

ANÁLISIS COMPARATIVO DE HERRAMIENTAS DISPONIBLES PARA EL
DESARROLLO DE SISTEMAS CON TECNOLOGÍA DE RECONOCIMIENTO
FACIAL

SEBASTIÁN DAVID IBARRA PÁEZ

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PAMPLONA
2022

ANÁLISIS COMPARATIVO DE HERRAMIENTAS DISPONIBLES PARA EL
DESARROLLO DE SISTEMAS CON TECNOLOGÍA DE RECONOCIMIENTO
FACIAL

SEBASTIÁN DAVID IBARRA PÁEZ

TRABAJO DE GRADO PARA OPTAR POR EL TÍTULO DE INGENIERO DE
SISTEMAS

TUTOR
JOSE ORLANDO MALDONADO BAUTISTA
PH.D. EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD DE PAMPLONA
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PROGRAMA DE INGENIERÍA DE SISTEMAS
PAMPLONA
2022

TABLA DE CONTENIDO

RESUMEN.....	11
ABSTRACT	11
CAPÍTULO 1: INTRODUCCIÓN	12
1.1 PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN	12
1.2 DELIMITACIÓN	13
1.2.1 Objetivo General	13
1.2.2 Objetivos Específicos.....	13
1.3 ESTADO DEL ARTE	14
1.3.1 en el contexto Ucraniano	14
1.3.2 en el contexto Kazajo.....	14
CAPÍTULO 2: MARCO TEÓRICO	15
2.1 TECNOLOGÍA DE RECONOCIMIENTO FACIAL.....	15
2.1.1 Detección de rostro y puntos de referencia	15
2.1.2 Normalización del rostro	15
2.1.3 Extracción de características	16
2.1.4 Reconocimiento	16
2.2 ANÁLISIS COMPARATIVO	16
2.3 PRUEBA DE RENDIMIENTO	16
2.4 USABILIDAD	17
CAPÍTULO 3: METODOLOGÍA Y RESULTADOS.....	18
3.1 CONSULTA DE HERRAMIENTAS.....	18
3.1.1 herramientas gratuitas	18
3.1.2 herramientas de pago	19
3.2 DEFINICIÓN DE CRITERIOS Y SELECCIÓN DE HERRAMIENTAS A ESTUDIAR	20
3.3 DEFINICIÓN DE CRITERIOS PARA LA REALIZACIÓN DE PRUEBAS DE RENDIMIENTO	21
3.3.1 Definición de método de operación de las herramientas	21

3.3.2 Creación de base de datos de rostros	21
3.3.3 División de base de datos creada	21
3.3.4 Definición de método de trabajo de las herramientas	22
3.3.5 Medición de rendimiento con respecto a la detección de rostros	22
3.3.6 Medición de rendimiento con respecto al reconocimiento de rostros	23
3.3.7 Comportamiento de las herramientas frente a rostros de personas no incluidas en la galería	23
3.3.8 Realización de prueba estandarizada y cálculo del resultado final	24
3.4 Procedimiento Prueba Estandarizada	25
3.4.1 Descripción del entorno de trabajo	25
3.4.2 Breve descripción de las herramientas seleccionadas	25
3.4.3 Creación de base de datos de rostros:	26
3.4.4 Definición de datos para galería y entrada para la base de datos creada	26
3.4.5 Descripción de base de datos pública obtenida:.....	27
3.4.6 Prueba de rendimiento para Compreface	27
3.4.7 Prueba de rendimiento para Azure Face Api	31
3.4.8 Prueba de rendimiento para Face Recognition.....	34
3.4.9 Cálculo del resultado final.....	37
3.5 DEFINICIÓN DE CRITERIOS PARA LA USABILIDAD	38
3.5.1 Definición de funcionalidades del prototipo de Software de control de asistencia.....	38
3.5.2 Toma de datos para analizar la usabilidad	39
3.5.3 Medición de la usabilidad.....	39
3.6 PROCEDIMIENTO PARA LA EVALUACIÓN DE LA USABILIDAD	39
3.6.1 Desarrollo e implementación de prototipos de software de control de asistencia.....	39
3.6.2 Toma de datos.....	43
3.6.3 Aspectos que se observaron	47
3.6.4 Análisis de resultados:	48
CAPÍTULO 3: CONCLUSIONES	49

CAPÍTULO 4: RECOMENDACIONES	50
CAPÍTULO 5: BIBLIOGRAFÍA	51
CAPÍTULO 6: ANEXOS	53

LISTA DE TABLAS

Tabla 1. Cuadro comparativo entre herramientas de pago	20
Tabla 2. Cuadro comparativo entre herramientas gratuitas	20
Tabla 3. Descripción de llamados declarados para la creación de una persona en Compreface	27
Tabla 4. . Descripción de llamados declarados para la detección en Compreface	27
Tabla 5. . Descripción de llamados declarados para la detección en Compreface	28
Tabla 6. Resultados para el proceso de detección Compreface.....	29
Tabla 7. Resultados para el reconocimiento Compreface	29
Tabla 8. Resultados para el proceso de detección Azure Face Api.....	32
Tabla 9. Resultados para el reconocimiento Azure Face Api	33
Tabla 10. Resultados para el proceso de detección Face Recognition	35
Tabla 11. Resultados para el reconocimiento Face Recognition	36
Tabla 12. Recopilación de resultados obtenidos durante las pruebas de rendimiento	38
Tabla 13. Asignación de puntos para el resultado final.....	38
Tabla 14. Compilación de funciones y bibliotecas usadas para implementar Compreface en el prototipo #1.....	43
Tabla 15. Compilación de funciones y bibliotecas usadas para implementar Azure Face Api en el prototipo #2	43
Tabla 16. Compilación de funciones y bibliotecas usadas para implementar Face Recognition en el prototipo #3	45
Tabla 17. Recopilación de número de funciones y bibliotecas usadas por cada prototipo.....	47
Tabla 18. Respuestas a preguntas planteadas durante el desarrollo de los prototipos	47

LISTA DE GRÁFICAS

Gráfica 1. Valores de confidencialidad de rostros de base de datos externa vs rostros de personas en la galería para Compreface	30
Gráfica 2. Valores de confidencialidad de rostros de base de datos externa vs rostros de personas en la galería para Azure Face Api	33
Gráfica 3. Valores de confidencialidad de rostros de base de datos externa vs rostros de personas en la galería para Face Recognition.....	37

LISTA DE FIGURAS

Figura 1. Descripción de los pasos para el reconocimiento facial. Adaptado al español de [3]	16
Figura 2. Ejemplo de tomas de la base de datos creada.	26
Figura 3. Diagrama de casos de uso que describe el funcionamiento de los prototipos	40
Figura 4. Modelo relacional de la base de datos creada para cada herramienta...	41
Figura 5. Captura de pantalla menú para la creación de un estudiante.	41
Figura 6. Captura de pantalla del menú de control de asistencia	42

RESUMEN

En este trabajo se realizó un análisis comparativo entre tres herramientas seleccionadas que permiten el desarrollo de sistemas con tecnología de reconocimiento facial.

Se definieron qué criterios se deben utilizar para la realización de pruebas de rendimiento y la evaluación de la usabilidad. Se llevó a cabo la realización de una prueba estandarizada, que incluye la realización de las mismas pruebas de rendimiento por cada herramienta y su posterior análisis y comparación.

También se define un conjunto de criterios para evaluar la usabilidad de las herramientas desde el punto de vista del desarrollador, los cuales se apoyan de la implementación de un prototipo de software de control de asistencia por cada herramienta. Al final se elabora un cuadro comparativo para verificar el cumplimiento de los diferentes criterios en cada herramienta.

Palabras Clave: Tecnología de Reconocimiento Facial, Análisis comparativo, Prueba de rendimiento, Usabilidad.

ABSTRACT

In this work, a comparative analysis was made between three selected tools that allow the development of systems with facial recognition technology.

The criteria to be used for performance testing and usability evaluation were defined. A standardized test was carried out, which includes performing the same performance tests for each tool for further analysis and comparison.

A set of criteria is also defined to evaluate the usability of the tools from the developer's point of view, which are supported by the implementation of a prototype time attendance software for each tool. At the end, a comparative table is elaborated to verify the fulfillment of the different criteria for each tool.

Keywords: Facial Recognition Technology, Comparative analysis, Performance test, Usability.

CAPÍTULO 1: INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

El reconocimiento facial es una tecnología que día a día va obteniendo más fuerza debido a la gran utilidad que esta posee, como la autenticación de usuarios en una aplicación, la seguridad en lugares públicos, identificación de documentos, entre otros.

Para el año 2021 hay una gran cantidad de herramientas que pueden ser utilizadas para implementar la tecnología de reconocimiento facial que desarrolladores pueden escoger para implementar en sus proyectos.

Un estudio comparativo entre algunas de estas herramientas ayudará a evaluar cuál de estas posee un mejor rendimiento a la hora de reconocer un rostro, teniendo en cuenta resultados obtenidos al hacerse pruebas de rendimiento. También se evaluará cuál de estas posee una mejor usabilidad, la cual es la capacidad que poseen las herramientas para ser entendidas, aprendidas y usadas cuando se usan bajo determinadas condiciones.

Teniendo en cuenta los resultados del estudio comparativo, este proyecto ayudará en la escogencia de herramientas, teniendo en cuenta cuáles son sus ventajas y desventajas en los aspectos de rendimiento y usabilidad.

1.2 DELIMITACIÓN

1.2.1 Objetivo General

Realizar un análisis comparativo de distintas herramientas disponibles para implementar aplicaciones utilizando tecnología de reconocimiento facial.

1.2.2 Objetivos Específicos

- Consultar las distintas herramientas disponibles para la implementación de tecnología de reconocimiento facial.
- Establecer los criterios para la selección de las herramientas a evaluar, así como para la realización de las pruebas de rendimiento y usabilidad.
- Seleccionar al menos tres herramientas teniendo en cuenta los criterios seleccionados.
- Desarrollar una prueba estandarizada para comparar el rendimiento de los algoritmos para la detección y reconocimiento.
- Implementar un prototipo de software de control de asistencia por cada herramienta que sirva de apoyo para determinar la usabilidad de las herramientas seleccionadas.

1.3 ESTADO DEL ARTE

A continuación, se presenta una recopilación de artículos científicos y conferencias relacionadas con el tema Análisis comparativo de herramientas usadas en el reconocimiento facial. Se hace una revisión a nivel internacional, la cual incluye estudios expuestos en conferencias realizadas en Ucrania y Kazajistán.

1.3.1 en el contexto Ucraniano

1.3.1.1 Evaluación Y Comparación Del Rendimiento De Software Para El Reconocimiento Facial, Basado En Dlib Y Opencv

La idea principal del trabajo es comparar dos populares bibliotecas de visión por ordenador, que son OpenCV y dlib, explorar sus características, analizar los pros y contras de cada una de ellas y entender en qué situación cuál de ellos se adapta mejor. En él se Realiza un análisis comparativo del rendimiento de bibliotecas OpenCV y dlib basadas en el método HOG para búsqueda y posterior reconocimiento facial. En el trabajo se concluyó que la biblioteca OpenCV es más productiva, tiene mejor rendimiento para la detección de caras. También significa que con OpenCV, es mejor construir aplicaciones de reconocimiento para la plataforma IoT. [1]

1.3.2 en el contexto Kazajo

1.3.2.1 Estudio Comparativo De Algoritmos De Reconocimiento Facial

En este estudio se presentaron aspectos del rendimiento de los algoritmos Eigenfaces, Fisherfaces, e Histogramas de Patrones Binarios Locales en diferentes plataformas de desarrollo: Arm y procesadores Intel. En el estudio se comparan los métodos Eigenfaces, Fisherfaces e Histogramas de Patrones Binarios Locales (LBPH) en el reconocimiento de rostros en diferentes entornos de hardware. Se mide su éxito y rendimiento en el reconocimiento. Los resultados mostraron que Fisherface es más rápido, pero posee falta de éxito en el reconocimiento y LBPH ha tenido éxito en reconocimiento con gasto de rendimiento. [2]

CAPÍTULO 2: MARCO TEÓRICO

2.1 TECNOLOGÍA DE RECONOCIMIENTO FACIAL

La tecnología de reconocimiento facial es un sistema informático que permite identificar la identidad de una persona en base a sus patrones faciales como los ojos, la nariz, la boca, entre otros.

Stan Z. Li and Anil K. Jain en su libro “Handbook of Face Recognition” definen que la tecnología de reconocimiento facial puede operar de dos maneras: Verificación de rostros e identificación de rostros. La verificación de rostros implica una relación uno a uno que compara si los rostros presentes pertenecen a la misma persona, mientras que la identificación de rostros implica una consulta de uno a varios que compara un rostro desconocido con múltiples rostros en una base de datos para identificar la identidad de la persona[3].

Stan Z. Li and Anil K. Jain también mencionan que, para lograr su objetivo, la tecnología de reconocimiento facial generalmente requiere completar una serie de pasos:

2.1.1 Detección de rostro y puntos de referencia

En este paso se separa el rostro del fondo de la imagen. Mientras la detección del rostro provee una estimación de la localización y tamaño del rostro, la detección de puntos de referencia detecta puntos como la nariz, ojos, boca y contorno facial.

2.1.2 Normalización del rostro

La normalización de la cara se realiza para normalizarla geométrica y fotométricamente. El proceso de normalización geométrica transforma el rostro en un marco estándar mediante el recorte de la cara y la normalización fotométrica normaliza el rostro basándose en propiedades como la iluminación y la escala de grises.

2.1.3 Extracción de características

La extracción de rasgos faciales se realiza sobre el rostro normalizado para extraer información útil para distinguir los rostros de diferentes personas.

2.1.4 Reconocimiento

En el paso de Reconocimiento las características extraídas del rostro de entrada se comparan con uno o varios de los rostros registrados en la base de datos. El emparejador da como resultado "sí" o "no" para la verificación 1:1; para la identificación 1:N, la salida es la identidad del rostro de entrada cuando la coincidencia superior se encuentra con suficiente confianza, o "se desconoce" cuando la puntuación de la coincidencia de la punta está por debajo de un umbral.

