Documentación del Proceso de Investigación

En el desarrollo de este proyecto, investigamos diversas APIs y bibliotecas para implementar la funcionalidad de detección de personas en imágenes capturadas desde una cámara. A continuación, se detalla el proceso de investigación, las opciones consideradas y las razones por las que se eligieron las herramientas finales.

1. AForge.Video y AForge.Video.DirectShow

- **Motivo de Investigación**: Estas bibliotecas son ampliamente utilizadas para trabajar con dispositivos de captura de video, como cámaras web, en aplicaciones .NET. Ofrecen una API sencilla para acceder a cámaras conectadas al sistema y capturar fotogramas en tiempo real.
- **Razón de Elección**:Proporcionan una integración directa con Windows Forms, lo que facilita la captura de video y la actualización de controles gráficos como PictureBox. Su documentación y ejemplos disponibles permitieron una implementación rápida y eficiente.

Explicación de Cómo Funciona el Programa

El programa es una aplicación Windows Forms desarrollada en C# que permite capturar imágenes desde una cámara conectada al sistema y analizar dichas imágenes para detectar personas utilizando la API de Azure Computer Vision. A continuación, se detalla el funcionamiento del programa paso a paso:

---

1. Inicialización de la Cámara

- **Proceso**:
    - Al iniciar la aplicación, se ejecuta el método Form1_Load, que llama a InicializarCamara.
    - En InicializarCamara, se detectan las cámaras disponibles en el sistema utilizando la biblioteca AForge.Video.DirectShow.
    - Si se encuentra al menos una cámara, se selecciona la primera y se inicializa un objeto VideoCaptureDevice para capturar video en tiempo real.
    - Se suscribe un evento (NewFrame) que se activa cada vez que la cámara captura un nuevo fotograma.
- **Resultado**: La cámara comienza a capturar video en tiempo real, y los fotogramas se procesan en el método CapturarFrame.

---

2. Captura de Fotogramas

- **Proceso**:
    - En el evento NewFrame, el método CapturarFrame se ejecuta cada vez que se recibe un nuevo fotograma de la cámara.
    - El fotograma capturado se clona como un objeto Bitmap y se muestra en un control PictureBox en la interfaz gráfica.

- Si ya existía un fotograma anterior, se libera la memoria asociada para evitar fugas de memoria.

- **Resultado**: El usuario puede ver en tiempo real lo que la cámara está capturando en el control PictureBox.

---

3. Captura de Imagen

- **Proceso**:
  - Cuando el usuario hace clic en el botón "Capturar Imagen", se ejecuta el método btnCapturar_Click.
  - Si hay un fotograma disponible, se guarda como un archivo de imagen (captura_afrorge.jpg) en el disco.
  - Se actualiza un Label en la interfaz gráfica para informar al usuario que la imagen fue capturada correctamente.

- **Resultado**: La imagen capturada se guarda en el disco y está lista para ser analizada.

