Proyecto primer trimestre PIA

Contenido

1. Introducción

El presente documento describe el desarrollo de un sistema automático de **identificación personal basado en reconocimiento facial y reconocimiento de voz**, cuyo objetivo es permitir la detección de usuarios, su registro inicial y su posterior identificación de forma totalmente autónoma.

El sistema funciona combinando técnicas de **visión por computador** y **procesamiento de audio**. En una primera fase, el programa captura una imagen del usuario a través de la cámara del dispositivo y analiza el rostro empleando modelos de reconocimiento facial. Si el sistema identifica que la persona ya se encuentra registrada, se procede a saludarla utilizando sus datos guardados y se muestran tanto su **nombre** como su **DNI**, recuperados de la base de datos local.

En caso de que el rostro capturado no coincida con ningún usuario existente, el sistema activa automáticamente un **procedimiento guiado de registro**. Durante este proceso se captura una imagen del usuario, se le solicita verbalmente su nombre y su DNI mediante reconocimiento de voz, y finalmente se almacena la información junto con la representación facial en formatos CSV y JSON. De esta forma, el usuario quedará registrado para futuras ejecuciones, pudiendo ser identificado automáticamente.

El documento incluye la arquitectura interna del sistema, la metodología de trabajo aplicada, la configuración del entorno de desarrollo, las instrucciones de instalación, el manual de uso, los resultados obtenidos y las posibles mejoras futuras.

1.1. Objetivo del proyecto

El objetivo del proyecto es crear un sistema capaz de:

- **Detectar y reconocer rostros** mediante captura de imagen en tiempo real.
- **Identificar usuarios previamente registrados** utilizando datos almacenados.
- **Saludar e informar del nombre y DNI** cuando el usuario ya esté registrado.
- **Registrar nuevos usuarios** cuando la cara no coincida con ninguna existente.
- Capturando una fotografía
- Solicitando el nombre por voz
- Solicitando el DNI por voz
- Guardando toda la información en CSV y JSON
- Mantener una base de usuarios fiable y accesible para futuras identificaciones.
- Integrar reconocimiento visual y de voz en un flujo unificado y automatizado.

El proyecto busca demostrar cómo la combinación de visión artificial, procesamiento de voz y persistencia de datos permite desarrollar un sistema básico de identificación personal completamente funcional.

1.2. Alcance

Funcionalidades incluidas

- Captura de fotografía del usuario.
- Comparación del rostro con registros previos.
- Identificación automática de usuarios registrados.
- Saludo verbal y visual mostrando nombre y DNI.
- Registro guiado mediante voz si el usuario no existe.
- Almacenamiento en **CSV y JSON**.
- Gestión de ficheros de imágenes para reconocimiento facial.

2. Metodología de trabajo

La metodología de trabajo aplicada en este proyecto se basa en un enfoque ágil, tomando como referencia el marco **Scrum** para organizar las tareas en ciclos cortos y estructurados. Dado que se trata de un proyecto individual, el uso de Scrum se adapta a un contexto unipersonal, permitiendo una planificación clara y una ejecución progresiva orientada a entregables concretos.

El trabajo se ha dividido en **dos sprints de una semana**, donde el primero se orienta completamente a la preparación, documentación y construcción del entorno, y el segundo se dedica al desarrollo integral del sistema de reconocimiento facial y de voz.

2.1. Metodología adoptada (Scrum)

Para organizar el proyecto, se ha utilizado Scrum como guía metodológica, adaptándolo a un escenario de desarrollo individual. Scrum proporciona una estructura orientada a planificación, revisión continua y mejora incremental del producto.

Los elementos principales aplicados han sido:

- **División del proyecto en sprints cortos**, de duración fija (1 semana).
- **Enfoque incremental**, desarrollando primero la base documental y técnica, y después la funcionalidad.
- **Revisión del avance** al final de cada sprint.
- **Priorización del backlog** según dependencias técnicas y objetivos.
- **Flexibilidad para ajustes**, especialmente útil en proyectos técnicos donde pueden surgir imprevistos de configuración o compatibilidad.

2.2. Planificación temporal

El proyecto se estructura en dos sprints consecutivos de una semana cada uno. Cada sprint tiene un objetivo claramente definido y un entregable asociado.

