UACM

Universidad Autónoma de la Ciudad de México

Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA DE SOFTWARE

Prototipo de un sistema web de videovigilancia para la identificación de personas usando redes neuronales de aprendizaje profundo

TESIS QUE PARA OBTENER EL TÍTULO DE
LICENCIADO EN INGENIERÍA DE SOFTWARE

PRESENTA

Abraham Osorio Vázquez

Director de la Tesis

Dr. Gerardo Hernández Hernández

Ciudad de México, junio de 2023.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

Dedicatoria

A mi padre, Armando Osorio, quien me brindó los recursos tanto materiales como emocionales, los cuales me permitieron tener una educación para poder sobresalir, así como sus valiosos consejos y apoyo que me permitieron cumplir con una de mis más grandes metas y sobre todo de siempre superarme en cada meta que me proponga.

A mi madre, Guadalupe Vázquez, quien siempre me brindó su apoyo y sus ánimos para seguir adelante, así como sus valiosos consejos, los cuales me permitieron llegar hasta este momento.

A mis dos mejores amigas:

Vanessa Aguirre, que, a pesar de ya no estar con nosotros, me brindó su apoyo incondicional y los ánimos a lo largo de los años en mi vida académica y personal, de siempre confiar en mí para lograr cada meta que me propusiera.

Sherling Odette López, por haberme acompañado durante la carrera, de brindarme su apoyo durante el proceso de la elaboración de mi tesis, y sobre todo de no rendirme sin importar las dificultades que eso implique.

Agradecimientos

Quiero brindar mi agradecimiento a mi director y codirector de tesis, el profesor Gerardo Hernández y Máximo Eduardo Sánchez Gutiérrez, quienes me brindaron su apoyo y tiempo para la realización de esta tesis, y sobre todo de brindarme su confianza para poder llevar este trabajo desde principio a fin.

A la institución de la UACM Cuautepec, por brindarme una oportunidad de tener una educación profesional, para poder crecer en mi vida profesional y personal.

A todos mis maestros de la carrera de Ingeniería de Software, por brindarme las herramientas para poder enfrentar los obstáculos que se van presentando en la vida laboral.

Por último, agradecer a mis amigos y amigas que conocí a lo largo de la carrera, y de los cuales formamos equipos de trabajo, para poder salir adelante cada semestre.

Índice General

Índice de figuras

Índice de tablas

Resumen

Este trabajo se realizó con el fin de elaborar un prototipo de un sistema Web para videovigilancia que permita identificar a personas mediante el uso de redes neuronales de aprendizaje profundo. Este prototipo permitirá al usuario subir videos que tenga almacenados en su computadora para realizar la identificación de individuos que aparezcan en dichos videos.

El prototipo también cuenta con una sección de registro de personas, el cual requiere de una sola imagen para registrar e identificar a un individuo. De tal forma que, en el proceso de identificación de personas en el vídeo, el prototipo mostrará el nombre de la persona, así como la fecha y hora de identificación.

Otra característica importante de mencionar es que el prototipo es capaz de identificar a varias personas en la misma escena y a diferente distancia focal, gracias al uso de redes neuronales convolucionales para la detección del rostro.

Para llevar a cabo la elaboración de este prototipo se utilizó el método de investigación aplicada, el cual consiste en llevar a cabo un estudio previo para el desarrollo de una estrategia, para así alcanzar un objetivo concreto. Además, se empleó una metodología de desarrollo de software, conocida como “Prototipo”, en la etapa de desarrollo de dicho software.

Para lograr este objetivo se hizo uso del API “Face-api”, el cual es una herramienta de código abierto, el cual permite hacer el reconocimiento facial, a través de “Descriptores faciales”, el cual es un vector de características del rostro humano con 128 valores.

Como primer paso se realizaron pruebas de funcionalidad del API, con el fin de observar el funcionamiento y visualizar cómo manejar los resultados de las detecciones, dichas pruebas fueron realizadas con 15 personas (hombres y mujeres), con edades de 23 a 70 años, se manejaron dos escenarios para las pruebas, en condiciones ideales (la persona se encuentra frente a la cámara) y en condiciones reales (la persona se mueve por un espacio), con la finalidad de determinar si los resultados son viables para ambos escenarios.

Los resultados que se obtuvieron, señalan que la detección del rostro en condiciones ideales es correcta en un 95% ± 4% de las personas con las que se realizaron las pruebas. Para las pruebas en condiciones reales es correcta en un 82% ± 13%.

Introducción

Actualmente existen varias opciones de sistemas de videovigilancia (como los NVR y DVR), que son dispositivos a los cuales se les pueden conectar cámaras de vigilancia y que también pueden almacenar las grabaciones en formato de video, sin embargo, esto llega a presentar un problema a la hora de revisar dichas grabaciones, ya que, si se quiere detectar a una persona, el sistema no tiene la capacidad de brindar esa información.

Tomando lo anterior como base, se realizó una investigación para determinar si existían sistemas de videovigilancia que permitieran el reconocimiento facial mediante videos que se tengan almacenados. Actualmente los sistemas que cuentan con dicha tecnología son escasos y en ocasiones costosos de adquirir, tanto en aplicaciones Web como aplicaciones para computadora, lo que sí se tiene son APIs para incorporar a un proyecto personal o de empresas.

Durante el desarrollo de esta tesis se llevó a cabo el análisis de diferentes APIs de código abierto, con el propósito de llegar a la elección de una API con la que se pueda dar una solución a la problemática, y a realizar una investigación teórica del funcionamiento de la misma, esto con el fin de explicar cómo es el funcionamiento de una API de reconocimiento facial.

Este proyecto de tesis tiene la finalidad de presentar un prototipo de un sistema Web que permitirá el uso de videos que se tengan almacenados en la computadora para posteriormente ser analizados con técnicas de reconocimiento facial y realizar la detección e identificación de personas en el vídeo, así como incorporar una sección, en donde se almacenarán las imágenes base de las personas a las que se desea detectar.

Capítulo 1
Problemática

1.1. Introducción

En este capítulo se presentará con más detalle la problemática para el presente trabajo de tesis, tomando como referencia el problema que se planea resolver, así como los objetivos, el alcance y limitaciones.

1.2. Identificación del problema

Actualmente existen diversos dispositivos que están enfocados directamente a la videovigilancia en tiempo real, sin embargo, los sistemas que permiten examinar videos que fueron grabados con anterioridad que pertenezcan al mismo equipo y no de otros videos que se tengan, y sobre todo ofrecer información que le pueda ayudar a identificar a las personas. Esto puede llegar a afectar a todas aquellas personas que necesiten información sobre las grabaciones, ya sea para armar un reporte (en caso de ser para una empresa), o identificar a una persona ajena a un domicilio. A pesar de que existen alternativas para poder solucionar este problema, solo existen complementos (API) que pueden ser incorporados a dichos sistemas, dichos complementos están orientados a personas que se dedican al desarrollo de sistemas, y no a un público en general.

Los sistemas de videovigilancia actuales, aunque son muy útiles cuentan con ciertas desventajas (Tabla 1.1), esto conlleva a que durante el análisis de los resultados que ofrecen se puede llegar a tener información incompleta, esto implica que en lugares donde se requieren reportes de personal, cabe la posibilidad de haber una pérdida de información en cuanto al registro de las personas.

	Sistemas de videovigilancia (NVR/DVR)	**Sistema de videovigilancia (Reconocimiento Facial)**
Permite la reproducción de videos	✓	✓
Permite la reproducción de videos ajenos al sistema	✗	✓
Muestra información del video	✗	✓
Detectan persona que aparecen en el video	✗	✓
Acepta diferentes formatos de vídeo (mp4, mkv, etc.)	✗	✓

Tabla 1.1: Comparación sistemas NVR/DVR contra sistema con reconocimiento facial.

La información de la tabla anterior fue revisada de los manuales técnicos de algunos dispositivos. (Hikvision, 2018; HiLook, 2020; Networks, 2021; IDIS, 2020)

Como se observa en la tabla anterior, los lugares donde se utilizan estos tipos de sistemas (NVR/DVR), llegan a tener limitaciones con los formatos de video e

información. Esto lleva a que mayormente en empresas y hogares, se tengan datos incompletos (Tabla 1.2).

	Sistemas de videovigilancia (NVR/DVR)	**Sistema de videovigilancia (Reconocimiento Facial)**
Proporciona fecha y hora del video	✓	✓
Proporciona el(los) nombre(s) de la(s) persona(s)	✗	✓
Aparece el porcentaje de detección facial	✗	✓
Confirma si hubo detección de persona(s)	✗	✓

Tabla 1.2: Comparación de la información mostrada en ambos sistemas.

La información de la tabla anterior fue revisada de los manuales técnicos de algunos dispositivos. (HiLook, 2020; Hikvision, 2018; IDIS, 2020; Networks, 2021).

Tomando como base todo lo anterior, se tiene como objetivo proporcionar un prototipo que permita resolver dicha problemática, para evitar en su mayoría la pérdida de información, al momento de revisar y analizar los videos que se tengan almacenados, se tenga la seguridad de que dichos datos estén completos a la hora de realizar reportes o simplemente tener información sobre quienes han estado en un lugar determinado.

1.3. Justificación

Dada las limitaciones con las que cuentan los sistemas de videovigilancia actuales para la detección de personas que se expusieron anteriormente, se tiene la oportunidad de desarrollar un nuevo sistema que le permita a los usuarios revisar grabaciones para la detección de personas que estuvieron en la zona de vigilancia, con una mayor disponibilidad a través de un celular o computadora conectada a Internet.

Con este trabajo se pretende proporcionar una alternativa a los usuarios para el ahorro tanto de tiempo y dinero, ya que solo se necesitan cámaras Web (de cualquier tipo), celular o computadora personal e Internet, esto para proporcionarle un ambiente más accesible.

1.4. Objetivos

1.4.1. Objetivo general

Realizar un prototipo de videovigilancia Web, que le permita a los usuarios subir y reproducir videos que tengan almacenados para su revisión, contar con reconocimiento facial y brindar información acerca de los videos (fecha y hora).

1.4.2. Objetivos específicos

- Determinar la viabilidad en el uso de cámaras Web convencionales para la construcción de un prototipo de videovigilancia.
- Investigar, analizar y seleccionar una API de código abierto que permita realizar reconocimiento facial y detección de personas sobre imágenes y videos.

- Investigar y analizar los diferentes *frameworks* existentes en el mercado actual para el desarrollo de prototipos Web.
- Diseñar y construir un primer prototipo funcional con la API de reconocimiento facial elegida con anterioridad, que permita realizar el análisis de videos almacenados en el ordenador o dispositivos de almacenamiento externos.
- Realizar pruebas de usabilidad del prototipo.

1.5. Estado del arte

Para dar solución a la problemática inicial se requiere el uso de un software cuya funcionalidad sea el reconocimiento facial. Actualmente existen diversas herramientas que ofrecen esta funcionalidad, sin embargo, cabe mencionar que son de paga y sólo proporcionan los permisos de uso, y el uso sólo puede ser para aquellas personas que conocen del tema y no están dirigidas a un público en general. Por otro lado, también existe software de código abierto, que es moldeable para cualquier proyecto, de los cuales podemos considerar los siguientes:

1.5.1. OpenFace

Es una API de código abierto (*open source*) que está basado en *FaceNet* (proyecto de investigación de Google para reconocimiento facial), tiene la función de captar e identificar rostros en tiempo real; para lograrlo, requiere **al menos diez fotos base** de la persona. (A. Llorca, 2015; Amos, s.f.)

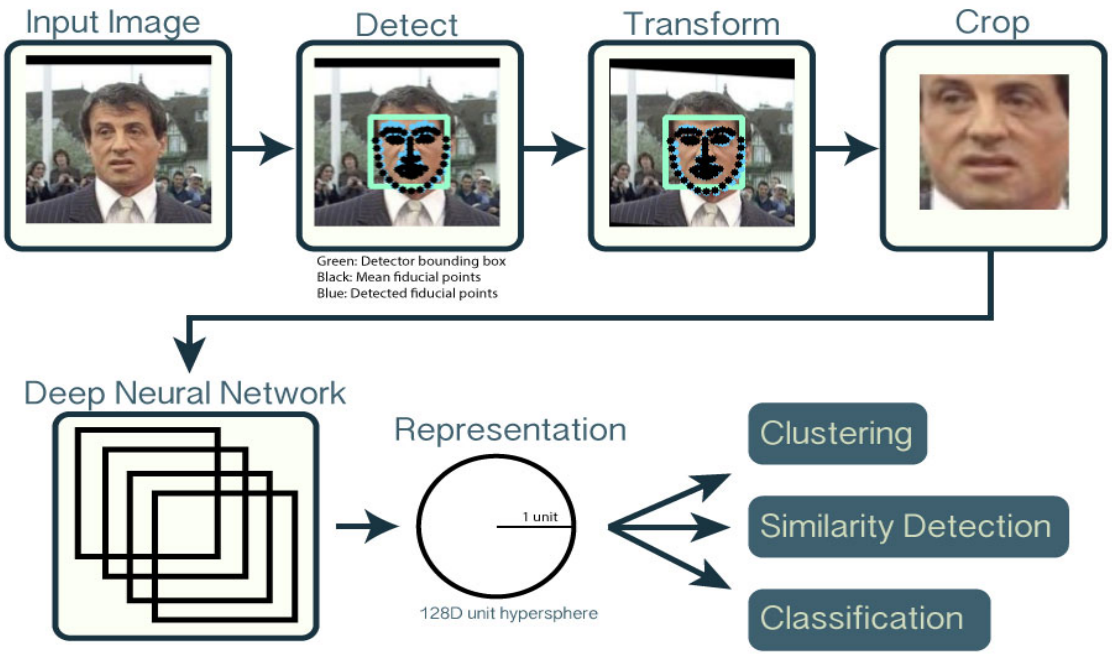

Figura 1.1: Funcionamiento de la API "OpenFace". (OpenFace, s.f.)

1.5.2. DeepFace

Es una API *open source* de marco ligero para reconocimiento facial y análisis de atributos faciales para el lenguaje de programación Python. Además, admite diferentes métodos de reconocimiento facial como *FaceNet* e *InsightFace*, pero solo admite métodos de verificación, esto quiere decir que no puede crear colecciones de rostro ni encontrar un rostro entre ellos. (Serengil S. , s.f.; Traichuk, 2021)

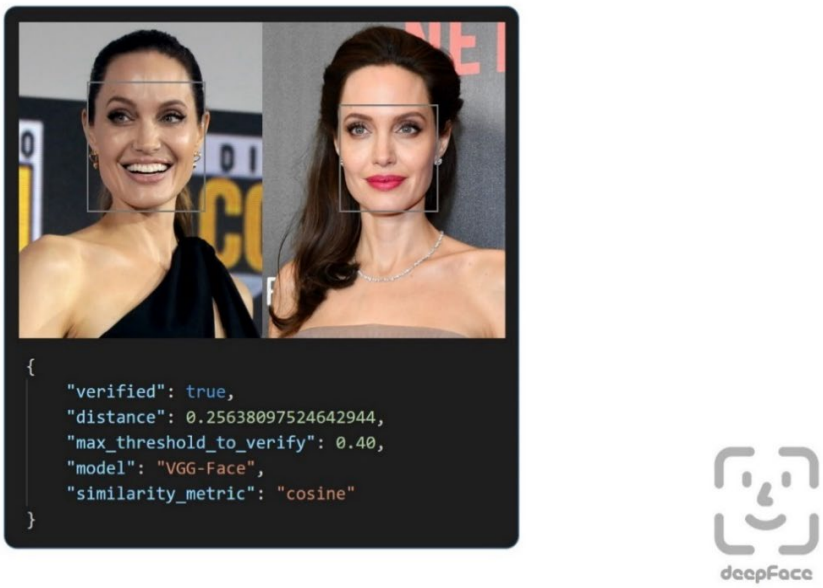

Figura 1.2: Ejemplo de funcionalidad de la API "deepFace". (Serengil, s.f.)

1.5.3. Compreface

Es una API *open source* que utiliza los dos métodos de reconocimiento facial más populares; utilizando métodos de reconocimiento facial como *FaceNet* e *InsightFace*, por lo que permite hacer reconocimiento facial en tiempo real y ubicación de rostro en entornos concurridos. (Khadzkou, s.f.)

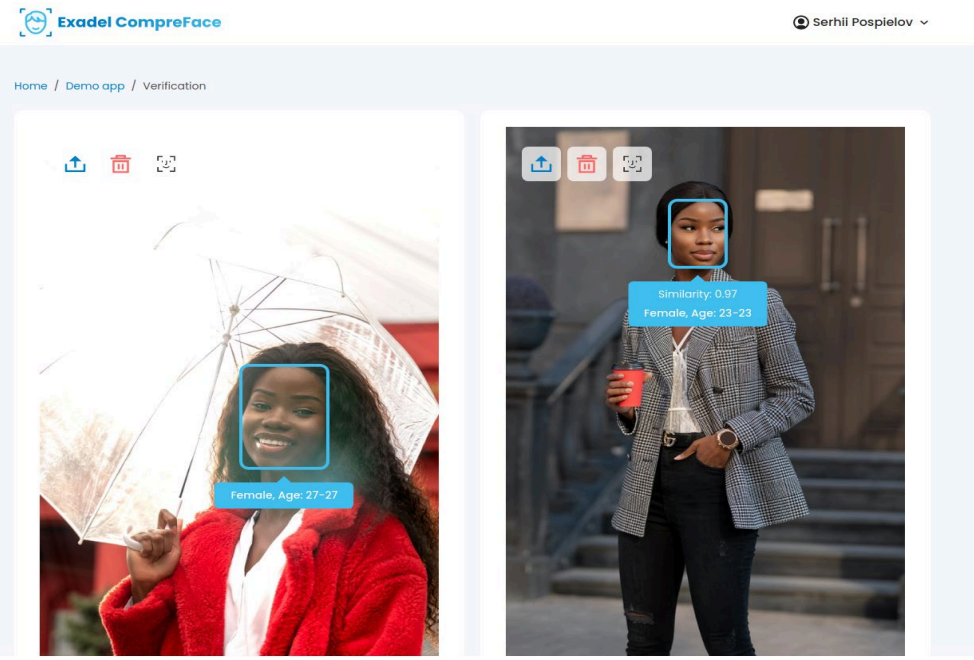

Figura 1.3: Ejemplo de funcionalidad de la API "CompreFace". (Saba, s.f.)

1.5.4. Face-api

Es una API *open source* para la detección y reconocimiento de rostros en navegadores Web, la cual está implementada sobre la API principal de *TensorFlow* (tfjs-core), esta API es capaz de identificar rostros en tiempo real y brindar información sobre la detección (como el nombre, porcentaje de detección, edad, etc.), basta **con una foto base para realizar la identificación de una persona**, sin embargo, se pueden tener más fotos base, pero lo recomendado es **dos fotos base**, ya que aumenta la precisión de detección y optimiza el procesamiento. (Mühler V. , JavaScript API for Face Recognition in the Browser with tensorflow.js, 2018; Mühler V. , face-api.js, 2020)

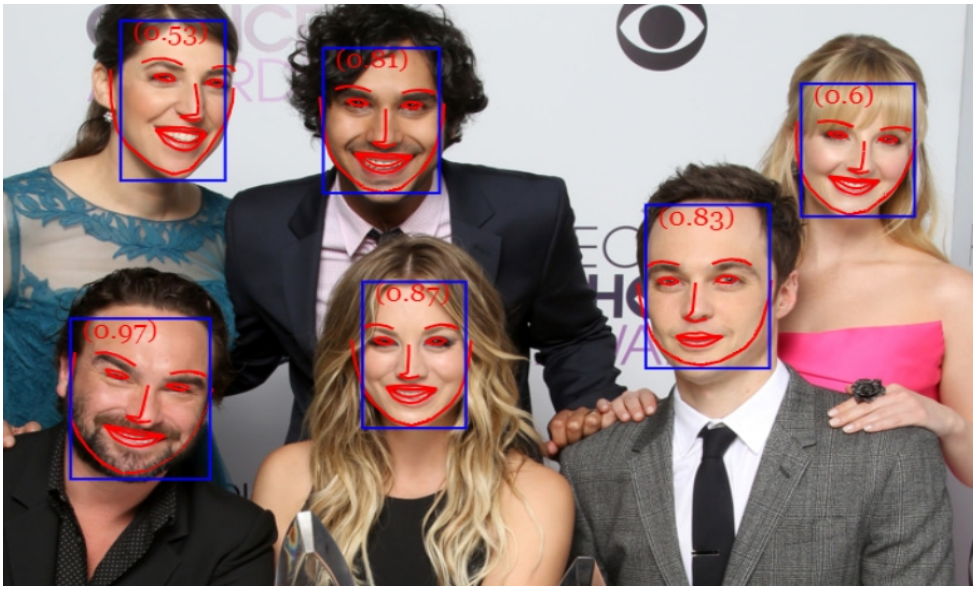

Figura 1.4: Ejemplo de funcionalidad de la API "Face-api". (Mühler V. , s.f.)

Para la construcción de este proyecto se cuentan con varias opciones que permitirán su realización, sin embargo, como se plantea un sistema Web se tendrá que elegir la opción que se adapte a este requerimiento. Por lo tanto, se propone la opción de "Face-api", ya que cuenta con la funcionalidad necesaria para poder desarrollar este proyecto, además de pedir la menor cantidad de fotos base, y sobre todo porque está pensado para sistemas Web.

1.6. Alcance y limitaciones

1.6.1. Alcances

- El proyecto no se limitará a elaborar una propuesta para un solo navegador, si no que se estima que sea compatible con cualquier navegador.
- El prototipo contará con un reproductor multimedia para aceptar cualquier formato de video, para evitar la conversión de archivos multimedia.
- El prototipo contará con una tabla de resultados, en la cual se imprimirá el resultado de la identificación y detección de personas.

- El diseño del prototipo contará con una interfaz sencilla para ser utilizada por cualquier usuario, tanto para usuarios con o sin experiencia utilizando páginas Web.
- El prototipo contará con una sección de registro de personas a “detectar”, en esta sección se registrará una o más fotografías de la persona. Dichas imágenes se almacenarán en un servidor de imágenes.

1.6.2. Limitaciones

- El proceso de detección se limitará a un máximo de 4 personas por video, ya que el hardware (computadora), cuenta con recursos limitados para la detección de un mayor número de personas por video.
- Los videos obtenidos deberán tener suficiente iluminación para poder realizar el reconocimiento facial.

Capítulo 2
Marco Teórico

2.1. Introducción

En este capítulo se analizarán las bases teóricas, así como las bases conceptuales para dar un mejor panorama sobre cómo será la posible solución de este problema. Se analizarán los diferentes métodos de detección facial, así como los diferentes clasificadores de imágenes, esto con el fin de explicar cómo funciona esta tecnología.

2.2. Técnicas de Reconocimiento Facial

Actualmente existen diferentes tipos de técnicas que se utilizan para el reconocimiento facial, las cuales son:

2.2.1. Holísticas

Para esta técnica se deben tener en cuenta los datos del rostro completo, esto quiere decir que se debe contar con una **imagen base** para obtener los datos necesarios del rostro.

Además, esta técnica sigue los siguientes pasos para la detección del rostro:

- Se debe de insertar un conjunto de imágenes a una Base de Datos, estas imágenes se deben nombrar con un **“id”** (un identificador como una matrícula o número de empleado), o el **“nombre de la persona”,** esto para “entrenar” al sistema para poder hacer las comparaciones entre imágenes.
- Los rostros propios se crean extrayendo los rasgos más característicos, estas imágenes de entrada se “normalizan” con el fin de alinear los ojos y la boca. En esta extracción de datos de imágenes se utiliza una herramienta matemática llamada “PCA”. (Domínguez Pavón, 2017)
- Para este paso a cada imagen se le representa como un vector de pesos. El sistema está listo para recibir consultas. El peso de la imagen entrante, será desconocida, de tal forma que el sistema le otorgará un peso y posteriormente se comparará con el peso de las imágenes que se insertaron con anterioridad.
- Si el peso de la imagen que entra es mayor que cualquiera de los registrados, se considera como **“no identificado o desconocido”**. La identificación de la imagen de entrada se realiza encontrando la imagen en la Base de Datos cuyo peso sea el más cercano al de la imagen registrada.
- La imagen que cuente con un peso cercano o similar al peso de la imagen que se encuentra almacenada en la Base de Datos, se considera como **“identificado”** y mostrará el nombre de la persona. (Grupo Atico34, s.f.; Lisa Institute, 2021)

2.2.2. Geométricas

Esta técnica se implementa principalmente en el **reconocimiento facial 2D** y **reconocimiento facial 3D**, a continuación, se detalla las características de cada una:

- **Reconocimiento Facial 2D**: Para esta técnica las imágenes se representan a menudo por una estructura geométrica o codificando sus valores de intensidad. Esto quiere decir que se obtiene una representación geométrica transformando la imagen en “primitivas geométricas” (Viejo & Cazorla, s.f.) como puntos y curvas. Esto se logra localizando los rasgos distintivos del rostro, por ejemplo,

los ojos, boca, nariz y el mentón, midiendo su posición relativa para poder hacer la comparación.

