Formando líderes para la construcción de un nuevo país en paz

SISTEMA DE MONITOREO DE ACCESO BASADO EN RECONOCIMIENTO FACIAL PARA EL CONTROL Y LA SUPERVISIÓN PRECISA EN UN SALÓN DE CLASE

Autor 1: Juan Pablo Marquez Sanchez

1004922828

Autor 2: Samuel Andres Celis Lizcano

1091964042

Autor 3: Yorman Rodolfo Rodriguez Jaimes

1005027437

Ingeniería de Software II

Universidad de Pamplona

Facultad de Ingeniería y Arquitectura

Ingeniería de Sistemas

2024

Villa del Rosario – Norte de Santander

Índice

1. Resumen 6
2. Abstract 7
3. Introducción 8
4. Planteamiento del problema 10
5. Justificación 11
6. Objetivos 12
   6.1 Objetivó General 12
   6.2 Objetivos Especificos 12
7. Estado del Arte 13
   7.1 Marco Referencial 13
      7.1.1 Referencias internacionales 13
      7.1.2 Referencias nacionales 15
      7.1.3 Referencias regionales 17
   7.2 Marco Conceptual 19
   7.3 Marco Contextual 20
   7.4 Marco teórico 21
   7.5 Marco Legal 22
8. Metodología 23
   8.1 Diagrama de Bloque de la Metodología XP: 24
   8.2 Fases de la Metodología XP 24

8.3 Tipos de datos 25

8.4 Técnicas de preprocesamiento de datos: 25

Detección de Rostros: 25

Normalización: 26

Extracción de Características: 26

9. Desarrollo 26

9.1 Identificar herramientas y tecnologías viables para el desarrollo de un sistema de reconocimiento facial. 26

9.2 Establecer lenguajes y tecnologías factibles identificados en documentos y publicaciones relevantes para el desarrollo de un sistema de reconocimiento facial. 27

Lenguaje: 27

Herramienta de desarrollo Front-end: 27

Librería de Reconocimiento Facial: 27

9.3 Crear un sistema de control de acceso basado en reconocimiento facial para la supervisión precisa en un salón clase 28

Front-end: 28

Captura y entrenamiento del reconocimiento Facial: 32

Sistema de reconocimiento facial: 38

Anexo de Pruebas del funcionamiento del reconocimiento: 42

10. Conclusiones 46
11. Referencias 48

Tabla de Ilustraciones

Ilustración 1 Diagrama en Bloques de la Metodología XP 24

Ilustración 2 Importaciones de las librerías para el Front-End 28

Ilustración 3 inicialización de la aplicación 29

Ilustración 4 iniciador de la aplicación 30

Ilustración 5 Sección de la función generar_formulario 31

Ilustración 6 importaciones del sistema de reconocimiento facial 32

Ilustración 7 Función main 33

Ilustración 8 Función de captura de imágenes 34

Ilustración 9 Función de entrenamiento 36

Ilustración 10 Importaciones de reconocimiento 38

Ilustración 11 Carga del modelo Pre-Entrenado 38

Ilustración 12 Activación del reconocimiento 40

Ilustración 13 Prueba reconocimiento Facial 42

Ilustración 14 Interfaz sección login (Fuente: los autores) 42

Ilustración 15 Interfaz sección materias creadas (Fuente: los autores) 43

Ilustración 16 Interfaz sección registro de estudiantes (Fuente: los autores) 43

Ilustración 17 Interfaz sección creación de clase (Fuente: los autores) 44

Ilustración 18 Interfaz sección creación de lista de asistencia (Fuente: los autores) 44

Ilustración 19 Interfaz sección creación de lista de asistencia (Fuente: los autores) 45

1. Resumen

Para nadie es un secreto que hoy en día la toma de asistencia es accesible a fraudes, en la mayoría de las aulas estudiantiles las asistencias son llevadas en planillas de firmas y hojas de Excel. El problema es que ambos métodos son alterables por compañeros o el mismo docente que no desea poner inasistencias, a esto se le suma el impacto ambiental que se genera con el uso de papel en las entidades no sistematizadas. Ahora, ¿Cómo puede un sistema de monitoreo de acceso basado en reconocimiento facial para el control y la supervisión precisa en un salón de clase, reducir el impacto ambiental y aumentar la eficiencia operacional en una institución educativa?, con esto en mente se planteó el objetivo de “crear un sistema de monitoreo de acceso basado en reconocimiento facial para el control y la supervisión precisa en un salón de clase”. Todo este desarrollo fue realizado bajo la metodología ágil Extreme Programming (XP) para lograr una entrega de software funcional de manera rápida aprovechando su excelente forma de comunicación y colaboración. Utilizaremos imágenes faciales de un grupo de sistemas inteligentes AR para verificar la identidad de las personas y garantizar que no haya suplantación. Los datos faciales nos permiten crear un sistema más seguro, eficiente y confiable para el control de acceso y la supervisión precisa de los estudiantes. Al culminar el desarrollo se entregará un prototipo ejecutable del sistema de reconocimiento facial.

Palabras clave: Control de Acceso, Reconocimiento facial, Papel, Eficiencia, Fraude.

2. Abstract

It is no secret to anyone that nowadays attendance taking is open to fraud; in most student classrooms, attendance is recorded on signature forms and Excel sheets. The problem is that both methods are alterable by classmates or the teacher himself who does not want to make absences. Added to this is the environmental impact generated by the use of paper in non-systematized entities. Now, how can an access monitoring system based on facial recognition for precise control and supervision in a classroom, reduce environmental impact and increase operational efficiency in an educational institution? With this in mind, the question was raised. objective of “creating an access monitoring system based on facial recognition for precise control and supervision in a classroom.” All this development was carried out under the agile Extreme Programming (XP) methodology to achieve a quick delivery of functional software taking advantage of its excellent form of communication and collaboration. We will use facial images from a group of intelligent AR systems to verify people's identity and ensure there is no impersonation. Facial data allows us to create a more secure, efficient, and reliable system for access control and accurate student supervision. Upon completion of development, an executable prototype of the facial recognition system will be delivered.

Keywords: Access Control, Facial recognition, Paper, Efficiency, Fraud.

3. Introducción

Es evidente que los sistemas actuales de registro de asistencia, que a menudo se realizan físicamente mediante hojas de firmas o digitalmente en hojas de Excel, presentan vulnerabilidades significativas. Estos métodos son susceptibles a manipulaciones fraudulentas por parte de los estudiantes o incluso del personal docente que puede estar reacio a registrar inasistencias. Además, el uso de papel para las listas de asistencia tiene implicaciones medioambientales perjudiciales. La fabricación de papel implica la tala de árboles (pochteca papel, 2021), lo que acelera la deforestación y la pérdida de biodiversidad. Este proceso también consume grandes volúmenes de agua y energía, y genera emisiones de gases de efecto invernadero.

En el caso de la Universidad de Pamplona, que cuenta con 1818 docentes según la base de datos INEST (Universidad de Pamplona, 2023), el impacto es aún mayor. Considerando un promedio de dos clases por docente, cada una con dos secciones semanales, resulta en cuatro listas de asistencia por docente cada semana. A lo largo de un semestre de 16 semanas, esto se traduce en un consumo de 64 hojas de papel por docente, lo que equivale a un total de 116.352 hojas de papel consumidas por todos los docentes durante un semestre. Este consumo masivo de papel tiene un impacto ambiental significativo.