2.2.1. Sprint 1 (1 semana)

Objetivo del sprint: Crear toda la estructura del proyecto, preparar la documentación inicial y dejar el entorno completamente configurado, incluyendo librerías, dependencias y base del proyecto.

Tareas principales:

- Elaboración completa de la documentación inicial:
  - Introducción
  - Metodología
  - Instalación del entorno
  - Arquitectura del proyecto
  - Índice y estructura formal del documento
- Creación del proyecto en el IDE (PyCharm).
- Configuración del entorno virtual `.venv`.
- Instalación de todas las librerías del proyecto.
- Verificación del entorno mediante scripts de prueba.
- Creación de la estructura de carpetas del proyecto.

Entregable del sprint:
Documentación inicial completa + Entorno del proyecto configurado y operativo.

2.2.2. Sprint 2 (1 semana)

Objetivo del sprint: Desarrollar completamente el sistema de reconocimiento facial, reconocimiento de voz e integración final entre ambos módulos, permitiendo registrar e identificar usuarios.

Tareas principales:

- Desarrollo del módulo de **reconocimiento facial** (detección, codificación, reconocimiento).
- Desarrollo del módulo de **reconocimiento de voz** (captura, transcripción, validación).
- Implementación del flujo lógico del sistema:
  - Si el usuario es reconocido por la cara → mostrar y anunciar nombre y DNI
  - Si NO es reconocido → activar proceso de registro:
    - Captura de imagen
    - Solicitud de nombre por voz
    - Solicitud de DNI por voz
- Implementación del almacenamiento en CSV y JSON.
- Manejo de excepciones y validaciones.
- Pruebas finales y refinamiento del sistema.
- Actualización final de la documentación técnica.

Entregable del sprint:
Sistema funcional completo: identificación y registro automático de usuarios utilizando voz y reconocimiento facial.

2.3. Roles y responsabilidades

Aunque Scrum define roles diferenciados, en este proyecto todas las responsabilidades han sido asumidas por mi. Aun así, se describen los roles para mantener coherencia metodológica:

- **Product Owner:** Define el alcance del sistema, prioriza el backlog y determina los objetivos de cada sprint.
- **Scrum Master:** Supervisa la correcta aplicación del marco de trabajo, la planificación, la revisión y la eliminación de impedimentos.
- **Developer:** Diseña, implementa y prueba todas las funcionalidades técnicas del proyecto, incluyendo reconocimiento de voz, visión por computador, registro y almacenamiento.

En este proyecto, **todos los roles son desempeñados por mi mismo**, manteniendo la lógica organizativa de Scrum.

3. Herramientas de desarrollo

Para el desarrollo del sistema se han utilizado herramientas profesionales orientadas a la programación en Python, la gestión de entornos, el control de versiones y la colaboración estructurada. La combinación de estas herramientas permite mantener un proyecto ordenado, fácil de mantener y reproducible en cualquier entorno.

3.1. IDE utilizado (PyCharm)

El desarrollo se ha realizado empleando **PyCharm**, un entorno de desarrollo integrado (IDE) especializado en Python y ampliamente utilizado en entornos profesionales. Su soporte avanzado para entornos virtuales, control de versiones y gestión de dependencias lo convierte en una herramienta adecuada para proyectos que combinan reconocimiento de voz, visión por computador y múltiples librerías externas.

3.1.1. Motivos de elección

Los motivos por los que se seleccionó PyCharm como IDE principal son los siguientes:

- **Soporte nativo para Python** y excelente integración con proyectos basados en módulos.
- **Gestión automática de entornos virtuales**, facilitando la instalación de librerías complejas como `dlib`, `face_recognition` y `opencv-python`.
- **Autocompletado inteligente** y herramientas de análisis de código que reducen errores durante el desarrollo.
- **Entornos de ejecución configurables**, lo cual facilita la prueba de funcionalidades de voz y vídeo.
- **Facilidad para estructurar proyectos grandes**, proporcionando paneles dedicados para archivos, dependencias y configuraciones.

PyCharm permite trabajar de manera más ordenada y eficiente, evitando problemas comunes en proyectos con múltiples dependencias externas.