- **Reconocimiento Facial 3D**: Para esta técnica se aprovecha la información geométrica 3D del rostro. Tomando los datos de dispositivos con sensores 3D que capturan la información del rostro. Para hacer la comparación de los rostros se basa en la coincidencia de los metadatos extraídos de las formas 3D de los rostros. (Grupo Atico34, s.f.; Lisa Institute, 2021)

Cabe mencionar que las dos técnicas antes mencionadas funcionan con la coincidencia de **uno a muchos**, esto quiere decir que ambos sistemas deben contar con un banco de imágenes base, ya que la detección compara con todos los datos que se encuentra en la Base de Datos, esto se realiza para encontrar la mejor coincidencia por encima del umbral de comparación.

2.2.3. Análisis de la textura de la piel

Esta técnica no utiliza dimensiones del rostro o captura la simetría del rostro, como las técnicas anteriormente mencionadas, esta técnica se basa más en los detalles visuales de la piel. Analiza líneas únicas, patrones y detalles que son evidentes como las manchas y/o cicatrices que pudiera tener en el rostro. Además, con esta técnica no se necesita comparar todas las imágenes en una Base de Datos.

Aunque, por otra parte, para poder obtener buenos resultados a la hora de la detección se requiere de una mayor cantidad de imágenes base para la fase del entrenamiento del detector, también se debe considerar los cambios en la iluminación o la expresividad, ya que estos factores influyen considerablemente a la hora del análisis y esto determina si la detección es acertada o no. (Grupo Atico34, s.f.; Lisa Institute, 2021)

A pesar de que el punto anterior se pueda considerar una desventaja, esta técnica se puede utilizar para sistemas que no soportan una calidad de imagen alta, sin embargo, el resultado de la detección no llega a ser precisa y no se pueden obtener buenos

resultados, es accesible para aquellos sistemas que cuentan con imágenes de calidad baja en resolución o son de mala calidad, y la detección del rostro puede ser rápida en la ejecución para aquellos sistemas de baja complejidad.

2.2.4. Basadas en videos

Esta técnica de reconocimiento facial basado en video se establece la identidad de una o más personas presentes en cualquier video, con función de las características faciales. Teniendo como referencia un video de entrada, esto significa que la imagen base no es fija, sino dinámica y capta muy bien los cambios en las expresiones. Esto implica una caracterización temporal del rostro para el reconocimiento, ya que construye un modelo 3D o una imagen de alta resolución de la cara, también va "aprendiendo" las variaciones de la apariencia de los múltiples cuadros de video.

Esta técnica es en su mayoría utilizada para la videovigilancia, aunque es viable sigue teniendo la tarea difícil del reconocimiento facial debido a las siguientes características:

- **Baja calidad de video**: Este tipo de grabaciones de videovigilancias se toman de las cámaras que se encuentran en la calle, esto hace que las grabaciones cuenten con una iluminación inadecuada y, además de ser un ambiente abierto en el cual las personas no son "colaborativas", en el sentido de que tratan de pasar inadvertidos o simplemente pasan por lugares donde no pueden ser identificados. Para solucionar este problema, se deben de aplicar **técnicas de superresolución** (Molina, s.f.).
- **Imagen pequeña del rostro**: Ya que la mayoría de las grabaciones que se toman de los puestos de vigilancia la adquisición de la imagen suele ser más pequeña que la que hay en una Base de Datos. Esto a su vez hace que el reconocimiento facial sea más difícil, sino que también puede llegar a afectar la exactitud de la detección de los puntos más importantes utilizados para el reconocimiento del rostro. (Grupo Atico34, s.f.; Lisa Institute, 2021)

Esta técnica como se observó se diferencia de las otras tres técnicas, ya que los rasgos para hacer la comparación provienen de videos y no de imágenes, aunque este método es más dinámico ya que se tienen cambios en las expresiones faciales resulta una mejor opción, pero a su vez presenta un mayor problema debido a que se deben contar con videos que cuenten con buena iluminación y sobre todo el rostro debe estar lo más cercano posible de la cámara.

2.3. Tipos de Clasificadores de imágenes

Jesús Martínez lo define como:

“La clasificación de imágenes, en su esencia, es la tarea de asignar una etiqueta a una imagen a partir de un conjunto predefinido de categorías. En la práctica esto significa que nuestra tarea es analizar una imagen de entrada y devolver una etiqueta que categorice la imagen. La etiqueta siempre proviene de un conjunto predefinido de posibles categorías.” (Martínez, 2022)

Existen varios métodos para la clasificación de imágenes, los principales son:

2.3.1. Clasificación supervisada

Los algoritmos de clasificación supervisada requieren de un conocimiento previo de la zona u objeto de estudio, que se puede adquirir tanto por la experiencia previa o por la realización de los trabajos de campo. Esto quiere decir, que el intérprete debe estar familiarizado con el área de interés, para que pueda interpretar y delimitar sobre la imagen, áreas representativas, denominadas áreas o regiones de interés. Los pasos para realizar la clasificación supervisada son las siguientes: (Ingeoexpert, s.f.; Martínez, 2022)

- **Etapa de entrenamiento**: En esta etapa el intérprete debe identificar las áreas de entrenamiento representativas, y a su vez genera una descripción numérica de todos los atributos espectrales de cada una de las categorías.
- **Etapa de clasificación**: Cada pixel se categoriza y se asocia a una determinada categoría de acuerdo a la mayor semejanza que presente. En caso de que un pixel no sea lo suficientemente similar, se catalogará como **desconocido**.
- **Etapa de análisis de precisión y verificación de resultados**: Para la estimación de exactitud de un clasificador, se debe de tener cierto grado de concordancia entre las clases asignadas por el clasificador. Por lo tanto, si el conjunto de píxeles es lo suficientemente similar, se clasificará en la categoría que más se asemeje.

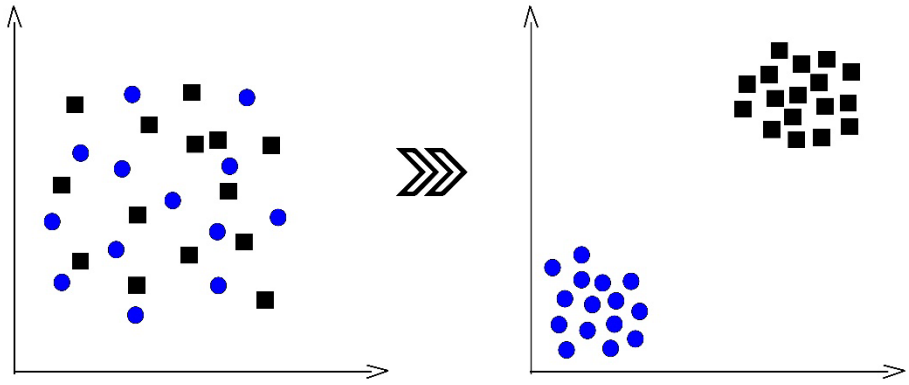

Figura: 2.1: Ejemplo de clasificación supervisada

2.3.2. Clasificación no supervisada

En la clasificación no supervisada hace uso de algoritmos para agregar píxeles en los grupos naturales o clúster con características espectrales similares que se presentan en la imagen. En esta clasificación, no se hace uso de un analista durante el proceso de clasificación y no existe la etapa de entrenamiento. Sin embargo, los resultados suelen ser poco exactos, esto es un paso que es obligatorio, porque sirve para que el

intérprete pueda asociar los tipos de coberturas a los *clusters* generados por la información auxiliar.

Una vez que se ha realizado la clasificación no supervisada se debe de analizar estadísticamente todas las clases resultantes. Se deben comprobar las correspondencias que existen entre las clases espectrales y las clases temáticas (Figura 2.2). Los resultados que se obtengan en esta clasificación, pueden servir para definir otras áreas de entrenamiento para las clasificaciones supervisadas. (Ingeoexpert, s.f.; Martínez, 2022)

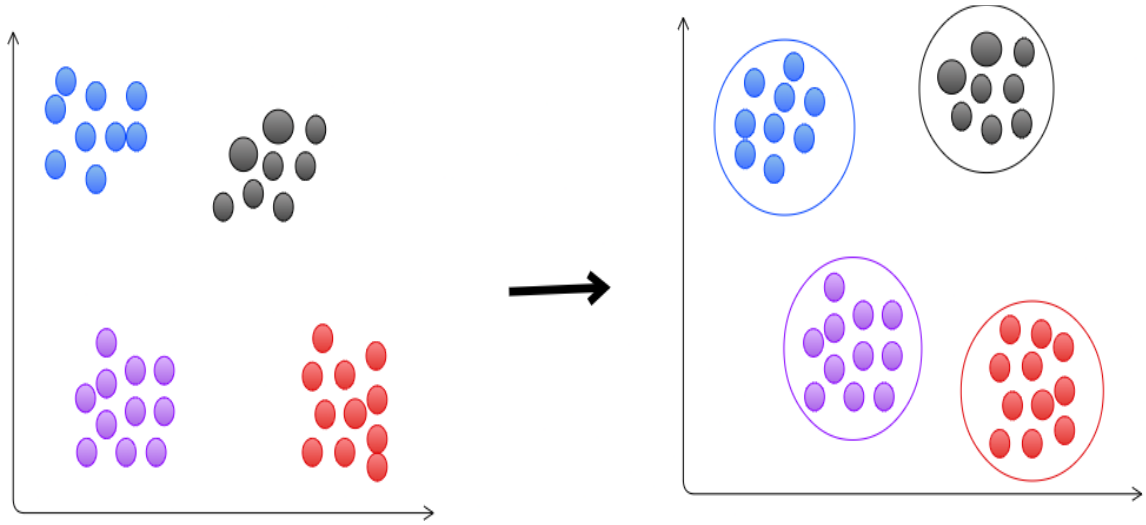

Figura 2.2: Ejemplo de clasificación no supervisada.

2.3.3. Clasificador por distancia mínima (DM)

En la clasificación por distancia mínima, en su fase de aprendizaje consiste únicamente en la elección del mejor representante de cada clase en el conjunto de prototipos que pertenezcan a la misma. Por lo general el representante elegido suele ser aquel que se encuentra más centrado dentro de la distribución de los prototipos de

la clase. Para este clasificador la muestra **x** se clasifica dentro de la clase, cuyo representante se encuentra a la menor distancia. (Knudby, s.f.)

En la Figura 2.3 se puede observar una representación gráfica del funcionamiento de este clasificador, dado un ejemplo de clases c_i, c_j y c_k, representados por los prototipos p_i, p_j y p_k en espacios de representación vectorial con un producto escalar. En la figura se puede observar que **x** se clasificará dentro de la clase c_k, ya que se encuentra más cercano a dicha clase. Las fronteras de decisión para este tipo de espacios que separan las clases son hiperplanos, mediatrices de los segmentos formados por todos los pares de prototipos.

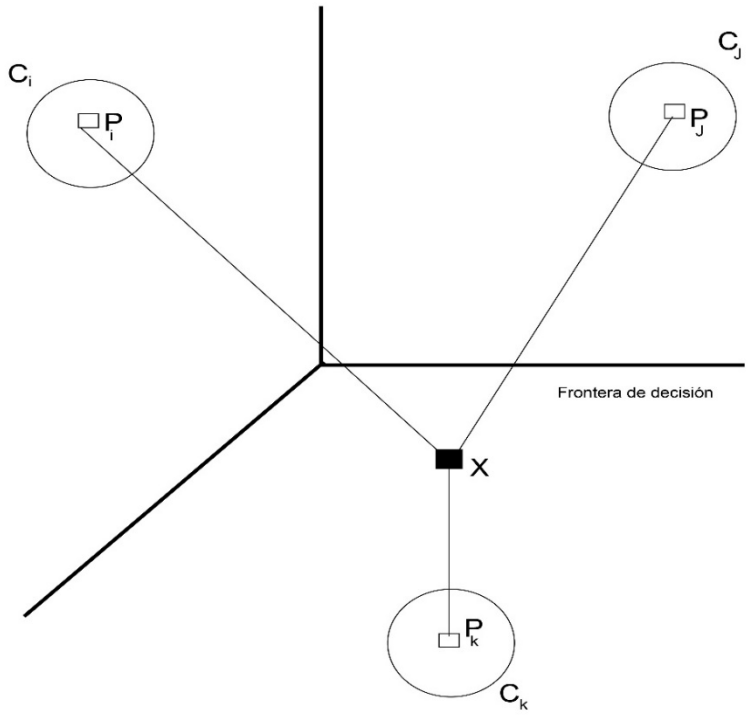

Figura 2.3: Ejemplo de clasificador por Distancia Mínima

2.4.4. Clasificador por Distancia Euclidiana o Distancia Euclídea

La clasificación por distancia euclidiana permite medir la distancia que existe en línea recta entre dos puntos en un espacio n-dimensional.

La principal funcionalidad de este clasificador se destaca por su utilidad para determinar la similitud entre dos imágenes o entre pares de datos. Teniendo esto en cuenta, podemos utilizar la similitud calculada a partir de los datos iniciales para otorgar un sistema de consultas de recomendación. Con ello podemos conseguir un sistema de datos que puedan identificar elementos que contengan características similares, con una “puntuación” o valoraciones que pueda brindar un resultado para la clasificación de la imagen. (Graph Everywhere, s.f.)

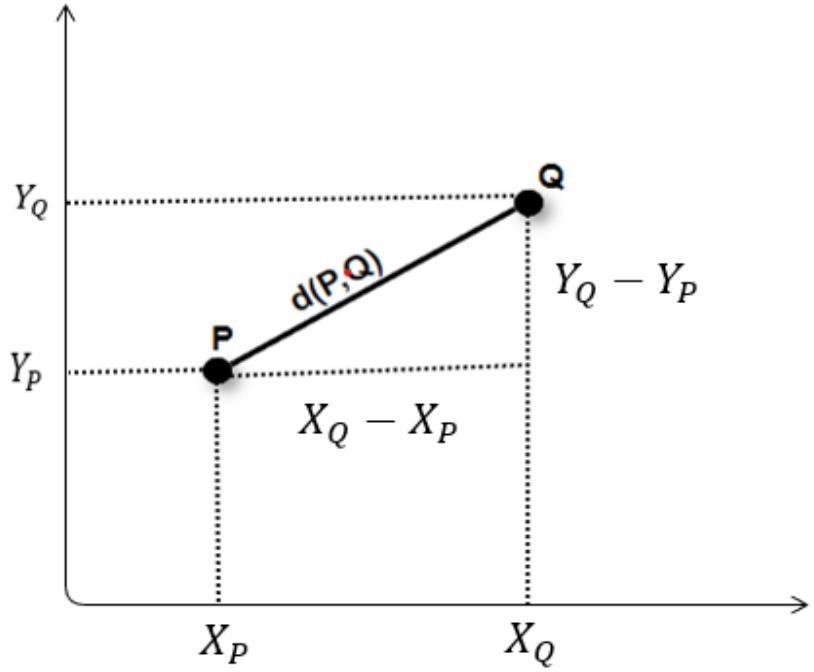

Figura 2.4: Representación gráfica de Distancia Euclidiana

2.4. Técnicas de Reconocimiento Facial usada por Face-api

Como se observó en el punto anterior existen diferentes técnicas utilizadas para llevar a cabo el reconocimiento facial, aunque cada técnica tiene sus ventajas y desventajas,

los sistemas de reconocimiento facial utilizan diferentes técnicas. En nuestro caso nos enfocaremos en las técnicas que utiliza la API Face-api.

Cabe mencionar que Face-api utiliza dos técnicas de reconocimiento facial las cuales son:

- Holísticas
- Geométricas

Como se mencionó en la **Sección 2.2.1**, Face-api utiliza una serie de procedimientos para poder identificar a una persona con las imágenes que se encuentran en la Base de Datos y con ello registra las partes más importantes del rostro.

Por su parte en la **Sección 2.2.2** se hace mención del Reconocimiento Facial 2D, ya que Face-api utiliza las curvas y líneas que se trazan sobre el rostro para medir la distancia que existe entre los puntos distintivos del rostro, además de que va buscando comparaciones con todas las imágenes de la Base de Datos para determinar cuál es el rostro que se asemeja con el que se está detectando.

2.5. Clasificador de imágenes utilizado por Face-api

Como se observó en la **Sección 2.3** existen cuatro métodos de clasificación que son los más utilizados en cuanto a clasificación de imágenes se refiere, sin embargo, Face-api utiliza el clasificador por “Distancia Euclidiana”. Este clasificador se utiliza para clasificar los “descriptores” del rostro el cual es un vector de 128 valores que describen el rostro de una persona.

Ya que Face-api, tiene dos métodos para poder hacer el reconocimiento facial, los cuales son por el algoritmo MTCNN y el otro mediante la primer red neuronal del algoritmo MTCNN, los Puntos de Referencia y Descriptores, debido a las limitantes del equipo utilizado se hará uso de la segunda opción (el cual se detallará a fondo en el siguiente capítulo), la cual se hace de la siguiente manera:

- Se debe detectar un rostro humano con los cuadros delimitadores.
- Una vez que se tiene el cuadro delimitador se extraen los puntos de referencia para corroborar que es un rostro humano el que se detectó.
- Una vez que se tiene identificado al rostro, se obtienen los descriptores para poder clasificar a la imagen si es de una persona conocida o no.
- Una vez que se obtuvieron los descriptores faciales, el método de Distancia Euclidiana compara los descriptores con los descriptores de la(s) imagen(es) de la Base de Datos y determina si la persona fue detectada o no.

2.6. Tecnologías para aplicaciones Web

Actualmente existen varias opciones en cuanto a tecnologías para desarrollo de aplicaciones Web, estas nos permiten desarrollar tanto para el *frontend* y *backend*, a continuación, se mencionan algunos ejemplos de estas tecnologías.

Algunas de las tecnologías para el desarrollo del *frontend* se encuentran los siguientes ejemplos:

- HTML
- CSS
- JavaScript

Los lenguajes anteriormente mencionados son los más comunes para el desarrollo de *frontend* en aplicaciones Web. También se cuenta con *frameworks* que de igual manera nos permiten el desarrollo de aplicaciones Web con la ventaja que se puede desarrollar sistemas más robustos, se tiene mayor rapidez en el desarrollo, no se requiere crear una estructura desde cero, se evita escribir código repetitivo, entre los ejemplos están:

- Angular
- Vue.JS

- React

En cuanto a tecnologías para el desarrollo del *backend* tenemos los siguientes ejemplos:

- PHP
- Python
- Java EE

Estos lenguajes son algunos que nos permiten el desarrollo de la parte del servidor de nuestra aplicación. También existen *frameworks* para el desarrollo del backend para el desarrollo de aplicaciones Web con la ventaja de que estos *frameworks* tenemos una rápida ejecución en el código, se tiene una transmisión rápida de datos, algunos ejemplos son:

- NodeJS
- Laravel

Y por último tenemos algunas tecnologías que nos permiten gestionar las Base de Datos de nuestra aplicación, algunos ejemplos son:

- MySQL
- MongoDB

Como se mencionó existen varias tecnologías que nos permiten desarrollar aplicaciones web, sin embargo, para este proyecto se hará uso de las siguientes tecnologías, ya que permitirán desarrollar el prototipo de forma rápida y robusta, tener un mejor manejo de los archivos (imágenes) y sobre todo de ser compatible con el lenguaje de la API de reconocimiento facial:

- Angular: Se hará uso del *framework* para el desarrollo de la interfaz de usuario, ya que utiliza HTML y CSS para construir dichas interfaces, además de brindar compatibilidad con JavaScript, que es donde se implementará la funcionalidad principal del reconocimiento facial.

- NodeJS: Se hará uso del *framework* para el desarrollo del servidor de imágenes, ya que el sistema guardará las imágenes de las personas que se desean identificar.
- MySQL: Se hará uso del gestor de MySQL para guardar la información de la persona, los cuales son: identificador de usuario (id), nombre de la persona y la ubicación de la imagen de la persona.
- JavaScript: En este lenguaje se desarrolló toda la funcionalidad del reconocimiento facial, ya que las principales funcionalidades del API son utilizadas con JS.

Además, se hará uso de algunas bibliotecas para el desarrollo del servidor de imágenes, los cuales facilitarán el desarrollo:

- Express.JS: Es un marco de desarrollo para NodeJS, que permite la estructuración de una aplicación Web de manera ágil y profunda, además de proporcionar algunas funcionalidades como el enrutamiento, opciones para poder gestionar sesiones y *cookies*, entre otros.
- Multer: Es un *middleware* para Express y NodeJS, el cual nos permite la manipulación de archivos más fácilmente.
- Cors: Nos permite acceder a los recursos que se encuentran en el *backend* y *frontend*, mediante el protocolo y encabezados HTTP, ya que el servidor se encuentra en un puerto y la aplicación Web en otro puerto diferente y con el uso de *cors*, evitamos los errores en las peticiones.
- Morgan: Es un *middleware* que captura las solicitudes HTTP para NodeJS para el registro y seguimiento de las respuestas del servidor.

Capítulo 3

Desarrollo Teórico

3.1. Introducción

En este capítulo se explicará de forma teórica la funcionalidad de Face-api, usando como base lo visto en el capítulo anterior, ya que se debe entender cómo trabaja esta API de manera interna, se explicará cómo es capaz de detectar un rostro humano y cómo “sustrae” la información del rostro que se registra. A su vez también se analizará cómo realiza la detección del rostro con un algoritmo para la recolección de los datos del rostro, y también cómo compara los resultados para brindar un resultado final, el cual es el nombre de la persona que se detectó en un video.

3.2. Algoritmo MTCNN para determinar la localización del rostro

Antes de empezar a explicar el funcionamiento del algoritmo MTCNN, debemos definir un término que se va utilizar en esta sección y al que se va hacer mención durante la explicación del algoritmo, el cual es “Cuadro Delimitador” (por su traducción del idioma inglés). El Cuadro Delimitador es un cuadro que la propia API “dibuja” sobre el rostro de la(s) persona(s) que ha detectado, ya sea en una imagen o en un video (Figura 3.1), o dicho de otra forma un Cuadro Delimitador es la sección de la imagen o video

donde se ha detectado un rostro humano, y la API lo dibuja para señalar que en esa sección se ha detectado un rostro.

Figura 3.1: Ejemplo de Cuadro Delimitador de Face-api.

Una vez definido lo que es un Cuadro Delimitador, vamos a detallar cómo funciona el algoritmo MTCNN. El algoritmo MTCNN (Multi-task Cascaded Convolutional Neural Network por su significado en inglés o Red Neuronal Convolucional en Cascada Multitarea por su traducción al español), es un algoritmo que nos permite la detección de uno o varios rostros tanto en imágenes como en videos usando una conexión en cascada de tres Redes Convolucionales:

- La primera red convolucional es denominada **"Red de propuestas o P-net",** la cual se encarga de recibir la imagen de entrada tanto en su tamaño original y esa misma imagen en diferentes tamaños de escala, con el fin de que la red pueda detectar rostros de diferentes tamaños dentro de la imagen de entrada (Figura 3.2). Esta primera red contiene pocas capas que permiten detectar con rapidez si la imagen contiene rostros, ya que va generando varios **"Cuadros delimitadores"**, estos son pequeños "cuadros", las cuales tienen secciones de la imagen para poder obtener información sobre si la imagen de entrada contiene un rostro. Si se obtienen varios cuadros delimitadores de tamaño similar y estos se encuentran en la misma porción de la imagen, entonces, se hace uso de un algoritmo llamado "**non-max suppression"**, la tarea de este

algoritmo es analizar la probabilidad de que cada cuadro delimitador contenga un rostro, el cual es generado por el clasificador de la red.

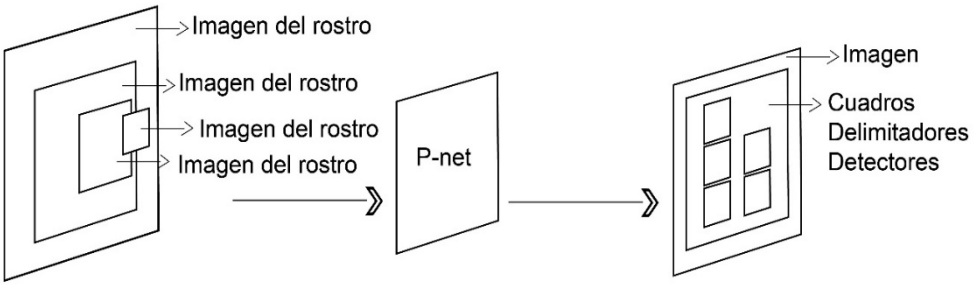

Figura 3.2: Ejemplo de la red convolucional P-net.