Con todo lo expuesto anteriormente se planteó la siguiente pregunta: ¿Cómo puede un Sistema de Monitoreo de Acceso Basado en Reconocimiento Facial mejorar la precisión y la integridad de los registros de asistencia en un salón de clases, al

tiempo que reduce el impacto ambiental y aumenta la eficiencia operacional en instituciones educativas?, esta pregunta busca formular la creación de un sistema de monitoreo de acceso basado en reconocimiento facial para el control y la supervisión precisa en un salón de clase. Para lograr al objetivo planteado se optó por utilizar la metodología ágil XP (Gnzo, 2024) para lograr una entrega de software funcional de manera rápida aprovechando su excelente forma de comunicación y colaboración entre los involucrados.

En cuanto a los datos utilizaremos imágenes faciales (kaspersky, 2024) del grupo de sistemas inteligentes AR de la universidad de pamplona sede Villa del Rosario para verificar la identidad de las personas y garantizar que no haya suplantación. Los datos faciales nos permiten crear un sistema más seguro, eficiente y confiable para el control de acceso y la supervisión precisa de los estudiantes. La validez de estos datos es alta, ya que las características faciales son únicas para cada individuo y difíciles de falsificar. Sin embargo, es importante asegurar la calidad de las imágenes capturadas, ya que factores como la iluminación, la orientación del rostro y la resolución de la imagen pueden afectar la precisión del reconocimiento facial.

Al culminar el desarrollo se entregará un prototipo ejecutable del sistema de reconocimiento facial para la implementación en las asignaturas de: Sistemas inteligentes AR y Fundamentos de computación paralela y distribuida AR para que el docente puede llevar de manera más optima el proceso de la toma de asistencia.

4. Planteamiento del problema

En la universidad, la asistencia se registra manualmente utilizando una hoja de firmas física. Los estudiantes firman su nombre al entrar a la clase. Una vez recogidas las firmas, el docente tiene que escanear cada hoja de asistencia y convertirla en un archivo PDF. Este archivo se envía por correo electrónico a los coordinadores de registro y control de la universidad. Este proceso se repite para cada una de las cuatro clases que el docente imparte cada semana durante el semestre de 16 semanas. Algunos docentes tienen sus propias impresoras para imprimir las hojas de asistencia, mientras que otros deben recurrir a servicios de impresión externos.

Analizando el funcionamiento actual del proceso, pensamos que sería mucho más optimo tener una herramienta que permita realizar el proceso de (creación, llenado y sistematización) de las listas de asistencia en las aulas, por medio de una aplicación que sistematice esos procesos, anulando el uso de papel en el proceso.

Pero, ¿Cómo puede un sistema de monitoreo de acceso basado en reconocimiento facial para el control y la supervisión precisa en un salón de clase, reducir el impacto ambiental y aumentar la eficiencia operacional en una institución educativa?

5. Justificación

Un sistema de reconocimiento facial garantiza la autenticidad de los registros de asistencia, eliminando la posibilidad de manipulaciones fraudulentas. Esto asegura que los datos recogidos son precisos y fiables, lo cual es crucial para la evaluación del rendimiento estudiantil y la planificación académica. Al eliminar la necesidad de papel para las listas de asistencia, se reduce la demanda de recursos naturales y se minimiza la huella de carbono de la institución. Esto está en línea con los esfuerzos globales para promover prácticas sostenibles y proteger el medio ambiente.

La automatización del proceso de toma de asistencia ahorra tiempo y recursos, permitiendo a los docentes y administradores centrarse en tareas más importantes. Esto mejora la eficiencia operacional de la institución. Un sistema de reconocimiento facial puede ser implementado en instituciones de cualquier tamaño, lo que lo hace adaptable y escalable. Esto es especialmente útil para instituciones grandes donde el manejo manual de la asistencia puede ser una tarea desalentadora.

6. Objetivos

6.1 Objetivó General

Crear un sistema de monitoreo de acceso basado en reconocimiento facial para el control y la supervisión precisa en un salón de clase.

6.2 Objetivos Especificos

- Identificar herramientas y tecnologías viables para el desarrollo de un sistema de reconocimiento facial.
- Establecer lenguajes y tecnologías factibles identificados en documentos y publicaciones relevantes para el desarrollo de un sistema de reconocimiento facial.
- Crear un sistema de control de acceso basado en reconocimiento facial para la supervisión precisa en un salón clase.
- Validar el correcto funcionamiento del sistema de control de Acceso Basado en Reconocimiento Facial con datos tomados en un salón de clases.

7. Estado del Arte

7.1 Marco Referencial

7.1.1 Referencias internacionales

Detección de objetos y seguimiento de su posición en tiempo real con Raspberry Pi: Una de las áreas de rápido crecimiento del aprendizaje profundo mediante inteligencia artificial es la visión por ordenador, cada vez más popular. Se trata de un campo de investigación en auge que busca crear técnicas que ayuden a los ordenadores a "ver" y reconocer la información de imágenes digitales como, por ejemplo, vídeos y fotografías. La detección de objetos es un método de visión por ordenador que nos permite reconocer objetos en una imagen o vídeo y localizarlos. Este artículo describe un método eficiente de identificación de objetos basado en la forma y su desplazamiento en tiempo real utilizando la librería OpenCV de roles de programación mayoritariamente orientados a visión por computador y Raspberry Pi con módulo de cámara. (Gokulnath Anand, 2021)

Plataforma de visión artificial de código abierto para fabricación y robótica: Este artículo describe el diseño y la implementación de una plataforma de visión artificial de código abierto para el guiado visual de robots y la inspección automatizada de productos en la fabricación, basada en la biblioteca OpenCV. El uso de esta plataforma permite relajar la organización rígida y específica de la industria de los flujos de materiales, haciendo que los recursos del taller sean conscientes de la realidad. Las principales funcionalidades son: adquisición de secuencias de vídeo de múltiples

fuentes, análisis de imágenes a nivel de escena, reconocimiento de piezas, localización e interacción con equipos industriales mediante protocolos de comunicación estándar y abiertos. El artículo describe los aspectos de diseño: arquitectura del sistema, adquisición de datos y normalización de la representación de imágenes utilizada por los algoritmos de análisis y el módulo de reconocimiento de objetos, y protocolos de interacción de entrada/salida para un conjunto de casos predefinidos. Se presentan los resultados de una aplicación de la plataforma que utiliza un dispositivo comercial de adquisición de imágenes y un robot industrial. (Silviu Răileanu, 2021)

Automatización de formularios de examen mediante reconocimiento facial: La forma en que la Ciencia de Datos y el Aprendizaje Automático han marcado las tendencias modernas para la automatización. Está pensado para simplificar nuestras actividades cotidianas relacionadas con estos dominios. Una de estas tareas es la presentación repetitiva de los mismos datos personales al enviar cualquier formulario en línea con fines académicos o de otro tipo. La repetitividad parece un poco dilatoria. Además, la implicación del aprendizaje automático fomenta aún más el concepto de automatizar la reiteración de los datos personales cada vez que rellenamos un formulario, en lugar de la introducción manual. Para obviar la redundancia anterior, hemos propuesto implementar "Exam Form Automation Using Facial Recognition" que incluye el reconocimiento facial en tiempo real con la ayuda de la webcam, pre-almacenamiento de datos en servidores web con Panda y NumPy biblioteca de Python y, a continuación, la automatización de las entradas en el formulario utilizando selenium

biblioteca de Python. El sistema captura una imagen en tiempo real de un usuario y recoge los datos detalles de los datos alojados en un servidor web. De este modo, se evita el monótono procedimiento de entrada manual. (Bishnu Deo Kumar, 2023)