3.1.2. Configuración recomendada

La configuración aplicada para este proyecto es la siguiente:

- **Crear proyecto nuevo en PyCharm** con entorno virtual automático.
- Instalar dependencias directamente desde la sección **Python Packages** o mediante archivo `requirements.txt`.
- Configurar la ejecución del script principal `main.py` como *Run Configuration*.

Esta configuración garantiza que el entorno virtual esté correctamente aislado y que todas las librerías utilizadas sean reproducibles en otro equipo.

3.2. Entorno virtual (.venv)

El proyecto utiliza un entorno virtual **creado automáticamente por PyCharm**, basado en `Virtualenv`.
Este entorno se genera dentro de la carpeta del proyecto bajo el nombre **`.venv`**, y contiene todas las dependencias necesarias para el correcto funcionamiento del sistema.

Características principales del entorno virtual:

- Entorno completamente **aislado** del sistema operativo.
- Instalación controlada de librerías mediante `pip`.
- Compatibilidad garantizada con versiones específicas (Python 3.11/3.12).
- Permite ejecutar el proyecto en cualquier equipo replicando únicamente el contenido del archivo `requirements.txt`.
- Evita conflictos entre librerías del sistema y las del proyecto.

El uso del entorno virtual asegura que las librerías críticas del proyecto —como `SpeechRecognition`, `opencv-python`, `dlib`, `face_recognition`, entre otras— funcionen correctamente sin afectar al sistema global.

3.3. Control de versiones (GitHub)

El control de versiones del proyecto se gestiona mediante **Git** y se aloja en **GitHub**, lo que permite mantener un historial claro de cambios, retroceder versiones en caso necesario y garantizar la trazabilidad del desarrollo.

El repositorio se ha organizado de forma estructurada en función de la metodología Scrum aplicada:

- **Ramas principales:**
  - `main` → versión estable del proyecto

    - `develop` → integración continua por sprint
    - Sprint1 → integración de tareas del sprint1
    - Sprint2 → integración de tareas del sprint2
- **Ramas adicionales por tareas:**
    - `Feature/crear_proyecto` → creación del proyecto en pycharm
    - `feature/voz` → implementación del reconocimiento de voz
    - `feature/vision` → implementación del reconocimiento facial
    - `feature/registro` → integración del flujo de registro y almacenamiento

Además, se ha organizado el avance por **sprints**, vinculando commits y pull requests a los objetivos definidos en la planificación.

El repositorio del proyecto puede consultarse en el siguiente enlace:

Repositorio del proyecto: ivan152005/PIA-Proyecto-Rec_voz_visual: Proyecto primer trimestre de reconocimiento facial y de voz

4. Instalación del entorno

La correcta instalación del entorno es fundamental para garantizar el funcionamiento del sistema de reconocimiento facial y de voz. Dado que el proyecto utiliza librerías que requieren compilación (como `dlib`) y componentes avanzados de visión por computador, es necesario preparar cuidadosamente el entorno antes de ejecutar el código.

En este apartado se detallan los requisitos previos, la configuración necesaria y el procedimiento recomendado para la instalación de todas las dependencias del proyecto.

4.1. Requisitos previos del sistema

Antes de instalar las librerías y ejecutar el proyecto, es necesario asegurar que el sistema cumple con una serie de requisitos técnicos fundamentales.

4.1.1. Python

El proyecto requiere una versión de Python compatible con las librerías de reconocimiento facial y de voz utilizadas.
Las versiones recomendadas y probadas para este sistema son:

- **Python 3.11** (recomendado)
- **Python 3.12** (compatible, aunque con algunas limitaciones en librerías avanzadas)

Importante:
Versiones superiores como **Python 3.13** no son compatibles con `dlib` ni con `face_recognition`, por lo que no deben utilizarse para este proyecto.

Requisitos adicionales relacionados con Python:

- Las librerías deben instalarse dentro del entorno virtual `.venv`.
- Es obligatorio utilizar un entorno aislado para evitar conflictos con otros paquetes del sistema.
- PyCharm crea automáticamente un entorno virtual compatible cuando se configura la opción *"New Environment using Virtualenv"*.