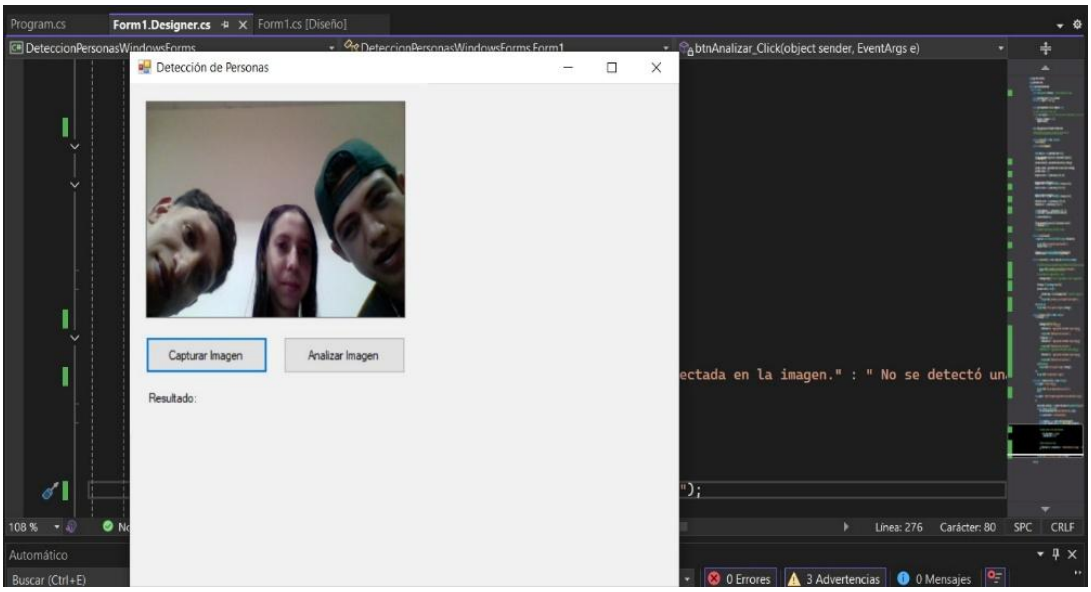
---

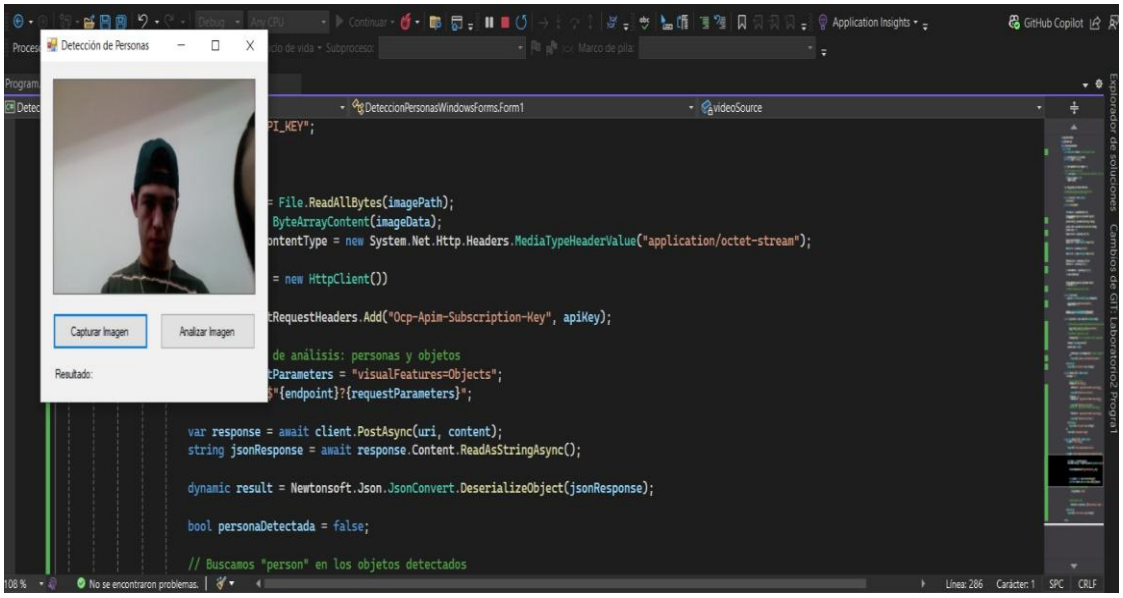
4. Análisis de Imagen

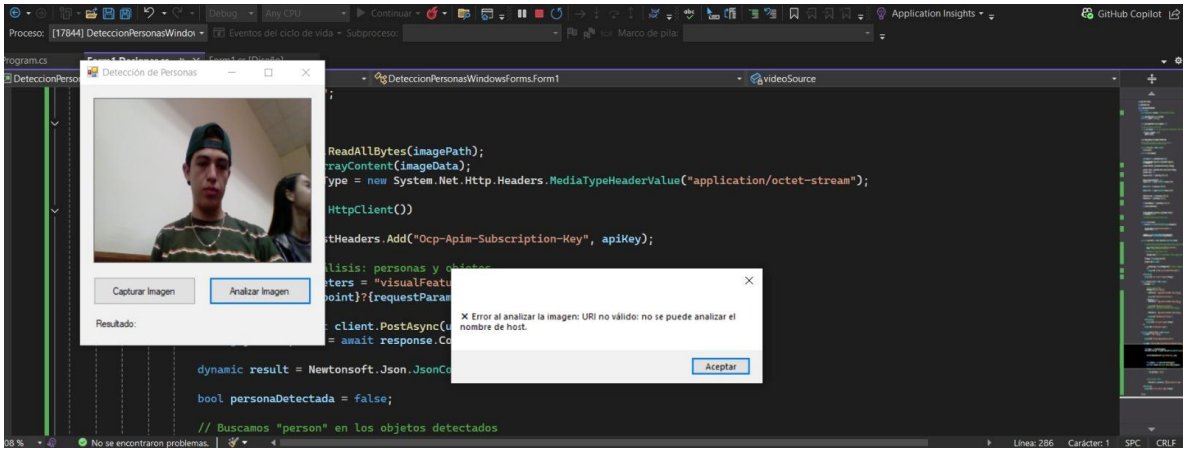
- **Proceso**:
  - Cuando el usuario hace clic en el botón "Analizar Imagen", se ejecuta el método btnAnalizar_Click.
  - El programa verifica si existe la imagen capturada en el disco.
  - La imagen se lee como un arreglo de bytes y se envía a la API de Azure Computer Vision mediante una solicitud HTTP POST.
  - La API analiza la imagen y devuelve un JSON con los objetos detectados.
  - El programa deserializa el JSON y busca objetos de tipo "person".
  - Si se detecta al menos una persona, se actualiza el Label para informar al usuario que se detectó una persona. De lo contrario, se indica que no se detectaron personas.