7.1.2 Referencias nacionales

Comparación de modelos de redes neuronales convolucionales para reconocimiento facial de usuarios: Este artículo compara modelos conocidos de redes neuronales convolucionales (CNN) para el reconocimiento facial. Para esto, utiliza su base de datos creada a partir de dos usuarios registrados y una categoría adicional de personas desconocidas. Se compararon ocho modelos base diferentes de arquitecturas convolucionales mediante la transferencia de aprendizaje, y se propusieron dos modelos adicionales llamados CNN superficial y gráfico acíclico dirigido con CNN (DAG-CNN), que son arquitecturas de poca profundidad (seis capas de convolución). Dentro de las pruebas con la base de datos, los mejores resultados fueron obtenidos por los modelos GoogLeNet y ResNet-101, logrando clasificar el 100% de las imágenes, incluso sin confundir a personas fuera de los dos usuarios. Sin embargo, en una prueba adicional en tiempo real, en la que uno de los usuarios cambió su estilo, los modelos que mostraron la mayor robustez en esta situación fueron el Inception y el ResNet-101, siendo capaces de mantener un reconocimiento constante. Esto demostró que las redes de mayor profundidad logran aprender características más detalladas de los rostros de los usuarios, a diferencia de las más superficiales; su

aprendizaje de características es más generalizado. Se debe declarar el término completo de una abreviatura/acrónimo cuando se menciona por primera vez.

Sistema de Control de Accesos basado en Voz y Reconocimiento Facial mediante Inteligencia Artificial: La seguridad informática es una preocupación global, ya que todas las empresas manejan información confidencial que requiere protección.

La falta de protección adecuada puede resultar en pérdidas financieras significativas y procedimientos complicados para resolver los problemas. Este artículo propone un sistema de autenticación basado en parámetros biométricos para fortalecer la seguridad de los activos sensibles. Se aplicó una red neuronal convolucional, MobileNet, para lograr una alta precisión en la clasificación de usuarios. El sistema, que se visualiza a través de un ordenador y dispositivos de bajo coste como Raspberry Pi 3, ofrece autenticación por voz y rostro con una precisión del 96% y 100%, respectivamente. En conclusión, el sistema demuestra que el aprendizaje profundo, en combinación con dispositivos como Raspberry, puede generar un sistema de alto rendimiento y bajo costo que ofrece a las empresas un sistema robusto y de alta precisión para el reconocimiento correcto de los patrones biométricos de los usuarios registrados y capacitados por el sistema.

Uso de algoritmos de aprendizaje profundo para la detección en tiempo real de vasos en espacios confinados utilizando el marco Tensorflow: El texto de la imagen es un resumen de un estudio sobre la aplicación del análisis y reconocimiento de imágenes utilizando aprendizaje profundo para evitar colisiones y

planificar rutas para botes. En 2012, se registraron más de 4515 accidentes de botes pequeños en los Estados Unidos, resultando en 651 víctimas y el 22% de los accidentes ocurrieron entre dos botes. El estudio comparó las métricas de rendimiento de diferentes algoritmos de aprendizaje profundo utilizando conjuntos de datos preentrenados. Los algoritmos utilizados fueron: redes neuronales convolucionales basadas en regiones más rápidas, red neuronal convolucional totalmente basada en regiones, y detector de cajas múltiples con disparo único utilizando los extractores de características: red neuronal residual, inception y redes neuronales convolucionales para aplicaciones móviles de visión para detectar botes genéricos en vías fluviales confinadas. Estos modelos se codificaron en Python, utilizando el marco TensorFlow y la biblioteca OpenCV para el procesamiento de imágenes. Los algoritmos se preentrenaron utilizando la base de datos de imágenes gratuitas publicadas en la web, Microsoft COCO. Como resultado, se encontró que los métodos de redes neuronales convolucionales basadas en regiones más rápidas y red neuronal convolucional totalmente basada en regiones ofrecen una pequeña ventaja en precisión si no se requiere detección de velocidad, pero el método de detector de cajas múltiples con disparo único es útil para detectores de casos en tiempo real, sin embargo, no se desempeñó de manera tan precisa al detectar objetos pequeños.

7.1.3 Referencias regionales

Sistema de reconocimiento facial para asistentes a estadios de futbol: En Colombia, la asistencia a los estadios de fútbol se considera peligrosa. Según una

encuesta realizada en 2014 por el Centro Nacional de Consultoría, solo un pequeño porcentaje de hombres y mujeres, principalmente jóvenes de entre 18 y 24 años, asisten a los partidos. Los principales factores que contribuyen a esta situación son el comportamiento de las 'barras' y la seguridad. Esto plantea un gran desafío para e Estado, los clubes, la Dimayor y la Federación Colombiana de Fútbol, quienes deben desarrollar estrategias para incentivar el regreso de los hinchas tradicionales. En este contexto, este proyecto busca crear un modelo de Red Neuronal Artificial (RNA) capaz de detectar fotografías de delincuentes. El objetivo es que el software del sistema pueda determinar, mediante el análisis de patrones, si una imagen coincide con alguna de las imágenes previamente registradas en una base de datos.

Sistema para el control de acceso de personas en lugares de la alta concentración poblacional empleando tecnología RFID: Este proyecto desarrolló un sistema de control de acceso para lugares con alta concentración de personas utilizando la tecnología RFID. Se creó una matriz de selección en Microsoft Excel y se recopiló información a través de investigaciones y artículos relacionados. Con esta información, se diseñó el sistema de control de acceso utilizando una base de datos y la tecnología RFID. Luego, se desarrolló una aplicación para el control y registro de personas. Finalmente, se evaluó el diseño del sistema y se presentaron las conclusiones del estudio.

Lineamientos desde la industria 4.0 a la educación 4.0: Caso tecnología IoT: Las tecnologías de la Industria 4.0 en el entorno universitario abren nuevas

posibilidades para aplicaciones y servicios, lo que conduce a innovaciones en el proceso de enseñanza-aprendizaje y facilita la llegada de la Educación 4.0. Este artículo propone una ruta para consolidar esfuerzos en la formación de ingenieros aptos para enfrentar los desafíos de la Industria 4.0, que incluye nuevos cursos en los programas de ingeniería, habilidades y competencias blandas, y fábricas de aprendizaje como enlace entre la academia y proyectos reales de la industria. Además, se destaca el Internet de las Cosas como una de las tecnologías clave para alcanzar la Educación 4.0, cuya aplicación mejora el proceso de enseñanza-aprendizaje y permite el monitoreo de la salud y las emociones de los estudiantes.

7.2 Marco Conceptual

OpenCv: OpenCV es una biblioteca de software de código abierto que permite a los desarrolladores acceder a rutinas utilizadas para aplicaciones de visión por computadora en la API (Interfaz de Programación de Aplicaciones). Si alguien hace clic en el timbre de la puerta o si la cámara detecta actividad sospechosa, la imagen se grabará utilizando el código de software para la detección de actividad escrito en Python y OpenCV. Después de que la imagen esté registrada, se detecta y segmenta el código de Python y OpenCV del fotograma. El código de reconocimiento facial para identificar las caras almacenadas en la base de datos se proporciona como una cara segmentada. (Zhu Zhiguo, 2020)

Python: Es un lenguaje de programación ampliamente utilizado en las aplicaciones web, desarrollo de software, la licencia de datos y machine learning (ML).