4.1.2. Librerias

El módulo de reconocimiento de voz del proyecto utiliza un conjunto de librerías especializadas que permiten capturar audio, procesarlo, transcribirlo y generar

respuestas por voz. A continuación se describen las librerías principales necesarias para su funcionamiento, junto con su instalación correspondiente.

A) Librerías del módulo de reconocimiento de voz

Estas son las librerías esenciales utilizadas para permitir la interacción mediante voz:

pyttsx3

Librería de síntesis de voz que permite al sistema generar respuestas habladas sin necesidad de conexión a internet.

Instalación:

```
pip install pyttsx3
```

SpeechRecognition

Es la librería principal encargada de capturar la entrada de audio a través del micrófono y convertirla en texto.
Ofrece compatibilidad con múltiples motores de reconocimiento.

Instalación:

```
pip install SpeechRecognition
```

PyAudio

Permite acceder al micrófono del sistema en tiempo real.
Es una dependencia fundamental para `SpeechRecognition`.

Nota: En Windows puede requerir instalación mediante wheel si falla la instalación estándar.

Instalación:

```
pip install pyaudio
```

B) Librerías del módulo de reconocimiento visual

El sistema de reconocimiento visual se basa en visión por computador y en técnicas de análisis facial. Para ello se emplean varias librerías avanzadas que permiten la captura de imagen, la detección de rostros y la comparación con rostros previamente registrados. Algunas de estas dependencias requieren instalación manual en Windows debido a su complejidad interna.

A continuación se describen las librerías principales utilizadas:

OpenCV (`opencv-python`)

Librería fundamental para visión artificial.
Permite:

- Acceder a la cámara
- Capturar imágenes
- Procesar y convertir fotogramas
- Manipular matrices de imagen

Es la base del módulo visual y gestiona toda la interacción con el flujo de vídeo.

Instalación:

```
pip install opencv-python
```

face_recognition

Librería de alto nivel basada en `dlib`, que simplifica el reconocimiento facial.

Permite:

- Detectar rostros en una imagen
- Obtener sus codificaciones
- Comparar un rostro nuevo con caras registradas
- Determinar si el usuario está ya en la base de datos

Esta librería abstrae gran parte del trabajo pesado de `dlib`, facilitando su uso.

Instalación:

```
pip install face_recognition
```

CMake

Requisito necesario para permitir la compilación interna de módulos que utiliza `dlib`.

Hay que descargar el archivo “cmake-4.2.0-windows-x86_64.msi”, para poder instalar dlib.

Instalación simple en el entorno:

```
$env:Path += ";C:\Program Files\CMake\bin"
```

Con esa ruta lo agregamos al entorno y con “cmake –version” comprobamos que está todo correcto.

Instalación recomendada (incluye PATH):
https://cmake.org/download/

Después de eso habrá que instalar:

https://visualstudio.microsoft.com/es/visual-cpp-build-tools/

Durante la instalación, selecciona:

- "Desktop development with C++"

Asegúrate de incluir:

- MSVC (Microsoft C++ compiler)
- Windows 10/11 SDK

Una vez hecho eso ya podremos ejecutar en nuestro entorno ya podremos instalar dlib.

dlib

Es la librería principal utilizada para:

- Codificación de rostros
- Comparación entre rostros
- Extracción de puntos faciales
- Modelos de reconocimiento facial de alta precisión

Instalación simple en el entorno:

pip install dlib

Boost

Librería de soporte utilizada por `dlib`.
En Windows debe instalarse manualmente descargando la versión precompilada:

Descarga:
https://www.boost.org/users/download/

Tras la descarga solo debe descomprimirse (por ejemplo en `C:\boost`); no requiere instalación manual adicional.

C) Instalación conjunta

Para instalar las librerias podrás hacer:

```
pip install -r requirements.txt
```

Sin embargo, las siguientes deben instalarse manualmente en Windows:

dlib

CMake (si la instalación desde pip no es suficiente)

Boost

5. Arquitectura del proyecto

La arquitectura del sistema se ha diseñado siguiendo un enfoque modular, lo que permite separar claramente la lógica de reconocimiento visual, reconocimiento de voz y almacenamiento de datos. Este diseño facilita la mantenibilidad, la escalabilidad y futuras ampliaciones del proyecto.
Cada módulo opera de forma independiente, pero se integran mediante un flujo lógico centralizado en el archivo **main.py**, que actúa como orquestador del sistema.