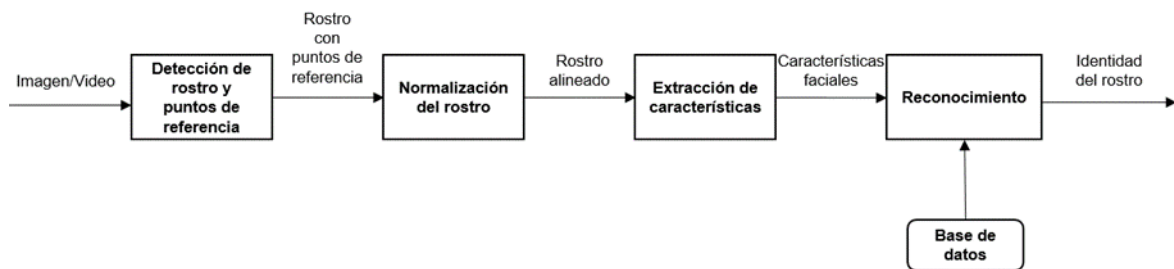


Figura 1. Descripción de los pasos para el reconocimiento facial. Adaptado al español de [3]

2.2 ANÁLISIS COMPARATIVO

Es un estudio que involucra distintas herramientas de tecnología de reconocimiento facial; en el cual se determinan sus características, fortalezas y debilidades para luego ser cotejadas y así determinar cuáles de ellas son mejores en ciertos campos. Para la realización del análisis comparativo se tienen en cuenta los resultados de pruebas de rendimiento y criterios de usabilidad.

2.3 PRUEBA DE RENDIMIENTO

Es una prueba que tiene como objetivo medir el rendimiento de las herramientas seleccionadas. La prueba de rendimiento se lleva a cabo en base a los criterios definidos. En la prueba estandarizada se realizan las pruebas de rendimiento por cada herramienta seleccionada y se hacen las comparaciones.

2.4 USABILIDAD

Según las normas ISO 25000, se define como “la capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones” [4]. La medición de usabilidad de las herramientas se lleva a cabo en base a los criterios definidos, y se apoya con la implementación de un prototipo de software de control de asistencia.

CAPÍTULO 3: METODOLOGÍA Y RESULTADOS

3.1 CONSULTA DE HERRAMIENTAS

Para la consulta de herramientas se investigó en distintas páginas web. A continuación, se encuentra una breve descripción de las herramientas candidatas, divididas entre herramientas gratuitas y herramientas de pago:

3.1.1 herramientas gratuitas

3.1.1.1 Face Recognition

Es una herramienta de reconocimiento facial desarrollada por Ageitei que permite reconocer y manipular rostros desde Python o desde la línea de comandos. La herramienta está construida basada en la biblioteca Dlib, y según el sitio web del Python Package Index (PyPi) tiene una precisión del 99,38% en la prueba de referencia Labeled Faces in the Wild. Esta posee 45.000 estrellas en Github en el momento de la realización del proyecto. [5]

3.1.1.2 Compreface

Exadel CompreFace es una herramienta de reconocimiento facial gratuita y de código abierto basada en Docker. Esta proporciona una API REST y SDK para Javascript y Python para las funciones de reconocimiento facial, verificación facial, la detección de caras, detección de puntos de referencia, la edad y el reconocimiento de género. Esta herramienta está basada en métodos y bibliotecas como Facenet e InsightFace. CompreFace. Según su página oficial, muestra una precisión del 99,83% en el conjunto de datos Labeled Faces in the Wild. Esta posee 2.000 estrellas en Github en el momento de la realización del proyecto. [6] [7]

3.1.1.3 DeepFace

Deepface es un framework de reconocimiento facial y análisis de atributos faciales como la edad, género, emoción y raza para el lenguaje de programación Python. Es un marco de reconocimiento facial híbrido que envuelve modelos de última generación como VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace,

DeepID, ArcFace y Dlib. La biblioteca se basa principalmente en TensorFlow y Keras. Esta posee 4.000 estrellas en Github en el momento de la realización del proyecto. [8]

3.1.1.4 InsightFace

InsightFace es una herramienta de análisis facial profundo 2D&3D de código abierto que se trabaja en el lenguaje de programación Python, basada principalmente en PyTorch y MXNet. InsightFace implementa eficientemente una gran variedad de algoritmos de reconocimiento facial, detección de rostros y alineación de rostros, optimizados tanto para el entrenamiento como para el despliegue. Esta posee 12.100 estrellas en Github en el momento de la realización del proyecto.[9] [10]

3.1.2 herramientas de pago

3.1.2.1 Azure Face Api

Es la API de reconocimiento facial en la nube de Microsoft Azure. Esta provee algoritmos de inteligencia artificial que permiten detectar, reconocer y analizar rostros humanos en imágenes. Esta herramienta posee SDK para .net, Node.js, Java, Python y una API Rest. Azure posee un plan para estudiantes, el cual permite trabajar el plan pago de manera gratuita por 12 meses. [11]

3.1.2.2 BETAFACE Api

Betaface API es un servicio web de detección y reconocimiento facial en la nube. Permite escanear archivos de imágenes cargados o URLs de imágenes, encontrar caras y analizarlas. La API también proporciona servicios de verificación (comparación de rostros) e identificación (búsqueda de rostros), además de poder mantener múltiples bases de datos de reconocimiento definidas por el usuario. Funciona mediante Api Rest. Este servicio posee varios planes de suscripción, como el plan Free (50 imágenes por día, gratis), Freemium (500 imágenes por día + 0,035 euros por cada imagen extra, gratis) o el plan básico (40.000 imágenes por mes + 0,025 euros por cada imagen extra) que cuesta 199 euros al mes. [12]

3.1.2.3 Kairos

Es una herramienta de visión artificial en la nube que posee servicios como detección de rostros, identificación de rostros, verificación de rostros, detección de

edad, detección de género, coordenadas faciales, etc. Esta permite integrar reconocimiento facial via Cloud API o alojarlo en servidores locales. Esta posee varios planes de pago, de los cuales algunos poseen un tiempo de prueba gratuito de 14 días. [13]

3.2 DEFINICIÓN DE CRITERIOS Y SELECCIÓN DE HERRAMIENTAS A ESTUDIAR

Los criterios definidos para la selección de herramientas a estudiar fueron los siguientes:

- Comparación entre herramientas gratuitas con no gratuitas, para determinar si hay superioridad entre una y otra.
- Entre las herramientas de pago, que tuvieran un plan estudiantil sin costo, para probarla de manera gratuita.
- Entre las herramientas gratuitas, escoger herramientas con la más alta y la más baja popularidad, basándose en la cantidad de estrellas que estas poseen en GitHub, que fue usado para medir el grado de popularidad que tienen.

También se debe aclarar que la herramienta, por el simple hecho de funcionar permite hacer pruebas de rendimiento, y dependiendo de las herramientas seleccionadas se definen qué criterios se plantean para hacer el análisis comparativo.

Para la escogencia de las herramientas se hicieron los siguientes cuadros comparativos:

Tabla 1. Cuadro comparativo entre herramientas de pago

Herramienta	¿Posee plan estudiantil sin costo?
Azure Face Api	Si
Betaface Api	No
Kairos	No

Tabla 2. Cuadro comparativo entre herramientas gratuitas

Herramienta	# de estrellas en Github
Face Recognition	45.000
Compreface	2.000

Deepface	4.000
InsightFace	12.100

Teniendo en cuenta los criterios de selección definidos y lo descrito anteriormete, se seleccionaron las herramientas Compreface, Face Recognition y Azure Face Api.

3.3 DEFINICIÓN DE CRITERIOS PARA LA REALIZACIÓN DE PRUEBAS DE RENDIMIENTO

Los criterios definidos para la realización de pruebas de rendimiento fueron los siguientes:

3.3.1 Definición de método de operación de las herramientas

Para la realización de pruebas de rendimiento se estableció que se debe trabajar mediante el método de identificación de rostros (1: N) [3], lo que requiere el uso de una base de datos que incluya a múltiples personas.

3.3.2 Creación de base de datos de rostros

Para la realización de pruebas de rendimiento se estableció que se debe crear una base de datos de rostros de múltiples personas, los cuales estén organizados de acuerdo a su identidad. Por cada identidad, se deben tomar cinco fotos de distintas posiciones. Las posiciones definidas son: Frente, perfil izquierdo, perfil derecho, parte inferior del rostro y una foto en un entorno cualquiera.

La principal motivación de crear una base de datos propia, es para poder observar el comportamiento de las herramientas con respecto a la posición de los rostros definidos. Esta es la base de datos usada en la medición de rendimiento con respecto a la detección de rostros y en la medición de rendimiento con respecto al reconocimiento de rostros, descritas a continuación.

3.3.3 División de base de datos creada

Manuel Freire López, en su tesis de master titulado “Método de optimización para reconocimiento facial basado en la fusión de algoritmos y segmentación según las características de las imágenes”, [14] define que las imágenes con las cuales interactúa un sistema de reconocimiento facial son las siguientes:

- Imágenes de galería: Son aquellas imágenes de los rostros de los individuos cuyas identidades son conocidas y se encuentran catalogadas dentro del sistema de reconocimiento facial, es decir, forman parte de la base de datos de identidades conocidas por el sistema.
- Imágenes de entrada: Son aquellas imágenes nuevas que ingresan al sistema de reconocimiento facial y corresponden a imágenes de rostros cuya identidad se desconoce y de los cuales el sistema tiene que determinar la identidad correspondiente.

Teniendo en cuenta estos conceptos, se debe especificar qué conjunto de fotos de la base de datos creada pertenece a la galería y que conjunto de datos pertenece a la entrada.

3.3.4 Definición de método de trabajo de las herramientas

En [15] se define que un sistema de reconocimiento facial puede trabajar de dos maneras:

- Identificación Closed-set: El sistema siempre devuelve una identidad.
- Identificación Open-set: El sistema devuelve una identidad o indica que el rostro evaluado no corresponde a ninguna identidad. Requiere establecer un umbral de reconocimiento.

Para la realización de pruebas de rendimiento se estableció que las herramientas deben trabajar mediante identificación **Closed-set**, ya que los resultados del reconocimiento podrían variar si se establece un umbral.

3.3.5 Medición de rendimiento con respecto a la detección de rostros

Para medir el rendimiento con respecto a la detección de rostros se estableció que se deben tomar los datos definidos a continuación:

- Aciertos (A): número de veces en las que la herramienta detecta un rostro, y efectivamente es un rostro.
- Desaciertos (D): número de veces en el que el sistema no detecta un rostro donde hay un rostro.

- Incongruencia (I): número de veces en el que el sistema detecta un rostro donde no hay un rostro.

Para medir el rendimiento con respecto a la detección de un rostro se debe aplicar la fórmula de precisión, la cual indica el porcentaje de exactitud en que la herramienta detecta un verdadero rostro. La precisión se calcula de la siguiente manera:

$$Pd = \frac{A}{A + D}$$

3.3.6 Medición de rendimiento con respecto al reconocimiento de rostros

Para medir el rendimiento con respecto a el reconocimiento de rostros se estableció que se deben tomar los datos definidos a continuación:

- Aciertos (A): número de veces en las que la herramienta detecta y reconoce la identidad del rostro correspondiente.
- Desaciertos (D): número de veces en las que la herramienta no detecta el rostro o si detecta el rostro, pero no reconoce la identidad correspondiente.

Para medir el rendimiento con respecto al reconocimiento se debe aplicar la fórmula de precisión, la cual indica el porcentaje de exactitud que hay entre los aciertos sobre el total de rostros evaluados. La fórmula se calcula de la siguiente manera:

$$Pr = \frac{A}{A + D}$$

3.3.7 Comportamiento de las herramientas frente a rostros de personas no incluidas en la galería

Para observar el comportamiento de las herramientas con respecto a rostros de personas no incluidas en la galería se estableció que se debe obtener una base de datos pública y tomar los datos definidos a continuación:

- valores del parámetro de confiabilidad de la herramienta que retorna al reconocer cada rostro de las personas incluidas en la galería (V1).

- valores del parámetro de confiabilidad de la herramienta que retorna al reconocer cada rostro de las personas no incluidas en la galería (base de datos pública) (V2).

Con estos resultados se calcula la diferencia, la cual se calcula de la siguiente manera:

$$D = \left| \frac{1}{n} \sum_{i=1}^n V1i - \frac{1}{m} \sum_{i=1}^m V2i \right|$$

De manera simplificada:

$$D = |\overline{V1} - \overline{V2}|$$

Donde:

- n = número de rostros identificados en las imágenes de personas en la galería.
- m = número de rostros identificados en las imágenes de una base de datos externa.

La diferencia indica el nivel de seguridad que poseen las herramientas al retornar un resultado. Entre más alta la diferencia, más seguridad posee la herramienta.

3.3.8 Realización de prueba estandarizada y cálculo del resultado final

Para la realización de la prueba estandarizada se deben realizar las pruebas de rendimiento para cada una de las herramientas seleccionadas. Para realizar la comparación de rendimiento se definió un sistema de puntos en el que se tiene en cuenta los resultados de los cálculos realizados anteriormente. El procedimiento es el siguiente:

- A la herramienta con mayor precisión en la detección se le asignan 3 puntos, a la segunda 2 puntos y a la tercera 1 punto. Lo mismo aplica para la precisión del reconocimiento.
- A la herramienta con menor cantidad de incongruencias se le asignan 3 puntos, a la segunda 2 puntos y a la tercera 1 punto.
- A la herramienta con mayor diferencia se le asignan 3 puntos, a la segunda 2 puntos y a la tercera 1 punto.
- En caso de empate, se asigna la misma cantidad de puntos, dependiendo de la posición.

- Finalmente, se determina el resultado sumando los puntajes.