- En la segunda red convolucional denominada como **"Refinar red o R-net"**, toma los cuadros delimitadores junto con la imagen original, como esta segunda red es más profunda, esto permite "refinar" los resultados anteriores y elimina algunos cuadros delimitadores que fueron generados por la red anterior (P-net), y mejora la precisión en las coordenadas (x,y) de la imagen de los nuevos cuadros delimitadores (Figura 3.3). Una vez obtenido los nuevos resultados con las coordenadas más precisas, se vuelve a hacer uso del algoritmo *non-max suppression* y los manda a la tercera red.

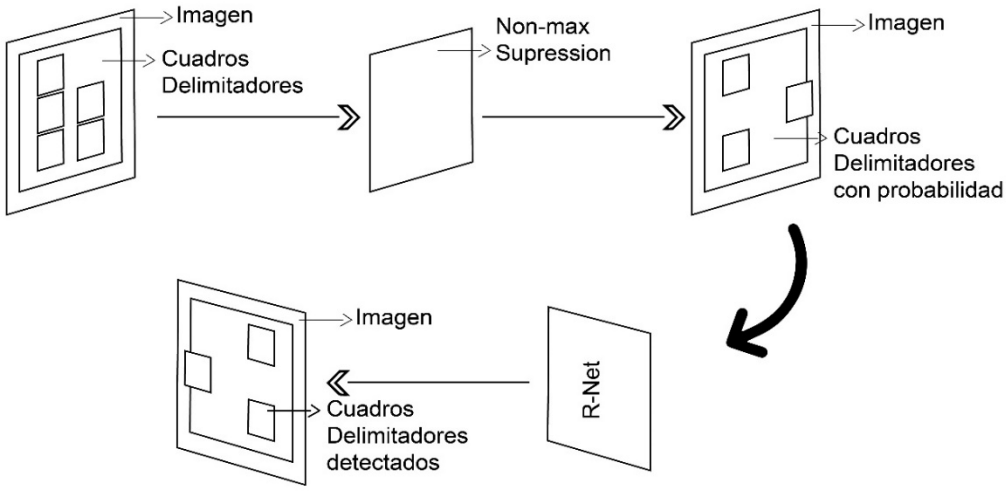

Figura 3.3: Ejemplo de la red convolucional R-net.

- En la tercera red convolucional denominada como **"Red de salida o O-net"**, toma los nuevos cuadros delimitadores y la imagen original, esta tercera red es más profunda que las dos redes anteriores, tomando esos resultados para formar un solo cuadro delimitador preciso para cada imagen de entrada (Figura 3.4). Aunque este algoritmo tiene una alta precisión en cuanto al resultado se refiere, presenta una desventaja, ya que utiliza tres redes convolucionales que requiere de una mayor cantidad de procesamiento, pero aplicado en sistemas robustos es una excelente opción. (Cordobés Menguiana, 2019; Sotaquirá, 2020)

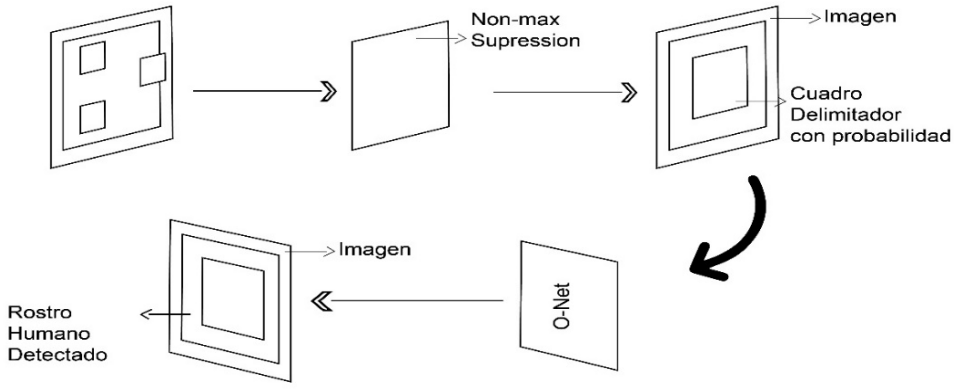

Figura 3.4: Ejemplo de la red convolucional O-net.

Una vez visto cómo funciona el algoritmo MTCNN analizaremos cómo se utiliza en Face-api, aunque cabe mencionar que la API tiene dos formas de utilizar el algoritmo MTCNN, una es la forma “completa” haciendo uso de las tres redes convolucionales en cascada, y para ello se debe utilizar la librería propia de Face-api, la cual es la siguiente:

```
1 await faceapi.loadMtcnnModel('/')
2 await faceapi.loadFaceRecognitionModel('/')
```

Aunque por las limitaciones del hardware con el que se cuenta, se hará uso de la primera red Convolucional del algoritmo MTCNN, y por lo cual se usarán las siguientes librerías:

```
1 faceapi.nets.faceRecognitionNet.loadFromUri('/'),
2 faceapi.nets.faceLandmark68Net.loadFromUri('/')
```

Las bibliotecas anteriores nos permiten realizar el reconocimiento facial con las bases del algoritmo MTCNN, la tarea que realizan las bibliotecas es tomar las características principales de la primera red convolucional (P-net), la cual nos permite hacer detecciones de varios rostros en el mismo espacio y hacer varias escalas de la imagen para poder tomar los cuadros delimitadores, esto con el fin de poder identificar un rostro a diferentes distancias focales. Y una vez detectado los cuadros delimitadores con el rostro detectado, realiza la extracción de los “Vectores de rasgos”, para obtener la identificación del rostro.

3.3. Extracción de los Puntos de Referencia

Antes de empezar a explicar cómo es el procedimiento de la extracción de los Vectores de rasgos, vamos a definir qué son los “Puntos de Referencia”, los Puntos de Referencia son “puntos” que la propia API “dibuja” sobre el rostro, esto con el fin de “trazar” medidas sobre las partes más representativas del rostro de una persona, como

los son los ojos, las cejas, nariz, boca y mentón, con el fin de hacer mediciones entre esos puntos para dictaminar los rasgos faciales de la persona. Dependiendo de las herramientas de reconocimiento facial que se use, puede utilizar una cantidad de puntos de referencia más que otros, para esta API que se eligió utiliza un total de 68 puntos (Figura 3.5), los cuales son dibujados sobre la imagen o video, esto con el fin de alinear el rostro hacia el centro del cuadro delimitador que se obtuvo anteriormente. (Khabarlak & Koriashkina, 2022; Ramos Almeida, 2018; Zhu & Ramanan)

Figura 3.5: Ejemplo de cómo se dibujan los Puntos de referencia con Face-api.

Podemos caracterizar a una región (imagen de la persona) por $(\mathrm{x}, \mathrm{y}, \mathrm{w}, \mathrm{h})$ donde (x,y) son las coordenadas del centro de la región y (w,h) representan el ancho y altura de la imagen con respecto al centro de la región (x,y) y se representa con la siguiente fórmula (Ranjan, Patel, & Chellappa, 2016) :

$$(a_i, b_i) = \left(\frac{x_i - x}{w}, \frac{y_i - y}{h}\right) \qquad \textbf{(E-1)}$$

Donde:

- (x_i, y_i) son las coordenadas de puntos de referencia verdaderas de la imagen.

- (a_i, b_i) son las coordenadas donde se ubican los puntos de referencia, que servirán para alinear el rostro.
- w representa el ancho de la imagen
- h representa la altura de la imagen

Dado que para la detección del rostro en condiciones reales (detección del rostro en los videos), puede ocurrir una pérdida al predecir la ubicación de los puntos de referencia, ya que tanto la iluminación, el movimiento del rostro y la resolución de la cámara llega a contribuir a la pérdida de información, para ello se utiliza la siguiente fórmula para reducir la pérdida al momento del cálculo de los puntos de referencia (Ranjan, Patel, & Chellappa, 2016) .

$$Pérdida_L = \frac{1}{2N}\sum_{i=1}^{N} v_i\left((\widehat{x_i} - a_i)^2 + ((\widehat{y_i} - b_i)^2)\right) \qquad \textbf{(E-2)}$$

Donde:

- $(\widehat{x_i}, \widehat{y_i})$ representa la i-ésima ubicación del punto de referencia predicha por la red
- N es el número total de puntos de referencia
- v_i representa el factor de visibilidad, y equivale a 1 si el i-ésimo punto de referencia es visible en la región candidata, de lo contrario equivale a 0.

Lo anterior ayuda a reducir los errores al momento de la extracción de los puntos de referencia, ya que, si algunos de los puntos de referencia no se logran identificar, no son considerados para el resultado final.

Una vez que el algoritmo MTCNN ha detectado los rostros, Face-api ha detectado los 68 puntos de referencia del rostro o de los rostros, ya que como se mencionó en el

punto anterior, la API es capaz de detectar varios rostros a la vez y a diferentes distancias focales. En la Figura 3.6 se puede observar del lado izquierdo como Face-api dibuja los puntos de referencia sobre la imagen y del lado derecho es como la api representa esos puntos a nivel de datos, que son matrices de datos en forma de coordenadas (x, y), que son el resultado que se obtiene de las fórmulas **(E-1)** y **(E-2)**.

Figura 3.6: Puntos de Referencia a nivel de datos

Los puntos de referencia son utilizados para que el rostro o los rostros detectados sean alineados de manera central para que la detección del rostro sea lo más precisa posible. Cuando la red P-net ha determinado los cuadros delimitadores donde se ha detectado un rostro, se deben de extraer los puntos de referencia, con el fin de centrar el rostro y así obtener una detección más precisa, ya que como se pudo observar anteriormente, estos puntos están representados por coordenadas que dan la ubicación de cada parte del rostro, y una vez que ya se ha alineado el rostro pasa a la última fase, la clasificación de la imagen para poder identificar a la(s) persona(s).

3.4. Clasificación de imágenes por método de Distancia Euclidiana

Como se observó en la **Sección 2.3** existen 4 métodos principales de clasificación de imágenes, sin embargo, Face-api utiliza el método de **clasificación por distancia Euclidiana,** ya que hace uso de los "Descriptores Faciales" que son las características del rostro humano que se mapean. En total son 128 valores descriptivos los que obtiene del rostro, estos se obtienen de las imágenes base que se encuentran en el servidor de imágenes, y también se sustraen los descriptores del rostro de la imagen o video que se está analizando.

Es por eso que la API utiliza distancia euclidiana para medir la distancia que existen entre el descriptor base (imagen del servidor) y el descriptor de entrada (del video o imagen a analizar) y para poder clasificarla como detectado o desconocido maneja un "umbral" de resultados que es de "0.6" (este umbral está definido por la propia Face-api), esto quiere decir que si el cálculo o la semejanza entre el descriptor base y descriptor calculado es menor o igual a el umbral, se puede tener la certeza que hemos identificado a la persona, si el cálculo del descriptor es mayor al umbral, pasa a el siguiente descriptor base, y si ha pasado por todos los descriptores base, entonces se determina como "desconocido", esto los podemos ver matemáticamente de la siguiente manera:

Dado $d_{i,j}$ la distancia euclidiana (Arriagada Rodríguez, 2015) que existe entre los descriptores base y los descriptores entrantes, y sean $1,2, \ldots, n$, el número de descriptores totales. Tenemos que $d_{i,j}$ se define como:

$$d_{i,j} = \sqrt{\sum_{k=1}^{n} (P_{xi} - Q_{xj})^2} \qquad \textbf{(E-3)}$$

donde:

- $d_{i,j} =$ Desconocido si es > 0.6 / Detectado si es ≤ 0.6
- $n = 1, 2, \ldots, 128$, el número total de descriptores
- P_{xi}, los descriptores de la imagen base
- Q_{xj}, los descriptores de la imagen entrante

Ahora veremos de manera práctica como usando la fórmula de Distancia Euclidiana ***(E-3)*** se clasifican las imágenes para poder arrojar el resultado final, el cual es dar el nombre de la persona que se detectó. Lo primero, dada una matriz de descriptores de la imagen base:

$$P_B = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \qquad \textbf{(E-4)}$$

Donde P_B representa al conjunto de los descriptores en condiciones ideales, esto quiere decir que son los descriptores que son extraídos de la(s) imagen(es) base, $x_1, x_2, x_3 \cdots x_n$ son las representaciones de cada descriptor del rostro, y n representa el total de descriptores que maneja Face-api, los cuales son 128.

Para el conjunto de los descriptores de las personas que se van a identificar para comparar se vería de la siguiente manera:

$$Q_1 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \quad \textbf{(E-6)} \qquad Q_2 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \quad \textbf{(E-5)} \qquad Q_3 = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix} \quad \textbf{(E-7)}$$

Donde $Q_1,\ Q_2\ y\ Q_3$ representan al conjunto de los descriptores en condiciones reales, esto quiere decir, que son los descriptores que son extraídos de los videos y decimos que son en "condiciones reales", ya que la persona está en movimiento, la luz de fondo puede variar, o puede haber demasiado movimiento y ruido ambiental, $x_1, x_2, x_3 \cdots x_n$ representa cada descriptor del rostro, y n es la representación de los descriptores faciales totales que maneja Face-api.

Tomando como primer caso un escenario donde se tiene una sola imagen base **(E-4)** y una persona detectada **(E-5)**, la distancia euclidiana se vería de la siguiente forma:

$$P_B = \begin{pmatrix} 0.0843948349 \\ 0.0629549100 \\ -0.0037903180 \\ \vdots \\ 0.3024762508 \end{pmatrix} \begin{pmatrix} -0.0440946349 \\ 0.0087683373 \\ -0.0249559102 \\ \vdots \\ 0.3127702902 \end{pmatrix} = Q_1$$

Figura 3.7: Obtención de la Distancia Euclidiana entre la imagen base y entrante.

En la Figura 3.7 se puede observar cómo se obtiene la distancia euclidiana con los descriptores de cada imagen, tomando la **(E-3)** los vectores se ven de la siguiente manera:

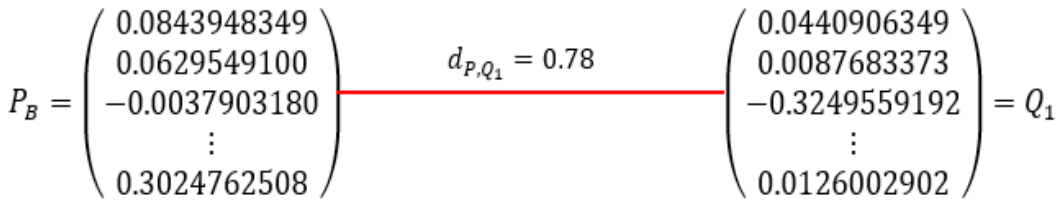

Figura 3.8: Distancia Euclidiana entre imagen base y un rostro detectado.

En la Figura 3.8 se puede observar que el resultado de la distancia euclidiana es de 0.78, la API maneja el umbral de 0.6, esto quiere decir que Q_1 no es la persona que se desea detectar y por lo tanto será clasificado como "desconocido", ya que resulta ser que $d_{P_B,Q_1} > 0.6$ y dará como resultado desconocido.

Tomando como segundo caso un escenario donde se tiene una sola imagen base **(E-4)** y varias personas detectadas **(E-5)**, **(E-6)** y **(E-7)**, la distancia euclidiana se vería de la siguiente forma:

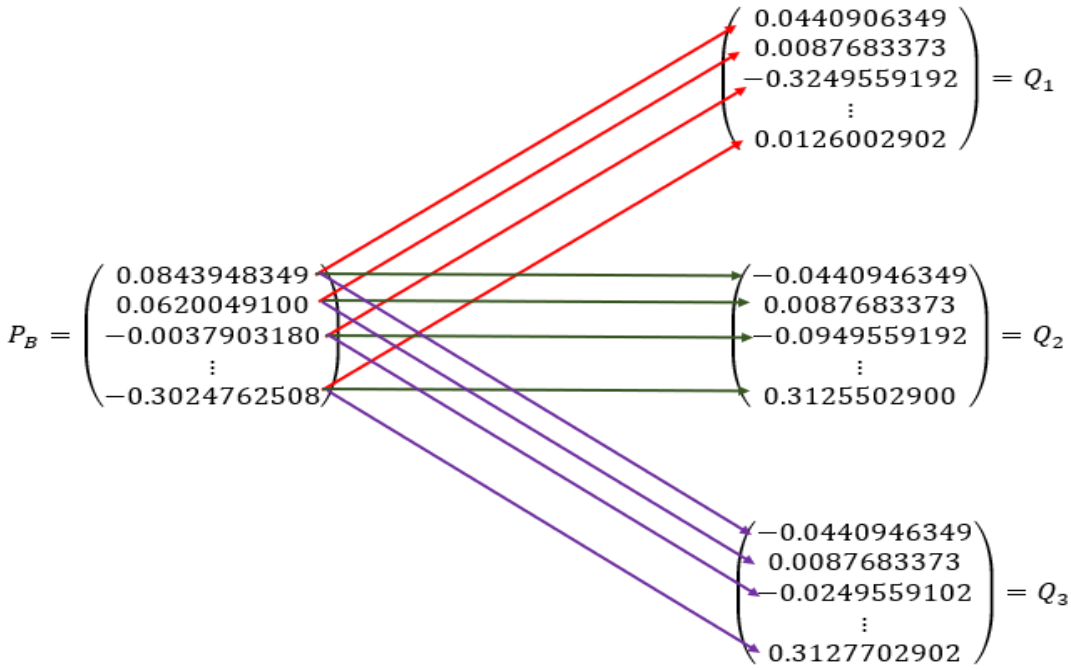

Figura 3.9: Obtención de la Distancia Euclidiana entre la imagen base y varios rostros de entrada.

En la imagen anterior se puede observar cómo se obtiene la distancia euclidiana con los descriptores de la imagen base con los descriptores de los rostros de entrada, tomando la **(E-3)** los vectores se ven de la siguiente manera:

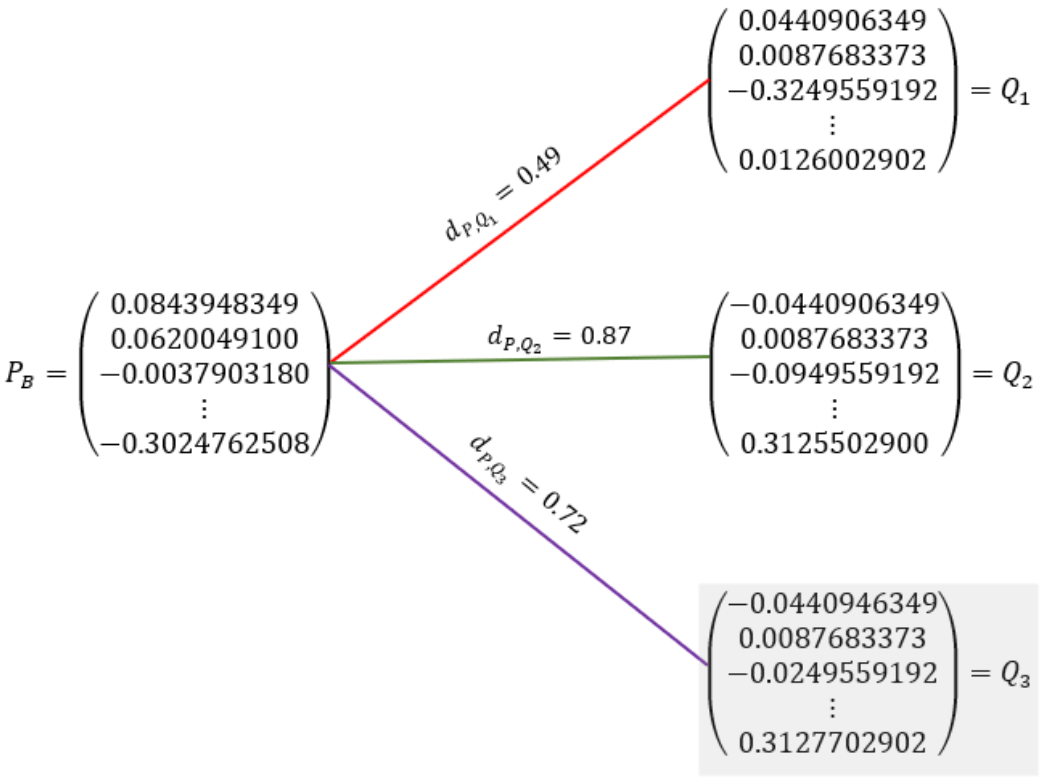

Figura 3.10: Distancia Euclidiana entre imagen base y varios rostros detectados.

En la imagen anterior se puede observar que el resultado de la distancia euclidiana que existe entre la imagen base y los rostros detectados, como se observa la imagen base se compara con todos los rostros detectados para así poder clasificar a los rostros, en este ejemplo Q_1 será clasificado dentro de P_B ya que $d_{P_B,Q_1} \leq 0.6$, mientras que los restantes serán clasificados como “desconocidos”.

Tomando como tercer caso un escenario donde se tienen varias imágenes base y varias personas detectadas, la distancia euclidiana se vería de la siguiente forma:

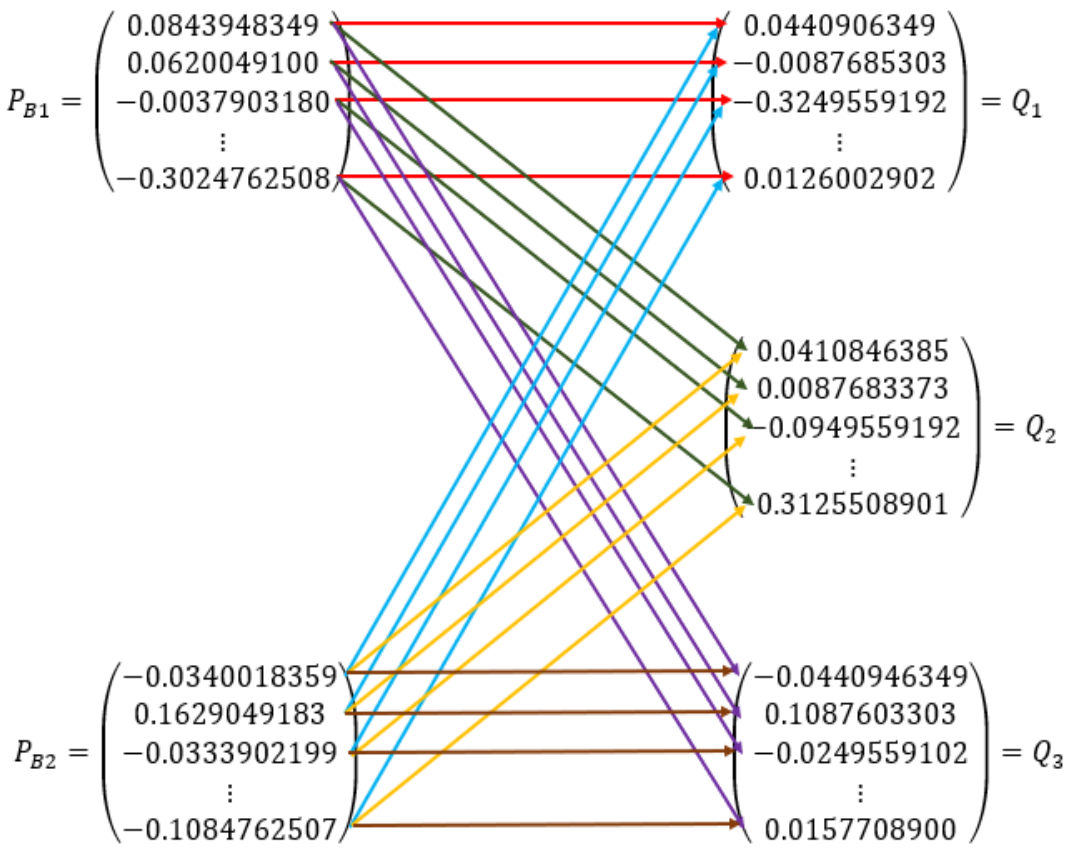

Figura 3.11: Obtención de la Distancia Euclidiana entre varias imágenes base y varios rostros de entrada.