5.1. Estructura general de carpetas

La estructura del proyecto es la siguiente:

```
Proyecto/
|
├── main.py
├── reconocimiento_facial.py
├── asistente_voz.py
├── almacenamiento.py
|
├── data/
|   ├── usuarios.csv
|   ├── usuarios.json
|
├── imagenes/
|   ├── usuario_1.jpg
|   ├── usuario_2.jpg
|
├── .venv/
├── requirements.txt
└── README.md
```

Descripción general de carpetas:

- **/data/** → almacena los ficheros CSV y JSON donde se guarda la información de los usuarios.
- **/imagenes/** → almacena las fotografías asociadas a cada usuario registrado.
- **main.py** → punto de entrada del sistema y coordinador de todos los módulos.
- **reconocimiento_facial.py** → módulo encargado de la captura de imagen y reconocimiento del usuario.
- **asistente_voz.py** → módulo encargado de la interacción mediante voz.
- **almacenamiento.py** → módulo dedicado a guardar y consultar datos.

5.2. Descripción de cada módulo

5.2.1. Modulo de reconocimiento de voz

Este módulo gestiona todo el proceso de interacción sonora entre el sistema y el usuario. Se encarga de:

- Capturar audio desde el micrófono.
- Convertir voz a texto mediante SpeechRecognition.
- Generar voz mediante pyttsx3.
- Validar y devolver los datos recogidos (nombre y DNI).

Este módulo se usa tanto en el registro de nuevos usuarios como en la interacción con usuarios ya existentes.

5.2.2. Módulo de reconocimiento visual

Es el módulo base del sistema y utiliza **OpenCV**, **face_recognition** y **dlib** para:

- Acceder a la cámara del dispositivo.
- Capturar una fotografía del usuario.
- Detectar y codificar el rostro.
- Comparar la imagen capturada con todas las almacenadas.
- Determinar si el usuario existe o debe ser registrado.

El módulo devuelve un identificador que permite continuar el flujo en el **main.py**.

5.2.3. Módulo de almacenamiento (CSV/JSON)

Este módulo centraliza la persistencia de todos los datos, permitiendo:

- Guardar nombre y DNI en formato **CSV**.

- Crear una copia estructurada en **JSON**.
- Asociar cada usuario a su imagen en la carpeta correspondiente.
- Consultar usuarios existentes (por nombre o por codificación facial).

Todo el flujo de persistencia está separado para evitar acoplamientos con la lógica de reconocimiento.

