UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

APLICACIÓN ANDROID PARA EL SEGUIMIENTO DE POTENCIALES MELANOMAS

ANDROID APPLICATION TO TRACK POTENTIAL MELANOMAS

Realizado por
Raúl López Rueda

Tutorizado por
Enrique Domínguez Merino

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, JUNIO DE 2020

Resumen

El melanoma es el tipo de cáncer de piel más peligroso debido a su facilidad para extenderse a otras zonas del cuerpo y a su alta mortalidad. La primera señal de que un melanoma puede estar desarrollándose en la piel es el cambio de tamaño, color o forma en un lunar, y es por ello que resulta de vital importancia realizar un seguimiento de los lunares que se tengan en el cuerpo para poder detectarlo de forma precoz, aumentando la probabilidad de supervivencia y minimizando las complicaciones.

Es aquí donde se presenta el problema, ya que resulta muy complicado detectar ciertos cambios menores en los lunares a simple vista, a lo que se le suma la complejidad de memorizar a lo largo del tiempo cada una de las características citadas de cada uno de los lunares que tenemos.

Este proyecto pretende ofrecer una solución, utilizando la visión por computador para realizar el seguimiento de los lunares de una manera sencilla y cómoda para el usuario, que solo tendrá que preocuparse de sacar algunas fotografías de sus lunares a lo largo del tiempo. La aplicación se encargará de analizar cada una de estas fotografías para extraer de ellas los cambios que se hayan podido producir relativos al tamaño, el color o la forma.

Palabras clave: Android, APP, cáncer, melanoma, lunar.

Abstract

Melanoma is the most dangerous type of skin cancer due to its ease of spreading to other body areas and its high mortality. The first sign of the development of a melanoma on the skin is the change in size, colour or shape in a mole, and that is why it is of crucial importance to keep track of the body moles, being able to detect it at an early stage, increasing the probability of survival and minimizing complications.

This is where the problem arises, since it is very difficult to detect certain minor changes in moles with the naked eye, to which is added the complexity of memorizing over time each of the specified characteristics of each mole we have.

This project aims to offer a solution, using computer vision to track moles in a simple and comfortable way for the user, who will only have to worry about taking some photos of their moles over time. The application will analyze each one of these photographs to extract from them any changes that may have occurred regarding size, colour or shape.

Keywords: Android, APP, cancer, melanoma, mole.

Agradecimientos

Agradecer a Enrique Domínguez Merino por su tutorización, su implicación y su paciencia. Sin él este proyecto no hubiera sido posible.

También deseo agradecer a Greta Briongos Parejo por su apoyo incondicional y su ayuda, tanto en el ámbito personal como en lo relativo a este proyecto.

Agradecer el apoyo y la amistad demostrada en todo momento durante tantos años a Gabriela Ruíz Tueros.

Por último, agradecer a mi familia, por su preocupación constante por mi formación académica y por el apoyo económico.

Índice

Índice de figuras

Índice de tablas

1

Introducción

1.1 Motivación

Cada año se detectan aproximadamente 3 millones de casos de cáncer de piel a nivel mundial, correspondiéndose con un tercio de todos los diagnósticos de cáncer [1]. Aproximadamente 132.000 de esos casos son de tipo melanoma, y aunque puede apreciarse que suponen un pequeño porcentaje de todos los cánceres de piel detectados, son los más peligrosos, porque crecen más rápido y tienen mayor probabilidad de propagarse a otras partes del cuerpo [2].

El seguimiento de los lunares que se pueden encontrar en la piel de las personas puede ser muy importante a la hora de detectar de forma temprana un cáncer de piel, por lo que muchos doctores recomiendan hacerlo en casa al menos con una frecuencia de una vez al mes [3]. En muchas ocasiones, la evolución de estos lunares es paulatina, por lo que la mayoría de las personas no son capaces de recordar con exactitud cómo era cada uno de ellos tiempo atrás, haciendo que sea prácticamente irreconocible el cambio de tamaño, color o forma en caso de que ocurra.

Gracias a la Inteligencia Artificial, el reconocimiento de imágenes mediante la utilización de software puede ser una alternativa para realizar el seguimiento de los lunares en la piel de una forma cómoda y precisa, aprovechando los algoritmos existentes hoy en día, que son capaces de detectar cualquier cambio entre dos imágenes tomadas con márgenes de error muy pequeños.

1.2 Objetivos

El objetivo principal de este Trabajo de Fin de Grado (TFG) es desarrollar una aplicación soportada por dispositivos Android que permita al usuario realizar un seguimiento de los lunares que tenga en distintas partes del cuerpo, mediante la toma de fotografías del mismo lunar en distintos momentos espaciados de tiempo, que serán comparadas para estimar los cambios de forma, tamaño o color que se hayan producido en dicho lunar. Para lograrlo, es necesario cumplir ciertos objetivos más concretos:

a) Realizar un estudio previo sobre los cánceres de piel, profundizando especialmente en los de tipo melanoma.
b) Estudiar y seleccionar las diferentes tecnologías y herramientas a utilizar durante el desarrollo.
c) Desarrollar los distintos prototipos de la aplicación hasta llegar a la aplicación final, creando nuevas funcionalidades para cada uno de ellos y acoplando estas a las anteriores.
d) Realizar pruebas de la aplicación en diferentes dispositivos y estudiar los resultados para realizar cambios o mejoras si es necesario.

1.3 Estructura de la memoria

La memoria está dividida en dos partes. La parte principal está estructurada por capítulos y explica las distintas fases de desarrollo software realizadas en el proyecto abarcado:

1. Introducción: se realiza una explicación de la motivación y los objetivos principales del proyecto.
2. Marco teórico: se expone el estudio teórico realizado sobre el cáncer de piel, con un énfasis especial en el de tipo melanoma. También se expone el estudio teórico realizado sobre la visión por computador, y de cómo utilizarla para hacer seguimiento de los lunares en la piel.
3. Especificación y análisis: se estudian diferentes metodologías de desarrollo a seguir y se escoge una de ellas. Se definen además los requisitos funcionales y los no funcionales, así como las tablas de los casos de uso asociados.

4. Tecnologías y herramientas: se hace un estudio de las diferentes tecnologías y herramientas que se han utilizado para desarrollar la aplicación.
5. Diseño: se define la estructura del proyecto de la aplicación, utilizando diagramas para facilitar la comprensión del mismo. También se definen las diferentes etapas de desarrollo del proyecto.
6. Implementación: se explican las diferentes clases codificadas para crear la aplicación, desde las clases que interactúan de forma directa con el usuario hasta las clases de apoyo o de cálculos más complejos.
7. Verificación y pruebas: se exponen las pruebas realizadas, los errores encontrados y las soluciones que se le han dado.
8. Conclusiones y líneas futuras: se analiza a nivel personal el desarrollo del trabajo realizado, mostrando además las posibles mejoras e ideas que podrían desarrollarse en un futuro a partir de este.

La otra parte de la memoria consta de dos anexos destinados a los usuarios de la aplicación:

- Manual de instalación, donde se explica paso por paso cómo se puede instalar la aplicación en un *Smartphone* con sistema operativo *Android*.
- Manual de usuario, donde se explica paso a paso cuáles son las funcionalidades de la aplicación y cómo utilizarlas.

La memoria contiene además un índice de contenido, un índice de figuras, un índice de tablas y un índice de referencias.

2

Marco teórico

2.1 Lunares

2.1.1 Lunares comunes

Los lunares comunes o nevos son tumores pequeños en la piel que se producen por el crecimiento agrupado de melanocitos, las células que dan coloración a la piel produciendo melanina. Pueden aparecer en cualquier parte del cuerpo, aunque principalmente pueden verse en el torso, los brazos y las manos. Los adultos tienen de media entre 10 y 40 lunares en todo su cuerpo, y hay personas especialmente propensas a desarrollarlos [4].

La mayoría de los lunares se encuentran en el cuerpo al nacer, pero pueden desarrollarse más tarde, generalmente en la etapa de la niñez, aunque su desarrollo sigue hasta aproximadamente los 40 años. En las personas que superan esta edad, muchos de los lunares tienden a ir desapareciendo.

2.1.2 Aspecto de los lunares comunes

El diámetro máximo habitual de los lunares normales es de 6 milímetros, aunque existen algunos que se salen de esta norma. Su forma suele ser prácticamente circular u ovalada, con un borde definido que hace que se distinga fácilmente del resto de piel.

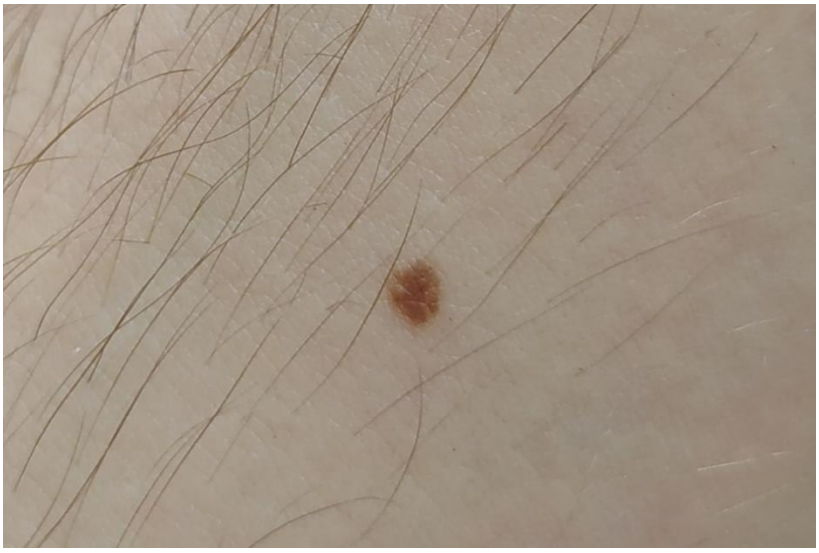

Fig. 1: Lunar común en el brazo izquierdo. Fuente: Elaboración propia.

Los lunares tienden a ser de colores que se mueven en el espectro de los tonos marrones o rosados. Generalmente su superficie es plana, aunque en algunas ocasiones pueden tener forma de cúpula [4].

2.1.3 Nevos displásicos

Existen lunares que, a pesar de ser diferentes a los lunares comunes, no son cancerosos. Son denominados nevos displásicos o lunares atípicos, y pueden ser más grandes que los lunares habituales y con color, superficie o bordes diferentes a lo habitual.

Fig. 2: Nevo displásico con un diámetro superior a 10 milímetros. Fuente: NIH.

2.2 Melanomas

2.2.1 Introducción: cáncer de piel

El cáncer de piel es el tumor más frecuente en el ser humano. Existen tres tipos principales: carcinoma de células escamosas, carcinoma de células basales y melanoma. Este último es el menos común de los tres, pero debido a su facilidad para invadir el tejido cercano y propagarse por el resto del cuerpo, es considerado el más peligroso, causando la mayoría de muertes producidas por cáncer de piel [5].

Fig. 3: El cáncer de piel. Fuente: AECC.

2.2.2 ¿Qué es el melanoma?

Como se ha mencionado en el apartado anterior, el melanoma es el tipo de cáncer de piel más agresivo que existe. Su desarrollo se debe al crecimiento descontrolado de los melanocitos, que acaban creando tumores que usualmente son de color negro o marrón oscuro, debido a que, por lo general, la producción de melanina no se detiene en las células que se ven afectadas por la enfermedad [6]. No obstante, en algunas ocasiones dejan de producirla, causando tumores de colores claros.

2.2.3 Causas del melanoma

La causa principal del melanoma, al igual que la del resto de cánceres de piel, es la radiación ultravioleta que recibe la piel al exponerse de forma inadecuada al Sol u otras fuentes que produzcan este tipo de radiación. Existen además diversos factores de riesgo a tener en cuenta [7]:

- **Edad:** la frecuencia es mayor en personas que se encuentran en el rango de entre los 30 y los 60 años.
- **Tipo de piel:** el riesgo de padecer este tipo de cánceres aumenta en las personas de piel, pelo y ojos claros.
- **Antecedentes familiares:** el melanoma puede ser hereditario si es causado por mutaciones genéticas. Alrededor de un 10% de los afectados por este tipo de cáncer tiene antecedentes familiares.
- **Antecedentes personales:** es más probable desarrollar un cáncer en la piel si ya se ha padecido uno, porque los daños que sufren las células al exponerse demasiado a la luz ultravioleta suelen ser irreversibles.

2.2.4 Detección precoz del melanoma

Frecuentemente, el cáncer de piel del tipo melanoma puede ser detectado de forma precoz, aumentando así considerablemente la probabilidad de que este pueda ser curado, y por extensión, la supervivencia de la persona afectada. Por este motivo, desde el ámbito oncológico se recomienda realizar un seguimiento de los lunares o anchas en la piel por todo el cuerpo, haciendo un énfasis especial en aquellos que presenten cambios de color, tamaño o forma, que parezcan diferentes al resto de lunares, que sean asimétricos o con bordes irregulares, que sean demasiado grandes (más de 6mm de diámetro), ásperos o escamosos, o si presentan algún síntoma alarmante como el sangrado [8].

Todo esto queda recogido en las famosas “reglas del ABCDE”.

Característica	Lunar común	Melanoma
A (Asimetría)	La forma es simétrica o prácticamente simétrica.	Una mitad es diferente de la otra.
B (Borde)	Los bordes son regulares y claramente definidos.	Los bordes son irregulares y no tienen una definición demasiado clara.
C (Color)	El color es de tonos marrones o rosados y uniforme.	Existe una mezcla de tonos marrones, negros, rojos, azules o blancos.
D (Diámetro)	Menor de 6 milímetros.	Mayor de 6 milímetros.
E (Evolución)	No se producen cambios a lo largo del tiempo.	Se producen cambios de color, forma o tamaño, en ocasiones desarrollando hemorragias y sensibilidad anormal.

Tabla 1: El “ABCDE” del melanoma. Fuente: Elaboración propia.

2.3 Visión por computador

2.3.1 Introducción a la visión por computador

La visión por computador es un campo de la inteligencia artificial cuyo objetivo es desarrollar y entrenar sistemas capaces de interpretar y entender imágenes en tiempo real o tomadas previamente. En el ámbito de la ingeniería, se utiliza para comprender y automatizar las tareas que normalmente realiza el ojo humano. Usualmente se apoya en técnicas de aprendizaje automatizado (del inglés, *machine learning*) para mejorar sus algoritmos [9].

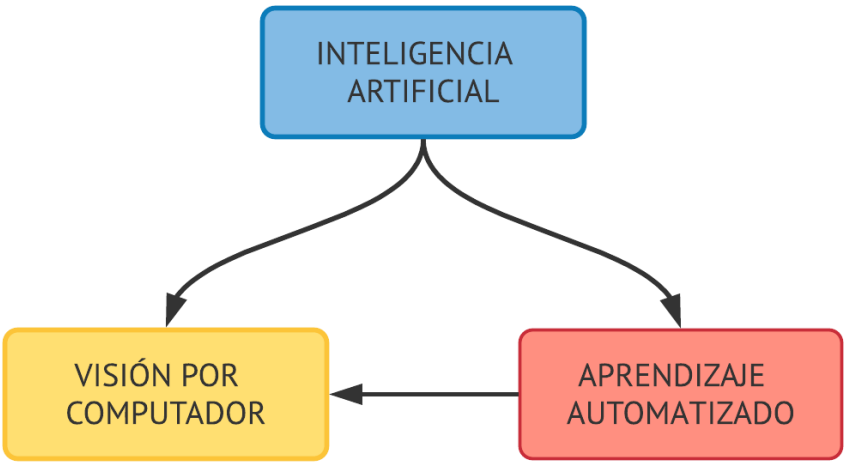

Fig. 4: Relación entre la IA y visión por computador. Fuente: Elaboración propia.

La visión por computador interpreta las imágenes como matrices de píxeles, en la que cada píxel está formado por una composición de tres colores: rojo, verde y azul (en inglés, *RGB*) [10]. En algunas ocasiones se utiliza un valor extra, denominado alfa, que almacena la transparencia del píxel.

2.3.2 Procesado digital de imágenes y visión por computador

El procesado digital de imágenes consiste en la manipulación de imágenes utilizando sistemas de computación, convirtiendo una imagen de entrada en una imagen de salida para mejorar su calidad o conseguir que el análisis posterior de esta sea más sencillo [11].

La visión por computador utiliza algoritmos de procesamiento digital de imágenes para posteriormente analizar la imagen de salida y obtener información cualitativa o cuantitativa de esta.

2.3.3 Segmentación de imágenes

La segmentación de imágenes es una técnica utilizada en imagen digital que consiste en dividir las imágenes de entrada en partes o segmentos, compuestas por un conjunto de píxeles (objetos o regiones de imagen). Su objetivo es simplificar o transformar la imagen de manera que esta sea más fácil de analizar.

