

# Detección de glaucoma de ángulo cerrado

Ignacio Martín Requena

## Resumen

**Palabras clave:** *visión por computador, programación en Android, OpenCV, análisis médico, glaucoma de ángulo cerrado*

Se pretende desarrollar una aplicación para el sistema operativo Android capaz de identificar y medir, mediante el uso de una fuente de luz, el tamaño de la sombra proyectada en el ojo de un paciente.

Este desarrollo tendrá como objetivo el estimar, en primer lugar, si un paciente posee la enfermedad glaucoma de ángulo cerrado y, en segundo lugar, proveer una estimación del grado del mismo. Con la detección generada se podría ayudar en el diagnóstico de la enfermedad a un médico no especializado. Además, se implantará un sistema de comunicación simulando un servidor médico, para en un futuro integrar el resultado de este proyecto en un protocolo médico de actuación.

Las herramientas básicas usadas serán las librerías *OpenCV* e *Image Cropper*, el IDE *Android Studio* y un *sistema de control de versiones*. A su vez se requiere que el software desarrollado posea una interfaz intuitiva para el usuario y un software libre de errores. Para hacer esto posible se usarán otras herramientas que permitan:

- Creación de imágenes e iconos
- Diseño de interfaces para Android
- Manejo y modificación de imágenes

# Closed-Angle glaucoma detection

Ignacio Martín Requena

## Abstract

**Keywords:** *computer vision, Android programming, OpenCV, medical analysis, Closed-Angle Glaucoma*

The aim of this project is the development of a medical system to diagnose close-eyed glaucoma. Glaucoma is a disease in which one, one of the symptoms is the increase of the eye pressure. Due to this high pressure, the iris changes its shape to an irregular one so, with the support of a source of light, we can generate a shadow in the eye that eventually could give to a doctor some information about the chance that the patient suffers this disease.

Having the image of an eye in the conditions described above, we try to measure the shadow generated using computer vision techniques. In addition, an Android application will be built on top of this algorithm in order to give to the user an easy and rapid way to generate a diagnostic.

Finally, a communication system will be established between both the Android application and a remote server. With this system, we target to have all the features needed to integrate this project into a medical software/protocol.

This document will describe all the steps followed in order to reach the described aims. To help us on that, different tools will be used, the most important are Android Studio, OpenCV, Android Crop Image library, and Visual Paradigm among others that help us in the design of the interface and the handling of the input image.

# Índice general

<b>1. Introducción</b>	<b>1</b>
<b>2. Objetivos</b>	<b>5</b>
2.1. Alcance de los objetivos . . . . .	6
2.2. Interdependencia de los objetivos . . . . .	7
<b>3. Análisis</b>	<b>9</b>
3.1. Análisis de requisitos . . . . .	9
3.1.1. Descripción de los actores . . . . .	9
3.1.2. Requisitos funcionales . . . . .	10
3.1.3. Requisitos no funcionales . . . . .	10
3.1.4. Requisitos de información . . . . .	11
3.2. Modelos de casos de uso . . . . .	11
3.2.1. Descripción básica de actores . . . . .	11
3.2.2. Descripción casos de uso . . . . .	12
3.3. Cobertura de los requisitos funcionales . . . . .	19
3.4. Diagrama de paquetes . . . . .	20
3.5. Diagramas de casos de uso . . . . .	20
3.6. Diagramas de actividad . . . . .	22
3.7. Diagrama conceptual . . . . .	28
3.8. Diagramas de algoritmo de detección . . . . .	32
<b>4. Implementación</b>	<b>37</b>
4.1. Recursos empleados . . . . .	37
4.2. Arquitectura del sistema . . . . .	37
4.3. Android Image Cropper . . . . .	38
4.4. Implementación . . . . .	39
4.4.1. Vista Diagnóstico . . . . .	39
4.4.2. Vista Historial . . . . .	41
4.4.3. Comunicación con el servidor . . . . .	46
4.4.4. Algoritmo de detección . . . . .	50
4.5. Diseño de las vistas y menús . . . . .	55
4.5.1. Tema . . . . .	56

---

4.5.2. Layouts . . . . .	56
4.5.3. Tabs . . . . .	57
<b>5. Pruebas</b>	<b>59</b>
5.1. Precisión de la detección del iris . . . . .	59
5.2. Precisión de la medida de la sombra . . . . .	61
<b>6. Conclusiones y trabajos futuros</b>	<b>63</b>
<b>Glosario de términos</b>	<b>65</b>
<b>Bibliografía</b>	<b>69</b>

# Índice de figuras

1.1. Mecanismo del cierre angular . . . . .	2
1.2. Luz en ojo sano vs ojo con glaucoma . . . . .	3
3.1. Diagrama de paquetes . . . . .	20
3.2. Diagrama de casos de uso: paquete Manager . . . . .	21
3.3. Diagrama de actividad CU-1. Inicio de la aplicación . . . . .	22
3.4. Diagrama de actividad CU-2. Realizar una nueva detección .	23
3.5. Diagrama de actividad CU-3. Guardar un diagnóstico . . . . .	24
3.6. Diagrama de actividad CU-4. Consultar historial de diagnósticos . . . . .	25
3.7. Diagrama de actividad CU-5. Consultar información detallada de un diagnóstico . . . . .	26
3.8. Diagrama de actividad CU-6. Consultar información sobre la app . . . . .	27
3.9. Diagrama conceptual del paquete Manager . . . . .	28
3.10. Diagrama conceptual del paquete Diagnóstico . . . . .	29
3.11. Diagrama conceptual del paquete Historial . . . . .	30
3.12. Diagrama conceptual del paquete comunicación . . . . .	31
3.13. Diagrama de flujo del algoritmo de detección general . . . . .	32
3.14. Diagrama de flujo del pre-procesado (Fase 1) del algoritmo de detección . . . . .	33
3.15. Diagrama de flujo de la obtención de la ROI del iris (Fase 2)	34
3.16. Diagrama de flujo de la obtención y medida del tamaño de la sombra (Fase 3) . . . . .	35
4.1. Modelo vista controlador para el caso concreto de nuestro sistema . . . . .	38
4.2. Vista diagnóstico por defecto . . . . .	39
4.3. Vista diagnóstico con resultado . . . . .	40
4.4. Vista historial por defecto . . . . .	42
4.5. Vista historial con resultados . . . . .	43
4.6. Vista detalles de un diagnóstico guardado . . . . .	45
4.7. Configuración directorio servidor . . . . .	47

---

4.8.	Datos diagnóstico con título paciente_1 . . . . .	48
4.9.	Imagen diagnóstico con título paciente_1 . . . . .	49
4.10.	Informe diagnóstico con título paciente_1 . . . . .	49
4.11.	Ejemplo del algoritmo de pre-procesado . . . . .	51
4.12.	Ejemplo del funcionamiento de la ecualización de histograma	52
4.13.	Ejemplo del algoritmo de selección del iris . . . . .	53
4.14.	Ejemplo del algoritmo de medida de la sombra . . . . .	54
4.15.	Ejemplo de la operación morfológica de apertura . . . . .	55
4.16.	Colores base de la aplicación . . . . .	56
5.1.	Resultados de los test de la detección del iris . . . . .	60
5.2.	Resultados de los test de medida de la sombra . . . . .	61
6.1.	Imagen del ángulo camerular tomada con OCT . . . . .	65

# Índice de tablas

3.1.	Curso normal de CU-1. Inicio de la aplicación . . . . .	13
3.2.	Curso alterno de CU-1. Inicio de la aplicación . . . . .	13
3.3.	Curso normal de CU-2. Realizar una nueva detección . . . . .	14
3.4.	Curso alterno de CU-2. Realizar una nueva detección . . . . .	15
3.5.	Curso normal de CU-3. Guardar un <i>diagnóstico</i> . . . . .	16
3.6.	Curso alterno de CU-3. Guardar un <i>diagnóstico</i> . . . . .	16
3.7.	Curso normal de CU-4. Consultar historial de diagnósticos . .	17
3.8.	Curso alterno de CU-4. Consultar historial de diagnósticos . .	17
3.9.	Curso normal de CU-5. Consultar información detallada de un diagnóstico guardado . . . . .	18
3.10.	Curso normal de CU-6. Consultar información sobre la app. .	19

# Índice de fragmentos de código

4.1.	Actualización de la vista nuevo diagnóstico . . . . .	41
4.2.	Actualización del historial . . . . .	44
4.3.	Fichero PHP para la comunicación con el servidor . . . . .	46
4.4.	Pseudocódigo de del envío del diagnóstico . . . . .	48
4.5.	Algoritmo de ecualización de Histograma . . . . .	52





# Capítulo 1

## Introducción

Unas de las características que diferencian al ser humano del resto de los animales es su capacidad para manejar conceptos e información. De hecho, se puede observar como el avance de la humanidad va correlado tanto con el acceso que esta tiene a la información de su entorno como con la sistematización de nuestro propio pensamiento para usarla apropiadamente.

Por ello, la informática como ciencia que estudia el manejo y el tratamiento automático de la información es actualmente el centro de gravedad para el desarrollo de la mayoría de disciplinas científicas, incluida la medicina.

Las nuevas técnicas de Inteligencia Artificial tales como el aprendizaje automático y la visión por computador nos brindan la posibilidad de construir sistemas capaces de detectar y diagnosticar automáticamente anomalías en el cuerpo humano. Desde aplicaciones para detectar cambios peligrosos en las células humanas y prevenir la creación de tumores [1], hasta la detección de cáncer en la piel humana [2], demuestran el potencial de la Inteligencia Artificial y, en concreto, de la visión por computador para resolver problemas médicos.

Debido a mi pasión por la visión por computador descubierta durante mi transcurso por la Universidad de Granada y el desarrollo de mis conocimientos durante mis prácticas en FrontierIP Group plc. unidos con la motivación y fuente de información por parte de Manuel Valdearenas Martín [3], tomé conciencia sobre la importancia que hoy en día supone la detección temprana del glaucoma, en concreto la del de ángulo cerrado.

Incluso recibiendo un tratamiento adecuado, aproximadamente el 10 % de las personas con glaucoma experimentan pérdida de la visión. Además, el glaucoma no tiene cura, y no es posible recuperar la visión perdida. Debido

a que tal enfermedad es una afección crónica, es necesario controlarlo de por vida. Todo esto revela la importancia de diagnóstico temprano del mismo para conservación de la visión del paciente.

Algunos datos más que nos muestran la necesidad de ofrecer técnicas de diagnóstico del glaucoma son:

- Se calcula que más de 2,2 millones de estadounidenses tienen glaucoma pero sólo la mitad sabe que lo tiene.
- En los EE. UU., más de 120.000 personas son ciegas debido al glaucoma, lo cual representa entre el 9 y el 12 de todos los casos de ceguera.
- El glaucoma es la segunda causa principal de ceguera en el mundo, según la Organización Mundial de la Salud.
- Después de la catarata, el glaucoma es la causa principal de ceguera en las personas afroamericanas.

Entre los síntomas que muestra el glaucoma se encuentra el cambio en la forma del iris. En concreto, el glaucoma de ángulo cerrado recibe su nombre debido al estrechamiento en el ángulo que forma el iris y la cornea a consecuencia el aumento de la presión ocular. Debido a este aumento de la presión intraocular, el nervio óptico recibe una fuerza cada vez más grande. Esta es la causa de que el riesgo de ceguera aumente exponencialmente con respecto a la presión ocular.

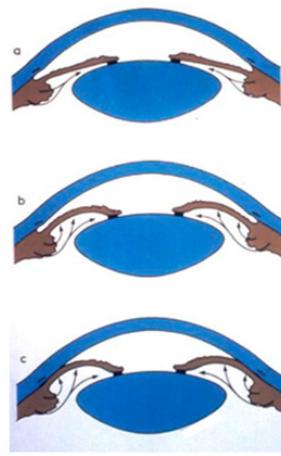


Figura 1.1: Mecanismo del cierre angular  
Originado por la presión del humor acuoso desde la cámara posterior  
levantando el iris y aproximándolo a la córnea.

Gracias a la irregularidad generada en la forma del iris es posible valorar el tamaño de la misma midiendo la sombra generada al poner una fuente de luz paralela al plano del iris:

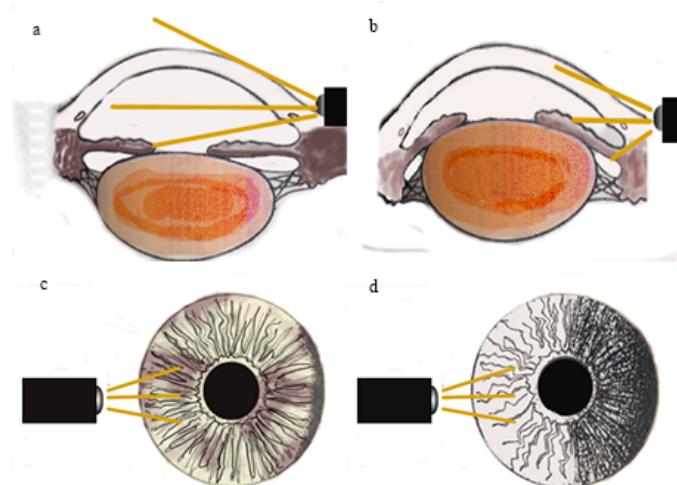


Figura 1.2: Luz en ojo sano vs ojo con glaucoma

a) ojo sano, b) ojo que sufre glaucoma de ángulo cerrado, c) sombra teórica proyectada por un ojo sano y d) sombra teórica proyectada por un ojo que sufre glaucoma de ángulo cerrado [4]

Por lo tanto, el objeto y principal motivación de este proyecto es el de ofrecer un sistema de detección y medida de la sombra proyectada en la sección del iris opuesta a la fuente de luz aplicada. Obteniendo esta medida, se podrá ofrecer al personal sanitario de un sistema para valorar si el paciente puede o no sufrir de glaucoma y necesita, por tanto, atención de un especialista.

La información detallada del diseño y desarrollo tanto del sistema de detección como de la aplicación Android será detallada en el *capítulo 4 (Implementación)*.

Además, previo al paso de los detalles técnicos, se ofrecerán otros aspectos del proyecto como son:

- En el *capítulo 2 (Objetivos)*, cuyo fin es detallar de forma concreta los objetivos determinados que se quieren cumplir con este proyecto.
- En el *capítulo 3 (Análisis)*, que describe todos los requisitos que se necesitan cubrir con el software a desarrollar, así como describir cómo esos requisitos se irán añadiendo al proyecto.

Una vez estén realizadas tanto la parte descriptiva como la parte de desarrollo, se añaden dos apartados finales; el *capítulo 5 (Pruebas)* en el que se explica las diferentes pruebas a las que ha sido sometido el software desarrollado para comprobar su correcto funcionamiento y evaluar la precisión del sistema de detección, y el *capítulo 6 (Conclusiones y trabajos futuros)* en el que se expondrán los conocimientos y cuestiones que el desarrollo del proyecto me ha brindado, además del futuro trabajo que podría desarrollarse.

## Capítulo 2

# Objetivos

Como se ha introducido en el apartado anterior, el objetivo de este proyecto es el desarrollo de una aplicación Android para el diagnóstico, por parte de un enfermero o médico general, del glaucoma de ángulo cerrado. Además, se ofrecerá la posibilidad al usuario de enviar el diagnóstico generado a un servidor con el fin de, posteriormente, implantar un protocolo de actuación para aquellos casos que sugieran indicios de padecer glaucoma. De esta forma se ofrece un sistema de ayuda para un médico no especializado brindándole el establecimiento de una comunicación con el especialista (oftalmólogo). De tal forma, el médico general podría, en caso de obtener un diagnóstico positivo y tras comprobar que esté libre de errores, tomar la decisión de derivarlo al especialista para que este realice un análisis ocular más exhaustivo.

Al ser un sistema orientado a personas que tienen conocimientos previos sobre la enfermedad a diagnosticar y las técnicas a emplear para generar correctamente la sombra, se supondrá que, al igual que ocurre con la mayoría de los sistemas de ayuda y asistencia médica (ej. radiografías), el usuario final posee los conocimientos necesarios para el uso de la aplicación y la correcta toma de imágenes.

Además, como la parte de protocolo de comunicación entre el médico general o enfermero y el especialista requeriría la implantación de un protocolo médico de actuación, lo que supondría un gran aumento tanto en la temporalización como en la complejidad de este proyecto, la implementación del mismo se dejará pendiente para futuros trabajos.