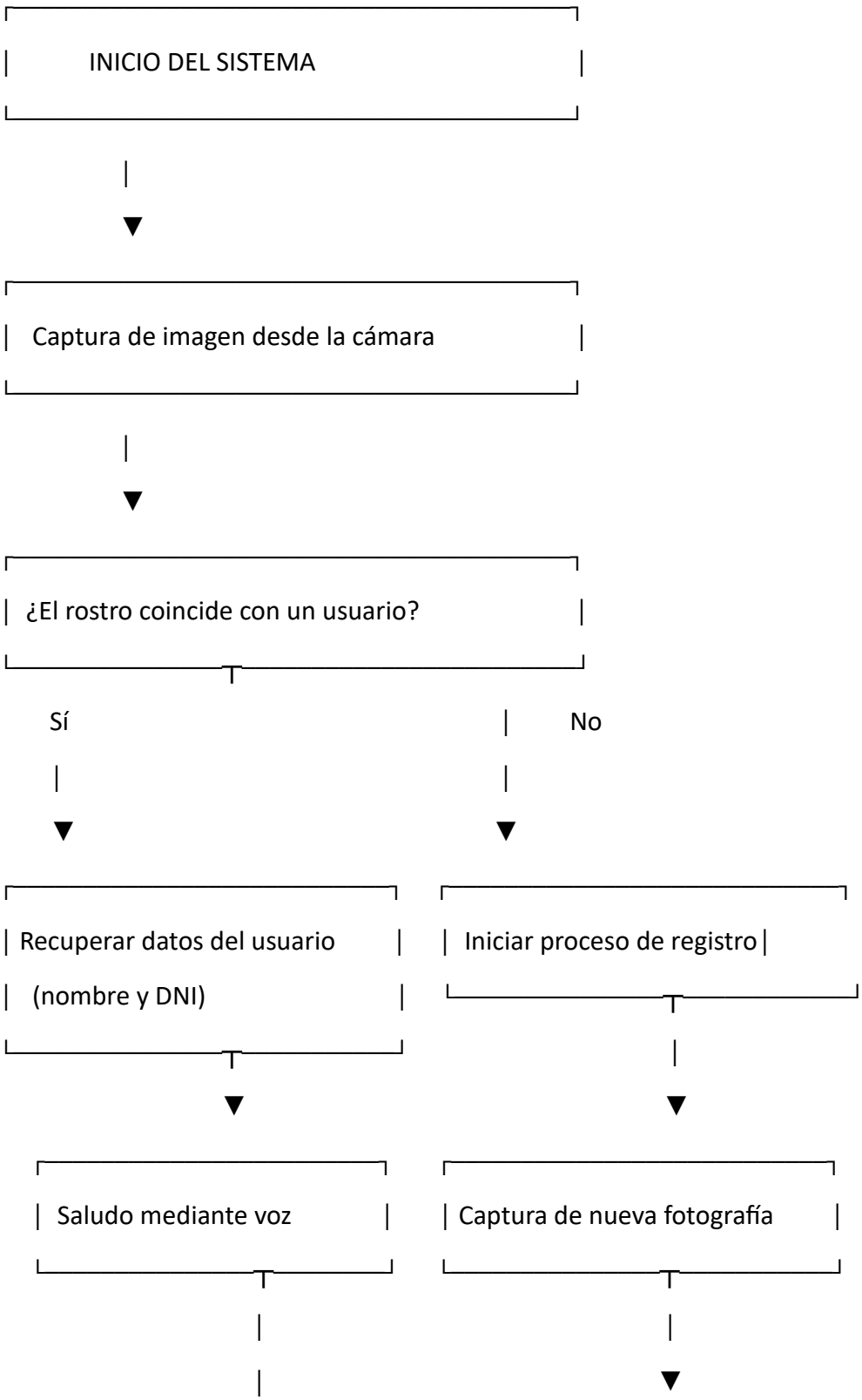
5.3. Flujo general del sistema

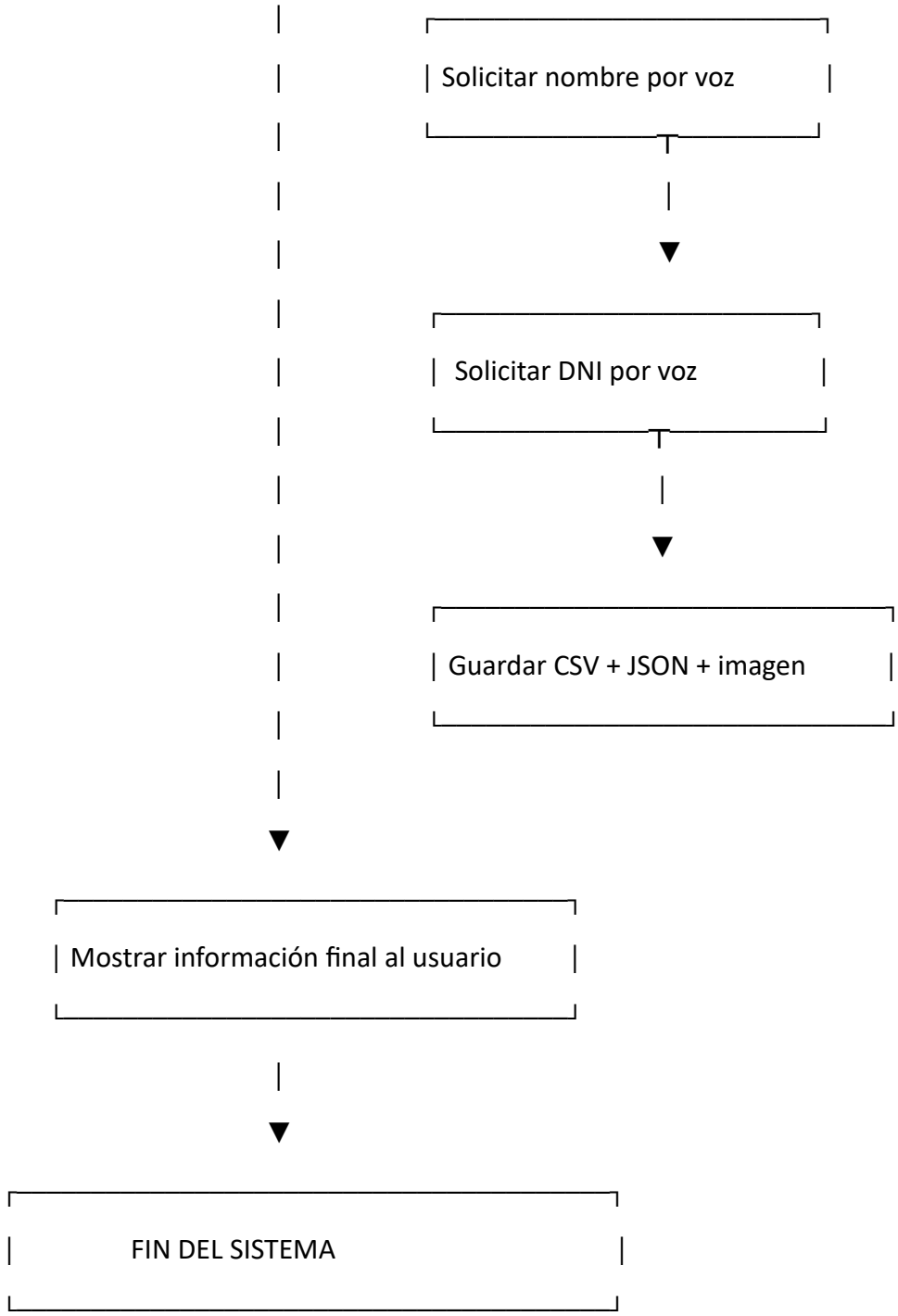
El funcionamiento del proyecto sigue el siguiente flujo:

1. El sistema abre la cámara y captura la imagen del usuario.
2. El módulo visual intenta identificar el rostro.
3. Si el usuario existe:
    - Se recuperan nombre y DNI.
    - El asistente de voz saluda e informa al usuario.
4. Si el usuario **no** existe:
    - Se toma una nueva fotografía.
    - El asistente de voz solicita nombre y DNI.
    - El sistema guarda los datos en CSV y JSON.
    - Se almacena la imagen asociada al nuevo registro.
5. El sistema finaliza mostrando un mensaje de confirmación.