En la imagen anterior se puede observar cómo se obtiene la distancia euclidiana con los descriptores de las imágenes base con los descriptores de los rostros de entrada, tomando la **(E-3)** los vectores se ven de la siguiente manera:

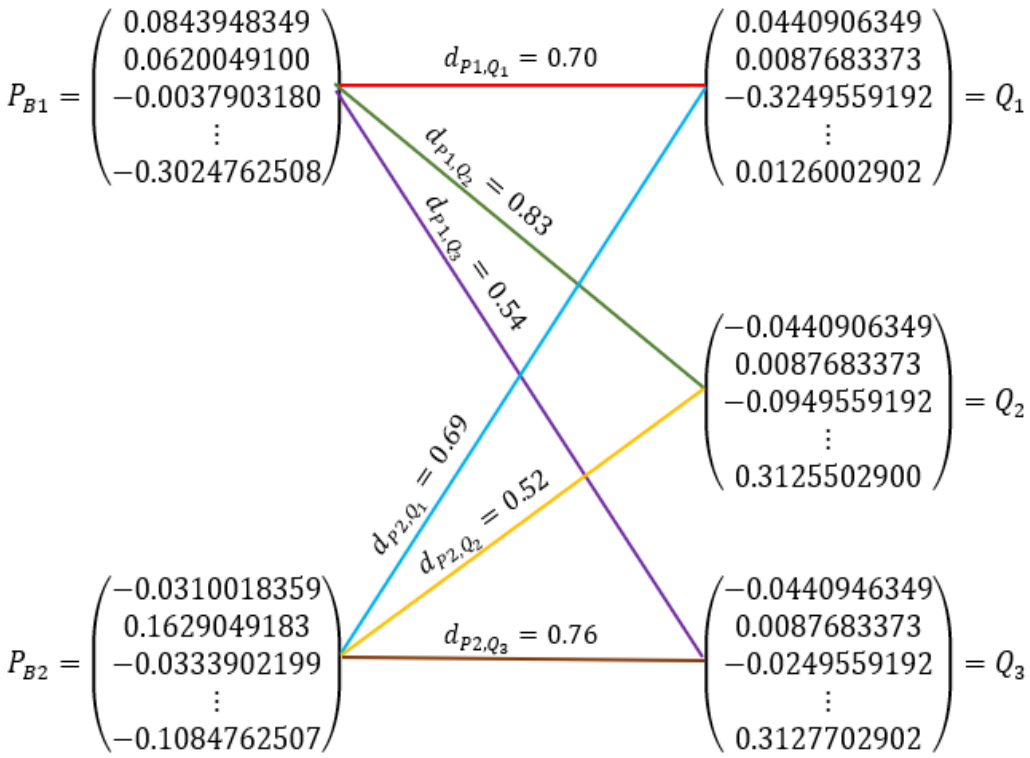

Figura 3.12: Distancia Euclidiana entre las imágenes base y varios rostros detectados.

En la imagen anterior se puede observar que el resultado de la distancia euclidiana que existe entre las imágenes base y los rostros detectados, en este ejemplo Q_3 será clasificado dentro de P_{B1} ya que $d_{P_{B1},Q_3} \leq 0.6$ y Q_2 será clasificado dentro de P_{B2} ya que $d_{P_{B2},Q_2} \leq 0.6$, mientras que los restantes serán clasificados como "desconocidos".

Capítulo 4
Desarrollo

4.1. Introducción

En este capítulo se describe la metodología utilizada para dar solución a la problemática que se planteó al inicio del documento, así mismo se presentarán los distintos diagramas UML para el sustento formal del desarrollo en cuanto a ingeniería de software se refiere, al igual que su funcionamiento del prototipo propuesto.

Cabe mencionar que también se pondrá parte del código fuente, sobre todo el código con el que el API hace la tarea del Reconocimiento Facial, así como el manejo de los datos biométricos del rostro.

4.2. Análisis de requerimientos

En esta etapa de análisis de requerimientos se dará a conocer el alcance del proyecto, así como las especificaciones de los requerimientos funcionales y no funcionales para llevar a cabo el desarrollo del proyecto.

4.2.1. Alcance del proyecto

Elaborar un prototipo de videovigilancia que cuente con reconocimiento facial para la detección de personas mediante la reproducción de videos que se tengan almacenados.

4.2.2. Funcionalidad del producto

El prototipo de videovigilancia con reconocimiento facial tendrá como principales funcionalidades las siguientes características:

- Permitir al usuario la subida de videos de cualquier formato de video (mp4, mkv, etc.).
- El usuario podrá reproducir dichos videos con el reproductor multimedia que contendrá el prototipo.
- Despliegue de la información de la detección (nombre de la persona) y del video (hora y fecha).
- El usuario podrá subir y eliminar las imágenes de las personas que necesite identificar.

4.2.3. Tipos de usuarios.

El prototipo de videovigilancia contará con un solo tipo de usuario, como se describe a continuación.

Tipo de usuario	Usuario general
Habilidades	Manejo de equipos de cómputo. Manejo de navegador de Internet.
Actividades	Administrar las imágenes, subir y eliminar imágenes del servidor. Subir videos de videovigilancia que hayan sido almacenados para su revisión. Reproducir los videos para ver los resultados de la detección.

Tabla 4.1: Tipo de usuario.

4.2.4. Entorno operativo

A continuación, se enlistan los elementos tanto de software como de hardware para que el prototipo pueda ser utilizado:

Software

- Sistema operativo Windows 8, 8.1, 10.
- Navegador web (Chrome, Firefox, Brave, etc.).
- Archivos multimedia (videos con formato mp4, mkv, etc.)

Hardware

Contar con un equipo de cómputo que cuente como mínimo con las siguientes especificaciones:

- Procesador:
  - ❖ AMD Ryzen 3 2300U (2.0 GHz, 4 MB de caché).
  - ❖ Intel Core i3-10110U (2.10 GHz, 4 MB de caché).
- GPU:
  - ❖ AMD Radeon (TM) Vega 6 Graphics.
  - ❖ NVIDIA GeForce GTX 1050 Max-Q.

- Memoria RAM de 12 GB a 2400 MHz.
- Disco duro HDD o SSD de 500 GB.
- Modem con internet.

4.2.5. Requerimientos funcionales

A continuación, se presentan los requerimientos funcionales para el Prototipo de videovigilancia con reconocimiento facial:

Identificador	RF-01
Título	Archivos multimedia.
Entrada	Video de videovigilancia.
Salida	Video cargado en la página.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	---
Descripción	
El prototipo deberá permitir al usuario la subida de archivos multimedia de cualquier tipo de formato.	

Tabla 4.2: Requerimiento funcional #1

Identificador	RF-02
Título	Reproductor multimedia.
Entrada	---
Salida	Reproductor multimedia funcional.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	---
Descripción	
El prototipo debe contar con un reproductor multimedia para permitir al usuario la reproducción de los archivos multimedia.	

Tabla 4.3: Requerimiento funcional #2

Identificador	RF-03
Título	Reconocimiento Facial.
Entrada	Video de videovigilancia.
Salida	Información sobre la detección.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	---
Descripción	
El prototipo debe contar con la funcionalidad de reconocimiento facial durante la reproducción de los videos.	

Tabla 4.4: Requerimiento funcional #3

Identificador	RF-04
Título	Tabla de resultados.
Entrada	Video de videovigilancia.
Salida	Datos impresos en tabla de resultados.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	---
Descripción	
El prototipo debe de contener una tabla de resultados en donde se impriman los sujetos que fueron en el video.	

Tabla 4.5: Requerimiento funcional #4

Identificador	RF-05
Título	Alta de imágenes.
Entrada	Imagen de alguna persona.
Salida	Imagen almacenada.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	----
Descripción	
El prototipo debe permitir al usuario subir imágenes para ser almacenadas en el servidor, para su posterior uso para proporcionar los puntos de referencia y descriptores faciales.	

Tabla 4.6: Requerimiento funcional #5

Identificador	RF-06
Título	Baja de imágenes.
Entrada	Seleccionar imagen a eliminar.
Salida	Imagen eliminada.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	---
Descripción	
El prototipo debe permitir al usuario eliminar una imagen cuando ya no sea de utilidad.	

Tabla 4.7: Requerimiento funcional #6

Identificador	RF-07
Título	Página de inicio.
Entrada	Navegador abierto.
Salida	Pantalla de inicio.
Prioridad	Alta
Versión	1.0
Requerimiento no funcional	RNF-01 RNF-02
Descripción	
El prototipo debe contar con una página de inicio donde se tenga las opciones de ingresar al apartado de “Reconocimiento Facial” y “Alta y Baja de Imágenes”.	

Tabla 4.8: Requerimiento funcional #7

4.2.6. Requerimientos no funcionales.

A continuación, se presentan los requerimientos no funcionales para el Prototipo de videovigilancia con reconocimiento facial:

Identificador	RNF-01
Título	Funcionamiento en navegadores web
Prioridad	Media
Versión	1.0
Descripción	
El prototipo debe estar desarrollado para poder funcionar en diferentes navegadores.	

Tabla 4.9: Requerimiento no funcional #1.

Identificador	RNF-02
Título	Interfaz del prototipo.
Prioridad	Media
Versión	1.0
Descripción	
El prototipo debe tener una interfaz de uso intuitiva y sencilla.	

Tabla 4.10: Requerimiento no funcional #2.

4.3. Diseño

En esta sección se detallará cómo está compuesto el prototipo de videovigilancia con reconocimiento facial, así como se definen los diferentes módulos que lo componen, se mostrará cómo el usuario interactúa con el prototipo; así como las acciones que podrá realizar.

En la Figura 4.1 se puede observar cómo está compuesto el prototipo de manera general, así como los módulos con los que contará.

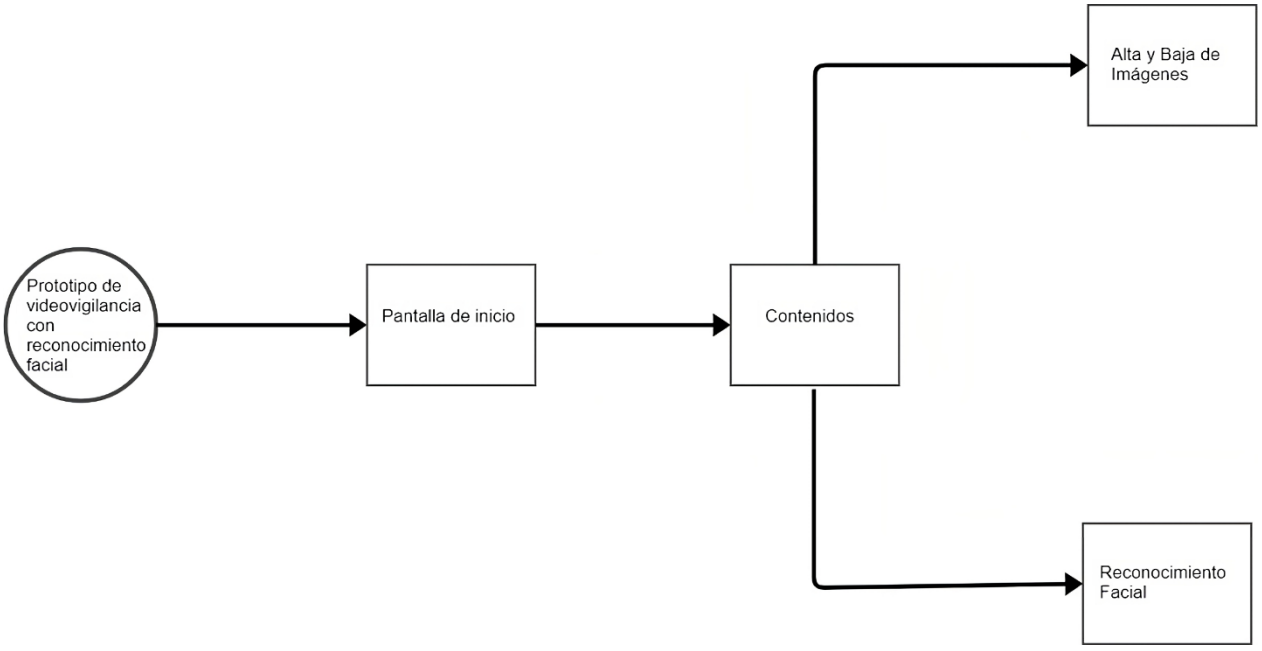

Figura 4.1: Diagrama de bloques general del prototipo de videovigilancia.

Como se puede observar en la imagen anterior, el prototipo contará con los siguientes módulos, de acuerdo a los requerimientos de la **Sección 4.2**:

- **Alta y Baja de imágenes de personas:** En este módulo la persona que utilice el prototipo tendrá la opción de subir las imágenes o fotografías del rostro de la persona, cabe mencionar que el nombre del archivo de la imagen debe de contener el nombre de la persona, dicha imagen se podrá subir al servidor de imágenes y quedará almacenada, de ser necesario tendrá la opción de eliminar la foto de la persona y se borrará del prototipo y del servidor de imágenes.

- **Reconocimiento facial:** En este módulo se podrán seleccionar videos, cabe mencionar que no se guardarán los videos, solo permanecerán en la página mientras se analice el video y despliegue la información.

4.3.1. Diagrama de casos de uso

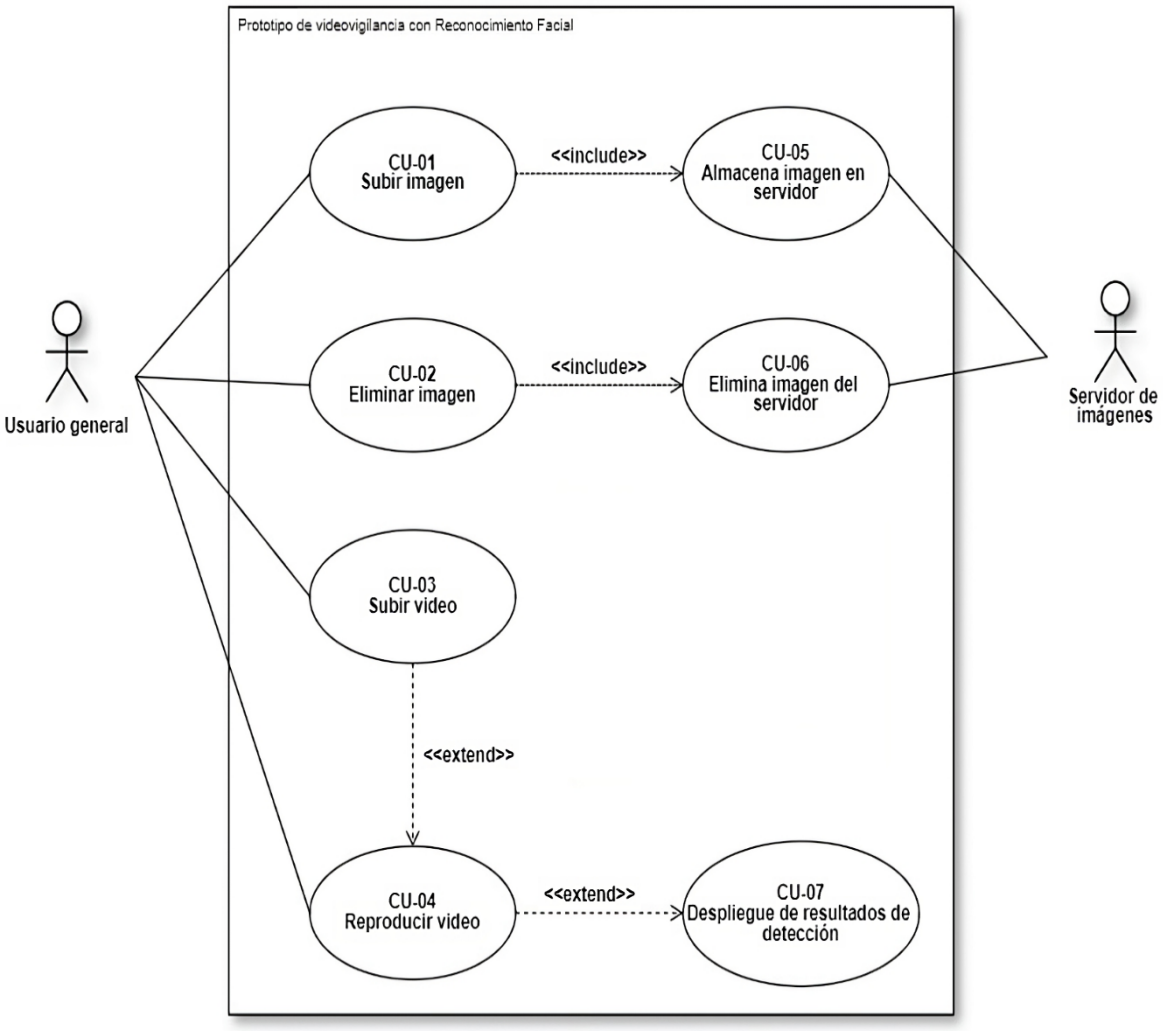

Figura 4.2: Diagrama de casos de uso (General).

4.3.1.1. Descripción CU-01

Caso de uso	Subir imagen
Actores	Usuario general
Resumen	El usuario puede subir una imagen o fotografía de las personas que desean identificar en videos de seguridad que fueron previamente obtenidos.
Precondiciones	El usuario debe subir imágenes con formato de imagen (jpg)
Postcondiciones	El usuario sube la imagen al servidor de imágenes
Incluye	CU-05
Extiende	--
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
1.- El usuario selecciona una imagen para subir.	2.- El sistema recibe la imagen y manda un mensaje de “imagen subida correctamente”. 3.- El sistema muestra en la pantalla la imagen con el nombre y un identificador (id).

Tabla 4.11: Descripción del Caso de uso de “Subir imagen”.

4.3.1.2. Descripción CU-02

Caso de uso	Eliminar imagen
Actores	Usuario general
Resumen	El usuario puede eliminar una imagen o fotografía que esté almacenada en el servidor, en el caso de que ya no se requiera registrar a una persona.
Precondiciones	El usuario no tiene ninguna restricción
Postcondiciones	El usuario elimina de forma definitiva la imagen del servidor
Incluye	CU-06
Extiende	--
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
1.- El usuario selecciona la imagen a eliminar	2.- El sistema manda un mensaje de alerta, antes de eliminar. 3.- De confirmar la eliminación, el sistema eliminará la imagen de forma permanente del servidor.

Tabla 4.12: Descripción del Caso de uso de “Eliminar imagen”.

4.3.1.3. Descripción CU-03

Caso de uso	Subir vídeo
Actores	Usuario general
Resumen	El usuario puede subir un video al sistema que necesite identificar a una o varias personas y saber la fecha y hora del video, el video se almacenará momentáneamente en la memoria cache del navegador hasta que el usuario decida subir otro video o salir de la opción de “Reconocimiento Facial”, ya que una vez salga de dicha opción el video se eliminara.
Precondiciones	El usuario debe subir videos con formato de video (mp4, mkv, etc.).
Postcondiciones	El usuario puede identificar a la o las personas que aparezcan en el video.
Incluye	--
Extiende	CU-04
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
1.- El usuario selecciona un video para analizar.	2.- El sistema acepta el video y está listo para la reproducción.

Tabla 4.13: Descripción Caso de uso “Subir vídeo”.

4.3.1.4. Descripción CU-04

Caso de uso	Reproducir video
Actores	Usuario general
Resumen	Una vez que el video fue seleccionado, el usuario podrá reproducir dicho video con el botón de reproducir.
Precondiciones	El usuario debió haber subido un video con anterioridad.
Postcondiciones	El video es reproducido y muestra los resultados de la detección
Incluye	--
Extiende	CU-07
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
1.- El usuario selecciona la opción de “reproducir”.	2.- El sistema muestra los resultados de la detección.

Tabla 4.14: Descripción Caso de uso “Reproducir video”.

4.3.1.5. Descripción CU-05

Caso de uso	Almacena imagen en servidor
Actores	Servidor de imágenes.
Resumen	El servidor recibe la imagen o fotografía del sistema de videovigilancia y la almacena.
Precondiciones	El servidor debe de recibir una imagen o fotografía.
Postcondiciones	El servidor almacena la imagen de forma permanente.
Incluye	--
Extiende	--
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
2.- El servidor “recibe” la imagen y la almacena de forma permanente, hasta que sea borrada.	1.- El sistema envía la imagen que el usuario subió previamente.

Tabla 4.15: Descripción Caso de uso “Almacenar imagen en servidor”.

4.3.1.6. Descripción CU-06

Caso de uso	Elimina imagen del servidor
Actores	Servidor de imágenes.
Resumen	El servidor recibe la indicación de “eliminar imagen” y la elimina de forma permanente.
Precondiciones	El servidor debe de recibir la instrucción de eliminar.
Postcondiciones	El servidor elimina la imagen de forma permanente.
Incluye	--
Extiende	--
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
2.- El servidor “recibe” la instrucción y elimina de forma permanente la imagen.	1.- El sistema envía la instrucción de eliminar la imagen.

Tabla 4.16: Descripción Caso de uso “Elimina imagen del servidor”.

4.3.1.7. Descripción CU-07

Caso de uso	Despliegue de resultados de detección
Actores	
Resumen	Una vez que el usuario da la opción de reproducir, se muestran los resultados de la detección en una tabla de resultados, mostrando el nombre de la persona, la fecha y hora del video.
Precondiciones	Previamente, se debió de reproducir el video.
Postcondiciones	Se despliega la información en una tabla de resultados.
Incluye	--
Extiende	--
Hereda de	--
Flujo de Eventos	
Actor	**Sistema**
	1.- El sistema despliega los resultados de la detección en una tabla de resultados debajo de los videos.

Tabla 4.17: Descripción Caso de uso "Despliegue de resultados de detección".

4.3.2. Diagrama de actividades

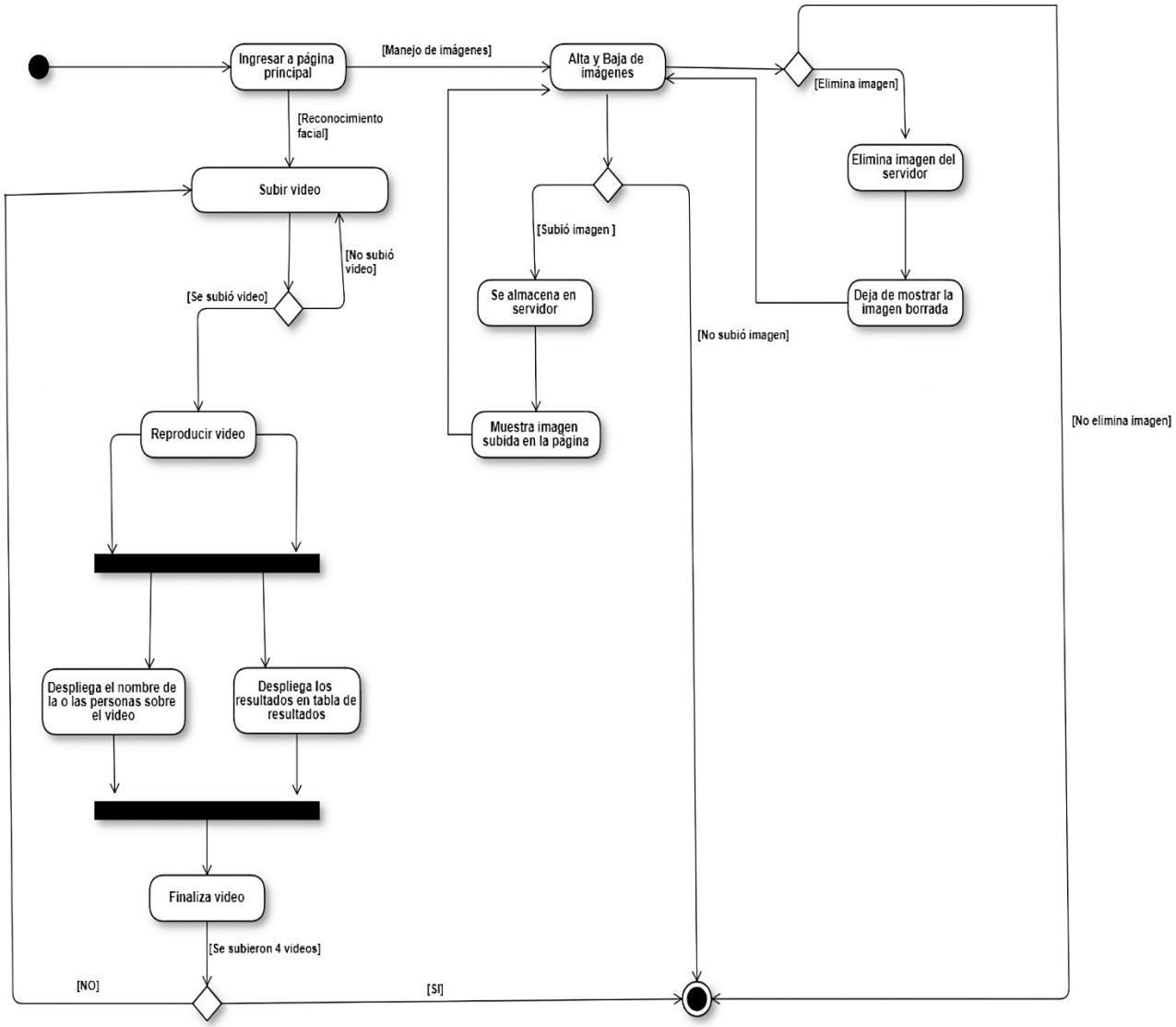

Figura 4.3: Diagrama de Actividades Prototipo de videovigilancia con Reconocimiento Facial.