3.4 Procedimiento Prueba Estandarizada

3.4.1 Descripción del entorno de trabajo

- Computador con 16 GB Ram, procesador AMD Ryzen 5 3600, tarjeta gráfica Nvidia 1660 Super, SSD 512 GB almacenamiento.
- Sistema operativo Ubuntu 18.04
- Python 3.6

3.4.2 Breve descripción de las herramientas seleccionadas

- **Compreface:** Esta herramienta, desarrollada por Exadel, funciona mediante la creación de aplicaciones, las cuales pueden poseer múltiples servicios. Los servicios pueden ser de “reconocimiento” (1: N), “detección” y “verificación” (1:1). Para el servicio de reconocimiento se crean “Subjects”, o sujetos. A un sujeto se le pueden asignar varios rostros para el posterior reconocimiento. Se puede trabajar mediante Api Rest, SDK de Python o SDK de Javascript. [16] [17]
- **Face Recognition:** Esta herramienta, desarrollada por Ageitei permite almacenar información de rostros (“face encodings”) conocidos en una lista o en un Numpy Array, y no almacena los nombres de las personas, por lo que se debe implementar esta funcionalidad por separado. Debido a la construcción de la herramienta, y basándose en los ejemplos que provee la documentación, se concluyó que la herramienta por si sola solo permite que un rostro sea asignado por persona. [18] [19] [20]
- **Azure Face Api:** Esta herramienta, desarrollada por Microsoft, funciona mediante la creación de “PersonGroup” (grupo de personas) o “LargePersonGroup” (gran grupo de personas). Estos almacenan objetos “Person” (persona), cada uno de los cuales representa a una persona individual [21]. Un grupo de personas o un gran grupo de personas trabaja con un modelo de reconocimiento, el cual puede ser “recognition_01”, “recognition_02”, “recognition_03” o “recognition_04” [22]. Para la detección posee tres modelos: “detection_01”, “detection_02” y “detection_03” [23]. En la guía de inicio rápido se puede comprobar que cada objeto persona puede tener múltiples rostros. [24]

3.4.3 Creación de base de datos de rostros:

Para la creación de la base de datos se tomaron cinco fotos por persona, cada foto corresponde a una posición. En total fueron 32 personas voluntarias a las que se le tomaron las cinco fotos. En algunas imágenes aparecieron rostros de personas ajenas a las voluntarias, por lo que estos se censuraron por razones éticas. Por cada persona, las imágenes se organizaron de la siguiente manera:

- bd_01.jpg: Frente.
- bd_02.jpg: Perfil derecho.
- bd_03.jpg: Perfil izquierdo.
- bd_04.jpg: Parte inferior del rostro.
- bd_05.jpg: Foto en un entorno cualquiera.



Figura 2. Ejemplo de tomas de la base de datos creada.

3.4.4 Definición de datos para galería y entrada para la base de datos creada

Debido al funcionamiento de la herramienta Face Recognition, se definió las imágenes de posición de frente sean las imágenes de galería; las imágenes que corresponden a las otras posiciones quedan como imágenes de entrada.

3.4.5 Descripción de base de datos pública obtenida:

Como base de datos externa se usó de la base de datos Tufts Face Database la modalidad “TD_RGB_E”, el set 1 [25]. Esta se usará únicamente para medir el comportamiento de las herramientas frente a rostros de personas no incluidas en imágenes de entrada.

3.4.6 Prueba de rendimiento para Compreface

En la realización de este proyecto se trabajó esta herramienta mediante API rest junto con la Biblioteca Requests de Python, ya que el SDK de Python no permite trabajar con la versión más reciente [26]. Se instaló la versión 1.0.0, la más reciente a la realización de este proyecto.

A continuación, se describen los llamados declarados junto con una descripción de los elementos usados relevantes y una descripción de los elementos opcionales cuyos valores se dejaron por defecto.

A continuación, se describen los llamados declarados junto con una descripción de los elementos usados relevantes y una descripción de los elementos opcionales cuyos valores se dejaron por defecto.

Crear persona (subject):

Tabla 3. Descripción de llamados declarados para la creación de una persona en Compreface

Nombre del elemento	Tipo	Valor
Subject	String	Nombre de la persona
File	Image	Imagen de la galería que contiene el rostro de la persona

Este llamado se usa para entrenar la herramienta con imágenes que contengan rostros conocidos. Esta solo recibe imágenes que contenga un rostro.

Detección:

Tabla 4. . Descripción de llamados declarados para la detección en Compreface

Nombre del elemento	Tipo	Valor
---------------------	------	-------

File	Image	Imagen con rostro
------	-------	-------------------

Este llamado fue usado para validar la detección de rostros en las imágenes de la galería. Este devuelve un diccionario con las coordenadas de los rostros detectados en la imagen.

Reconocimiento:

Tabla 5. . Descripción de llamados declarados para la detección en Compreface

Nombre del elemento	Tipo	Valor
File	Image	Imagen de entrada
prediction_count (opcional)	integer	Número de predicciones retornadas. Su valor por defecto es 1

Este llamado devuelve un diccionario que contiene las coordenadas del rostro detectado, la identidad predicha de la persona y el valor de similaridad.

Con estos llamados se crearon las siguientes funciones:

- Automatización de guardado de resultados de validación para la detección de rostros en las imágenes de la galería
- Automatización de creación de subjects.
- Automatización de guardado de resultados para el reconocimiento.

Para almacenar los resultados de la detección y reconocimiento de imágenes de entrada se usó la biblioteca OpenCV únicamente para dibujar las ubicaciones de los rostros detectados en base a las coordenadas devueltas junto con la identidad predicha, los valores de similaridad se almacenaron en un documento de texto. Para el caso de la validación de imágenes de la galería solamente se dibujaron las ubicaciones de los rostros detectados. En la sección de anexos se puede encontrar el código completo para este procedimiento.

Los resultados obtenidos para la detección de rostros son los siguientes:

Tabla 6. Resultados para el proceso de detección Compreface

Ángulo	# Aciertos (A)	#Desaciertos (D)	#Incongruencias (I)
Frente (Galería)	32	0	0
Perfil Derecho	32	0	0
Perfil Izquierdo	32	0	0
Parte inferior rostro	32	0	1
Fondo cualquiera	32	0	0
Total	160	0	1

Aplicando la fórmula de precisión:

$$Pd = \frac{160}{160 + 0}$$

$$Pd = 1$$

Observando los resultados de la tabla 1 se puede observar que para las fotos de la galería el proceso de detección retornó resultados correctos, y el proceso de creación de personas se ejecutó de manera exitosa por cada una de las fotos. Por lo que se puede suponer que los rostros correspondientes se almacenaron correctamente para el posterior reconocimiento.

Para la toma de resultados para el reconocimiento no se tiene en cuenta las identidades arrojadas en las incongruencias.

Los resultados obtenidos para el reconocimiento son los siguientes:

Tabla 7. Resultados para el reconocimiento Compreface

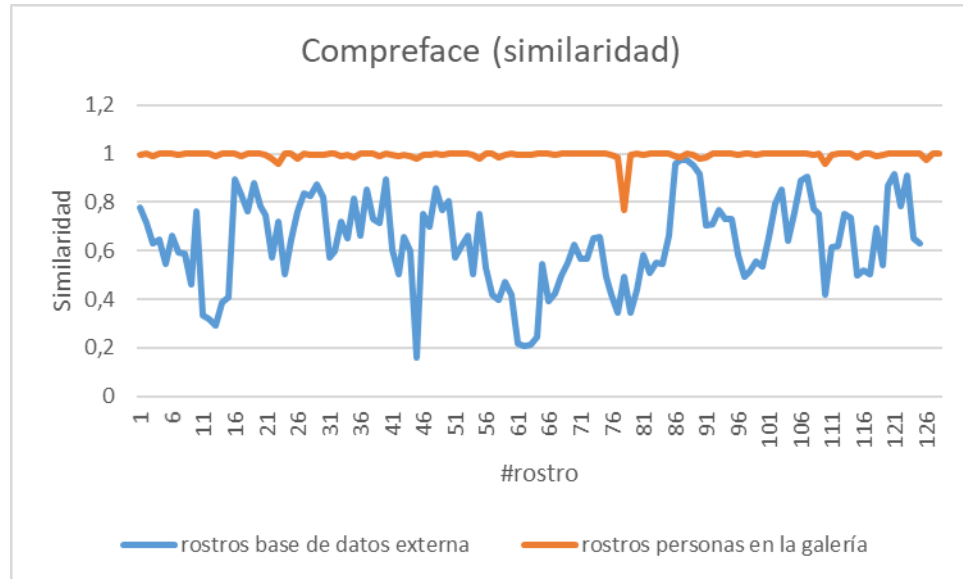
Ángulo	# Aciertos	# Desaciertos
Perfil derecho	32	0
Perfil Izquierdo	32	0
Parte inferior rostro	32	0
Fondo Cualquiera	32	0
Total	128	0

Aplicando la fórmula de precisión:

$$Pr = \frac{128}{128 + 0}$$

$$Pr = 1$$

Del comportamiento de la herramienta frente a rostros de personas no incluidas en imágenes de galería se obtuvo la siguiente información:



Gráfica 1. Valores de confiabilidad de rostros de base de datos externa vs rostros de personas en la galería para Compreface

En la gráfica se puede observar que, para rostros de personas de la galería, por lo general devuelve similaridades de valores cercanos a 1, dibujando una línea casi recta pero con un valle inferior a 0,8; pero para rostros de la base de datos externa los valores oscilan a través de la gráfica, con un pico a un valor muy cercano a 1. Esto quiere decir que si se decide trabajar la herramienta mediante validación Open-set, se dificultaría el establecimiento de un umbral.

Para valorar este comportamiento de manera cuantitativa se calculó la diferencia:

$$D = |0,993599688 - 0,6323368|$$

$$D = 0,361262888$$

3.4.7 Prueba de rendimiento para Azure Face Api

En la realización de este proyecto se trabajó mediante el SDK de Python.

A continuación, se describen las funciones de la herramienta implementadas. La implementación completa se encuentra en los anexos.

- **person_group.create:** Crea un Grupo de personas. se usa el modelo de reconocimiento "recognition_04"
- **person_group_person.create:** Crea una persona dentro de un grupo de personas. devuelve un person_id, que es generado automáticamente.
- **person_group_person.add_face_from_stream:** Asigna un rostro a una persona. Por defecto queda configurado para que solo se permita enviar un rostro por imagen.
- **person_group.train:** Entrena el grupo de personas para el posterior reconocimiento.
- **face.detect_with_stream:** Proceso de detección de rostros. Por cada rostro detectado retorna un face_id y sus coordenadas. Este proceso es requerido para el posterior reconocimiento. También se usa para validar la detección de rostros en las imágenes de la galería. Se usa el modelo de detección "detection_03".
- **face.identify:** Reconoce a una persona. Retorna el person_id y la confianza de cada persona. Se configuró para que el número máximo de candidatos por rostro que retorne sea 1.
- **person_group_person.get:** Obtener el nombre de las personas reconocidas en base a los person_id retornados por face.identify.

Con la ayuda de estas funciones se crearon las siguientes:

- Automatización de creación de personas.
- Automatización de guardado de resultados de validación para la detección de rostros en las imágenes de la galería.

- Automatización de asignación de imágenes de galería a cada persona.
- Automatización de guardado de resultados de reconocimiento

Para almacenar los resultados del reconocimiento y la detección se usó la biblioteca OpenCV únicamente para dibujar las ubicaciones de los rostros detectados en base a las coordenadas devueltas junto con la identidad predicha, los valores de confianza se almacenaron en un documento de texto. Para el caso de la validación de imágenes de la galería solamente se dibujaron las ubicaciones de los rostros detectados. En la sección de anexos se puede encontrar el código completo para este procedimiento.

Los resultados para la detección de rostros obtenidos son los siguientes:

Tabla 8. Resultados para el proceso de detección Azure Face Api

Ángulo	# Aciertos	# Desaciertos	# incongruencias
Frente	32	0	0
Perfil Derecho	32	0	0
Perfil Izquierdo	32	0	0
Parte inferior rostro	32	0	0
Fondo cualquiera	32	0	0
Total	160	0	0

Aplicando la fórmula de precisión:

$$Pd = \frac{160}{160 + 0}$$

$$Pd = 1$$

Observando los resultados de la tabla 1 se puede observar que para las fotos de la galería el proceso de detección retornó resultados correctos, y el proceso de creación de personas se ejecutó de manera exitosa por cada una de las fotos. Por lo que se puede suponer que los rostros correspondientes se almacenaron correctamente para el posterior reconocimiento.

Para la toma de resultados para el reconocimiento no se tiene en cuenta las identidades arrojadas en las incongruencias.

Los resultados obtenidos para el reconocimiento son los siguientes:

Tabla 9. Resultados para el reconocimiento Azure Face Api

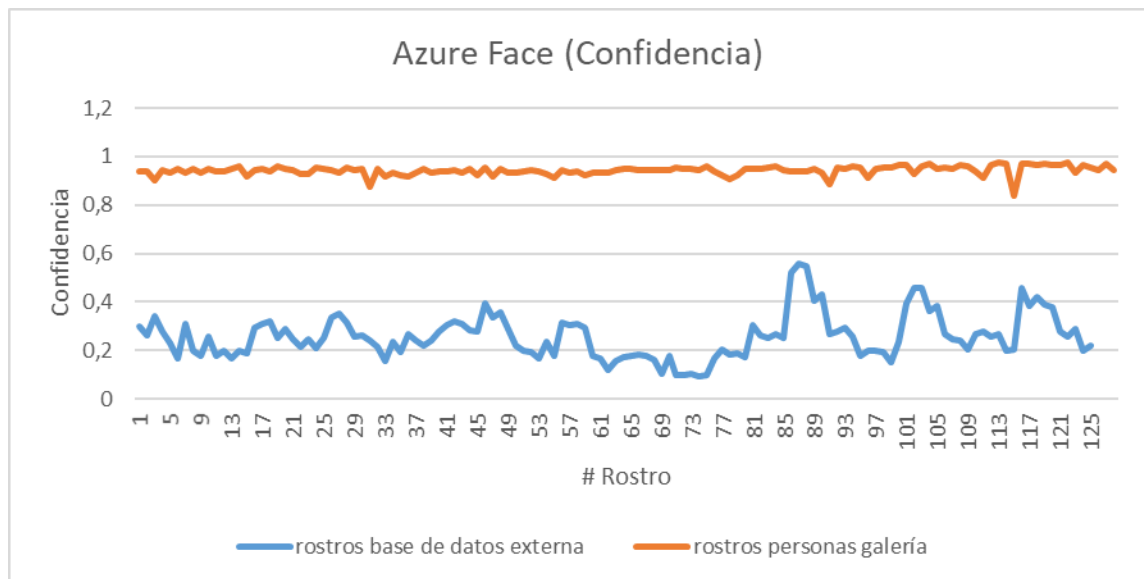
Ángulo	# Aciertos	# Desaciertos
Perfil Derecho	32	0
Perfil Izquierdo	32	0
Parte inferior del rostro	32	0
Fondo Cualquiera	32	0
Total	128	0

Aplicando la fórmula de la precisión:

$$Pr = \frac{128}{128 + 0}$$

$$Pr = 1$$

Del comportamiento de la herramienta frente a rostros de personas no incluidas en imágenes de galerías se obtuvo la siguiente información:



Gráfica 2. Valores de confidencialidad de rostros de base de datos externa vs rostros de personas en la galería para Azure Face Api

En la gráfica se puede observar que, para rostros de personas de la galería, por lo general devuelve confianzas de valores entre 0,8 y 1, dibujando una línea casi

recta; y para rostros de la base de datos externa los valores oscilan entre 0,1 y 0,5 aproximadamente. Esto quiere decir que si se decide trabajar la herramienta mediante validación Open-set, se podrían establecer umbrales mayores a 0,6.