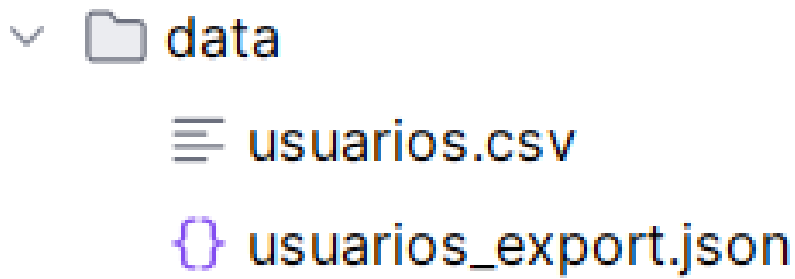
Este flujo combina visión artificial y reconocimiento de voz de forma coordinada.

5.4. Diagrama de funcionamiento




Solicitar nombre por voz
Solicitar DNI por voz
Guardar CSV + JSON + imagen
Mostrar información final al usuario
FIN DEL SISTEMA

6. Gestión y persistencia de datos



7. Manual de uso

Esta sección describe los pasos necesarios para que cualquier usuario final pueda instalar, ejecutar y utilizar correctamente el sistema de identificación mediante reconocimiento facial y de voz. Su objetivo es proporcionar una guía clara, estructurada y accesible, independientemente del nivel técnico del usuario.