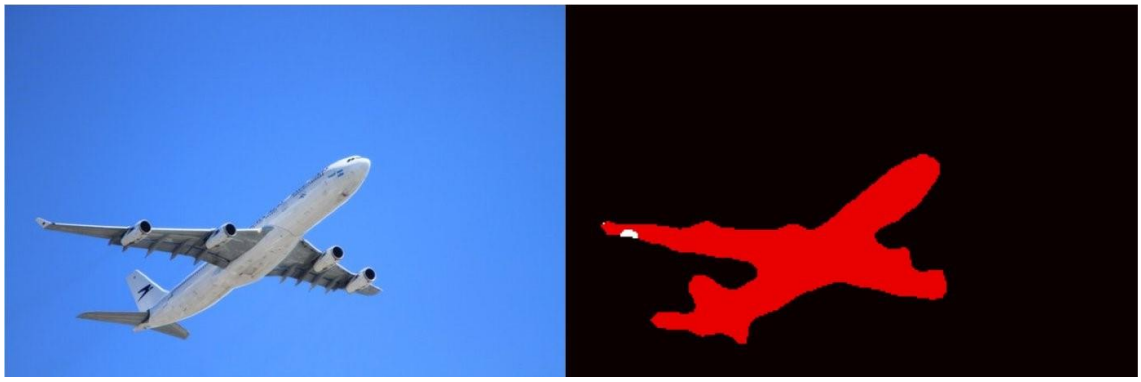

Fig. 5: Ejemplo de segmentación de imagen. Fuente: Amazon Web Service.

La segmentación de imágenes constituye un proceso complejo y diverso, existiendo varias técnicas para poder realizarlo, que pueden a su vez ser combinadas. A continuación, quedan expuestas las más habituales [12].

- **Segmentación basada en regiones:** inicialmente, la imagen se divide de forma iterativa en regiones conectadas que poseen propiedades

diferentes, y en las cuales pueden encontrarse conjuntos de objetos o subregiones. Las técnicas de segmentación basada en regiones más utilizadas son el crecimiento de regiones y la división y fusión de regiones. El crecimiento de regiones parte de un grupo de puntos denominados "semilla" (escogidos de forma manual o automática), a partir de los cuales van creciendo las regiones atendiendo a propiedades similares en los puntos vecinos, como pueden ser el color o la textura.

En la división y fusión de regiones, se divide primero la imagen en regiones arbitrarias, para posteriormente unir las regiones que tengan características similares definidas, como por ejemplo la cantidad de color verde, o dividir las regiones que no compartan esas características.

- **Segmentación basada en detección de bordes:** utilizan la diferencia que puede encontrarse entre regiones adyacentes para determinar la existencia de un borde. Los bordes pueden dividirse además dependiendo de sus ángulos, anchura u otros parámetros. Para detectar las discontinuidades, se utilizan derivadas de primer y segundo orden. Las primeras son especialmente útiles para la detección de bordes en dos dimensiones, obteniendo el vector gradiente, que será máximo en la dirección en que la variación es máxima.

$$G\big(F(x,y)\big) = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{dF}{dx} \\ \dfrac{dF}{dy} \end{bmatrix}$$

En algunas ocasiones, se utiliza el valor absoluto de este para un procesamiento más rápido, ya que el error resultante suele ser despreciable. Las derivadas de segundo orden se utilizan para obtener el laplaciano de funciones bidimensionales.

$$\Delta^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

Tras realizar la detección de bordes, es necesario realizar enlazado y detección de fronteras, para descartar los falsos bordes creados por el ruido, iluminación u otros factores. Algunas técnicas para conseguir esto son el procesamiento local, que analiza la similitud de características entre los puntos vecinos a cada punto, y el procesamiento global, que

utiliza la *Transformada de Hough* para crear campos de acumulación según la los parámetros de la función buscada. Los lugares donde los espacios de acumulación son mayores, son aquellos donde se encuentran los objetos que se están buscando.

- **Segmentación basada en umbralización:** utilizando un umbral que queda definido como una función capaz de convertir imágenes a color en imágenes en blanco y negro. Si tomamos una imagen original con puntos definidos por *f(x,y)*, y un umbral *U*, entonces los puntos que definen la imagen umbralizada son los resultantes de aplicar *g(x,y)*, de manera que

$$g(x,y) = 255 \qquad si\ f(x,y) > U$$
$$g(x,y) = 0 \qquad si\ f(x,y) \leq U$$

  El umbral seleccionado o umbrales seleccionados permitirán que se diferencien los píxeles de los objetos que componen la imagen del fondo de esta. Los umbrales pueden ser globales, si depende solo de la intensidad de gris de los puntos, locales, si además depende de propiedades de los puntos vecinos, o dinámicos, si dependen de los parámetros anteriores y de la posición en coordenadas del punto.

2.3.4 Extracción de la información

Una vez se ha segmentado la imagen y se tenga aislado el objeto a estudiar, se pueden obtener diversas características de este. A continuación, se expone cómo se podrían calcular las más relevantes para este caso concreto.

- **Dimensiones reales del objeto segmentado:** pueden calcularse conociendo diversos parámetros. Por ejemplo, puede aproximarse el diámetro de un objeto que tenga una tendencia de forma circular de la siguiente forma [13]:

$$Diámetro\ real = \frac{Distancia\ * Diámetro\ en\ imagen * Altura\ del\ sensor}{Distancia\ focal * Altura\ de\ la\ imagen\ original}$$

  Donde la distancia, la altura del sensor y distancia focal están dadas en milímetros, mientras que el diámetro del objeto y la altura de la imagen original en la imagen están dadas en píxeles. El diámetro real, por lo tanto, estará dado en milímetros.

La distancia al objeto a la hora de realizar la fotografía puede calcularse a partir de la distancia de enfoque. En este caso, puede obtenerse en dioptrías analizando el resultado de una captura, por lo que solo habría que convertirla a milímetros sabiendo que un metro el inverso de una dioptría. El diámetro en imagen puede medirse una vez se ha segmentado el objeto, y el resto de parámetros (sensor, distancia focal y altura de la imagen original) pueden obtenerse fácilmente analizando las características e información adicional de la imagen tomada y la cámara del dispositivo utilizada.

- **Color o tonalidad media:** se pueden calcular localizando los píxeles que ocupa el objeto y haciendo un sumatorio de las tonalidades que se recogen de cada uno de ellos. Finalmente, solo es necesario dividir por el número de píxeles (media aritmética) para obtener el valor de la tonalidad media. La fórmula a utilizar, por tanto, es la siguiente [14]:

$$Tonalidad\ media = \frac{\sum_{i=1}^{n} Tonalidad\ del\ píxel\ i}{n}$$

Siendo N el número total de píxeles estudiados, es decir, el número de pixeles pertenecientes al objeto segmentado.

- **Momentos y forma:** se pueden calcular los momentos de imagen, que representan promedios ponderados de las intensidades de los píxeles del objeto segmentado. Con estos momentos se pueden calcular características del objeto tales como su centroide, su área y su orientación. La fórmula general para calcular los momentos de imagen es [15]:

$$M_{ij} = \sum_{x}\sum_{y} x^i * y^j * I(x,y)$$

donde I(x, y) es la intensidad del píxel que se encuentra en el punto (x, y) de la matriz representativa de la imagen. En el caso de estudiar una imagen segmentada binaria, la intensidad de cada píxel solo toma valores cero o uno. Una vez se han calculado estos momentos (los más relevantes son los momentos de imagen de orden 0, 1, 2 y 3), pueden calcularse los invariantes *hu* [16] a partir de ellos.

$$
\begin{aligned}
hu[0] &= \eta_{20} + \eta_{02} \\
hu[1] &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
hu[2] &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
hu[3] &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
hu[4] &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
hu[5] &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
hu[6] &= (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}
$$

Fig. 6: Momentos hu. Fuente: OpenCV Doc.

Estos valores son independientes de la escala de la imagen, la rotación y los reflejos, por lo que se pueden utilizar, por ejemplo, para medir la correlación entre las formas de dos objetos segmentados en diferentes imágenes.

2.4 Seguimiento de lunares utilizando la visión por computador

2.4.1 ¿Por qué utilizar la visión por computador?

La evolución de los lunares se produce comúnmente de manera paulatina a lo largo del tiempo. Si unimos este factor a la gran cantidad de lunares que tenemos por todo el cuerpo, el seguimiento de estos sin ayuda y utilizando simplemente la vista humana puede ser un proceso muy complicado, que puede además llevar a errores, ya que la visión humana tiene unas limitaciones relativamente grandes. La visión por computador utiliza algoritmos que consiguen una alta precisión en la detección de cambios entre pares de imágenes, de una forma rápida y sencilla para el usuario si se aplica de forma adecuada en el software desarrollado.

2.4.2 Cómo utilizar la visión por computador en este ámbito

Para realizar el seguimiento de un lunar en una determinada parte del cuerpo, es necesario obtener fotografías en momentos temporales distintos, que preferentemente diferirán en algunos días de diferencia. Con algoritmos de segmentación, puede obtenerse la parte de cada imagen en la que se encuentra el lunar, para posteriormente analizar las diferencias entre dichas imágenes, en concreto atendiendo a los parámetros de:

- **Forma:** utilizando los momentos *hu* calculados para cada imagen consiguiendo determinar una correlación entre pares de imágenes. Teniendo en cuenta que las fotografías deberán ser tomadas desde cerca

de manera que el usuario coloque el lunar en un área marcada en la pantalla del dispositivo, no debería haber problema con los cambios de orientación en las imágenes, aunque pueden crear pequeños errores que no son preocupantes.

- **Color:** comparando las tonalidades del lunar en cada imagen para determinar si existen cambios de este. Como los canales del modelo de color RGB quedan muy afectado por la luminosidad a la hora de tomar la fotografía, puede utilizarse el canal H (Tonalidad) del modelo de color HSV para calcular la tonalidad media, ya que este canal es afectado mínimamente por los cambios de luminosidad.
- **Tamaño:** aproximando la forma del lunar a una circunferencia y determinando si el diámetro del lunar en cada imagen ha variado a lo largo del tiempo, especialmente en el caso de aumento de tamaño.

3

Especificación y análisis

3.1 Metodología de desarrollo

Para conseguir una correcta estructuración, planificación y control del desarrollo de la aplicación es necesario definir una metodología a seguir desde el inicio del proyecto. En la actualidad existe un amplio abanico de enfoques que pueden utilizarse a la hora de desarrollar software. A continuación, se exponen los más comunes, analizando sus pros y sus contras.

- **Modelo en cascada:** es el modelo más utilizado, y consiste en un modelo lineal que se basa en la sucesión de etapas. Es un modelo simple y que necesita de pocos recursos, pero no permite volver atrás si algo ha salido mal en las fases de especificación o diseño.

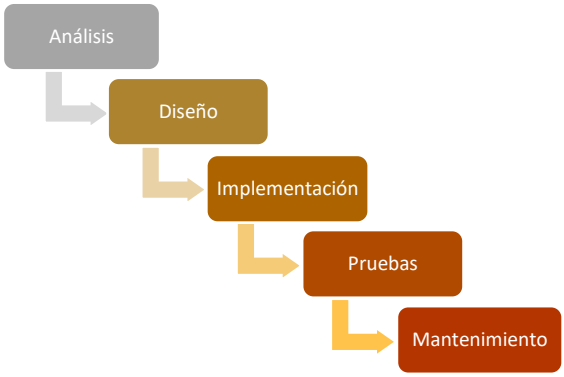

Fig. 7: Modelo en cascada. Fuente: Elaboración propia.

- **Modelo incremental:** es una evolución del modelo en cascada, en la que este queda realimentado, reduciendo sus desventajas. Se implementan las funcionalidades por fases y de forma incompleta hasta llegar al producto final.
- **Modelo basado en prototipos:** es un modelo de desarrollo evolutivo en el que se van construyendo prototipos con pocos recursos. Permite realizar cambios en los requisitos a lo largo de la implementación.
- **Modelo en espiral:** es un modelo que se utiliza en proyectos que conllevan un riesgo alto, y se basa en un crecimiento incremental de forma cíclica. Necesita una gran cantidad de recursos para ser implementado.

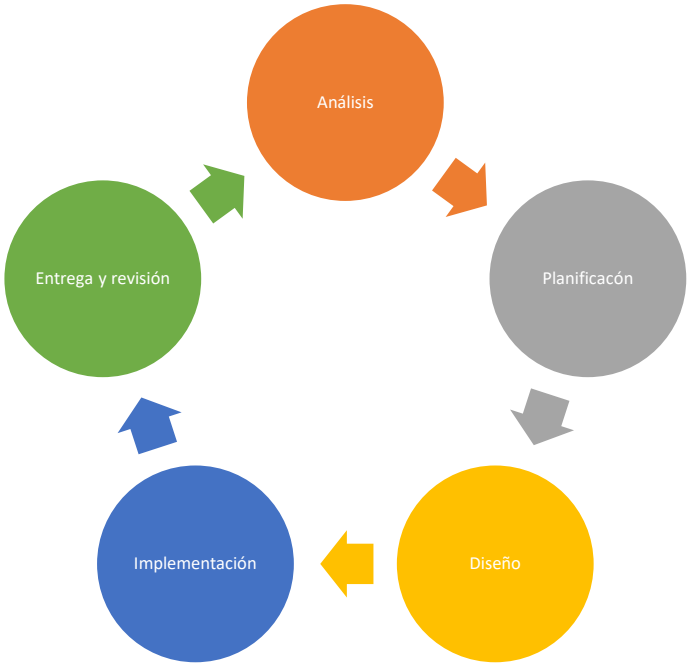

Fig. 8: Modelo basado en prototipos. Fuente: Elaboración propia.

- **Desarrollo rápido de aplicaciones (RAD):** es una técnica ágil que da prioridad a las entregas de prototipos e iteraciones rápidas. Permite una gran adaptabilidad y gestión de riesgos, pero es difícil medir su progreso y puede dar lugar a muchos errores.
- **Metodología Scrum:** es una metodología ágil basada en iteraciones. Cada iteración cuenta con objetivos y tareas concretas, lo que consigue resultados en cada una de ellas, con una buena gestión de riesgos y una gran adaptabilidad y flexibilidad. Es una buena opción para equipos de trabajo pequeños, aunque requiere de una buena planificación.

Fig. 9: Metodología Scrum. Fuente: Sinnaps.

- **Programación extrema (XP):** es otra técnica ágil que se orienta plenamente a las necesidades del cliente. Consigue una tasa de errores pequeña y se adapta a los cambios fácilmente, siendo además una técnica muy recomendada para proyectos de corta duración, pero, en caso de fallos, el coste es alto.

Se ha considerado que la mejor metodología para realizar el desarrollo de este proyecto es la metodología Scrum, ya permite realizar pequeñas versiones bien planificadas en iteraciones que se evalúen constantemente, contando con la posibilidad de cambiar requisitos y diseño a lo largo del proceso de implementación.

3.2 Requisitos funcionales

A continuación, quedan detallados los requisitos funcionales que se definieron para desarrollar la aplicación:

RF1. El sistema debe mostrar al usuario la figura de un ser humano al acceder a la aplicación.

RF2. El sistema debe permitir al usuario acceder a las distintas partes del cuerpo a través de la figura mostrada.

RF3. El sistema debe permitir al usuario visualizar una lista de los lunares existentes en la parte del cuerpo seleccionada.

RF4. El sistema debe permitir al usuario eliminar la carpeta de un lunar con todas las fotografías contenidas.

RF5. El sistema debe permitir al usuario crear la carpeta de un nuevo lunar con su primera fotografía.

RF6. El sistema debe permitir al usuario acceder a la carpeta de un lunar para visualizar su historial de fotografías.

RF7. El sistema debe permitir al usuario eliminar una fotografía del historial de un lunar.

RF8. El sistema debe permitir al usuario actualizar el historial de un lunar tomando nuevas fotografías.

RF9. El sistema debe permitir al usuario realizar un estudio de los cambios de un lunar a lo largo del tiempo.

RF10. El sistema debe permitir al usuario visualizar los resultados de los estudios realizados sobre los cambios de los lunares.

3.3 Requisitos no funcionales

A continuación, quedan detallados los requisitos no funcionales que se definieron para desarrollar la aplicación:

RNF1. La aplicación debe tener una interfaz intuitiva y sencilla, de manera que el usuario pueda navegar a través de ella sin problema

RNF2. Cualquier usuario estándar puede ser capaz de aprender a manejar la aplicación por sí mismo en un corto periodo de tiempo, de como máximo una hora.

RNF3. El sistema debe tener un tiempo de respuesta razonable, a pesar de que la velocidad no sea un parámetro crítico en esta aplicación.

RNF4. Los números mostrados al usuario tendrán como máximo dos dígitos decimales.

RNF5. La aplicación debe estar disponible para su uso las 24 horas del día y todos los días del año.