4.3.3. Diagramas de secuencia

4.3.3.1. Diagrama de secuencia (Alta y baja de imágenes)

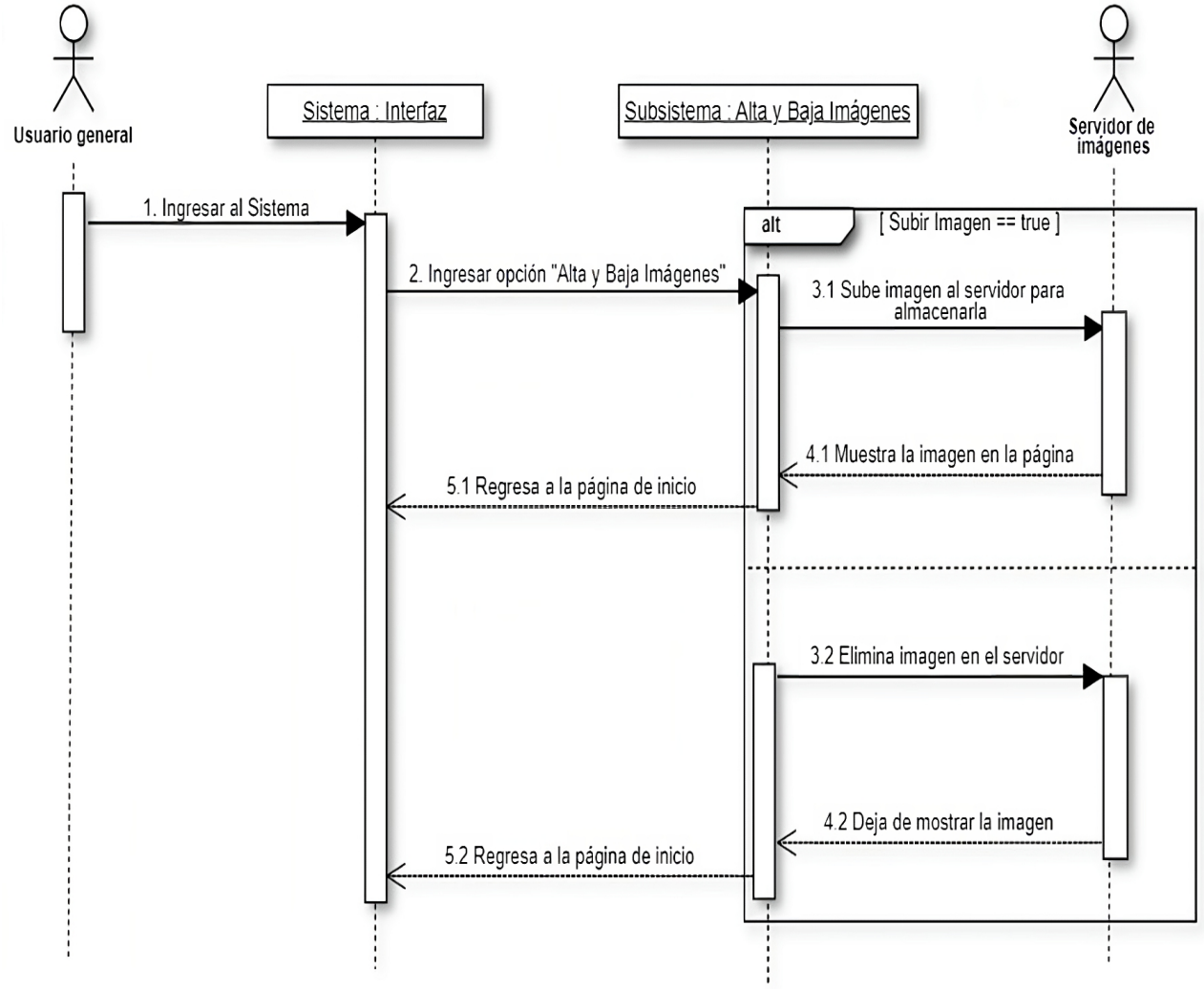

Figura 4.4: Diagrama de secuencia (Alta y Baja de imágenes).

4.3.3.2. Diagrama de secuencia (Reconocimiento facial)

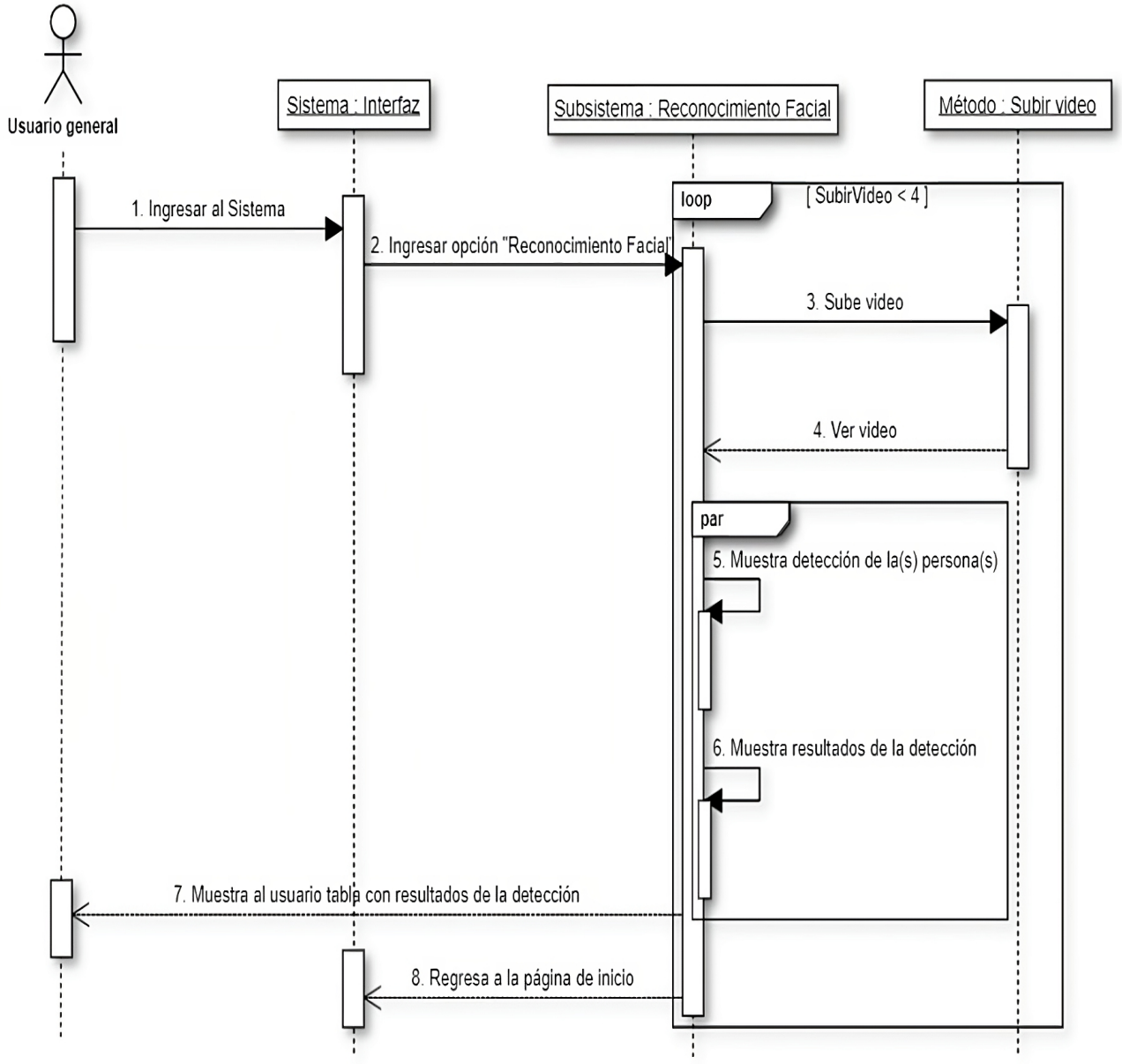

Figura 4.5: Diagrama de secuencia (Reconocimiento facial).

4.3.4. Diagramas de comunicación/colaboración

4.3.4.1. Diagrama de comunicación (Alta y Baja Imágenes)

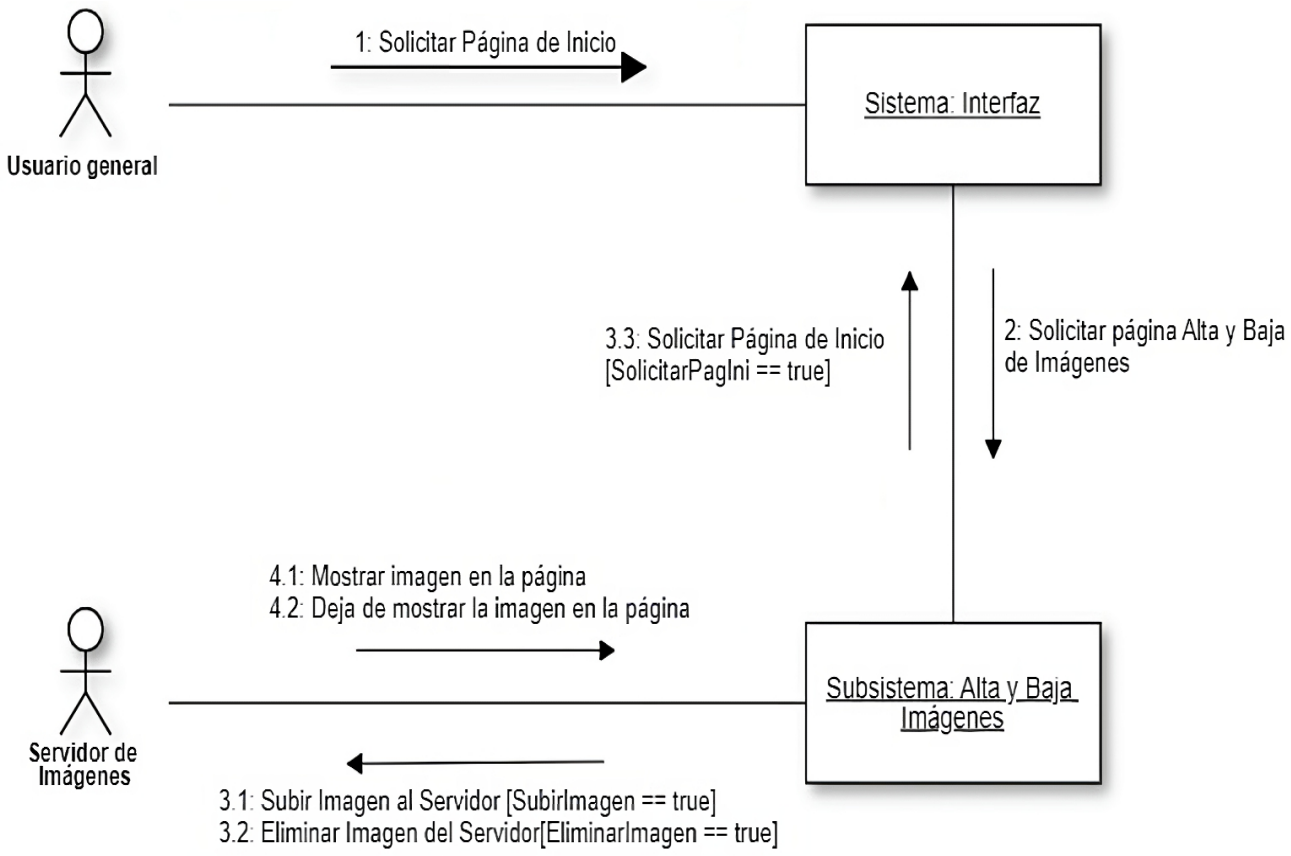

Figura 4.6: Diagrama de comunicación (Alta y Baja Imágenes).

4.3.4.2. Diagrama de comunicación (Reconocimiento Facial)

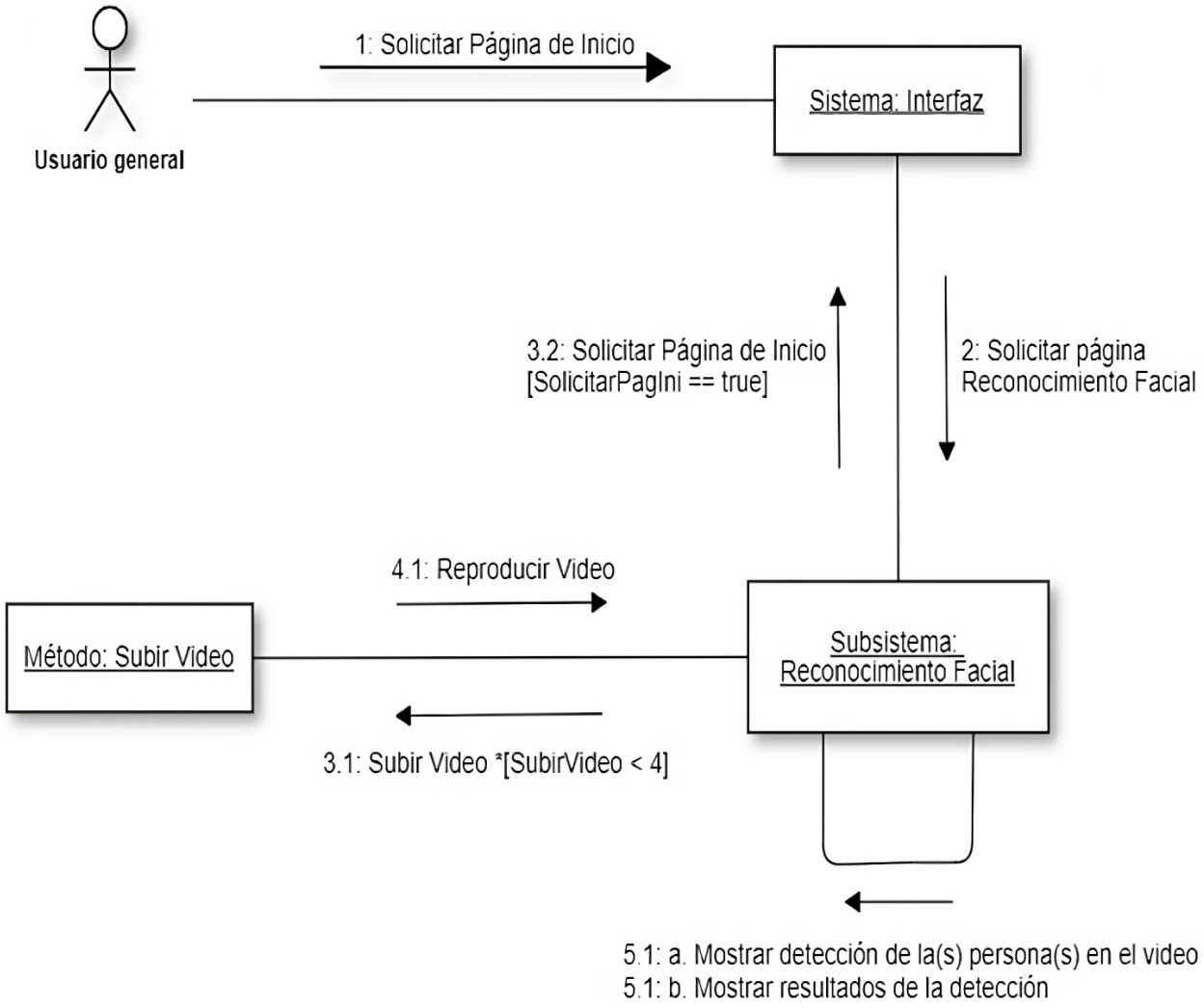

Figura 4.7: Diagrama de comunicación (Reconocimiento Facial).

4.3.5. Diagramas de estados

4.3.5.1. Diagrama de estados (Alta y Baja Imágenes)

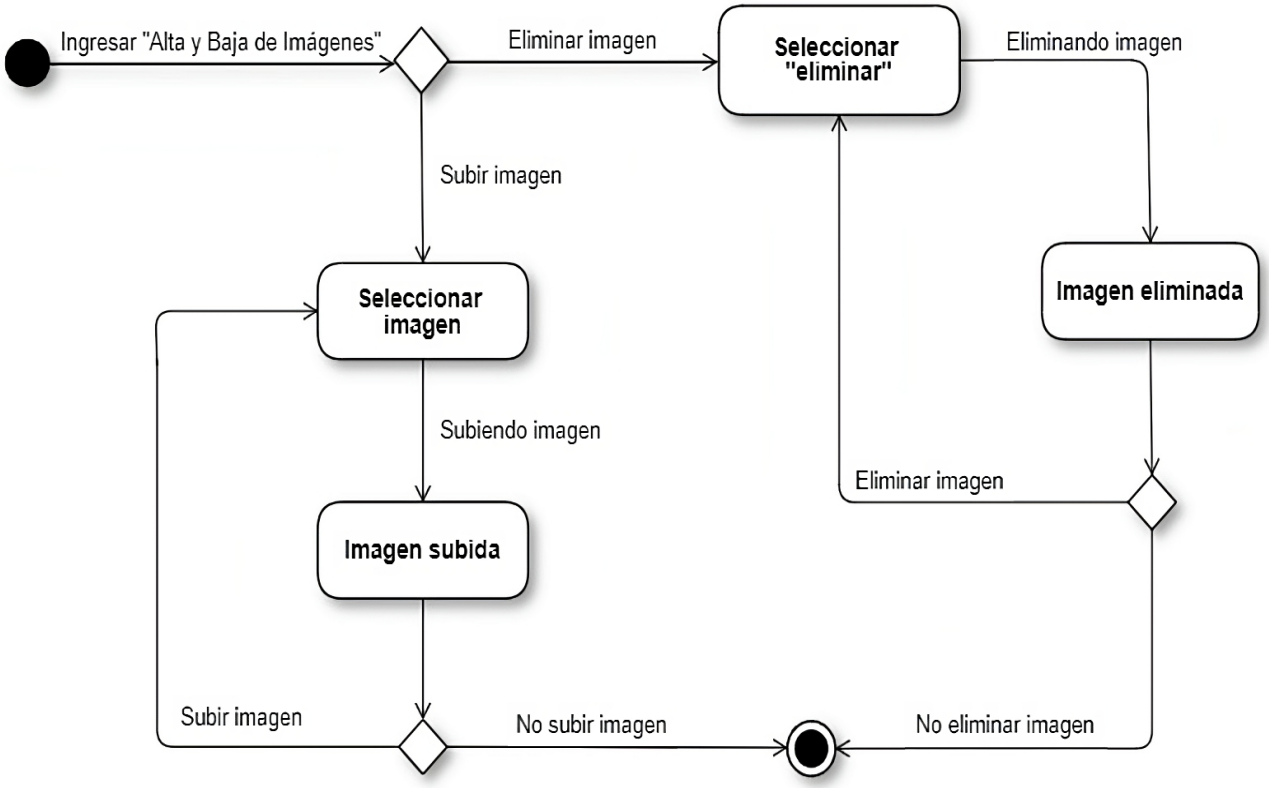

Figura 4.8: Diagrama de estados (Alta y Baja de Imágenes).

4.3.5.2. Diagrama de estados (Reconocimiento Facial)

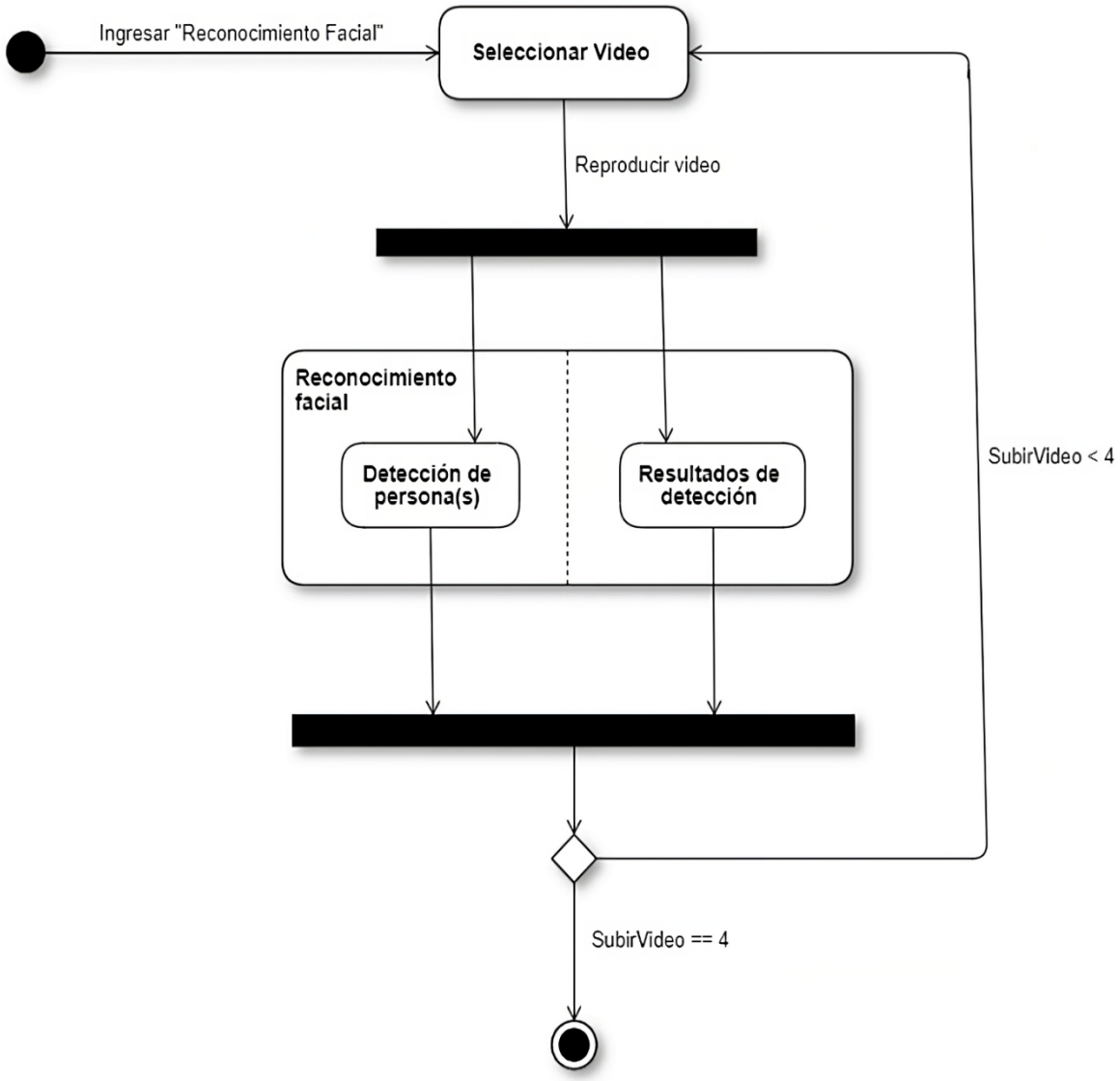

Figura 4.9: Diagrama de estados (Reconocimiento Facial).