- **Resultado**: El usuario recibe un mensaje en la interfaz gráfica indicando si se detectaron personas en la imagen capturada

5. Capturas de pantalla:







Explicación de Cómo Funciona el Programa

El programa es una aplicación Windows Forms desarrollada en C# que permite capturar imágenes desde una cámara conectada al sistema y analizar dichas imágenes para detectar personas utilizando la API de Azure Computer Vision. A continuación, se detalla el funcionamiento del programa paso a paso:

---

1. Inicialización de la Cámara

- **Proceso**:
  - Al iniciar la aplicación, se ejecuta el método Form1_Load, que llama a InicializarCamara.
  - En InicializarCamara, se detectan las cámaras disponibles en el sistema utilizando la biblioteca AForge.Video.DirectShow.
  - Si se encuentra al menos una cámara, se selecciona la primera y se inicializa un objeto VideoCaptureDevice para capturar video en tiempo real.
  - Se suscribe un evento (NewFrame) que se activa cada vez que la cámara captura un nuevo fotograma.
- **Resultado**: La cámara comienza a capturar video en tiempo real, y los fotogramas se procesan en el método CapturarFrame.

2. Captura de Fotogramas

- **Proceso**:
  - En el evento NewFrame, el método CapturarFrame se ejecuta cada vez que se recibe un nuevo fotograma de la cámara.
  - El fotograma capturado se clona como un objeto Bitmap y se muestra en un control PictureBox en la interfaz gráfica.
  - Si ya existía un fotograma anterior, se libera la memoria asociada para evitar fugas de memoria.
- **Resultado**:
  - El usuario puede ver en tiempo real lo que la cámara está capturando en el control PictureBox.

---

3. Captura de Imagen

- **Proceso**:
  - Cuando el usuario hace clic en el botón "Capturar Imagen", se ejecuta el método btnCapturar_Click.
  - Si hay un fotograma disponible, se guarda como un archivo de imagen (captura_aforge.jpg) en el disco.
  - Se actualiza un Label en la interfaz gráfica para informar al usuario que la imagen fue capturada correctamente.

- **Resultado**:La imagen capturada se guarda en el disco y está lista para ser analizada.

---

4. Análisis de Imagen

- **Proceso**:
    - Cuando el usuario hace clic en el botón "Analizar Imagen", se ejecuta el método btnAnalizar_Click.
    - El programa verifica si existe la imagen capturada en el disco.
    - La imagen se lee como un arreglo de bytes y se envía a la API de Azure Computer Vision mediante una solicitud HTTP POST.
    - La API analiza la imagen y devuelve un JSON con los objetos detectados.
    - El programa deserializa el JSON y busca objetos de tipo "person".
    - Si se detecta al menos una persona, se actualiza el Label para informar al usuario que se detectó una persona. De lo contrario, se indica que no se detectaron personas.

- **Resultado**: El usuario recibe un mensaje en la interfaz gráfica indicando si se detectaron personas en la imagen capturada.

5. Interfaz Gráfica

- **Componentes Principales**:
    - **PictureBox**: Muestra en tiempo real el video capturado por la cámara y la imagen capturada.
    - **Button**:
        1. "Capturar Imagen": Permite al usuario guardar el fotograma actual como una imagen.
        2. "Analizar Imagen": Envía la imagen capturada a la API de Azure para su análisis.
    - **Label**: Muestra mensajes al usuario, como el estado de la captura o los resultados del análisis.

PARTES TRABAJADAS EN EQUIPO:

JEFERSON: Configuración de la Captura de Video

- Investigó y seleccionó la biblioteca **AForge.Video** para capturar video desde una cámara en tiempo real.
- Configuró la inicialización de la cámara, detectando dispositivos disponibles y seleccionando la cámara principal.
- Implementó la funcionalidad para capturar fotogramas en tiempo real y mostrarlos en la interfaz gráfica.

- Validó errores como la ausencia de cámaras disponibles o problemas al capturar fotogramas.

LEBISNON: 2: Captura y Almacenamiento de Imágenes

- Diseñó la interfaz gráfica, incluyendo el PictureBox para mostrar el video, botones para capturar y analizar imágenes, y etiquetas para mostrar resultados.
- Implementó la funcionalidad para capturar un fotograma específico desde el video en tiempo real y guardarlo como una imagen en el disco.
- Validó errores relacionados con el almacenamiento de imágenes, como la falta de un fotograma capturado o problemas al guardar el archivo.

LITSI: Análisis de Imágenes con Azure Computer Vision

- Investigó y configuró la integración con la API de **Azure Computer Vision** para analizar imágenes y detectar personas.
- Implementó la funcionalidad para enviar imágenes capturadas a la API y procesar la respuesta JSON.
- Mostró los resultados del análisis en la interfaz gráfica, indicando si se detectaron personas en la imagen.
- Validó errores como la falta de una imagen capturada o problemas de conexión con la API.