RNF6. La aplicación debe ser capaz de soportar el manejo de una gran cantidad de archivos de imagen, sin llegar a perder mucho rendimiento cuando aumente.

RNF7. La aplicación puede ser utilizada en cualquier lugar con un teléfono inteligente que utilice un sistema operativo Android de la versión 5.0 (*Lollipop*) o superior.

RNF8. La aplicación debe tener disponibilidad y coherencia de los datos en todos los ficheros manejados.

RNF9. La aplicación estará bajo la jurisdicción de la Ley Orgánica de Protección de Datos de Carácter Personal del Estado Español.

RNF10. La aplicación debe ser capaz de explicarse por sí misma, a través de ventanas y diálogos, sin necesidad de que el usuario necesite un manual.

RNF11. El logo de la aplicación debe aparecer en una pantalla de carga inicial.

3.4 Casos de uso

La funcionalidad del sistema completo quedará explicada a través de los casos de uso.

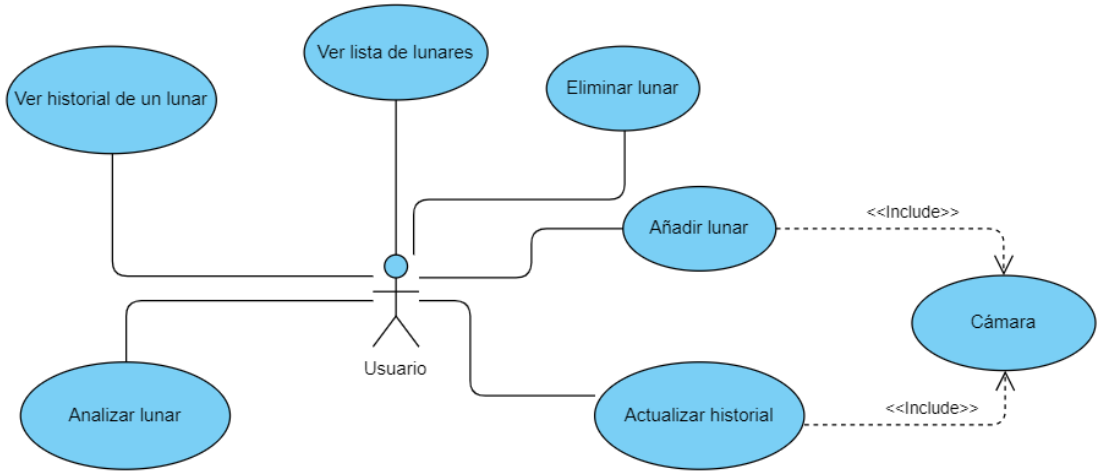

Fig. 10: Diagrama de casos de uso de la aplicación. Fuente: Elaboración propia

Para facilitar la comprensión de los casos de uso, se detallan en las siguientes tablas las funcionalidades de cada uno de ellos:

Título	Ver lista de lunares
Descripción	Permite al usuario visualizar la lista de lunares de la parte del cuerpo seleccionada
Actor principal	Usuario
Precondición	El usuario se encuentra en la vista inicial
Postcondición	Se muestra la lista de lunares de la parte del cuerpo seleccionada
Escenario principal	
1. El usuario pulsa sobre la parte del cuerpo deseada en la figura de ser humano	
Escenario alternativo	
-	

Tabla 2: Caso de uso "Ver lista de lunares". Fuente: Elaboración propia.

Título	Añadir lunar
Descripción	Permite al usuario añadir un nuevo lunar en una parte del cuerpo
Actor principal	Usuario
Precondición	El usuario ha seleccionado una parte del cuerpo
Postcondición	Se ha añadido un nuevo lunar a la parte del cuerpo seleccionada
Escenario principal	
2. El usuario pulsar sobre el botón de añadir un nuevo lunar 3. El sistema muestra una ventana para introducir el nombre del lunar 4. El usuario introduce el nombre del lunar 5. El sistema muestra la funcionalidad de la cámara 6. El usuario fotografía el nuevo lunar 7. El sistema notifica al usuario de que el lunar se ha creado correctamente	
Escenario alternativo	
3.a. El usuario introduce un nombre vacío o de un lunar ya existente 3.b. El sistema notifica el error al usuario y vuelve atrás 4.a. La cámara no se puede abrir porque se está utilizando en otra APP 4.b. El sistema notifica el error al usuario y vuelve atrás	

Tabla 3: Caso de uso "Añadir lunar". Fuente: Elaboración propia.

Título	Eliminar lunar
Descripción	Permite al usuario eliminar un lunar existente en una parte del cuerpo junto a todo su historial
Actor principal	Usuario
Precondición	El usuario ha seleccionado una parte del cuerpo
Postcondición	Se ha eliminado el lunar seleccionado junto a su historial
Escenario principal	
1. El usuario selecciona un lunar 2. El usuario pulsa sobre el botón de eliminar lunar 3. El sistema muestra un diálogo de confirmación 4. El usuario acepta el diálogo 5. El sistema notifica al usuario de que el lunar se ha eliminado	
Escenario alternativo	
-	

Tabla 4: Caso de uso "Eliminar lunar". Fuente: Elaboración propia.

Título	Ver historial de un lunar
Descripción	Permite al usuario visualizar el historial de fotografías existentes de un lunar
Actor principal	Usuario
Precondición	El usuario se encuentra en la vista del listado de lunares de una parte del cuerpo
Postcondición	Se muestra el historial de fotografías existentes del lunar seleccionado
Escenario principal	
1. El usuario selecciona el lunar sobre el que quiere ver el historial de fotografías	
Escenario alternativo	
-	

Tabla 5: Caso de uso "Ver historial de un lunar". Fuente: Elaboración propia.

Título	Actualizar historial
Descripción	Permite al usuario añadir una nueva fotografía al historial de un lunar
Actor principal	Usuario
Precondición	El usuario ha seleccionado un lunar
Postcondición	Se ha añadido una nueva fotografía al historial del lunar seleccionado
Escenario principal	
1. El usuario pulsar sobre el botón de añadir una nueva fotografía 2. El sistema muestra la funcionalidad de la cámara 3. El usuario fotografía el lunar 4. El sistema notifica al usuario de que el historial se ha actualizado correctamente	
Escenario alternativo	
4.a. La cámara no se puede abrir porque se está utilizando en otra APP 4.b. El sistema notifica el error al usuario y vuelve atrás	

Tabla 6: Caso de uso "Actualizar historial". Fuente: Elaboración propia.

Título	Analizar lunar
Descripción	Permite al usuario analizar los cambios de un lunar a lo largo del tiempo
Actor principal	Usuario
Precondición	El usuario ha seleccionado un lunar
Postcondición	Se muestran los resultados del análisis sobre los cambios del lunar
Escenario principal	
1. El usuario pulsa sobre el botón de analizar lunar 2. El sistema realiza los cálculos necesarios 3. El sistema muestra al usuario los resultados en formato de texto y gráficas	
Escenario alternativo	
-	

Tabla 7: Caso de uso "Analizar lunar". Fuente: Elaboración propia.

Título	Cámara
Descripción	Permite al usuario utilizar la cámara para tomar una nueva fotografía
Actor principal	Usuario
Precondición	El usuario ha entrado a la funcionalidad de la cámara
Postcondición	Se ha tomado una nueva fotografía
Escenario principal	
1. El sistema muestra un diálogo explicativo al usuario 2. El usuario acepta el diálogo 3. El sistema muestra una previsualización de la cámara junto a un área indicativa donde colocar el lunar 4. El usuario pulsa el botón de capturar	
Escenario alternativo	
-	

Tabla 8: Caso de uso "Cámara". Fuente: Elaboración propia.

4

Tecnologías y herramientas

4.1 Sistema operativo objetivo

La aplicación ha sido desarrollada para dispositivos Android por las múltiples ventajas que se ha considerado que tiene este sistema operativo para móviles con respecto a otros en este caso concreto. La ventaja principal considerada es que se pretende que la aplicación sea accesible al mayor número de personas posible, y a nivel mundial Android es el sistema operativo más elegido por los usuarios a la hora de escoger un *Smartphone*.

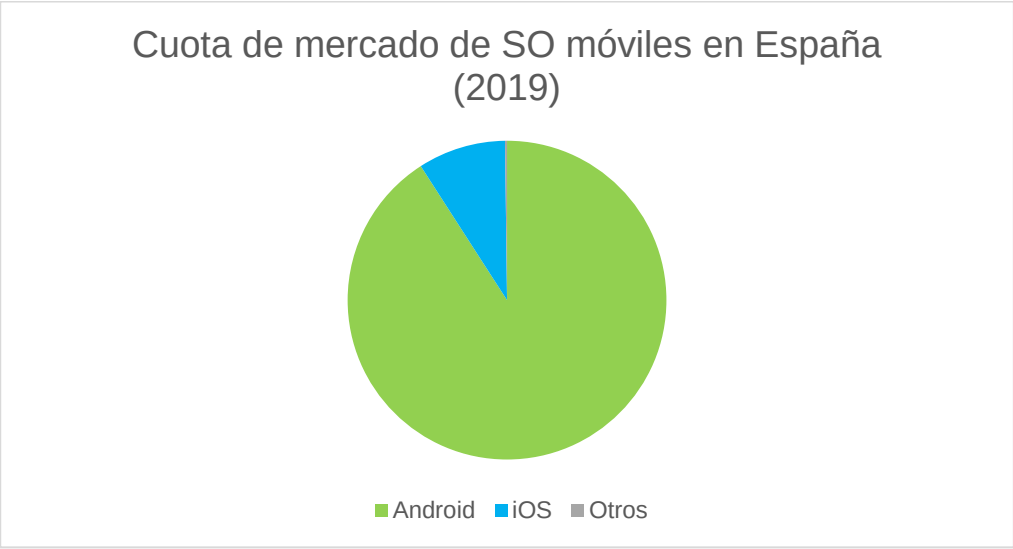

Fig. 11: Cuota de mercado de SO móviles en España (2019). Fuente: Elaboración propia.

Otras de las ventajas consideradas que ofrece este sistema operativo con respecto a otros son las siguientes:

- El desarrollo de aplicaciones para Android se basa en el lenguaje de programación Java, uno de los lenguajes más utilizados en el mundo del desarrollo software, lo que hace que haya una extensa documentación disponible sobre él.
- El IDE oficial que se utiliza para el desarrollo pertenece y es mantenido por Google, la misma empresa a la que pertenece Android. Gracias a esto existe una documentación oficial detallada, y se puede trabajar en un entorno pensado única y exclusivamente para el desarrollo de este tipo de aplicaciones.
- La libertad de desarrollo es mayor en este sistema operativo, ya que Android es un sistema abierto que permite que cualquiera pueda acceder a su código fuente y que no impone demasiadas restricciones a los desarrolladores de aplicaciones para los dispositivos que lo usan.

4.2 Entorno de desarrollo integrado (IDE)

Para el desarrollo de la aplicación se ha utilizado Android Studio [16], que reemplazó a Eclipse en 2014 como el IDE oficial para la plataforma Android. Es un IDE gratuito con licencia Apache, una licencia de software libre que puede utilizarse con cualquier propósito. Este IDE permite al usuario crear aplicaciones para Android combinando un editor de código fuente con un editor gráfico para la interfaz de usuario de dichas aplicaciones, permitiendo además probar estas tanto en el emulador de dispositivos móviles que tiene integrado como en el propio dispositivo móvil del desarrollador, que deberá estar conectado mediante un cable USB al ordenador donde se ejecute el IDE.

4.3 Lenguajes de programación

Los dos lenguajes que Android Studio permite utilizar para desarrollar aplicaciones Android son Java y Kotlin. el desarrollo de aplicaciones en dos lenguajes: Java y Kotlin [18]. Ambos son lenguajes de código abierto, y aunque Kotlin presenta una curva de aprendizaje más sencilla que la de Java, y la seguridad de Kotlin es mayor en ciertos aspectos a la hora de programar (por ejemplo, no acepta valores nulos), la familiaridad con Java y la gran comunidad

y documentación que tiene detrás, así como la cantidad inmensa de bibliotecas existentes compatibles con el lenguaje, han hecho que haya sido la elección para desarrollar la aplicación de este Trabajo de Fin de Grado.

Para el desarrollo de aplicaciones Android con el IDE elegido, el lenguaje XML [19] es el utilizado para el desarrollo de las interfaces de usuario. Además, el manifiesto de la aplicación, en el que se declaran permisos, características necesarias y clases, utiliza este lenguaje. También se usa XML para el desarrollo de los archivos que definen valores de la aplicación, como los colores, los estilos de las vistas y las cadenas de caracteres.

4.4 Dispositivos de soporte y depurado

Durante el desarrollo de la aplicación, ha sido necesario ir comprobando el funcionamiento de las nuevas funcionalidades, así como ir depurando el código implementado. Para ello, se han utilizado varios teléfonos inteligentes:

- Un teléfono inteligente físico, en concreto, un Redmi Note 8 [20] de la marca Xiaomi, que será utilizado como dispositivo de soporte y depurado principal, combinado con el depurador propio del IDE utilizado.
- Varios teléfonos inteligentes emulados con el emulador que puede descargarse integrado con Android Studio, conocido como AVD Emulator.

4.5 Librerías y paquetes

Se han utilizado los paquetes nativos del lenguaje Java, así como los paquetes nativos que contiene Android Studio para el desarrollo de aplicaciones Android en este lenguaje.

Además, se han utilizado dos librerías externas:

- OpenCV [21], cuyo propósito principal es el de la visión por computador en tiempo real, concretamente para todo el procesado de imágenes, desde la segmentación hasta las comparaciones oportunas para poder detectar los cambios relevantes en los lunares fotografiados.
- GraphView [22], que permite crear a través de la programación gráficas de barras, líneas y puntos en la interfaz de usuario.

4.6 Automatización del compilado

Para la automatización del compilado se ha utilizado el propio sistema de código abierto que ya trae integrado Android Studio, Gradle [22]. Utiliza un lenguaje específico basado en el lenguaje de programación Apache Groovy.

Gracias a Gradle, el compilado de la aplicación puede realizarse de forma automática, y la gestión de dependencias del proyecto se vuelve mucho más sencilla.

4.7 Control de versiones

Para realizar el control de las versiones del desarrollo de la aplicación, se ha utilizado GiHub [23]. La utilización de esta herramienta permite guardar en la nube los distintos cambios que se hagan sobre los archivos de la aplicación, facilitando además la revisión de estos por parte del tutor, ya que GitHub permite que todos los integrantes de un proyecto puedan visualizar los cambios subidos en cada momento, además de contener un sistema de registro de incidencias.

4.8 Archivos de configuración

La aplicación necesitaba recoger ciertos parámetros de configuración (como las rutas donde se guardarán los archivos) e información adicional sobre las fotografías. Se han utilizado archivos JSON [24] para guardarlo todo a nivel local en el dispositivo del usuario. Así mismo, se ha utilizado la librería Gson (Google JSON), que permite trabajar con JSON en Java facilitando mucho el proceso.

5

Diseño

5.1 Estructura

Todos los proyectos de aplicaciones para Android realizados con Android Studio tienen una jerarquía de archivos muy similar [25], si bien permite cierta flexibilidad para reorganizarla. El proyecto se divide en varios módulos, algunos de los cuales contienen varios directorios y archivos dentro.

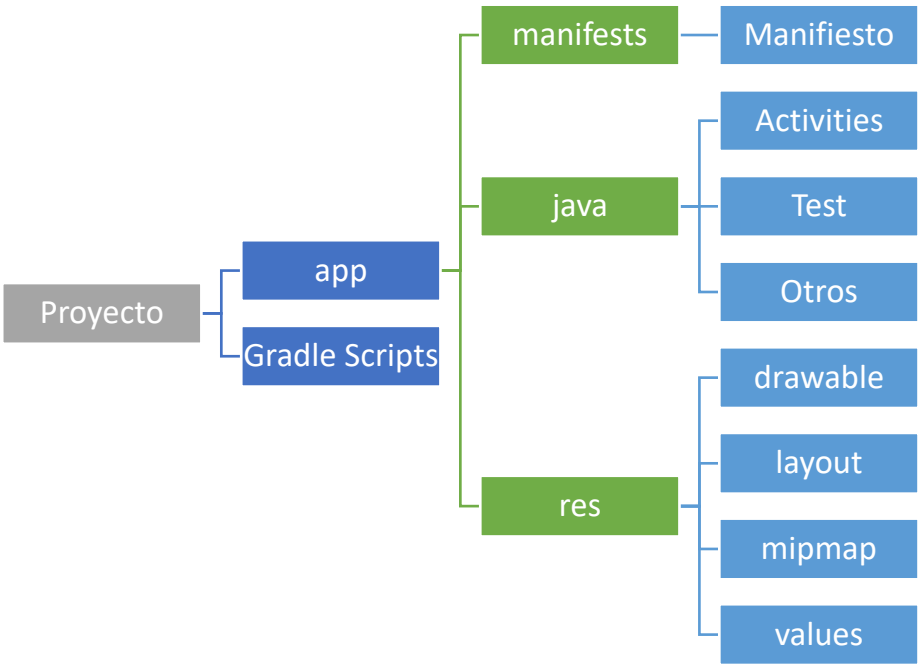

Fig. 12: Estructura principal del proyecto. Fuente: Elaboración propia.

