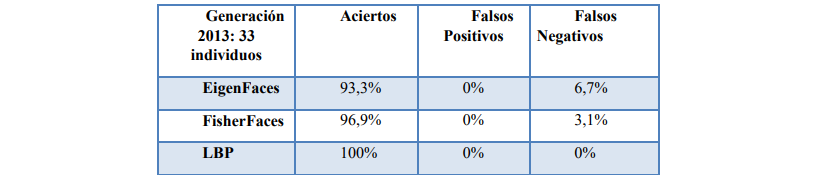


Los resultados son óptimos debido a las condiciones ideales de las fotos entrenadas y de las imágenes de entrada, puesto que ambas imágenes corresponden a la misma, una foto extraída de un sitio externo, por lo tanto, son independientes de la iluminación, de la posición del individuo y de la distancia con respecto a la cámara.

**Estudio N°2:** El segundo set de pruebas está compuesto por 33 clases, que fueron entrenadas de fotos tomadas por una cámara digital. En este caso agregamos rostros sacados de internet para completar nuestra base de datos.

****

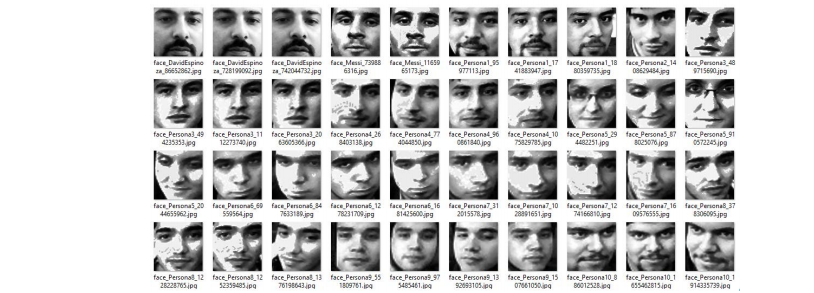
Resultados del estudio:

****

El éxito de este estudio se debe a la cantidad de fotos por clase y al nulo cambio de luz.

**Estudio N° 3:**

Para la prueba número 3 se registraron los rostros de 11 individuos, para cada uno de éstos se les tomó 3 fotos, 1 de rostro frontal y 2 con leves inclinaciones hacia los costados, las cuales posteriormente fueron entrenadas y almacenadas en la base de datos local. El registro de cada persona se realizó por medio del nombre: Persona 1, Persona 2 y así sucesivamente hasta la Persona número 11. El algoritmo utilizado en esta prueba fue Local Binary Pattern, se decidió usar éste por los buenos resultados obtenidos en las pruebas anteriores. En la siguiente imagen se visualizará de mejor manera lo explicado.

****

Los resultados obtenidos durante la prueba número 3 fueron de cierta manera los esperados dado el algoritmo utilizado. Se situó al individuo frente a la cámara web más otros rostros que fueron buscados en la red. para poder realizar el reconocimiento de su rostro, postura ideal para poder realizar el estudio. Los resultados fueron los siguientes:

* De 11 Personas 7 fueron reconocidas de manera correcta. (63.63%)
* 2 personas nunca fueron reconocidas por el nombre que se le había asignado. (18.18%)
* 1 persona no fue reconocida ni detectada por la toma de fotografías realizada anteriormente. (9.09%)
* 1 persona falto al estudio. (9.09%)

Como conclusión a la prueba realizada podemos referirnos a que el algoritmo Local Binary Pattern es el algoritmo que mejor funcionó en el ámbito de reconocimiento durante las 3 pruebas realizadas en el proceso de estudio. Éste funciona de gran manera generando un grado de acierto de aproximadamente mayor a 80%.