4.3.6. Diagrama de paquetes

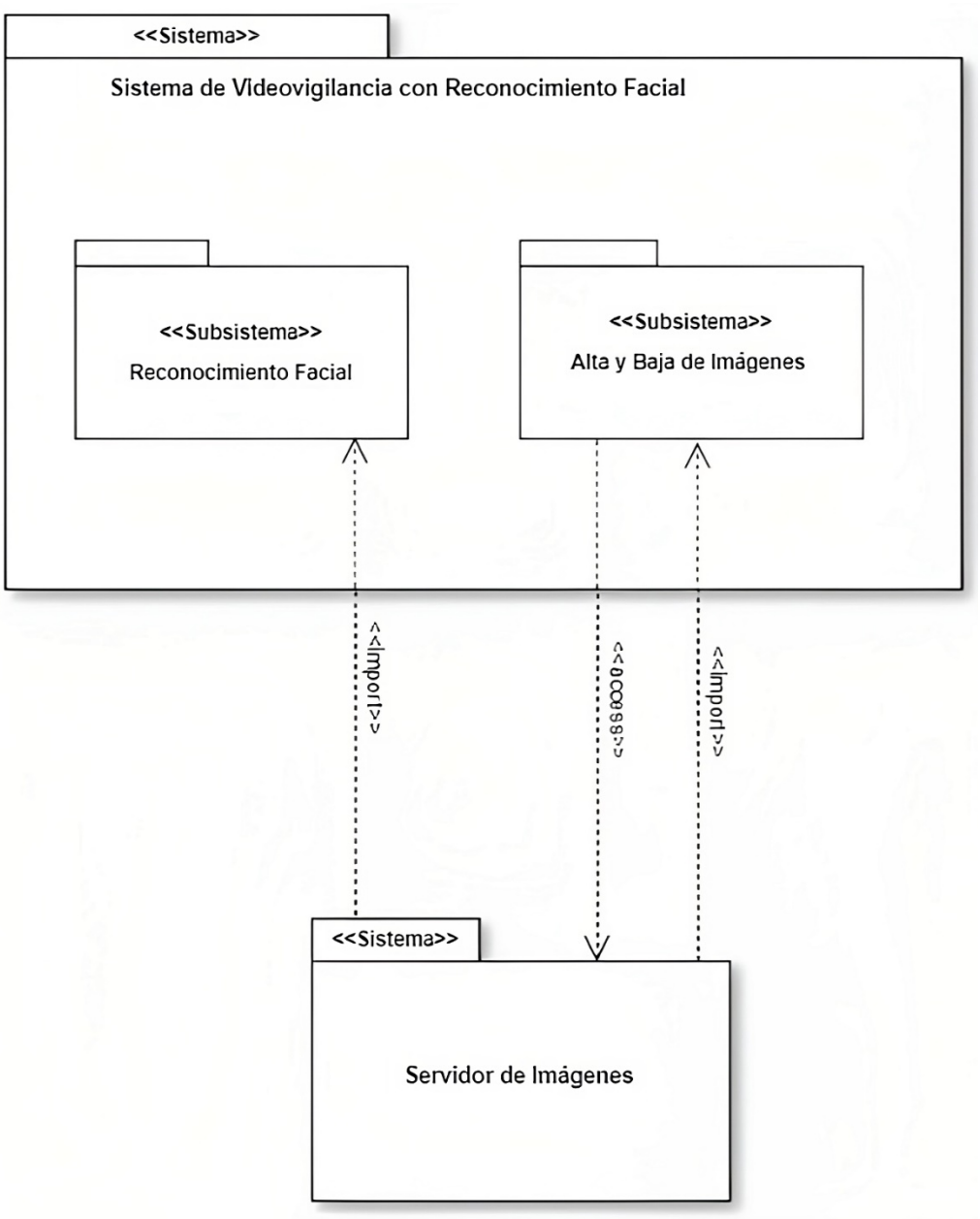

Figura 4.10: Diagrama de Paquetes Prototipo de Videovigilancia con Reconocimiento Facial.

4.4. Construcción del prototipo

4.4.1. Construcción y resultados “Página de inicio”

Como se mencionó en la **Sección 2.6**, se hará uso de Angular para el diseño del prototipo, y como la mayoría de las aplicaciones Web se cuenta con un archivo HTML principal el cual es “index.html”, en el cual se hará uso de un script donde se alojara Face-api, al igual de incorporar un icono y el título del sistema principal, por último el cuerpo del HTML se añadió una etiqueta propia de Angular que nos permitirá la ruta de los diferentes módulos del prototipo el cual es *“app-root”*, por lo que el código queda de la siguiente manera :

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Sistema Videovigilancia con Reconocimiento Facial</title>
        <base href="/">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <script src="./assets/js/face-api.min.js" defer></script>
        <link rel="icon" type="image/x-icon" href="assets/camera.png">
    </head>
    <body>
        <app-root></app-root>
    </body>
</html>
```

Tabla 4.18: Código HTML para incorporar la librería de Face-api, además de poner un título al encabezado y un icono.

Con el código anterior lo que se obtiene es un título para cada uno de los módulos del prototipo, además de incorporar una imagen al encabezado:

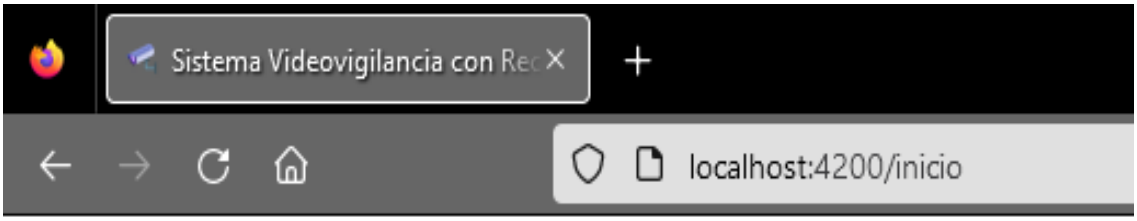

Figura 4.11: Resultado del archivo index.html.

La etiqueta *"app-root"* se utilizó para que se pudiera dar diferentes rutas a los módulos que se requerían para el prototipo, los cuales son:

- http://localhost:4200/inicio
- http://localhost:4200/upload-images
- http://localhost:4200/reconocimiento-facial

Para la página de inicio del prototipo se utilizó un documento HTML y CSS para el diseño de la página y dar enlace a los módulos de "Alta y Baja de Imágenes" y "Reconocimiento Facial", el código queda de la siguiente manera:

```
<div id="contenido">
  <div id="contenido_pagina">
    <header>
      <div class="cabecera_menu">
        <a href="http://localhost:4200/inicio" id="logo">
          <img src="/assets/camera.png">
        </a>
        <nav>
          <ul>
            <li>
              <a href="http://localhost:4200/upload-images">Alta y Baja de Imágenes</a>
            </li>
            <li>
              <a href="http://localhost:4200/reconocimiento-facial"> Reconocimiento Facial</a>
            </li>
          </ul>
        </nav>
      </div>
    </header>
     <hr>
     <h1 class="text-center">Sistema de Videovigilancia con Reconocimiento
```

```
    <hr>
    <h1 class="text-center">Sistema de Videovigilancia con Reconocimiento Facial</h1>
      <div>
        <img class="redimension" src="/assets/reconocimiento_facial.jpg">
      </div>
  </div>
</div>
```

Tabla 4.19: Código HTML para la construcción de la "Página de inicio" del prototipo.

```
#contenido{
    max-width: 1200px;
    margin: 0 auto;
}
#contenido_pagina{
    max-width: 1200px;
    margin: 0 auto;
    padding: 0 0 3% 0;
}
header{
    width: 100%;
    height: 100px;
    top: 0;
    left: 0;
}
header #logo img{
    margin: 20px;
    float: left;
    width: 50px;
}
nav ul li a{
    color: #6991ac;
}
```

Tabla 4.20: Código CSS para el diseño de la "Página de inicio" del prototipo.

Y como resultado de este código obtenemos el siguiente resultado para el diseño de la página de inicio:

Figura 4.12: Página de inicio Prototipo de videovigilancia.

4.4.2. Construcción y resultados "Servidor de imágenes"

Para resolver el problema de cómo obtener las imágenes base para la extracción de los descriptores faciales (como se explicó en el **Capítulo 3**), se hará uso de un servidor de imágenes, en donde se almacenarán todas las imágenes de las personas.

Para el desarrollo del servidor de imágenes se hará uso del *framework* NodeJS y de las bibliotecas Morgan, Express, Multer y Cors, para almacenar la información de las imágenes de las personas se hará uso del Gestor de Base de Datos MySQL, el siguiente código SQL es para mostrar cómo se creó dicha Base de Datos que solo contará con una tabla llamada *"files"* la cual tendrá 4 campos:

```
CREATE TABLE `files` (
  `id` int(11) NOT NULL,
  `nombre` varchar(255) NOT NULL,
  `imagen` varchar(255) NOT NULL,
  `fecha_creacion` timestamp NOT NULL DEFAULT current_timestamp() ON UPDATE
current_timestamp()
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_spanish_ci;
```

Tabla 4.21: Código SQL para la creación de la Base de Datos del prototipo.

Y como resultado obtenemos la siguiente Base de Datos:

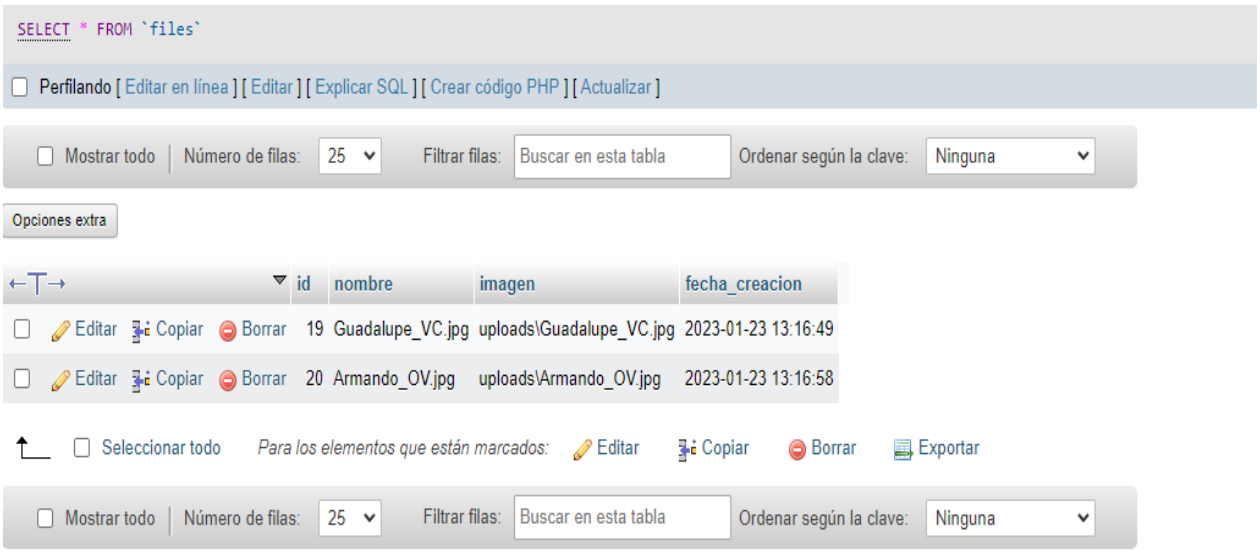

Figura 4.13: Base de Datos del Prototipo de Videovigilancia.

Para poder establecer una conexión con la Base de Datos desde el sistema usamos el siguiente código usando NodeJS:

```
const mysql = require('mysql')
const mysqlConnect = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'uploaddb'
})
mysqlConnect.connect(function(err){
    if(err){
        console.log(err)
    }else{
        console.log('La Base de Datos esta conectada!')
    }
})
module.exports = mysqlConnect
```

Tabla 4.22: Fragmento NodeJS para establecer conexión con la Base de Datos.

Una vez que el servidor se inicialice, hará una petición a la Base de Datos y en caso de que no se tenga ningún problema con la conexión dará un mensaje de que la base de datos está conectada, en caso de haber un error mandará un mensaje de error. Cuando la conexión es correcta se exporta esta conexión al módulo donde se almacenarán las imágenes para obtener los datos de la imagen y que sean guardadas en la Base de Datos.

Del lado del servidor se requiere importar las librerías mencionadas al inicio de este punto, y establecer el puerto de conexión del servidor donde se establecerá la conexión con el prototipo, usamos el siguiente código:

```
const express = require('express')
const cors = require('cors')
const mysqlConnect = require('./database')
const multer = require('multer')
const morgan = require('morgan')

app.listen(3000, () => {
    console.log('El servidor esta corriendo en el puerto 3000')

})
```

Tabla 4.23: Fragmento NodeJS para importar las librerías y establecer el puerto de conexión del servidor.

Una vez que se reciba una petición para almacenar una imagen en el server, el siguiente código lo que hará es ubicar en donde se almacenará la imagen, en este caso será dentro de una carpeta llamada *"uploads"* y obtendrá el nombre de la imagen con su extensión:

```
app.use('/uploads', express.static(path.join(__dirname, 'uploads')))

const storage = multer.diskStorage({

    destination: (req, file, callback) => {
        callback(null, 'uploads')
    },
    filename: (req, file, callback) => {
        callback(null, file.originalname)
    }
})
```

Tabla 4.24: Fragmento NodeJS para indicar el lugar donde se almacenarán las imágenes.

En la siguiente imagen se muestra que una vez aceptada la petición la imagen se almacena en la carpeta antes mencionada.

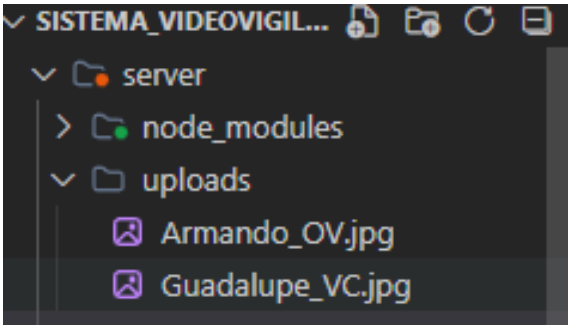

Figura 4.14: Almacenamiento de las imágenes subidas.

Una vez que se obtuvo la imagen y se obtuvieron los datos, se envía la información a la Base de Datos, haciendo un *"INSERT"*, pasando el nombre de las personas y la ubicación de la carpeta donde se encuentra la imagen almacenada, los datos del id y la fecha se pasan vacíos ya que esos datos los añade la propia tabla.

```
const upload = multer({storage})

app.post('/file', upload.single('file'),(req, res, next) => {

    const file = req.file

    const filesImg = {
        id: null,
        nombre: file.filename,
        imagen: file.path,
        fecha_creacion: null
    }

    res.send(file)

    mysqlConnect.query('INSERT INTO files set ?',[filesImg])
})
```

Tabla 4.25: Fragmento NodeJS para "insertar" los datos de la imagen en la Base de Datos.

Si el usuario desea eliminar una imagen, el servidor recibirá una petición de eliminación, el cual dicha petición vendrá con el identificador de la imagen y hará una consulta a la Base de Datos para encontrar el id y la ruta de almacenamiento de la imagen y una vez encontrada la eliminará de forma permanente, tanto del servidor como de la Base de Datos.

```
app.delete('/delete/:id', (req, res) => {

    const {id} = req.params
    mysqlConnect.query('DELETE FROM files WHERE id= ?', [id])
    res.json({message: 'La imagen ha sido eliminada'})
})
```

Tabla 4.26: Fragmento NodeJS para eliminar imagen del servidor y de la Base de Datos.

Tomando como base la Figura 4.14 y la Figura 4.13, inicialmente se tenían dos imágenes en el servidor, y usando el código anterior el servidor y la Base de datos eliminaron la imagen.

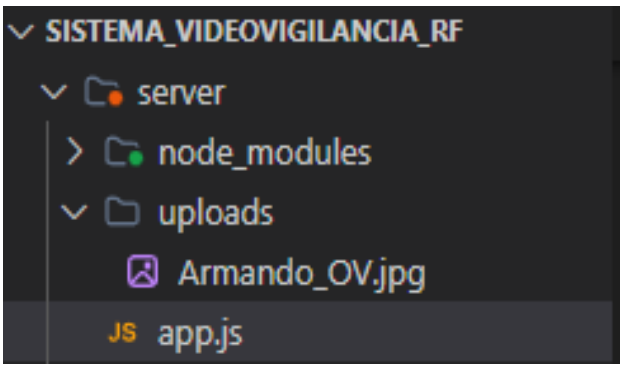

Figura 4.15: Imagen eliminada del server.

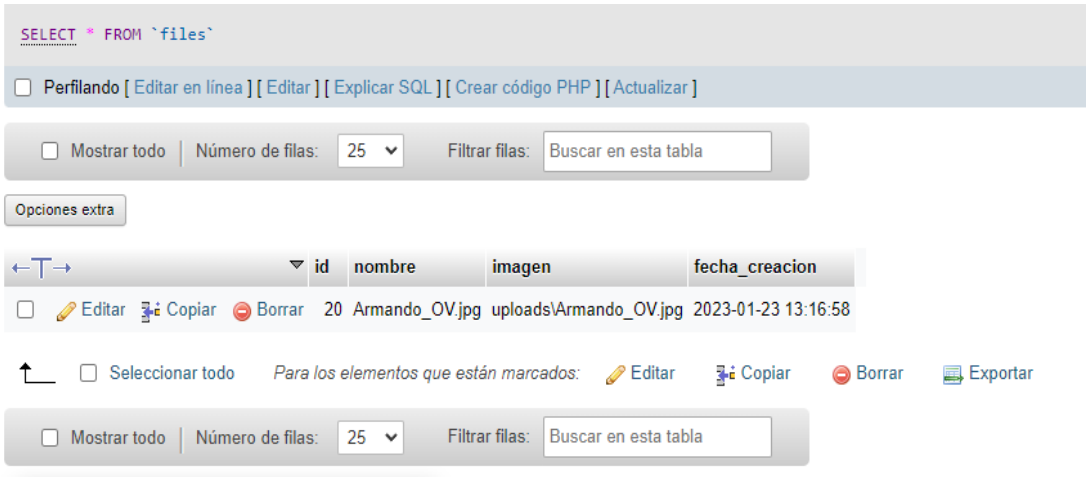

Figura 4.16: Datos eliminados de la Base de Datos.

Y como última tarea del servidor, es enviar la información de todas las imágenes que estén almacenadas en la Base de Datos, ya que al ingresar a la página de Alta y Baja de Imágenes se solicitará la información de las imágenes, los datos que se envían desde el servidor son: el id de la imagen, el nombre de la persona, y la ubicación de la imagen, el código queda de la siguiente manera:

```
app.get('/upload', (req, res) => {
    mysqlConnect.query('SELECT * FROM files', (err, rows, fields) => {
        if(!err){
            res.json(rows)
        }else{
            console.log(err)
        }
    })
})
```

Tabla 4.27: Fragmento NodeJS para enviar la imagen y los datos de la imagen a la "Página Alta y Baja de Imágenes".

4.4.3. Construcción y resultados “Página Alta y Baja de Imágenes”

Para el desarrollo de la vista de Alta y Baja de Imágenes se realizó usando código HTML, la página contará con un botón donde el usuario podrá seleccionar la imagen de la persona, una vez que la haya seleccionado, le aparecerá la imagen de lado derecho, y para subir la imagen al servidor, una vez que se suba la imagen al servidor la imagen aparecerá en la parte inferior de la página, el código queda de la siguiente manera:

```
<h1 class="text-center">Alta y Baja de Imágenes</h1>
<button class="boton-inicio" onclick="location.href='http://localhost:4200/inicio'">Página
Principal</button>
<div class="container">
  <div class="row">
    <div class="card-body">
      <form (ngSubmit)="onSubmit()">
        <div class="form-group">
          <label>Seleccione la imagen a subir</label><br>
          <input class="form-control-file" type="file" name="image" accept="image/*"
(change)="selectImage($event)"><br><br>
          <button class="btn btn-outline-primary" type="submit">Subir Imagen</button>
        </div>
      </form>
    </div>
    <div class="col-md-6">
      <div class="card">
        <img class="card-img-top" [src]="imgURL">
      </div>
    </div>
  </div>
</div>
<hr>
```

Tabla 4.28: código HTML para la construcción de la “Página Alta y Baja de Imágenes”.

Para la parte superior de la página donde se le permite al usuario el alta de imágenes, se obtiene el siguiente resultado:

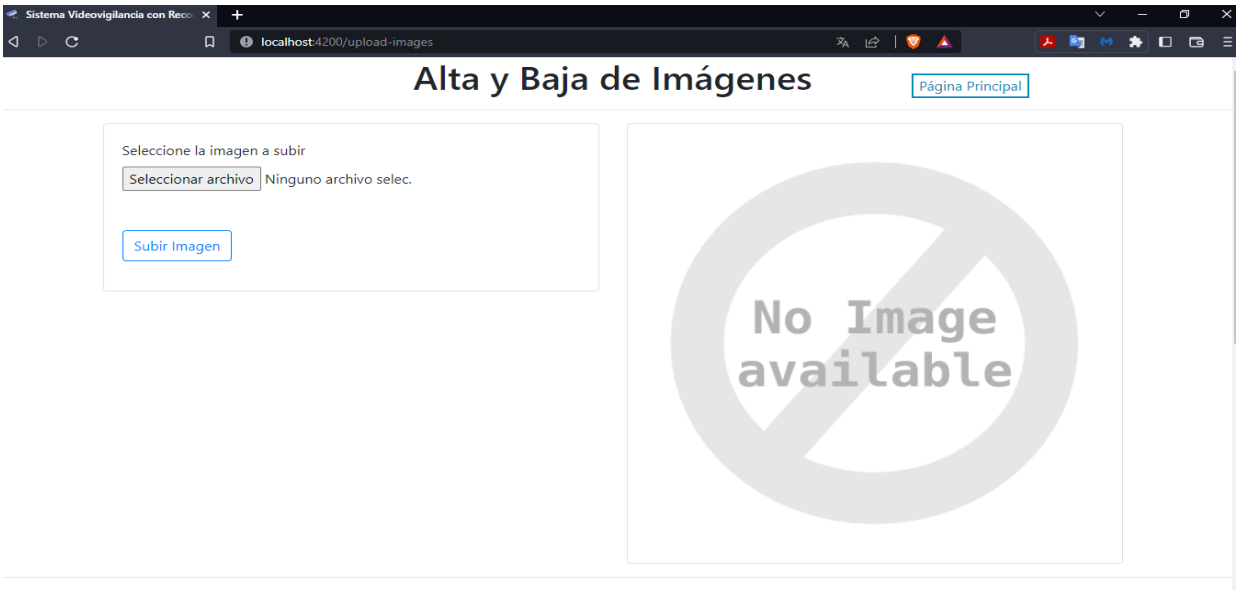

Figura 4.17: Vista del alta de imágenes de la página Alta y Baja de Imágenes.