Para valorar este comportamiento de manera cuantitativa se calculó la diferencia:

$$D = |0,942298359 - 0,25711472|$$

$$D = 0,685183639$$

3.4.8 Prueba de rendimiento para Face Recognition

A continuación, se describen las funciones más importantes. Para ver la implementación completa ver los anexos.

- **load_image_file:** Cargar imagen con rostros:
- **Face_Locations:** Recibe la imagen cargada. Retorna las coordenadas de los rostros detectados. Como parámetro extra se configuró `number_of_times_to_upsample = 3`, el cual indica el número de veces que se muestrea la imagen en busca de caras.
- **face_encodings:** Obtener información de rostros (Face Encodings), ya sea para rostros conocidos y desconocidos y recibe las coordenadas de rostros detectados. Los Face Encodings de rostros conocidos se guardan en una lista.
- **face_distance:** Dada una lista con Face Encodings de rostros conocidos, realiza la comparación con los Face Encodings de un rostro a reconocer. Devuelve una lista con las distancias euclidianas por cada cara, en orden. De esa lista se obtiene el índice con la menor distancia euclidiana.

Se trabajó de manera que los nombres de las personas conocidas se guardaran en una lista y se aseguró que las posiciones de los nombres en la lista coincidan con las posiciones de sus correspondientes face encodings en la lista con Face Encodings.

Con la ayuda de estas funciones se crearon las siguientes:

- Automatización de guardado de resultados de validación para la detección de rostros en las imágenes de la galería.

- Automatización de creación de lista de face encodings.
- Automatización de guardado de resultados de reconocimiento

Para almacenar los resultados de la detección y reconocimiento de imágenes de entrada se usó la biblioteca OpenCV únicamente para dibujar las ubicaciones de los rostros detectados en base a las coordenadas devueltas junto con la identidad predicha, los valores de distancia euclideana se almacenaron en un documento de texto. Para el caso de la validación de imágenes de la galería solamente se dibujaron las ubicaciones de los rostros detectados. En la sección de anexos se puede encontrar el código completo para este procedimiento.

Los resultados obtenidos para la detección son los siguientes:

Tabla 10. Resultados para el proceso de detección Face Recognition

Posición	# Aciertos	# Desaciertos	# Incongruencias
Frente	32	0	4
Perfil Derecho	30	2	9
Perfil Izquierdo	31	1	2
Parte inferior rostro	30	2	6
Fondo cualquiera	32	0	16
Total	155	5	37

Aplicando la fórmula de precisión:

$$Pd = \frac{155}{155 + 5}$$

$$Pd = 0,96875$$

En la tabla se observa que para la posición frente se encontraron cuatro incongruencias, pero detectó todos los rostros verdaderos correctamente; lo que significa que para el almacenamiento de los face encodings los rostros se deben escoger manualmente. En el proceso de detección de imágenes de la galería se observó que los aciertos corresponden a los rostros más grandes, por lo que estos fueron los que se seleccionaron. Para esto se usa el primer elemento de la lista

retornada por Face_Locations. Para validar esta afirmación se creó la función verifyFirst (ver anexo).

Para la toma de resultados para el reconocimiento no se tiene en cuenta las identidades arrojadas en las incongruencias.

Los resultados para el reconocimiento son los siguientes:

Tabla 11. Resultados para el reconocimiento Face Recognition

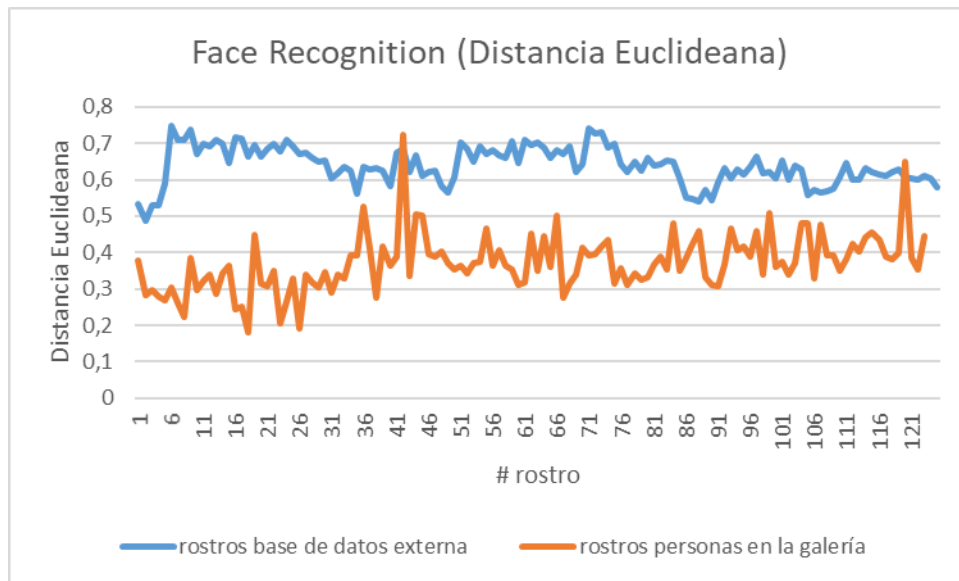
Posición	# Aciertos	# Desaciertos
Perfil Derecho	30	2
Perfil Izquierdo	31	1
Parte inferior rostro	29	3
Fondo cualquiera	32	0
Total	122	6

Aplicando la fórmula de precisión:

$$Pr = \frac{122}{122 + 6}$$

$$Pr = 0,953125$$

Del comportamiento de la herramienta frente a rostros de personas no incluidas en imágenes de galería se obtuvo la siguiente información:



Gráfica 3. Valores de confiabilidad de rostros de base de datos externa vs rostros de personas en la galería para Face Recognition

En la gráfica se puede observar que, para rostros de personas de la galería, las distancias euclidianas oscilan entre 0,2 y 0,7 aproximadamente. Para rostros de personas de la base de datos externa las distancias euclidianas oscilan entre 0,5 y 0,75 aproximadamente, incluso ambas líneas se llegan a tocar en dos oportunidades. Esto quiere decir que si se decide trabajar la herramienta mediante validación Open-set, se dificultaría el establecimiento de un umbral, y podría ser propensa a arrojar resultados erróneos.

Para valorar este comportamiento de manera cuantitativa se calculó la diferencia:

$$D = |0,371249153 - 0,639482375|$$

$$D = 0,268233222$$

3.4.9 Cálculo del resultado final

Recopilando los resultados:

Tabla 12. Recopilación de resultados obtenidos durante las pruebas de rendimiento

Herramienta	Precisión detección	Precisión reconocimiento	# Incongruencias	Diferencia
Compreface	1	1	1	0,361262888
Azure Face Api	1	1	0	0,685183639
Face Recognition	0,96875	0,953125	37	0,268233222

Aplicando Puntajes

Tabla 13. Asignación de puntos para el resultado final

Herramienta	Precisión detección	Precisión reconocimiento	# Incongruencias	Diferencia	Total
Compreface	3	3	2	2	10
Azure Face Api	3	3	3	3	12
Face Recognition	2	2	1	1	6

Teniendo en cuenta estos resultados, se puede concluir que Azure Face Api posee mejor rendimiento con respecto a los criterios seleccionados. Seguido de Compreface y Face Recognition queda en tercer lugar.

3.5 DEFINICIÓN DE CRITERIOS PARA LA USABILIDAD

3.5.1 Definición de funcionalidades del prototipo de Software de control de asistencia

Se definió que el prototipo de software de control de asistencia por cada herramienta debe tener las siguientes funcionalidades:

- Ingresar estudiantes. Requiere el almacenamiento de información del rostro del estudiante.
- Eliminar estudiantes. Requiere la eliminación de la información del rostro del estudiante.
- Crear clases.

- Controlar asistencia de estudiantes por cada clase. Esto incluye registrar la hora de entrada y la hora de salida de cada estudiante. El registro de la hora de entrada y hora de salida se debe hacer mediante reconocimiento facial.

3.5.2 Toma de datos para analizar la usabilidad

Para el análisis de la usabilidad de las herramientas se deben recopilar que funciones o llamados se utilizaron para garantizar el funcionamiento la herramienta de reconocimiento facial en cada prototipo de software de control de asistencia, incluidas funciones ajenas a la herramienta.

3.5.3 Medición de la usabilidad

Para medir la usabilidad de las herramientas se estableció que se debe determinar cuan complicada es la implementación de las herramientas. Entre menos funciones y menos bibliotecas se requieran para implementar una herramienta, menos complicada es. Esto parte desde la premisa de que una persona que use estas herramientas busque su implementación lo más rápido posible.

Teniendo en cuenta estas características, se debe describir cuál de estas fue más complicada de implementar y cual fue menos complicada de implementar.

3.6 PROCEDIMIENTO PARA LA EVALUACIÓN DE LA USABILIDAD

3.6.1 Desarrollo e implementación de prototipos de software de control de asistencia

Se desarrollaron e implementaron tres prototipos de software de control de asistencia. Los prototipos se nombraron de la siguiente manera:

- Prototipo #1: Utiliza Compreface para funcionalidades relacionadas con reconocimiento facial.
- Prototipo #2: Utiliza Azure Face Api para funcionalidades relacionadas con reconocimiento facial.
- Prototipo #3: Utiliza Face Recognition para funcionalidades relacionadas con reconocimiento facial.

Para el desarrollo de los prototipos se usó el framework para creación de aplicaciones web Flask y el patrón de arquitectura de software Modelo-Vista-Controlador. A continuación, se muestra el diagrama de casos de uso de los prototipos:

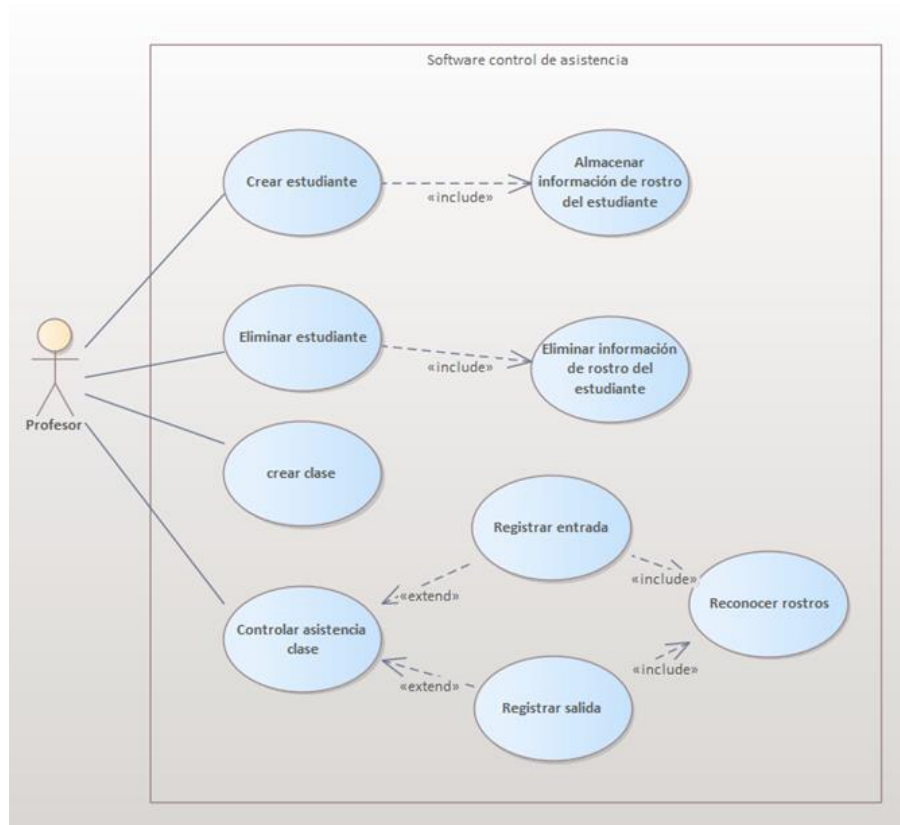


Figura 3. Diagrama de casos de uso que describe el funcionamiento de los prototipos

Las bases de datos de cada prototipo fueron creadas con el sistema manejador de base de datos Postgresql. El modelo relacional que describe las bases de datos se puede observar a continuación:

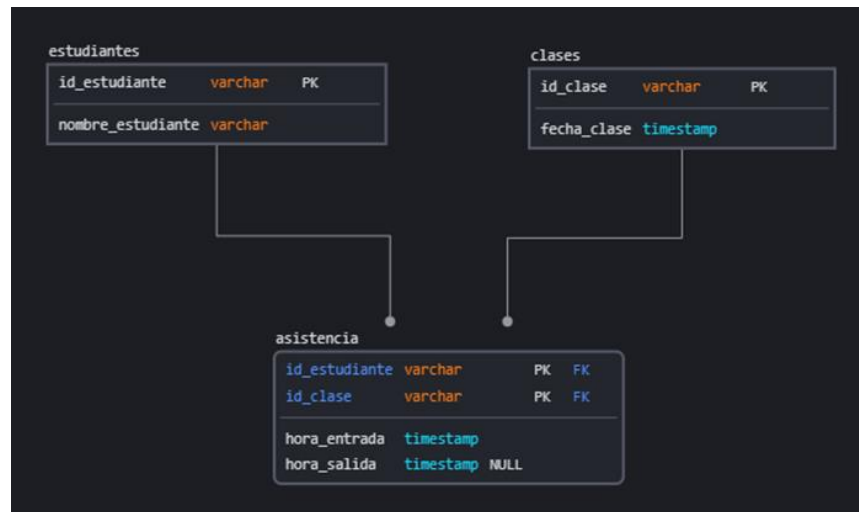


Figura 4. Modelo relacional de la base de datos creada para cada herramienta

Estudiantes

Identificación

Nombre

Foto Estudiante



[Crear Estudiante](#)

Figura 5. Captura de pantalla menú para la creación de un estudiante.

