## Capítulo 3. Diseño del Sistema

En el presente capítulo se describe el proceso, módulos, arquitectura y principales controles utilizados, para el desarrollo de un prototipo con la finalidad de comprobar la hipótesis y objetivos planteados en el capítulo 1.

Para el desarrollo del prototipo se seleccionó el lenguaje C# de la plataforma ASP.Net para escritorio.

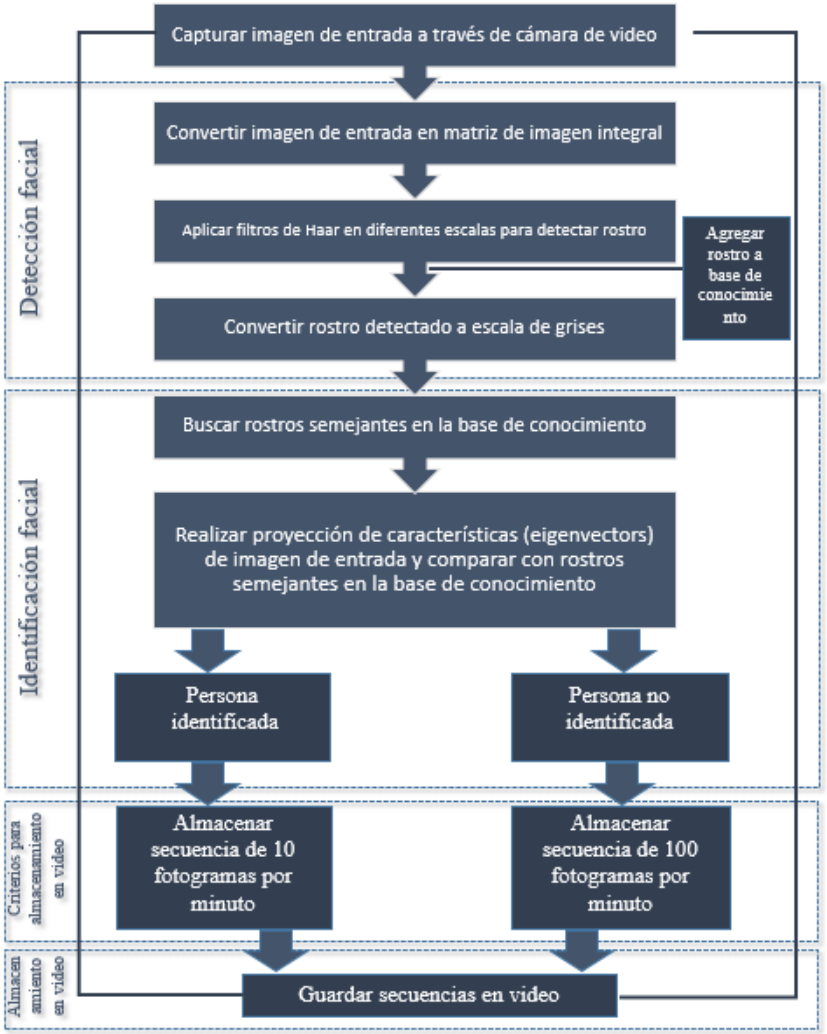



Figura 3. Diagrama funcional del prototipo.

La Figura 3 muestra el diagrama funcional del sistema, en las secciones siguientes se explica el funcionamiento de los módulos indicados en el diagrama.

### 3.1 Diagramas UML del Sistema

A continuación se muestran los diagramas UML que describen de manera gráfica el comportamiento e interacción del usuario con el sistema.

#### 3.1.1 Caso de Uso

Para iniciar la ejecución del programa, el usuario debe encender el sistema, una vez hecho esto, se ejecutarán los procesos descritos en el Diagrama 7. Para terminar la ejecución del programa, el usuario deberá apagar el sistema.