De acuerdo con todo lo anteriormente descrito, se puede concluir que el objeto de este proyecto serán la implementación de las funcionalidades necesarias para cubrir desde la implantación del sistema de detección usando un dispositivo móvil hasta el envío, en caso de que el usuario lo solicite, del diagnóstico a un servidor.

Una enumeración de los principales objetivos a alcanzar es:

- **OBJ-1.** Detectar el tamaño de la sombra proyectada en el iris al aplicar una fuente de luz paralela al plano de este con el fin de diagnosticar glaucoma de ángulo cerrado.
- **OBJ-2.** Identificar y localizar correctamente, dada una imagen de un ojo, donde se sitúa el iris del mismo.
- **OBJ-3.** Desarrollar una aplicación de móvil en Android que sirva a un médico o enfermero de interfaz para diagnosticar irregularidades en el iris de un paciente.
- **OBJ-4.** Ofrecer una comunicación con un servidor en el cual enviar los datos de un diagnóstico para su posterior evaluación.

Además, como objetivos secundarios tendremos:

- **OBJ-5.** Estudiar la posibilidad de ser capaz de detectar la región del iris y medir el tamaño de la sombra proyectada en el mismo independientemente de las condiciones de luz ambiental.
- **OBJ-6.** Estudiar la posibilidad de ser capaz de detectar la región del iris y medir el tamaño de la sombra proyectada en el mismo independientemente del ángulo que forman la cámara del dispositivo y el ojo a detectar.
- **OBJ-7.** Implementar un historial de diagnóstico donde el usuario pueda guardar localmente aquellos que requieran su atención.

Destacar en los aspectos formativos previos más utilizados para el desarrollo del proyecto los conocimientos sobre visión por computador y procesado de imágenes para el algoritmo de detección, ingeniería de software y programación y desarrollo orientado a objetos para el análisis del proyecto e implementación del mismo, nuevos paradigmas de interacción para entender lo concerniente a la programación en Android e ingeniería del conocimiento para la extracción de la información del especialista.

## **2.1. Alcance de los objetivos**

La aplicación resultante será visualmente intuitiva para el usuario haciendo un gran hincapié en aspectos como la navegabilidad entre menús.

En esta aplicación se usará además como herramienta para un primer diagnóstico por parte tanto de médicos generales que requieran una fuente de información rápida y sencilla como paso previo a un estudio más detenido sobre la posibilidad de que el paciente posea glaucoma.

## 2.2. Interdependencia de los objetivos

Todos los objetivos son independientes entre sí, pero el primer objetivo (**OBJ-1**) es el principal motivador de este proyecto, por consiguiente, es este el que va a escudar y avalar el desarrollo de los otros. En un primer paso, el segundo objetivo (**OBJ-2**) es el que se solucionará de forma inmediata ya que es este el que hace posible (**OBJ-1**). El resto de objetivos serán tratados en mayor medida durante los capítulos de implementación y pruebas.



# Capítulo 3

## Análisis

### 3.1. Análisis de requisitos

Tal y como hemos especificado previamente en el *Capítulo 2*, todo el software se desarrolla con el fin de cubrir una serie de necesidades, entre las cuales la principal es la de diagnosticar el glaucoma de ángulo cerrado. Por ello en este apartado se describirán los requisitos que se estiman necesarios para cubrir los objetivos propuestos previamente.

#### 3.1.1. Descripción de los actores

Los actores implicados serán tres: el **desarrollador**, el **usuario** y el **servidor web**.

El **desarrollador** será el encargado de desarrollar y mejorar el algoritmo de detección para medir la sombra del iris y de ofrecer una interfaz amena e intuitiva. También asumirá la labor tanto de explicación del uso de la interfaz como del mantenimiento y gestión del servidor que alojará los diagnósticos guardados.

El **usuario** de la aplicación será cualquier personal médico que tenga interés por conocer la posibilidad de que su paciente de glaucoma de ángulo cerrado. El usuario usual por tanto sera un médico general, un enfermero o, con menor probabilidad, un oftalmólogo especializado.

El **servidor web** de la aplicación será el sistema encargado de recopilar los informes con los diagnósticos generados. Este servirá para simular la futura integración del proyecto en un servidor médico.

Los actores descritos son la generalización de todos los tipos de personas y sistemas que requerirán del acceso a la aplicación, si bien es cierto que en un futuro esto puede ser ampliado.

### 3.1.2. Requisitos funcionales

Los requisitos funcionales son las características que tiene que implementar el sistema para cubrir todas las necesidades de los distintos usuarios.

El interés único del usuario es el de obtener una información sobre el tamaño de la sombra proyectada en el iris del paciente para así poder tomar la decisión de derivarlo a un especialista. Para ello el desarrollador deberá hacer que sea posible que se genere siempre una salida fácilmente interpretable e intuitiva.

Para proporcionar esto, el desarrollador debe implementar un software con una serie de servicios y funciones particulares brindadas al usuario.

- **RF-1.** Crear una nueva detección:
  - **RF-1.1.** Pedir al paciente la imagen del ojo a detectar.
  - **RF-1.2.** Pedir al usuario que recorte la sección donde se encuentra el ojo.
  - **RF-1.3.** Proporcionar al usuario la posibilidad de girar y voltear la imagen.
  - **RF-1.4.** Mostrar el resultado de la detección y, en caso satisfactorio, el tamaño de la sombra detectada.
  - **RF-1.5.** Ofrecer la posibilidad de envío del diagnóstico a un servidor para integrarlo, en un futuro, con un sistema médico.
- **RF-2.** Acceso de información:
  - **RF-2.1.** Consultar información sobre diagnósticos previos enviados.
  - **RF-2.2.** Consultar información sobre la versión, el desarrollador y el contacto del mismo.

### 3.1.3. Requisitos no funcionales

Los requisitos no funcionales son las características propias del desarrollo, pero que no tienen que estar relacionadas con su funcionalidad. Por tanto, se concretarán las librerías, plataformas y softwares empleados para cumplir los requisitos funcionales.

- **RN-1.** Toda la programación del sistema se hará usando la plataforma **Android Studio** y, por tanto, haciendo uso del lenguaje de programación Java para la implementación de las funcionalidades de la aplicación y XML para el desarrollo de la interfaz gráfica de usuario.

- **RN-2.** La parte de la detección y medida de la sombra se desarrollará haciendo uso de la librería OpenCV 3.4.
- **RN-3.** La aplicación Android se desarrollará utilizando la versión de SDK 27 para dispositivos con versiones superiores a Android KitKat 6.0.
- **RN-4.** El diseño de la interfaz se guiará por las reglas Material Design.
- **RN-5.** Todos los menús de la interfaz de usuario se construirán utilizando Fragments.
- **RN-6.** Para la petición de la selección de la región de interés al usuario se usará la librería Android Image Cropper[7].
- **RN-7.** La comunicación con el servidor se realizará utilizando php y la librería de Android Apache.
- **RN-8.** Los test de precisión del algoritmo de detección se basarán en un conjunto de datos de 15 pacientes para comprobar la correcta localización del iris y 6 para la evaluación de la medida de la sombra.
- **RN-9.** La navegación por los diferentes menús se realizará mediante el uso de Navigation Drawer y Action Bar proporcionado por Android SDK.

### 3.1.4. Requisitos de información

Los requisitos de información se refieren a los datos que son necesarios almacenar en el sistema. En este caso, estos datos serán los relacionados con el resultado del diagnóstico. Una vez mostrada la detección junto con el tamaño de la sombra, se le proporcionará al usuario la opción de guardar el resultado en el historial de detecciones y enviarla a un servidor para un análisis futuro por parte de un especialista.

- **RI-1.** Resultado de la detección.
  - Información sobre el resultado de la detección de la sombra producida en el iris.
  - Contenido: Imagen del ojo, título, fecha, tamaño de la sombra.

## 3.2. Modelos de casos de uso

### 3.2.1. Descripción básica de actores

- **Ac-1.** Desarrollador.

- Descripción: Encargado del desarrollo y administración de la aplicación.
- Características: Su trabajo es el de actualizar, corregir errores, mantener y adecuar la aplicación a las necesidades del usuario como el de asegurar el acceso al servidor que alacena los diagnósticos enviados.
- Relaciones: Ninguna.
- Atributos: Ninguno.
- Comentarios: Ninguno.

- **Ac-2. Usuario.**

- Descripción: Persona que usa la aplicación para realizar un diagnóstico.
- Características: Es el médico no especializado que accederá a la aplicación.
- Relaciones: Ninguna.
- Atributos: Ninguno.
- Comentarios: El usuario necesita tener conocimientos sobre qué es el glaucoma y la técnica que se va a usar para diagnosticarlo para el correcto uso de la aplicación.

- **Ac-3. Servidor web.**

- Descripción: Servidor donde se alojarán los resultados de un diagnóstico para poder ser recibidos por un especialista.
- Características: Comunicará el médico general con el especialista.
- Relaciones: Ninguna.
- Atributos: Ninguno.
- Comentarios: El usuario necesita tener conocimientos sobre qué es el glaucoma y la técnica que se va a usar para diagnosticarlo para el correcto uso de la aplicación.

### **3.2.2. Descripción casos de uso**

- **CU-1. Inicio de la aplicación.**

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: La aplicación debe haber sido instalada previamente.

- Postcondición: La aplicación estará lista para ser usada.
- Autor: Ignacio Martín Requena.
- Versión: 1.0.
- Propósito: Inicializar todo lo necesario para el correcto uso de la aplicación.
- Resumen: Cuando se lanza la aplicación, se inicializan todos los menús de navegación, animaciones y se carga la librería OpenCV e Image Cropper encargadas de la detección y la selección del área de interés respectivamente.

<b>Curso normal</b>			
	<b>Actor</b>	<b>Sistema</b>	
1	Usuario: Pulsa el icono que ejecuta la aplicación desde el menú de su dispositivo.		
		2a	Comprueba que OpenCV se ha cargado satisfactoriamente, inicia los menús y animaciones de la aplicación.

Tabla 3.1: Curso normal de CU-1. Inicio de la aplicación

<b>Curso alterno</b>	
2b	Si no se encuentra la librería OpenCV, se muestra un mensaje de error.

Tabla 3.2: Curso alterno de CU-1. Inicio de la aplicación

■ **CU-2.** Realizar una nueva detección.

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: CU-1 previamente realizado. Que exista una imagen de un ojo iluminado por una fuente de luz perpendicular al plano del iris del mismo.
- Postcondición: El usuario recibirá una imagen con el resultado de la detección de la sombra del iris.
- Autor: Ignacio Martín Requena.
- Versión: 1.0.
- Propósito: El usuario desea obtener un diagnóstico sobre el grado de glaucoma de ángulo cerrado de un paciente.
- Resumen: El usuario, una vez lanzada la aplicación, pulsa el botón nuevo diagnóstico e introduce la imagen del ojo a diagnosticar, obteniendo así la información solicitada.

Curso normal		
	Actor	Sistema
1	Usuario: Inicia una nueva detección.	
		2 Se solicita la introducción de la imagen del ojo a diagnosticar.
		3a El usuario selecciona el método de introducción de la imagen: Cámara, galería, sistema de archivos, servicio Cloud y, en general, cualquier opción proporcionada por el sistema Android.
		4a El usuario selecciona la región de la imagen donde se encuentra el iris del ojo. A continuación, pulsa recortar.
		5a Con la imagen recortada, el algoritmo de detección realiza lo necesario para medir el tamaño de la sombra. El resultado final se muestra en la pantalla.

Tabla 3.3: Curso normal de CU-2. Realizar una nueva detección

Curso alterno	
3b	Si no se introduce ninguna imagen, se vuelve a la pantalla de inicio.
4b	Si se pulsa el botón retroceder en esta sección, se vuelve a la pantalla de inicio.
5b	Si la detección no es satisfactoria (ej. no se encuentra el iris) se solicita al <b>Usuario</b> intentar realizarla de nuevo.

Tabla 3.4: Curso alterno de CU-2. Realizar una nueva detección

■ **CU-3.** Guardar un *diagnóstico*.

- Actores: Usuario, servidor web.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: Que se haya realizado previamente una detección de la sombra.
- Postcondición:
- Autor: Ignacio Martín Requena.
- Versión: 1.0.
- Propósito: El usuario requiere guardar el resultado de la detección para un futuro análisis de la misma.
- Resumen: El usuario, una vez obtenida el diagnóstico, pulsa el botón de guardar diagnóstico y guarda el mismo en la sección historial de diagnósticos y lo envía al servidor que almacena los diagnósticos a analizar.

Curso normal			
	Actor	Sistema	
1	Usuario: solicita guardar el diagnóstico.		
		2a	Se pide un nombre para el diagnóstico.
		3a	Se guarda la imagen, el tamaño de la sombra, el título y la fecha del diagnóstico en el historial.
		4a	Se envía la imagen, el tamaño de la sombra, el título y la fecha del diagnóstico al servidor.
5	Servidor web: recibe el resultado del diagnóstico.		

Tabla 3.5: Curso normal de CU-3. Guardar un *diagnóstico*

Curso alterno	
2b	Si no se introduce ningún nombre, se muestra un mensaje de error.
3b	Si el diagnóstico falló, se advierte de que esta guardando un posible resultado defectuoso.
4b	Si la conexión con el servidor falló, se avisa de que ha sido imposible enviar el diagnóstico.

Tabla 3.6: Curso alterno de CU-3. Guardar un *diagnóstico*

- **CU-4.** Consultar historial de diagnósticos.

- Actores: Usuario.
- Tipo: Primario, esencial.
- Referencias:
- Precondición: Existan diagnósticos previamente guardados.
- Postcondición:
- Autor: Ignacio Martín Requena.
- Versión: 1.0.
- Propósito: El usuario consulta datos de diagnósticos previamente realizados.
- Resumen: El usuario que usa la aplicación selecciona la sección de historial de diagnósticos y consulta la información de sus diagnósticos guardados.

Curso normal			
	Actor	Sistema	
1	Usuario: Solicita consultar el historial de <i>diagnósticos</i> seleccionándolo en el menú de navegación.	2	Se actualiza el historial de diagnósticos.
		3a	Se activa el <i>Fragment</i> encargado de la vista del historial.
		4	Se muestra una miniatura de las detecciones guardadas y el título de las mismas.

Tabla 3.7: Curso normal de CU-4. Consultar historial de diagnósticos

Curso alterno	
3b	Si no existe ningún diagnóstico guardado, la vista aparecerá vacía.

Tabla 3.8: Curso alterno de CU-4. Consultar historial de diagnósticos

- **CU-5.** Consultar información detallada de un diagnóstico guardado.
  - Actores: Usuario.
  - Tipo: Primario, esencial.
  - Referencias:
  - Precondición: Existan diagnósticos guardados previamente.
  - Postcondición:
  - Autor: Ignacio Martín Requena.
  - Versión: 1.0.
  - Propósito: El usuario consulta los detalles de un diagnóstico.
  - Resumen: El usuario que accede al historial de diagnósticos desea conocer la información relativa a un diagnóstico concreto del historial.

Curso normal		
	Actor	Sistema
1	Usuario: consulta la información de un diagnóstico guardado.	
		2 Se activa una nueva vista con la imagen del diagnóstico, la fecha de guardado el tamaño de la sombra y el nombre del diagnóstico.
		3 Cuando el usuario lo solicita, se retrocede al historial de diagnósticos.

Tabla 3.9: Curso normal de CU-5. Consultar información detallada de un diagnóstico guardado

- **CU-6.** Consultar información sobre la app.
  - Actores: Usuario.
  - Tipo: Primario, esencial.
  - Referencias:
  - Precondición: Menú de navegación desplegado.
  - Postcondición: Se genera un cuadro de diálogo con la información de contacto.
  - Autor: Ignacio Martín Requena.
  - Versión: 1.0.
  - Propósito: Consultar el desarrollador, la versión y como contactar con el desarrollador.
  - Resumen: El usuario desea consultar datos sobre la aplicación como la versión, el desarrollador y como contactar con el mismo.

Curso normal		
	Actor	Sistema
<i>1</i>	Usuario: consultar información de la aplicación.	
		<i>2</i> Comprueba se muestra un cuadro de diálogo con la información solicitada.
		<i>3</i> El usuario pulsa el botón “Ok”. El cuadro de diálogo se cierra.

Tabla 3.10: Curso normal de CU-6. Consultar información sobre la app.