Para ingresar a un estudiante en el sistema se requiere que se le tome una foto, de la cual se extraen sus características faciales para el posterior reconocimiento.

Asistencia a clase

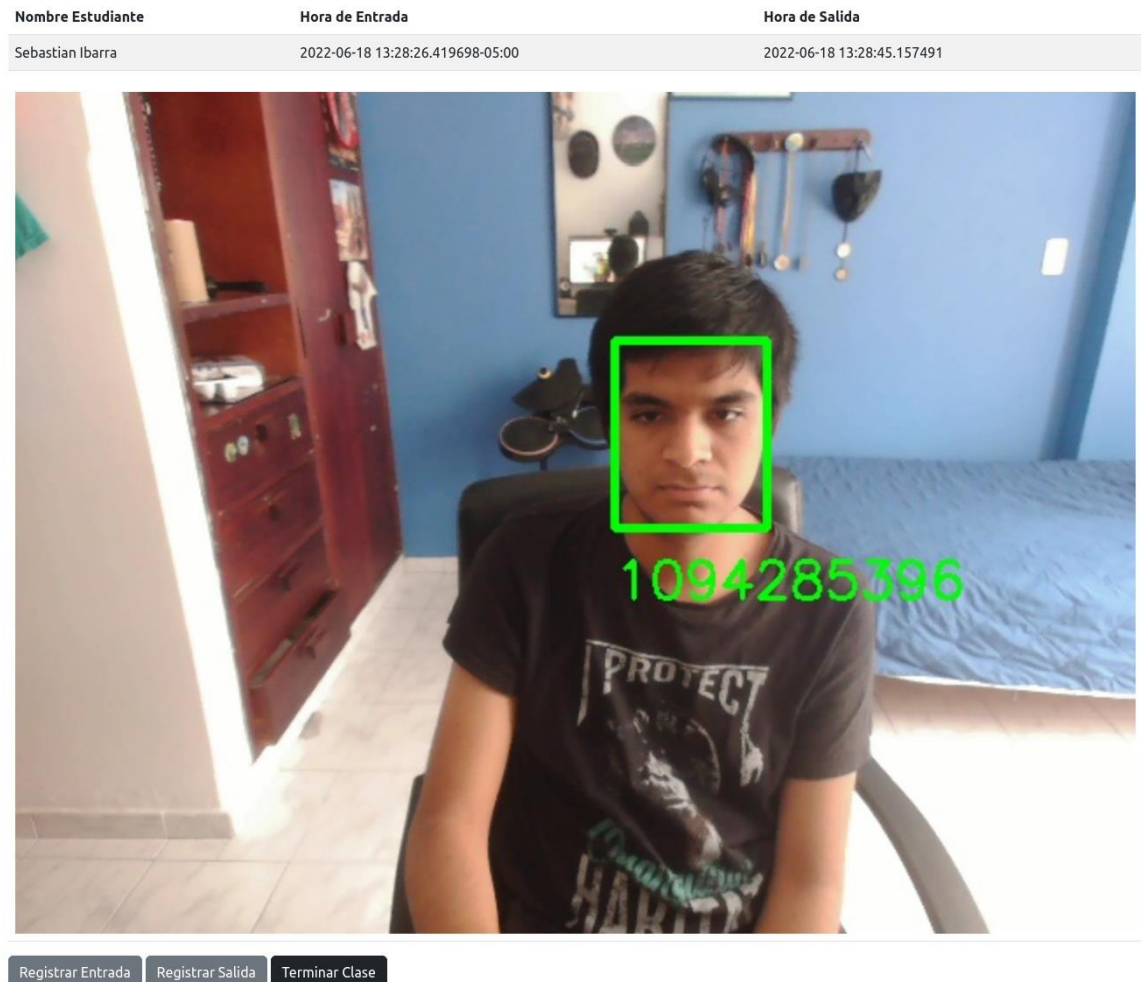


Figura 6. Captura de pantalla del menú de control de asistencia

Para la toma de asistencia el prototipo realiza el reconocimiento facial en vivo mediante el uso de una webcam (video), en la cual cada frame tomado por la webcam es tratado como si fuera una imagen. Al momento de presionar el botón “Registrar Entrada” se registra la entrada de los estudiantes que reconoce al momento de presionar el botón. Lo mismo aplica para el registro de salida.

En los anexos se pueden encontrar más imágenes de los prototipos en funcionamiento.

3.6.2 Toma de datos

3.6.2.1 Recopilación de funciones y bibliotecas

Prototipo #1:

Para el prototipo #1, para garantizar la implementación del reconocimiento facial, se utilizaron las siguientes funciones:

Tabla 14. Compilación de funciones y bibliotecas usadas para implementar Compreface en el prototipo #1

Biblioteca	Nombre	Breve descripción de su uso
Requests	Requests.post	Crear estudiante, enviar foto para el reconocimiento
	Requests.delete.	Eliminar estudiante
Sistema	Open	Abrir imagen a enviar
OpenCV	cv2.VideoCapture	Prender Camara
	read	Obtener frame
	imencode	Codificar imagen
Base64	b64encode	Guardar frame en buffer

Total de funciones: 7

Total de Bibliotecas : 4

Prototipo #2

Para el prototipo #2, para garantizar la implementación del reconocimiento facial, se utilizaron las siguientes funciones:

Tabla 15. Compilación de funciones y bibliotecas usadas para implementar Azure Face Api en el prototipo #2

Biblioteca	Nombre	Breve descripción de su uso
msrest.authentication	CognitiveServicesCredentials	Autenticacion para los servicios

		cognitivos de azure
azure.cognitiveservices.vision.face	FaceClient	Una API para la detección, verificación e identificación de rostros.(conexión)
	person_group.create	Crear grupo de personas
	person_group_person.create	Crear personas dentro del grupo de personas
	person_group_person.add_face_from_stream	Añadir rostro a persona (imagen) para el reconocimiento
	person_group.train	Entrenar el grupo de personas, prepararla para el reconocimiento
	person_group_person.delete	Borrar persona del grupo de personas
	face.detect_with_stream	Detectar rostros en imagen
	face.identify	Reconocimiento facial.

		retorna id generado internamente. recibe rostros detectados
	person_group_person.get	Obtener nombre de persona en base al id generado internamente
Sistema	Open	Abrir imagen para que se pueda enviar
OpenCV	cv2.VideoCapture	Prender Camara
	read	Obtener frame
	imencode	Codificar imagen
io	bytesio	Guardar frame en buffer

Número de funciones: 15

Número de bibliotecas: 5

Prototipo #3

Para el prototipo #3, para garantizar la implementación del reconocimiento facial, se utilizaron las siguientes funciones:

Tabla 16. Compilación de funciones y bibliotecas usadas para implementar Face Recognition en el prototipo #3

Biblioteca	Nombre	Breve descripción de su uso
Face Recognition	load_image_file	carga imagen
	face_locations	Obtener coordenadas de rostros (es opcional, pero

		se usa para mejor deteccion)
	face_encodings	Obtener informacion de rostros. se usa para rostros conocidos y para rostros desconocidos
	face_distance	Obtener distancia euclideanas para el reconocimiento (lista de face encodings vs face encoding de rostro a evaluar)
Numpy	savetxt	Guardar face encodings en archivo .csv
	loadtxt	cargar archivo .csv con face encodings
	empty	crear array con face encodings si es la primera vez que se usa el software
	vstack	añadir face encoding a array
	delete	borrar face encoding de array
	argmin	Obtener distancia euclidean mas pequeña
Sistema	creacion lista	Crear lista con nombres de personas
	append	Agregar nombre a lista
	del	Borrar nombre de lista
	open	abrir archivos .csv
Csv	csv writer	guardar nombres de personas en .csv
	csv reader	cargar nombres de personas en .csv
OpenCV	cv2.VideoCapture	Prender Camara
	read	Obtener frame
	cvtColor	Convertir frame de BGR a RGB

Número de funciones: 19

Número de bibliotecas: 5

Recopilando los resultados:

Tabla 17. Recopilación de número de funciones y bibliotecas usadas por cada prototipo

	Prototipo #1 (Compreface)	Prototipo #2 (Azure Face Api)	Prototipo #3 (Face Recognition)
Número de funciones	7	15	19
Número de bibliotecas	4	5	5

3.6.3 Aspectos que se observaron

Durante el desarrollo de los prototipos se plantearon las siguientes preguntas para cada herramienta:

- ¿Almacena información de rostros conocidos para posterior uso?
- ¿Guarda los nombres de rostros conocidos?
- Después del reconocimiento ¿La herramienta retorna toda la información necesaria?

Las respuestas son:

Tabla 18. Respuestas a preguntas planteadas durante el desarrollo de los prototipos

Pregunta	Compreface	Azure Face Api	Face Recognition
¿Almacena información de rostros conocidos para posterior uso?	Si	Si	No
¿Guarda los nombres de rostros conocidos?	Si	Si	No
Después del reconocimiento ¿La herramienta retorna toda la información necesaria?	Si	No	No

--	--	--	--

3.6.4 Análisis de resultados:

Con respecto a Compreface, se puede observar que se requiere dominio de Api Rest, pero a partir de ahí es una herramienta sencilla de implementar, ya que solo se requiere tener conocimiento de siete funciones y cuatro bibliotecas para implementar la herramienta, además de conocimientos propios del lenguaje Python como el manejo de diccionarios. En base a este análisis se determinó que esta es la herramienta menos complicada de implementar.

Con respecto a Azure Face Api, se puede observar que requiere el conocimiento de 15 funciones repartidas en 5 bibliotecas.

Con respecto a Face Recogniton, se puede observar que requiere el conocimiento de 19 funciones repartidas en 5 bibliotecas. También se pudo observar que las funcionalidades como el almacenamiento de información de rostros y los nombres se tuvieron que implementar con la ayuda de otras bibliotecas. En base a este análisis se determinó que esta es la herramienta más complicada de implementar.

CAPÍTULO 3: CONCLUSIONES

- De acuerdo a los estudios realizados se definieron parámetros para realizar pruebas de rendimiento y para evaluar la usabilidad.
- Se seleccionaron tres herramientas que permiten la implementación de tecnología de reconocimiento facial entre un listado consultado. Se hizo una prueba estandarizada para comparar el rendimiento de los algoritmos y se evaluó la usabilidad, todo esto en base a los criterios definidos.
- Se implementó un prototipo de software de control de asistencia por cada herramienta, en el cual cada prototipo posee el mismo diseño. De cada prototipo se extrajo el número de funciones y bibliotecas que se utilizaron para implementar correctamente la herramienta de reconocimiento facial correspondiente, así ayudando a la evaluación de la usabilidad.
- Se determinó que, teniendo en cuenta los resultados arrojados por la prueba estandarizada, Azure Face Api posee mejor rendimiento con respecto a la detección y reconocimiento de rostros bajo las condiciones trabajadas, seguido de Compreface y por última, Face Recognition.
- Se determinó que, teniendo en cuenta los resultados arrojados por la evaluación de la usabilidad, Compreface es la herramienta menos complicada de implementar bajo las condiciones trabajadas, seguido de Azure Face Api. Face Recognition es la herramienta más complicada de implementar.

CAPÍTULO 4: RECOMENDACIONES

Se recomienda para futuros proyectos la investigación de métodos para complementar la evaluación de la usabilidad, como tener en cuenta el procedimiento de instalación de una herramienta, la complejidad de las funciones que poseen o el estudio de la documentación.

CAPÍTULO 5: BIBLIOGRAFÍA

- [1] N. Boyko, O. Basystiuk, and N. Shakhovska, "Performance Evaluation and Comparison of Software for Face Recognition, Based on Dlib and Opencv Library," *Proc. 2018 IEEE 2nd Int. Conf. Data Stream Min. Process. DSMP 2018*, pp. 478–482, Oct. 2018, doi: 10.1109/DSMP.2018.8478556.
- [2] A. Özdil and M. M. Özbilen, "A survey on comparison of face recognition algorithms," 2014. doi: 10.1109/ICAICT.2014.7035956.
- [3] S. Z. Li and A. K. Jain, "Introduction," in *Handbook of Face Recognition*, Springer, London, 2011, pp. 1–15. doi: 10.1007/978-0-85729-932-1_1.
- [4] "Usabilidad." <https://iso25000.com/index.php/normas-iso-25000/iso-25010/23-usabilidad> (accessed Jun. 02, 2022).
- [5] "GitHub - ageitgey/face_recognition: The world's simplest facial recognition api for Python and the command line." https://github.com/ageitgey/face_recognition (accessed Jun. 05, 2022).
- [6] "CompreFace — Face Recognition Service | Exadel." <https://exadel.com/solutions/compreface/> (accessed Jun. 02, 2022).
- [7] "GitHub - exadel-inc/CompreFace: Leading free and open-source face recognition system." <https://github.com/exadel-inc/CompreFace> (accessed Jun. 05, 2022).
- [8] "GitHub - serengil/deepface: A Lightweight Face Recognition and Facial Attribute Analysis (Age, Gender, Emotion and Race) Library for Python." <https://github.com/serengil/deepface> (accessed Jun. 02, 2022).
- [9] "GitHub - deepinsight/insightface: State-of-the-art 2D and 3D Face Analysis Project." <https://github.com/deepinsight/insightface> (accessed Jun. 21, 2022).
- [10] "InsightFace: an open source 2D&3D deep face analysis library." <https://insightface.ai/> (accessed Jun. 02, 2022).
- [11] "Documentación de Face: inicios rápidos, tutoriales, referencia de API (Azure Cognitive Services) | Microsoft Docs." <https://docs.microsoft.com/es-es/azure/cognitive-services/face/> (accessed Jun. 02, 2022).
- [12] "Betaface API | Open API for face recognition." <https://www.betafaceapi.com/wpa/> (accessed Jun. 02, 2022).
- [13] "Kairos: Serving Businesses with Face Recognition." <https://www.kairos.com/> (accessed Jun. 02, 2022).
- [14] M. F. López, "Método de optimización para reconocimiento facial basado en la fusión de algoritmos y segmentación según las características de las imágenes," E.T.S. de Ingenieros Informáticos (UPM), 2016. [Online]. Available: <https://oa.upm.es/42933/>
- [15] E. Learned-Miller, G. B. Huang, A. RoyChowdhury, H. Li, and G. Hua, "Labeled Faces in the Wild: A Survey," *Adv. Face Detect. Facial Image Anal.*, pp. 189–248, Jan. 2016, doi: 10.1007/978-3-319-25958-1_8.
- [16] "CompreFace/Rest-API-description.md at master · exadel-inc/CompreFace ·