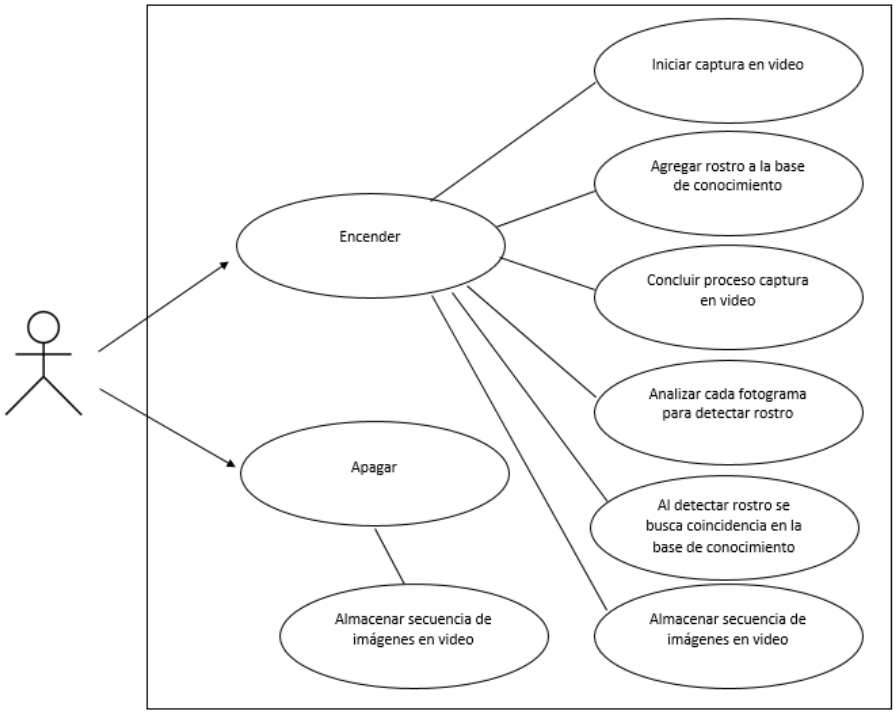



Diagrama 7. Caso de uso del prototipo.

### 3.1.2 Diagrama de Actividades

El Diagrama 8 ilustra el esquema secuencial de las actividades, que se ejecutan durante el proceso de optimización de almacenamiento de video.

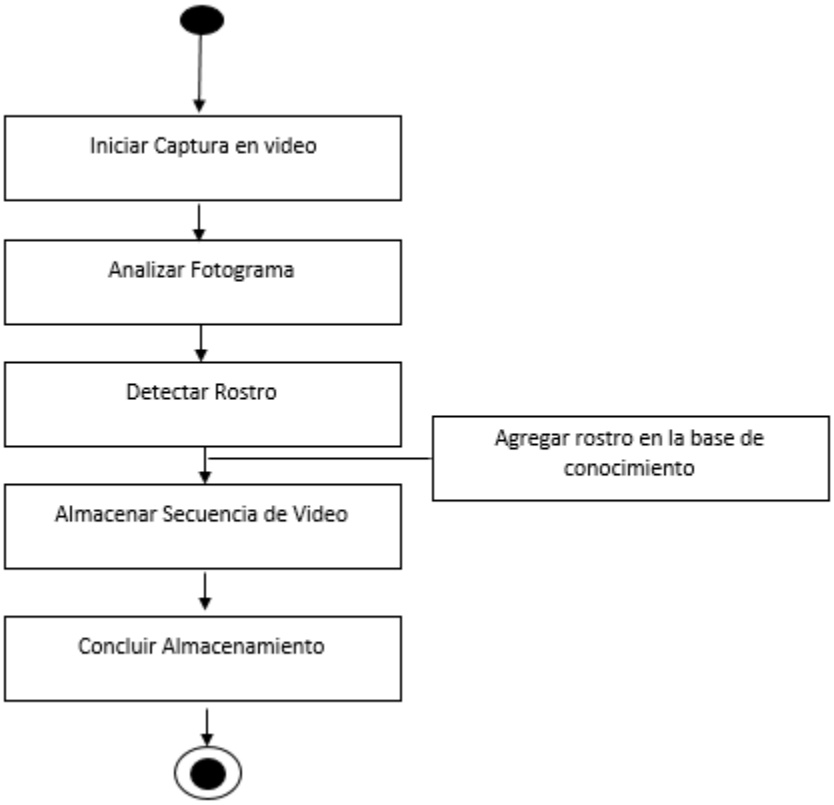



Diagrama 8. Diagrama de actividades.

### 3.1.3 Diagrama de Estado

El Diagrama 9 indica las transformaciones que se aplica a cada fotograma de entrada para lograr la detección e identificación de un rostro.

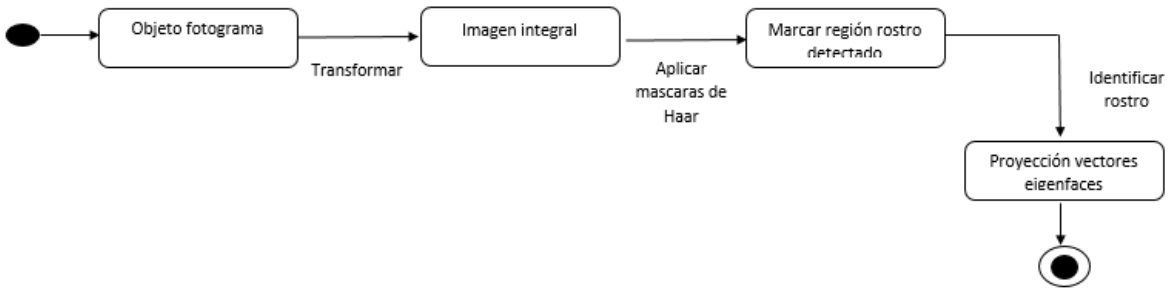
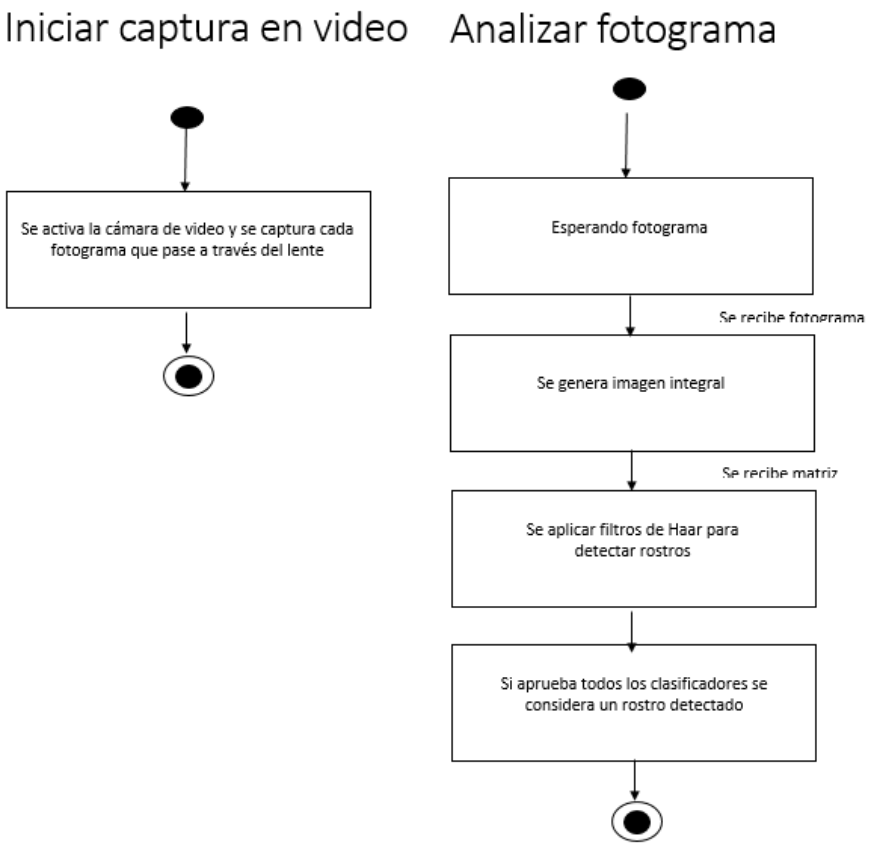



Diagrama 9. Diagrama de estado

### 3.1.4 Diagramas de Secuencia

A continuación se describen los diagramas de secuencia donde se indican los procesos que se ejecutan al encender el sistema (Diagrama 10).



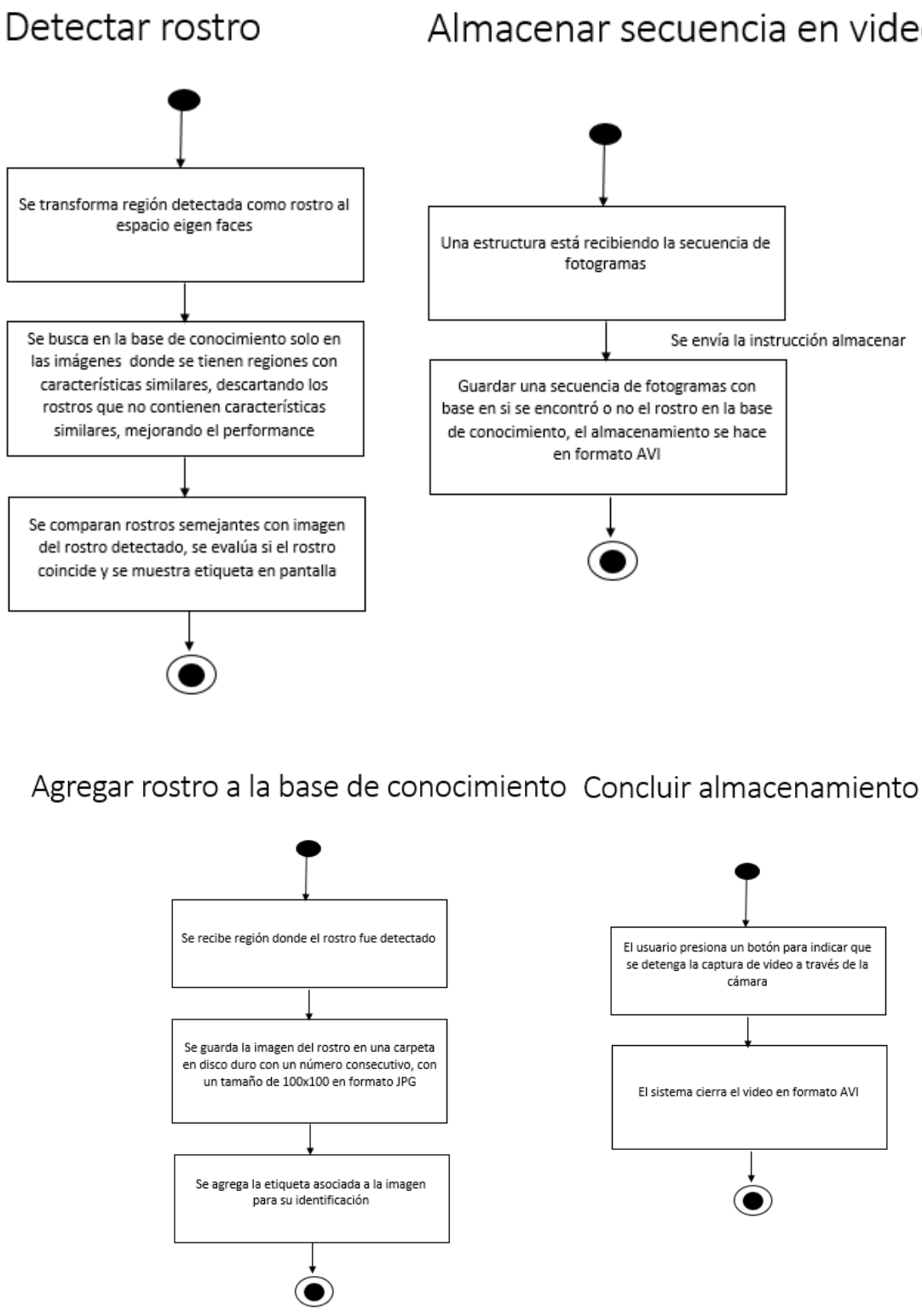



Diagrama 10. Diagramas de secuencia del sistema

### 3.2 Detección Facial

La primera fase estuvo enfocada a la detección de rostros en un fotograma, para llevar a cabo esto, se agregó la librería OpenCV al proyecto y se utilizó el objeto *ImageBox* para mostrar todos los fotogramas que se capturan a través de la cámara (Figura 4).

Figura 4. Objeto *ImageBox* que muestra cada fotograma capturado a través de la cámara.

Para realizar la detección del rostro, sobre el fotograma se realiza una transformación para convertir la imagen de entrada en la matriz de imagen integral, sobre esta matriz se aplicarán las operaciones con los filtros de Haar utilizando diferentes escalas, cada clasificador está entrenado para detectar rostros, si uno de los clasificadores indica que no hay un rostro en el fotograma este es rápidamente desechado y se interrumpe el proceso en cascada y se pasa al siguiente fotograma de entrada donde se repite el proceso, solo si el fotograma aprueba todos los clasificadores, es decir, todos los clasificadores detectaron un rostro entonces existe una región con un rostro en el fotograma. El concepto de imagen integral permite que las operaciones se hagan muy rápido y la detección se realice en tiempo real.


Figura 5. Detección de rostro en imagen de entrada.

### 3.3 Agregar Rostro a Base de Conocimiento

Una vez detectado un rostro, para agregarlo a la base de conocimiento se presiona el botón agregar y se almacena en disco, la región del rostro identificado con características de 100 pixeles de ancho por 100 pixeles de alto, el campo identificador es una etiqueta de texto que se asocia a la imagen agregada (Figura 6).

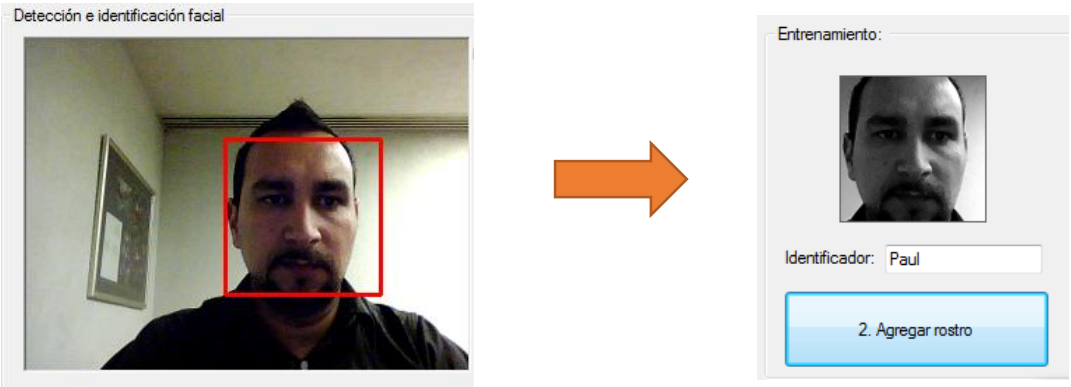



Figura 6. Detección y botón para agregar un rostro en la base de conocimiento.

### 3.4 Identificación Facial

La técnica PCA (también llamada *eigenfaces*) a partir de subespacios, se considera una de las de mayor rendimiento y funciona proyectando las imágenes faciales, sobre un espacio de facciones que engloba las facciones significativas (*eigenvectors*). La proyección de un rostro es la suma de los diferentes pesos de todas las facciones.

Para el primer prototipo, se desarrolló un método de identificación facial secuencial, el cual, su funcionamiento tenía tiempos de procesamiento muy altos ya que se tomaba como imagen de entrada el fotograma donde se detectó un rostro, se calculaba su proyección (*eigenvectors*) y se hacía una comparación de los pesos de cada una de las imágenes que se encontraban en la base de conocimiento, esta implementación disminuía notablemente el rendimiento al momento de identificar un rostro, por lo que, se implementó una estrategia para la identificación facial basada en *eigenfaces* y la distancia euclidiana (ec. 4) la cual busca objetos similares con el objetivo de encontrar solo las imágenes que tienen características parecidas, logrando reducir notablemente el tiempo de procesamiento al comparar solo las imágenes con características semejantes, evitando de esta manera, la comparación secuencial de la imagen de entrada con cada una de las imágenes en la base de conocimiento.

$$dE(P_1, P_2) = \sqrt{(x_2 - x_1)^2 - (y_2 - x_1)^2} \qquad (4)$$

La Figura 7 muestra como la interfaz, indica visualmente la detección facial dibujando un cuadro rojo en la región que se detectó un rostro, utilizando el algoritmo de reconocimiento facial *eigenfaces* nos permite saber si el rostro se encuentra en la base de conocimiento, si el rostro es conocido, se muestra la etiqueta que identifica el rostro en la parte superior del cuadro dibujado.


Figura 7. Identificación facial utilizando e*igenfaces*.

### 3.5 Criterios Para Almacenamiento de Secuencia de Video

El siguiente paso, es almacenar una secuencia en video con base a la detección facial. La Figura 8 muestra el diagrama que indica los criterios que sigue el sistema, para definir la secuencia en video que se almacenará en disco.

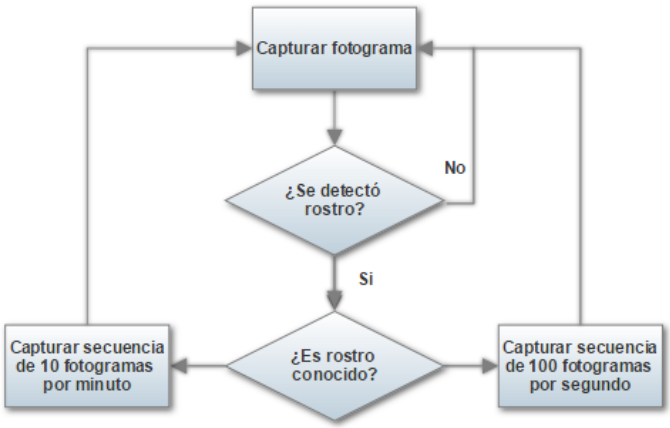



Figura 8. Diagrama de flujo Criterios de Almacenamiento

### 3.6 Almacenamiento en Video

El objetivo de esta etapa es almacenar el video en el disco duro, para este proceso se utiliza la librería *AForge.NET* la cual almacena los fotogramas en formato AVI.

Durante esta etapa se almacenan dos videos:

1. El video uno, almacena fotogramas con base a los criterios de almacenamiento propuestos.
2. El video dos, almacena todos los fotogramas que se capturan a través de la cámara.

Almacenar dos videos permitirá obtener resultados que permitirán comparar y evaluar el rendimiento de los criterios de almacenamiento en la etapa de pruebas del prototipo.

La Figura 9 está dividida en dos partes, la primera del lado izquierdo muestra visualmente secuencia e indicador de almacenamiento, la segunda del lado derecho muestra cada fotograma que se captura (CCTV).

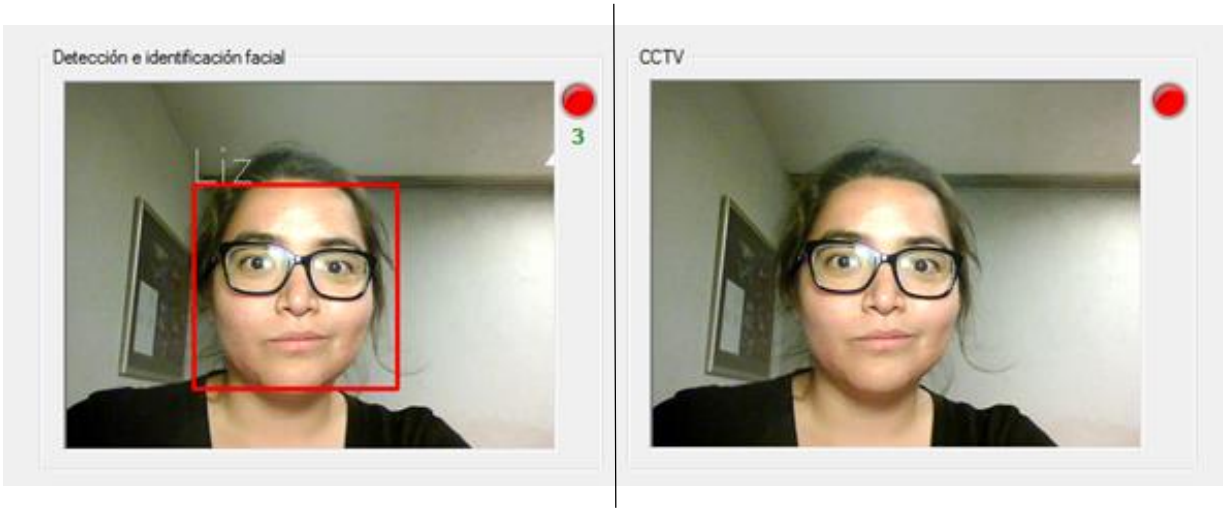



Figura 9. Imagen del prototipo.

Finalmente, con el objetivo de documentar, construir y realizar especificaciones del sistema, se utilizó el lenguaje UML, lo que permitió modelar y mostrar la interacción del usuario con

el prototipo desarrollado. Esta información también fue útil, para elaborar las pruebas de caja blanca que se anexaron en el apéndice del documento, las cuales permitieron generar los caminos de ejecución para evaluar el correcto funcionamiento de los diferentes módulos del sistema.

Como pudimos observar a lo largo del capítulo, se describieron los diferentes elementos de la interfaz de usuario, con el objetivo de identificar cada componente visual del sistema.