Los desarrolladores utilizan Python poque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. (AWS, 2024)

Modelos de OpenCV: Durante la investigación de OpenCV y su aplicación, encontramos diferentes modelos, tales como: Eigenfaces, Fisherfaces y Local Binary Patterns Histograms (LBPH), todos estos son técnicas efectivas para el reconocimiento facial. Todos los modelos presentaban la cualidad de identificar características y diferencias entre individuos con alta precisión. (OpenCV, 2024)

PyQt6: Es una biblioteca de enlace Python para la biblioteca de widgets Qt, que proporciona herramientas para la creación de interfaces graficas de usuario (GUI) robustas y visualmente atractivas. (Riverbank Computing, 2024)

FaceNet: Es un modelo de reconocimiento facial basado en aprendizaje profundo desarrollado por Google, que utiliza redes neuronales convolucionales para mapear rostros a un espacio vectorial donde las distancias euclidianas directamente corresponden a la similitud de los rostros. (Cornell University, 2015)

7.3 Marco Contextual

El proyecto presentado tiene lugar en la clase de sistemas Inteligentes AR de la universidad de Pamplona sede villa del Rosario. Este busca prevenir la suplantación por parte de compañeros a la hora de la toma de asistencia del curso. Uno de los motivos para la creación del proyecto es ayudar a reducir el tiempo invertido por los docentes a este proceso que se puede hacer en unos minutos, a su vez reduciendo el costo que este implica a los docentes por la impresión de cada hoja de asistencia.

7.4 Marco teórico

Asistencia basada en tecnología de reconocimiento facial Aplicación del sistema de gestión en Smart Classroom: Los métodos tradicionales de registro de asistencia pueden reemplazarse con métodos de inteligencia artificial (IA). Este documento tiene como objetivo desarrollar un sistema de asistencia automatizado con una interfaz gráfica fácil de usar en árabe para mejorar la interacción del usuario. El lenguaje de programación Python se utiliza para construir el sistema, que emplea algoritmos de detección de objetos y extracción de características. Este sistema procesa una retransmisión en directo desde una cámara IP instalada en el aula. El sistema emplea algoritmos de detección de objetos para detectar los rostros en el video. Luego, las ubicaciones de las caras se envían a los algoritmos de extracción de características para extraer y comparar características con las almacenadas en la base de datos. Nuestro artículo concluye que automatizar el proceso de registro de asistencia utilizando tecnología de reconocimiento facial puede lograr hasta un 100% de precisión. Comparando nuestro sistema propuesto con estudios anteriores, encontramos que los supera en varios aspectos. Nuestro método maneja pruebas en tiempo real con un mayor número de estudiantes. Se adapta a los estudiantes ubicados a diferentes distancias de la cámara dentro del aula, lo que demuestra la eficacia de nuestro método para automatizar el proceso de registro de asistencia. (Lateef, 2023)

7.5 Marco Legal

La Ley 1581 de 2012: tiene como finalidad regular el tratamiento de datos personales para garantizar y proteger el derecho fundamental de habeas data, que consiste en el derecho de las personas a conocer, actualizar y rectificar la información que se haya recogido sobre ellas en bases de datos.

Consentimiento: Es necesario obtener el consentimiento previo, expreso e informado del titular de los datos para el tratamiento de sus datos personales.

Responsabilidades: Las organizaciones que recolecten y traten datos personales deben garantizar la seguridad de la información y prevenir su alteración, pérdida, consulta, uso o acceso no autorizado. (Colombia potencia de la vida, 2012)

Constitución Política de Colombia, Artículo 15: Este artículo garantiza el derecho a la intimidad y a la protección de los datos personales. Cualquier recolección, tratamiento y uso de datos personales debe respetar este derecho constitucional. (Colombia presidencia de la republica, 2011)

Ley 527 de 1999 (Ley de Comercio Electrónico): Esta ley regula los documentos electrónicos y la firma digital, estableciendo la equivalencia de los documentos electrónicos y las firmas digitales a los documentos y firmas físicas. (Colombia potencia de la vida, 1999)

8. Metodología

Se decidió usar la metodología XP porque es especialmente adecuada para proyectos de tecnología avanzada como el de reconocimiento facial debido a su capacidad para adaptarse rápidamente a los cambios. En proyectos donde las investigaciones, descubrimientos o requisitos legales pueden cambiar con frecuencia, XP facilita la incorporación rápida de estos cambios gracias a su enfoque en entregas pequeñas y frecuentes. Esto asegura que el sistema se mantenga relevante y actualizado.

Además, XP promueve una mejora continua y una alta calidad del software mediante prácticas como la integración continua y el desarrollo guiado por pruebas. Estas prácticas garantizan que el sistema de reconocimiento facial sea siempre funcional y de alta calidad, facilitando la detección temprana de errores y asegurando que el software cumpla con los requisitos necesarios. En un sistema donde la precisión y la fiabilidad son esenciales, como en el reconocimiento facial, minimizar errores y asegurar un funcionamiento correcto en diferentes escenarios es crucial.

8.1 Diagrama de Bloque de la Metodología XP:

Ilustración 1 Diagrama en Bloques de la Metodología XP

(Santiago Quilachamín Simbaña, 2021)

8.2 Fases de la Metodología XP

Las fases de la metodología XP son como se muestran en el gráfico: planificación, diseño, codificación, pruebas y lanzamiento (Mancuzo, 2020).

Su sincronización con los objetivos específicos planteados por nosotros, se puede verificar que son acordes a lo expuesto, ya que “Identificar herramientas y tecnologías viables para el desarrollo de un sistema de reconocimiento facial” es parte de la fase de **Planificación,** “Establecer lenguajes y tecnologías factibles identificados en documentos y publicaciones relevantes para el desarrollo de un sistema de

reconocimiento facial.” Haría parte de la fase de **Diseño**, “Crear un sistema de control de acceso basado en reconocimiento facial para la supervisión precisa en un salón clase” Conformaría la fase de **Codificación,** “Validar el correcto funcionamiento del sistema de control de Acceso Basado en Reconocimiento Facial con datos tomados en un salón de clases” concluyendo con la fase de **Pruebas.**

8.3 Tipos de datos

Utilizaremos imágenes faciales de un grupo de sistemas inteligentes AR para verificar la identidad de las personas y garantizar que no haya suplantación. Los datos faciales nos permiten crear un sistema más seguro, eficiente y confiable para el control de acceso y la supervisión precisa de los estudiantes y el docente. La validez de estos datos es alta, ya que las características faciales son únicas para cada individuo y difíciles de falsificar.

Sin embargo, es importante asegurar la calidad de las imágenes capturadas, ya que factores como la iluminación, la orientación del rostro y la resolución de la imagen pueden afectar la precisión del reconocimiento facial.

8.4 Técnicas de preprocesamiento de datos:

Detección de Rostros: Antes de extraer las características faciales, es necesario localizar y segmentar el rostro en la imagen. Esto se puede hacer utilizando técnicas como el algoritmo de Viola-Jones o la Red Neural Convolucional (CNN).

Normalización: Las imágenes de los rostros se normalizan para reducir las variaciones causadas por la iluminación y la orientación del rostro. Esto puede implicar técnicas como la ecualización del histograma y la rotación de la imagen.

Extracción de Características: Las características faciales se extraen de las imágenes normalizadas. Esto puede implicar el uso de técnicas como el análisis de componentes principales (PCA), el análisis discriminante lineal (LDA) o las redes neuronales convolucionales (CNN). Estas técnicas de preprocesamiento de datos ayudarán a mejorar la precisión y eficiencia del sistema de reconocimiento facial.

9. Desarrollo

Durante esta sección del documento vamos a dar cumplimientos a los objetivos específicos planeados durante la fase inicial del proyecto.

