WhosHere:
Sistema de Control de Asistencia con Reconocimiento Facial usando ESP32-CAM

Autor:
Ángel Sánchez Guerrero

Fecha:
9 de enero de 2025

Universidad de Granada

Grado en Ingeniería Informática

Tecnologías Emergentes

Índice general

Presentación

En este proyecto se presenta el diseño y desarrollo de WhosHere: un sistema de control de asistencia con reconocimiento facial y confirmación por código numérico. Está construido utilizando un ESP32-CAM como dispositivo para la captura de imágenes, junto con un teclado numérico. Además, incluye una aplicación web que permite administrar los usuarios, las sesiones y las asistencias.

A lo largo de este documento se describirán los detalles técnicos, el hardware utilizado y los desafíos encontrados durante el desarrollo de este sistema, destacando su potencial como una solución eficaz para el control de asistencia en diferentes contextos.

Capítulo 1

Necesidad o problema

En el contexto de mi universidad, el control de asistencia en las clases se realiza mediante un código alfanumérico que el profesor proporciona de forma presencial y que los estudiantes deben introducir en la plataforma Prado para registrar su asistencia. Aunque este sistema cuenta con ciertas medidas de seguridad, como la comprobación de que la dirección IP coincida con la red de la Universidad de Granada, existen comentarios entre los estudiantes que indican que este requisito puede ser fácilmente evadido mediante el uso de una VPN, lo que permite conectarse a la plataforma simulando estar dentro de la red universitaria. Este método abre la posibilidad de que los estudiantes falseen la asistencia enviando el código a amigos que no se encuentran físicamente en la clase. Ante esta idea, decidí intentar desarrollar un sistema de asistencia más robusto que reduzca la posibilidad de fraudes, utilizando reconocimiento facial.

Sin embargo, ya puedo adelantar que este sistema, en su estado actual, es fácilmente falsificable mostrando una fotografía de otra persona a la cámara. Si este proyecto se quisiera escalar a un nivel de producto real, sería necesario entrenar un modelo de reconocimiento facial más avanzado, capaz de detectar intentos de fraude, como cuando se presenta una foto mediante un móvil. La otra alternativa simplemente sería que mientras los alumnos pasan lista, haya un profesor vigilando el proceso para garantizar que se use de manera adecuada.

Capítulo 2

Propuesta básica de solución

El sistema propuesto se divide principalmente en dos partes, que se describen a continuación:

Aplicación web

La aplicación web es el núcleo del sistema y está completamente implementada en Python. Para gestionar la información clave del sistema, se utiliza una base de datos SQLite. SQLite es una base de datos ligera que no requiere un servidor separado para funcionar, ya que almacena los datos en un único archivo local. Esta base de datos se utiliza para almacenar la información de los alumnos, las sesiones y las asistencias, siendo estas últimas una combinación de los dos primeros elementos.

El backend de la aplicación está desarrollado con Flask, un framework minimalista y eficiente para la creación de aplicaciones web. Flask se encarga de gestionar las rutas del sistema, donde se implementa toda la lógica del proyecto. Entre estas rutas destacan aquellas que permiten la transmisión en tiempo real del video capturado del ESP32-CAM, el procesamiento de la detección de rostros y la interacción con la base de datos.

El reconocimiento facial se realiza mediante la librería DeepFace, que permite realizar comparaciones entre las imágenes capturadas por el ESP32-CAM y las referencias almacenadas para cada alumno. DeepFace utiliza modelos preentrenados de redes neuronales para analizar las características faciales y realizar verificaciones con una alta precisión. Sin embargo, en este prototipo, el reconocimiento se basa únicamente en una imagen de referencia por alumno, lo que lo hace vulnerable a errores.

ESP32-CAM

El ESP32-CAM es el dispositivo encargado de capturar el video en tiempo real y recopilar datos del teclado numérico conectado a él. Ejecuta un código basado en un ejemplo proporcionado por el fabricante, que configura un servidor web para exponer el feed de la cámara. Este código ha sido modificado para incluir la funcionalidad de capturar las teclas presionadas en un teclado numérico.