En el proyecto existen dos grupos superiores principales, *Gradle Scripts* y *app*. El primero contiene todos los archivos de configuración necesarios para compilar

el proyecto, y el segundo contiene la aplicación en sí misma. Sus directorios principales se exponen a continuación.

5.1.1 Manifests

Este directorio contiene el archivo AndroidManifest.xml, que se utiliza para declarar:

- El nombre del paquete de la aplicación.
- Los componentes que utiliza la aplicación, desde las actividades hasta los proveedores de contenido.
- Los permisos necesarios para que la aplicación acceda a distintas partes protegidas del dispositivo, además de los permisos que necesitan otras aplicaciones para acceder a esta.
- Los requisitos de hardware y software que necesita la aplicación para ser instalada desde Google Play.

Es un archivo fundamental para el proyecto, ya que sin él no puede realizarse el compilado de la aplicación.

5.1.2 Java

Contiene todos los archivos Java con código fuente, separados por paquetes, entre los que se encuentran los paquetes de prueba. En el paquete principal, se incluyen las clases que necesita la aplicación para funcionar, como las actividades o las clases de soporte.

5.1.3 res

Contiene los recursos necesarios para la aplicación, como los diseños de interfaz, los valores de las cadenas de caracteres utilizadas en estas o las imágenes utilizadas.

5.2 Diagrama de clases

A continuación, se muestra el diagrama de clases de la aplicación. Por cuestiones de espacio, en la página, no contiene los campos, constructores y métodos de cada clase de forma detallada. Se mostrará la información detallada de las clases más relevantes en el apartado 6, al tratar la implementación.

El diagrama de clases ha sido generado a partir de la herramienta *simpleUML*, una extensión disponible para Android Studio que permite generar este tipo de diagramas a partir del código del proyecto.

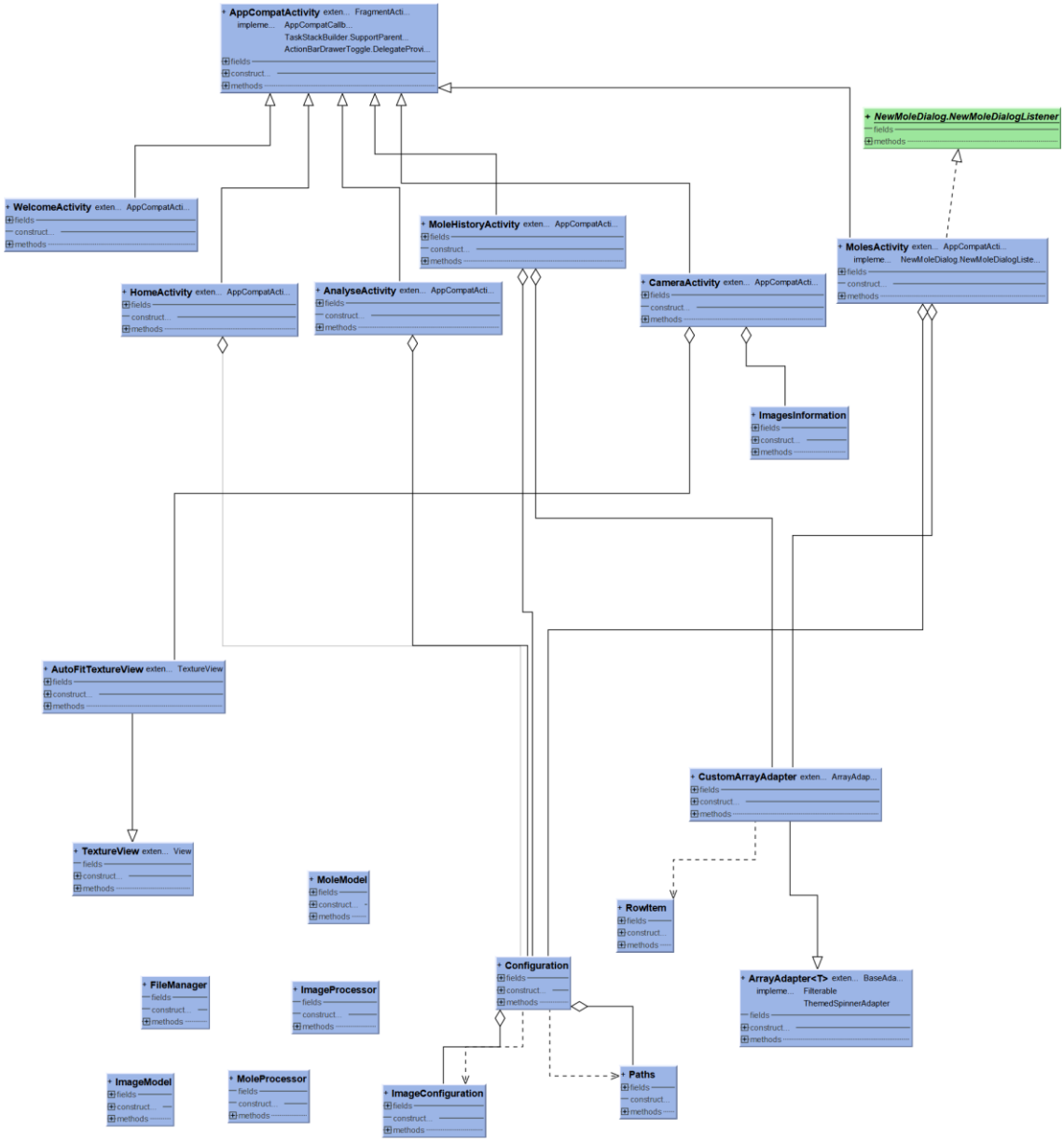

Fig. 13: Diagrama de clases de la aplicación. Fuente: Elaboración propia.

5.3 Diagramas de actividad

Para diseñar la aplicación, ha sido necesario definir los diagramas de actividad de cada una de las funcionalidades que se ofrecen al usuario. Estos diagramas están estrechamente relacionados con los casos de uso (Apartado 3.4).

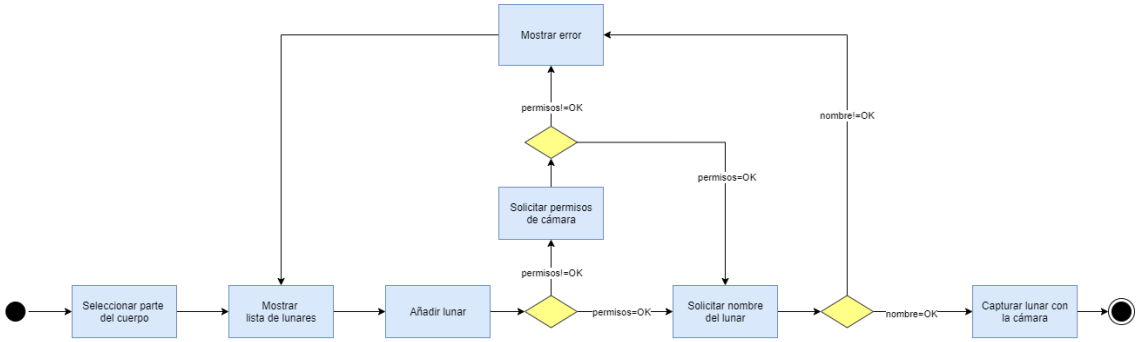

Fig. 14: Diagrama de actividad para añadir un lunar. Fuente: Elaboración propia.

Como se puede observar en la figura 14, el diagrama de actividad para añadir un lunar contiene a su vez al diagrama de actividad para mostrar la lista de lunares. Así mismo, puede verse que contiene tres nodos de decisión: dos de ellos son utilizados para comprobar y solicitar (si es necesario) los permisos para acceder a la cámara del dispositivo, y el otro se encarga de comprobar si se ha introducido un nombre válido para el nuevo lunar.

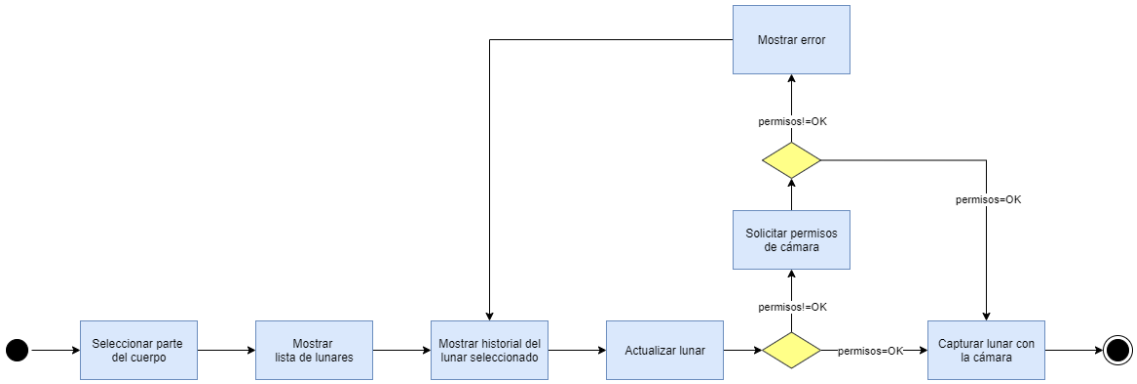

Fig. 15: Diagrama de actividad para actualizar un lunar. Fuente: Elaboración propia.

En la figura 15 puede verse que el diagrama de actividad para actualizar un lunar es bastante parecido al de añadir un lunar, solo que en esta ocasión no existe el nodo de decisión para comprobar el nombre, ya que no es necesario introducirlo (simplemente se está actualizando el historial de un lunar ya existente). Este diagrama, además de contener el diagrama de actividad para mostrar la lista de lunares, también contiene el diagrama de actividad para mostrar el historial de un lunar.

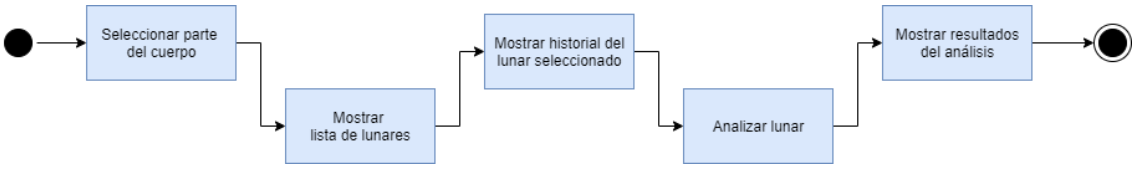

Fig. 16: Diagrama de actividad para analizar un lunar. Fuente: Elaboración propia.

En la figura 16 se muestra el diagrama de actividad para analizar un lunar. El diagrama es igual que el diagrama de actividad para actualizar un lunar hasta el nodo “Mostrar historial del lunar seleccionado”, que representa una vista / actividad en la cual el usuario debe decidir si actualiza el historial, lo analiza, o borra alguna de las fotografías pertenecientes a este.

En las siguientes figuras (17 y 18), se muestran los diagramas de actividad para eliminar un lunar con todo su historial o una fotografía concreta del historial de este respectivamente. No se profundizará en ellos por su trivialidad.

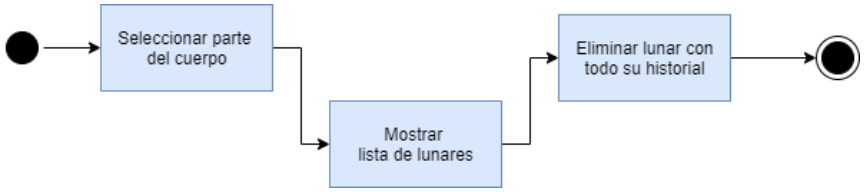

Fig. 17: Diagrama de actividad para eliminar un lunar. Fuente: Elaboración propia.

Los diagramas de actividad para mostrar la lista de lunares o para mostrar el historial de un lunar concreto no se muestran de forma explícita porque, como se ha mencionado, son subdiagramas que ya están incluidos en las figuras anteriores.

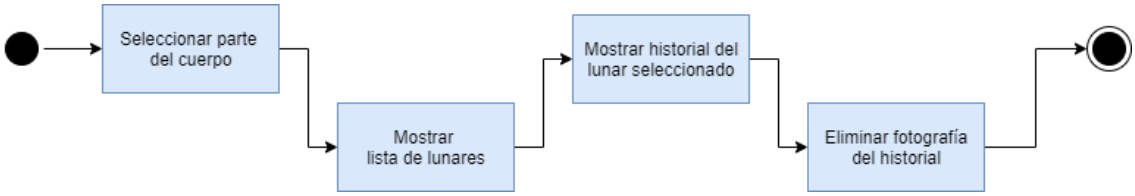

Fig. 18: Diagrama de actividad para eliminar una fotografía del historial. Fuente: Elaboración propia.

5.4 Fases de la implementación

La implementación se ha realizado en varias fases en las que se han desarrollado diversos prototipos, de manera que cada uno de ellos contuviera las funcionalidades de los prototipos anteriores y las mejorarán o añadieran nuevas. En total, se han desarrollado cuatro prototipos:

1. Un prototipo capaz de llamar a la aplicación externa de la cámara para realizar fotografías que se guardan en la memoria externa del dispositivo. Las fotografías se almacenan divididas en diferentes carpetas generando un árbol de directorios y archivos.
2. Un prototipo capaz de crear nuevos lunares y de actualizar los existentes, que utiliza una cámara personalizada e integrada en la propia aplicación sin necesidad de llamar a aplicaciones externas para realizar las fotografías.
3. Un prototipo capaz de analizar las imágenes de la carpeta de un lunar, extrayendo de cada una de las fotografías realizadas características del lunar como la forma, el color o el tamaño.
4. Un prototipo final en el que se ha realizado una refactorización final y una revisión de la integración completa de todas las funcionalidades que debía implementar la aplicación.

Para cada uno de los prototipos, se han realizado pruebas específicas, determinando si las funcionalidades implementadas son correctas. Al terminar el prototipo final, se han realizado pruebas finales en las que se ha comprobado que se satisfacen todos los requisitos definidos para la aplicación, realizando algunos cambios menores en la implementación.

6

Implementación

6.1 Actividades

Las actividades en Android representan acciones únicas y centradas que puede realizar el usuario [26]. La mayoría de estas interactúan con el usuario a través de la pantalla, por lo que las clases Java que las implementan suelen estar enlazadas con archivos XML que se corresponden con la vista de cada una de ellas.

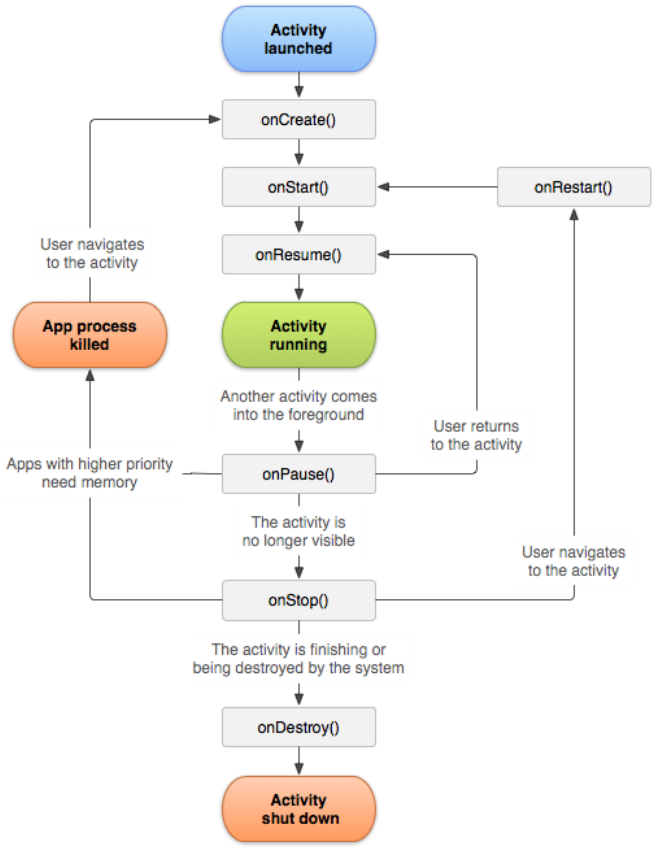

Fig. 19: Diagrama de estados de una actividad en Android. Fuente: Android Developers.