- GitHub.” <https://github.com/exadel-inc/CompreFace/blob/master/docs/Rest-API-description.md#managing-subjects> (accessed Jun. 03, 2022).
- [17] “CompreFace/How-to-Use-CompreFace.md at master · exadel-inc/CompreFace · GitHub.” <https://github.com/exadel-inc/CompreFace/blob/master/docs/How-to-Use-CompreFace.md> (accessed Jun. 03, 2022).
 - [18] “face_recognition package — Face Recognition 1.4.0 documentation.” https://face-recognition.readthedocs.io/en/latest/face_recognition.html (accessed Jun. 03, 2022).
 - [19] “face_recognition/face_recognition_knn.py at master · ageitgey/face_recognition · GitHub.” https://github.com/ageitgey/face_recognition/blob/master/examples/face_recognition_knn.py (accessed Jun. 03, 2022).
 - [20] “face-recognition · PyPI.” <https://pypi.org/project/face-recognition/> (accessed Jun. 02, 2022).
 - [21] “Conceptos del reconocimiento facial - Azure Cognitive Services | Microsoft Docs.” <https://docs.microsoft.com/es-es/azure/cognitive-services/face/concepts/face-recognition> (accessed Jun. 03, 2022).
 - [22] “Especificación de un modelo de reconocimiento: Face - Azure Cognitive Services | Microsoft Docs.” <https://docs.microsoft.com/es-es/azure/cognitive-services/face/face-api-how-to-topics/specify-recognition-model> (accessed Jun. 03, 2022).
 - [23] “Especificación de un modelo de detección: Face - Azure Cognitive Services | Microsoft Docs.” <https://docs.microsoft.com/es-es/azure/cognitive-services/face/face-api-how-to-topics/specify-detection-model> (accessed Jun. 05, 2022).
 - [24] “Inicio rápido: Uso de la biblioteca cliente de Face - Azure Cognitive Services | Microsoft Docs.” <https://docs.microsoft.com/es-es/azure/cognitive-services/face/quickstarts/client-libraries?tabs=visual-studio&pivots=programming-language-python> (accessed Jun. 03, 2022).
 - [25] K. Panetta *et al.*, “A Comprehensive Database for Benchmarking Imaging Systems,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 509–520, Mar. 2020, doi: 10.1109/TPAMI.2018.2884458.
 - [26] “GitHub - exadel-inc/compreface-python-sdk: Python SDK for CompreFace - free and open-source face recognition system from Exadel.” <https://github.com/exadel-inc/compreface-python-sdk> (accessed Jun. 05, 2022).

CAPÍTULO 6: ANEXOS

- Capturas de pantalla de Interfaz gráfica de los prototipos de control de asistencia

Software de Control de Asistencia

[Estudiante](#)

[Clase](#)

Estudiantes

	Nombre	Codigo
☐	Sebastian Ibarra	1094285396
☐	Elsa	789123

[Crear](#)

[Eliminar](#)

[Atras](#)

Estudiantes

Identificación

1094285396

Nombre

Sebas Ibarra

Foto Estudiante



Crear Estudiante

Clases

Fecha de la clase

- ☐ 2022-05-28 10:47:19.690193-05:00
- ☐ 2022-05-28 10:50:43.153969-05:00
- ☐ 2022-06-13 18:03:58.284502-05:00
- ☐ 2022-06-13 19:15:01.607875-05:00
- ☐ 2022-06-18 09:39:26.726582-05:00
- ☐ 2022-06-18 09:39:48.658723-05:00
- ☐ 2022-06-18 09:42:10.215740-05:00
- ☐ 2022-06-18 09:44:59.608930-05:00
- ☐ 2022-06-18 09:45:43.274949-05:00
- ☐ 2022-06-18 09:46:45.725587-05:00
- ☐ 2022-06-18 09:48:14.030717-05:00
- ☐ 2022-06-18 09:52:43.137630-05:00

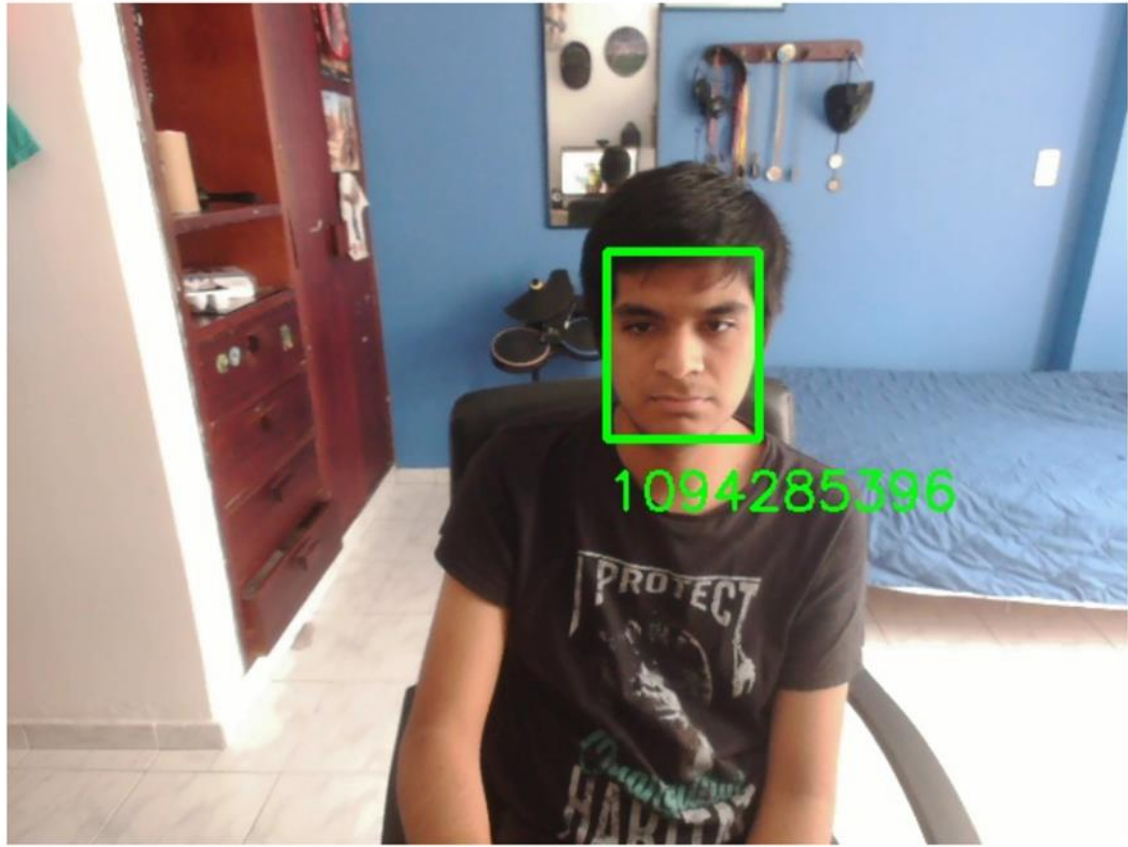
Crear

Ver Informacion

[Atras](#)

Asistencia a clase

Nombre Estudiante	Hora de Entrada	Hora de Salida
Sebastian Ibarra	2022-06-18 13:28:26.419698-05:00	2022-06-18 13:28:45.157491



[Registrar Entrada](#) [Registrar Salida](#) [Terminar Clase](#)

Informacion de clases

Id de clase	Fecha de la clase
8d71f7a1-3541-4828-8463-a53c058395f8	2022-05-28 10:47:19.690193-05:00

Nombre del estudiante	hora de entrada	hora de salida
Sebastian Ibarra	2022-05-28 10:47:27.964629-05:00	2022-05-28 10:48:09.773369

[Atras](#)

- Código para creación de la base de datos prototipos

```
CREATE TABLE estudiantes (  
    id_estudiante VARCHAR  
    ,nombre_estudiante VARCHAR NOT NULL  
);  
  
CREATE TABLE clases (  
    id_clase VARCHAR  
    ,fecha_clase TIMESTAMP WITH TIME zone NOT NULL  
);  
  
CREATE TABLE asistencia (  
    id_estudiante VARCHAR  
    ,id_clase VARCHAR  
    ,hora_entrada TIMESTAMP WITH TIME zone NOT NULL  
    ,hora_salida TIMESTAMP  
);  
  
ALTER TABLE estudiantes ADD CONSTRAINT estudiantes_pk PRIMARY KEY (id_estudiante);  
  
ALTER TABLE clases ADD CONSTRAINT clases_pk PRIMARY KEY (id_clase);  
  
ALTER TABLE asistencia ADD CONSTRAINT asistencia_clases_fk FOREIGN KEY (id_clase) REFERENCES clases (id_clase);  
  
ALTER TABLE asistencia ADD CONSTRAINT asistencia_estudiantes_fk FOREIGN KEY (id_estudiante) REFERENCES estudiantes (id_estudiante);  
  
ALTER TABLE asistencia ADD CONSTRAINT asistencia_pk PRIMARY KEY (  
    id_estudiante  
    ,id_clase  
);
```

- Enlace Prototipo #1 (Compreface)
[Hipervinculo](#)
- Enlace Prototipo #2 (Azure Face Api)
[Hipervinculo](#)
- Enlace Prototipo #3 (Face Recognition)
[Hipervinculo](#)

```
# -*- coding: utf-8 -*-  
"""
```

Created on Mon Jun 6 00:34:36 2022

```
@author: Sebastian  
"""
```

```
#bibliotecas
```

```
import requests
```

```
import os
```

```
import cv2
```

```
#_____Validación resultados de la galeria  
Compreface_____
```

```
#Detectar rostros
```

```
def detectFace(directory, url, headers):  
    image_file_descriptor = open(directory, 'rb')  
    files = {'file': image_file_descriptor}  
    response = requests.post(url, headers=headers, files=files)  
    image_file_descriptor.close()  
    return (response.json())
```

```
#Automatizar guardado de resultados
```

```
def saveResultsFront(url, headers):  
    for i in range(32):  
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'bd_01.jpg'  
        result = detectFace(directory, url, headers)  
        saveImageBounds(directory, result, 'frente/persona' + str(i+1) + '.jpg')
```

```
#dibujar area de rostros
```

```
def saveImageBounds(url, result, imgName):  
    image = cv2.imread(url)  
    for value in result['result']:  
        bounds = value['box']  
        cv2.rectangle(image, (bounds['x_max'],bounds['y_max']), (bounds['x_min'],bounds['y_min']), (0,255,0),3)  
    cv2.imwrite('/home/sebas/resultados deteccion/compreface/' + imgName, image)
```

```
detectionUrl = 'http://localhost:8000/api/v1/detection/detect'
```

```
detectionHeaders = {'x-api-key': '7b35d1b3-7f77-4b05-9646-6edc34a43b1c'}
```

```
saveResultsFront(detectionUrl, detectionHeaders)
```

```
#_____Preparación y resultados para el reconocimiento  
Compreface_____
```

```
#Crear subject y añadir imagen
```

```
def addImage(url, headers, directory, subject):  
    image_file_descriptor = open(directory, 'rb')  
    files = {'file': image_file_descriptor}  
    params={'subject': subject}  
    response = requests.post(url, headers=headers, params=params, files=files)  
    image_file_descriptor.close()  
    return (response.json())
```

```
#Crear todos los subjects
```

```
def assignImages(url, headers):  
    for i in range(32):  
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'bd_01.jpg'  
        subject = 'Persona'+str(i+1)  
        result = addImage(url, headers, directory, subject)  
        print(result)
```

#Reconocer persona que aparece en la imagen

```
def reconocerImagen(directory, url, headers):
    image_file_descriptor = open(directory, 'rb')
    files = {'file': image_file_descriptor}
    response = requests.post(url, headers=headers, files=files)
    image_file_descriptor.close()
    return (response.json())
```

#Automatizar guardado de resultados en un archivo .txt de perfil derecho

```
def saveResultsRight(url, headers):
    file = open("resultadosderecho.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/bd_02.jpg'
        result = reconocerImagen(directory, url, headers)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result['result']:
            file.write('Resultado: ' + value['subjects'][0]['subject'] + '. Nivel de similaridad: ' + str(value['subjects'][0]['similarity']) +
os.linesep)
            print(result['result'][0]['subjects'][0]['subject'] + directory + str(result['result'][0]['subjects'][0]['similarity']))
            saveImageBoundsNames(directory, result, 'derecho/persona' + str(i+1) + '.jpg')
    file.close()
```

#Automatizar guardado de resultados en un archivo .txt de perfil izquierdo

```
def saveResultsLeft(url, headers):
    file = open("resultadosizquierdo.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/bd_03.jpg'
        result = reconocerImagen(directory, url, headers)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result['result']:
            file.write('Resultado: ' + value['subjects'][0]['subject'] + '. Nivel de similaridad: ' + str(value['subjects'][0]['similarity']) +
os.linesep)
            print(result['result'][0]['subjects'][0]['subject'] + directory + str(result['result'][0]['subjects'][0]['similarity']))
            saveImageBoundsNames(directory, result, 'izquierdo/persona' + str(i+1) + '.jpg')
    file.close()
```