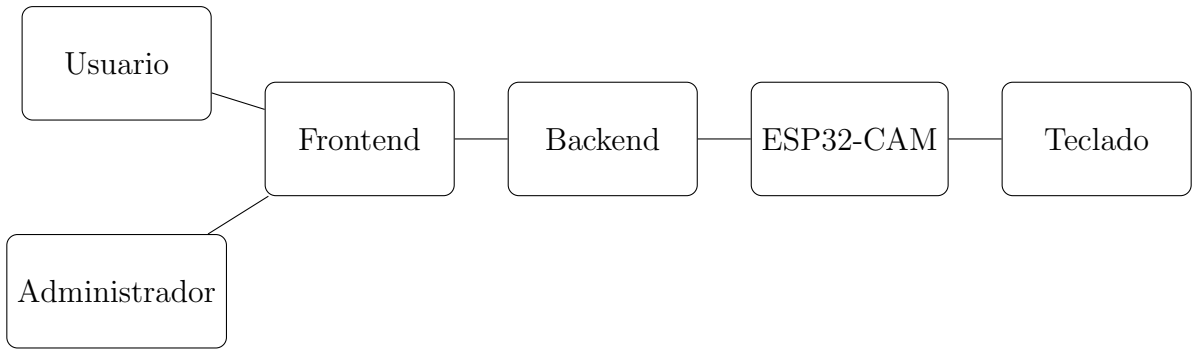

Figura 2.1: Diagrama de comunicación entre usuario, administrador, frontend, backend, ESP32-CAM y el teclado numérico.

La información de las teclas es procesada y enviada al backend, donde se combina con el reconocimiento facial para autenticar a los alumnos. La implementación del sistema de confirmación por código responde a varias motivaciones. Por un lado, incrementa la complejidad del proyecto, permitiendo usar más elementos del kit de Arduino. Por otro lado, compensa las limitaciones del reconocimiento facial basado en una sola imagen de referencia, reduciendo el riesgo de marcar como presente a alguien de forma equivocada.

En resumen, el ESP32-CAM no solo transmite el video, sino que también actúa como un intermediario capaz de capturar y enviar los PINs introducidos por los usuarios. Esto complementa el proceso de autenticación, proporcionando una capa adicional de verificación en el sistema.

Capítulo 3

Listado de componentes

El sistema desarrollado requiere tanto componentes de hardware como de software, los cuales se detallan a continuación:

Hardware

Los componentes de hardware utilizados en el proyecto son los siguientes:

- Ordenador: Utilizado como servidor para ejecutar la aplicación web y gestionar la base de datos.
- ESP32-CAM: Dispositivo principal encargado de capturar el video y enviar los datos al backend.
- Módulo USB para ESP32-CAM: Necesario para programar el ESP32-CAM y conectarlo al ordenador.
- Teclado numérico: Dispositivo que permite la entrada del código PIN por parte de los usuarios.

Software

En cuanto al software, se emplearon las siguientes herramientas y librerías:

- Python: Lenguaje principal en el que está desarrollada la aplicación web. En concreto se usa la version 3.11.
- SQLite: Base de datos ligera utilizada para almacenar los datos de los alumnos, sesiones y asistencias.
- Flask: Framework web para gestionar las rutas y la lógica del backend.
- DeepFace: Librería utilizada para implementar el reconocimiento facial mediante redes neuronales.

Capítulo 4

Esquemático

En este proyecto se utilizó un ESP32-CAM propio que ya tenía en casa, en lugar de emplear los materiales del laboratorio. Viene conectado por defecto a un módulo USB que permite proporcionarle corriente y cargar el código necesario para su funcionamiento.

Para conectar el teclado numérico, fue necesario separar el ESP32-CAM del módulo USB y volver a conectar todos los pines mediante cables, exceptuando los destinados al teclado. A diferencia del dispositivo disponible en el laboratorio, mi ESP32-CAM cuenta con menos pines utilizables, lo que imposibilitó conectar la matriz completa de 4x4 del teclado, que requiere 8 pines. Inicialmente, se intentó implementar una matriz 3x3 para los números, pero tras múltiples pruebas, el programa fallaba constantemente.