**Conclusiones**

El ser humano en su quehacer diario siempre ha buscado aumentar la seguridad en lo referente a bienes que poseen y que pudieran ser robados por medio de una simple contraseña o una clonación de la tarjeta. El reconocimiento facial le da al usuario lo que realmente necesita, más seguridad y sentir la confianza de que no les pasará nada a sus cuentas o los bienes con que interactúa. Cada sistema biométrico posee características que lo llevan a crear quiebres en su seguridad ya sea por ejemplo la clonación de una mano para la huella dactilar o un simple lente de contacto que refleje el iris de otra persona. En la actualidad no existe una técnica de reconocimiento que en realidad pueda ser considerada como la mejor, todo depende de las condiciones en la que se quiera implementar el sistema, sin embargo, la herramienta biométrica de reconocimiento facial es una casi impenetrable. El reconocimiento facial no requiere un contacto físico con la persona solo una cámara web y un software eficiente y es lo que se intentará implementar en este proyecto.

Ha sido una experiencia muy satisfactoria trabajar con todo esto, abriéndome un mundo de posibilidades con las cuales trabajar. El mundo de la inteligencia artificial es muy grande y una nueva revolución para la humanidad. Por eso me gustaría experimentar más con ello. Gracias a todo este proyecto en el que me he visto envuelto, viendo estas nuevas alternativas de solución, solo me queda pensar que este es el camino el que nos llevara hacia una nueva transformación en los procesos de los sistemas de información y toda la tecnología que actualmente existe.

**Recomendaciones**

Para que un sistema sea perfecto y funcione de forma correcta se debe ir mejorando y actualizando hasta quedar un prototipo dispuesto a realizar sus funcionalidades de manera correcta y con pequeñas tazas de errores. Es por esto que el sistema no queda exento de estos problemas los cuales se explicarán detalladamente a continuación:

# Primero que todo se debe optimizar el sistema para que cuando haya muchas imágenes entrenadas en la base de datos local, el programa no sufra de lentitud al momento de reconocer al individuo, esto se produce posiblemente por la cantidad de veces que recorre las imágenes entrenadas, por ejemplo, en un segundo.

# Mejoras en el apartado de interfaz para que ésta sea más acorde a la tecnología que es usada en la actualidad, por ejemplo, colores o ventanas más llamativas

# Se debe mejorar el ámbito de almacenamiento de fotografías entrenadas en la base de datos, esto para optimizar el reconocimiento facial, a fin de mejorar el recorrido del algoritmo por las imágenes, permitiendo recorrer por imágenes candidatas y no por el número de imágenes totales. Algunas soluciones para este problema serían:

* Agrupar imágenes por medio del promedio de umbral que posee cada una para así generar una comparación más fluida
* Agrupar las imágenes por carpeta, siendo ésta una por persona para poder realizar la búsqueda de manera más rápida

# Mejorar el momento de generar el reconocimiento del rostro de los individuos para que éste reconozca con una menor tasa de error y así poseer un software más confiable dentro de lo que se requiera en su uso

# **Anexos**

**Uso del programa**

Para el uso correcto del programa de sistema de reconocimiento facial se deben seguir los siguientes pasos.

* Se debe verificar que la cámara web pueda detectar el rostro de la persona para tener una correcta posición de la cara del individuo en ésta.
* Luego debemos entrenar las imágenes, esto se realiza en la interfaz de entrenamiento en donde primero que todo se debe escribir el nombre del individuo a entrenar para luego, teniendo el rostro d la persona detectado en la cámara web, presionar el botón de agregar imagen lo cual almacenara la imagen en tono de grises en una carpeta interna dentro del proyecto, cabe destacar que esta carpeta puede ser modificada dentro del código del proyecto para re direccionarla a cualquier sitio de nuestro computador.
* En la interfaz de entrenamiento como se explicó anteriormente se pueden ejecutar opciones extras para aso ahorrar tiempo en cada entrenamiento.
* Terminado el proceso de entrenamiento se procede a elegir a través del botón Algoritmo el tipo de algoritmo que se va a utilizar para el reconocimiento de rostro del individuo, como se explicó anteriormente se tienen 4 algoritmos de siendo cada uno de éstos de alguna forma aptos para el reconocimiento facial.
* Ya eligiendo el algoritmo a utilizar se podrá visualizar el nombre del individuo, si es que fue correcto o no, encima del recuadro de detección, con esto se podrán realizar estudios relacionados con cada algoritmo de reconocimiento