#Automatizar guardado de resultados en un archivo .txt de parte inferior

```
def saveResultsLower(url, headers):
    file = open("resultadosinferior.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/bd_04.jpg'
        result = reconocerImagen(directory, url, headers)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result['result']:
            file.write('Resultado: ' + value['subjects'][0]['subject'] + '. Nivel de similaridad: ' + str(value['subjects'][0]['similarity']) +
os.linesep)
            print(result['result'][0]['subjects'][0]['subject'] + directory + str(result['result'][0]['subjects'][0]['similarity']))
            saveImageBoundsNames(directory, result, 'inferior/persona' + str(i+1) + '.jpg')
    file.close()
```

#Automatizar guardado de resultados en un archivo .txt de toma aleatoria

```
def saveResultsRandom(url, headers):
    file = open("resultadosaleatorio.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/bd_05.jpg'
        result = reconocerImagen(directory, url, headers)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result['result']:
            file.write('Resultado: ' + value['subjects'][0]['subject'] + '. Nivel de similaridad: ' + str(value['subjects'][0]['similarity']) +
```

```
os.linesep)
    print(result['result'][0]['subjects'][0]['subject'] + directory + str(result['result'][0]['subjects'][0]['similarity']))
    saveImageBoundsNames(directory, result, 'aleatorio/persona' + str(i+1) + '.jpg')
file.close()
```

#encierra en un cuadro rostros encontrados y les pone el nombre

```
def saveImageBoundsNames(url, result, imgName):
    image = cv2.imread(url)
    for value in result['result']:
        bounds = value['box']
        cv2.rectangle(image, (bounds['x_max'],bounds['y_max']), (bounds['x_min'],bounds['y_min']),(0,255,0),3)
        cv2.putText(image, value['subjects'][0]['subject'], (bounds['x_min'] , bounds['y_max'] +40),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0),3)
    cv2.imwrite('/home/sebas/resultados reconocimiento/compreface/' + imgName, image)
```

#llamar funciones

#relacionados con imágenes de caras

imageUrl = 'http://localhost:8000/api/v1/recognition/faces'

imageHeaders = {'x-api-key': 'fc822e66-02cf-49bc-963c-e614e886e542'}

#Relacionados con reconocimiento de rostros

recognizeUrl = 'http://localhost:8000/api/v1/recognition/recognize'

recognizeHeaders = {'x-api-key': 'fc822e66-02cf-49bc-963c-e614e886e542'}

saveResultsRandom(recognizeUrl, recognizeHeaders)

saveResultsLower(recognizeUrl, recognizeHeaders)

saveResultsLeft(recognizeUrl, recognizeHeaders)

saveResultsRight(recognizeUrl, recognizeHeaders)

```
# -*- coding: utf-8 -*-  
"""
```

Created on Wed Nov 10 21:29:56 2021

```
@author: Sebastian  
"""
```

```
#bibliotecas
```

```
import requests
```

```
import os
```

```
import cv2
```

```
# _____ Obtener resultados base de datos externa  
Compreface _____
```

```
#Reconocer persona que aparece en la imagen
```

```
def recognizelImage(directory, url, headers):  
    image_file_descriptor = open(directory, 'rb')  
    files = {'file': image_file_descriptor}  
    response = requests.post(url, headers=headers, files=files)  
    image_file_descriptor.close()  
    return (response.json())
```

```
#Automatizar guardado de resultados en un archivo .txt
```

```
def saveResults(url, headers):  
    file = open("resultadosnobd.txt", "w")  
    similarities = []  
    for i in range(25):  
        for j in range(5):  
            directory = '/home/sebas/TD_RGB_E_Set1/'+str(i+1)+'TD_RGB_E_'+str(j+1)+'.jpg'  
            result = recognizelImage(directory, url, headers)  
            file.write('Archivo: ' + directory + os.linesep)  
            similarities.append(result['result'][0]['subjects'][0]['similarity'])  
            for value in result['result']:  
                file.write('Resultado: ' + value['subjects'][0]['subject'] + '. Nivel de similaridad: ' + str(value['subjects'][0]['similarity']) +  
os.linesep)  
                print(result['result'][0]['subjects'][0]['subject'] + directory + str(result['result'][0]['subjects'][0]['similarity']))  
                saveImageBoundsNames(directory, result, 'persona' + str(i+1) + str(j+1) + '.jpg')  
    file.close()  
    return similarities
```

```
#encierra en un cuadro rostros encontrados y les pone el nombre
```

```
def saveImageBoundsNames(url, result, imgName):  
    image = cv2.imread(url)  
    for value in result['result']:  
        bounds = value['box']  
        cv2.rectangle(image, (bounds['x_max'], bounds['y_max']), (bounds['x_min'], bounds['y_min']), (0,255,0),3)  
        cv2.putText(image, value['subjects'][0]['subject'], (bounds['x_min'], bounds['y_max'] +40),  
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0),3)  
        cv2.imwrite('/home/sebas/pruebas rostros no guardados/compreface/' + imgName, image)
```

```
#llamar funciones
```

```
#relacionados con subjects
```

```
subjectUrl = 'http://localhost:8000/api/v1/recognition/subjects/'
```

```
subjectHeaders = {'Content-Type': 'application/json', 'x-api-key': 'fc822e66-02cf-49bc-963c-e614e886e542'}
```

```
#relacionados con imágenes de caras
```

```
imageUrl = 'http://localhost:8000/api/v1/recognition/faces'
```

```
imageHeaders = {'x-api-key': 'fc822e66-02cf-49bc-963c-e614e886e542'}
```

```
#Relacionados con reconocimiento de rostros
```

```
recognizeUrl = 'http://localhost:8000/api/v1/recognition/recognize'
```

```
recognizeHeaders = {'x-api-key': 'fc822e66-02cf-49bc-963c-e614e886e542'}
```

```
saveResults(recognizeUrl, recognizeHeaders)
```

NOTA: SI NO DETECTA UN ROSTRO EN LA IMAGEN DEVUELVE LO SIGUIENTE{'message': 'No face is found in the given image', 'code': 28}

```
# -*- coding: utf-8 -*-  
"""
```

Created on Mon May 2 15:48:05 2022

@author: Sebastian
"""

```
import face_recognition  
import cv2
```

```
# _____ Validaci3n rostros Galer3a Face  
Recognition _____
```

```
def detectFace(url):  
    image = face_recognition.load_image_file(url)  
    locations = face_recognition.face_locations(image, number_of_times_to_upsample=3)  
    return locations
```

```
def saveResultsFront():  
    for i in range(32):  
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_01.jpg'  
        locations = detectFace(directory)  
        saveImageBounds(directory, locations, 'frente/persona' + str(i+1) + '.jpg')
```

```
def saveImageBounds(url, locations, imgName):  
    image = cv2.imread(url)  
    for result in locations:  
        cv2.rectangle(image, (result[3],result[0]), (result[1],result[2]),(0,255,0),3)  
    cv2.imwrite('/home/sebas/resultados deteccion/face recognition/' + imgName, image)
```

```
saveResultsFront()
```

```
# -*- coding: utf-8 -*-  
"""
```

Created on Mon Feb 21 16:19:11 2022

@author: Sebastian
"""

#bibliotecas

```
import face_recognition  
import os  
import numpy as np  
import cv2
```

```
# _____ Reconocimiento Facial Face Recognition
```

#almacenar informacion de rostros

```
def encodeFaces():  
    encodedFaces = []  
    names = []  
    for i in range(32):  
        newImage = face_recognition.load_image_file('/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_01.jpg')  
        locations = face_recognition.face_locations(newImage, number_of_times_to_upsample=3)  
        encodedFaces.append(face_recognition.face_encodings(newImage, locations, model='large', num_jitters=2)[0])  
        names.append('Persona'+str(i+1))  
    print ('Persona'+str(i+1))  
    return (encodedFaces, names)
```

#Mostrar distancia euclidean por cada rostro. La distancia indica cuan similar son los rostros

```
def faceDistance(image, encodedFaces):  
    result = []  
    newImage = face_recognition.load_image_file(image)  
    locations = face_recognition.face_locations(newImage, number_of_times_to_upsample=3)  
    encodedUnknown = face_recognition.face_encodings(newImage, locations, model='large', num_jitters=2)  
    for value in encodedUnknown:  
        result.append(face_recognition.face_distance(encodedFaces, value))  
    return (result, locations)
```

#Obtener nombre de persona en base al indice de rostro con la menor distancia euclidean

```
def getNameDistance(image, names, encodedFaces):  
    result, locations = faceDistance(image, encodedFaces)  
    dictNames = []  
    for value in result:  
        index = np.argmin(value)  
        print (index)  
        dictNames.append({'Name': names[index], 'distance': value[index]})  
    return (dictNames, locations)
```

#Automatizar guardado de resultados en un archivo .txt

```
def saveResultsRight(names, encodedFaces):  
    file = open("resultadosderecho.txt", "w")  
    for i in range(32):  
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)  
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_02.jpg'  
        result, locations = getNameDistance(directory, names, encodedFaces)  
        file.write('Archivo: ' + directory + os.linesep)  
        for value in result:  
            file.write('Resultado: ' + value['Name'] + 'Distancia Euclidean: ' + str(value['distance']) + os.linesep)  
            print('result: ' + value['Name'] + 'Distancia Euclidean: ' + str(value['distance']))  
        saveImageBoundsNames(directory, result, locations, 'derecho/persona' + str(i+1) + '.jpg')  
    file.close()
```

```
def saveResultsLeft(names, encodedFaces):  
    file = open("resultadosizquierdo.txt", "w")  
    for i in range(32):  
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
```

```

directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_03.jpg'
result, locations = getNameDistance(directory, names, encodedFaces)
file.write('Archivo: '+ directory + os.linesep)
for value in result:
    file.write('Resultado:' + value['Name'] + 'Distancia Euclideana: ' + str(value['distance']) + os.linesep)
    print('result: ' + value['Name'] + 'Distancia Euclideana: ' + str(value['distance']))
saveImageBoundsNames(directory, result, locations, 'izquierdo/persona' + str(i+1) + '.jpg')
file.close()

```

```

def saveResultsLower(names, encodedFaces):

```

```

    file = open("resultadosinferior.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_04.jpg'
        result, locations = getNameDistance(directory, names, encodedFaces)
        file.write('Archivo: '+ directory + os.linesep)
        for value in result:
            file.write('Resultado:' + value['Name'] + 'Distancia Euclideana: ' + str(value['distance']) + os.linesep)
            print('result: ' + value['Name'] + 'Distancia Euclideana: ' + str(value['distance']))
        saveImageBoundsNames(directory, result, locations, 'inferior/persona' + str(i+1) + '.jpg')
    file.close()

```

```

def saveResultsRandom(names, encodedFaces):

```

```

    file = open("resultadosaleatorio.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_05.jpg'
        result, locations = getNameDistance(directory, names, encodedFaces)
        file.write('Archivo: '+ directory + os.linesep)
        for value in result:
            file.write('Resultado:' + value['Name'] + 'Distancia Euclideana: ' + str(value['distance']) + os.linesep)
            print('result: ' + value['Name'] + 'Distancia Euclideana: ' + str(value['distance']))
        saveImageBoundsNames(directory, result, locations, 'aleatorio/persona' + str(i+1) + '.jpg')
    file.close()

```

```

#encodedFaces, names = encodeFaces()

```

```

def verifyFirst():

```

```

    for i in range(32):
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_01.jpg'
        image = cv2.imread(directory)
        picture = face_recognition.load_image_file(directory)
        locations = face_recognition.face_locations(picture, number_of_times_to_upsample=3)
        cv2.rectangle(image, (locations[0][3],locations[0][0]), (locations[0][1],locations[0][2]),(0,255,0),3)

```

```

#encierra en un cuadro rostros encontrados y les pone el nombre

```

```

def saveImageBoundsNames(url, result, locations, imgName):

```

```

    image = cv2.imread(url)
    for value in range(len(result)):
        cv2.rectangle(image, (locations[value][3],locations[value][0]), (locations[value][1],locations[value][2]),(0,255,0),3)
        cv2.putText(image, result[value]['Name'], (locations[value][3] , locations[value][2] +40), cv2.FONT_HERSHEY_SIMPLEX,
2, (0,255,0),3)
    cv2.imwrite('/home/sebas/resultados reconocimiento/face recognition/' + imgName, image)

```

```

#NOTA: si no se detecta un rostro en la imagen devuelve un error de 'IndexError: list index out of range'

```

```
# -*- coding: utf-8 -*-  
"""
```

Created on Mon Feb 21 16:19:11 2022

```
@author: Sebastian  
"""
```

```
#bibliotecas
```

```
import face_recognition  
import os  
import numpy as np  
import cv2
```

```
# _____ Obtener resultados base de datos externa  
Face Recognition _____
```

```
#almacenar informacion de rostros
```

```
def encodeFaces():  
    encodedFaces = []  
    names = []  
    for i in range(32):  
        newImage = face_recognition.load_image_file('/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_01.jpg')  
        locations = face_recognition.face_locations(newImage, number_of_times_to_upsample=3)  
        encodedFaces.append(face_recognition.face_encodings(newImage, locations, model='large', num_jitters=2)[0])  
        names.append('Persona'+str(i+1))  
    print ('Persona'+str(i+1))  
    return (encodedFaces, names)
```

```
#Mostrar distancia euclidean por cada rostro. La distancia indica cuan similar son los rostros
```

```
def faceDistance(image, encodedFaces):  
    result = []  
    newImage = face_recognition.load_image_file(image)  
    locations = face_recognition.face_locations(newImage, number_of_times_to_upsample=3)  
    encodedUnknown = face_recognition.face_encodings(newImage, locations, model='large', num_jitters=2)  
    for value in encodedUnknown:  
        result.append(face_recognition.face_distance(encodedFaces, value))  
    return (result, locations)
```

```
#Obtener nombre de persona en base al indice de rostro con la menor distancia euclidean
```

```
def getNameDistance(image, names, encodedFaces):  
    result, locations = faceDistance(image, encodedFaces)  
    dictNames = []  
    for value in result:  
        index = np.argmin(value)  
        print (index)  
        dictNames.append({'Name': names[index], 'distance': value[index]})  
    return(dictNames, locations)
```

```
#Automatizar guardado de resultados en un archivo .txt
```

```
def saveResults(names, encodedFaces):  
    file = open("resultadosnobd.txt", "w")  
    euclideanDistances = []  
    for i in range(25):  
        for j in range(5):  
            directory = '/home/sebas/TD_RGB_E_Set1/'+str(i+1)+'/'+'TD_RGB_E_'+str(j+1)+'.'+'jpg'  
            result, locations = getNameDistance(directory, names, encodedFaces)  
            file.write('Archivo: ' + directory + os.linesep)  
            euclideanDistances.append(result[0]['distance'])  
            for value in result:  
                file.write('Resultado: ' + value['Name'] + 'Distancia Euclidean: ' + str(value['distance']) + os.linesep)  
            print('result: ' + value['Name'] + 'Distancia Euclidean: ' + str(value['distance']))  
            saveImageBoundsNames(directory, result, locations, 'persona' + str(i+1) + str(j+1) + '.jpg')  
    file.close()  
    return euclideanDistances
```