Finalmente, se descubrió que algunos de los pines asignados al teclado numérico causaban conflictos con el módulo USB, procesos internos del ESP32-CAM o el código base del servidor web de la cámara. Esto llevó a reducir la configuración del teclado a una matriz 2x2, utilizando los pines IO12, IO13, IO14 e IO15, que no presentaron conflictos. Con esta configuración más sencilla, el sistema funcionó correctamente.

Con esta configuración reducida, los números funcionales en el teclado quedaron limitados a 1, 2, 4 y 5. Por lo tanto, los códigos generados y aceptados por el sistema se componen exclusivamente de combinaciones de estos dígitos. Aunque esta restricción no estaba presente en el diseño original, fue una decisión necesaria debido a las limitaciones del ESP32-CAM propio utilizado en lugar de los dispositivos del laboratorio, que disponen de más pines y hubieran permitido la implementación completa de un teclado 4x4.

El esquemático que se presenta a continuación ilustra esta configuración, mostrando cómo se conectaron los componentes para garantizar el correcto funcionamiento del sistema.

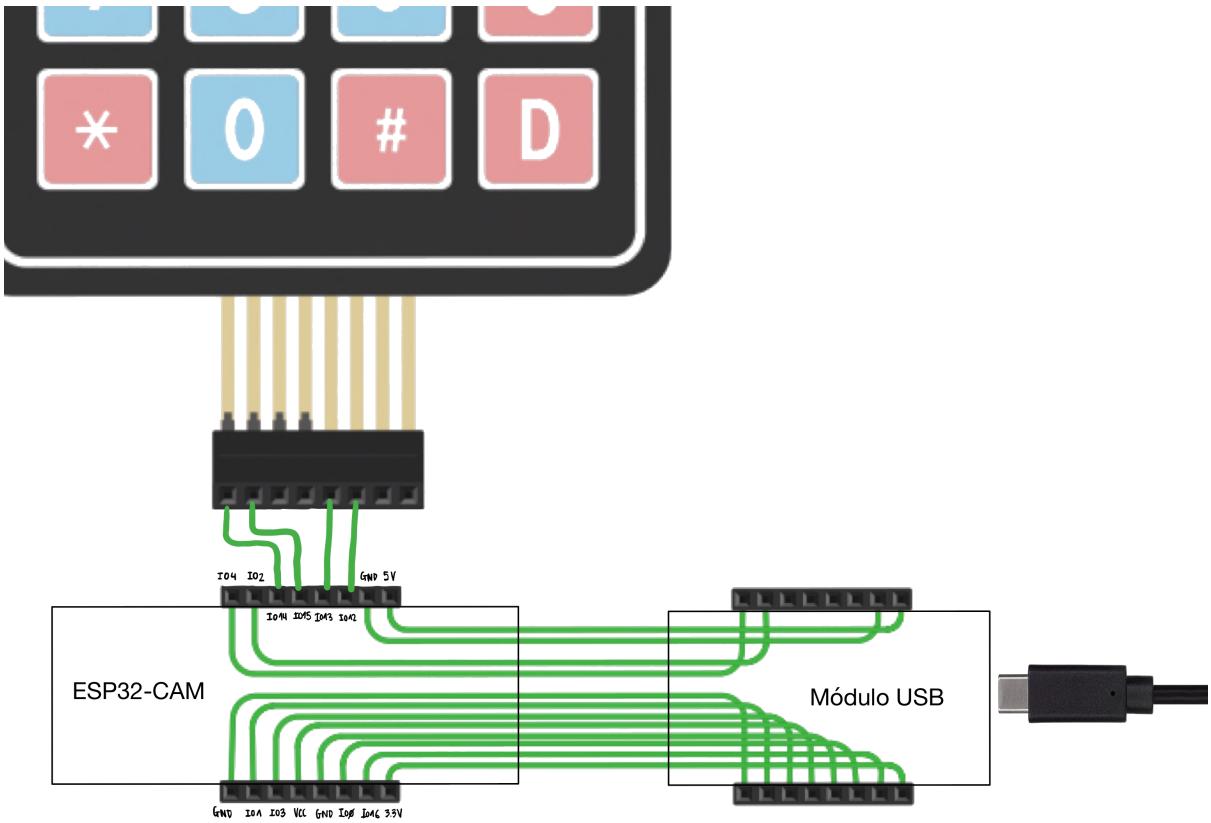

Figura 4.1: Esquemático del sistema con ESP32-CAM y teclado numérico.