6.1.1 Welcome Activity

Es la actividad que se inicia al comienzo de la ejecución de la aplicación. En ella se comprueba si existen los archivos de configuración necesarios en el dispositivo del usuario. Esos archivos pueden no existir en caso de que la aplicación se esté ejecutando por primera vez desde que se instaló en el dispositivo, o en caso de que el usuario los haya borrado por error. En ese caso, se crearán los archivos con valores por defecto.

Tras realizar la comprobación de la existencia de dichos archivos y crearlos si era necesario, se leerán para determinar las rutas donde deben buscarse los directorios que contendrán el resto de archivos de la aplicación en el dispositivo, como, por ejemplo, las fotografías tomadas. Si esos archivos no existen aún, se crearán.

El archivo de configuración será utilizado posteriormente por la mayoría de actividades, ya sea para determinar las rutas donde deben guardar los archivos o para utilizar ciertos parámetros de configuración para las imágenes, como por ejemplo el área de recorte que se utilizará.

Esta actividad también mostrará el logo de la aplicación durante dos segundos al usuario, a modo de introducción. Una vez pasen esos dos segundos, la actividad da paso a Home Activity.

6.1.2 Home Activity

En esta actividad se muestra al usuario un avatar con forma de ser humano dividido en cinco partes: cabeza, torso, brazo izquierdo, brazo derecho, pierna izquierda y pierna derecha, que representan las distintas carpetas de lunares principales existentes en el proyecto. Cuando se pulsa sobre una de ellas, se crea una nueva actividad *Moles Activity*.

6.1.3 Moles Activity

Una vez se ha seleccionado la parte del cuerpo deseada, esta actividad es la encargada de mostrar al usuario la lista de lunares creados anteriormente en esa parte del cuerpo, con los nombres que el usuario eligió para cada uno de ellos. El usuario tiene ahora tres alternativas:

- Crear un nuevo lunar: para ello pulsará sobre el botón de “añadir lunar”, representado como un botón circular con un “+” dentro de él. Cuando esto ocurra, se mostrará al usuario un diálogo que pedirá el nombre para el nuevo lunar. Este nombre deberá ser diferente a los nombres de los lunares ya creados en esa parte del cuerpo, y no vacío. Una vez se introduzca el nombre, se dará paso a la actividad *Camera Activity*.
- Borrar un lunar existente: si el usuario deja pulsado sobre uno de los lunares que aparecen en la lista, se mostrará un diálogo preguntando si se quiere borrar ese lunar con todo su contenido (su historial de fotografías). Si se acepta el diálogo, el lunar será eliminado junto a su historial.
- Seleccionar un lunar existente: si el usuario realiza una pulsación corta sobre uno de los lunares de la lista, se dará paso a la actividad *Mole History Activity*.

6.1.4 Mole History Activity

Cuando se selecciona un lunar en una parte del cuerpo determinada, esta actividad muestra el historial de fotografías de ese lunar, incluyendo en el nombre de cada imagen la fecha y hora en la que se realizó. En esta actividad, al usuario se le presentan también tres alternativas:

- Actualizar un lunar: para ello pulsará sobre el botón de “actualizar lunar”, representado como un botón circular con un “+” dentro de él. Al hacerlo, se dará paso a la actividad *Camera Activity*.
- Borrar una fotografía del historial: si el usuario deja pulsado sobre una de las fotografías que aparecen en la lista, se mostrará un diálogo preguntando si se quiere borrar dicha fotografía. Si se acepta el diálogo, la fotografía será eliminada, así como sus parámetros asociados en el *JSON* que se va generando al realizar fotografías.
- Analizar lunar: el usuario deberá pulsar sobre el botón de “analizar lunar”, representado con un botón circular con el dibujo de una lupa en su interior. Al hacerlo, se dará paso a la actividad *Analyze Activity*.

6.1.5 Camera Activity

Cuando un usuario pulsa sobre el botón de “añadir lunar” desde la actividad *Moles Activity* o sobre el botón de “actualizar lunar” desde la actividad *Mole History Activity*, se comprueba que la aplicación tenga los permisos necesarios para utilizar la cámara. En caso de que no sea así, se pedirá al usuario que conceda dichos permisos. Si los rechaza, se notificará de que estos permisos son necesarios para poder utilizar la aplicación de forma completa pudiendo tomar nuevas fotografías. Si los acepta, se dará paso a esta actividad. La actividad utiliza la API *Camera2* [27], que fue introducida en Android a partir de la API 21 (Versión 5.0, Lollipop), del Sistema Operativo.

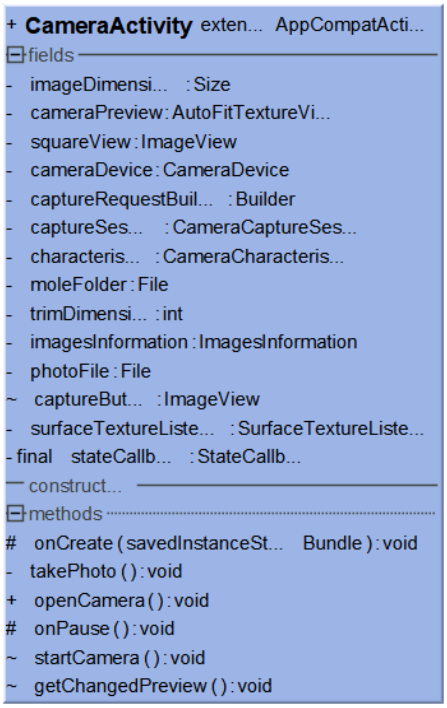

Fig. 20: Clase *CameraActivity.java*. Fuente: Elaboración propia.

La actividad en cuestión muestra al iniciarse un diálogo explicando al usuario que debe posicionar el lunar que quiere fotografiar dentro de un cuadrado verde semitransparente que aparece en la zona central de las imágenes capturas por la cámara. Cuando el usuario acepte el diálogo, deberá realizar lo indicado, para posteriormente pulsar sobre el botón de “capturar imagen”. Una vez el usuario pulse el botón, se toma la fotografía. Para ello la actividad debe crear un archivo temporal sobre el que se escribirá la imagen capturada en ese momento. El archivo tendrá un formato JPG. La fotografía será tomada con la mayor resolución que permita la cámara del dispositivo, para conseguir la mejor calidad

y detalles. Antes de guardar la imagen en el almacenamiento del *smartphone*, la fotografía será procesada:

- Se recortará un cuadrado central en la imagen con un tamaño predefinido en el archivo de configuración de la aplicación. Este recuadro debe contener al lunar fotografiado, ya que previamente se le habrá marcado al usuario en la vista previa esta zona para que encajara el lunar en ella al tomar la fotografía.
- Se rotará la imagen para que quede con una vista vertical. Esto nos influye en el procesado posterior pero sí que es estéticamente más agradable para el usuario a la hora de ver las miniaturas de las fotografías tomadas.

6.1.6 Analize Activity

Cuando un usuario pulsa sobre el botón de “analizar lunar” desde la actividad *Mole History Activity*, la aplicación comienza el análisis del historial del lunar seleccionado. Para ello, buscará en la carpeta del lunar cada uno de los archivos con la extensión *.jpg*, que contendrán las fotografías realizadas en distintos momentos del tiempo de ese lunar. De forma paralela, se leerán los parámetros guardados sobre cada fotografía en el momento de su captura desde el archivo JSON correspondiente (nombre, distancia de enfoque, longitud focal, altura del sensor y altura de la imagen original).

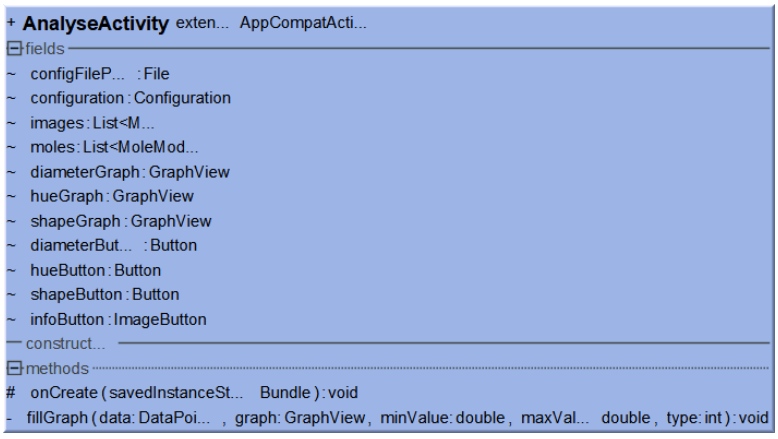

Fig. 21: Clase *AnalyseActivity.java*. Fuente: Elaboración propia.

Una vez se tienen los distintos archivos y datos cargados, se segmenta el lunar en cada una de las fotografías, mediante una llamada a un método estático de la clase *MoleProcessor.java*, que será vista más adelante. Después, utilizando un método estático de esta misma clase, se obtienen las características de

interés del lunar segmentado en cada una de las fotografías, es decir, el diámetro (tamaño), el matiz (tonalidad) y la correlación de la forma.

Cuando se tienen todas las características del lunar en cada una de las fotografías, se muestran los datos al usuario en forma de gráficas. El usuario puede utilizar diversos botones en la vista de la actividad para seleccionar la gráfica que quiere visualizar en cada momento y para obtener información de ayuda de cada una de las gráficas.

Las gráficas contendrán unos límites (mínimos y máximos o máximos, dependiendo de la gráfica) marcados con líneas de un color diferente a la línea y puntos utilizados para representar la información extraída del lunar.

6.2 Clases de apoyo

Es recomendable que las actividades sean clases simples (lo máximo posible) y que no contengan gran cantidad de código. En el caso de la actividad que hace uso de la API *Camera2*, es complicado, pero en el resto de actividades, pueden utilizarse clases de apoyo para externalizar las implementaciones necesarias, consiguiendo que todo funcione correctamente.

6.2.1 Clases de configuración

La clase *Configuration.java* se encarga de mapear los objetos que serán guardados en el archivo JSON de configuración. Cada objeto configuración está compuesto a su vez por dos objetos, *paths* e *imageConfiguration*, que quedan implementados en las siguientes clases:

- *ImageConfiguration.java*: su objetivo es mapear ciertos parámetros de imagen predeterminados, como, por ejemplo, la dimensión de cada lado del recuadro que se recortará en cada imagen al realizar la fotografia, que por defecto es de 512 (píxeles).
- *Paths.java*: mapea las rutas principales que utilizará la aplicación para guardar las fotografías generadas.

6.2.2 Clases para manejar las imágenes

Cuando el usuario va a realizar una nueva fotografía, se le indica un área central en la vista previa de la cámara donde debe colocar el lunar para capturarlo. Una vez se realiza la fotografía, la aplicación recorta la imagen generada por esa

zona, para obtener una subimagen más pequeña y manejable que contenga el lunar. Además, se debe rotar la imagen para que quede su vista previa quede en una posición vertical. Todo este preprocesado, que resultará útil para el análisis posterior, lo realiza la clase *ImageProcessor.java*.

Se ha implementado una clase *ImageModel.java*, que sirve como un modelo inicial para guardar algunos parámetros necesarios cuando el usuario realiza la captura de un lunar con la cámara. En concreto, esta clase recoge el nombre de la nueva fotografía, la distancia de enfoque utilizada para tomarla, la longitud focal de la cámara utilizada, la altura del sensor de la cámara utilizada y la altura de la imagen capturada antes de ser recortada (altura de la imagen original).

Así mismo, se ha implementado una clase *ImagesInformation.java* que contiene una lista de modelos de imagen, y que permite eliminar un modelo dado de la lista y guardar la información de los modelos en un archivo JSON en la memoria externa del dispositivo para hacerla permanente, mapeando cada uno de los modelos de las imágenes capturadas con su respectiva información.

6.2.3 Clases para la interfaz de usuario

Android Studio ofrece un amplio abanico de posibilidades para crear interfaces de usuario simples y estéticas. Una de ellas es el uso de diálogos para notificar al usuario ciertas cosas. Estos diálogos pueden ser modificados para conseguir diálogos en los que se permita al usuario introducir datos en la aplicación, a modo de formulario. Este es el caso de la clase *NewMoleDialog.java*, que implementa un diálogo que pide al usuario un nombre para el nuevo lunar que esté creando. Además, contiene una interfaz que permite que la aplicación pueda recoger la cadena de texto introducida por el usuario.

Las listas predeterminadas que permite mostrar al usuario Android Studio, solo permiten listar cadenas de caracteres. Para poder mostrar listas que combinen, por ejemplo, caracteres y miniaturas de imágenes, es necesario crear un adaptador de *array* personalizado, en este caso, *CustomArrayAdapter.java*. Adapta ítems de la clase implementada *RowItem.java*, que representa cada elemento de la lista y que contiene una cadena de caracteres y una imagen.

Para la vista previa de la cámara, se utiliza una *TextureView*. El problema de esta vista es que no es adaptativa, por lo que su relación de aspecto no depende de la relación de aspecto de la cámara, y la imagen previa que se mostrará al usuario aparecerá, dependiendo del dispositivo, más o menos achatada. Para corregir esto es necesario implementar una clase *AutoFitTextureView.java* que corrija los defectos de la clase mencionada y que pueda adaptarse a la relación de aspecto de la cámara del dispositivo para mostrar la vista previa con las dimensiones deseadas.

6.2.4 Manejador de archivos

La clase *FileManager.java* contiene varios métodos estáticos que pueden utilizarse para crear o eliminar archivos. En el caso de eliminar un archivo, si es una carpeta, se debe borrar primero su contenido de forma recursiva, para evitar errores del Sistema Operativo. Nótese que, en este caso, al borrar, por ejemplo, una fotografía concreta, se debe buscar el mapeo de esa fotografía en el archivo JSON que contiene información sobre las fotografías, para eliminar la información asociada a la fotografía eliminada. Así mismo, se deberán borrar las imágenes segmentadas asociadas a la imagen original eliminada, si existen.

6.3 Clases para analizar los lunares

6.3.1 MoleModel.java

Es una clase utilizada para representar el modelo de cada fotografía de un lunar. Contiene información relevante de cada una de ellas, como el nombre o la fecha de captura (que se obtiene a partir del nombre dentro de la propia clase, en el constructor). También contiene parámetros como las distancias de enfoque con la que se realizó la fotografía e información importante sobre la cámara del dispositivo que se utilizó para ello. Además, en las instancias de esta clase se guardarán los diámetros, matices y correlaciones de forma calculados posteriormente, así como la matriz utilizada para segmentar la imagen y las matrices de la imagen segmentada.

La clase contiene un método que se utiliza para guardar las imágenes segmentadas de la fotografía original asociada a las instancias en el almacenamiento externo del dispositivo.

6.3.2 MoleProcessor.java

Esta clase contiene un método público estático llamado *segmentMole*, que procesa la imagen original obtenida del *MoleModel* que recibe para conseguir las imágenes segmentadas. Primero, se convierte la imagen en original (RGB) al modelo de color HSV para conseguir eliminar algunos pelos que puedan encontrase cerca del lunar en la imagen.

Después, se convierte la imagen en HSV a escala de grises, para facilitar la segmentación, y se le aplica un desenfoque gaussiano para conseguir eliminar los pelos que pudieran quedar porque no hayan sido eliminados completamente con la conversión a otro modelo de color, gracias al suavizado que consigue esta técnica mezclando los colores de pixeles vecinos en la imagen.

Fig. 22: Imagen de un lunar en el brazo derecho. Fuente: Elaboración propia.

Cuando se tiene la imagen en escala de grises y levemente desenfocada, de manera que solo debería quedar un lunar claramente distinguido en ella, se aplica el algoritmo *threshold*, un algoritmo que se basa en un umbral inferior y un umbral superior para convertir una imagen en blanco y negro en una imagen binaria, de manera que los píxeles que tengan un valor dentro de los límites marcados por el umbral serán convertidos a ceros y los que estén fuera serán convertidos a unos, o viceversa.

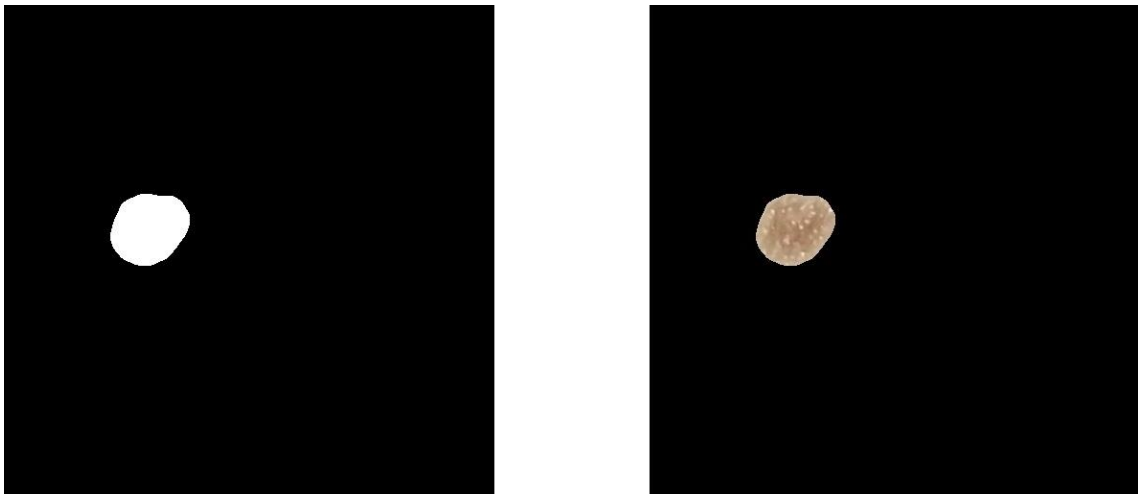

Fig. 23: Imágenes segmentadas de la imagen de la figura 22. Fuente: Elaboración propia.