#encierra en un cuadro rostros encontrados y les pone el nombre

```
def saveImageBoundsNames(url, result, locations, imgName):
```

```
    image = cv2.imread(url)
```

```
    for value in range(len(result)):
```

```
        cv2.rectangle(image, (locations[value][3],locations[value][0]), (locations[value][1],locations[value][2]),(0,255,0),3)
```

```
        cv2.putText(image, result[value]['Name'], (locations[value][3] , locations[value][2] +40), cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0),3)
```

```
    cv2.imwrite('/home/sebas/pruebas rostros no guardados/face recognition/' + imgName, image)
```

#NOTA: si no se detecta un rostro en la imagen devuelve un error de 'IndexError: list index out of range'

```
# -*- coding: utf-8 -*-  
"""
```

Created on Mon May 2 15:06:07 2022

```
@author: Sebastian  
"""
```

```
from azure.cognitiveservices.vision.face import FaceClient  
from msrest.authentication import CognitiveServicesCredentials  
from azure.cognitiveservices.vision.face.models import TrainingStatusType, Person, QualityForRecognition  
import cv2
```

```
# _____ Validaci3n rostros galer3a Azure Face Api
```

```
def detectFaces(url):  
    face_rectangle = []  
    image = open(url, 'r+b')  
    # se usa el mismo modelo de reconocimiento que fue asignado en el mismo grupo de personas  
    faces = face_client.face.detect_with_stream(image, detection_model='detection_03', recognition_model='recognition_04')  
    for face in faces:  
        face_rectangle.append(face.face_rectangle)  
    return (face_rectangle)
```

```
def saveResultsFront():  
    for i in range(32):  
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_01.jpg'  
        face_rectangle = detectFaces(directory)  
        saveImageBounds(directory, face_rectangle, 'frente/persona' +str(i+1) + '.jpg')
```

```
def saveImageBounds(url, faceRectangle, imgName):  
    image = cv2.imread(url)  
    for i in range(len(faceRectangle)):  
        cv2.rectangle(image, (faceRectangle[i].left ,faceRectangle[i].top),  
(faceRectangle[i].width+faceRectangle[i].left,faceRectangle[i].height+faceRectangle[i].top),(0,255,0),3)  
        cv2.imwrite('/home/sebas/resultados deteccion/azure/' + imgName, image)
```

```
key = "Key obtenida de Azure"
```

```
endpoint = "https://comparison.cognitiveservices.azure.com/"
```

```
face_client = FaceClient(endpoint, CognitiveServicesCredentials(key))
```

```
saveResultsFront()
```

```
# -*- coding: utf-8 -*-  
"""
```

Created on Thu Mar 17 18:10:12 2022

```
@author: Sebastian  
"""
```

```
#Bibliotecas
```

```
from azure.cognitiveservices.vision.face import FaceClient  
from msrest.authentication import CognitiveServicesCredentials  
from azure.cognitiveservices.vision.face.models import TrainingStatusType, Person, QualityForRecognition  
import os  
import cv2  
#Funciones
```

```
# _____ Resultados reconocimiento Azure Face  
Api _____
```

```
#Crear grupo de personas
```

```
def createPersonGroup(PERSON_GROUP_ID):  
    face_client.person_group.create(person_group_id = PERSON_GROUP_ID, name = PERSON_GROUP_ID,  
    recognition_model='recognition_04')
```

```
#crear personas pertenecientes a el grupo de personas creado
```

```
def createPerson(PERSON_GROUP_ID):  
    personas = []  
    for i in range(32):  
        personas.append(face_client.person_group_person.create(PERSON_GROUP_ID, 'persona' + str(i+1)))  
        print ('persona' + str(i+1))  
    return personas
```

```
#asignar imagenes a personas
```

```
def assignImages(PERSON_GROUP_ID, personas):  
    for i in range(32):  
        try:  
            print ('Archivo: ' + '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'bd_01.jpg')  
            image = open('/home/sebas/base_de_datos_def_copy/'+str(i+1)+'bd_01.jpg', 'r+b')  
            face_client.person_group_person.add_face_from_stream(PERSON_GROUP_ID, personas[i].person_id, image,  
            detection_model='detection_03')  
        except Exception as e:  
            print (e)
```

```
#Identificar personas en la foto
```

```
def detectFaces(url):  
    face_ids = []  
    face_rectangle = []  
    image = open(url, 'r+b')  
    # se usa el mismo modelo de reconocimiento que fue asignado en el mismo grupo de personas  
    faces = face_client.face.detect_with_stream(image, detection_model='detection_03', recognition_model='recognition_04')  
    for face in faces:  
        face_ids.append(face.face_id)  
        face_rectangle.append(face.face_rectangle)  
    return (face_ids, face_rectangle)
```

```
#Reconocer a una persona. Para esto se necesita identificar a una persona
```

```
def recognizePerson(url, PERSON_GROUP_ID):  
    face_ids, face_rectangle = detectFaces(url)  
    results = face_client.face.identify(face_ids, PERSON_GROUP_ID)  
    dictNames = []  
    for value in results:  
        try:  
            person = face_client.person_group_person.get(person_group_id = PERSON_GROUP_ID, person_id =
```

```

value.candidates[0].person_id)
    dictNames.append({'name': person.name, 'confidence': value.candidates[0].confidence})
except Exception as e:
    dictNames.append({'error': str(e)})
return (dictNames, face_rectangle)

```

#Guardar resultados en archivo .txt

```

def saveResultsRight(PERSON_GROUP_ID):
    file = open("resultadosderecho.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_02.jpg'
        result, face_rectangle = recognizePerson(directory, PERSON_GROUP_ID)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result:
            file.write('Resultado: ' + value['name'] + ' Confidencia: ' + str(value['confidence']) + os.linesep)
            print('result: ' + value['name'] + ' Confidencia: ' + str(value['confidence']))
        saveImageBounds(directory, result, face_rectangle, 'derecho/persona' + str(i+1) + '.jpg')
    file.close()

```

```

def saveResultsLeft(PERSON_GROUP_ID):
    file = open("resultadosizquierdo.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_03.jpg'
        result, face_rectangle = recognizePerson(directory, PERSON_GROUP_ID)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result:
            file.write('Resultado: ' + value['name'] + ' Confidencia: ' + str(value['confidence']) + os.linesep)
            print('result: ' + value['name'] + ' Confidencia: ' + str(value['confidence']))
        saveImageBounds(directory, result, face_rectangle, 'izquierdo/persona' + str(i+1) + '.jpg')
    file.close()

```

```

def saveResultsLower(PERSON_GROUP_ID):
    file = open("resultadosinferior.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_04.jpg'
        result, face_rectangle = recognizePerson(directory, PERSON_GROUP_ID)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result:
            file.write('Resultado: ' + value['name'] + ' Confidencia: ' + str(value['confidence']) + os.linesep)
            print('result: ' + value['name'] + ' Confidencia: ' + str(value['confidence']))
        saveImageBounds(directory, result, face_rectangle, 'inferior/persona' + str(i+1) + '.jpg')
    file.close()

```

```

def saveResultsRandom(PERSON_GROUP_ID):
    file = open("resultadosaleatorio.txt", "w")
    for i in range(32):
        file.write(os.linesep + 'PRUEBAS PARA Persona' + str(i+1) + os.linesep)
        directory = '/home/sebas/base_de_datos_def_copy/'+str(i+1)+'/'+'bd_05.jpg'
        result, face_rectangle = recognizePerson(directory, PERSON_GROUP_ID)
        file.write('Archivo: ' + directory + os.linesep)
        for value in result:
            try:
                file.write('Resultado: ' + value['name'] + ' Confidencia: ' + str(value['confidence']) + os.linesep)
                print('result: ' + value['name'] + ' Confidencia: ' + str(value['confidence']))
            except Exception as e:
                file.write('Error: ' + value['error'] + os.linesep)
        saveImageBounds(directory, result, face_rectangle, 'aleatorio/persona' + str(i+1) + '.jpg')
    file.close()

```

```

def saveImageBounds(url, result, faceRectangle, imgName):
    image = cv2.imread(url)
    for i in range(len(faceRectangle)):
        cv2.rectangle(image, (faceRectangle[i].left, faceRectangle[i].top),

```

```
(faceRectangle[i].width+faceRectangle[i].left,faceRectangle[i].height+faceRectangle[i].top),(0,255,0),3)
    try:
        cv2.putText(image, result[i]['name'], (faceRectangle[i].left , faceRectangle[i].height+faceRectangle[i].top +40),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0),3)
    except:
        pass
cv2.imwrite('/home/sebas/resultados reconocimiento/azure/' + imgName, image)
```

#variables

key = "key obtenida de Azure"

endpoint = "https://comparison.cognitiveservices.azure.com/"

face_client = FaceClient(endpoint, CognitiveServicesCredentials(key))

PERSON_GROUP_ID = 'nuevadbpersonasno'

#print ('ID DEL PERSON GROUP: ' + PERSON_GROUP_ID)

#createPersonGroup(PERSON_GROUP_ID)

#personas = createPerson(PERSON_GROUP_ID)

#assignImages(PERSON_GROUP_ID, personas)

face_client.person_group.train(PERSON_GROUP_ID)

#face_client.person_group.list()[0].person_group_id obtener los ids de cada uno de los person groups

#face_client.person_group.delete

```
# -*- coding: utf-8 -*-  
"""
```

Created on Thu Mar 17 18:10:12 2022

```
@author: Sebastian  
"""
```

```
#Bibliotecas
```

```
from azure.cognitiveservices.vision.face import FaceClient  
from msrest.authentication import CognitiveServicesCredentials  
from azure.cognitiveservices.vision.face.models import TrainingStatusType, Person, QualityForRecognition  
import os  
import cv2  
#Funciones
```

```
# _____ Obtener resultados base de datos externa  
Azure Face Api _____
```

```
#Identificar personas en la foto
```

```
def detectFaces(url):  
    face_ids = []  
    face_rectangle = []  
    image = open(url, 'r+b')  
    # se usa el mismo modelo de reconocimiento que fue asignado en el mismo grupo de personas  
    faces = face_client.face.detect_with_stream(image, detection_model='detection_03', recognition_model='recognition_04')  
    for face in faces:  
        face_ids.append(face.face_id)  
        face_rectangle.append(face.face_rectangle)  
    return (face_ids, face_rectangle)
```

```
#Reconocer a una persona. Para esto se necesita identificar a una persona
```

```
def recognizePerson(url, PERSON_GROUP_ID):  
    face_ids, face_rectangle = detectFaces(url)  
    results = face_client.face.identify(face_ids, PERSON_GROUP_ID, confidence_threshold=0)  
    dictNames = []  
    for value in results:  
        try:  
            person = face_client.person_group_person.get(person_group_id = PERSON_GROUP_ID, person_id =  
value.candidates[0].person_id)  
            dictNames.append({'name': person.name, 'confidence': value.candidates[0].confidence})  
        except Exception as e:  
            dictNames.append({'error': str(e)})  
    return (dictNames, face_rectangle)
```

```
#Guardar resultados en archivo .txt
```

```
def saveResults(PERSON_GROUP_ID):  
    file = open("resultadosnobd.txt", "w")  
    confidences = []  
    for i in range(25):  
        for j in range(5):  
            directory = '/home/sebas/TD_RGB_E_Set1/'+str(i+1)+'/' + 'TD_RGB_E_'+str(j+1)+'.jpg'  
            result, face_rectangle = recognizePerson(directory, PERSON_GROUP_ID)  
            file.write('Archivo: ' + directory + os.linesep)  
            try:  
                confidences.append(result[0]['confidence'])  
            except:  
                pass  
        for value in result:  
            try:  
                file.write('Resultado: ' + value['name'] + ' Confidencia: ' + str(value['confidence']) + os.linesep)  
                print('result: ' + value['name'] + ' Confidencia: ' + str(value['confidence']))  
            except Exception as e:  
                file.write('Error: ' + value['error'] + os.linesep)
```

```
        saveImageBounds(directory, result, face_rectangle, 'persona' +str(i+1) + str(j+1) + '.jpg')
    file.close()
    return confidences
```

```
def saveImageBounds(url, result, faceRectangle, imgName):
    image = cv2.imread(url)
    for i in range(len(faceRectangle)):
        cv2.rectangle(image, (faceRectangle[i].left ,faceRectangle[i].top),
(faceRectangle[i].width+faceRectangle[i].left,faceRectangle[i].height+faceRectangle[i].top),(0,255,0),3)
        try:
            cv2.putText(image, result[i]['name'], (faceRectangle[i].left , faceRectangle[i].height+faceRectangle[i].top +40),
cv2.FONT_HERSHEY_SIMPLEX, 2, (0,255,0),3)
        except:
            pass
    cv2.imwrite('/home/sebas/pruebas rostros no guardados/azure/' + imgName, image)
```

#variables

key = "llave obtenida de Azure"

endpoint = "https://comparison.cognitiveservices.azure.com/"

face_client = FaceClient(endpoint, CognitiveServicesCredentials(key))

```
PERSON_GROUP_ID = 'nuevadbpersonasno'
saveResults(PERSON_GROUP_ID)
#print ('ID DEL PERSON GROUP: ' + PERSON_GROUP_ID)
#createPersonGroup(PERSON_GROUP_ID)
#personas = createPerson(PERSON_GROUP_ID)
#assignImages(PERSON_GROUP_ID, personas)
#face_client.person_group.train(PERSON_GROUP_ID)
```

#face_client.person_group.list()[0].person_group_id obtener los ids de cada uno de los person groups

#face_client.person_group.delete