Capítulo 5

Código

El desarrollo del sistema implicó la implementación y modificación de código tanto para el ESP32-CAM como para la aplicación web. A continuación, se describe la estructura general de ambos:

Código del ESP32-CAM

El código del ESP32-CAM se basa en el ejemplo proporcionado por la biblioteca ESP32, accesible en el Arduino IDE a través de: `Archivo >Ejemplos >ESP32 >Camera >CameraWebServer`. Este ejemplo incluye varios archivos clave:

- **CameraWebServer.ino**: Es el archivo principal del proyecto que inicializa la cámara, configura el servidor web y actúa como el punto de entrada del programa ESP32.
- **app_httpd.cpp**: Maneja el servidor web de la cámara ESP32, incluyendo el streaming de video y el control de la cámara.
- **camera_index.h**: Contiene el código HTML/CSS/JS comprimido de la interfaz web que se muestra en el navegador.
- **camera_pins.h**: Define qué pin del ESP32 se conecta a cada función de la cámara según el modelo se use.
- **ci.json**: Archivo de configuración que especifica cómo debe compilarse el código para diferentes modelos de ESP32.
- **partitions.csv**: Define cómo se divide la memoria flash del ESP32, especificando el tamaño y propósito de cada sección

En este proyecto, el archivo principal `CameraWebServer.ino` fue modificado para incluir la funcionalidad de capturar las teclas presionadas en el teclado numérico conectado al ESP32-CAM y enviarlas a la aplicación web mediante peticiones HTTP.

Código de la aplicación web

El código de la aplicación web se organiza en varios archivos, cada uno con una función específica:

- **app.py**: Núcleo del sistema. Contiene los endpoints del servidor Flask y la lógica principal para gestionar las solicitudes y registrar la asistencia.
- **cam.py**: Genera los frames de la cámara en tiempo real y, en cada frame, invoca la función de reconocimiento facial.
- **face_recognition.py**: Detecta las caras en los frames. Si un alumno ya ha pasado asistencia, no realiza ninguna acción. Si no, utiliza `socket.io` para solicitar el PIN correspondiente al alumno.
- **database.py**: Ofrece funciones para modificar la base de datos SQLite.

Notas adicionales

Por simplicidad, ya que es bastante extenso, el código completo no se incluye en este documento, pero está disponible públicamente en un repositorio de GitHub:

Enlace: github.com/angeloyo/WhosHere.

Además, el código incluye una funcionalidad adicional que genera un código QR único para cada alumno, permitiendo registrar asistencia mediante el escaneo del mismo. Aunque esta funcionalidad no se utilizó en este proyecto, se ha mantenido en el repositorio para futuros usos o para que otros puedan probarla.

Capítulo 6

Capturas de Pantalla

En esta sección se presentan capturas de pantalla de las diferentes interfaces del sistema para ilustrar su funcionalidad y diseño.