9.1 Identificar herramientas y tecnologías viables para el desarrollo de un sistema de reconocimiento facial.

Durante varias semanas investigando y recopilando información de las diferentes tecnologías y herramientas a utilizar para el desarrollo del proyecto, encontramos que fueron descritas en el marco conceptual que fueron:

- Python
- OpenCV
- Modelos de OpenCV
- PyQt6

- FaceNet

En la investigación encontramos todas las herramientas y tecnologías anteriormente mencionadas, todas estas fueron consultadas con un conocimiento previo de las tecnologías.

9.2 Establecer lenguajes y tecnologías factibles identificados en documentos y publicaciones relevantes para el desarrollo de un sistema de reconocimiento facial.

Lenguaje: Durante la investigación previa se decidió utilizar el lenguaje de Python para el desarrollo, esto por el uso de las librerías de OpenCV y PyQt6.

Herramienta de desarrollo Front-end: Para esta sección se decidió utilizar la librería de PyQt6 por su posibilidad de implementación de la librería de OpenCV y su forma para crear aplicaciones por medio de sus atributos y métodos.

Librería de Reconocimiento Facial: Para este apartado investigación modelos directamente de OpenCV, pero se decidió implementar FaceNet para la parte del núcleo del proyecto, resulto ser un mejor modelo en cuanto a velocidad y procesamiento de imágenes.

9.3 Crear un sistema de control de acceso basado en reconocimiento facial para la supervisión precisa en un salón clase.

En este objetivo vamos a documentar la parte de creación del sistema de control, por medio de imágenes del código vamos a dar explicación a las partes elementales de la aplicación.

Esta sección presentara explicaciones desde la parte del front-end hasta la del sistema de reconocimiento facial.

Front-end:

```python
import sys
from PyQt6.QtCore import Qt
from PyQt6.QtWidgets import (
    QApplication, QWidget, QLabel, QHBoxLayout, QVBoxLayout, QPushButton,
    QLineEdit, QMessageBox, QSizePolicy, QScrollArea, QTableWidget, QHeaderView)
from PyQt6.QtGui import QFont, QIcon
from PyQt6 import QtGui, QtCore
from components import login
```

Ilustración 2 Importaciones de las librerías para el Front-End

En la ilustración 6 se nos presentan las importaciones realizadas de PyQt6 para la creación de todos los elementos en la interfaz de la aplicación, también encontramos la importación de una clase login de una carpeta components.

Ilustración 3 inicialización de la aplicación

Estas líneas de código establecen la base para una aplicación de GUI en Python utilizando PyQt6. Verifican si el archivo se ejecuta directamente, crean una instancia de la aplicación, inicializan una ventana de inicio de sesión y ejecutan el bucle de eventos de la aplicación.

```python
class inicio (QWidget):

    def __init__(self):
        super().__init__()
        self.InicializarUI()

    def InicializarUI(self):
        self.showMaximized()
        # self.setGeometry(500, 100, 400, 150)
        self.setWindowTitle("Login GenList")
        self.setWindowIcon(QIcon('src/images/LogoempresaA.ico'))
        self.generar_formulario()
        self.show()
```

Ilustración 4 iniciador de la aplicación

En esta sección del código se nos presentan dos funciones la primera toma todos los valores por defecto del __init__() y llama a la función InicializarUI que permite la creación y aparición de la ventana donde tendremos nuestra aplicación, en esta función definimos como cargara la misma, el titulo de la ventana, su icono y realizamos el llamada a una función generar_formulario que nos carga la pantalla inicial de la aplicación y por ultimo mostramos la ventana.

```python
contenedor_principal = QVBoxLayout()
widget_contenedor_principal = QWidget()
widget_contenedor_principal.setStyleSheet("""QWidget{
                            background-color: white;
                            }""")
# widget_contenedor_principal.showMaximized()
widget_contenedor_principal.setLayout(contenedor_principal)

# crear layaout
contenedor_titulo = QHBoxLayout()
# Cambiar el backgroud
widget_contenedor_titulo = QWidget()
diseño_titulo = """QWidget{background: qlineargradient(x1:0, y1:0, x2:0.707, y2:0.707, stop:0.1 rgba(46, 145, 221, 255),
    stop:0.4 rgba(40, 125, 190, 255),
    stop:0.6 rgba(33, 104, 158, 255));
    max-height: 55px;
    }"""
widget_contenedor_titulo.setLayout(contenedor_titulo)
widget_contenedor_titulo.setStyleSheet(diseño_titulo)

# Crear Layout Parte de Registro

self.contenedor_pre_registro = QVBoxLayout()
self.widget_con_pre_registro = QWidget()
self.widget_con_pre_registro.setLayout(self.contenedor_pre_registro)

self.contenedor_pre_pre_registro = QHBoxLayout()
self.widget_contenedor_pre_pre_registro = QWidget()
self.widget_contenedor_pre_pre_registro.setLayout(
    self.contenedor_pre_pre_registro)

self.contenedor_pre_registro.addWidget(
    self.widget_contenedor_pre_pre_registro)

# CAMBIO DE PANTALLA CENTRAL
# boton login
self.boton_registrar = QPushButton("REGISTRAR")
self.usuario_input = QLineEdit()
self.Contra_input = QLineEdit()
self.login_widget = login.generar_formulario_login(
    self.boton_registrar, self.usuario_input, self.Contra_input)

self.contenedor_pre_pre_registro.addWidget(
    self.login_widget)
self.mensaje_emergente = QMessageBox()
# self.boton_registrar.clicked.connect(self.haz_dado_click)
self.boton_registrar.clicked.connect(self.cambiar_pantalla)
```

Ilustración 5 Sección de la función generar_formulario

En estas líneas podemos visualizar el funcionamiento de los componentes en PyQt6, por medio de layouts agregamos elementos a la ventana, a cada layout se le

asigna un respectivo widget para poder editar sus propiedades, como observamos en la imagen modificamos el background de algunos layouts de la aplicación, por último, en la ilustración 9 tenemos la creación del primer botón y campos para digitar la información del usuario y contraseña.

La estructura del front-end se compone del llamado y modificaciones de uno de los 3 layouts principales, el cual se le cambia sus elementos dependiendo de la opción elegida por el usuario.

Captura y entrenamiento del reconocimiento Facial:

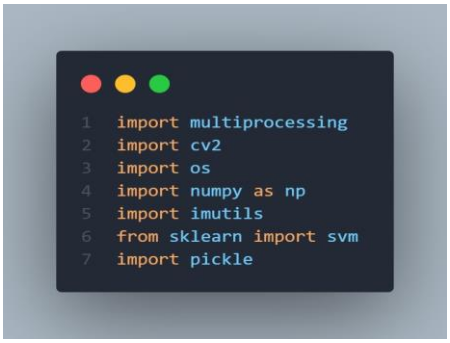

Ilustración 6 importaciones del sistema de reconocimiento facial

```python
if __name__ == '__main__':
    while True:
        personName = input("Ingrese el nombre de la persona (o 'salir' para terminar): ")
        if personName.lower() == 'salir':
            break
        p1 = multiprocessing.Process(target=capture_and_save, args=(personName,))
        p1.start()
        p1.join()

    p2 = multiprocessing.Process(target=train_model)
    p2.start()
    p2.join()
```

Ilustración 7 Función main

El programa entra en un bucle infinito, pidiendo al usuario que ingrese el nombre de una persona. Si el usuario ingresa ‘salir’, el programa termina el bucle y procede a la siguiente parte.

Si el usuario ingresa un nombre, el programa inicia un nuevo proceso utilizando la biblioteca de multiprocesamiento de Python. Este proceso ejecuta la función capture_and_save con el nombre ingresado como argumento. La función capture_and_save captura y guarda los rostros de la persona en tiempo real. Una vez que este proceso ha terminado, el programa vuelve a pedir otro nombre al usuario.