```
<h2 class="text-center">Imágenes Subidas al Servidor</h2>
<div class="container">
  <div class="row">
    <div class="col-md-4" *ngFor="let image of this.images">
      <div class="card-header bg-primary text-white d-flex justify-content-
between align-items-center"> {{image.id}}.{{image.nombre}}
      </div>
    </div>
    <div class="card-body">
      <img class="card-img-top" src="http://localhost:3000/{{image.imagen}}">
      <button (click)="deleteImg(image.id)" class="btn btn-outline-success btn-
block" id="image.id">Eliminar</button>
    </div>
</div>
</div>
```

Tabla 4.29: Fragmento HTML para mostrar las imágenes alojadas en el servidor.

El código anterior tiene la funcionalidad de recibir la información del servidor y mostrarla en la página, muestra el id de la imagen, junto con el nombre de la persona y la imagen de la persona, al igual que muestra un botón el cual servirá para poder eliminar la imagen del servidor, si así se requiere, y se tiene el siguiente resultado:

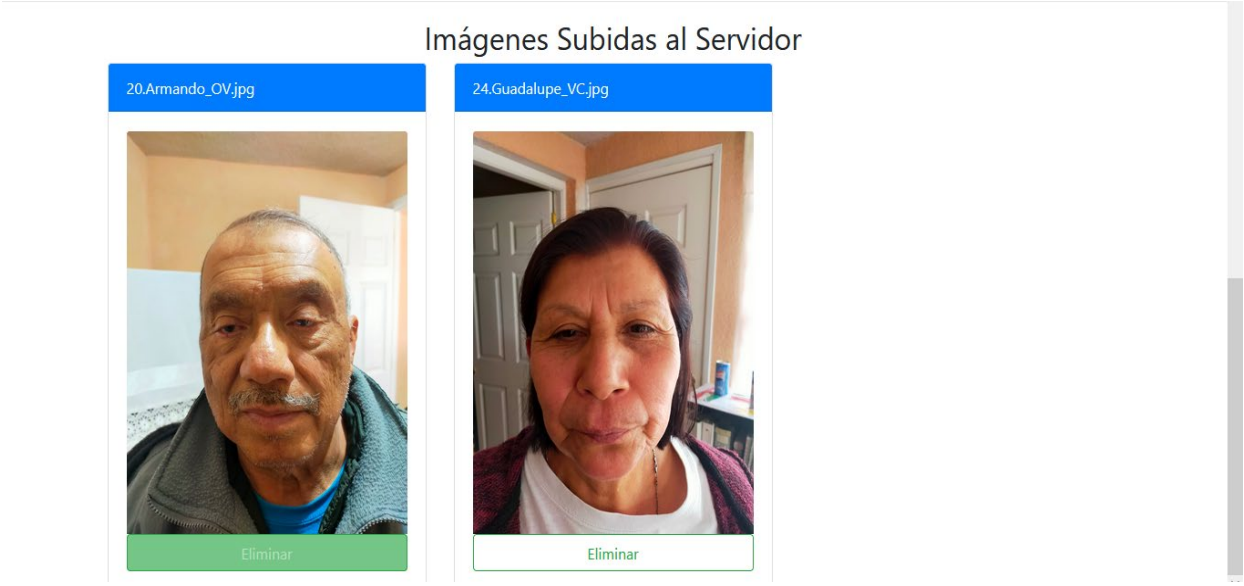

Figura 4.18: Vista de las imágenes almacenadas en el servidor.

Para llevar a cabo la funcionalidad de subir imagen, la cual contará con un botón para poder seleccionar alguna imagen almacenada en el equipo, otro botón de “Subir imagen”, en caso de que el usuario no seleccione una imagen, el botón mandará un mensaje donde le indicará que debe seleccionar alguna imagen, primero veremos cómo queda el código para seleccionar la imagen, ya que una vez que se cargue, se obtendrá el nombre de la imagen, que será el nombre de la persona, para enviar ese dato a la Base de Datos, el código es el siguiente:

```
selectImage(event:any){
  if(event.target.files.length > 0){
    const file = event.target.files[0]
    const reader = new FileReader()
    reader.readAsDataURL(file)
    reader.onload = (event: any) => {

      this.imgURL = event.target.result
    }
    this.image = file
  }
}
```

Tabla 4.30: Fragmento NodeJS para obtener los datos de la imagen que se seleccionó.

Una vez que la imagen esté cargada y lista para enviar, se obtiene la siguiente vista.

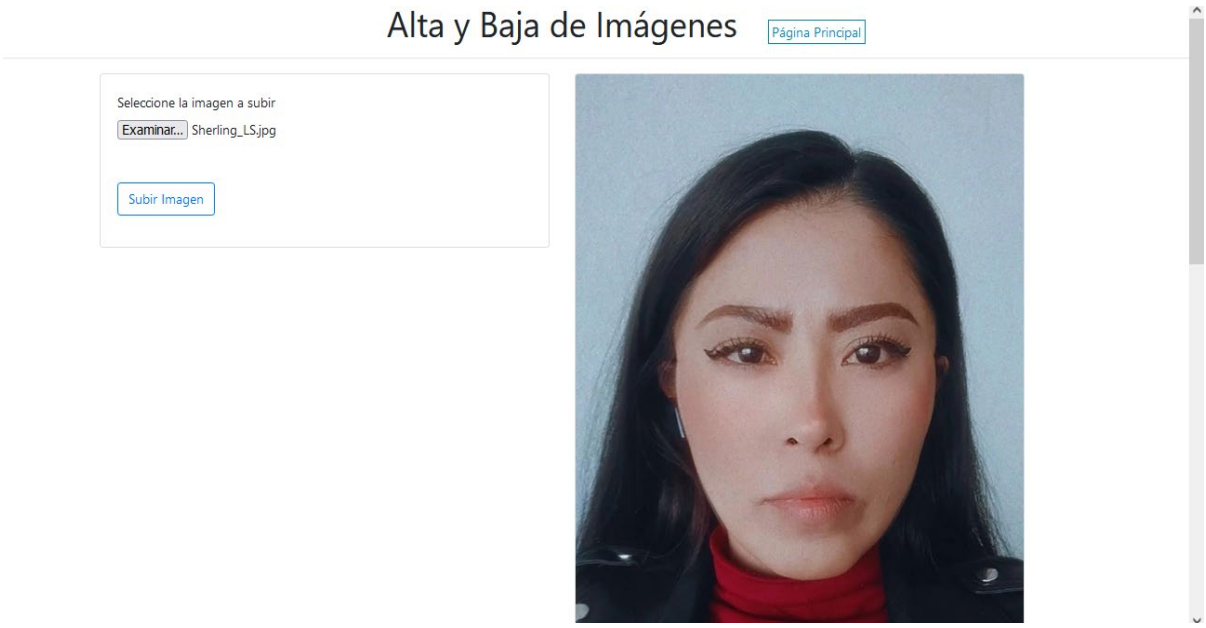

Figura 4.19: Vista de imagen lista para subir al servidor.

Una vez que la imagen esté cargada en la página, se debe dar clic en el botón de enviar imagen, una vez subida la imagen aparecerá un mensaje de que se subió la imagen, y si no se selecciona alguna imagen y se da clic en el botón, aparecerá un mensaje donde se indica que se debe seleccionar una imagen, el código es el siguiente:

```
export class UploadImagesComponent implements OnInit {
  onSubmit(){
    const formData = new FormData()
    formData.append('file', this.image)
    this.http.post<any>('http://localhost:3000/file', formData).subscribe({
      next: res => (Swal.fire({

          icon: 'success',
          title: 'Imagen enviada correctamente',
          text: 'La imagen se envió correctamente'

          }).then((result) => {
            if(result){
              location.reload()
            }
        })
      ),

      error: err => (Swal.fire({

          icon: 'error',
          title: '¡Error!',
          text: '¡Debes seleccionar una imagen!'

            })
      )
    })
  }
}
```

Tabla 4.31: Fragmento NodeJS que indica que se subió la imagen al servidor, en su defecto señala que se debe seleccionar una imagen.

Obtenemos las siguientes vistas como resultado de subir una imagen al servidor.

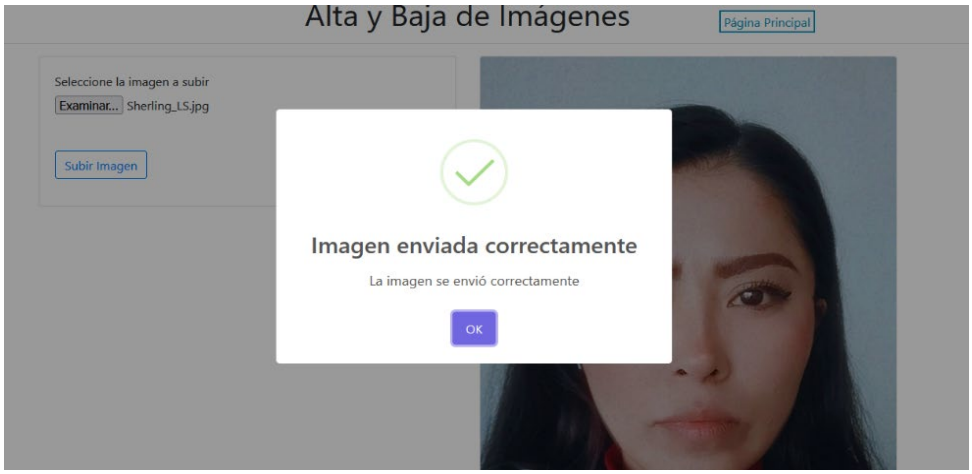

Figura 4.20: Vista de mensaje de imagen subida al servidor.

Figura 4.21: Vista no selecciono una imagen.

Para la funcionalidad de eliminar una imagen del servidor, se tiene un botón debajo de cada imagen el cual dice “eliminar”, al dar clic aparecerá un mensaje de aviso para saber si está seguro de eliminar la imagen, en caso de confirmar la eliminación la

imagen se borrará de la Base de Datos y del servidor, además se dejará de mostrar en la página, el código es el siguiente:

```
deleteImg(id:any){
  Swal.fire({

      icon: 'info',
      title: '¿Está seguro de eliminar la imagen del servidor?',
      showCancelButton: true,
      confirmButtonText: 'Eliminar',

  }).then((result) => {
    if(result.isConfirmed){
      this.http.delete<any>(`http://localhost:3000/delete/${id}`).subscribe({next:
 res => {
         location.reload()
     }})
    }
 })
}
```

Tabla 4.32: Fragmento NodeJS para indicar que se va eliminar una imagen del servidor y Base de Datos.

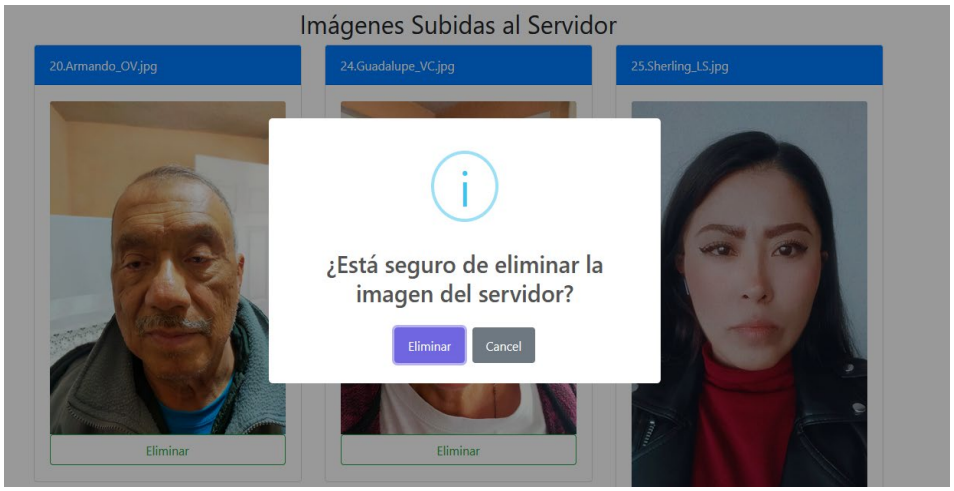

Figura 4.22: Vista de mensaje de confirmación para eliminar imagen.

4.4.4. Construcción y resultados “Página Reconocimiento Facial”

Para el desarrollo de la vista de Reconocimiento Facial se realizó usando código HTML y CSS, para la funcionalidad se usó JavaScript, ya que es el lenguaje en el que la API es funcional, en esta página se podrá seleccionar 4 videos para la detección de las personas que aparezcan en el video, y debajo de los espacios donde se encuentran los videos, habrá 4 tablas de resultados, los cuales indicarán el nombre de la persona detectada (en caso de que una persona no esté en la Base de Datos, se pondrá como *“unklown”* o desconocido), la fecha y hora de creación del video, el código de la vista es la siguiente:

```
<h1 class="text-center">Reconocimiento Facial</h1>
<button class="boton-inicio"
onclick="location.href='http://localhost:4200/inicio'">Página Principal</button>
<hr>

<h1 class="titulo_1">Cámara 1</h1>
<canvas class="canvas_1" id="video_1" width="500" height="300"></canvas>
<video class="video_1" id="videoInput" width="500" height="300" muted
controls></video>
<input type="file" class="input1" id="input_1" name="file" accept="video/*">

<h1 class="titulo">Resultados de la Detección </h1>

<h1 class="titulo_tabla1">Tabla Cámara 1</h1>
<table border="2" class="table_video1">
   <thead class="cabezera_tabla1">
       <tr>
           <th>Nombre de la Persona</th>
           <th>Detectado</th>
           <th>Fecha del Video</th>
           <th>Hora del Video</th>
       </tr>
   </thead>
   <tbody class="body_tabla1" id="tabla_video1">
       <tr>
           <td align="center">-</td>
           <td align="center">-</td>
           <td align="center">-</td>
           <td align="center">-</td>
       </tr>
   </tbody>
</table>
```

Tabla 4.33: Código HTML para la construcción de la “Página Reconocimiento Facial”.

La vista de la página de Reconocimiento Facial queda de la siguiente manera, primero como se ve la sección de cargar videos, y la sección de las tablas de resultados:

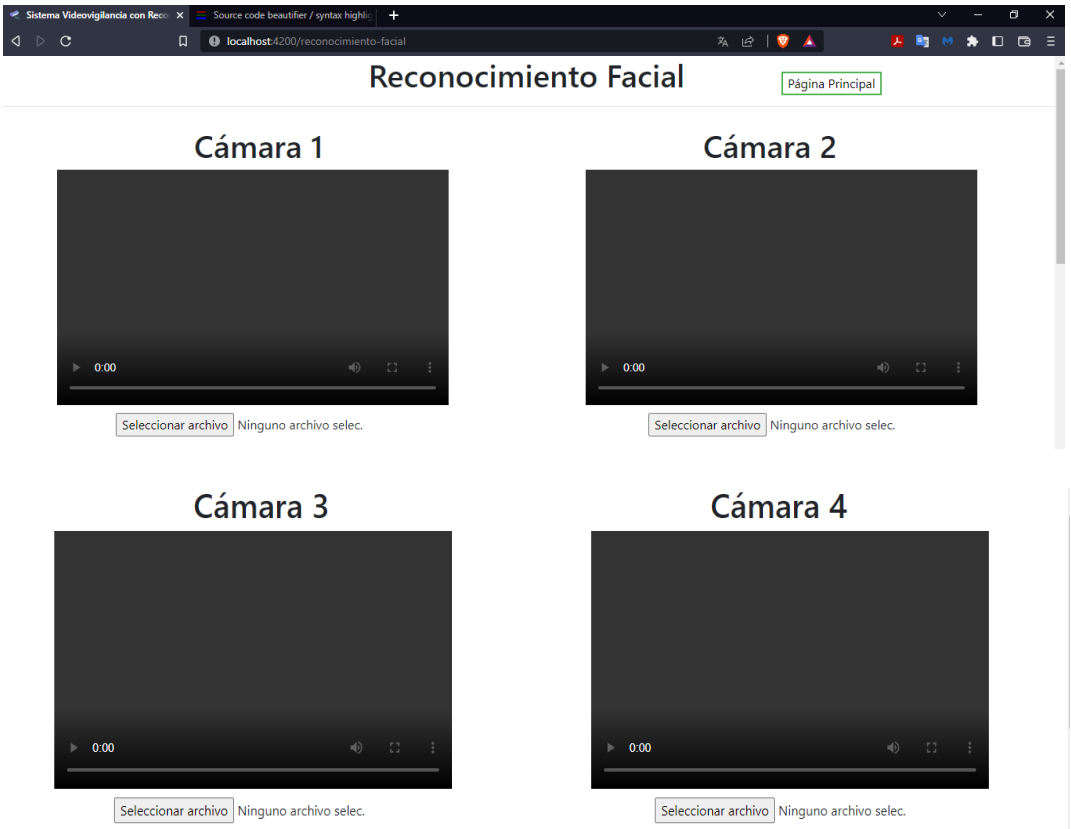

Figura 4.23: Vista de la sección de videos de Reconocimiento Facial.

Figura 4.24: Vista de la sección de tablas de resultados de la detección.

Para la página de reconocimiento facial se tiene como principales funcionalidades el que el usuario pueda subir videos de cualquier formato (mp4, mkv, etc.), una vez subidos al dar clic en el botón de "*play* o reproducir", el prototipo estará detectando los rostros que aparecen en el video y mostrando el nombre de la persona, al mismo tiempo que está haciendo las detecciones, estará imprimiendo los nombres de las personas que estén en ese momento en el video, así como la fecha y hora del video.

Como primer paso se debe indicar al prototipo que se hará uso de videos y se deben cargar las bibliotecas de Face-api, se deben cargar las bibliotecas para que la página funcione y para ello se debe indicar dónde se encuentra la API y así, el código es el siguiente:

```
const fileInput = document.getElementById('input_1')
const fileInput2 = document.getElementById('input_2')
const fileInput3 = document.getElementById('input_3')
const fileInput4 = document.getElementById('input_4')

const video = document.getElementById('videoInput')
const video_2 = document.getElementById('videoInput2')
const video_3 = document.getElementById('videoInput3')
const video_4 = document.getElementById('videoInput4')

Promise.all([
    faceapi.nets.faceRecognitionNet.loadFromUri('./assets/models'),
    faceapi.nets.faceLandmark68Net.loadFromUri('./assets/models'),
    faceapi.nets.ssdMobilenetv1.loadFromUri('./assets/models')
]).then(start)
```

Tabla 4.34: Fragmento JavaScript para hacer uso de las funciones de Face-api e incorporar el reproductor de video junto con el botón para seleccionar el video.

El código indica que se hará uso de un video y un botón que permite seleccionar un video almacenado en el equipo y cargar las bibliotecas para la detección del cuadro delimitador, el uso de los puntos de referencia y los descriptores faciales.

Una vez que el video ha sido seleccionado, se tomará la información de la hora y fecha de cuando fue creado, para esto en Windows al revisar los detalles del video muestra 3 datos los cuales son:

- Creado: muestra la fecha de cuando el archivo fue creado, sin embargo, esta fecha puede cambiar, en el caso de que se cree una copia del video y mostrará la fecha y hora de cuando fue copiado.
- Modificado: muestra la fecha de creación, con la diferencia que este no cambia a pesar de que el video sea copiado en diferentes ocasiones, este es el dato que se obtendrá y posteriormente imprimir en la tabla de resultados.
- Último acceso: muestra la fecha y hora de la última vez que se usó.

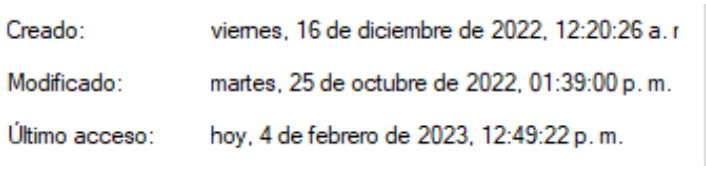

Figura 4.25: Detalles de las fechas de un video en Windows.

El siguiente paso es el permitir al usuario subir un video, como se mostró antes se cuenta con un botón el cual permitirá seleccionar un video almacenado y una vez que se cargue el video en la página se tomará la hora y fecha del video para su uso más adelante, el código queda de la siguiente manera:

```
function start(){

        fileInput.addEventListener('change', function(e) {
        video.src = URL.createObjectURL(this.files[0])
    })

    filepicker1.addEventListener('change', (event) => {
        const files1 = event.target.files
        output1.textContent = ''
        for(const file of files1){
            const date1 = new Date(file.lastModified)
            const fecha_video1 = output1.textContent += `(${date1})`
            fv1 = fecha_video1.slice(10, 25)
            hv1 = fecha_video1.slice(26, 34)
        }
    })
}
```

Tabla 4.35: Fragmento JavaScript para obtener la hora y fecha del video seleccionado.

Una vez que el video se ha cargado y obtenido la fecha y hora, se debe hacer una consulta al server para obtener las imágenes base y extraer los descriptores faciales y el nombre de las personas, este proceso se lleva a cabo como se mencionó en el **Capítulo 3** de manera teórica, ahora se verá a nivel de programación, el código queda de la siguiente manera:

```
function obtencionURL(){
    fetch("http://localhost:3000/upload")
        .then((res) => res.json())
        .then((data) => {
            recognitedFaces(data)
        })
}

function recognitedFaces(data){
  const labeles = data.map((user) => `${user.nombre}`)
  const nombres = p_e_1.split(",", p_e_1.length)

  Promise.all(
    nombres.map(async label => {
      const descriptions = []
      for(let i=1; i<=1;i++){
        const img = await
faceapi.fetchImage(`http://localhost:3000/uploads/${label}.jpg`)
        const detections = await
faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor()
        descriptions.push(detections.descriptor)
      }
      const descriptores_personas = new
faceapi.LabeledFaceDescriptors(label, descriptions)
    }
  )
}
```

Tabla 4.36: Fragmento JavaScript que realiza la petición al servidor de todas las imágenes y obtiene los descriptores faciales.

Del código anterior, la parte más importante es la siguiente:

```
const detections = await faceapi.detectSingleFace(img).withFaceLandmarks().withFaceDescriptor()
```

Tabla 4.37: Fragmento JavaScript que hace uso de Face-api para la obtención de los cuadros delimitadores, puntos de referencia y descriptores faciales.

Ya que aquí es donde se obtiene el cuadro delimitador de la imagen, se extraen los puntos de referencia y se extraen los descriptores faciales, estos últimos se guardan en un variable la cual se denota como “descriptores_personas”, dentro se encuentran los 128 datos del descriptor de cada persona que esté en el servidor.

Una vez obtenido los descriptores se le establecerán a una nueva variable, que además de tener los descriptores de todas las personas, además se establecerá el umbral de resultado para que se pueda clasificar la imagen.

```
const faceMatcher = new faceapi.FaceMatcher(labeledDescriptors, 0.6)
```

Tabla 4.38: Fragmento JavaScript que hace uso de Face-api para almacenar los descriptores faciales y establecer el umbral de resultado.

Una vez que se cargó el video en la página, se obtuvo la fecha y hora, se obtuvieron los descriptores faciales de todas las personas y se calculó el umbral de resultado, se mostrará la parte final, el reconocimiento facial con la impresión de los resultados.

Para hacer la detección del rostro se debe establecer un “*canvas*”, que es el espacio donde la API hará la detección, para este proyecto se establecerá sobre el alto y ancho del video, este “*canvas”* lo que nos permite es “dibujar” el cuadro delimitador sobre la persona detectada y a su vez imprime el nombre de la persona, así como el resultado del cálculo del umbral, también extrae los puntos de referencia y los descriptores de la persona que está detectando, esto con el fin de poder clasificar la imagen para proporcionar el nombre de la persona que se detecta, el código queda de la siguiente manera:

```
video.addEventListener('play', async() => {
  const canvas = document.getElementById('video_1')
  document.body.append(canvas)
  const displaySize = {
    width: video.width,
    height: video.height
  }
  faceapi.matchDimensions(canvas, displaySize)
  setInterval(async () => {
    const detections = await
faceapi.detectAllFaces(video).withFaceLandmarks().withFaceDescriptors()
    const resizedDetections = faceapi.resizeResults(detections, displaySize)
    canvas.getContext('2d').clearRect(0, 0, canvas.width, canvas.height)
  }
}
```

Tabla 4.39: Fragmento JavaScript para determinar el tamaño del canvas y obtener los cuadros delimitadores, puntos de referencia y descriptores faciales del video.

Una vez obtenido los descriptores de las personas que han sido detectadas en el video, se debe clasificar a la persona, la API lo hace mediante Distancia Euclidiana para poder clasificar a la persona y mostrar el nombre y la coincidencia, el código queda de la siguiente manera:

```
const results = resizedDetections.map((d) => {
    return faceMatcher.findBestMatch(d.descriptor)
})
```

Tabla 4.40: Fragmento de JavaScript para determinar la Distancia Euclidiana entre los descriptores base y del video.

Ya que se tiene la clasificación de la persona, lo que queda es mostrar el nombre y la coincidencia, el código queda de la siguiente manera:

```
results.forEach((result, i) => {
    const box = resizedDetections[i].detection.box
    const drawBox = new faceapi.draw.DrawBox(box, { label: result.toString() })
    drawBox.draw(document.getElementById('video_1'))
})
```

Tabla 4.41: Fragmento JavaScript para mostrar el nombre de la persona detectada dentro del cuadro delimitador.

Y lo que se obtiene como resultado es una impresión del cuadro delimitador junto con el nombre de la persona y la coincidencia.

Figura 4.26: Resultados de la detección en la cámara 1 a una distancia focal lejana.

Figura 4.27: Resultados de la detección en la cámara 2 con una cámara a una altura de 2.15 mts.

Figura 4.28: Resultados de la detección en la cámara 3 con tres personas en el mismo espacio.

Figura 4.29: Resultado de la detección en la cámara 4 con dos personas en el mismo espacio.

Figura 4.30: Resultado de la detección en la cámara 3 con una persona de perfil.

Por último, queda la impresión de los resultados de la detección (cabe mencionar que la impresión de los resultados se realiza a la par de la detección), una vez que la persona ha sido identificada se imprime el nombre junto con la fecha y hora, el código queda de la siguiente manera:

```
if(!pv1.includes(persons_dv1) && persons_dv1.toString().trim().length != 0){
    pv1[pv1.length] = persons_dv1
    var datos_v1 = document.getElementById('tabla_video1').insertRow()
    var cell1_v1 = datos_v1.insertCell(0)
    var cell2_v1 = datos_v1.insertCell(1)
    var cell3_v1 = datos_v1.insertCell(2)
    var cell4_v1 = datos_v1.insertCell(3)
    pv1.forEach((result, i) => {
        if((result.toString().trim().length != 0)){
            cell1_v1.innerHTML = persona_fecha_hora + result + cierre
            cell2_v1.innerHTML = imagen
            cell3_v1.innerHTML = persona_fecha_hora + fv1 + cierre
            cell4_v1.innerHTML = persona_fecha_hora + hv1 + cierre
        }
    })
}
```

Tabla 4.42: Fragmento JavaScript que obtiene los resultados de la detección y los imprime en la tabla de resultados.

La impresión de los datos de la detección en la tabla de resultados se ve de la siguiente manera:

Tabla Cámara 4

Nombre de la Persona	Detectado	Fecha del Video	Hora del Video
-	-	-	-
Guadalupe_VC	✓	Mon Oct 24 2022	13:11:09
Armando_OV	✓	Mon Oct 24 2022	13:11:09
unknown	✓	Mon Oct 24 2022	13:11:09

Figura 4.31: Tabla de resultados de la detección de la cámara 4.

4.5. Pruebas

En la etapa de pruebas se realizarán pruebas de usabilidad, esto quiere decir que se le harán pruebas al prototipo con usuarios que sean expertos en el manejo de equipos de cómputo y páginas Web, al igual que con usuarios inexpertos en el manejo de equipos de cómputo y páginas Web, dichos usuarios manipularán el prototipo y se analizaran los resultados que se obtengan de los casos de pruebas que a continuación se detallaran, estos casos de pruebas son tomados de los requerimientos funcionales.

4.5.1. Casos de pruebas de usabilidad

En la Tabla 4.46 se describe la relación de los requerimientos funcionales con los casos de pruebas que se realizaron al Prototipo de videovigilancia, los cuales son los siguientes:

Requerimiento funcional	Nombre	Prueba de usabilidad
RF-01	Archivos multimedia.	CPU-01
RF-02	Reproductor multimedia.	CPU-02
RF-03	Reconocimiento Facial.	CPU-02
RF-04	Tabla de resultados.	CPU-03
RF-05	Alta de imágenes.	CPU-04
RF06	Baja de imágenes.	CPU-05
RF-07	Página de inicio	CPU-06, CPU-07 y CPU-08

Tabla 4.43: Relación requerimientos funcionales con pruebas de usabilidad.

A continuación, se presenta la descripción de los casos de pruebas de usabilidad del prototipo.

Detalle CPU-01.

Tarea	Subir video al prototipo.
Número	1
Definición	El usuario debe subir un video en los espacios donde se encuentra dicha opción.
Objetivo	El usuario debe ser capaz de subir un video al prototipo.
Salida o resultado esperado	El video debe estar cargado en el reproductor multimedia.
Frecuencia	4
Duración esperada	2 minutos como máximo.
Longitud de ruta	3
Flexibilidad	Se puede considerar una longitud más extensa, dependiendo de donde este almacenado el video.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber diferenciar que video es el que desea subir.

Tabla 4.44: Detalle caso de prueba de usabilidad #1.

Detalle CPU-02.

Tarea	Uso del reproductor multimedia.
Número	2
Definición	El usuario debe reproducir un video que anteriormente haya subido.
Objetivo	El usuario debe reproducir el video en curso.
Salida o resultado esperado	El video debe estar reproduciéndose hasta el final.
Frecuencia	1
Duración esperada	5 segundos
Longitud de ruta	1
Flexibilidad	Ninguna.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora.

Tabla 4.45: Detalle caso de prueba de usabilidad #2.

Detalle CPU-03.

Tarea	Ubicación de la tabla de resultados de la detección.
Número	3
Definición	El usuario debe identificar la tabla de resultados correspondiente al video reproducido.
Objetivo	El usuario debe ser capaz de identificar la tabla de resultados que corresponde con el video que reprodujo anteriormente.
Salida o resultado esperado	El usuario encuentra los resultados de la detección.
Frecuencia	1
Duración esperada	40 segundos.
Longitud de ruta	1
Flexibilidad	Ninguna.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber leer.

Tabla 4.46: Detalle caso de prueba de usabilidad #3.

Detalle CPU-04

Tarea	Subir imagen al servidor.
Número	4
Definición	El usuario debe subir una imagen al servidor
Objetivo	El usuario debe ser capaz de subir una imagen al servidor.
Salida o resultado esperado	El prototipo muestra un mensaje de que la imagen fue subida correctamente y la imagen debe aparecer en la página.
Frecuencia	1
Duración esperada	2 minutos como máximo.
Longitud de ruta	3
Flexibilidad	Se puede considerar una longitud más extensa, dependiendo de donde este almacenado el video.

Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber diferenciar que imagen es la que desea subir.

Tabla 4.47: Detalle caso de prueba de usabilidad # 4.

Detalle CPU-05

Tarea	Eliminar imagen del servidor.
Número	5
Definición	El usuario debe eliminar una imagen del servidor
Objetivo	El usuario debe ser capaz de eliminar una imagen del servidor.
Salida o resultado esperado	El prototipo muestra un mensaje de que la imagen fue eliminada correctamente y la imagen ya no debe ser vista en la página.
Frecuencia	1
Duración esperada	10 segundos.
Longitud de ruta	2
Flexibilidad	Ninguna.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber leer.

Tabla 4.48: Detalle de prueba de usabilidad #5.

Detalle CPU-06

Tarea	Ingresar a la página “Alta y Baja de Imágenes”.
Número	6
Definición	El usuario debe acceder a la página de Alta y Baja de Imágenes.
Objetivo	El usuario debe ser capaz de ubicar la opción de Alta y Baja de imágenes e ingresar.

Salida o resultado esperado	Acceder a la página de Alta y Baja de Imágenes.
Frecuencia	1
Duración esperada	30 segundos.
Longitud de ruta	1
Flexibilidad	Ninguna.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber leer.

Tabla 4.49: Detalle de prueba de usabilidad #6

Detalle CPU-07

Tarea	Ingresar a la página “Reconocimiento Facial”.
Número	7
Definición	El usuario debe acceder a la página de Reconocimiento Facial.
Objetivo	El usuario debe ser capaz de ubicar la opción de Reconocimiento Facial e ingresar.
Salida o resultado esperado	Acceder a la página de Reconocimiento Facial.
Frecuencia	1
Duración esperada	30 segundos.
Longitud de ruta	1
Flexibilidad	Ninguna.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber leer.

Tabla 4.50: Detalle de prueba de usabilidad #7.

Detalle CPU-08

Tarea	Ingresar a la página de Inicio.
Número	8
Definición	El usuario debe regresar a la página de inicio estando, ya sea en “Alta y Baja de Imágenes” o en “Reconocimiento Facial”.
Objetivo	El usuario debe ser capaz de ubicar la opción de Página de Inicio.
Salida o resultado esperado	Regresar a la página de inicio del prototipo.
Frecuencia	1
Duración esperada	20 segundos.
Longitud de ruta	1
Flexibilidad	Ninguna.
Requerimientos físicos y mentales	El usuario debe saber usar el mouse de la computadora. El usuario debe saber leer.

Tabla 4.51: Detalle caso de prueba de usabilidad #8.

4.5.2. Resultados de las pruebas de usabilidad.

A continuación, se muestran los resultados que se obtuvieron de los casos de pruebas de usabilidad anteriores, los cuales se les realizaron a diferentes usuarios, un total de 8 personas que aceptaron probar el prototipo y de los cuales se escogieron dependiendo su nivel en manejo de equipos de cómputo y manejo de páginas Web, los cuales están divididos de la siguiente manera:

- Nivel experto: Usuarios que tienen conocimientos y experiencia en el manejo de equipos de cómputo y páginas Web, los cuales son el usuario 2, usuario 3 y usuario 8.
- Nivel intermedio: Usuarios que tienen conocimientos y experiencia mediana en el manejo de equipos de cómputo y páginas Web, los cuales son el usuario 1, usuario 4 y usuario 5

- Nivel bajo: Usuarios que tiene pocos conocimientos y experiencia en equipos en el manejo de equipos de cómputo y páginas web, los cuales son el usuario 6 y usuario 7.

Y tomando como resultado final éxito o fracaso para cada una de las pruebas, se considera:

- Éxito: El resultado cumple con el objetivo del caso de prueba
- Fracaso: El resultado no cumple con el objetivo del caso de prueba.

Caso de prueba	Usuario 1	Usuario 2	Usuario 3	Usuario 4	Usuario 5	Usuario 6	Usuario 7	Usuario 8
CPU-01	Éxito	Éxito	Éxito	Fracaso	Fracaso	Fracaso	Fracaso	Éxito
CPU-02	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
CPU-03	Éxito	Éxito	Éxito	Éxito	Éxito	Fracaso	Fracaso	Éxito
CPU-04	Éxito	Éxito	Éxito	Éxito	Éxito	Fracaso	Fracaso	Éxito
CPU-05	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
CPU-06	Éxito	Éxito	Éxito	Éxito	Éxito	Fracaso	Éxito	Éxito
CPU-07	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito
CPU-08	Éxito	Éxito	Éxito	Éxito	Éxito	Éxito	Fracaso	Éxito

Tabla 4.52: Resultados de los casos de pruebas de cada usuario.

En la tabla anterior se detallan los resultados que se obtuvieron durante la ejecución de las pruebas de usabilidad, para los usuarios expertos se obtuvo una tasa de éxito del 100%, para los usuarios intermedios se obtuvo una tasa de éxito del 78.57% donde la ubicación del archivo fue la causa del fracaso y para los usuarios inexpertos se obtuvo una tasa de éxito del 50% donde la ubicación del archivo, tanto para subir los videos e imágenes, a su vez fue la causa del fracaso.

Capítulo 5
Conclusiones y trabajo a futuro

5.1. Conclusiones

El objetivo de este trabajo fue el desarrollar un prototipo de videovigilancia con reconocimiento facial vía Web, que permitiera subir y reproducir videos que estuvieran almacenados en el ordenador o en algún dispositivo de almacenamiento externo con el fin de identificar a las personas que aparezcan en dichos videos, de saber si en un espacio físico estuvo alguna persona ajena al lugar y generar una tabla en la interface grafica la cual muestre a las personas que se detectaron como resultado del análisis del video.

Para llevar a cabo esto se utilizó la API Face-api, la cual hace uso de redes neuronales profundas de convolución, las cuales están entrenadas para la detección de personas. De tal forma que con solo una foto base se pudiese realizar la detección e identificación de la persona en cuestión, Face-api brindó una solución funcional para el desarrollo de esta tesis, ya que fue lo suficientemente moldeable para la construcción del prototipo de videovigilancia con reconocimiento facial vía Web.

Con el desarrollo de esta tesis se obtuvieron los siguientes resultados:

- Es posible realizar la identificación de personas en entornos abiertos, con el uso de una cámara Web convencional y un equipo de cómputo con características reducidas.

- Es posible la realización de un prototipo de videovigilancia que permita a los usuarios hacer uso del reconocimiento facial y detección de personas, donde se permita subir videos almacenados en su ordenador o dispositivos de almacenamiento externos y mostrar los resultados de dicha detección.
- Es posible hacer uso de APIs *open source,* ya que se pueden moldear y adaptar a los proyectos de sistemas de videovigilancia web.
- Se pudo entregar una versión beta funcional utilizando *frameworks* de última generación para la construcción de un prototipo web.
- Se verifico la usabilidad del prototipo mediante la ejecución de pruebas de usabilidad

Con lo antes mencionado, se pudo dar una solución que cubre el problema inicial y dar un margen de referencia a futuros proyectos que estén orientados a una problemática similar.

5.2. Trabajo a futuro

A continuación, se presentan algunas características o modificaciones que puedan desarrollarse a futuro o que permitan que el prototipo pueda brindar una mejor funcionalidad en otros equipos de menor rendimiento en procesamiento:

- Aumentar el número de videos que se puedan subir para tener una cobertura más amplia en lugares que tengan una mayor cantidad de videos a analizar.
- Para la etapa de clasificación donde se utiliza el clasificador euclidiano se puede mejorar implementando métodos de clasificación más robustos y eficientes como lo son algoritmos de clasificación no supervisados o redes neuronales.
- Incorporar las diferentes versiones del algoritmo MTCNN ya con estas últimas versiones se obtendrían detecciones más precisas a diferentes distancias focales y con un número variable de personas presentes en el video.
- Incorporar un nuevo módulo al prototipo para el uso de Reconocimiento Facial en grabaciones en vivo, ya sea para cámaras Web que estén conectadas al equipo o para cámaras IP que estén conectadas a la misma red que el equipo.
- Incorporar este proyecto a otra API *open source* que utilice un algoritmo diferente para la detección de rostros.

.

Glosario

A

API (Application Programming Interface)

Segmento de código que permite a diferentes aplicaciones comunicarse entre sí., 3

B

Backend

Consiste en el acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, 25

C

Canvas

Elemento HTML que se utiliza para dibujar gráficos, hacer composiciones de fotos o incluso realizar animaciones., 81

D

DVR (Digital Video Recorder)

Dispositivo de grabación de video en formato digital. Además de visualización en vivo, monitoreo remoto a través de internet., 3

F

FaceNet

Sistema desarrollado por Google, y que es uno de los sistemas de reconocimiento facial más precisos que existen., 8

Framework

Esquema o marco de trabajo que ofrece una estructura base para elaborar un proyecto con objetivos específicos, sirve como punto de partida para el desarrollo de software, 25

Frontend

Consiste en la conversión de datos en una interfaz gráfica para que el usuario pueda interactuar con el sistema, 25

I

InsightFace

Biblioteca integrada de Python de análisis de rostros 2D y 3D de código abierto., 9

M

Middleware

Software con el que diferentes aplicaciones se comunican entre sí. Brinda funcionalidad para conectar aplicaciones de manera inteligente y eficiente, 26

MTCNN (Multit-task Cascaded Convolutional Networks)

Red neuronal que se utiliza para tratar simultáneamente la detección de rostros y el posicionamiento de los puntos clave del rostro., 24

N

NVR (Network Video Recorder)

Dispositivo físico que opera cámaras de videovigilancia IP a través de una red inalámbrica., 3

O

Open Source

Código diseñado de manera que sea accesible al público, todos pueden modificar y distribuir el código de cualquier manera., 8

P

PCA (Principal Component Analysis)

Método estadístico que permite simplificar la complejidad de espacios muestrales con muchas dimensiones a la vez que conserva su información., 15

T

TensorFlow

Biblioteca de código abierto para aprendizaje automático, desarrollado por Google para el reconocimiento de imágenes., 10

Bibliografía

A. Llorca, Á. (15 de Octubre de 2015). *OpenFace, un nuevo software de reconocimiento facial, de código abierto*. Obtenido de Genbeta: https://www.genbeta.com/actualidad/openface-un-nuevo-software-de-reconocimiento-facial-de-codigo-abierto

Alonso. (13 de Febrero de 2006). *6 Clasificación de imágenes*. Obtenido de https://www.um.es/geograf/sigmur/temariohtml/node74.html

Amos, B. (s.f.). *OpenFace*. Recuperado el 6 de Agosto de 2022, de OpenFace: https://cmusatyalab.github.io/openface/

Arriagada Rodríguez, M. (Agosto de 2015). Comparación de métricas de distancia en el algoritmo K-Vecinos Más Cercanos para el problema de Reconocimiento Automático de Dígitos Manuscritos. Valparaíso, Región de Valparaíso, Chile.

Azcárate Aragón, M. (Junio de 2022). Estudio sobre las Técnicas de Extracción de Landmarks para la detección Facial. Madrid, España.

Cabello Pardos, E. (2004). Técnicas de reconocimiento facial mediante redes neuronales. Madrid, España. Recuperado el 24 de Septiembre de 2022

Cordobés Menguiana, A. B. (2019). Integración y evaluación de sistemas de reidentificación de caras usando MTCNN y FaceNet en C++. Sevilla, España. Recuperado el 19 de Septiembre de 2022

Domínguez Pavón, S. (2017). Fundamentos teóricos de PCA. *Reconocimiento facial mediante el Análisis de Componentes Principales (PCA)*. Sevilla, España.

Gil Leiva, I., Díaz Ortuño, P., & Rodríguez Muñoz, J. V. (s.f.). Técnicas y usos en la clasificación automática de imágenes. Región de Murcia, España. Recuperado el 2 de Agosto de 2022

Gimeno Hernández, R., & Morros i Rubio, J. R. (Mayo de 2010). Estudio de Técnicas de Reconocimiento Facial. Barcelona, España. Recuperado el 25 de Noviembre de 2022

Gómez, J. C. (s.f.). Procesamiento Digital de Imágenes. Recuperado el 1 de Septiembre de 2022

Graph Everywhere. (s.f.). *Algoritmo de distancia euclidiana*. Recuperado el 5 de Diciembre de 2022, de Graph Everywhere: https://www.grapheverywhere.com/algoritmo-de-distancia-euclidiana/

Grupo Atico34. (s.f.). *Reconocimiento facial ¿Qué es y cómo funciona?* Recuperado el 29 de Noviembre de 2022, de Grupo Atico34: https://protecciondatos-lopd.com/empresas/reconocimiento-facial/#Tecnicas_de_reconocimiento_facial

Hikvision. (2018). DS-7100HGI-F1 Series. *Especificaciones técnicas*. Recuperado de: https://ftp3.syscom.mx/usuarios/ftp/2020/04/03/86dde/UD12968B%20DS-7100HGHI-F1%20V3.4.89%2020181228.pdf

HiLook. (2020). DVR-200G-F1 Series Turbo HD DVR. Especificaciones técnicas. Recuperado de: https://ftp3.syscom.mx/usuarios/ftp/2020/04/24/935a7/UD12972B%20Datasheet%20of%20DVR-200G-F1.pdf

IDIS. (2020). DR-1308P. Especificaciones técnicas. Recuperado de: https://ftp3.syscom.mx/usuarios/luis.vargas/IDIS/Brochure/DR-1308P.pdf

Ingeoexpert. (s.f.). *IngeOexpert*. Recuperado el 5 de 12 de 2022, de Clasificaciones de imágenes de sátelite: https://ingeoexpert.com/articulo/clasificaciones-de-imagenes-de-satelite/

Keong, J., Dong, X., Jin, Z., Mallat, K., & Dugelay, J.-L. (s.f.). Multi-spectral Facial Landmark Detection. Recuperado el 22 de Septiembre de 2022

Khabarlak, K., & Koriashkina, L. (25 de Abril de 2022). Fast Facial Landmarks Detection and Applications: A Survey.

Khadzkou, A. (s.f.). *Exadel CompreFace is a leading free and open-source face recognition system*. Recuperado el 12 de Agosto de 2022, de GitHub: https://github.com/exadel-inc/CompreFace#getting-started-with-compreface

Knudby, A. (s.f.). *LibreText*. Recuperado el 5 de Enero de 2023, de 1.6: Clasificaión: https://espanol.libretexts.org/Geociencias/Geograf%C3%ADa_(F%C3%ADsica)/Teledetecci%C3%B3n_(Knudby)/01%3A_Cap%C3%ADtulos/1.06%3A_Clasificaci%C3%B3n

Kopaczka, M., Acar, K., & Merhof, D. (s.f.). Robust Facial Landmark Detection and Face Tracking in Thermal Infrared Images using Active Appearance Models. Aquisgrán, Alemania. Recuperado el 24 de Septimebre de 2022

Kostinger, M., Wohlhart, P., Roth, P. M., & Bischof, H. (s.f.). Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. Recuperado el 8 de Diciembre de 2022

Lisa Institute. (7 de Abril de 2021). *Reconocimiento facial: Descubre cómo funciona y quién (y para qué) lo utiliza*. Obtenido de Lisa Institute: https://www.lisainstitute.com/blogs/blog/reconocimiento-facial-como-funciona-quien-utiliza

Liu, Z., Zhu, X., Hu, G., Guo, H., Tang, M., Lei, Z., . . . Wang, J. (s.f.). Semantic Alignment: Finding Semantically Consistent Ground-truth for Facial Landmark Detection. Beijing, China. Recuperado el 10 de Diciembre de 2022

López, P. D., Valle, R., & Baumela, L. (s.f.). Facial Landmarks Detection using a Cascade of Recombinator Networks. Madrid, España. Recuperado el 4 de Septiembre de 2022

Mahbub, U., Sarkar, S., & Chellappa, R. (10 de Enero de 2018). Segment-based Methods for Facial Attribute Detection from Partial Faces.

Martínez, J. (24 de Junio de 2022). *Fundamentos de la Clasificación de Imágenes*. Obtenido de DataSmarts: https://www.datasmarts.net/fundamentos-de-la-clasificacion-de-imagenes/

Merget, D., Rock, M., & Rigoll, G. (s.f.). Robust Facial Landmark Detection via a Fully-Convolutional Local-Global Context Network. Recuperado el 20 de Septiembre de 2022

Molina, R. (s.f.). Super resolucion de imágenes y vídeo (I). Granada, España. Recuperado el 5 de Enero de 2023

Moujahid, A., Inza, I., & Larrañaga, P. (s.f.). Tema 5. Clasificadores K-NN. Recuperado el 3 de Septiembre de 2022

Mühler, V. (25 de Junio de 2018). *JavaScript API for Face Recognition in the Browser with tensorflow.js*. Obtenido de ITNEXT: https://itnext.io/face-api-js-javascript-api-for-face-recognition-in-the-browser-with-tensorflow-js-bcc2a6c4cf07

Mühler, V. (16 de Julio de 2018). *Realtime JavaScript Face Tracking and Face Recognition using face-api.js' MTCNN Face Detector*. Obtenido de ITNEXT: https://itnext.io/realtime-javascript-face-tracking-and-face-recognition-using-face-api-js-mtcnn-face-detector-d924dd8b5740

Mühler, V. (22 de Marzo de 2020). *face-api.js*. Obtenido de GitHub: https://github.com/justadudewhohacks/face-api.js/

Networks, U. (2021). UniFi Protect Network Video Recorder. Especificaciones técnicas. Recuperado de: https://ftp3.syscom.mx/usuarios/luis.vargas/IDIS/Brochure/DR-1308P.pdf

Ouanan, H., Ouanan, M., & Aksasse, B. (s.f.). Facial Landmark Localization: past, present and future. Recuperado el 21 de Diciembre de 2022

Pantic, M., & Rothkrantz, L. J. (Junio de 2004). Facial Action Recognition for Facial Expression Analysis From Static Face Images. Recuperado el 20 de Diciembre de 2022

Ramos Almeida, D. (20 de Septiembre de 2018). DETECCIÓN AUTOMÁTICA DE PUNTOS FACIALES. Bogotá, Colombia.

Ranjan, R., Patel, V. M., & Chellappa, R. (2016). HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition. Recuperado el 9 de Diciembre de 2022

Restrepo Vargas, J. L. (s.f.). REVISIÓN DE LAS TÉCNICAS BÁSICAS PARA EL RECONOCIMIENTO DE ROSTROS. Recuperado el 10 de Agosto de 2022

Serengil, S. (s.f.). *deepface*. Recuperado el 10 de Agosto de 2022, de GitHub: https://github.com/serengil/deepface?ref=hackernoon.com

Song, H., Yang, U., Lee, S., & Sohn, K. (5 de Octubre de 2022). 3D Face Recognition Based on Facial Shape Indexes with Dynamic Programming . Seúl, Corea del Sur.

Sotaquirá, M. (15 de Junio de 2020). *Detección de Rostros con Machine Learning*. Obtenido de Codificandobits: https://www.codificandobits.com/blog/deteccion-de-rostros-machine-learning/#:~:text=Multi%2DTask%20cascaded%20convolutional%20networks%20(MTCNN)&text=Cuando%20para%20un%20rostro%20se,conocido%20como%20non%2Dmax%20supression.

Traichuk, A. (27 de Abril de 2021). *Los 6 mejores proyectos de código abierto para el reconocimiento facial en tiempo real*. Obtenido de Hackernoon: https://hackernoon.com/es/6-mejores-proyectos-de-codigo-abierto-para-reconocimiento-facial-en-tiempo-real-vr1w34x5

Vázquez López, M. Á. (Marzo de 2014). Sistema de Reconcocimiento Facial Mediante Técnicas de Visión Tridimensional. León, Guanajuato, México.

Vélez, N. J., Erazo, J. H., & Loaiza, H. (s.f.). Sistema de clasificación de imágenes basado en técnicas de reconocimiento de patrones aplicado en termografía y robótica. Recuperado el 3 de Agosto de 2022

Viejo, D., & Cazorla, M. (s.f.). Construcción de mapas 3D y extracción de primitivas geométricas del entorno. Alicante, España. Recuperado el 16 de Diciembre de 2022

Wu, Y., & Ji, Q. (26 de Abril de 2018). Facial Landmark Detection: A Literature Survey.

Wu, Y., Gou, C., & Ji, Q. (s.f.). Simultaneous Facial Landmark Detection, Pose and Deformation Estimation under Facial Occlusion. Recuperado el 21 de Septiembre de 2022

Yan, Y., Duffner, S., Phutane, P., Berthelier, A., Naturel, X., Blanc, C., . . . Chateau, T. (6 de Julio de 2020). Fine-grained facial landmark detection exploiting intermediate feature representations.

Yan, Y., Naturel, X., Chateau, T., Duffner, S., Garcia, C., & Blanc, C. (7 de Julio de 2020). A survey of deep facial landmark detection. Paris, Francia.

Yusuf, A. A., Sufyanu, Z., Mohamad, F. S., & Nanaa, K. (s.f.). Facial Landmark Detection and Estimation under Various Expressions and Occlusions. Terengganu, Malasia. Recuperado el 14 de Diciembre de 2022

Zhang, Z., Luo, P., Loy, C. C., & Tang, X. (s.f.). Facial Landmark Detection by Deep Multi-task Learning. Hong Kong, China. Recuperado el 7 de Diciembre de 2022

Zhap, W., Chellappa, R., Phillips, P. J., & Rosenfeld, A. (s.f.). Face Recognition: A Literature Survey. Recuperado el 5 de Octubre de 2022

Zhu, X., & Ramanan, D. (s.f.). Face Detection, Pose Estimation, and Landmark Localization in the Wild. Recuperado el 15 de Diciembre de 2022

Referencias de imágenes

Mühler, V. (s.f.). *Face Landmark Detection [figura]*. Obtenido de face-api.js: https://justadudewhohacks.github.io/face-api.js/docs/index.html

OpenFace. (s.f.). *Overview [figura]*. Obtenido de OpenFace: https://cmusatyalab.github.io/openface/

Saba. (s.f.). *Screenshots [figura]*. Obtenido de Github: https://github.com/exadel-inc/CompreFace#getting-started-with-compreface

Serengil, S. (s.f.). *[Ejemplo de funcionalidad de "deepFace"] [figura]*. Obtenido de Github: https://github.com/serengil/deepface?ref=hackernoon.com