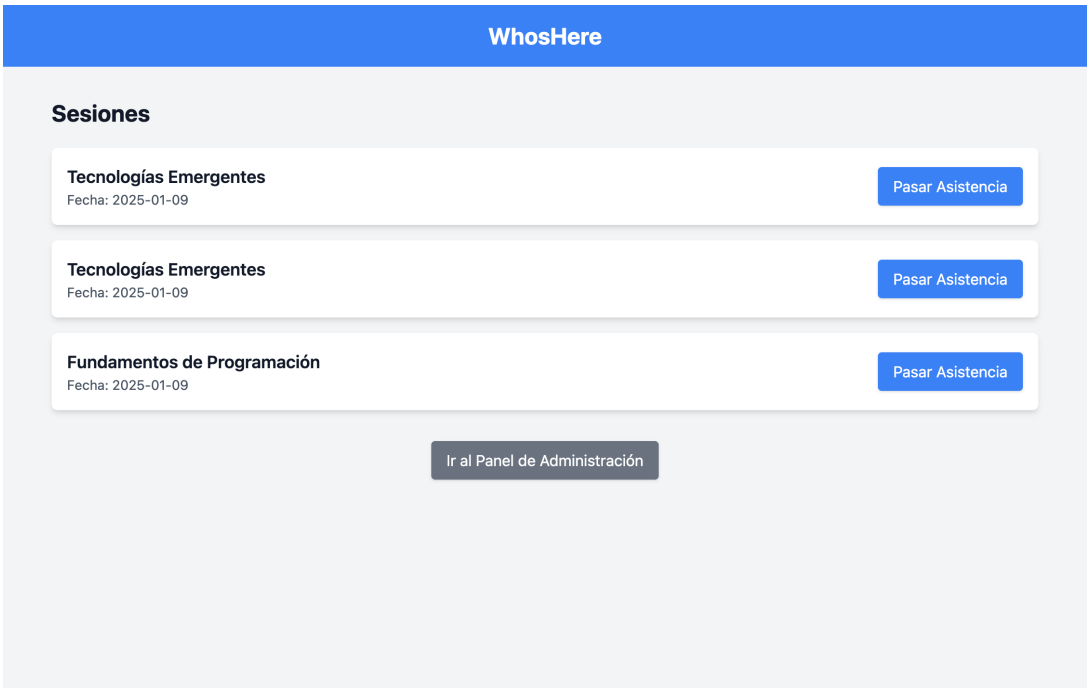

Figura 6.1: Página principal del sistema.

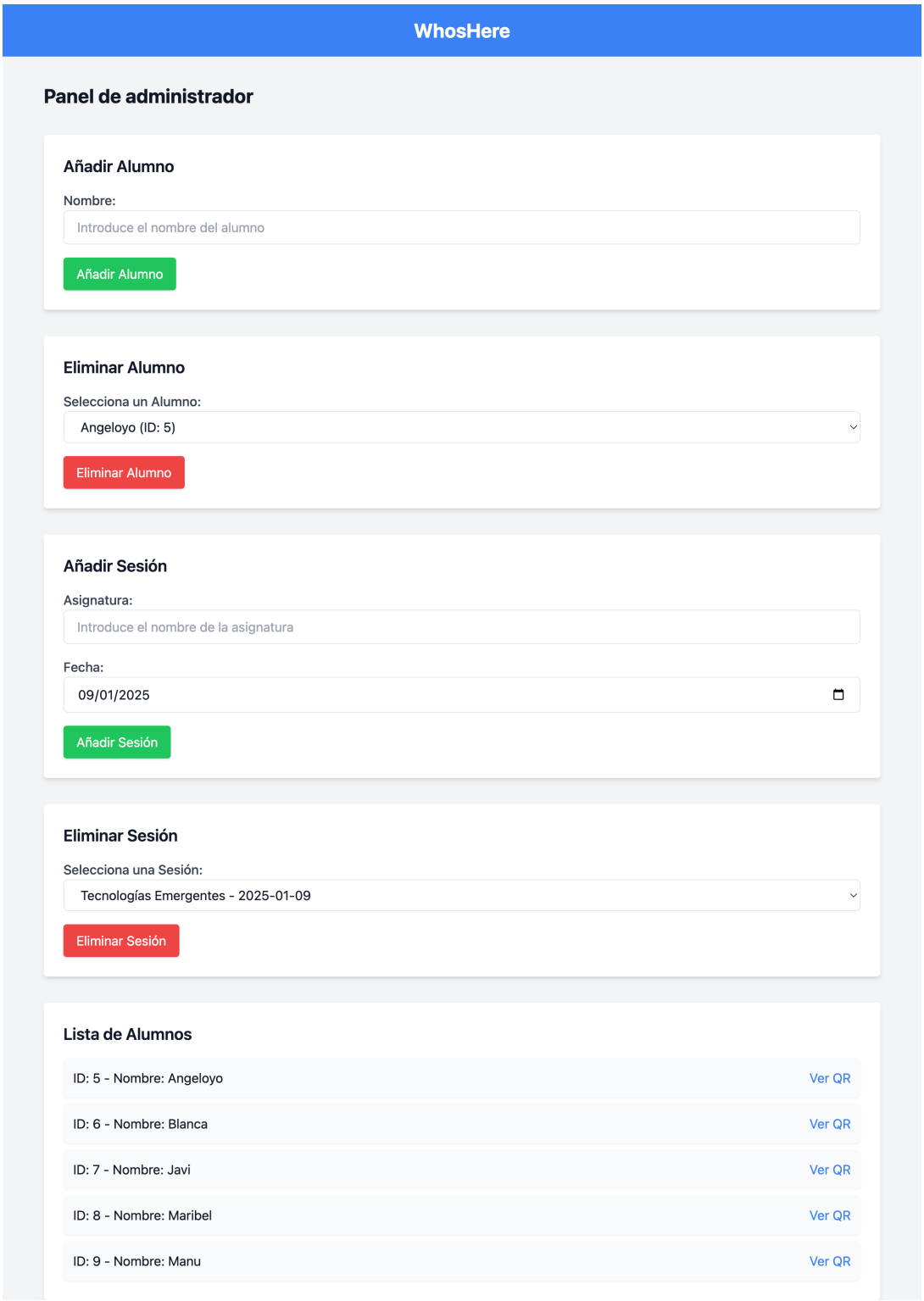

Figura 6.2: Panel de administrador.

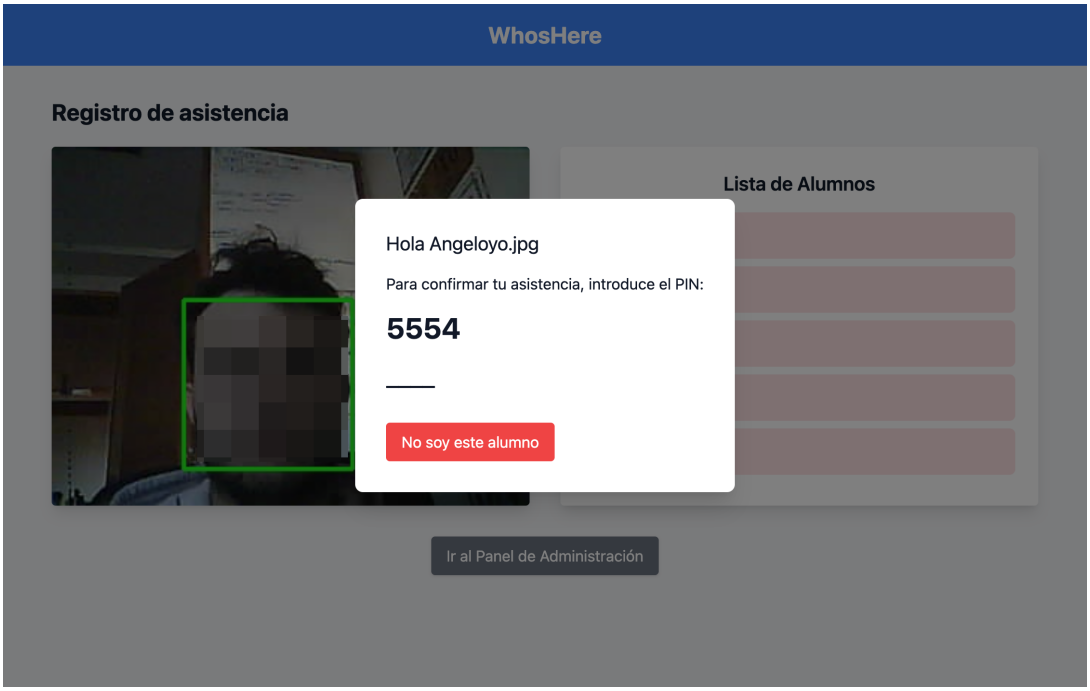

Figura 6.3: Página para pasar asistencia. Se muestra cómo se pide el PIN al usuario.