Después de que el usuario decide salir del bucle, el programa inicia otro proceso para ejecutar la función train_model. Esta función no se define en el código proporcionado, pero su nombre sugiere que podría ser una función para entrenar un

modelo de aprendizaje automático con los rostros capturados y guardados anteriormente. Una vez que este proceso ha terminado, el programa termina.

```python
def capture_and_save(personName):
    personPath = dataPath + '/' + personName

    # Cargar el modelo pre-entrenado de FaceNet
    model = cv2.dnn.readNetFromTorch('openface.nn4.small2.v1.t7')

    if not os.path.exists(personPath):
        print('Carpeta Creada: ', personPath)
        os.makedirs(personPath)

    cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

    count = 0

    while True:
        ret, frame = cap.read()
        if ret == False:
            break

        frame = imutils.resize(frame, width=320)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        auxFrame = frame.copy()

        # Detectar rostros en la imagen
        face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
            rostro = auxFrame[y:y + h, x:x + w]
            rostro = cv2.resize(rostro, (96, 96))
            rostro_blob = cv2.dnn.blobFromImage(rostro, 1.0/255, (96, 96), (0, 0, 0), swapRB=True, crop=False)
            model.setInput(rostro_blob)
            vec = model.forward()
            np.save(personPath + '/rostro_{}.npy'.format(count), vec)
            count = count + 1
            print(count)

        cv2.imshow('frame', frame)

        k = cv2.waitKey(1)
        if k == 27 or count >= 300:
            break

    cap.release()
    cv2.destroyAllWindows()
```

Ilustración 8 Función de captura de imágenes

La función capture_and_save toma como parámetro el nombre de una persona y realiza las siguientes operaciones:

Primero, define la ruta del directorio donde se guardarán los archivos de la persona. Luego, carga el modelo pre-entrenado de FaceNet para la detección de rostros. Si el directorio no existe, lo crea.

Después, inicia la captura de video desde la cámara web. Para cada cuadro del video, redimensiona el cuadro y lo convierte a escala de grises. Detecta los rostros en el cuadro utilizando el clasificador de cascada Haar pre-entrenado de OpenCV.

Para cada rostro detectado, dibuja un rectángulo alrededor del rostro en el cuadro, extrae el rostro del cuadro, y redimensiona el rostro a 96x96 píxeles. Luego, convierte el rostro a un blob para la entrada del modelo y obtiene la representación vectorial del rostro alimentando el blob al modelo.

Guarda la representación vectorial en un archivo en el directorio de la persona. Muestra el cuadro con los rostros detectados. Si se presiona la tecla ESC o si se han guardado 300 rostros, termina la captura.

Finalmente, libera los recursos de la captura de video y cierra las ventanas de OpenCV. Esta función es útil para capturar y guardar rostros de una persona en tiempo real para su posterior análisis o reconocimiento.

```python
def train_model():
    peopleList = os.listdir(dataPath)
    print('Lista de Personas: ', peopleList)

    labels = []
    facesData = []
    label = 0

    for nameDir in peopleList:
        personPath = dataPath + '/' + nameDir
        print('Leyendo imagenes')

        for fileName in os.listdir(personPath):
            print('Rostros: ', nameDir + '/' + fileName)
            labels.append(label)

            vec = np.load(personPath + '/' + fileName)
            facesData.append(vec.flatten())

        label = label + 1

    cv2.destroyAllWindows()

    # Entrenar un clasificador SVM con los embeddings de FaceNet
    clf = svm.SVC(gamma='scale', probability=True)
    print("Entrenando...")
    clf.fit(facesData, np.array(labels))

    # Guardar el modelo entrenado
    with open('Models/ModeloFaceFrontalData2024.pkl', 'wb') as f:
        pickle.dump(clf, f)
    print('Modelo Guardado')

```

Ilustración 9 Función de entrenamiento

Este código define una función llamada train_model que entrena un modelo de clasificación de rostros utilizando un clasificador SVM (Support Vector Machine) y los embeddings de FaceNet. Aquí está la explicación en formato de párrafo:

La función train_model comienza listando todos los directorios en la ruta de datos especificada, que se supone que contienen los datos de los rostros capturados para diferentes personas. Cada directorio se asocia con una etiqueta numérica única, que se incrementa para cada persona.

Para cada persona, la función recorre todos los archivos en el directorio de esa persona. Cada archivo se supone que contiene un vector de características (o embedding) de un rostro, que se cargó previamente utilizando la función np.load. Este vector de características se aplana y se añade a la lista de datos de rostros, y la etiqueta correspondiente se añade a la lista de etiquetas.

Después de recoger todos los datos de los rostros y las etiquetas correspondientes, la función entrena un clasificador SVM con estos datos. El parámetro gamma='scale' y probability=True se utilizan para la configuración del clasificador.

Finalmente, el modelo entrenado se guarda en un archivo utilizando la biblioteca pickle. Esto permite que el modelo sea cargado y utilizado más tarde sin tener que ser reentrenado.

Por lo tanto, esta función es útil para entrenar un modelo de reconocimiento de rostros con los datos de rostros capturados y guardados previamente. Sin embargo, depende de que los datos de los rostros estén disponibles y correctamente formateados en la ruta de datos especificada. También depende de la disponibilidad de las bibliotecas os, cv2, svm, numpy y pickle.

Sistema de reconocimiento facial:

```python
import cv2
import os
import numpy as np
import pickle
```

Ilustración 10 Importaciones de reconocimiento

```python
imagePaths = os.listdir(dataPath)
print('imagePaths', imagePaths)

# Cargar el modelo pre-entrenado de FaceNet
model = cv2.dnn.readNetFromTorch('openface.nn4.small2.v1.t7')

# Cargar el clasificador SVM entrenado
with open('Models/ModeloFaceFrontalData2024.pkl', 'rb') as f:
    clf = pickle.load(f)

cap = cv2.VideoCapture(0, cv2.CAP_DSHOW)

faceClassif = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

```

Ilustración 11 Carga del modelo Pre-Entrenado

Primero, el código lista todos los archivos en la ruta de datos especificada y los imprime. Estos archivos se supone que contienen los datos de los rostros capturados previamente.

Luego, carga el modelo pre-entrenado de FaceNet utilizando la función cv2.dnn.readNetFromTorch. FaceNet es una red neuronal que se utiliza para generar embeddings (o vectores de características) de rostros, que luego se pueden utilizar para comparar y reconocer rostros.

Después, carga un clasificador SVM previamente entrenado desde un archivo utilizando la biblioteca pickle. Este clasificador se supone que fue entrenado con los embeddings de FaceNet de los rostros capturados, como se muestra en el código anterior.

A continuación, inicia la captura de video desde la cámara web utilizando la función cv2.VideoCapture.

Finalmente, carga el clasificador de cascada Haar pre-entrenado de OpenCV para la detección de rostros frontales. Este clasificador se utiliza para detectar rostros en las imágenes de video capturadas.

```python
while True:
    ret, frame = cap.read()
    if ret == False: break
    auxFrame = frame.copy()

    faces = faceClassif.detectMultiScale(frame, 1.3, 5)
    for (x, y, w, h) in faces:
        rostro = auxFrame[y:y+h, x:x+w]
        rostro = cv2.resize(rostro, (96, 96))
        rostro_blob = cv2.dnn.blobFromImage(rostro, 1.0/255, (96, 96), (0, 0, 0), swapRB=True, crop=False)
        model.setInput(rostro_blob)
        vec = model.forward()
        result = clf.predict([vec.flatten()])
        proba = clf.predict_proba([vec.flatten()])

        cv2.putText(frame, '{}'.format(result),(x, y-5), 1, 1.3, (255, 255, 0), 1, cv2.LINE_AA)

        # Si la probabilidad máxima es menor que el umbral, entonces la persona es "Desconocida"
        print(np.max(proba))
        if np.max(proba) < 0.5:
            cv2.putText(frame, 'Desconocido', (x,y-20), 2, 0.8, (0, 0, 255), 1, cv2.LINE_AA)
            cv2.rectangle(frame, (x,y), (x+w, y+h), (0, 0, 255), 2)
        else:
            cv2.putText(frame, '{}'.format(imagePaths[result[0]]), (x, y-25), 2, 1.1,(0, 255, 0), 1, cv2.LINE_AA)
            cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    cv2.imshow('frame', frame)
    k = cv2.waitKey(1)
    if k == 27:
        print(imagePaths[result[0]])
        break

cap.release()
cv2.destroyAllWindows()
```

Ilustración 12 Activación del reconocimiento

El código entra en un bucle infinito, capturando cuadros de video en tiempo real. Si no se puede capturar un cuadro, el bucle se rompe.

Para cada cuadro, el código detecta los rostros utilizando el clasificador de haarcascade cargado previamente. Para cada rostro detectado, el código extrae el

rostro del cuadro, lo redimensiona a 96x96 píxeles, y lo convierte a un blob para la entrada del modelo FaceNet. Luego, alimenta el blob al modelo y obtiene la representación vectorial del rostro.

El código luego utiliza el clasificador SVM cargado previamente para predecir la etiqueta del rostro, que se supone que corresponde a una persona en la lista de personas. También obtiene las probabilidades de cada etiqueta.

Si la probabilidad máxima es menor que 0.5, el código considera que el rostro es de una persona desconocida, y dibuja un rectángulo rojo alrededor del rostro y escribe 'Desconocido' encima del rectángulo. Si la probabilidad máxima es mayor o igual que 0.5, el código considera que el rostro es de la persona con la etiqueta predicha, y dibuja un rectángulo verde alrededor del rostro y escribe el nombre de la persona encima del rectángulo.

El código muestra el cuadro con los rostros detectados y sus etiquetas. Si se presiona la tecla ESC, el código imprime el nombre de la última persona reconocida, rompe el bucle, y libera los recursos de la captura de video y cierra las ventanas de OpenCV.

Anexo de Pruebas del funcionamiento del reconocimiento:

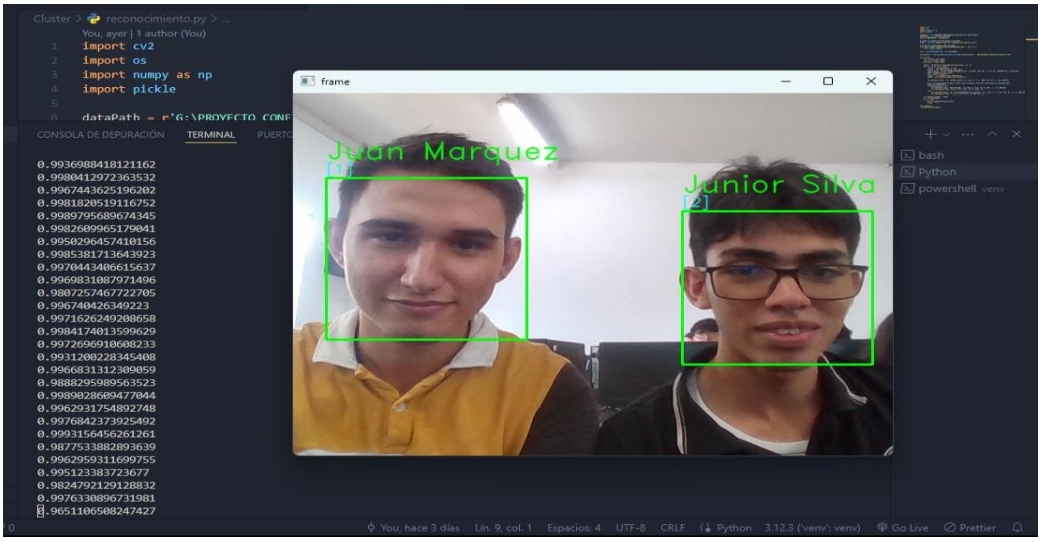

Ilustración 13 Prueba reconocimiento Facial

En la ilustración 14 se puede ver la interfaz en la parte del inicio de sesión del prototipo:

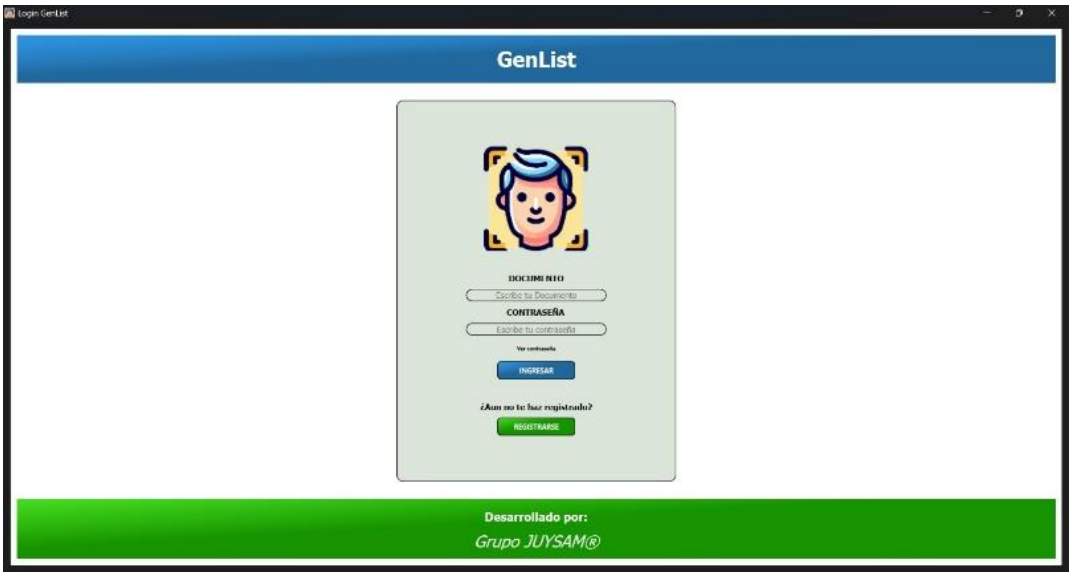

Ilustración 14 Interfaz sección login (Fuente: los autores)

En la ilustración 15 se puede ver la interfaz en la parte de las materias que el profesor vería:

Ilustración 15 Interfaz sección materias creadas (Fuente: los autores)

En la ilustración 16 se puede ver la interfaz en la parte del registro de cada estudiante capturando su biometría:

Ilustración 16 Interfaz sección registro de estudiantes (Fuente: los autores)

En la ilustración 17 se puede ver la interfaz en la parte de la creación de una clase donde aparecerían los estudiantes que se incorporaran a la clase:

Ilustración 17 Interfaz sección creación de clase (Fuente: los autores)

En la ilustración 18 se puede ver la interfaz en la parte de la creación de lista de asistencia de una clase donde cada estudiante con el botón de “agregar estudiante” se registraría en la asistencia:

Ilustración 18 Interfaz sección creación de lista de asistencia (Fuente: los autores)

En la ilustración 19 se puede ver como la aplicación genera el pdf de la lista de asistencia con todos los datos insertados automáticamente de los estudiantes y profesor:

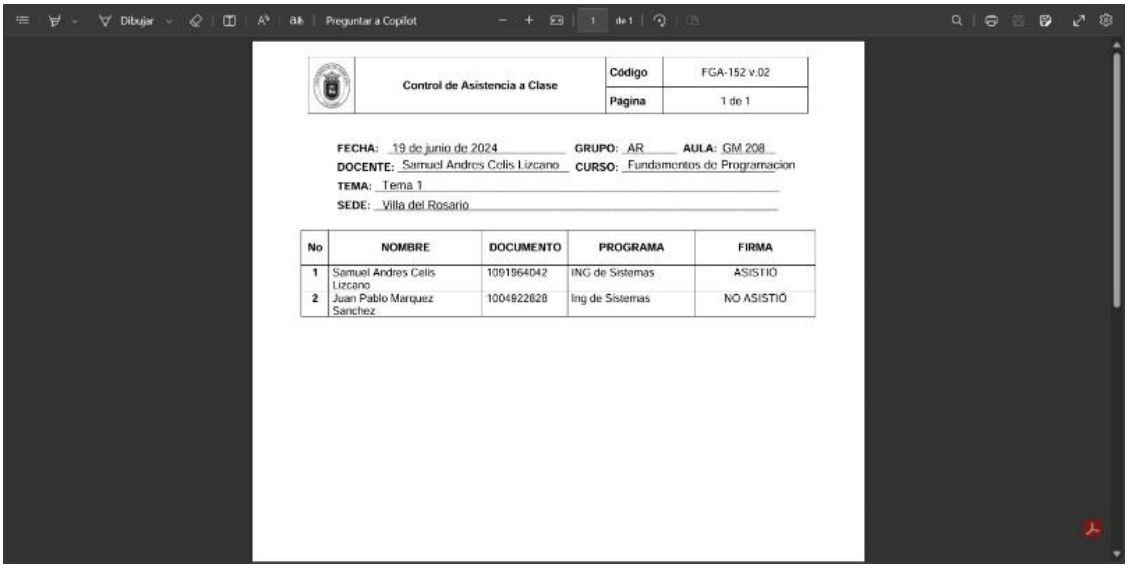

Ilustración 19 Interfaz sección creación de lista de asistencia (Fuente: los autores)

10. Conclusiones

La identificación de herramientas y tecnologías viables fue crucial para garantizar la eficiencia y precisión de un sistema de reconocimiento facial. Estas herramientas permitieron el desarrollo de algoritmos complejos de manera eficiente y flexible. La elección se alineó con los requisitos específicos del proyecto y las habilidades del equipo de desarrollo.

El mercado en general, exige en los profesionales actitudes creativas e innovadoras que permitan potenciar el desarrollo de las mismas (Claudia Marcela Durán Chinchilla, 2020).

La revisión de documentos y publicaciones relevantes permitió identificar lenguajes de programación y tecnologías que han demostrado ser eficaces en el desarrollo de sistemas de reconocimiento facial. Lenguajes como Python y C++ son frecuentemente mencionados debido a su flexibilidad y amplia gama de bibliotecas compatibles como OpenCV, TensorFlow y Keras. Según nuestro criterio es recomendable seleccionar Python como el lenguaje principal debido a sus extensas bibliotecas y la facilidad para prototipar y ajustar algoritmos de aprendizaje automático.

El desarrollo de un sistema de control de acceso basado en reconocimiento facial para un salón de clases es viable y puede mejorar significativamente la precisión y eficiencia de la supervisión. También es importante mencionar, el implementar medidas de seguridad para proteger los datos biométricos y garantizar la privacidad de los estudiantes.

Además, realizar pruebas exhaustivas en diferentes condiciones de iluminación y ángulos faciales puede asegurar la robustez del sistema. Según los resultados vistos en el sistema de reconocimiento facial, los modelos pre-entrandos de OpenCV, OpenFace y FaceNet requieren mayor robustez a la hora de la detección, dichos modelos pueden ser más precisos al implementar redes neuronales más complejas y estructuradas usando herramientas como TensorFlow.

11. Referencias

AWS. (29 de 5 de 2024). *AWS*. Obtenido de https://aws.amazon.com/es/what-is/python/

Bishnu Deo Kumar, H. A. (2023). Exam form automation using facial recognition. *Materials Today: Proceedings 80*, 2236–2240.

Claudia Marcela Durán Chinchilla, A. A. (2020). Aprendizaje activo e innovación en estudiantes de ingeniería. *Revista Colombiana de Tecnologías de Avanzada*, 127-135.

Colombia potencia de la vida. (21 de 8 de 1999). *Colombia potencia de la vida*. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=4276

Colombia potencia de la vida. (10 de 10 de 2012). *Colombia potencia de la vida*. Obtenido de https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=49981

Colombia presidencia de la republica. (2011). *Constitucion politica.* Obtenido de https://www.acnur.org/fileadmin/Documentos/BDL/2001/0219.pdf

Cornell University. (17 de 6 de 2015). *Cornell University*. Obtenido de https://arxiv.org/abs/1503.03832

Gnzo. (3 de 6 de 2024). *Gnzo*. Obtenido de https://ginzo.tech/como-funciona-metodologia-xp-desarrollo-

software/#:~:text=La%20metodolog%C3%ADa%20XP%20o%20Extreme,del%20c%C3%B3digo%20desarrollado%20y%20realimentaci%C3%B3n.

Gokulnath Anand, A. K. (2021). Object detection and position tracking in real time using Raspberry Pi. *Materials Today: Proceedings 47*, 3221-3226.

kaspersky. (3 de 6 de 2024). *latam kaspersky*. Obtenido de https://latam.kaspersky.com/resource-center/definitions/what-is-facial-recognition

Lateef, A. S. (2023). Facial Recognition Technology-Based Attendance Management System Application in Smart Classroom. *Iraqi Journal for Computer Science and Mathematics*, 136-158.

Mancuzo, G. (8 de agosto de 2020). *Compara Software*. Obtenido de Metodología XP: La Mejor Vía para el Desarrollo de Software: https://blog.comparasoftware.com/metodologia-xp/#Ciclo-de-un-proyecto-XP

OpenCV. (28 de 5 de 2024). *Open Source Computer Vision*. Obtenido de https://docs.opencv.org/4.x/d5/d26/tutorial_py_knn_understanding.html

pochteca papel. (26 de 5 de 2021). *pochteca papel*. Obtenido de https://tiendapapel.pochteca.com.mx/blog/como-se-fabrica-el-papel.html#:~:text=En%20general%2C%20todos%20los%20papeles,y%20extraer%20las%20fibras%20vegetales.&text=Esas%20fibras%20se%20transforman%20en,a%20las%20hojas%20de%20papel.

Riverbank Computing. (29 de 5 de 2024). *Riverbank Computing*. Obtenido de https://riverbankcomputing.com/software/pyqt/intro

Santiago Quilachamín Simbaña, F. D. (agosto de 2021). *ResearchGate*. Obtenido de Programacion Extrema XP: https://www.researchgate.net/figure/Figura-1-Metodologia-XP-Fuente-Muradas-2018_fig15_353826882

Silviu Răileanu, T. B. (2021). Open source machine vision platform for manufacturing and robotics. *IFAC PapersOnLine 54-1*, 522–527.

Universidad de Pamplona. (2023). *INEST.* Obtenido de https://aplicaciones.unipamplona.edu.co/unipamplona/inest/publico/index.jsp

Zhu Zhiguo, Y. C. (2020). Application of attitude tracking algorithm for face recognition based on OpenCV in the intelligent door lock. *Computer Communications 154*, 390–397.