7.1. Requisitos del usuario

Para utilizar el sistema correctamente, el usuario debe cumplir los siguientes requisitos mínimos:

Requisitos técnicos

- Ordenador con Windows 10 o superior.
- Cámara web funcional.
- Micrófono operativo (integrado o externo).
- Permisos para acceder a la cámara y al micrófono.
- Python instalado en el sistema (versión 3.11 recomendada).
- Capacidad para ejecutar aplicaciones desde consola o IDE.

Conocimientos mínimos

- Conocer cómo abrir un proyecto desde un IDE (PyCharm recomendado).
- Saber ejecutar un archivo `.py`.

7.2. Instalación del sistema

El proceso de instalación consiste en preparar todos los componentes necesarios para que el reconocimiento facial y de voz funcione correctamente.

1. Descargar el proyecto

Clonar o descargar el repositorio desde GitHub:

`git` clone `https://github.com/ivan152005/PIA-Proyecto-Rec_voz_visual`

2. Abrir el proyecto en PyCharm

Abrir PyCharm.

Seleccionar **Open** y cargar la carpeta del proyecto.

3. Activar el entorno virtual

El proyecto incluye un entorno `.venv`.
Si no se activa automáticamente:

Ir a *File → Settings → Project → Python Interpreter*.

Seleccionar el intérprete situado en `.venv`.

4. Instalar dependencias

Ejecutar en terminal dentro del proyecto y del entorno:

```
pip install -r requirements.txt
```

7.3. Cómo ejecutar el proyecto

El sistema se ejecuta desde el archivo principal del proyecto.

Método 1: desde PyCharm

1. Abrir `main.py`.
2. Pulsar **Run** ▶.
3. La consola mostrará los mensajes del sistema.

Método 2: desde terminal

Desde la raíz del proyecto:

```
python main.py
```

El sistema abrirá la cámara automáticamente e iniciará el proceso de identificación.

7.4. Explicación del funcionamiento paso a paso

1. Inicio del sistema

- El programa enciende la cámara.
- Captura una imagen del usuario.

2. Intento de reconocimiento

El sistema compara el rostro con la base de datos local (CSV + JSON).

CASO A — El usuario ya está registrado

- Su rostro coincide con un registro existente.
- Se recuperan **nombre** y **DNI**.
- El asistente de voz saluda:
  - "Hola [nombre], tu DNI es [DNI]."
- La información aparece también en consola.

CASO B — El usuario NO está registrado

El sistema inicia automáticamente el flujo de registro:

1. Captura una nueva fotografía.
2. Solicita el **nombre** por voz.
3. Solicita el **DNI** por voz.
4. Guarda:
    - Imagen del usuario
    - Datos en CSV
    - Datos en JSON
5. Confirma el registro:
    - “Usuario registrado correctamente.”

7.5. Casos de uso

Caso de uso 1 — Identificación automática

1. El usuario se coloca frente a la cámara.
2. El sistema detecta su rostro y lo identifica.
3. Se muestra y reproduce su información.

Caso de uso 2 — Registro de un nuevo usuario

1. Un rostro no registrado aparece frente a la cámara.
2. El sistema guía al usuario para completar registro mediante voz.
3. Se guarda su ficha completa.

Caso de uso 3 — Consulta manual de un usuario

1. El usuario ejecuta el comando de consulta de nombre.
2. Se devuelve su DNI desde los ficheros CSV/JSON.

7.6. Mensajes del sistema y resolución de problemas

1. “No se pudo acceder a la cámara.”

Causas:

Cámara ocupada por otra aplicación.

No tiene permisos.

Solución:

Cerrar apps como Zoom, Teams, Discord.

Revisar permisos de cámara en Windows.

2. “No se detecta ningún micrófono.”

Causas:

Micrófono desactivado.

Drivers no instalados.

Solución:

Activarlo desde Configuración → Sonido.

Probar otro micrófono.

3. “No se ha entendido el audio. Repite por favor.”

Causa: Ruido o mala pronunciación.
Solución: Repetir con mayor claridad.

4. Fallos instalando dlib

Causas comunes:

Falta CMake.

Falta Build Tools.

Python incompatible.

Solución:

Instalar CMake.

Instalar Build Tools.

Usar Python 3.11.

5. “No se ha encontrado ningún rostro en la imagen.”

Solución:

Asegurarse de estar centrado en la cámara.

Aumentar iluminación.