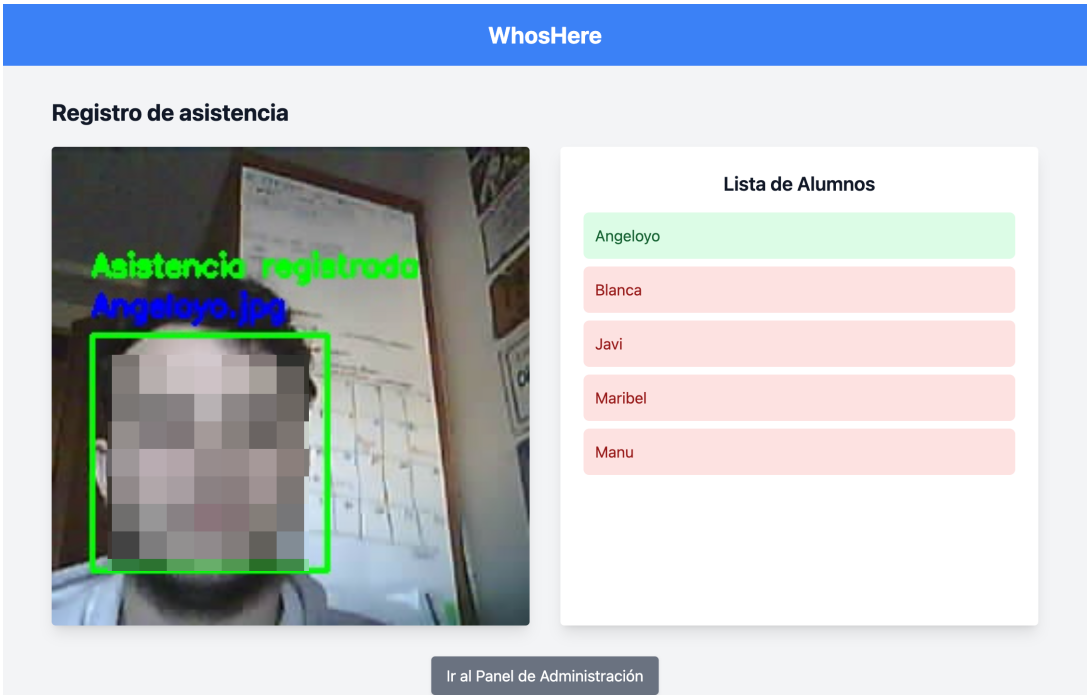

Figura 6.4: Página para pasar asistencia (II). Se muestra cómo se ha registrado la asistencia correctamente.

Capítulo 7

Conclusiones y posibles aplicaciones

A pesar de las dificultades encontradas durante el desarrollo, el sistema es completamente funcional y cumple con los objetivos planteados. Se ha logrado integrar un sistema de control de asistencia que combina reconocimiento facial y confirmación por PIN, utilizando componentes accesibles y tecnologías modernas.

Sin embargo, es importante destacar que este sistema es un prototipo. Para convertirlo en un producto real, sería necesario comenzar desde el principio estudiando y eligiendo un nuevo hardware, software y arquitectura. Además, en un entorno real, sería imprescindible garantizar el consentimiento de los alumnos para poder usar sus imágenes, respetando las normativas legales y éticas relacionadas con la privacidad.

En cuanto a sus posibles aplicaciones, las posibilidades son enormes. Universidades, empresas y cualquier tipo de organización podrían beneficiarse de un sistema de control de asistencia como este, ya que proporciona un método eficiente, seguro y moderno para gestionar la presencia de personas en diferentes entornos.

Bibliografía

[1] Arduino-ESP32. *CameraWebServer Example*. Recuperado de:
Enlace: github.com/arduino-esp32/CameraWebServer

[2] Flask. *Flask Documentation*. Recuperado de:
Enlace: flask.palletsprojects.com

[3] DeepFace. *DeepFace GitHub Repository*. Recuperado de:
Enlace: github.com/serengil/deepface

[4] SQLite. *About SQLite*. Recuperado de:
Enlace: sqlite.org

[5] OpenAI. *ChatGPT*. Recuperado de:
Enlace: chatgpt.com

[6] Cursor AI. *Cursor AI for Coding*. Recuperado de:
Enlace: cursor.com