En este caso, se ha conseguido que los píxeles pertenecientes a los lunares sean convertidos en unos y los píxeles que no pertenezcan a ellos sean convertidos a ceros, creando una matriz binaria que se guardará para el análisis posterior y que además puede utilizarse como máscara para la matriz original dando como resultado una matriz segmentada a color, en la que todos los píxeles que no pertenezcan al lunar serán convertidos al color negro. Esta matriz también será guardada para utilizarla más tarde.

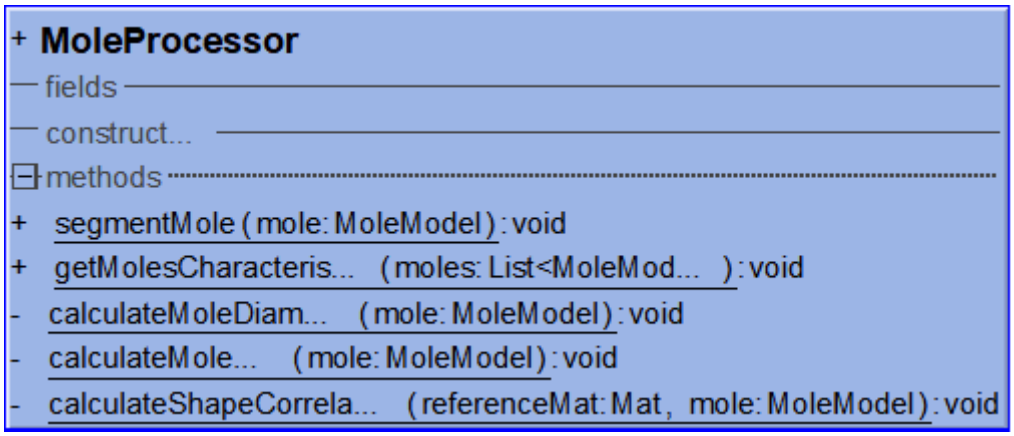

Fig. 24: Clase *MoleProcessor.java*. Fuente: Elaboración propia.

La clase también contiene otro método público estático que recibe una lista de *MoleModel*, de manera que coge el primero de ellos para tomar su matriz binaria como matriz de referencia para la forma del lunar, y para cada lunar, calcula el diámetro, el matiz y la correlación de forma llamando a tres métodos estáticos privados de la propia clase, cada uno de los cuales tiene una de las siguientes funciones:

- Calcular el diámetro del lunar: aplica la fórmula vista en el apartado 2.3.4 para conseguir esta característica. Para ello, utiliza el *MoleModel* recibido, en el cual se han guardado previamente la distancia focal dada en dioptrías, la altura del sensor y la longitud focal de la cámara utilizada para capturar la fotografía, y la altura de la imagen original capturada. Además, se utiliza la imagen segmentada binaria de la imagen para contar el número de píxeles que forman parte del lunar (píxeles con valor distinto de cero).
- Calcular el matiz del lunar: se convierte la imagen original del lunar, obtenida a partir del *MoleModel* recibido, en una imagen en el modelo de color HSV. Esta nueva imagen puede separarse en sus tres canales (*Hue*, *Saturation* y *Value*), de manera que puede utilizarse el canal que contiene los valores del matiz de cada píxel para calcular el matiz medio del lunar, utilizando una media aritmética (Apartado 2.3.4).
- Calcular la correlación de forma: se utiliza una matriz de referencia para compararla con la matriz segmentada binaria obtenida del *MoleModel* recibido para comparar los contornos del lunar utilizando los momentos *hu* (apartado 2.3.4), consiguiendo un factor de correlación que es mejor cuanto más cercano a cero sea su valor.

6.4 Idioma y traducción

La aplicación original fue desarrollada en inglés, pero más tarde ha sido traducida al español. En los proyectos de aplicaciones de Android se define un archivo *strings.xml* donde se definen las cadenas de caracteres que son utilizadas en las interfaces de usuario de la aplicación. Android Studio permite definir múltiples archivos como este, a través de la herramienta *Translations Editor*, de manera que cada uno de ellos será utilizado para un idioma. En este caso, se ha definido un archivo extra para el idioma español, quedando por defecto el archivo que contiene el idioma inglés.

Cuando el usuario abre la aplicación en su dispositivo, se detecta automáticamente el idioma en el que este está configurado. Si el idioma es español, la aplicación mostrará la interfaz completamente en español, y si es cualquier otro, la mostrará en el idioma por defecto, el inglés.

7

Verificación y pruebas

7.1 Fase de pruebas

Tras la implementación de cada uno de los prototipos desarrollados, se han llevado a cabo varias pruebas con un grupo reducido de usuarios, siendo la mayor parte de ellos estudiantes en la Universidad de Málaga. A lo largo de este proceso, se han detectado algunos errores que se han corregido y se han mejorado ciertos aspectos de la aplicación en base a la opinión de estos usuarios.

7.2 Pruebas del primer prototipo

El primer prototipo se encargaba de realizar una llamada a la aplicación externa de la cámara para conseguir capturar una fotografía y guardarla en el almacenamiento externo del dispositivo, organizándolas en un árbol de directorios según las partes del cuerpo y lunar seleccionados.

Al realizar varias pruebas de funcionamiento, se detectaron algunos errores con el manejo de directorios. Para solventarlos, se implementó el archivo JSON de configuración, de manera que las rutas de los directorios principales sean leídas desde dicho archivo en lugar de estar codificadas de forma directa.

7.3 Pruebas del segundo prototipo

El segundo prototipo tenía también como objetivo conseguir capturar fotografías y guardarlas en el almacenamiento externo del dispositivo, organizándolas en un árbol de directorios según las partes del cuerpo y lunar seleccionados, pero en esta ocasión, utilizando una cámara implementada e integrada en la propia aplicación sin necesidad de realizar llamadas a la aplicación externa de la cámara.

Al realizar pruebas de funcionamiento en diferentes dispositivos, se pudo observar que la vista previa de la cámara se estiraba o estrechaba dependiendo de la resolución de pantalla, deformando las imágenes. Para solucionarlo, se implementó una nueva clase que sustituye a la clase *TextureView.java* y hace posible que la vista previa en la pantalla se adapte a la resolución de la cámara del dispositivo utilizada.

Así mismo, se detectó que el área semitransparente que se muestra al usuario en la vista previa de la cámara para que realice la fotografía con el lunar dentro de ella, no se correspondía con las proporciones reales de la parte de la imagen que sería recortada posteriormente. Para solucionarlo, se definió una nueva fórmula que redimensionara el área semitransparente dependiendo del ratio existente entre la resolución de la cámara y la resolución de la vista previa en pantalla.

Por último, surgió el problema de que las imágenes quedaban rotadas 90º hacia la derecha al guardarlas después de realizar una fotografía. Para solucionarlo, se definió un método que las rota a la posición original antes de guardarlas.

7.4 Pruebas del tercer prototipo

El tercer prototipo tenía como objetivo conseguir analizar las fotografías pertenecientes al historial de los lunares, extrayendo de estas características relativas a la forma, el color o el tamaño y utilizando estas para mostrar al usuario la evolución de un lunar a lo largo del tiempo.

Por un lado, se han realizado varias pruebas realizando fotografías a lunares a lo largo del tiempo con condiciones diferentes de luminosidad y acercamiento de la cámara al lunar para verificar el correcto funcionamiento de los análisis.

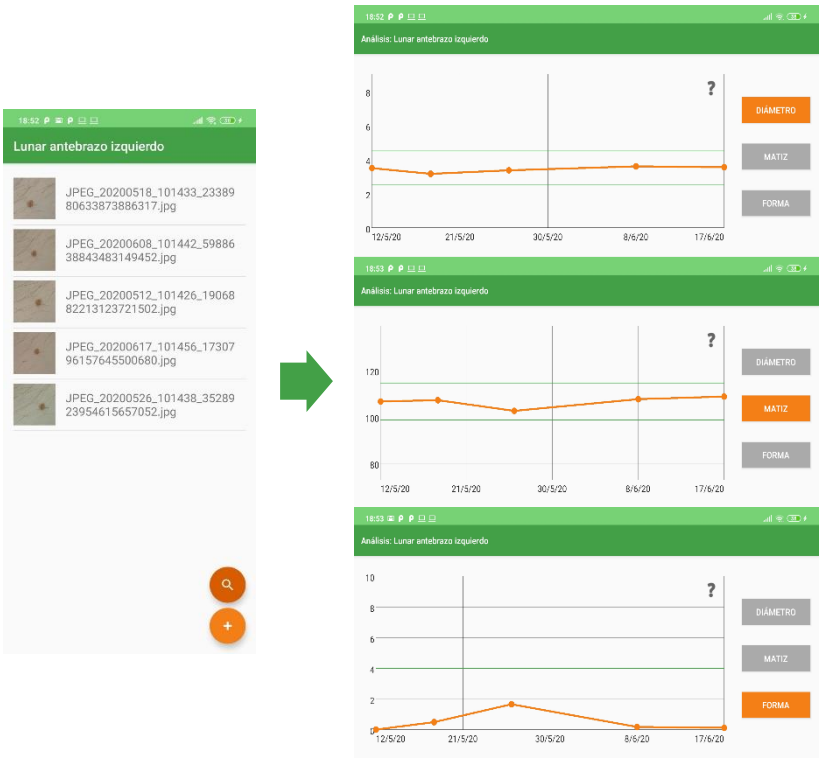

Fig. 25: Prueba del análisis de un lunar en el antebrazo izquierdo. Fuente: Elaboración propia.

Por otro lado, se han realizado varias pruebas utilizando fotografías de lunares diferentes en un mismo historial para verificar que se detectan los cambios de forma, color o tamaño cuando estos se dan.

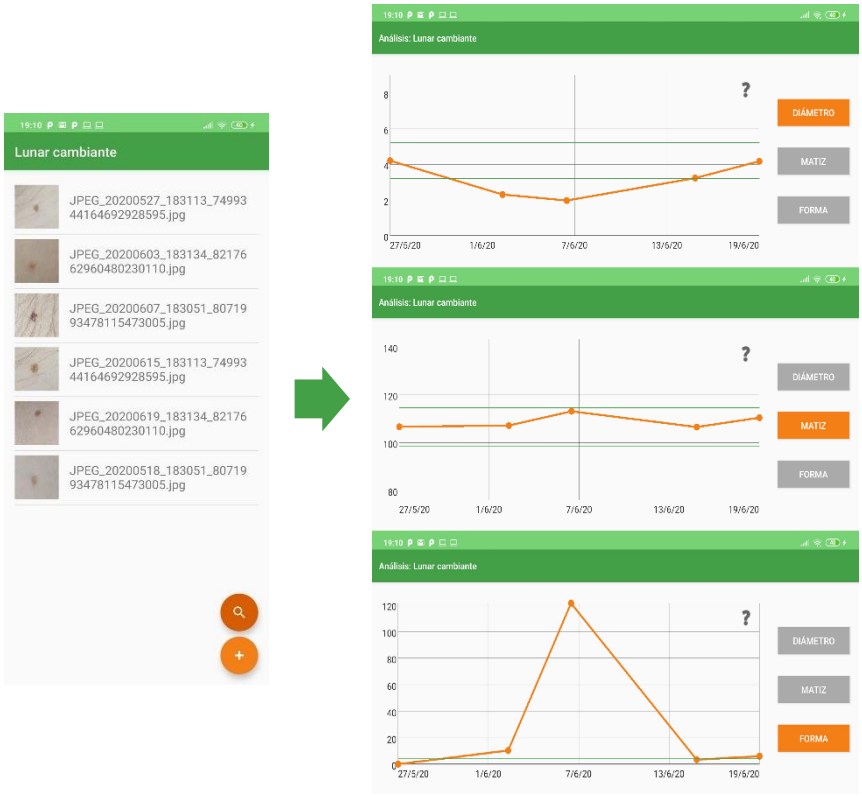

Fig. 26: Pruebas de un análisis utilizando fotografías de diferentes lunares. Fuente: Elaboración propia.

7.5 Pruebas del prototipo final

El último prototipo es una mejora del tercero en el que se integran todas las funcionalidades de forma completa y se refactorizan las vistas para mejorar la estética.

Se han realizado varias pruebas del funcionamiento de la aplicación, añadiendo y eliminando lunares, actualizando el historial de lunares ya existentes y analizando el historial de varios lunares. Todas las pruebas han tenido resultados satisfactorios.

Se han probado además situaciones críticas para comprobar el correcto funcionamiento de la cámara, por ejemplo, minimizando la aplicación durante la utilización de la cámara y volviendo de nuevo a ella para verificar que la cámara no queda bloqueada.

Se detectaron algunos elementos vistas que podían resultar confusos y se cambiaron para facilitar el manejo de la aplicación al usuario.

8

Conclusiones y líneas futuras

8.1 Conclusiones

El desarrollo de la aplicación ha supuesto un reto a nivel personal debido a la poca experiencia con la programación orientada a aplicaciones Android. No obstante, la curva de aprendizaje, a pesar de ser compleja al principio, se suaviza relativamente rápido, y una vez se entiende el funcionamiento de las actividades y las principales librerías a fondo, se vuelve mucho más sencillo acoplar nuevas funcionalidades no solo a la aplicación desarrollada, sino a cualquiera.

Si bien el desarrollo de aplicaciones está muy documentado a nivel oficial y no oficial, existiendo muchísimas herramientas que facilitan el trabajo al desarrollador. Para desarrollar este Trabajo de Fin de Grado se han utilizado las librerías de Androidx, que son más modernas y contienen más funcionalidades y mejoras que las librerías clásicas de Android, con la desventaja de que algunas características pueden llegar a variar bastante y no están bien documentadas.

Una de las partes más complicadas de la implementación fue aprender a manejar de forma correcta la API *Camera2*, utilizada para poder crear una cámara personalizada integrada en la propia aplicación en lugar de realizar una llamada a la aplicación externa de la cámara que viene instalada de forma predeterminada en cada dispositivo. También ha sido complicado el aprendizaje

de cómo manejar los archivos e información general en la memoria externa del dispositivo. Así mismo, al principio es difícil entender el funcionamiento de cómo se comportan los diferentes objetos y textos en las interfaces de usuario, entendiendo cómo y cuándo utilizar cada una de las diferentes capas que ofrece el entorno de desarrollo para crearlas.

Con respecto a la parte de la segmentación y el análisis de lunares, tiene cierta complejidad, y es necesario ir probando diferentes algoritmos y analizando su comportamiento para seleccionar siempre el mejor. Gracias a la extensa documentación y comunidad que tiene detrás OpenCV, pueden ir resolviéndose todos los problemas que van surgiendo.

La realización de este proyecto, por lo tanto, ha conseguido que se entienda mucho mejor el funcionamiento del sistema operativo Android y cómo desarrollar aplicaciones para este, adquiriendo además algunas nociones de cómo funciona la segmentación de imágenes y la extracción de características posterior.

El objetivo principal de este proyecto era dar la oportunidad a los usuarios de este sistema operativo de poder realizar un seguimiento de los lunares que tengan en diferentes partes del cuerpo de una forma sencilla y fácil de entender. Para ello, era necesario conseguir implementar una aplicación intuitiva y que consiguiera explicarse por sí misma, integrando algunas de las funcionalidades que ofrece una librería de visión por computador tan potente como es OpenCV. Todo esto se ha conseguido de forma satisfactoria, solucionando todos los problemas que han surgido a lo largo del desarrollo de la aplicación.

Para finalizar, destacar el gran potencial que tiene la visión por computador (y la Inteligencia Artificial en general), no solo en el ámbito relacionado con la medicina, sino en otros como pueden ser la conducción, la identificación de personas o la seguridad. Es una disciplina que bien utilizada puede aportar muchísimos avances.