### 3.3. Cobertura de los requisitos funcionales

	RF-1.1	RF-1.2	RF-1.3	RF-1.4	RF-1.5	RF-2.1	RF-2.2
<b>CU-1</b>							
<b>CU-2</b>	X	X	X	X			
<b>CU-3</b>					X		
<b>CU-4</b>						X	
<b>CU-5</b>						X	
<b>CU-6</b>							X

### 3.4. Diagrama de paquetes

Este diagrama representa la estructura lógica del sistema basado en las dependencias existentes entre cada unidad lógica. Una vez analizados los casos de usos requeridos por nuestro sistema, la definición de los paquetes y la interrelación entre ellos se resulta más fácil de organizar.

Lo lógico será disponer de cuatro unidades lógicas (paquetes) diferenciadas:

- **Paquete manager**, que gestiona los eventos de la aplicación.
- **Paquete diagnóstico**, que engloba todo el ciclo de generación de un nuevo diagnóstico.
- **Paquete comunicación**, encargado del envío de datos desde la app al servidor.
- **Paquete historial**, encargado de la gestión y visualización del histórico de diagnósticos guardados localmente.

Como se ya se intuye, el paquete Manager será el principal y del que dependan el resto de paquetes.

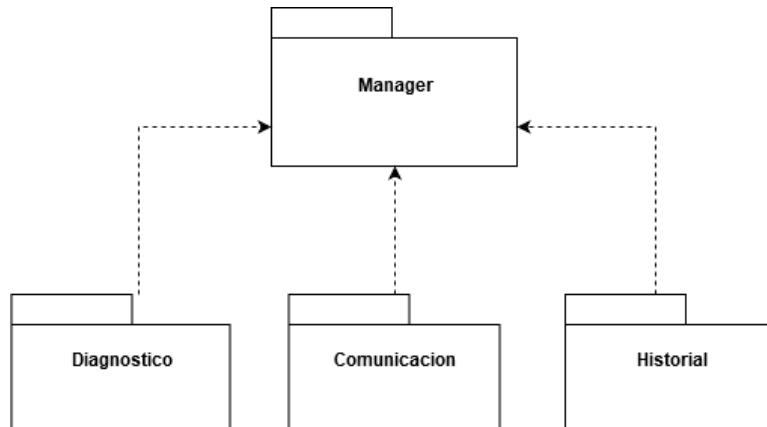


Figura 3.1: Diagrama de paquetes

### 3.5. Diagramas de casos de uso

Los diagramas de casos de uso representan como los diferentes actores se relacionan con el sistema para usar sus funciones. En nuestro caso concreto, después del desarrollo de casos de uso, podemos ver como el usuario será

el único que tendrá acceso directo al sistema. Además, esta interacción es gestionada únicamente por el paquete Manager, encargado de la gestión de los eventos en la app. El desarrollador, por tanto, quedará delegado a tareas de mantenimiento del servidor y actualización y mejora de las características del sistema.

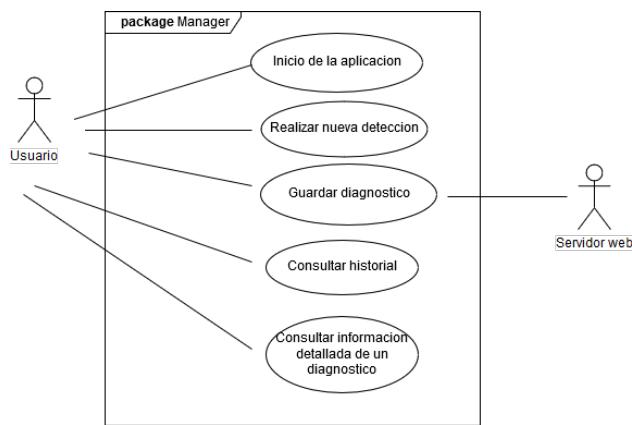


Figura 3.2: Diagrama de casos de uso: paquete Manager

El resto de paquetes se implementarán con el fin de dividir el sistema en módulos con funciones específicas y claramente diferenciadas. El objetivo de esto es el de conseguir un software modular, mantenable y fácilmente escalable con una distinción entre la gestión de los eventos y las acciones a realizar a partir de los mismos.

### 3.6. Diagramas de actividad

Los diagramas de actividad sirven para representar la descomposición de un proceso en las diferentes acciones de las que está compuesto. Las actividades de consultar algún tipo de información son procedimientos secuenciales en los que la ejecución es bastante simple; pero la actividad de iniciar la aplicación, realizar un diagnóstico y gestionar el historial de diagnósticos requieren de una complejidad adicional.

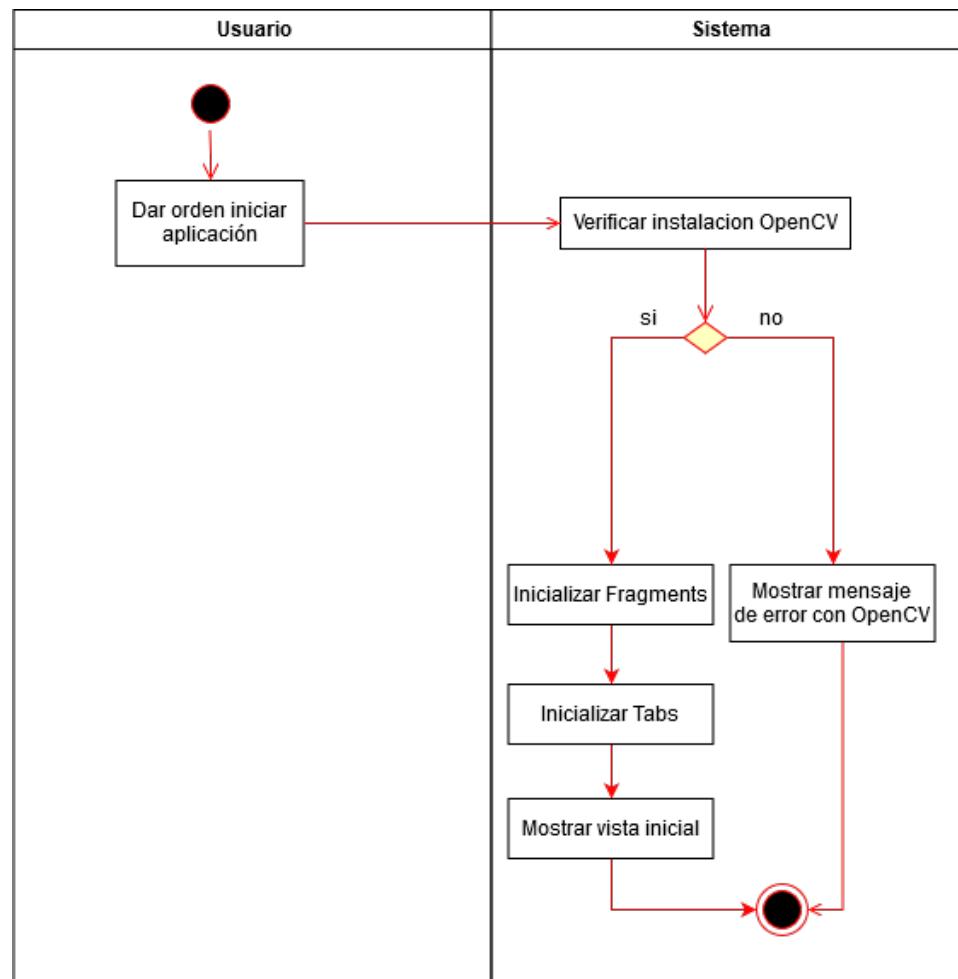


Figura 3.3: Diagrama de actividad CU-1. Inicio de la aplicación

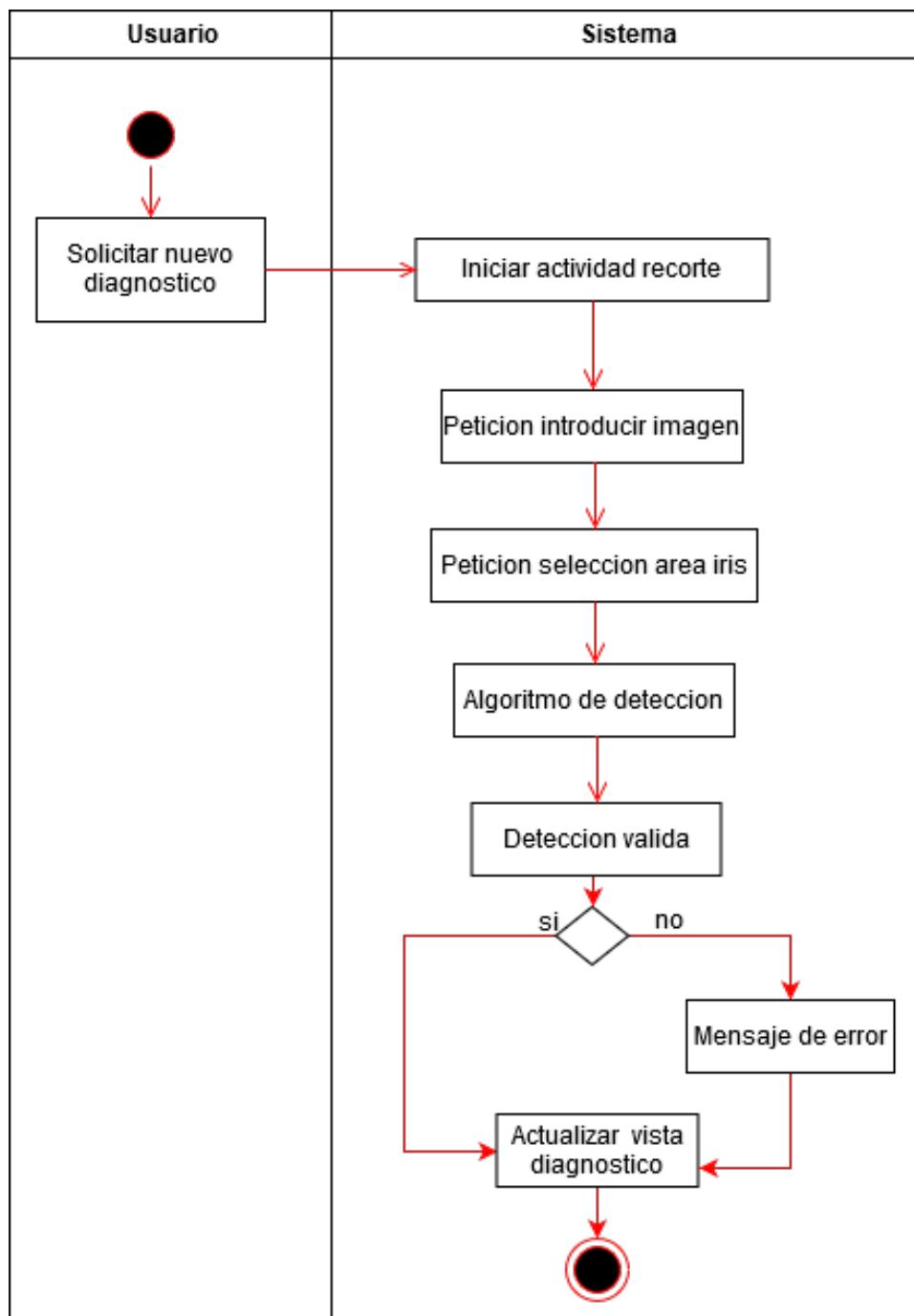


Figura 3.4: Diagrama de actividad CU-2. Realizar una nueva detección

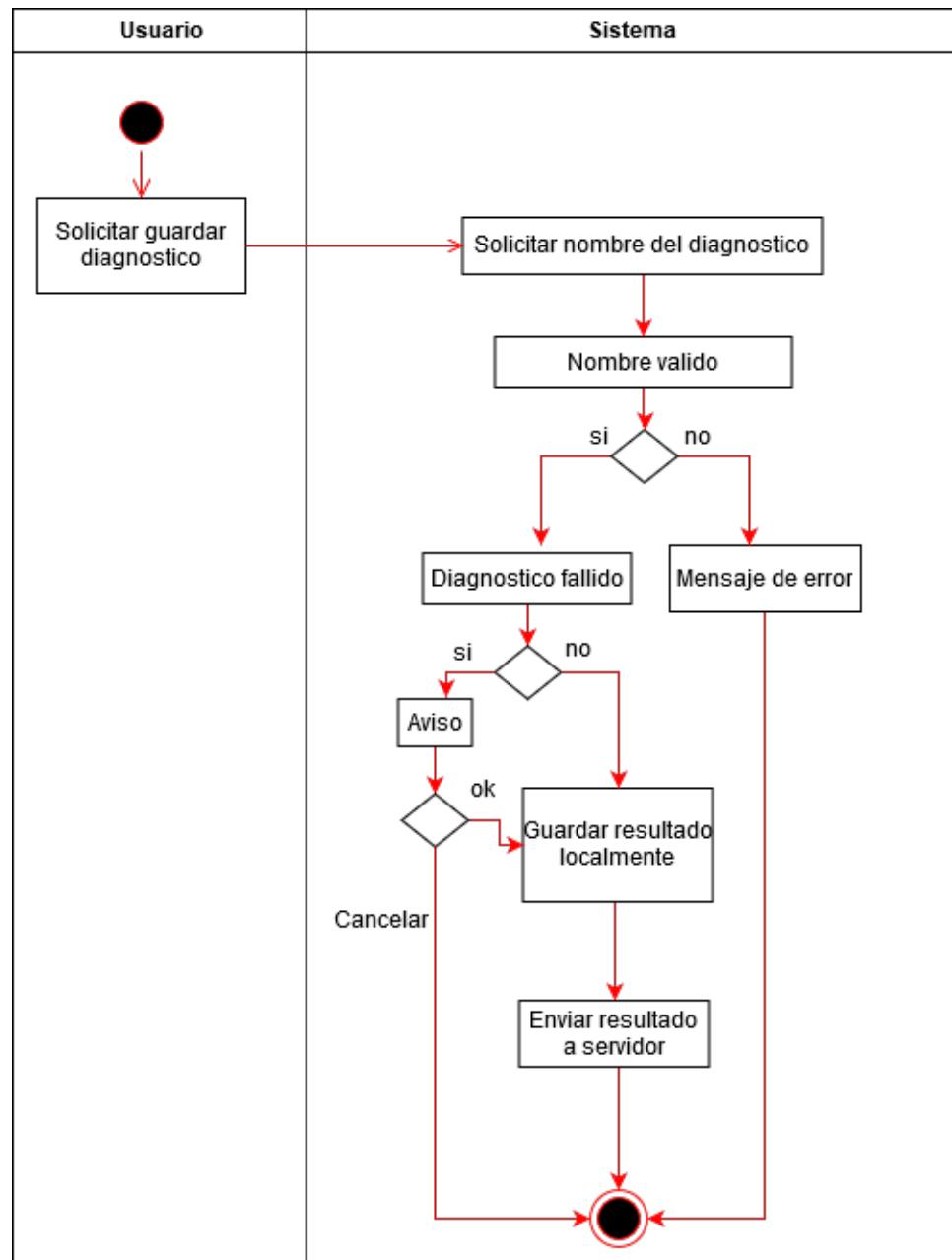


Figura 3.5: Diagrama de actividad CU-3. Guardar un diagnóstico

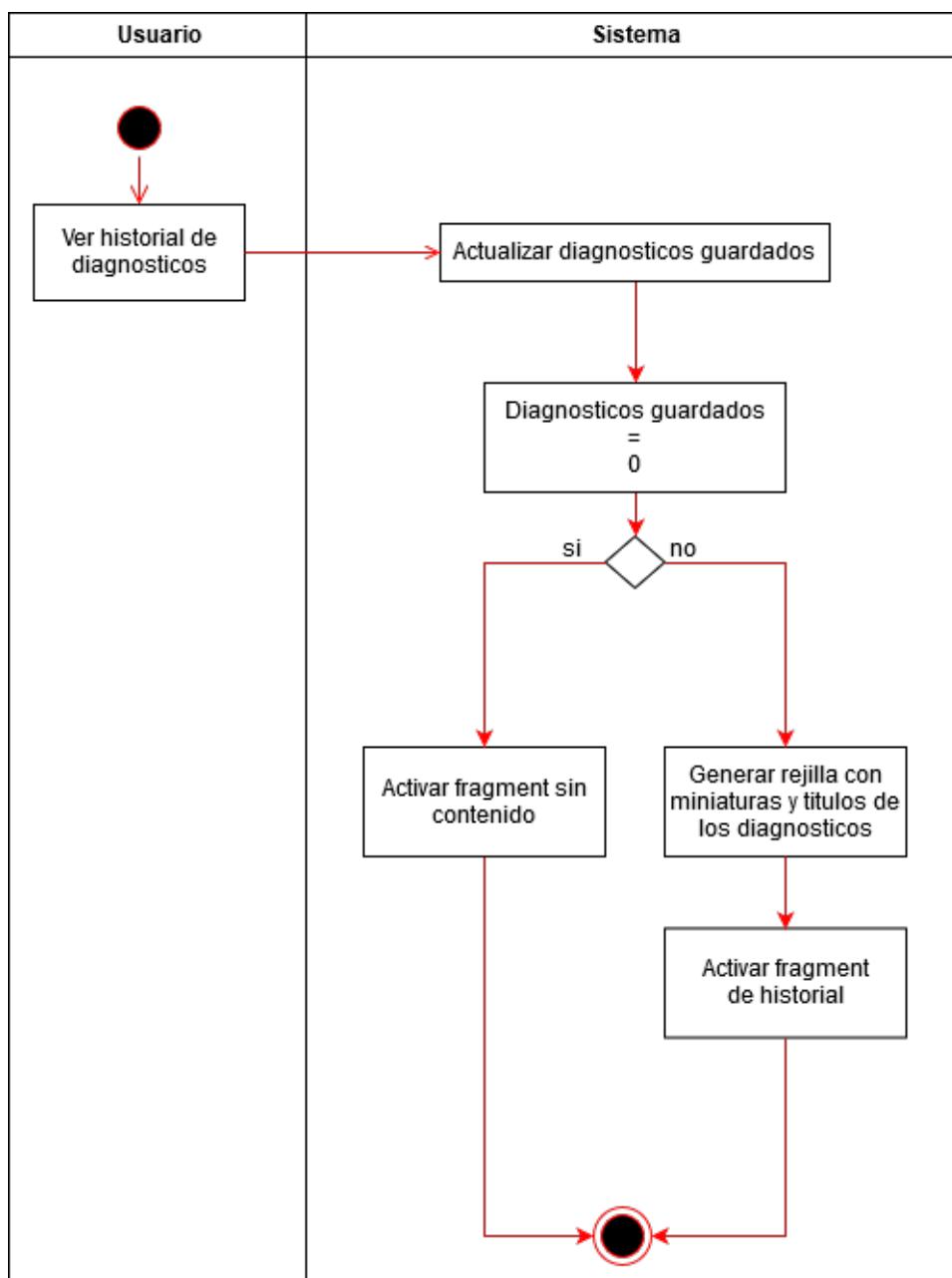


Figura 3.6: Diagrama de actividad CU-4. Consultar historial de diagnósticos

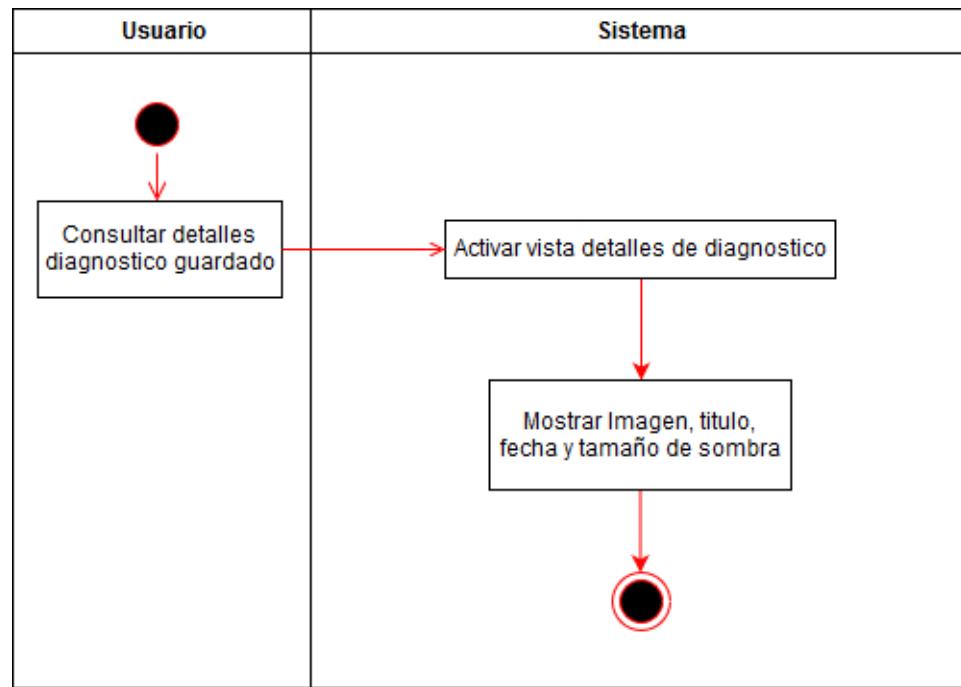


Figura 3.7: Diagrama de actividad CU-5. Consultar información detallada de un diagnóstico

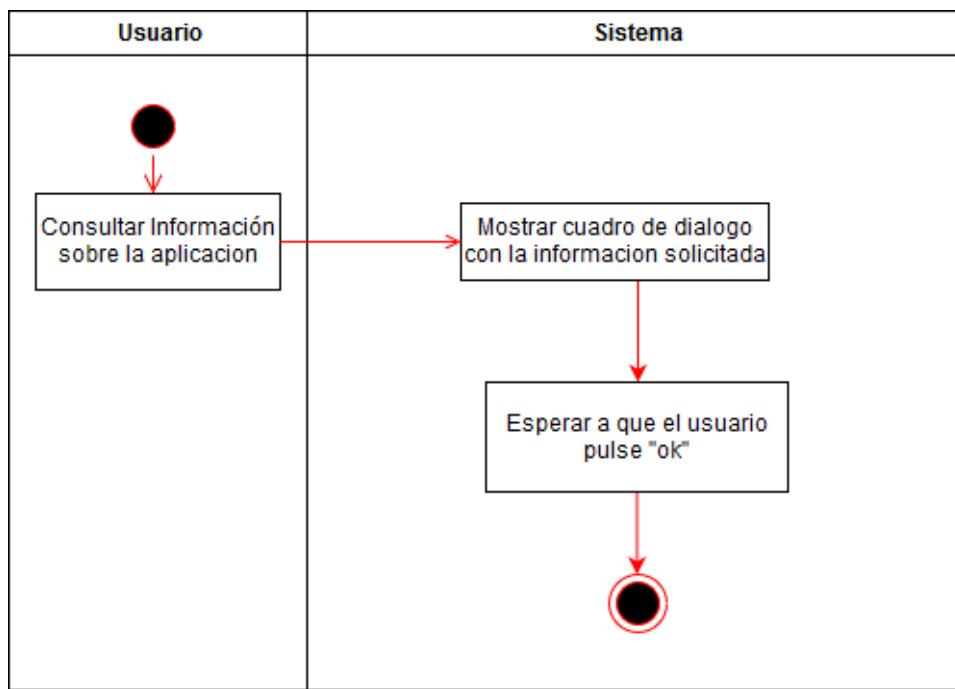


Figura 3.8: Diagrama de actividad CU-6. Consultar información sobre la app

### 3.7. Diagrama conceptual

En el diagrama conceptual podemos ver una representación de la estructura de la implementación. Para facilitar la lectura se han omitido los argumentos de la mayoría de métodos, así como la especificación de los tipos de datos de cada variable.

A continuación, se muestra la representación de cada paquete junto a una pequeña descripción:

- **El paquete Manager:** es el encargado de implementar y administrar los datos necesarios para el correcto funcionamiento de nuestra aplicación, por eso es el más importante. Dentro de este, la clase MainActivity es la encargada de la inicialización de cada una de las vistas de la interfaz gráfica.

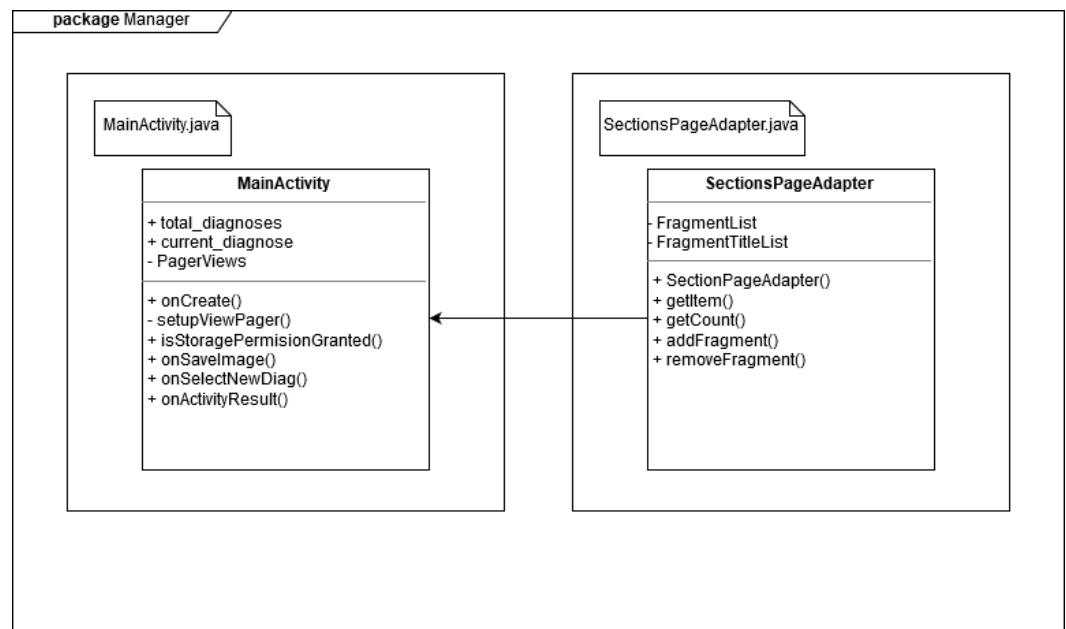


Figura 3.9: Diagrama conceptual del paquete Manager

- **El paquete Diagnóstico:** encargado de la generación de un nuevo diagnóstico, se compone de la clase que realiza la detección sobre una imagen (Detection), la clase que gestiona los datos mostrados por pantalla (DiagnoseView) y la clase DataDiagnose, cuyo fin es representar el resultado de un diagnóstico.

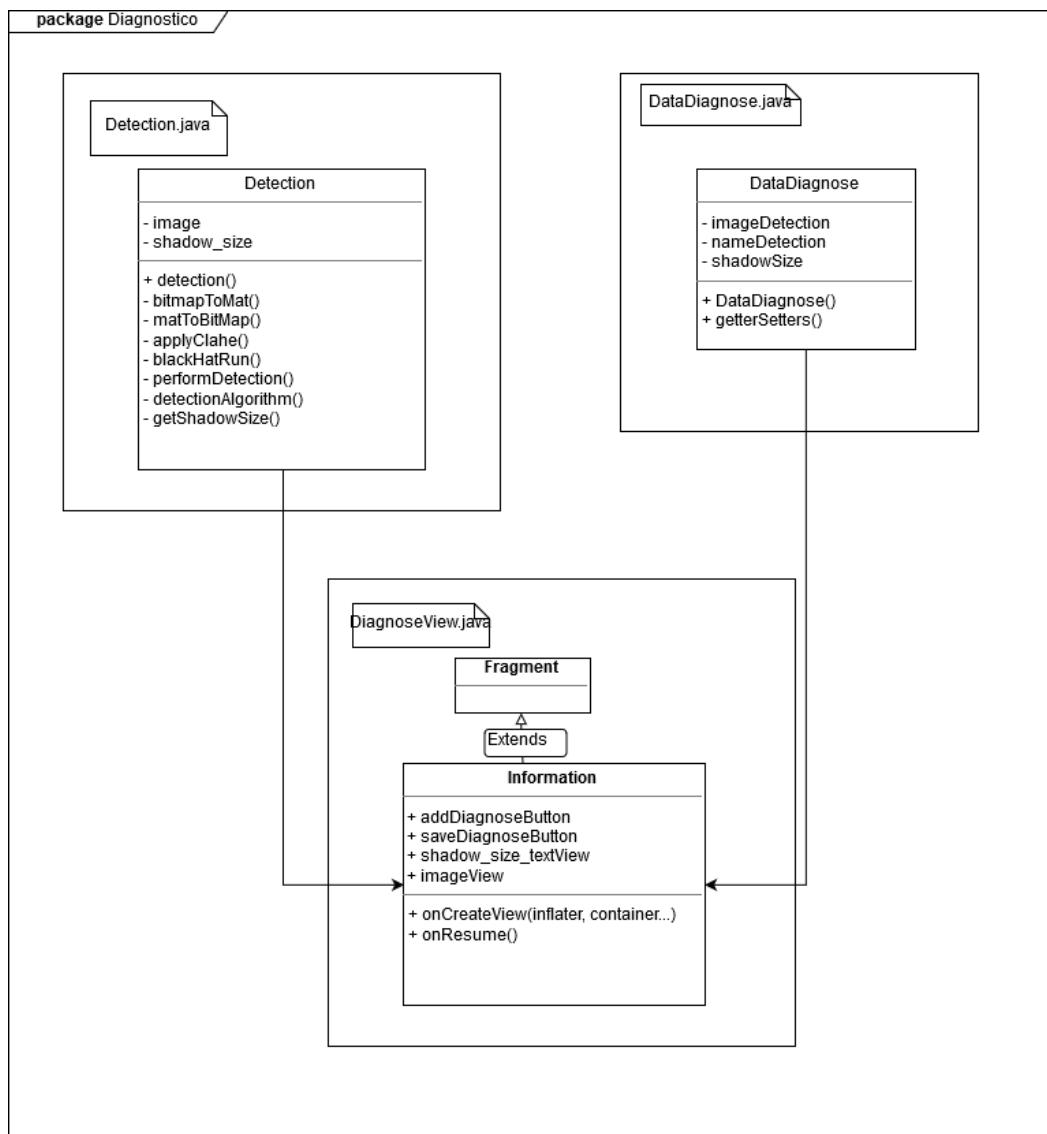


Figura 3.10: Diagrama conceptual del paquete Diagnóstico

- **El paquete Historial:** encargado de la visualización de la sección historial de la aplicación. Se compone de la clase que gestiona los dato mostrados por pantalla (`DiagHistory`), la encargada de representar las miniaturas de las imágenes guardadas (`GridViewAdapter`) y la que representa la vista detallada de un diagnóstico guardado (`DetailsActivity`).

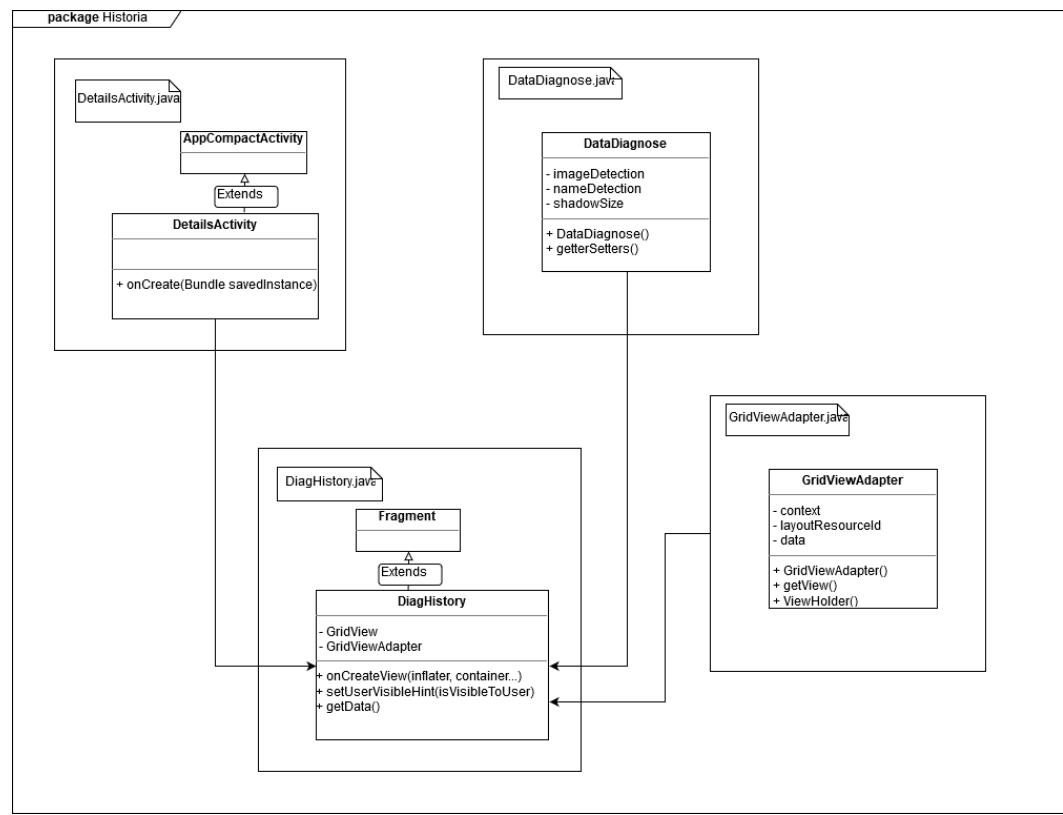


Figura 3.11: Diagrama conceptual del paquete Historial

- El paquete **Comunicación**, encargado del envío de un diagnóstico al servidor. Este se compone de una única clase, la cual proporciona dicha funcionalidad.

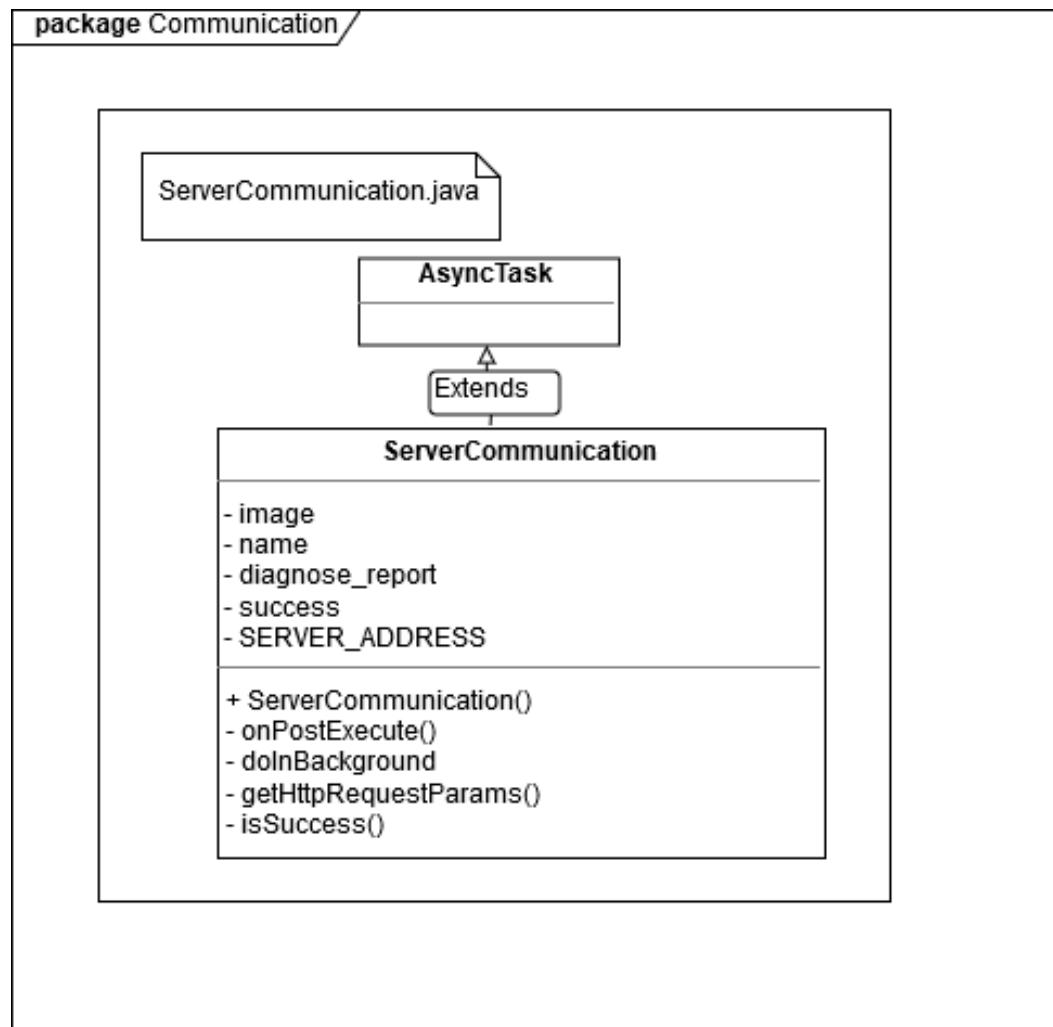


Figura 3.12: Diagrama conceptual del paquete comunicación

### 3.8. Diagramas de algoritmo de detección

Un diagrama que puede resultar interesante es el diagrama de flujo del algoritmo de detección. Este representa las operaciones realizadas sobre la imagen obtenida tras la selección por parte del usuario de la región del iris. En el *Apartado 5* se mostrará, paso a paso, el resultado de cada una de estas operaciones.

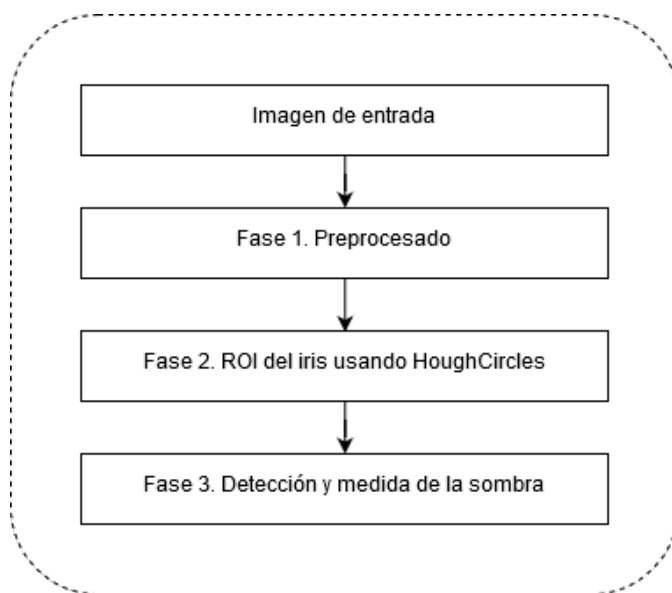


Figura 3.13: Diagrama de flujo del algoritmo de detección general

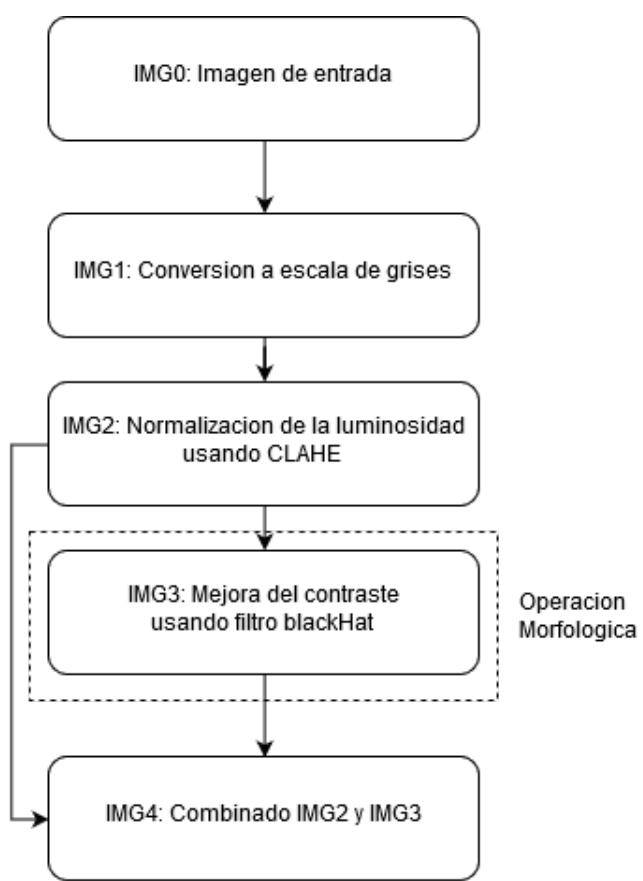


Figura 3.14: Diagrama de flujo del pre-procesado (Fase 1) del algoritmo de detección

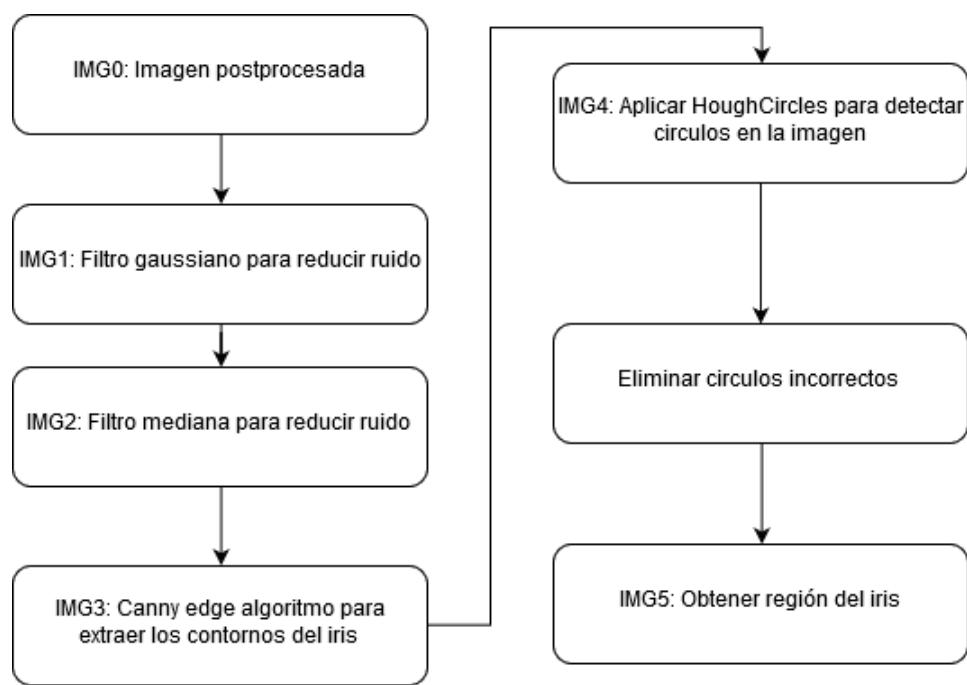


Figura 3.15: Diagrama de flujo de la obtención de la ROI del iris (Fase 2)

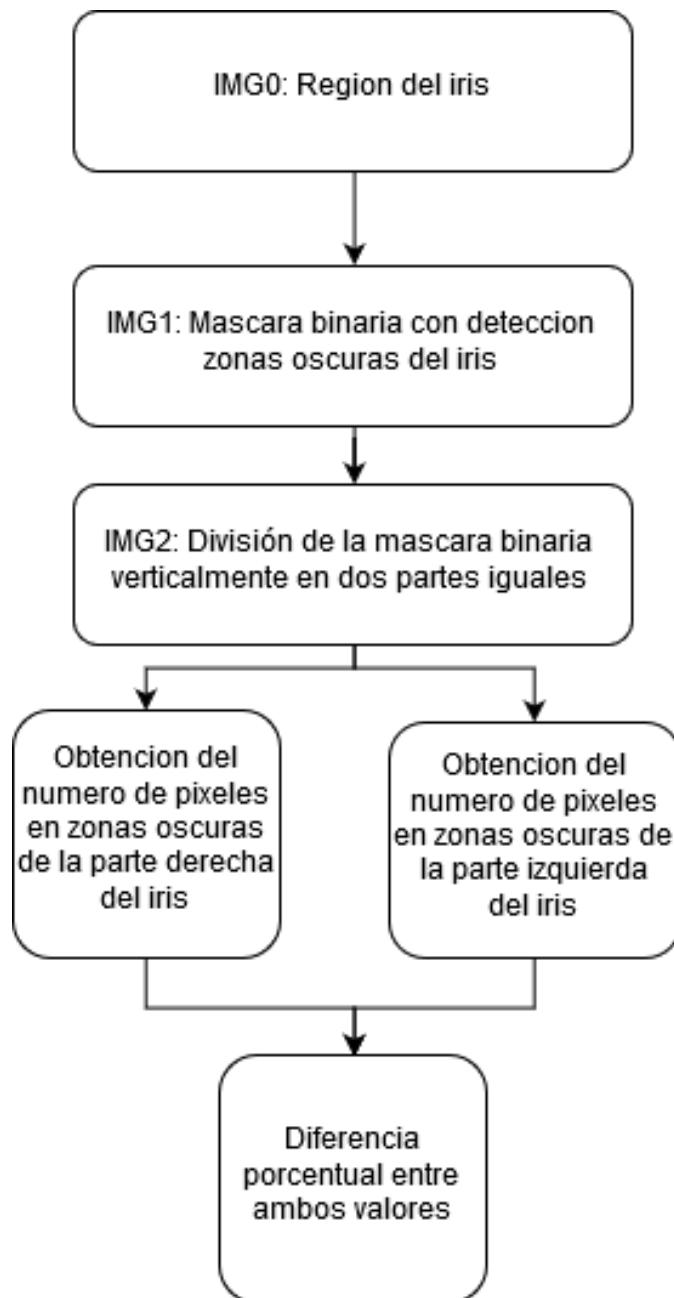


Figura 3.16: Diagrama de flujo de la obtención y medida del tamaño de la sombra (Fase 3)



# Capítulo 4

# Implementación

## 4.1. Recursos empleados

Para la implementación se han utilizado una serie de recursos listados a continuación.

- Ordenador portátil con Windows 10.
- Smartphone con Android superior a 6.0 Marshmallow.
- IDE Android Studio con Android SDK 27.
- Librería de manejo de imágenes OpenCV.
- Paquete Android Image Cropper[7].

## 4.2. Arquitectura del sistema

Nuestro sistema se basa en la metodología Modelo Vista Controlador. El MVC es una forma de pensar el desarrollo de un software a través de dividirlo en tres unidades lógicas: los modelos, los cuales se encargan de recoger y organizar la información recibida; la vista, que básicamente se refiere a lo que el usuario ve por pantalla en cuánto ejecuta la aplicación; y el controlador, que es el que define las funcionalidades presentes en una aplicación.

En nuestro caso concreto, cada una de estas partes está representada por:

- El modelo vendrá determinado por la clase DataDiagnose, la cual define los atributos que un diagnóstico posee en el historial de diagnósticos.
- La vista es definida por todos los ficheros XML que definen los Layouts, menús, botones, animaciones etc. de nuestra aplicación.

- El controlador se implementa a través de los métodos asociados a la acción que resulta de seleccionar cada objeto interaccionable de la vista, ya sea pulsar un botón, desplegar un menú de navegación, cerrar un cuadro de texto...

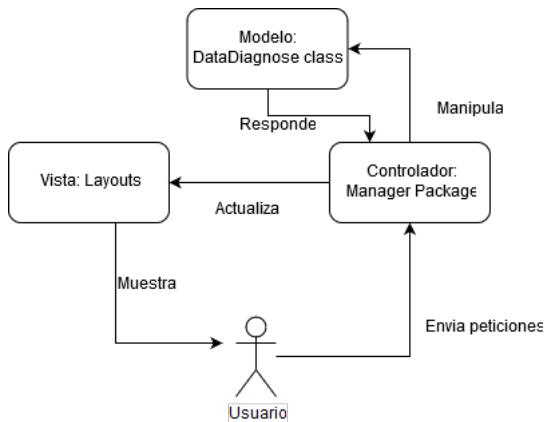


Figura 4.1: Modelo vista controlador para el caso concreto de nuestro sistema

Además, este modelo se encaja dentro de una arquitectura cliente servidor, debido a la comunicación con el servidor que almacena los diagnósticos.

### 4.3. Android Image Cropper

Aunque en un primer lugar se probaron técnicas de localización automática de la región de interés (ROI) como el caso del uso de HarrCascade para la detección de ojos en una imagen dada, su poca precisión y su alto costo computacional hizo que al final se optara por pedir al usuario la selección de la ROI manualmente.

La librería Android Image Cropper[7] ofrece todo lo necesario para la petición al usuario de una imagen con la que realizar el diagnóstico. Algunas de sus características son:

- Petición al usuario del método de entrada de la imagen.
- Rotación/volteo de imagen en el menú de recorte.
- Auto zoom-in/out sobre el área a recortar.
- Establecer un número máximo/mínimo de la imagen recortada.
- Establecer la localización inicial de la ventana de recorte.

Debido a estas características el uso de la misma es más que conveniente para implementar la fase previa a la detección.

## 4.4. Implementación

Para ofrecer una mejor comprensión del funcionamiento del sistema, en esta sección se explicarán las características más relevantes en cuanto a la implementación de cada una de las funciones del mismo.

### 4.4.1. Vista Diagnóstico

Esta vista es la principal en el sistema. Se encarga de proporcionar al usuario todo lo necesario para la introducción de la imagen a evaluar. Además, es la vista por defecto una vez iniciada la aplicación. Su apariencia es:



Figura 4.2: Vista diagnóstico por defecto

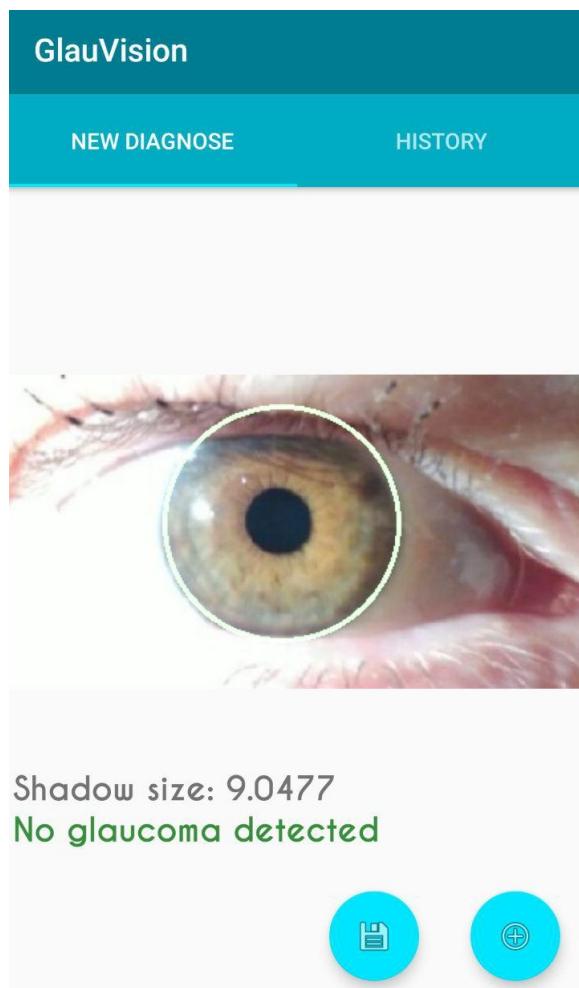


Figura 4.3: Vista diagnóstico con resultado

Como podemos ver, en ella existen básicamente cuatro componentes que forman nuestro Layout: la imagen de la detección, el campo de texto tamaño de sombra, el campo con la decisión obtenida y los botones flotantes de nuevo diagnóstico y guardar diagnóstico.

En concreto, el método encargado de actualizar esta vista una vez obtenido el resultado del diagnóstico es el método `onResume`, el cual es llamado por defecto por Android cada vez que obtenemos el resultado del algoritmo de detección justo después de que el usuario seleccione el área en la que se encuentra el iris. El funcionamiento concreto del mismo es:

Fragmento de código 4.1: Actualización de la vista nuevo diagnóstico

```
1 image = getDetectionResult();
2 size = getSizeShadowResult();
3
4 if(size == -1.0){
5     hideResultView();
6     showDefaultView();
7 }
8 else{
9     hideDefaultView();
10    showResultView();
11
12    size = roundDoubleDecimals();
13    field_shadow_size.setText("Shadow size: " + size);
14 }
15 }
```

Cabe destacar que las funciones `hideDefaultView()` y `showResultView()` son las encargadas de ocultar o mostrar los elementos dependiendo de si se ha producido un diagnóstico o no.

Por último, con respecto a la decisión sugerida al usuario, esta ha sido determinada con solamente cuatro imágenes de pacientes con glaucoma, por lo que es bastante probable que estos no sean validos. Aun así, los rangos son:

- **No glaucoma:** tamaño de la sombra menor que 12.
- **Diagnóstico dudoso:** tamaño de la sombra entre 12 y 15.
- **Glaucoma detectado:** tamaño de la sombra mayor que 15.

#### 4.4.2. Vista Historial

La otra vista a la que el usuario puede hacer uso es la de consulta de los diagnósticos guardados. El objetivo de esta es ofrecer un espacio en el que consultar de nuevo el resultado de algún diagnóstico que haya resultado interesante para el usuario.

Por defecto, esta se presenta vacía y se va actualizando conforme el usuario guarda diagnósticos. Por tanto, los dos casos posibles de esta vista son:

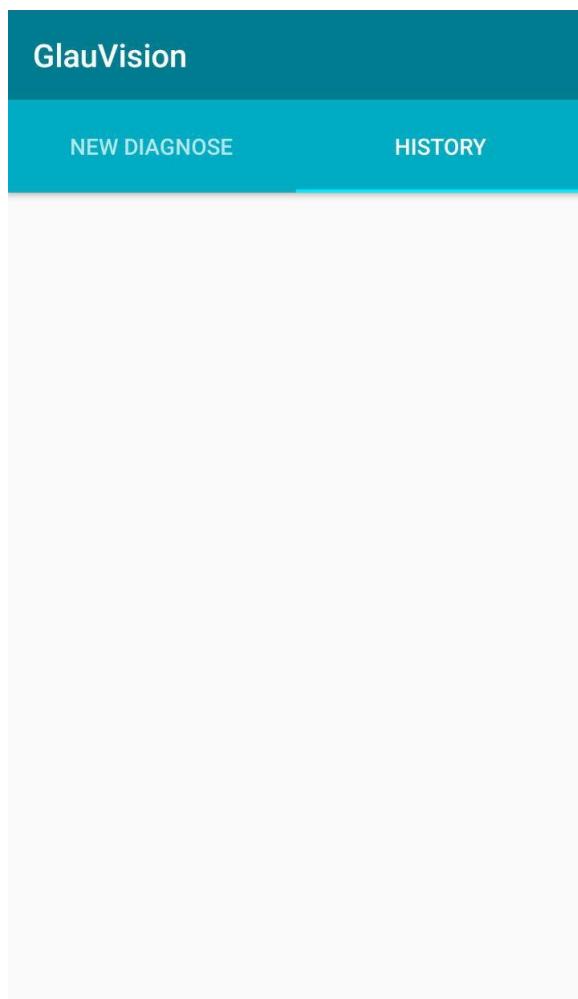


Figura 4.4: Vista historial por defecto

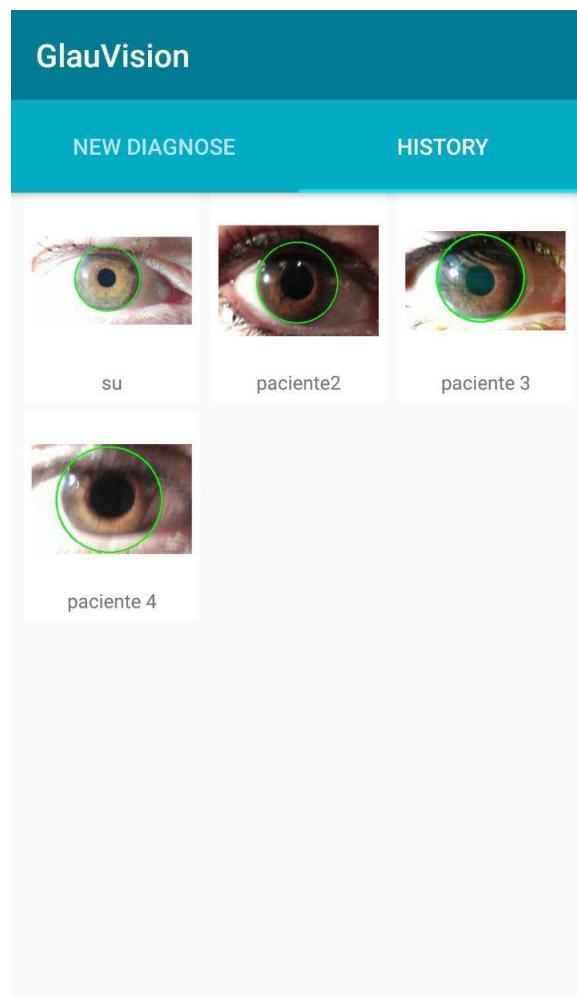


Figura 4.5: Vista historial con resultados

En este caso, la vista historial posee un GridView, un tipo de representación determinada que Android ofrece para mostrar datos en una cuadrícula o rejilla. Para construir esta vista, en primer lugar, se especifica la apariencia de cada cuadrícula y los datos que esta almacenara. Después, se actualiza la vista del historial añadiendo una entrada por cada diagnóstico guardado.

Un aspecto interesante de la implementación de ese apartado es la forma en la que se realiza la actualización de la rejilla, especificada en el método `setUserVisibleHint`, que se ejecuta cada vez que el usuario selecciona la ventana “History”:

## Fragmento de código 4.2: Actualización del historial

```

1 public void setUserVisibleHint(boolean isVisibleToUser) {
2     super.setUserVisibleHint(isVisibleToUser);
3     //Si el usuario selecciona la pestaña historial
4     if (isVisibleToUser) {
5         try {
6             //Obtenemos los datos de una rejilla y la mostramos
7             gridAdapter = new GridViewAdapter(this.getContext(), R.layout.
8                 grid_item_layout, getData());
9         } catch (FileNotFoundException e) {
10             e.printStackTrace();
11         }
12         gridView.setAdapter(gridAdapter);
13         //Especificamos la acción generada al pulsar una rejilla
14         gridView.setOnItemClickListener(new AdapterView.OnItemClickListener
15             () {
16             public void onItemClick(AdapterView<?> parent, View v, int
17                 position, long id) {
18                 ImageItem item = (ImageItem) parent.getItemAtPosition(position);
19                 //Enviamos datos a la vista detalles
20                 Intent intent = new Intent(getActivity(), DetailsActivity.class);
21                 intent.putExtra("title", item.getTitle());
22                 intent.putExtra("image", item.getImage());
23                 intent.putExtra("date", item.getDate());
24                 intent.putExtra("shadow", item.getSizeShadow());
25             }
26         });
27     }

```

En el código anterior se pueden observar dos particularidades:

- La forma en la que esta vista obtiene los datos de una rejilla es a través del método `getData`, el cual recibe los resultados de un diagnóstico desde la clase *MainActivity* en el paquete *Manager*.
- Se define una acción para cuando se selecciona un elemento de la rejilla. Esta acción concretamente disparará una nueva vista que mostrará los detalles del diagnóstico seleccionado:

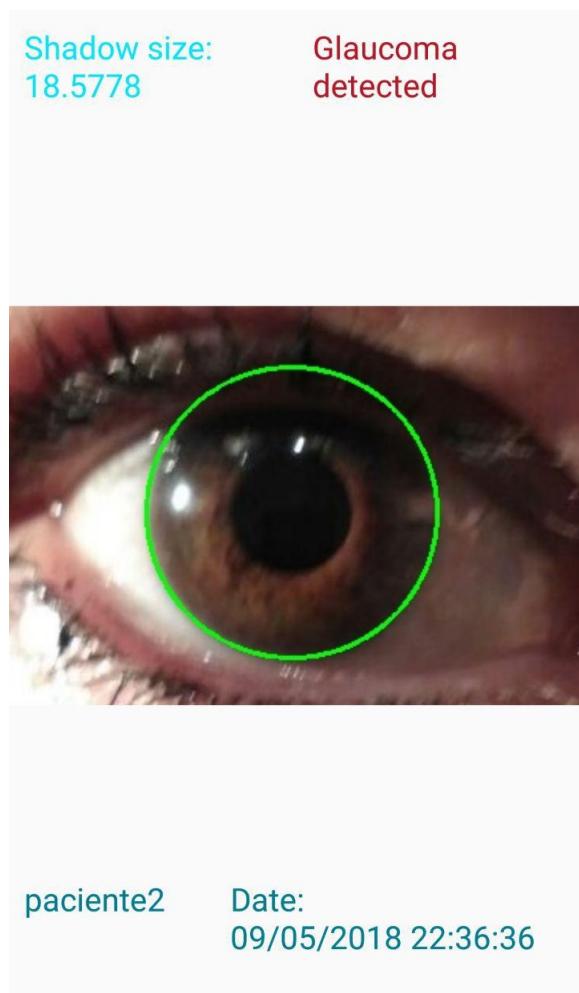


Figura 4.6: Vista detalles de un diagnóstico guardado

#### 4.4.3. Comunicación con el servidor

Otro apartado importante que da sentido al objetivo de la aplicación es la capacidad de enviar diagnósticos a un servidor. Para la implementación de tal comunicación se han seguido los siguientes pasos:

#### 1. Adquisición del servidor de almacenamiento remoto:

El primer paso para establecer tal comunicación ha sido obtener un proveedor de un servidor privado con el fin de poseer un sistema con alto tiempo medio entre fallos para así conseguir la mayor disponibilidad posible. Para ello se ha hecho uso de la plataforma 000webhost, la cual nos ofrece este servicio de forma gratuita; ideal para probar nuestro sistema.

## 2. Configuración del servidor:

Al tratarse de un proceso de comunicación sencillo (los dos únicos datos a almacenar serán la imagen resultante y un informe sobre el diagnóstico de la misma) con un simple archivo PHP nos basta para concretar el funcionamiento del back-end:

Fragmento de código 4.3: Fichero PHP para la comunicación con el servidor

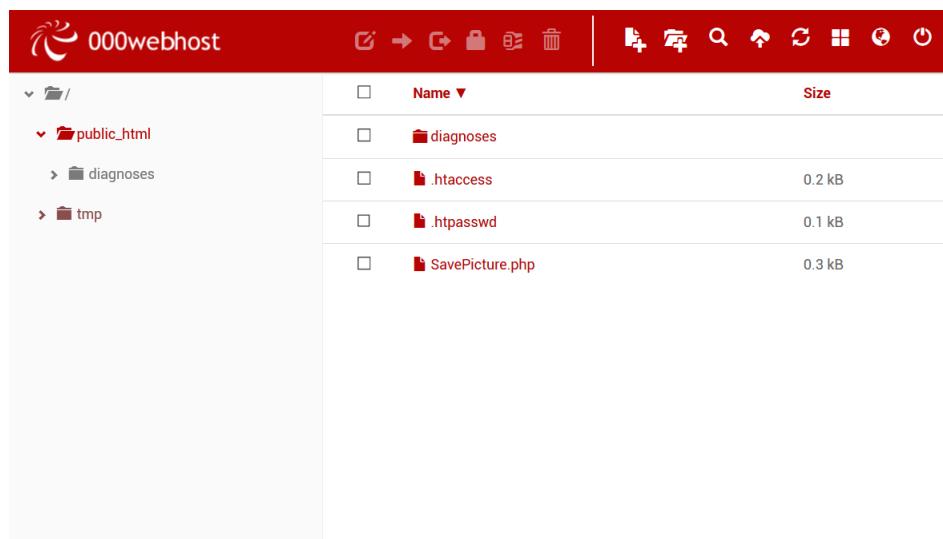


Figura 4.7: Configuración directorio servidor

Como se puede observar, el funcionamiento es sencillo: recibimos tres parámetros: el nombre del informe, la imagen y los datos del informe. A continuación, almacenamos estos últimos dos valores en el directorio remoto “diagnoses”, ambos con el nombre proporcionado.

### 3. Establecimiento de la comunicación app-servidor:

La implementación de esta parte se realiza, como ya se comentó en el *Capítulo 4*, a través de la clase `ServerCommunication`.

En concreto, mediante el uso del modulo “Http” la librería “Apache”, Android proporciona todo lo necesario para la definición de los parámetros necesarios para la comunicación con el servidor como tiempos de espera, dirección del servidor o archivo PHP a ejecutar cuando se establece la comunicación.

Una descripción en pseudocódigo de la ejecución de esta comunicación es:

Fragmento de código 4.4: Pseudocódigo de del envío del diagnóstico

```
1     Compresión de la imagen del diagnóstico
2     Codificación en base64 de la imagen
3
4     Nuevo array datos_a_enviar
5     datos_a_enviar.add(imagen)
6     datos_a_enviar.add(nombre)
7     datos_a_enviar.add(diagnóstico_reporte)
8
9     Establecimiento de dirección del servidor
10    Establecimiento tiempos de espera
11
12    Definición nombre archivo PHP a ejecutar
13
14    try{
15        EnviarDatos
16    }catch (Exception e){
17        MostrarError
18    }
```

Los resultados obtenidos por el servidor se muestran de la siguiente forma:

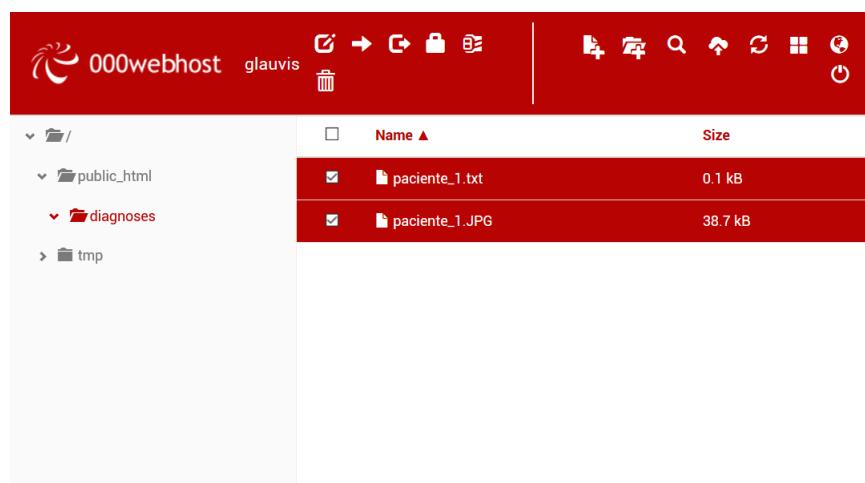


Figura 4.8: Datos diagnóstico con título paciente\_1

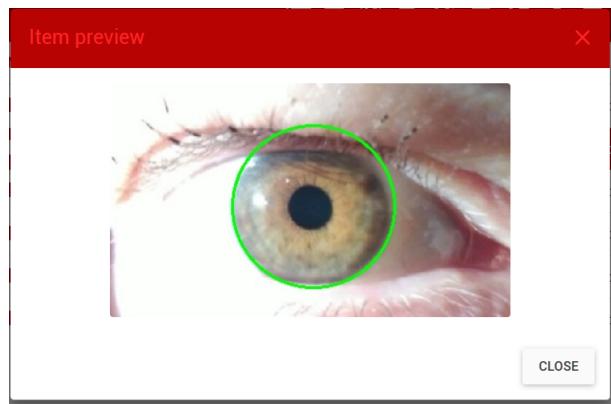


Figura 4.9: Imagen diagnóstico con título paciente\_1

/public\_html/diagnoses/paciente\_1.txt

1 Date: 09/02/2018 17:58:57
2 Size of the shadow: 8.561872482299805
3 Decision: Needs further attention

SAVE & CLOSE      SAVE

A screenshot of a diagnostic report interface. On the left, there is a sidebar with the path "/public\_html/diagnoses/paciente\_1.txt". The main area contains a table with three rows of data. The first row shows the date and time. The second row shows the size of the shadow. The third row shows the decision, which is "Needs further attention". At the bottom right, there are two buttons: "SAVE & CLOSE" and a larger red "SAVE" button.

Figura 4.10: Informe diagnóstico con título paciente\_1

Cabe destacar que, en el caso del informe, el campo “Decision” se ha concretado solamente para exemplificar el resultado de un informe generado. La decisión sobre a partir de qué tamaño de sombra el paciente puede poseer glaucoma llevaría un estudio exhaustivo con una muestra significativa de pacientes.

#### 4.4.4. Algoritmo de detección

Mediante el uso de la librería OpenCV en su versión 3.4 es posible realizar transformaciones, filtrado, detecciones y, en general, un amplio rango de operaciones sobre una imagen.

Tras consultar la bibliografía existente sobre esta librería [5] construí una serie de fases con el fin de pre-procesar, detectar y cuantificar la sombra generada en el iris de una persona al aplicar una luz paralela al plano del mismo.

A continuación, se describen los pasos que realiza tal algoritmo hasta llegar a la medición final. Cabe notar que algunos de los parámetros de las funciones de OpenCV utilizadas (tamaño de *kernels*, *threshold* para la construcción de mascara binaria, rangos de *Canny Edge...*) han sido ajustados a base de prueba y error hasta conseguir el correcto funcionamiento del sistema.

##### ■ Preprocesado

Antes de nada, ante cualquier problema de detección de características en una imagen, es recomendable realizar una serie de operaciones para normalizar las condiciones de luminosidad y eliminar el ruido presente.

A continuación, se muestra un ejemplo de todas las fases que realiza el algoritmo de pre-procesado:

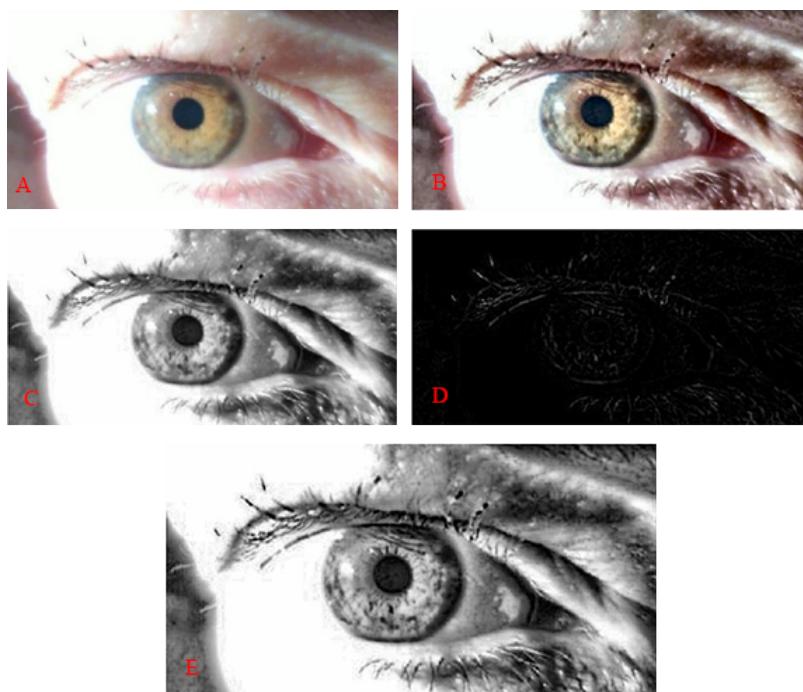


Figura 4.11: Ejemplo del algoritmo de pre-procesado. A)Imagen de entrada, B) Imagen tras aplicar CLAHE en el canal L de la imagen en formato Lab, C) Conversión a escala de grises, D) Filtro blackHat para mejorar contraste y E) Suma de C y D

La inclusión de un método de inicialización de histograma ( CLAHE) en este caso nos brinda la oportunidad de eliminar el ruido que puede incluir el foco de luz aplicado al ojo a la hora de generar la sombra. El hecho de que estemos ante una imagen con una iluminación directa sobre el objeto a detectar hace que, sin una correcta normalización de la luz ambiental, el ruido introducido sea enorme. Con el uso de CLAHE nos aseguramos que, en cierta medida, la luz no afecte en la detección del iris.

La clave del correcto funcionamiento de este algoritmo reside en aplicar la ecualización del histograma sobre el canal L del espacio de colores Lab. Este canal es el que codifica los valores de luminosidad de una imagen en color. A diferencia de otros espacios de colores como RGB o HSV, Lab nos separa la luminosidad sobre el resto de valores. En concreto, la ecualización del histograma sobre el canal L funciona de la siguiente manera:

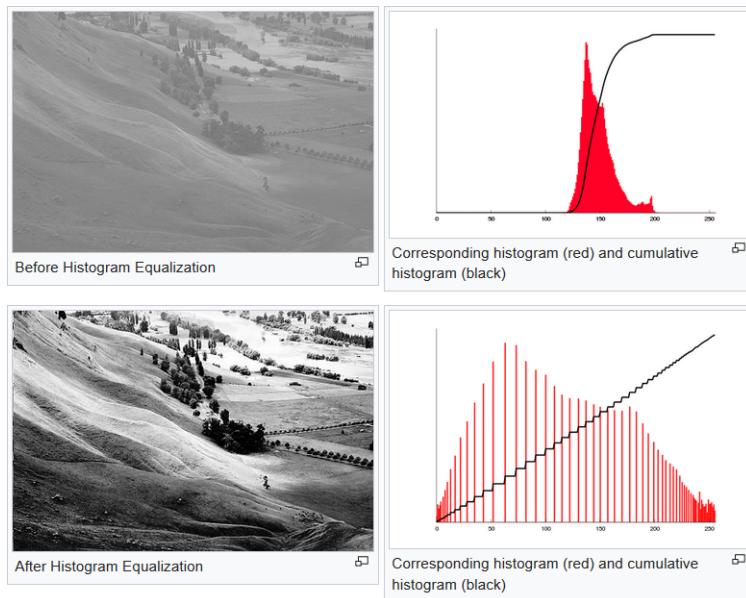


Figura 4.12: Ejemplo del funcionamiento de la ecualización de histograma[10]

A continuación se muestra la implementación del mismo:

#### Fragmento de código 4.5: Algoritmo de ecualización de Histograma

```

1 // Leer la imagen en RGB y convertirla en formato Lab
2 Mat channel = new Mat();
3 Imgproc.cvtColor(srcArry, dstArry, Imgproc.COLOR_BGR2Lab)
4 ;
5 // Obtener el canal L
6 Core.extractChannel(dstArry, channel, 0);
7
8 // Aplicar el algoritmo CLAHE sobre el canal L
9 CLAHE clahe = Imgproc.createCLAHE();
10 clahe.setClipLimit(4);
11 clahe.apply(channel, channel);
12
13 // Unir de nuevo el canal modificado con el resto de
14 // canales
15 Core.insertChannel(channel, dstArry, 0);
16
17 // Convertir de nuevo al formato RGB
18 Imgproc.cvtColor(dstArry, dstArry, Imgproc.COLOR_Lab2BGR)
19 ;
20
21 // Eliminar la matriz temporal creada
22 channel.release();

```

Una vez controladas las condiciones de luminosidad, se realiza una operación morfológica a través de un filtro **blackHat** con un *kernel* de

tamaño 5 para aumentar el contraste de la imagen. Finalmente, las dos imágenes (escala de grises y la resultante del filtro `blackHat`) se unen para obtener una imagen mejorada.

- **Detección del iris.**

Teniendo una imagen con el contraste mejorado y la luminosidad medianamente normalizada, es hora de detectar el iris. En primer lugar, eliminamos el ruido de la imagen a partir del uso de dos filtros: el filtro Gaussiano y el mediana. En concreto, el uso del filtro mediana nos ayudará a detectar de manera mas precisa los contornos de la imagen.

En un siguiente paso, la extracción de bordes mediante el algoritmo `Canny Edge` nos descubrirá el contorno donde se sitúa el iris. Por último, al describir el iris una superficie redonda, el correcto uso de la función `HoughCircles` sobre la imagen previamente obtenida devolverá aquellos bordes cuya morfología se aproximan a una circular.

Por tanto, los pasos seguidos en esta fase son:

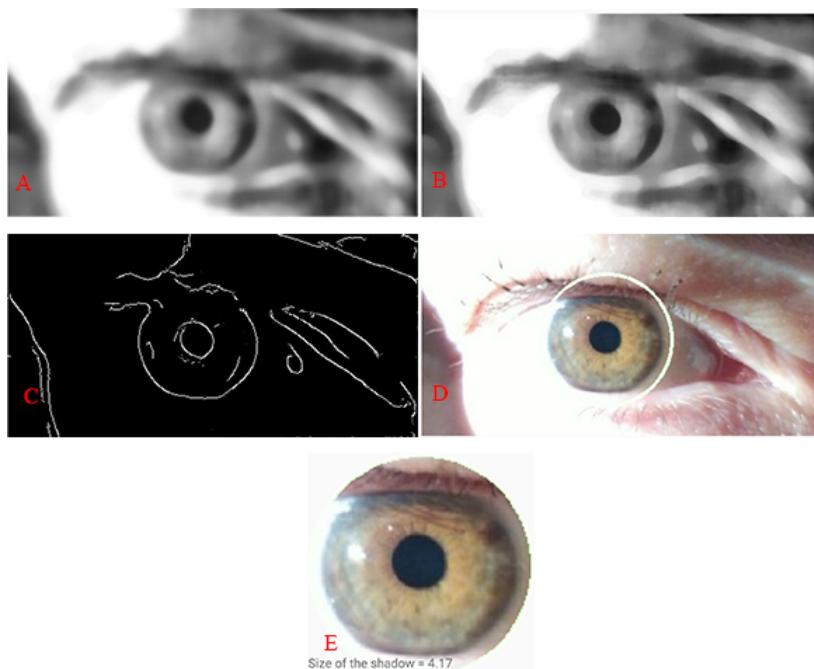


Figura 4.13: Ejemplo del algoritmo de selección del iris  
A)Imagen tras filtro gaussiano, B) Imagen tras filtro mediana, C) Imagen tras Canny Edge, D) Circulo retornado por HoughCircles y E) Región de la imagen dentro del circulo

Cabe mencionar que, tras obtener los círculos detectados por `HoughCirclesP`, se lleva a cabo un proceso de decisión para seleccionar, en caso de haber más de uno, aquel con mayor posibilidad de ser el que marca la región del iris. Este proceso consiste en seleccionar el círculo cuyo centro coincide con una región oscura y este situado lo más cercano al centro de la imagen.

#### ■ Medida de la sombra

Por último, una vez obtenida la región en la que se encuentra el iris, es momento de centrarnos en la detección y medida de la sombra. Los pasos seguidos en este caso son:

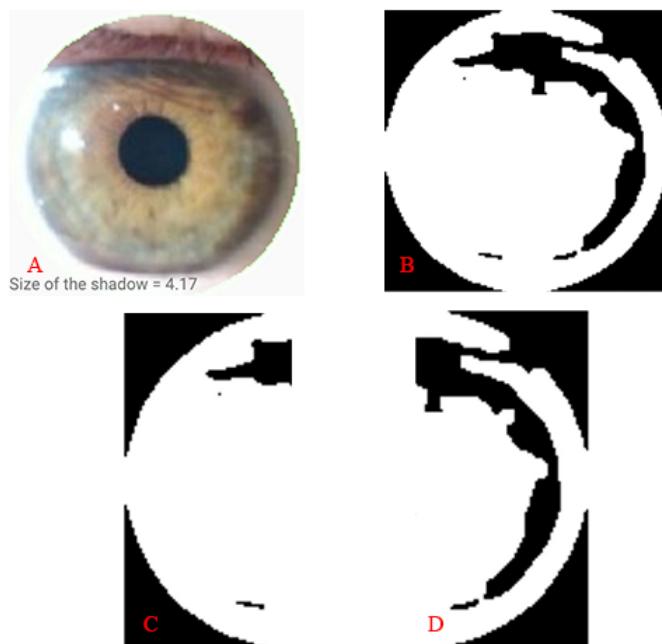


Figura 4.14: Ejemplo del algoritmo de medida de la sombra  
 A)Área del Iris, B) Mascara que indica las zonas oscuras, C) Porción derecha del iris, D) Porción izquierda del Iris

En primer lugar, construimos una máscara binaria en la cual los píxeles cuyo valor sea menor que 100 (escala de grises) los contaremos como regiones con poca luminosidad. Esta máscara nos descubre la región que abarca la sombra generada en el iris.

Con la máscara binaria obtenida, se realiza una operación típica para

la eliminación de ruido de una máscara: la apertura<sup>1</sup>. Esta operación elimina las pequeñas detecciones fruto del posible ruido existente (erosión) y, a continuación, conecta regiones vecinas mediante el agrandado de las zonas previamente detectadas (dilatación):



Figura 4.15: Ejemplo de la operación morfológica de apertura

Obtenida la máscara con ausencia de ruido, es hora de intentar encontrar una forma de, independientemente de la posición en la que este la sombra (independientemente de si es el ojo derecho o el izquierdo) conseguir obtener una información sobre el tamaño de la misma.

Para conseguir esto, se divide verticalmente la máscara en dos partes iguales y se cuenta el número de píxeles con valor 0 (los que marcan las zonas de baja luminosidad) en cada una de las dos mitades.

Para finalizar se aplica la siguiente formula:

$$Sombra = \left| \frac{pxl - pxr}{(pxl + pxr)/2} \right| * 100$$

Donde  $pxl$  representa el número de píxeles con valor 0 en el lado izquierdo y  $pxr$  el número de píxeles con valor 0 en el lado derecho.

Nótese que el resultado obtenido no es más que la distancia entre ambos valores expresada en % y, por tanto, normalizada.

## 4.5. Diseño de las vistas y menús

Debido al avance de las técnicas de diseño en los últimos años, Google ha desarrollado Material Design[8], una normativa de diseño enfocado en la

---

<sup>1</sup>[https://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_imgproc/py\\_morphological\\_ops/py\\_morphological\\_ops.html](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html)

visualización del sistema operativo Android. El principal objetivo del mismo es el de ofrecer una interfaz intuitiva y clara que ofrezca al usuario una experiencia a tal extremo que se olvide de que existe tal interfaz. Con el desarrollo de cualquier aplicación Android se hace casi obligatorio el uso de este estándar.

Todos los menús, vistas, botones, imágenes... intentan por tanto ajustarse a este estándar.

#### 4.5.1. Tema

Los colores elegidos para el tema principal seguirán la paleta de colores *Cyan 50* definida por Material Design:

Cyan 50	#E0F7FA
100	#B2EBF2
200	#80DEEA
300	#4DD0E1
400	#26C6DA
500	#00BCD4
600	#00ACC1
700	#0097A7
800	#00838F
900	#006064
A100	#84FFFF
A200	#18FFFF
A400	#00E5FF
A700	#00B8D4

Figura 4.16: Colores base de la aplicación

#### 4.5.2. Layouts

Nuestro sistema posee los siguientes Layouts, cada uno representando una vista o elemento del mismo:

- **activity\_main.xml:** en el que se especifica el diseño de la barra de título de la aplicación y de las pestañas de navegación.
- **details\_activity.xml:** que especifica los elementos que posee la actividad de vista detallada de un diagnóstico guardado.

- **fragment\_diagnosis.xml:** concreta todo lo relativo a la ventada de nuevo diagnóstico.
- **fragment\_history.xml:** contiene el diseño de la ventana de historial.
- **grid\_item\_layout.xml:** especifica el diseño de una celda de la ventana de historial.

#### 4.5.3. Tabs

Como nuestro sistema únicamente posee dos vistas, según las reglas de MaterialDesign<sup>2</sup> lo más apropiado es hacer uso de pestañas de navegación. Cada pestaña irá acompañada tanto del título representando el contenido de la misma.

---

<sup>2</sup><https://material.io/design/components/tabs.html>



# Capítulo 5

## Pruebas

En el capítulo anterior se describía como se había realizado toda la implementación del proyecto, por lo que ahora es momento de comprobar cómo funciona la detección de nuestro sistema. Como las dos características principales de este son la detección del iris y la medición de la sombra generada en el mismo, las pruebas se enfocarán a evaluar estos dos casos.

Por tanto, se han realizado dos tipos de pruebas del algoritmo de detección: Un primer test consistirá en una comprobación de la detección de interés (el iris) independientemente de la sombra a medir. En segundo lugar, se intentará concretar hasta qué punto, dado un iris, el algoritmo es capaz de diferenciar satisfactoriamente entre un ojo sano y uno con glaucoma.

### 5.1. Precisión de la detección del iris

Las pruebas realizadas para obtener la precisión de la detección de la región del iris se han basado en el establecimiento la siguiente estrategia:

1. Obtención de un conjunto de datos con 15 imágenes de ojos de diferentes tamaños y colores.
2. Recorte de cada imagen fijando diferentes selecciones dentro el área comprendida el lagrimal hasta la v externa.
3. Etiquetado de cada resultado valorando si el resultado selecciona únicamente la región del iris o no.

Los resultados obtenidos de los test son:

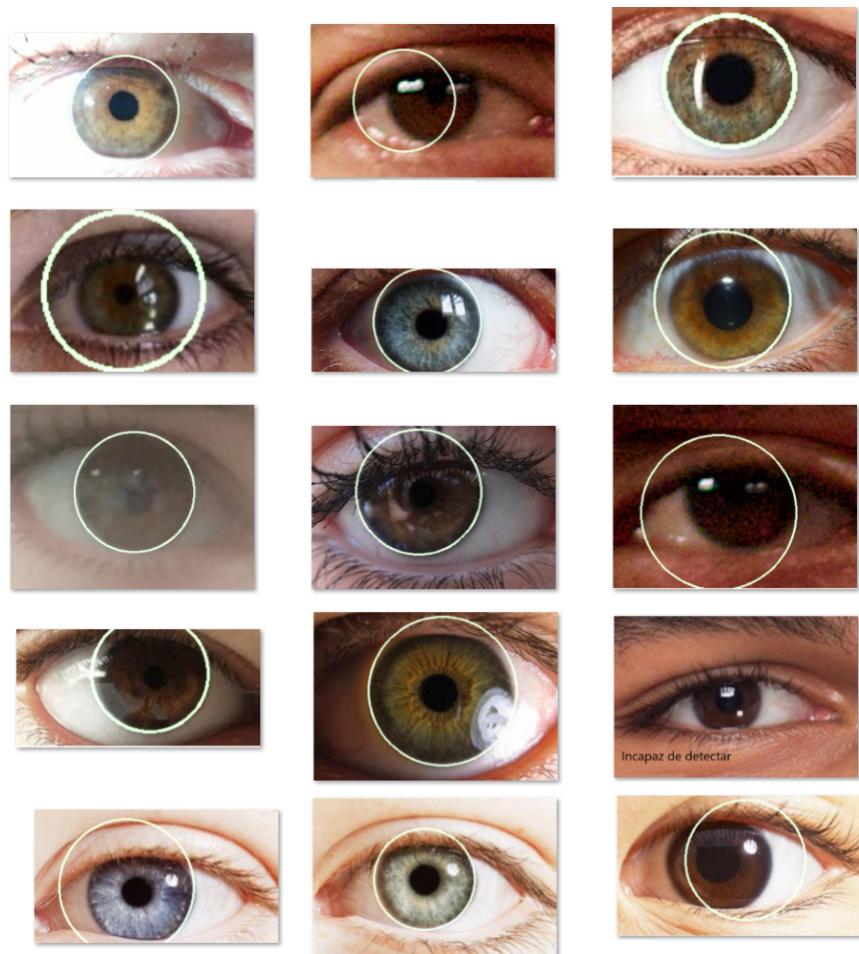


Figura 5.1: Resultados de los test de la detección del iris

Las conclusiones obtenidas a partir de los mismos son varias:

- En condiciones de baja luminosidad y baja resolución el algoritmo de detección presenta falsos positivos con alta probabilidad.
- Los resultados sugieren que el color del iris no supone ningún cambio significativo en las medidas.
- El hecho de largas pestañas (fuente de ruido) no parece cambiar la detección.
- En situaciones en las que el paciente no tiene el ojo completamente abierto la precisión disminuye considerablemente.

- Los reflejos grandes en la pupila pueden alterar el resultado de la detección.
- Como es de esperar, cuanto más se recorte la zona del iris, más probabilidad hay de obtener un resultado satisfactorio.

## 5.2. Precisión de la medida de la sombra

Para la correcta evaluación de la medida del tamaño de la sombra necesitaríamos un conjunto de muestras en el que, al menos, la mitad de ellas representaran a pacientes con diferentes grados de glaucoma. Debido a la dificultad de obtener datos de este tipo la evaluación consistirá en la medida de la sombra de 3 pacientes que presentan glaucoma avanzado frente a otros tres que no lo presentan.

En todos los casos se aplicará una luz paralela al plano del iris, tal y como sería la forma de evaluar la enfermedad.



Figura 5.2: Resultados de los test de medida de la sombra

Los resultados parecen corresponder con los objetivos planteados. A mayor tamaño de la sombra, el valor de la medición del tamaño de la sombra aumenta. Un factor a destacar es que, en condiciones de baja luminosidad (imagen inferior derecha) los resultados podrían originar falsos positivos. Aun así, todas estas conclusiones están sujetas a un estudio con un conjunto

de datos significativo.

Los resultados sugieren que un valor mayor de 15 en el tamaño de la sombra indicaría la necesidad de acudir a un especialista, un valor entre 12 y 15 indicaría un diagnóstico en el cual es difícil tomar una decisión y un valor menor que 12 indicaría que el paciente está sano.

## Capítulo 6

# Conclusiones y trabajos futuros

Durante la realización de este proyecto el aprendizaje y conocimiento obtenido ha sido enorme. El aprender técnicas para la obtención de información a partir de un caso técnico, trabajar mano a mano con un especialista, ampliar mis conocimientos sobre visión por computador, Android, documentación, planificación y un sin fin de ejemplos. Como hablar de todo es imposible, describiré los aspectos que creo que se asemejan más a lo explicado durante esta memoria.

La principal conclusión a la que se puede llegar es que con un dispositivo móvil el cual hoy en día todo el mundo posee se pueden realizar tareas de diagnóstico sencillas con el fin de ayudar al personal sanitario, lo que tiene un impacto directo en nuestra calidad de vida.

Uno de los principales problemas encontrados ha sido a la hora de calibrar correctamente el algoritmo de detección, lo cual me hace consciente de la dificultad de la visión por computador cuando se aplica a campos en los que se requiere un nivel de precisión considerable. A su vez, gracias a la magnífica librería OpenCV, las tareas de optimización, rendimiento e implementación de técnicas consistentes se simplifican muchísimo.

Otra de las conclusiones que he podido sacar de este proyecto es la importancia de poseer conocimientos sobre control de versiones. Incluso para un proyecto de una sola persona, la diferencia que supone la integración de un software de este tipo en cuanto a prevención de perdida de datos y organización de la información es enorme.

La ultima conclusión que me gustaría destacar es sobre la importancia de conocer personas de diferentes disciplinas con las que poder interinar. No es otra cosa si no la capacidad de la informática para solucionar problemas en

cualquier disciplina la que la hace tan valiosa. Por ello, un conocimiento sobre otros campos posibilita el acceso a fuentes de ideas con las que trabajar.

En cuanto a los trabajos futuros, tal y como se ha comentado en el principio de esta memoria, el objetivo de este proyecto es resolver una parte de una idea más grande. Por ello, la cantidad de trabajo que se puede seguir realizando en el sistema es considerable:

- Integrar los resultados del diagnóstico con un sistema médico real basado en un protocolo médico abalado.
- Seguir mejorando el algoritmo de detección tanto del iris como de la sombra hasta, al menos, llegar a un mínimo del 95 % de diagnósticos satisfactorios.
- Seguir trabajando con un médico especializado con el fin de entender la relación entre las escalas actuales de medida del glaucoma y el valor del tamaño de la sombra. De esta manera se podría fijar con mayor seguridad, a partir de qué valor el paciente sufre glaucoma.
- Implementar una vista para el especialista médico a través de la misma aplicación. De esta forma, se construiría un sistema con gestión de usuarios que ofreciera tanto notificaciones a especialistas sobre diagnósticos que requieren su atención como notificaciones al médico general sobre los comentarios del especialista.

# Glosario de términos

**Detección:** es el resultado de aplicar un algoritmo de visión por computador a una imagen con el fin de identificar alguna característica de la misma. En este caso, la detección de la sombra generada en el iris del paciente

**Glaucoma de ángulo cerrado:** anomalía ocular presente en aquellos ojos en los que la salida del humor acuoso (el líquido que produce el ojo) está comprometida al obstruir el iris el ángulo camerular (Ángulo formado entre el iris y la córnea).

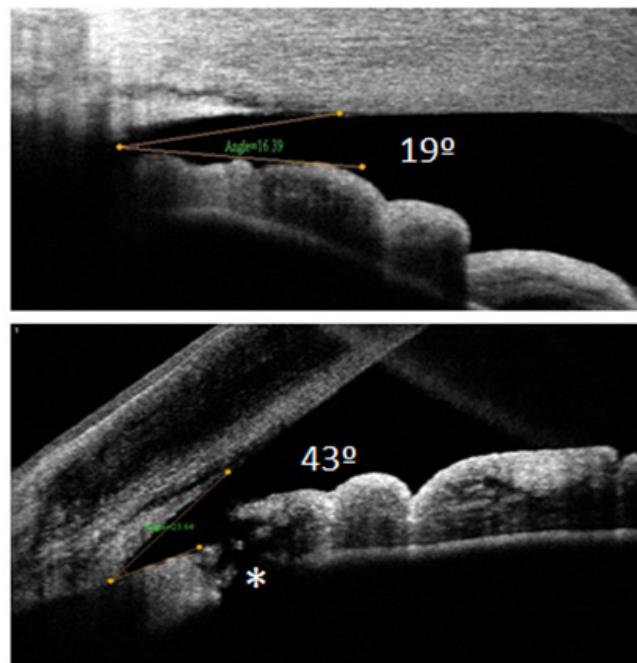


Figura 6.1: Imagen del ángulo camerular tomada con OCT

En la imagen anterior se pueden identificar las estructuras anatómicas y la medida del ángulo, en este caso, 19° en la imagen superior y 43° en la imagen inferior

**Material Design:** Material Design es una normativa de diseño enfocado en la visualización del sistema operativo Android, además en la web y en cualquier plataforma.

**OpenCV:** OpenCV (Open Source Computer Vision) es una librería software open-source de visión artificial y machine learning. Provee una infraestructura para aplicaciones de visión artificial.

**Android SDK:** SDK responde a las siglas Software Development Kit, lo que viene a ser un kit de desarrollo de software. Con él se desarrollan aplicaciones y se ejecutan emuladores del sistema Android de la versión que sea. Todas las aplicaciones Android se desarrollan en lenguaje Java con este kit.

**Android Studio:** Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android.

**Diagnóstico:** resultado de la detección y medida de la sombra del ojo.

**Activity:** un Activity es cada una de las pantallas o vistas que forman una aplicación.

**Fragment:** un Fragment representa un comportamiento o una parte de la interfaz de usuario en una Activity. Puedes combinar múltiples fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en múltiples actividades. Puedes pensar en un fragmento como una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando (algo así como una "subactividad" que puedes volver a usar en diferentes actividades).

**Action Bar:** la barra de app, también conocida como barra de acciones, es uno de los elementos de diseño más importantes en las actividades de una app, ya que proporciona una estructura visual y elementos interactivos conocidos por los usuarios. El uso de la barra de app permite, por ejemplo, la alineación de la app con otras apps para Android.

**Layout:** un diseño (o layout) define la estructura visual para una interfaz de usuario, como la IU para una actividad o widget de una app.

**Java:** Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA,

o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra.

**XML:** XML, siglas en inglés de eXtensible Markup Language, traducido como "Lenguaje de Marcado Extensible." "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible.

**LaTeX:** sistema de composición de documentos que permite crear textos en diferentes formatos (artículos, cartas, libros, informes...) obteniendo una alta calidad en los documentos generados.

**Módulo:** fragmento de un programa desarrollado para realizar una tarea específica.



# Bibliografía

## **Libros, artículos e información digital consultada:**

- [1] “Video Abstract: Project InnerEye – Assistive AI for Cancer Treatment”. <https://www.microsoft.com/en-us/research/video/video-abstract-project-inner-eye/>
- [2] “Dermatologist-level classification of skin cancer”. <https://cs.stanford.edu/people/esteva/nature/>
- [3] “EL OCULISTA”. <https://www.clinicavaldearenas.eloculista.es/>
- [4] “ VALORACIÓN DE LA PROFUNDIDAD DE LA CÁMARA ANTERIOR ILUMINACIÓN PARALELA”. Manolo Valdearenas, 18/09/2013. [https://oftalmologia.eloculista.es/index.php?option=com\\_k2&view=item&id=126:valoraci%C3%B3n-del-%C3%A1ngulo-camerular-iluminaci%C3%B3n-paralela](https://oftalmologia.eloculista.es/index.php?option=com_k2&view=item&id=126:valoraci%C3%B3n-del-%C3%A1ngulo-camerular-iluminaci%C3%B3n-paralela)
- [5] ‘Learning Opencv 3’. Adrian Kaehler & Gary Bradski, 15/12/2017. ISBN:978-1-491-93799-0
- [6] Atlas of Glaucoma, Third Edition - Neil T. Choplin & Carlo E. Traverso

## **Páginas consultadas para el desarrollo de la app**

- [7] Android Image Cropper. <https://github.com/ArthurHub/Android-Image-Cropper>
- [8] MATERIAL DESIGN. <https://material.io/design/>
- [9] Documentación Android. <https://developer.android.com/topic/libraries/support-library/?hl=es-419>
- [10] [https://en.wikipedia.org/wiki/Histogram\\_equalization](https://en.wikipedia.org/wiki/Histogram_equalization)

- [11] Documentación OpenCV. <https://docs.opencv.org/3.4.3/>
- [12] Java. <https://www.doc.ic.ac.uk/csg-old/java/jdk6docs/>
- [13] pyimagesearch. <https://www.pyimagesearch.com/>
- [14] learnopencv. <https://www.learnopencv.com/>

### Otro material

- Diversas consultas al sitio Stack OverFlow.
- Diversas consultas a tutoriales de YouTube.
- Material docente de las asignaturas **Fundamentos de Ingeniería del Software**, **Nuevos paradigmas de interacción y Visión por computador** impartidas en **Grado en Ingeniería Informática** en la **Universidad de Granada**.