8.2 Líneas futuras

Si bien se han cubierto los objetivos iniciales de este Trabajo de Fin de Grado con la aplicación desarrollada, existen ciertas mejoras y ampliaciones que podrían realizarse:

- Migración de la aplicación a otros sistemas operativos para *Smartphones* y *Tablets*, como, por ejemplo, *iPhone OS* (iOS) o *Windows Phone* (WP). Con algo de conocimiento sobre la programación orientada a estos sistemas operativos, no debería ser demasiado complejo.
- Optimización de la aplicación para reducir al mínimo el espacio requerido en el dispositivo para instalarla.
- Cambio del análisis de los lunares para que pueda realizarse en tiempo real en la vista previa de la cámara. Para ello sería necesario utilizar la cámara que ofrece la librería OpenCV en lugar de utilizar la cámara que ofrece la API *Camera2*.
- Mejora de los algoritmos de segmentación y análisis, aumentando su precisión y rendimiento en un número mayor de condiciones de luminosidad o con cámaras con resoluciones más pequeñas.
- Considerar la posibilidad de que los resultados obtenidos en el análisis del historial de un lunar pudieran enviarse vía correo electrónico, SMS u otro servicio de mensajería instantánea, para, por ejemplo, notificar al doctor de cada paciente.
- Realizar un estudio a fondo junto a profesionales médicos y utilizar redes neuronales entrenadas para que la aplicación sea capaz de detectar los melanomas con una precisión muy alta.
- Considerar la posibilidad de permitir crear diferentes cuentas de usuario y que las fotografías tomadas se guarden de alguna manera en la nube en lugar de en el propio almacenamiento del dispositivo.

Referencias

[1] AECC, *Cáncer de Piel*. [En línea]. Enlace: https://bit.ly/2WvSjF7 (Consulta en enero de 2020).

[2] ACS, *¿Qué es el cáncer de piel tipo melanoma?*. [En línea]. Enlace: https://bit.ly/2WwkFiy (consulta en enero de 2020).

[3] ACS, *¿Se puede encontrar el cáncer de piel tipo melanoma en sus comienzos?*. [En línea]. Enlace: https://bit.ly/3b8x3KV (consulta en enero de 2020).

[4] NIH, *Lunares comunes, nevos displásicos y el riesgo de melanoma*. [En línea]. Enlace: https://bit.ly/2YLQNBg (consulta en febrero de 2020).

[5] NIH, *Cánceres de piel (incluye el melanoma)*. [En línea]. Enlace: https://bit.ly/2SGs3Xg (consulta en febrero de 2020).

[6] ACS, *Acerca del cáncer de piel tipo melanoma*. [En línea]. Enlace: https://bit.ly/2WwkFiy (consulta en marzo de 2020).

[7] AECC, *Cáncer de Piel: Factores de riesgo*. [En línea]. Enlace: https://bit.ly/2L730IQ (consulta en marzo de 2020).

[8] Euromelanoma, *Check for skin Cancer – What to look for*. [En línea]. Enlace: https://bit.ly/2WbA7lk (consulta en abril de 2020).

[9] Jason Brownlee, *A gentle introduction to Computer Vision*, 19/03/2019. [En línea]. Enlace: https://bit.ly/3b7fBXe (consulta en mayo de 2020).

[10] Iván de Paz Centeno, *La Visión Artificial y el procesamiento de imágenes*, 06/05/2019. [En línea]. Enlace: https://bit.ly/3fp28xh (consulta en mayo de 2020).

[11] RSIP Vision, *What's the Difference between Computer Vision, Image Processing and Machine Learning?*. [En línea]. Enlace: https://bit.ly/2L2Got9 (consulta en mayo de 2020).

[12] Nora La Serna Palomino, Ulises Román Concha, *Técnicas de Segmentación en Procesamiento Digital de Imágenes*. [En línea]. Enlace: https://bit.ly/2WAaDOG (consulta en mayo de 2020).

[13] Stack OverFlow, *Calculate image size of an object from real size*. [En línea]. Enlace: https://bit.ly/2YfnbKO (consulta en mayo de 2020).

[14] Wikipedia, *Media aritmética*. [En línea]. Enlace: https://bit.ly/37xY8XG (consulta en junio de 2020).

[15] Wikipedia, *Momentos de imagen*. [En línea]. Enlace: https://bit.ly/30KmFr0 (consulta en junio de 2020).

[16] Wikipedia, *Momentos de imagen*. [En línea]. Enlace: https://bit.ly/30KmFr0 (consulta en junio de 2020).

[17] Doxygen, *OpenCV: Hu moments*. [En línea]. Enlace: https://bit.ly/3frZdTv (consulta en junio de 2020).

[18] EDUCBA, *Java vs Kotlin*. [En línea]. Enlace: https://bit.ly/3bE78uA (consulta en mayo de 2020).

[19] W3C, *Extensible Markup Language (XML)*. [En línea]. Enlace: https://bit.ly/3cEMOuC (consulta en mayo de 2020).

[20] Xiaomi, *Redmi Note 8*. [En línea]. Enlace: https://bit.ly/3bAoPeF (consulta en mayo de 2020).

[21] OpenCV team, *About*. [En línea]. Enlace: https://bit.ly/3cExdLE (consulta en mayo de 2020).

[22] Gradle Inc, *Gradle Build Tool*. [En línea]. Enlace: https://bit.ly/2zEsRFD (consulta en mayo de 2020).

[23] Github Inc, *Github*. [En línea]. Enlace: https://bit.ly/2AuMgsV (consulta en mayo de 2020).

[24] JSON ORG, *Introducing JSON*. [En línea]. Enlace: https://bit.ly/2Z9ziv5 (consulta en mayo de 2020).

[25] Android Developers, *Descripción general de proyectos*. [En línea]. Enlace: https://bit.ly/367nPhe (consulta en mayo de 2020).

[26] Android Developers, *Documentation: Activity*. [En línea]. Enlace: https://bit.ly/3cDfNPB (consulta en mayo de 2020).

[27] Android Developers, *Documentation: Camera2*. [En línea]. Enlace: https://bit.ly/3g6asCs (consulta en mayo de 2020).

Apéndice A

Manual de Instalación

A.1 Requerimientos

Para que la aplicación pueda funcionar correctamente en el dispositivo utilizado, este debe cumplir las siguientes condiciones:

- Debe ser un ***Smartphone*** o una ***Tablet***.
- Debe utilizar el **sistema operativo Android**, concretamente con la versión 5.0 (*Lollipop*) o superior.
- Debe tener un **espacio de almacenamiento disponible de al menos 80.1 MB**, si bien se recomienda algo de espacio extra para poder almacenar las fotografías que se realicen.
- Debe disponer de al menos una **cámara**, preferentemente con flash.

A.2 Instalación a partir del paquete de aplicación (APK)

Si se tiene al archivo con extensión *.apk* de la aplicación, es decir, el paquete de aplicación Android, se puede instalar la aplicación facilmente ejecutándolo. Para ello, es necesario habilitar primero la instalación de aplicaciones desde “orígenes desconocidos” en el dispositivo, si esta opción no estaba habilitada previamente. Para las versiones más recientes de Android:

- Acceder a la aplicación de **Configuración** del dispositivo.
- Entrar en el apartado **Seguridad → Ajustes adicionales**.
- Entrar en el apartado **Instalar aplicaciones de fuentes externas**.
- Seleccionar la aplicación *Molecare* y activar los permisos.

A.3 Instalación a partir del proyecto de Android Studio

Para poder instalar la aplicación a partir del proyecto de Android Studio, es necesario seguir una serie de pasos:

1. Es necesario tener instalado Android Studio instalado en el ordenador. Si ya se tiene instalado, continuar desde el paso 2. Si no se tiene instalado, puede descargarse desde https://developer.android.com/studio/.
2. Abrir Android Studio e importar el proyecto:
    - Guardar el proyecto en un directorio del ordenador.
    - Pulsar sobre **File → Open** en Android Studio.
    - Buscar la ruta donde se ha guardado el proyecto y seleccionarlo.
3. Activar la instalación USB / depuración en el dispositivo donde se instalará la aplicación:
    - Acceder a la aplicación de **Configuración** del dispositivo.
    - Para Android 8.0 o superior, seleccionar **Sistema**. Para versiones anteriores, continuar en el paso siguiente.
    - Seleccionar **Acerca del teléfono** y pulsar siete veces sobre **Número de compilación**.
    - Volver a la pantalla anterior y seleccionar **Opciones para desarrolladores**.
    - Buscar la opción **depuración de USB** y activarla.
4. Conectar mediante un cable USB el dispositivo donde se instalará la aplicación al ordenador.
5. En la ventana de herramientas de Android Studio, seleccionar como dispositivo para ejecutar la aplicación el dispositivo que se ha conectado al ordenador en el paso anterior.
6. En la ventana de herramientas de Android Studio, seleccionar el botón “ejecutar”, representado con un triángulo verde.

Apéndice B

Manual de Usuario

B.1 Introducción

Molecare es una aplicación con el objetivo de ayudar al usuario a realizar un seguimiento de sus lunares para que se pueda realizar una evaluación del riesgo que estos tienen de convertirse en melanomas. Al entrar en la aplicación, se muestra una ventana de bienvenida con el logo y el nombre de esta, y pasados dos segundos, se mostrará la ventana principal, que contiene un avatar humano que representa las diferentes partes del cuerpo disponibles a las que acceder para ver, añadir, actualizar, eliminar y analizar lunares.

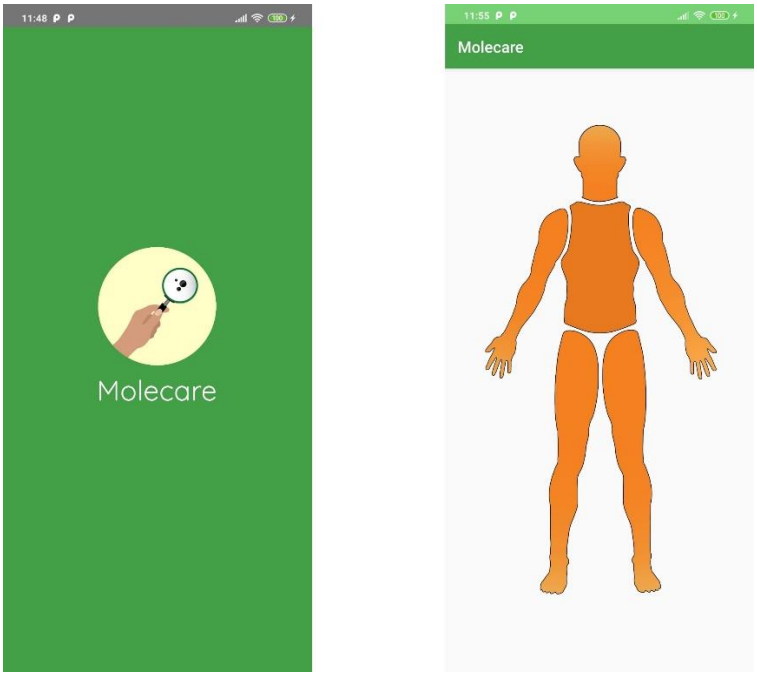

Fig. 27: Ventana de bienvenida (izquierda) y ventana principal (derecha). Fuente: Elaboración propia.

B.2 Ver lista de lunares

Para ver la lista de lunares existentes en una parte del tiempo, solo hay que pulsar sobre la parte del cuerpo correspondiente en el avatar humano que puede verse en la ventana principal de la aplicación, que está dividido en seis partes: cabeza, torso, brazo izquierdo, brazo derecho, pierna izquierda y pierna derecha.

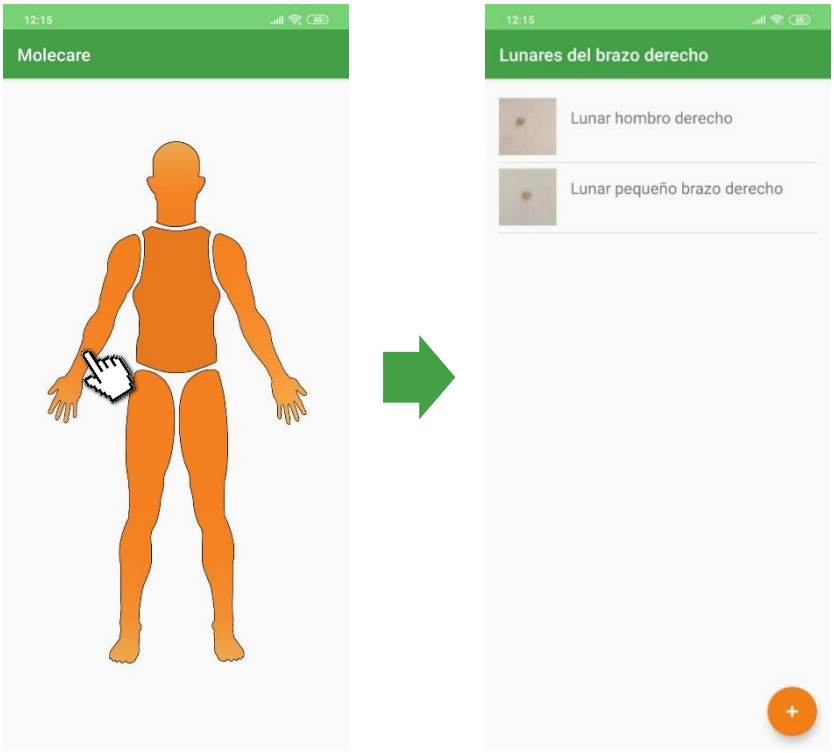

Fig. 28: Ver lista de lunares de una parte del cuerpo. Fuente: Elaboración propia.

B.3 Añadir lunar

Para añadir un lunar, se debe navegar hasta la ventana de la lista de lunares de una parte del cuerpo (Apartado B.2). Una vez ahí, hay que pulsar sobre el botón de “Añadir lunar”, representado por un círculo naranja con el símbolo “+” en el centro. Al hacer esto, aparecerá un diálogo preguntando por el nombre con el que se quiere guardar el nuevo lunar. Una vez introducido el nombre, que no puede ser vacío ni puede ser el mismo que el nombre de uno de los lunares ya existentes en esa parte del cuerpo, se abrirá la vista previa de la cámara con un mensaje informativo que explica el procedimiento a seguir. Tras leerlo, puede cerrarse pulsando sobre “OK” o sobre cualquier parte de la pantalla fuera del diálogo. Ahora puede realizarse la fotografía para el nuevo lunar, que aparecerá en la lista de lunares de la parte del cuerpo que se hubiera seleccionado.

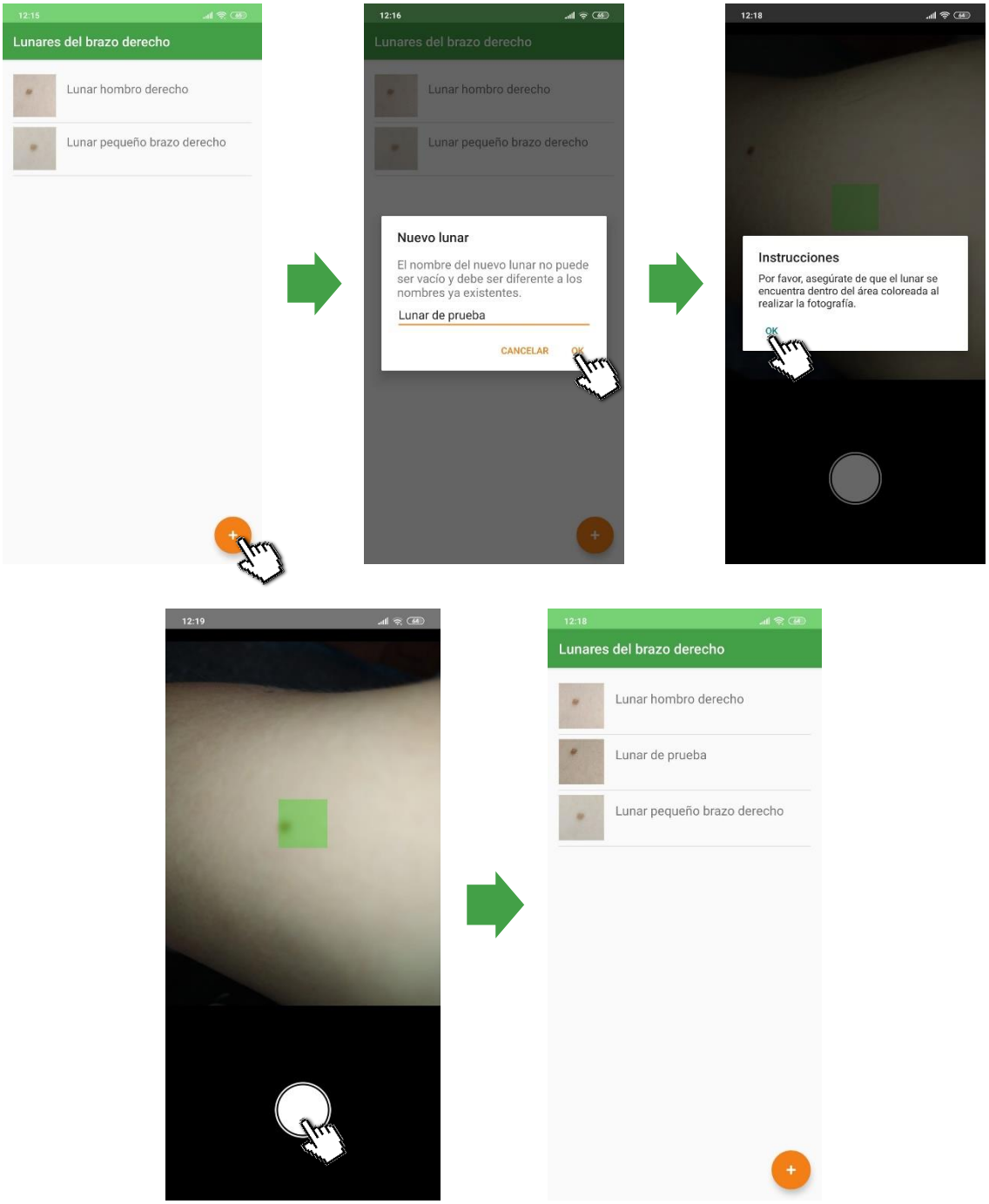

Fig. 29: Añadir lunar en una parte del cuerpo. Fuente: Elaboración propia.

B.4 Ver historial de un lunar

Desde la ventana de la lista de lunares en una parte del cuerpo seleccionada (Apartado B.2), también puede seleccionarse un lunar ya existente para ver su historial de fotografías pulsando sobre él en la lista. Se mostrará una nueva lista que contendrá las fotografías realizadas de ese lunar, en la que los nombres de cada una de ellas representan, en formato POSIX, la fecha y hora en la que se tomó cada una de ellas.

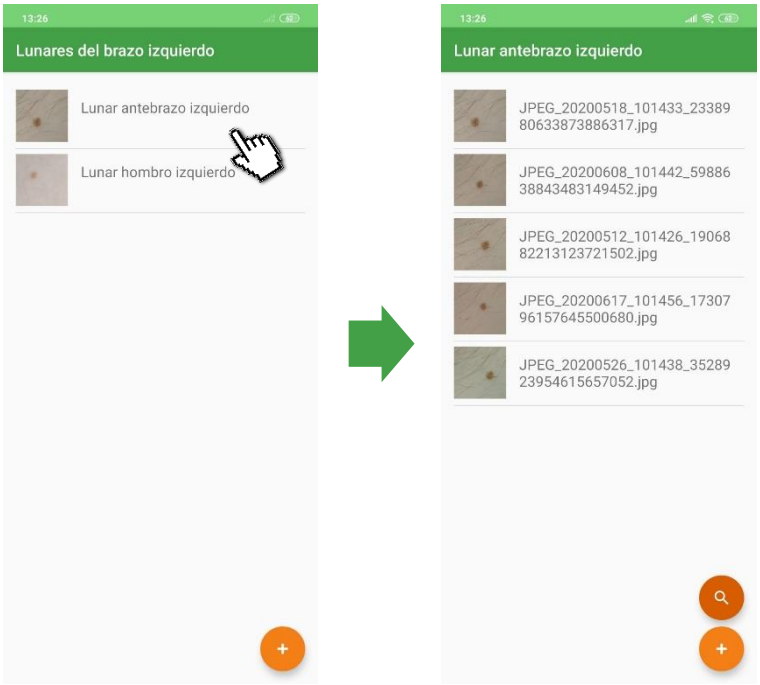

Fig. 30: Ver historial de un lunar ya existente. Fuente: Elaboración propia.

B.5 Actualizar el historial de un lunar

Desde la ventana del historial de un lunar (Apartado B.4) se puede actualizar el historial de dicho lunar pulsando sobre el botón de “Actualizar lunar”, representado por un círculo naranja con el símbolo “+” en el centro. A partir de aquí, el procedimiento es muy parecido al que se sigue para crear un nuevo lunar, solo que en este caso no será necesario escribir un nombre, ya que los nombres serán asignados a las fotografías de forma automática dependiendo del momento en el que estás se realicen. Al terminar, la nueva fotografía aparecerá en el historial del lunar que se hubiera seleccionado.

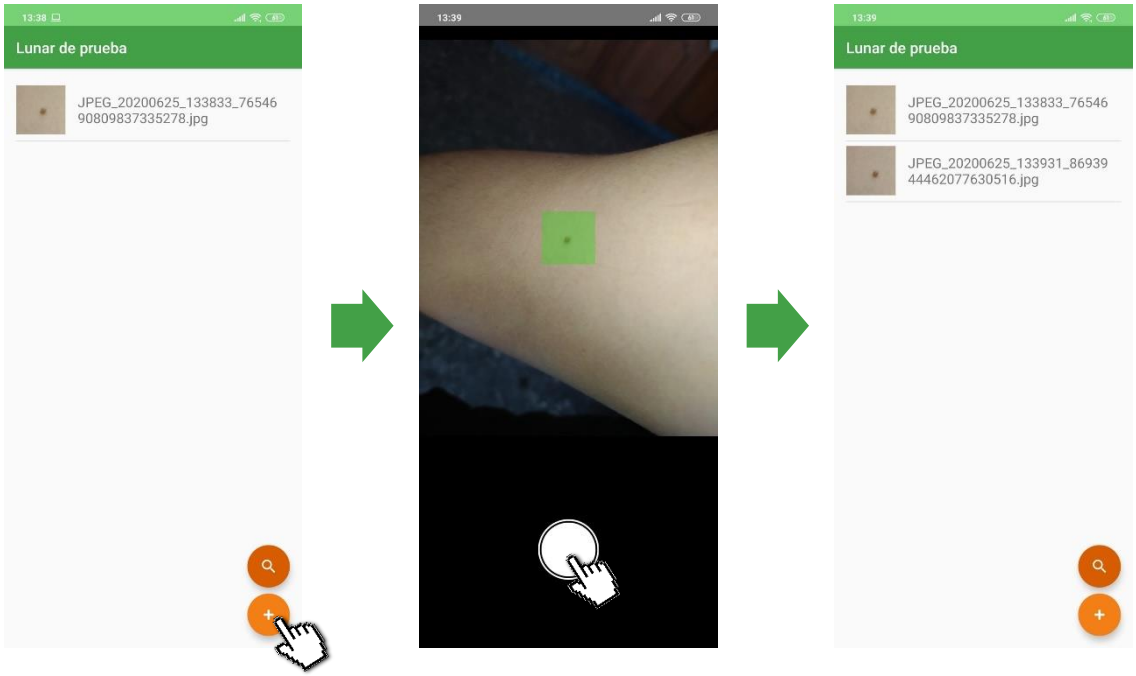

Fig. 31: Actualizar historial de un lunar ya existente. Fuente: Elaboración propia.

B.6 Eliminar un lunar o una fotografía de su historial

Desde la ventana de la lista de lunares en una parte del cuerpo seleccionada (Apartado B.2), puede eliminarse un lunar por completo manteniendo pulsado sobre él. Aparecerá un mensaje informativo para confirmar que realmente se quiere eliminar el lunar, y al aceptarlo, el lunar desaparecerá de la lista de lunares de la parte del cuerpo seleccionada.

Si lo que se quiere es eliminar solo una fotografía del historial de un lunar, se debe mantener pulsado sobre esta en la ventana del historial de un lunar. Aparecerá un mensaje informativo para confirmar que realmente quiere eliminarse la fotografía, y al aceptarlo, la fotografía desaparecerá del historial del lunar seleccionado.

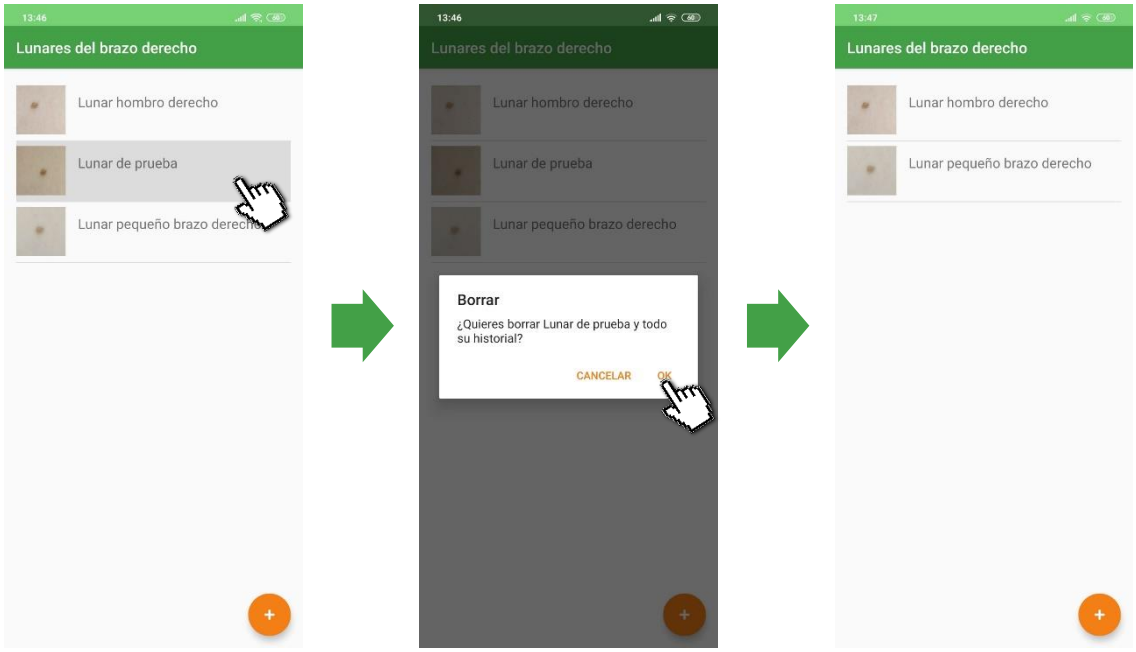

Fig. 32: Eliminar un lunar existente y todo su historial. Fuente: Elaboración propia.

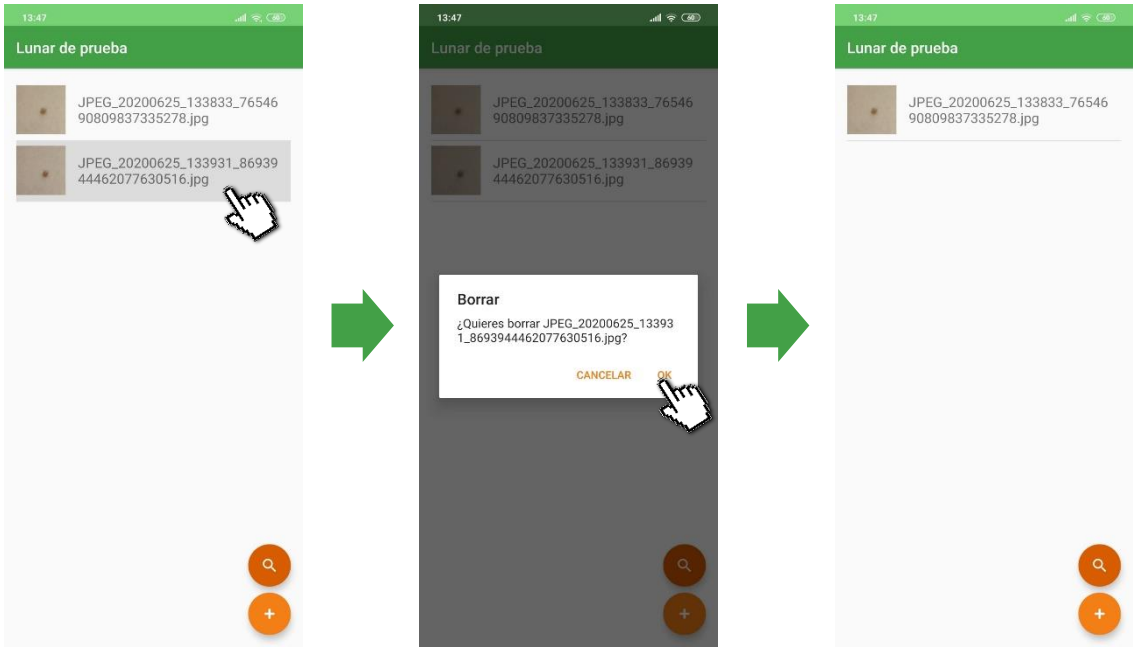

Fig. 33: Eliminar una fotografía del historial de un lunar ya existente. Fuente: Elaboración propia.

B.7 Analizar lunar

Desde la ventana del historial de un lunar (Apartado B.4) se puede analizar un lunar para ver su evolución a lo largo del tiempo. Para ello, hay que pulsar sobre el botón de “Analizar lunar”, representado con un círculo de un naranja más oscuro que el resto de botones, y una lupa blanca en el centro. La aplicación procesará las diferentes fotografías tomadas del lunar y mostrará los datos extraídos en tres gráficas: una para el diámetro a lo largo del tiempo, otra para el matiz a lo largo del tiempo y otra para la correlación de forma a lo largo del tiempo. Cada una de las gráficas contiene un botón en la esquina superior derecha (signo de interrogación) que aportará más información sobre el significado de cada una de las gráficas.

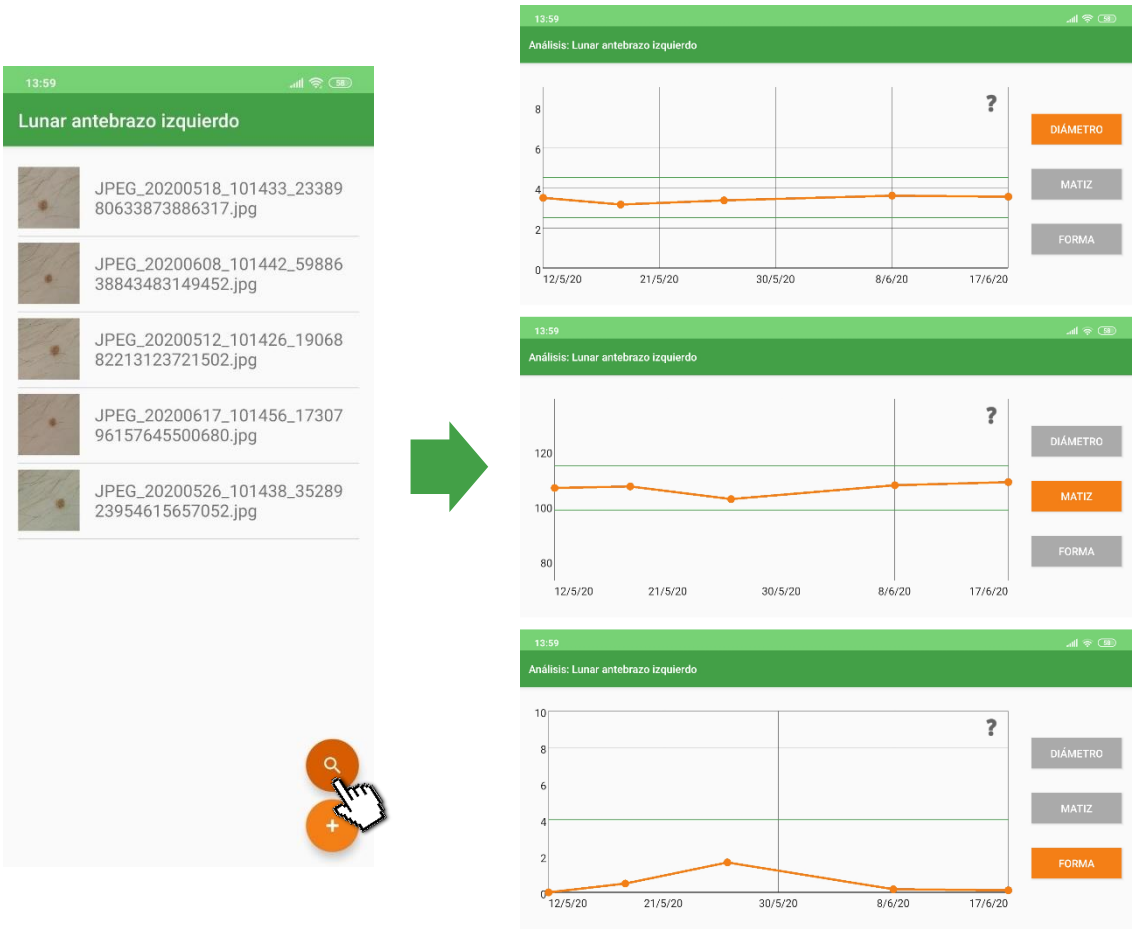

Fig. 34: Analizar un lunar ya existente. Fuente: Elaboración propia.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga