



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Aplicación Android para la
detección de retinopatía
diabética**



Presentado por Miguel Fuente García
en Universidad de Burgos — 4 de julio de 2023
Tutor: D. Daniel Urda Muñoz y D. Nuño
Basurto Hornillos

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	12
Apéndice B Especificación de Requisitos	23
B.1. Introducción	23
B.2. Objetivos generales	23
B.3. Catálogo de requisitos	24
B.4. Actores	26
B.5. Especificación de requisitos	26
Apéndice C Especificación de diseño	45
C.1. Introducción	45
C.2. Diseño de datos	45
C.3. Diseño procedimental	46
C.4. Diseño arquitectónico	48
C.5. Diseño de pruebas	51
C.6. Diseño de la aplicación	52
Apéndice D Documentación técnica de programación	57

D.1. Introducción	57
D.2. Estructura de directorios	57
D.3. Manual del programador	58
D.4. Compilación, instalación y ejecución del proyecto	61
D.5. Pruebas del sistema	63
Apéndice E Documentación de usuario	65
E.1. Introducción	65
E.2. Requisitos de usuarios	65
E.3. Instalación	65
E.4. Manual del usuario	66

Índice de figuras

A.1. Sprint 1	3
A.2. Sprint 2	4
A.3. Sprint 3	5
A.4. Sprint 4	6
A.5. Sprint 5	7
A.6. Sprint 6	7
A.7. Sprint 7	8
A.8. Sprint 8	9
A.9. Sprint 9	9
A.10. Sprint 10	10
A.11. Sprint 11	11
A.12. Sprint 12	11
A.13. Sprint 13	12
A.14. Sprint 14	13
A.15. Licencia GPLv3. Imagen obtenida de https://choosealicense.com/licenses/	20
 B.1. Diagrama de casos de uso	27
C.1. Diagrama entidad relación	46
C.2. Diagrama de secuencia representando a la clase Foto	47
C.3. Diagrama de secuencia representando a la clase SeleccionarRNE	48
C.4. Modelo-Vista-Presentador	49
C.5. Diagrama de clases de Utils e Interprete	50
C.6. Diagrama de clases relativos a la base de datos	51
C.7. Interfaz inicial de la aplicación	54
C.8. Logo obtenido de Dalle	54
C.9. Logo final	54

C.10. Interfaz final de la aplicación	55
C.11. Diagrama de navegación de la aplicación	56
D.1. Android Studio con un dispositivo virtual.	59
D.2. Gradle incluye el JDK.	60
D.3. Dispositivos virtuales Android.	60
D.4. SDKs ofrecidos.	60
E.1. Interfaz para seleccionar paciente.	67
E.2. Interfaces para obtener la foto.	68
E.3. Interfaz para seleccionar la red.	69

Índice de tablas

A.1. Costes de personal	13
A.2. Costes de Hardware	14
A.3. Costes de Software	14
A.4. Costes adicionales	15
A.5. Costes totales	15
A.6. Dependencias del proyecto Android	18
A.7. Dependencias del proyecto Python	19
B.1. CU-1 Gestión de usuarios.	28
B.2. CU-2 Gestión de médico.	29
B.3. CU-3 Mostrar datos médico.	29
B.4. CU-4 Gestión de pacientes.	30
B.5. CU-5 Mostrar datos pacientes.	30
B.6. CU-6 Escoger un paciente.	31
B.7. CU-7 Gestión de informes.	32
B.8. CU-8 Creación de informes.	33
B.9. CU-9 Obtención de fecha.	34
B.10.CU-10 Obtención de ojo.	34
B.11.CU-11 Obtención de foto.	35
B.12.CU-12 Hacer foto.	36
B.13.CU-13 Escoger foto.	37
B.14.CU-14 Calidad foto.	38
B.15.CU-15 Obtención del resultado.	39
B.16.CU-16 Escoger red neuronal.	40
B.17.CU-17 Mostrar informe.	41
B.18.CU-18 Iniciar sesión.	42
B.19.CU-19 Iniciar sesión como médico.	43
B.20.CU-20 Iniciar sesión como invitado	44

C.1. Pruebas CP-1	52
C.2. Pruebas CP-2	52
C.3. Pruebas CP-3	53
C.4. Pruebas CP-4	53

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se expone como se ha planteado el proyecto a lo largo del tiempo, y su viabilidad tanto económica como legal.

A.2. Planificación temporal

Para la planificación temporal, cabe destacar que se ha seguido la metodología SCRUM, la cual no se ha podido seguir al 100 %, puesto que es una metodología planeada para equipos de trabajo, siendo este un proyecto personal de final de carrera.

- Se programó el *sprint* de dos semanas, donde, se podrían haber hecho de una semana, pero la metodología SCRUM recomienda de dos a cuatro.
- Durante, el *sprint* se planean unas tareas a terminar en el periodo de tiempo.
- Cada tarea tiene un objetivo y una estimación; se organizan en un tablero canvas con las siguientes columnas consideradas:
 - “tareas a realizar” : Son las tareas planeadas para el sprint actual o próximos.
 - “tareas en proceso” : Son aquellas tareas que se están realizando en el momento.

- “tareas terminadas” : Son las tareas que ya se han terminado, las cuales se comprueban al final del sprint si se han terminado correctamente, en tal caso se cierran.
 - “tareas cerradas” : Son aquellas tareas que se han cerrado en *sprints* anteriores, las cuales se han implementado correctamente en el producto final.
- Con los gráficos *burndown*, se puede observar el proceso del proyecto de cada *sprint*
 - Como se ha comentado, al terminar el sprint, se realizaba una reunión personal, donde se comprobaba si se habían completado correctamente las tareas terminadas. En caso de no ser así, se añadía para el siguiente *sprint*, además, se añadían las nuevas tareas a realizar en el siguiente.

Esta metodología se ha implementado usando la herramienta ZenHub, la cual permite hacer los apartados anteriores con facilidad. Para asignar los pesos de la tarea, se utiliza una aproximación de la secuencia de fibonacci. Donde los valores dependen del tiempo y dificultad que llevará la tarea, esta estimación, a lo largo del proyecto va siendo más precisa, puesto que, al principio, no se controlaban los tiempos que conllevarían las dificultades del proyecto.

Los *sprints* que se realizaron se comentan a continuación.

Sprint 0 - Anterior al 27/12/2022

En este sprint inicial, del que no se poseen fechas, se realizaron diversas reuniones, en las cuales se explicaron en que iba a consistir el proyecto.

Durante este sprint, se estuvo investigando sobre aplicaciones Android con el mismo objetivo, sobre las que basar el proyecto. Encontrando de esta forma la aplicación Ret-iN CaM. La cual sirve para realizar estudios sobre los pacientes, de forma que se pueda ver la evolución. Además, se realizaron búsquedas sobre redes neuronales ya creadas las cuales añadir a la aplicación.

Como este sprint fue más un periodo de búsqueda de información, sin realizar ningún trabajo real, no se contabilizó el trabajo realizado.

Sprint 1 - 27/12/2022 - 10/01/2023

En este primer Sprint, se estuvo pensando con que herramienta realizar el proyecto. Teniendo como objetivo crear tanto el proyecto, como el LaTeX.

Para la aplicación móvil, se pensaba utilizar o Android Studio o Unity, de Android Studio ya se tenían conocimientos de uso, pero de Unity no, por ello, se realizó un proyecto "Hola Mundo" en Unity, para ver la complejidad; al ver que la herramienta estaba más destinada a aplicaciones gráficas, se decidió realizar la aplicación en Android Studio, creando el proyecto.

Para la realización del documento, se pensó hacer de forma local, utilizando MiKTeX, pero por facilidad de uso de los tutores, quienes preferían el uso de OverLeaf, que les permitía corregir de forma concurrente el documento, se utilizó esta.

Como este primer sprint se estaba empezando, y además era Navidad, se olvidó mover las tareas de ZenHub a terminadas, y la gráfica que proporciona ZenHub es incorrecta. Se puede ver en A.1

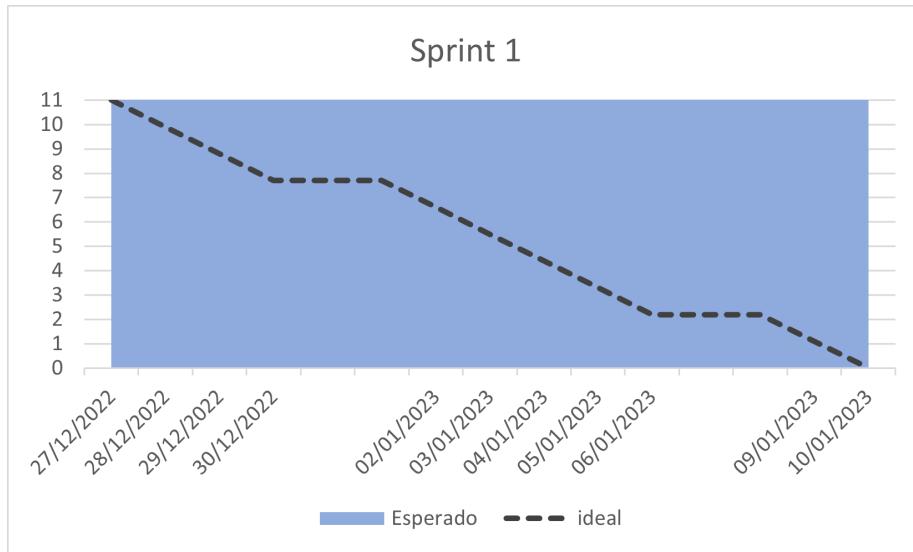


Figura A.1: Sprint 1

Sprint 2 - 10/01/2023 - 24/01/2023

En el segundo Sprint se planeó hacer el menú principal de la aplicación, para ir haciendo la interfaz de usuario y a su vez, se hizo una investigación sobre que colores relaciona el ser humano con la retinopatía diabética.

Para hacer el menú, primero se hizo un boceto a mano, por ese motivo, se puso un valor alto de la estimación. Por otro lado, se investigó sobre como implementar las redes neuronales en Android Studio, e ir añadiendo apartados al documento LaTeX.

De estas tareas, no se pudieron terminar ni la de investigar como implementar una red neuronal en Android Studio, ni añadir el apartado en concreto del LaTeX. Aunque si que se avanzó, SCRUM no permite dividir la tarea en parte terminada y parte que no, por lo que se va a considerar que no se han realizado.

El gráfico burndown se vería incorrectamente, debido al inconveniente ocurrido en el primer sprint, arreglando este defecto, se vería en la imagen A.2.

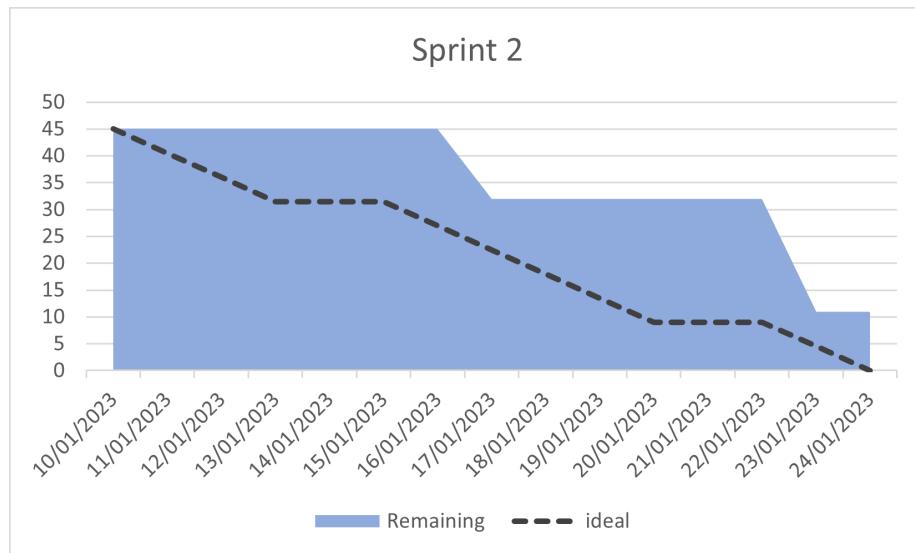


Figura A.2: Sprint 2

Sprint 3 - 24/01/2023 - 07/02/2023

En el tercer Sprint, se añadieron las tareas no terminadas en el anterior, y se añadió también, el terminar el boceto inicial de la aplicación. Al iniciar el sprint, se pensó en que la aplicación tenía que tener nombre y un logo, pero se programó para futuros sprints porque ya se tenían varias tareas para este. Además, no se pudo avanzar mucho en la introducción del LaTeX porque era necesario aclarar una duda presencialmente, y la reunión se hizo a finales del sprint.

Como se iba avanzando en las tareas simultáneamente, tanto terminar el boceto de la aplicación y buscar información de la implementación, se terminaron el mismo día. A.3

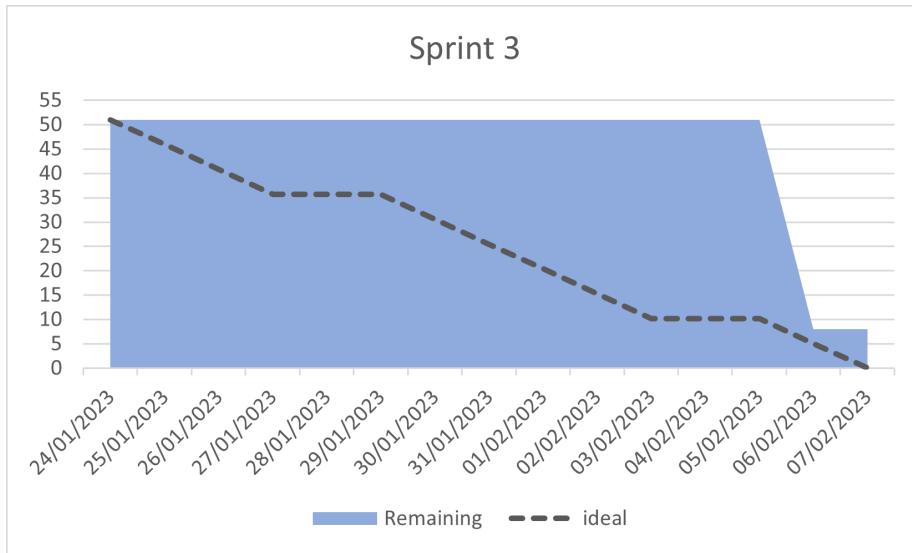


Figura A.3: Sprint 3

Sprint 4 - 07/02/2023 - 21/02/2023

Para el cuarto sprint, se planeó realizar como una investigación sobre cómo se podría incluir la aplicación en la Play Store, y sobre como integrar Tensorflow en Android Studio.

Al investigar sobre la Play Store, se dio cuenta que era necesario realizar un pago de 25\$, por ese mismo motivo, se preguntó a los tutores, quienes dijeron que no hacía falta subir la aplicación a ninguna tienda.

Por otro lado, se terminaron las demás tareas, cumpliendo los objetivos del sprint. Y aunque se podría haber eliminado la tarea de investigación de la Play Store, el tiempo gastado en la investigación se había realizado, por lo que no se eliminó.

El burndown quedaría como indica la figura A.4.

Sprint 5 - 21/02/2023 - 07/03/2023

Para este sprint, como no había tareas atrasadas, se añadieron la creación del logo y el nombre de la aplicación, de la investigación anterior de como

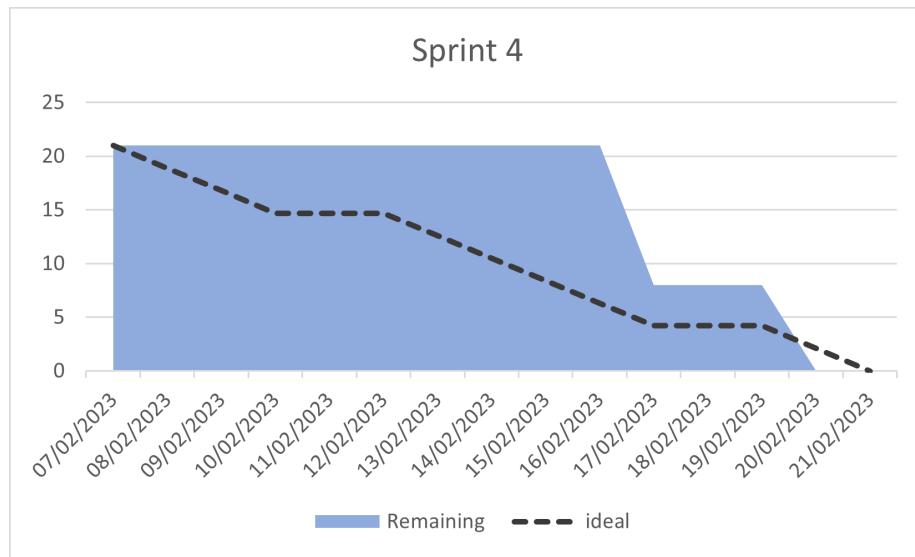


Figura A.4: Sprint 4

implementar una red neuronal en Android Studio, el caso concreto de una red neuronal .h5; realizar cambios en la interfaz para añadir y quitar funcionalidad. Y añadir una base de datos; para lo que se pensó realizar un conjunto de clases java, las cuales simulaban la base de datos, y con un constructor, inicializar los datos.

Debido a que no se me da muy bien en el diseño artístico, se me ocurrió la idea de utilizar la herramienta DALL · E2 para un diseño preliminar, una vez se proporcionó un resultado aceptable, se editó el ícono eliminando ruido, y cambiando los colores.

Debido a las diversas actividades designadas en este sprint, no se pudo realizar la base de datos.

Obteniendo el burndown [A.5](#)

Sprint 6 - 07/03/2023 - 21/03/2023

En el sexto sprint, se añadieron realizar el apartado de trabajos relacionados del documento LaTeX, y probar una integración de una red neuronal básica. Ambas tareas se terminaron, y se avanzó en la creación de la base de datos; pero no se terminó debido a que se querían añadir más datos, los cuales solo se crearon un médico, un paciente y un informe.

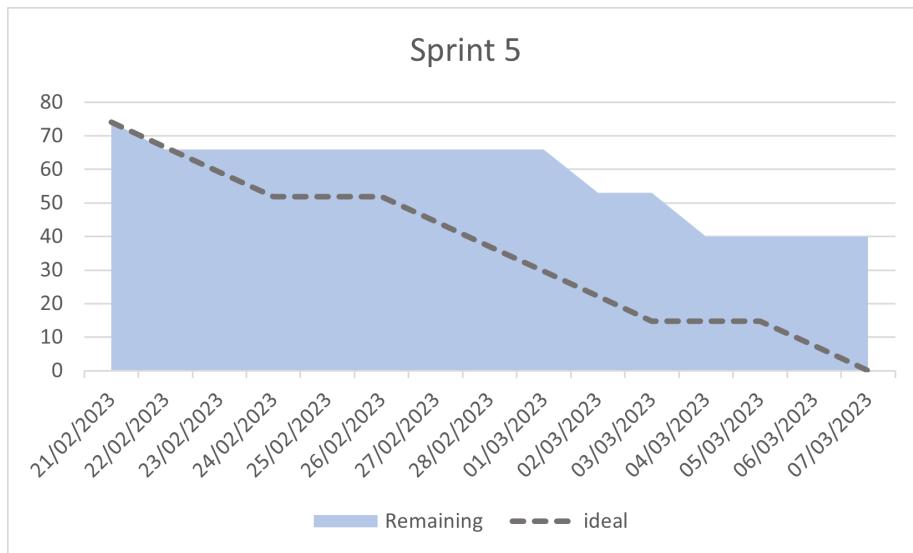


Figura A.5: Sprint 5

Sin contar que se podría haber considerado como la tarea terminada, el gráfico burndown, sería A.6.

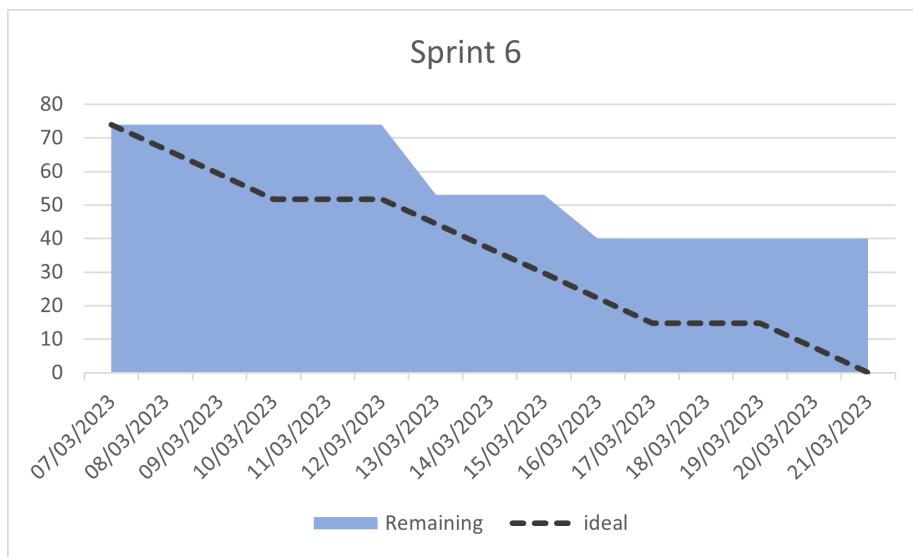


Figura A.6: Sprint 6

Sprint 7 - 21/03/2023 - 04/04/2023

En este sprint, se terminó la base de datos añadiendo otro par de pacientes y médicos, y se planearon hacer una eliminación de bugs, debido al modo oscuro, y que no guardaba bien datos entre las distintas actividades. Y también se pensó hacer el apartado “añadir técnicas y herramientas”, pero se avanzó, sin poder acabar la tarea. Quedando para el siguiente sprint. [A.7](#)

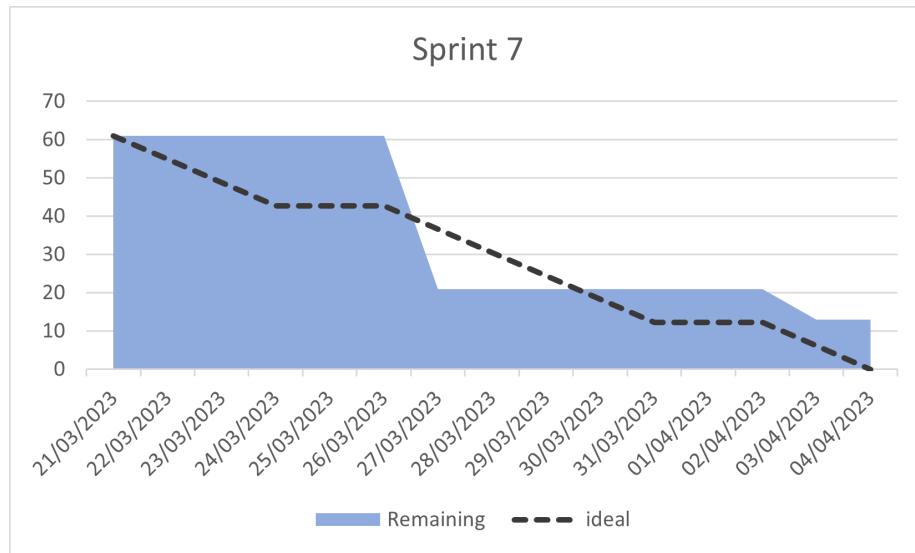


Figura A.7: Sprint 7

Sprint 8 - 04/04/2023 - 18/04/2023

En el octavo sprint, planeó implementar la red neuronal convolucional VGG-16 para ir probando la implementación de Tensorflow Lite en Android Studio, además se programó una tarea para comentar y refactorizar el código, esta tarea se planificó como si tuviera mucha dificultad, puesto que se era bastante código, pero finalmente, no fue tan laboriosa. Estas tareas se implementaron correctamente pudiendo terminar el sprint antes de tiempo. [A.8](#)

Sprint 9 - 18/04/2023 - 02/05/2023

El noveno sprint se complicó el trabajo, puesto que se tenían que realizar varios exámenes y trabajos, teniendo así, un sprint con menos trabajo. Por lo que se planeó cambiar la base de datos a una consistente, de forma que los

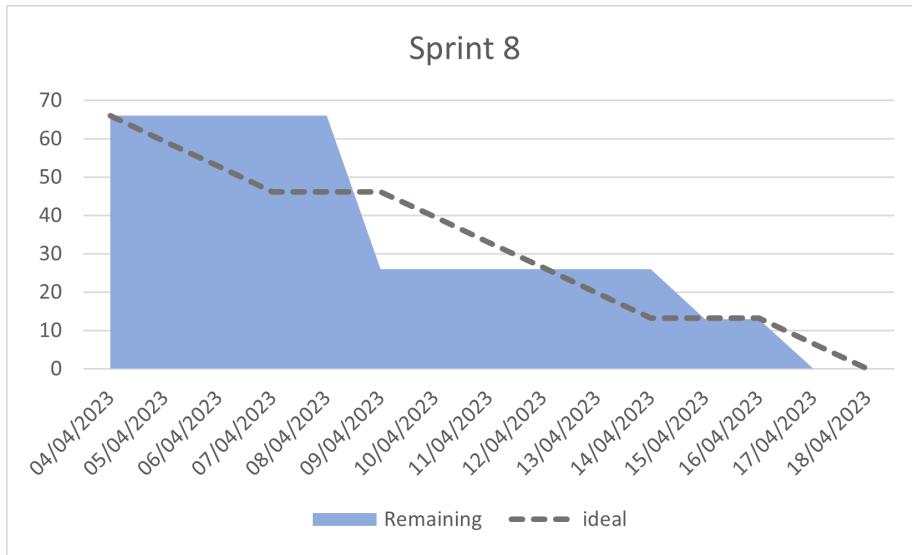


Figura A.8: Sprint 8

datos fueran persistentes entre sesiones, y seguir avanzando en el documento LaTeX, en concreto la introducción del plan de proyecto.

Considerando lo anterior, en el *burndown* A.9, se puede apreciar como la primera semana no se avanzó mucho. Aun así, se consiguió terminar apuradamente las tareas del sprint.

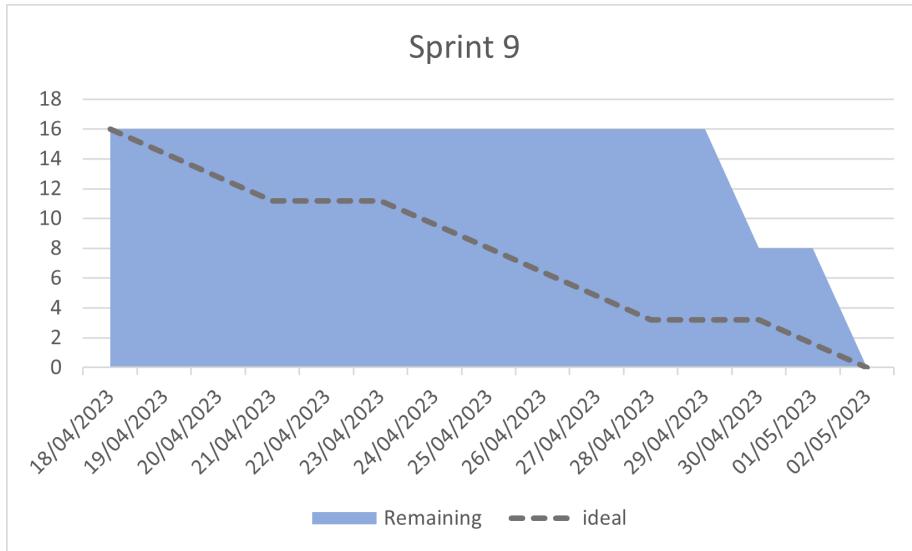


Figura A.9: Sprint 9

Sprint 10 - 02/05/2023 - 16/05/2023

Al inicio de este sprint, se decidió no hacer desarrollo continuo con respecto al documento LaTeX, puesto que era complicado determinar cuándo hacer los commits, haciendo un commit al final del sprint con todo el cambio realizado. Además, se volvió a planear una eliminación de bugs y comentar el código, puesto que se habían ido haciendo pequeños cambios.

Al terminar el sprint se terminó el apartado “Plan de proyecto” aunque este se tendría que completar más adelante. [A.10](#)

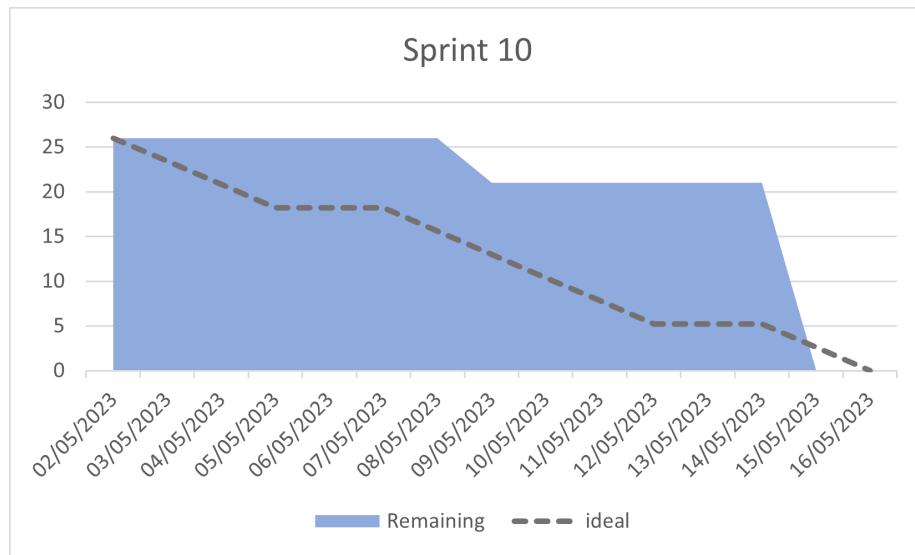


Figura A.10: Sprint 10

Sprint 11 - 16/05/2023 - 30/05/2023

En este sprint, se obtuvo la primera red neuronal, planeando la tarea para este sprint. Además, se proporcionaron las imágenes, cada una con su respectiva calidad, pudiendo hacer una red neuronal convolucional con la que predecir si una foto tenía una calidad aceptable o no. Y como tarea continua, se siguió realizando el documento LaTeX. [A.11](#)

Sprint 12 - 30/05/2023 - 13/06/2023

Al iniciar el sprint, se descubrió que el conjunto de datos tenía desbalanceo, y, por tanto, la “accuracy” no era una métrica correcta para su uso.

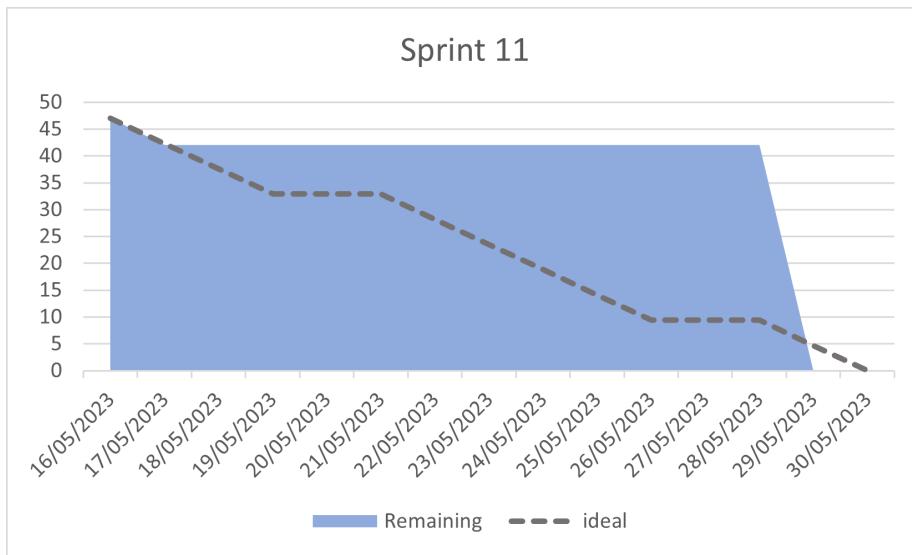


Figura A.11: Sprint 11

Por ese motivo, se decidió planear otra vez la tarea, cambiando las métricas. Y se siguió con el fichero LaTeX. [A.12](#)

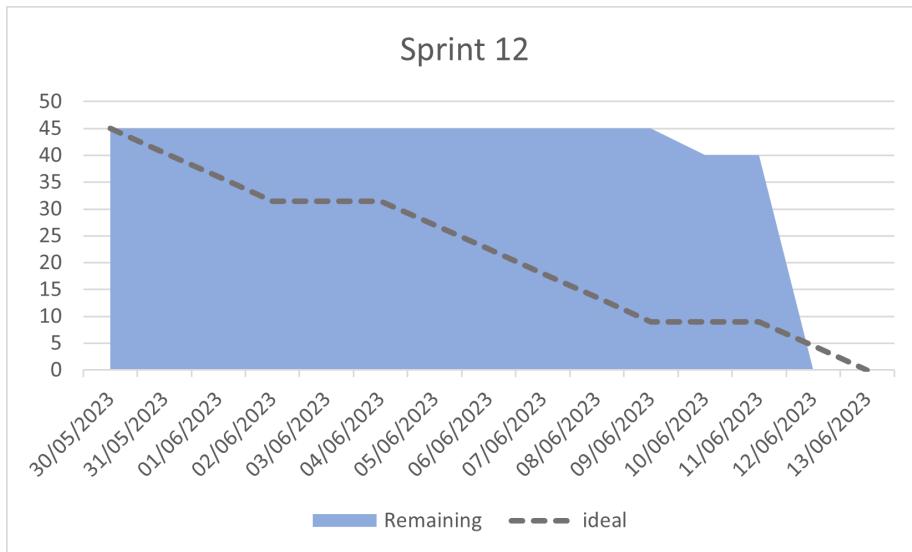


Figura A.12: Sprint 12

Sprint 13 - 13/06/2023 - 27/06/2023

En este sprint se terminó el documento LaTeX, y haciendo los retoques finales al proyecto para dejarlo entregable. [A.13](#)

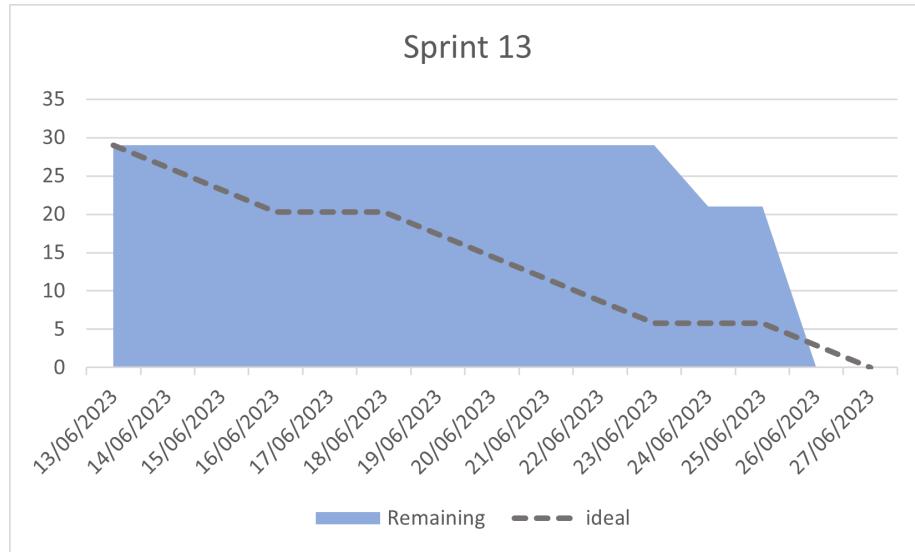


Figura A.13: Sprint 13

Sprint 14 - 27/06/2023 - 11/07/2023

En este último sprint, se completó el documento LaTeX modificando partes para una mejor comprensión del texto. [A.14](#)

A.3. Estudio de viabilidad

Este proyecto, al estar orientado a la sanidad pública, no busca obtener beneficios, sino a facilitar la labor del médico.

En caso de querer desarrollar este proyecto con una intención comercial se muestra a continuación el estudio de viabilidad.

Viabilidad económica

Costes

En personal

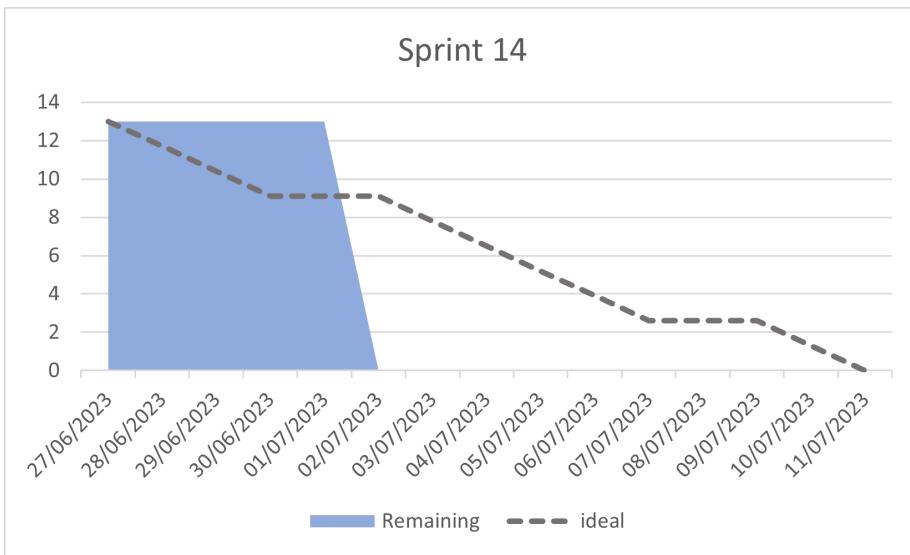


Figura A.14: Sprint 14

Concepto	Coste
Salario mensual neto	973€
Tipo de retención del IRPF	11.69 %
Cuota a pagar del IRPF	195€
Seguridad Social (29.9 %)	498€
Salario mensual bruto	1.666€
Total 6 meses	9.996€

Tabla A.1: Costes de personal

Para el desarrollo del proyecto, se ha habría contratado a un programador *junior* a tiempo parcial durante siete meses. Para la cotización de la seguridad social, se consideran los siguientes porcentajes: un 23,60 % debido a las contingencias comunes, debido al desempleo de tipo general, al ser un contrato de tiempo indefinido, es un 5,50 %; un 0,2 % para el Fondo de Garantía Salarial, y un 0,60 % para formación profesional. Sumando estos porcentajes, un 29,9 % de retribución a la seguridad social.

El coste se puede ver en la tabla: [A.1](#)

En hardware

Concepto	Coste	Amortización
Ordenador	1.200€	100€
Dispositivo móvil	300 €	25€
Oftalmoscopio Portátil D-Eye	400€	33,33€
Total	1.900€	158,33€

Tabla A.2: Costes de Hardware

Concepto	Coste	Amortización
Windows 11 Pro	259€	129,5€
ZenHub para equipos (3 meses)	36€	18 €
Total	295€	147,5€

Tabla A.3: Costes de Software

Para realizar el proyecto, se hará uso de un ordenador portátil y opcionalmente, un dispositivo móvil, para comprobar el correcto funcionamiento de la aplicación.

Suponiendo que los dispositivos hardware se amortizan en seis años, y se han utilizado durante siete meses:

El coste se puede ver en la tabla: [A.2](#)

En *software*

Se muestran aquellas licencias de *software* que no son gratuitas, y considerando que su amortización es de un año.

El coste se puede ver en la tabla: [A.3](#)

Costes adicionales

Como se pretende comercializar la aplicación, la *Play Store* es una de las mejores plataformas de distribución de aplicaciones móviles. Pero, para publicar una aplicación, es necesario una cuenta de desarrollador, la cual cuesta 25€.

El coste se puede ver en la tabla: [A.4](#)

Costes totales

Siendo los costes totales, los vistos en: [A.5](#)

Concepto	Coste
Cuenta de desarrollador Play Store	25€
Internet	210€
Alquiler de oficina	800€
Total	1.035€

Tabla A.4: Costes adicionales

Concepto	Coste
En personal	9.996€
En hardware	1.900€
En software	295€
Adicionales	1.035€
Total	13.226€

Tabla A.5: Costes totales

Beneficios

Como se ha comentado anteriormente, el proyecto tiene un objetivo altruista de ayudar en la labor del médico, dejando a un lado el objetivo de lucrarse con el proyecto. Hay varios focos en los que se podrían obtener beneficios:

- Vender la aplicación al servicio sanitario, con un beneficio del 30 %. Dando la opción de comprar servicios de mantenimiento.
- Ofrecer licencias de la aplicación, cobrando cada año por mantener la suscripción.
- Introducir anuncios en la aplicación.
- Distintas licencias de la aplicación, cada una con unas ventajas, con diferentes precios.

Entre las opciones anteriores, se ha descartado el uso de anuncios, principalmente, porque elimina profesionalidad a la aplicación; y también la opción de las distintas licencias, pero al ofrecer un único servicio, no se podía plantear esta opción.

Como las otras dos opciones se han considerado viables, se desarrollarán a continuación, además, es importante recalcar que los valores pueden cambiar debido a la gran cantidad de centro sanitarios públicos que hay, proporcionando ofertas según esa cantidad.

Vender la aplicación

Vendiendo la aplicación al servicio sanitario, solo se realizaría un pago por la compra, de forma que la empresa obtenga un 30 % de beneficios. De esta forma, 13.226€ sería el 70 % del precio y la aplicación se vendería por 18.894,28€; dando un beneficio de 5.668,28 €.

También se podría ofrecer al servicio sanitario una opción de compra, donde la empresa se encarga de hacer el mantenimiento y actualizaciones de la aplicación. Siendo un servicio que depende del numero de centros sanitarios que haya en la comunidad autónoma. Un posible precio razonable, sería que, por cada centro sanitario, anualmente se pague una tarifa de 10€.

Poniendo el caso de que el SACYL comprase la aplicación para el 25 % de los centros sanitarios, como tiene 10.218, debería pagar por $(10.218 * 0,25 * 10)$, siendo de esta forma anualmente se alcanzaría una cifra de 25.545€al año. Como una persona cuesta cada 6 meses, 9.996€, al año serian 19.992€, teniendo beneficios anuales.

Licencias de la aplicación

Como el caso anterior, se va a hacer una oferta por cada centro sanitario, donde anualmente, se cobraría 100€.

Suponiendo que el SACYL pidiese una licencia para todos los centros, se haría el cálculo de $(10,218 * 0,25 * 100)$ dando una cifra anual de 255.450€en este precio ya estaría incluido el precio del mantenimiento.

En este caso, la empresa se encargaría de ofrecer los servicios al SACYL, se debería contratar un servidor con una base de datos, como el servidor guarda imágenes, se necesitaría bastante espacio, pongamos que se contrata un servidor Microsoft SQL Server de 1 GB, con un precio de 350€al mes. Como 1GB pueden ser poco para Castilla y León, se considera que se compran 50 servidores. Añadiendo al coste anual, $4200 * 50$ siendo así, 210.000€

Por tanto, con 255.450 €al año; habría que restar el coste de los servidores y el del personal donde se podría aumentar el número a 2 personas, $255.450 - 210.000 - 39.984$, haciendo que anualmente, se tengan 5.466 €de beneficio, como el coste de la aplicación era de 13.226 €, harían falta 3 años para obtener un beneficio real sobre el proyecto.

Otros casos

Hay un caso que es importante destacar, y es el caso concreto de que el SACYL fuese la empresa encargada.

En este caso, no hay ningún beneficio monetario para la empresa, puesto que es sanidad pública. Pero, realizando el proyecto, a futuro se pueden evitar costes adicionales, como, por ejemplo:

- Coste de transporte de la ambulancia.
- Costes de citas innecesarias y el tiempo que conllevan estas.
- Costes de material utilizado.

Viabilidad legal

Software

Esta sección aborda el contexto legal del proyecto, junto con las licencias *software* utilizadas.

Dependencia	Versión	Descripción	Licencia
Appcompat	1.6.1	Permite a versiones anteriores de Android una compatibilidad mejorada	Apache License 2.0
Material	1.8.0	Proporciona componentes de Material Design para Android.	Apache License 2.0
Constraint Layout	2.1.4	Permite crear diseños complejos y flexibles	Apache License 2.0
JUnit	4.13.2	Framework de pruebas unitarias en Java	Eclipse Public License 1.0
Ext JUnit	1.1.5	Biblioteca de pruebas unitarias para Android que extiende JUnit	Apache License 2.0

continúa en la página siguiente

continúa desde la página anterior			
Dependencia	Versión	Descripción	Licencia
Espresso-core	3.5.1	Biblioteca de pruebas de interfaz de usuario	Apache License 2.0
TensorFlow Lite	2.7.0	Biblioteca de aprendizaje automático, más ligera que TensorFlow	Apache License 2.0
SQLite	2.2.0	Biblioteca que utiliza bases de datos SQLite, proporcionando una API para interactuar con la base de datos	No requiere

Tabla A.6: Dependencias del proyecto Android

Como el proyecto también ha tenido una parte realizada en Python, se incluyen las dependencias utilizadas.

Dependencia	Versión	Descripción	Licencia
sys	3.10.8	Módulo de la librería estándar de Python que permite trabajar con tiempos	Python Software Foundation
NumPy	1.23.5	Permite manejar <i>arrays</i> , álgebra lineal, transformaciones de fourier y matrices	BSD 3-Clause
Pandas	2.0.1	Permite analizar, limpiar, explorar y manipular el conjuntos de datos.	BSD 3-Clause
matplotlib	3.6.2	Permite ver visualmente gráficas de nivel bajo	Python Software Foundation
scikit-learn	1.2.2	Es un modulo para aprendizaje automático.	BSD 3-Clause
TensorFlow	2.14.0	Biblioteca de aprendizaje automático	Apache License 2.0

Tabla A.7: Dependencias del proyecto Python

Imágenes

Hay varios tipos de imágenes utilizadas en el proyecto, donde el logo, se ha utilizado usando DALLE2, la cual en sus políticas y términos se explica que el usuario tiene todos los derechos, tanto comerciales y no comerciales.

Por otro lado, para los iconos de la aplicación, se usaron tanto iconos por defecto de Android Studio, como iconos de Google Fonts, donde todas las imágenes son de código abierto y no tienen costo. Con una licencia Apache 2.0.

Licencia del proyecto

En resumen, el proyecto utiliza varios tipos de licencias que son por orden de más permisivas a menos: BSD 3-Clause License, Python Software Foundation License, Apache License 2.0 y Eclipse Public License 1.0.

Una licencia que se adapta a las condiciones buscadas es GNU General Public License v3. En la imagen A.15 se puede observar correctamente las características que ofrece.

Permissions	Conditions	Limitations
● Commercial use	● Disclose source	● Liability
● Distribution	● License and copyright notice	● Warranty
● Modification		
● Patent use	● Same license	
● Private use	● State changes	

Figura A.15: Licencia GPLv3. Imagen obtenida de <https://choosealicense.com/licenses/>

Al comprobar la compatibilidad de las licencias con la GNU, se comprobó que Eclipse Public License 1.0 no es compatible con esta. En este caso, se opto por utilizar una doble licencia EPL y Apache License 2.0; donde EPL afectaría solo a los test unitarios y la demás parte del proyecto sería liberada con Apache License 2.0.

Marco legal del proyecto

Es importante recalcar que al estar trabajando sobre una aplicación que usa datos delicados de las personas, siendo estos, además, datos relacionados con la salud.

RGPD

Según viene descrito en el artículo 9 del Reglamento General de Protección de Datos, está prohibido hacer un tratamiento de categorías especiales de datos personales a menos que ocurra:

- El interesado ha dado el consentimiento explícito para el tratamiento de datos.
- Resulta necesario para proteger los intereses vitales del interesado u otra persona física cuando no esté capacitado para dar su consentimiento.
- Es necesario para fines de medicina preventiva o laboral.
- Es necesario con fines de archivo en interés público, fines de investigación científica o histórica o fines estadísticos.

En el **artículo 9**, vienen más puntos, pero solo se explican los referentes al uso de la aplicación.

En el **capítulo 4** del Reglamento General de Protección de Datos, se trata sobre el responsable de tratar los datos y el encargado del tratamiento.

Por ello, con el fin de evitar que los datos sensibles se confundan entre los distintos pacientes, se ha implementado una interfaz, en la cual el médico introduce el DNI del paciente a ser tratado en ese momento, mostrando el nombre del paciente del DNI, en caso de haberse confundido a la hora de poner el DNI, se podrá dar cuenta al ver el nombre del paciente.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En este apartado, se recogen los requisitos del proyecto, ya sean requisitos funcionales o no funcionales.

Como el proyecto se pueden dividir en dos proyectos, la aplicación Android y la creación de una red neuronal convolucional, se separarán estos entre sí, para una mayor visibilidad.

B.2. Objetivos generales

Los objetivos del proyecto Android buscan:

- Proporcionar a los centros sanitarios una aplicación, que permita diferenciar cuándo un paciente posee o no retinopatía diabética.
- Guardar las respectivas imágenes en los pacientes indicados, con el resultado proporcionado.
- Mostrar los resultados para que el médico pueda proporcionar un primer diagnóstico.

Los objetivos del proyecto de la red neuronal:

- Desarrollar un modelo que diferencie entre fotos con calidad buena y fotos con calidad mala.

- Desarrollar un modelo que clasifique correctamente el mayor número de instancias.

B.3. Catálogo de requisitos

Los requisitos funcionales para la aplicación Android son:

- **RF-1: Gestión de usuarios:** Se desea que la aplicación permita gestionar los usuarios.
 - **RF-1.1: Gestión de médicos**
 - **RF-1.1.1: Mostrar datos médico:** Se desea poder ver los datos de los médicos.
 - **RF-1.2: Gestión de pacientes**
 - **RF-1.2.1: Mostrar datos paciente:** Se desea poder ver los datos de los pacientes.
- **RF-2: Gestión de informes:** Se desea que la aplicación permita gestionar los informes.
 - **RF-2.1: Creación de informes:** Se desea que los médicos puedan crear un informe a partir de una foto y un paciente.
 - **RF-2.1.1: Obtención de fecha:** La aplicación debe tomar la fecha en la que se crea el informe.
 - **RF-2.1.2: Obtención del ojo:** La aplicación debe tomar el ojo al que se va a realizar la foto.
 - **RF-2.1.3: Obtención de la imagen:** La aplicación debe mostrar la opción de cómo realizar la imagen.
 - ◊ **RF-2.1.3.1: Tomar fotos:** Se desea que la aplicación permita tomar fotos de la cámara del usuario.
 - ◊ **RF-2.1.3.2: Escoger fotos:** Se desea que la aplicación permita escoger una foto desde la galería del paciente.
 - ◊ **RF-2.1.3.3: Calidad de la foto:** Se desea implementar un sistema que permita comprobar que la foto tomada es de calidad. En caso contrario, mostrar al usuario que la imagen no tiene una calidad aceptable, y que tome otra foto.
 - **RF-2.1.4: Obtención del resultado:** Se desea que la aplicación de un resultado basado en una red neuronal.

- ◊ **RF-2.1.4.1: Escoger red neuronal:** Se desea que el usuario pueda elegir con que red neuronal analizar la imagen con la que obtener los resultados.
- **RF-2.2: Mostrar informes:** Se desea poder ver los informes según el paciente.
- **RF-3: Iniciar sesión:** Se desea que la aplicación permita iniciar sesión.
 - **RF-3.1: Iniciar sesión como médico:** Los médicos pueden iniciar sesión con su email y su contraseña.
 - **RF-3.2: Iniciar sesión como invitado:** Los médicos pueden iniciar sesión sin proporcionar email ni contraseña.
- **RF-4: Escoger un paciente:** Se desea que el médico identificado introduzca el DNI de un paciente para gestionar sus informes o datos.

Requisitos no funcionales:

- **RFNF-1: Base de datos:** Se debe almacenar los datos introducidos.
 - **RFNF-1.1: Datos de los médicos:** Se desea almacenar la contraseña y el centro médico donde trabaja el médico.
 - **RFNF-1.2: Datos de los pacientes:** Se desea almacenar una lista de informes, información sobre el paciente y el estado en el que se encuentra.
 - **RFNF-1.3: Datos de los usuarios:** Se desea almacenar nombre, apellidos, fecha de nacimiento, DNI y el correo electrónico de los usuarios.
 - **RFNF-1.4: Datos de los informes:** Se desea almacenar la foto del informe, la fecha en la que se obtuvo, el ojo al que hace referencia y el resultado de dicho informe.
- **RFNF-2: Concurrencia:** El usuario debe poder utilizar la aplicación mientras se comprueban las imágenes.
- **RFNF-3: Aplicación:** La aplicación debe funcionar correctamente, no dando errores al usuario y en caso de algún error, controlarlo.
- **RFNF-4: Interfaz:** La interfaz debe ser sencilla, para que usuarios nuevos puedan utilizarla.

- **RFNF-5: Dispositivos:** La aplicación debe ejecutarse correctamente en el máximo número de dispositivos.

Por otro lado, no hay requisitos funcionales para el proyecto de la red neuronal, siendo los requisitos no funcionales:

- **RFNF-1: Red neuronal de calidad:** Se debe crear una red neuronal que diferencie entre una calidad APTA y una calidad NO_APTA.
- **RFNF-2: Escoger el mejor modelo:** Se debe escoger el mejor modelo obtenido.

B.4. Actores

Para la aplicación se consideran 2 actores, pero se podría considerar solo uno.

- **Actor médico:** Este actor es un médico identificado, quien ha iniciado sesión con su email y contraseña; además, proporciona el DNI del paciente para trabajar sobre el paciente que ha ido a consulta.
- **Actor invitado:** Este actor es un médico no identificado, de esta forma, puede comprobar el grado de retinopatía diabética sin crear un informe a un paciente. Este actor es el indicado, cuando se ha proporcionado una imagen a analizar y no se sabe el paciente en concreto.

B.5. Especificación de requisitos

En este apartado se explicarán los casos de uso.

Diagrama de casos de uso

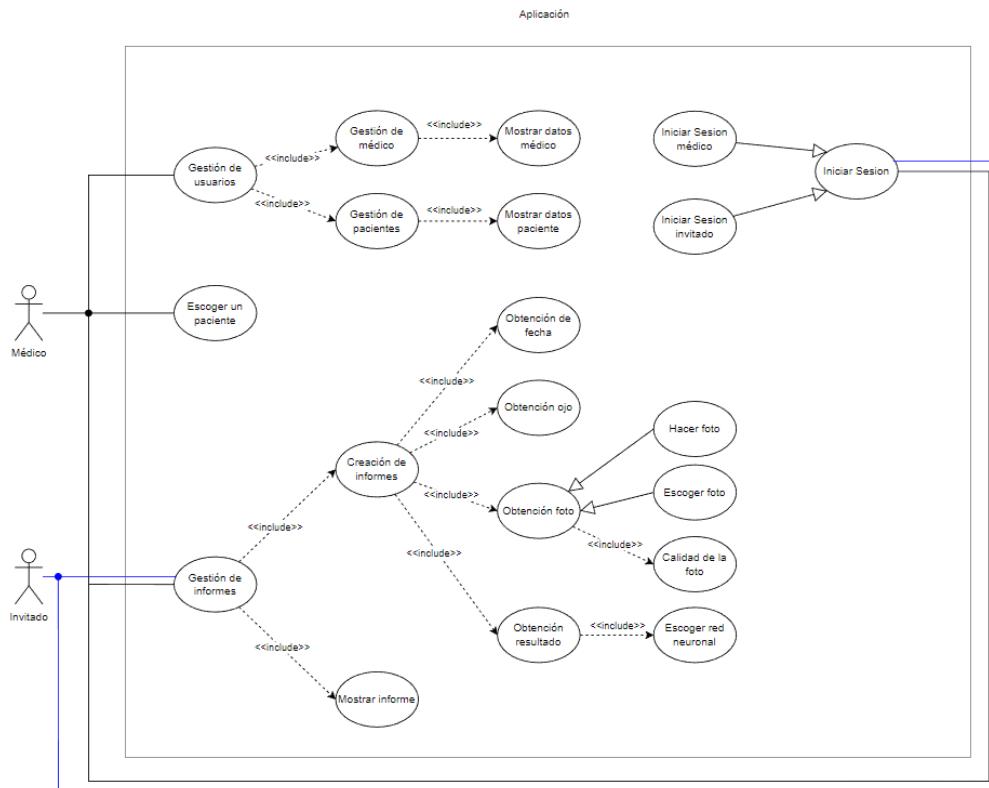


Figura B.1: Diagrama de casos de uso

Casos de uso

CU-1	Gestión de usuarios
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-1, RF-1.1, RF-1.1.1, RF-1.2, RF-1.2.1
Descripción	Se desea que la aplicación permita gestionar los usuarios.
Precondición	El médico debe estar identificado, y haya puesto un DNI de paciente valido.
Acciones	<ol style="list-style-type: none"> 1. El usuario llega a la página principal de la aplicación. 2. Se ofrece la opción de ver los datos del médico. 3. Se ofrece la opción de ver los datos del paciente.
Postcondición	
Excepciones	
Importancia	Media

Tabla B.1: CU-1 Gestión de usuarios.

CU-2	Gestión de médico
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-1.1, RF-1.1.1
Descripción	Se desea que la aplicación permita gestionar los médicos.
Precondición	El médico debe estar identificado.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el botón para ver sus datos.
Postcondición	
Excepciones	
Importancia	Baja

Tabla B.2: CU-2 Gestión de médico.

CU-3	Mostrar datos médico
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-1.1.1
Descripción	Se desea poder ver los datos de los médicos.
Precondición	El médico debe estar identificado.
Acciones	<ol style="list-style-type: none"> 1. La aplicación obtiene los datos del médico de la base de datos. 2. La aplicación muestra los datos del médico en la interfaz.
Postcondición	
Excepciones	
Importancia	Baja

Tabla B.3: CU-3 Mostrar datos médico.

CU-4	Gestión de pacientes
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-1.2, RF-1.2.1
Descripción	Se desea que la aplicación permita gestionar los pacientes.
Precondición	El médico debe estar identificado, y haya puesto un DNI de paciente valido.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona el botón para ver los datos de los pacientes.
Postcondición	
Excepciones	
Importancia	Baja

Tabla B.4: CU-4 Gestión de pacientes.

CU-5	Mostrar datos paciente
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-1.2.1
Descripción	Se desea poder ver los datos de los pacientes.
Precondición	El médico debe estar identificado, y haya puesto un DNI de paciente valido.
Acciones	<ol style="list-style-type: none"> 1. La aplicación obtiene los datos del paciente de la base de datos. 2. La aplicación muestra los datos del paciente en la interfaz.
Postcondición	
Excepciones	
Importancia	Baja

Tabla B.5: CU-5 Mostrar datos pacientes.

CU-6	Escoger un paciente
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-4
Descripción	Se desea que el médico identificado introduzca el DNI de un paciente para gestionar sus informes o datos.
Precondición	El médico debe estar identificado.
Acciones	<ol style="list-style-type: none"> 1. El médico introduce un DNI. 2. El médico pulsa el botón de comprobación de DNI. 3. La aplicación comprueba que se haya introducido un número de 8 dígitos. 4. <i>a)</i> Si se ha introducido correctamente, la aplicación sigue con su ejecución. <i>b)</i> Si no, se vuelve al paso 1, mostrando un mensaje al usuario de que introduzca un DNI sin la letra final. 5. La aplicación comprueba en la base de datos si hay un usuario con ese DNI. 6. <i>a)</i> Si existe, la aplicación muestra el nombre del usuario del DNI. <i>b)</i> Si no existe, la aplicación muestra un mensaje de que no existe paciente con ese DNI.
Postcondición	Permite al usuario pasar a la siguiente actividad.
Excepciones	
Importancia	Alta

Tabla B.6: CU-6 Escoger un paciente.

CU-7	Gestión de informes
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2, RF-2.1, RF-2.1.1, RF-2.1.2, RF-2.1.3, RF-2.1.3.1, RF-2.1.3.2, RF-2.1.3.3, RF-2.1.4, RF-2.1.4.1, RF-2.2
Descripción	Se desea que la aplicación permita gestionar los informes.
Precondición	
Acciones	<ol style="list-style-type: none"> 1. Se ofrece la opción de crear informes. 2. Se ofrece la opción de ver los informes.
Postcondición	
Excepciones	
Importancia	Alta

Tabla B.7: CU-7 Gestión de informes.

CU-8	Creación de informes
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1, RF-2.1.1, RF-2.1.2, RF-2.1.3, RF-2.1.3.1, RF-2.1.3.2, RF-2.1.3.3, RF-2.1.4, RF-2.1.4.1
Descripción	Se desea que la aplicación permita crear nuevos informes.
Precondición	Se sabe el paciente en el que añadir el informe.
Acciones	<ol style="list-style-type: none"> 1. La aplicación obtiene la fecha en la que se crea el informe. 2. La aplicación obtiene el ojo sobre el que se realiza el informe. 3. La aplicación obtiene la foto que se guardará en el informe. 4. La aplicación calcula el resultado del informe
Postcondición	Se crea un nuevo informe para el paciente y se almacena en la base de datos.
Excepciones	
Importancia	Alta

Tabla B.8: CU-8 Creación de informes.

CU-9	Obtención de fecha
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.1
Descripción	La aplicación debe tomar la fecha en la que se crea el informe.
Precondición	Se tienen los demás datos para crear el informe.
Acciones	<ul style="list-style-type: none"> 1. La aplicación obtiene la fecha en la que se crea el informe.
Postcondición	Se crea un nuevo informe para el paciente.
Excepciones	
Importancia	Baja

Tabla B.9: CU-9 Obtención de fecha.

CU-10	Obtención de ojo
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.2
Descripción	La aplicación debe tomar el ojo del informe.
Precondición	Se sabe el paciente en el que añadir el informe.
Acciones	<ul style="list-style-type: none"> 1. La aplicación muestra una imagen de ojo derecho y ojo izquierdo. 2. El usuario escoge sobre cual se realiza el informe.
Postcondición	
Excepciones	
Importancia	Baja

Tabla B.10: CU-10 Obtención de ojo.

CU-11	Obtención de foto
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.3, RF-2.1.3.1, RF-2.1.3.2, RF-2.1.3.3
Descripción	La aplicación debe mostrar la opción de como realizar la foto.
Precondición	Se sabe el paciente en el que añadir el informe y el ojo.
Acciones	<ol style="list-style-type: none"> 1. La aplicación ofrece un botón para realizar una foto. 2. La aplicación ofrece un botón para escoger una foto de la galería. 3. El usuario pulsa una de las opciones, seleccionando una foto. 4. La aplicación muestra al usuario la imagen obtenida. 5. La aplicación comprueba la calidad de la imagen. 6. El usuario puede volver al paso 3, o pasar a la siguiente pantalla.
Postcondición	Se ha guardado la foto del informe.
Excepciones	
Importancia	Alta

Tabla B.11: CU-11 Obtención de foto.

CU-12	Hacer foto
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.3.1
Descripción	Se desea que la aplicación permita tomar fotos de la cámara del usuario.
Precondición	Se sabe el paciente en el que añadir el informe y el ojo.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa el botón de hacer foto. 2. La aplicación abre la cámara del dispositivo. 3. El usuario realiza la foto. 4. La aplicación guarda la imagen.
Postcondición	Se ha guardado la foto internamente.
Excepciones	
Importancia	Alta

Tabla B.12: CU-12 Hacer foto.

CU-13	Escoger foto
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.3.2
Descripción	Se desea que la aplicación permita escoger una foto desde la galería del paciente.
Precondición	Se sabe el paciente en el que añadir el informe y el ojo.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa el botón de escoger foto. 2. La aplicación abre la galería del dispositivo. 3. El usuario selecciona la foto. 4. La aplicación guarda la imagen seleccionada.
Postcondición	Se ha guardado la foto internamente.
Excepciones	
Importancia	Alta

Tabla B.13: CU-13 Escoger foto.

CU-14	Calidad foto
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.3.3
Descripción	Se desea implementar un sistema que permita comprobar que la foto tomada es de calidad.
Precondición	Se sabe el paciente en el que añadir el informe y el ojo. Y se ha mostrado la foto al usuario.
Acciones	<ol style="list-style-type: none"> 1. La aplicación obtiene la imagen mostrada al usuario. 2. La aplicación trabaja en segundo plano. 3. La aplicación realiza un preprocesado a la imagen. 4. La aplicación introduce a la red neuronal de calidad la imagen. 5. La aplicación obtiene el resultado. 6. <i>a)</i> Si es una calidad apta, se permite al usuario avanzar a la siguiente actividad. <i>b)</i> Si es una calidad no apta, se muestra un mensaje de omitir comprobación de calidad, que si pulsa, permite al usuario avanzar a la siguiente actividad.
Postcondición	Se ha comprobado la calidad de la foto.
Excepciones	
Importancia	Alta

Tabla B.14: CU-14 Calidad foto.

CU-15	Obtención del resultado
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.4
Descripción	Se desea que la aplicación de un resultado basado en una red neuronal.
Precondición	Se sabe el paciente en el que añadir el informe y el ojo. Y se ha guardado la foto.
Acciones	<ol style="list-style-type: none"> 1. La aplicación obtiene la imagen mostrada al usuario. 2. El usuario selecciona las redes neuronales con las que analizar la imagen. 3. La aplicación obtiene los resultados y calcula la moda y en caso de empate, la media.
Postcondición	Se crea el informe con todos los atributos.
Excepciones	
Importancia	Alta

Tabla B.15: CU-15 Obtención del resultado.

CU-16	Escoger red neuronal
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.1.4.1
Descripción	Se desea que el usuario pueda elegir con que red neuronal analizar la imagen con la que obtener los resultados.
Precondición	Se tiene la imagen para el informe.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona las redes neuronales con las que analizar la imagen. 2. La aplicación preprocesa la imagen para las redes neuronales seleccionadas. 3. La aplicación introduce las imágenes preprocesadas en sus respectivos modelos.
Postcondición	La aplicación obtiene los resultados y calcula la moda y en caso de empate, la media.
Excepciones	
Importancia	Alta

Tabla B.16: CU-16 Escoger red neuronal.

CU-17	Mostrar informe
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-2.2
Descripción	Se desea poder ver los informes según el paciente.
Precondición	Se sabe el paciente.
Acciones	<ol style="list-style-type: none"> 1. El usuario selecciona la opción para ver los informes del paciente. 2. La aplicación muestra un listado de los informes del paciente ordenados de más nuevos a más viejos. 3. La aplicación muestra los informes de 3 en 3 permitiendo avanzar y retroceder entre el listado.
Postcondición	
Excepciones	
Importancia	Media

Tabla B.17: CU-17 Mostrar informe.

CU-18	Iniciar sesión
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-3, RF-3.1, RF-3.2
Descripción	Se desea que la aplicación permita iniciar sesión.
Precondición	El usuario debe estar registrado en la base de datos.
Acciones	<ol style="list-style-type: none">1. La aplicación ofrece iniciar sesión por email y contraseña.2. La aplicación ofrece iniciar sesión como invitado.
Postcondición	
Excepciones	
Importancia	Media

Tabla B.18: CU-18 Iniciar sesión.

CU-19	Iniciar sesión como médico
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-3.1
Descripción	Los médicos pueden iniciar sesión con su email y su contraseña.
Precondición	El usuario debe estar registrado en la base de datos.
Acciones	<ol style="list-style-type: none"> 1. El usuario introduce el email y la contraseña. 2. El usuario pulsa el botón para iniciar sesión. 3. <i>a)</i> La aplicación comprueba si no se han introducido campos vacíos. <i>b)</i> La aplicación comprueba que el usuario existe. <i>c)</i> La aplicación comprueba si la contraseña de ese usuario es la introducida. <ol style="list-style-type: none"> 4. <i>a)</i> Si cumple las 3 condiciones anteriores, el usuario inicia sesión. <i>b)</i> Si no, se muestra al usuario un mensaje de que el email o contraseña son incorrectos.
Postcondición	
Excepciones	
Importancia	Media

Tabla B.19: CU-19 Iniciar sesión como médico.

CU-20	Iniciar sesión como invitado
Versión	1.0
Autor	Miguel Fuente García
Requisitos asociados	RF-3.1
Descripción	Los médicos pueden iniciar sesión con su email y su contraseña.
Precondición	El usuario debe estar registrado en la base de datos.
Acciones	<ol style="list-style-type: none"> 1. El usuario pulsa el botón para iniciar sesión como invitado. 2. La aplicación inicia sesión como médico “invitado” y paciente “invitado”.
Postcondición	
Excepciones	
Importancia	Media

Tabla B.20: CU-20 Iniciar sesión como invitado.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado se define como se ha diseñado el programa para los requisitos anteriores.

Como en el proyecto del modelo neuronal no se tiene un diseño como tal, en este apartado solo se expondrá sobre el proyecto Android.

C.2. Diseño de datos

En la aplicación se usan las siguientes entidades:

- Usuario: contiene los datos del usuario, almacenando el nombre, los apellidos, el correo electrónico, el DNI y la fecha de nacimiento.
- Médico: son datos adicionales al usuario, que permite almacenar el centro médico asociado, la contraseña que tienen para iniciar sesión, y la preferencia de la ubicación del ojo.
- Paciente: son datos adicionales al usuario, que permite almacenar información adicional del paciente y el estado diagnosticado por el último informe realizado.
- Informe: contiene los datos relativos a los informes generados por la aplicación, siendo la imagen de la retina obtenida, la fecha en la que se realizó el informe, el ojo del que se tomó la foto y el resultado proporcionado por la aplicación.

De esta forma, el diagrama entidad relación de la aplicación quedaría como la figura C.1.

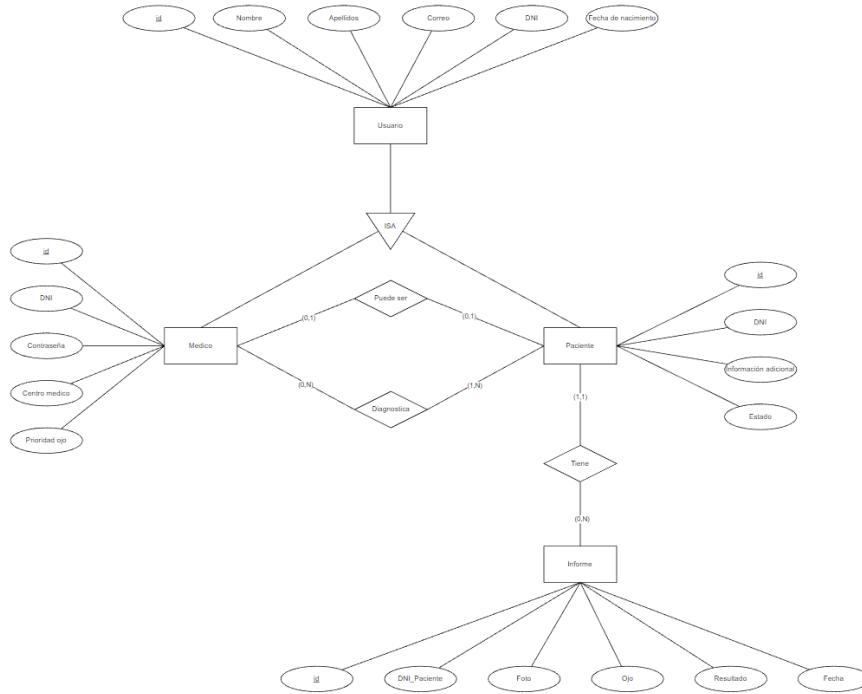


Figura C.1: Diagrama entidad relación.

C.3. Diseño procedimental

Como la interacción de un usuario con una aplicación es demasiado larga, se van a representar aquellas que se consideran más importantes.

Los diagramas de secuencia quedarían de esta forma:

Se ha intentado realizar de la mejor forma posible puesto que en las aplicaciones siempre afecta el factor humano; y también hay que considerar que se usan hilos para la ejecución de la aplicación por lo que no se puede seguir una secuencia segura.

En C.2, se puede obtener una foto ya sea por sacar la foto o escogerla de la galería del usuario, hasta que se pase a la siguiente actividad. Cada vez que se obtiene una foto, la aplicación comprueba su calidad, habilitando un botón u otro para pasar a la siguiente actividad.

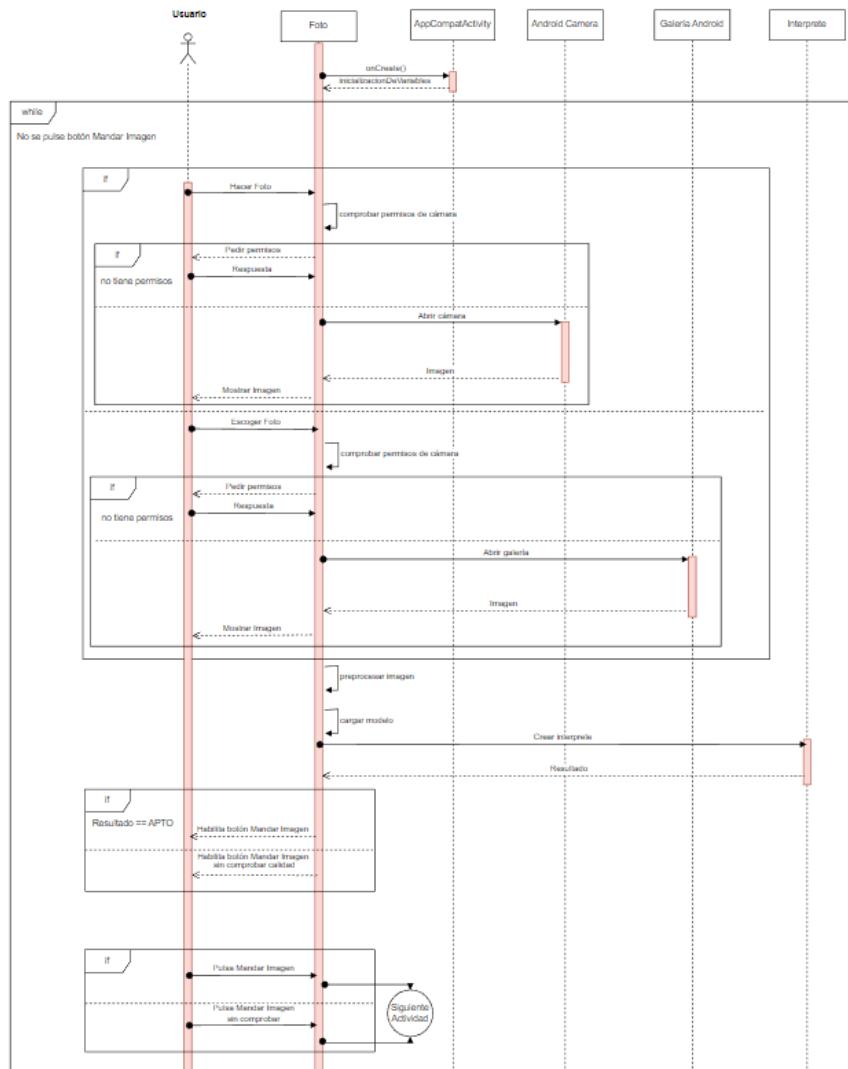


Figura C.2: Diagrama de secuencia representando a la clase Foto.

En C.3, en este caso, el usuario selecciona las redes neuronales que quiera, y cuando pulsa el botón para crear, se crean tantos hilos como modelos seleccionados, y en el resultado, se calcula la moda, y en caso de empate, la media de ellos. Este resultado, es el que se considera en el expediente, añadiendo los valores al expediente creado y al paciente.

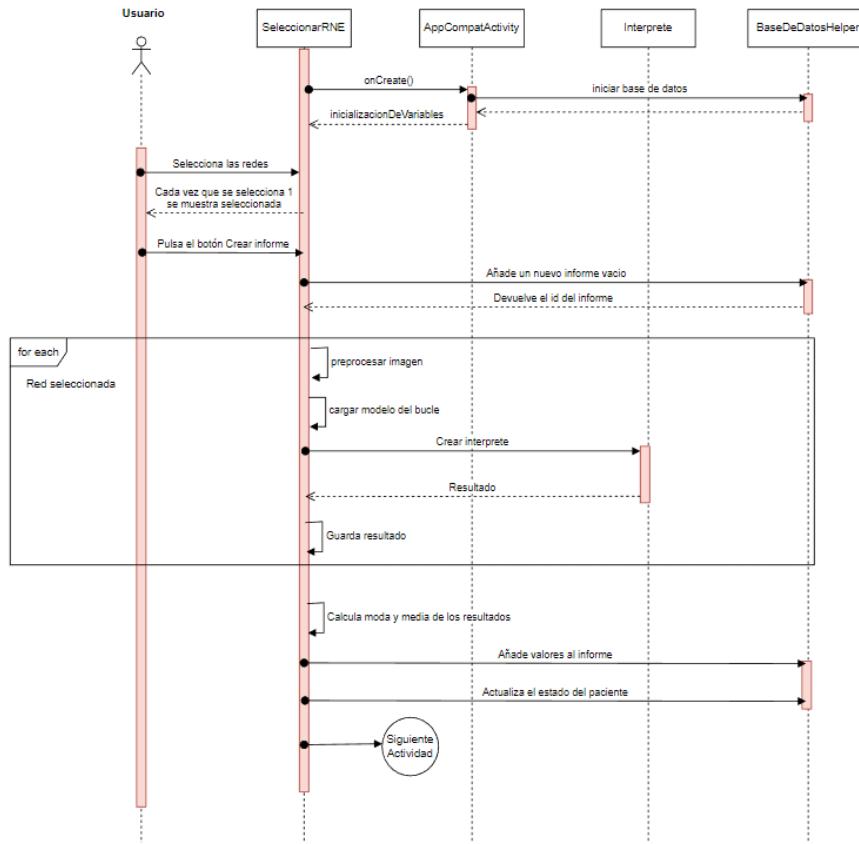


Figura C.3: Diagrama de secuencia representando a la clase SeleccionarRNE.

C.4. Diseño arquitectónico

Como ya se ha explicado en el apartado de “Técnicas y herramientas”, se ha utilizado un patrón Modelo-Vista-Presentador y un patrón *Singleton*.

Modelo-Vista-Presentador

Es una derivación del patrón modelo-vista-controlador, la diferencia es que se cambia el controlador por un presentador, el cual sirve como intermediario entre el modelo y la interfaz.

- El modelo, define los datos que se utilizan en la interfaz.
- El presentador, funciona como intermediario, recuperando los datos del modelo, y cambiando la vista de la interfaz.
- La vista, es la interfaz de usuario.

Puesto que Android Studio, ya proporciona separaciones para hacer uso de un modelo-vista-presentador, su implementación ha sido sencilla, donde las *activities* son las diferentes interfaces, los archivos java del directorio “com.example.retinopatia” son los presentadores de las vistas, y en el directorio DataBase se encuentra el modelo de datos.

En la figura C.4, se puede observar su funcionamiento.

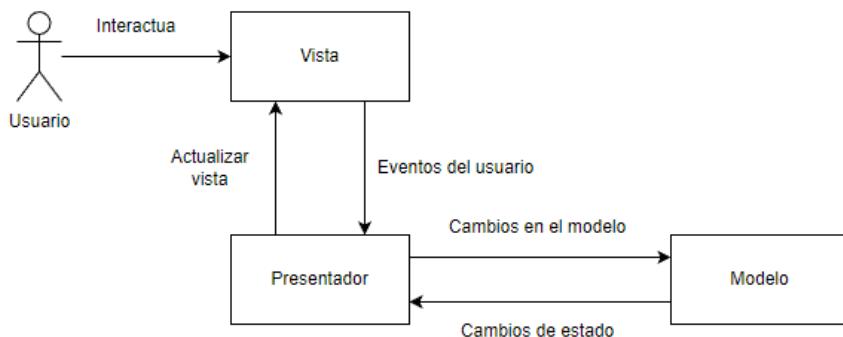


Figura C.4: Modelo-Vista-Presentador

Patrón singleton

Este patrón sirve para restringir la creación de objetos. En el caso utilizado, ha sido para la base de datos, de esta forma, no se creaba de cero cada vez que se iniciaba la aplicación.

Diseño de clases

Las clases han sido diseñadas todas en el mismo paquete a excepción de la base de datos, la cual debería ser externa a la aplicación y por ello, se ha considerado en otro paquete.

Cada clase se corresponde con cada una de las vistas del modelo-vista-presentador, estas vistas son las Actividades que se encuentran en el directorio “res.layout”; a excepción de la clase Utils, la cual es una clase que permite utilidades como abrir un fichero y un switch con los resultados de las imágenes, considerandolo como un convertidor de numero a tipo de retinopatia.

A continuación se muestran los diagramas de clases resumidos para una mayor comprensión de ellos.

En el primer diagrama de clases, C.5, se puede observar las clases Utils, Foto, Resultados, SeleccionarRNE e Interpreter. Como estas incluyen muchas variables y métodos, no se han insertado, exceptuando Utils, que solo incluye 2 métodos estaticos.

En la figura se puede observar como la clase Foto y SeleccionarRNE hacen uso de la clase Interprete, esto es así, puesto que estas clases son las que integran los modelos. Por otro lado, tanto Foto, como Resultados, como SeleccionarRNE hacen uso de Utils. Foto y SeleccionarRNE hacen uso del método loadFile y SeleccionarRNE y Resultados hacen uso del método SwitchResultados.

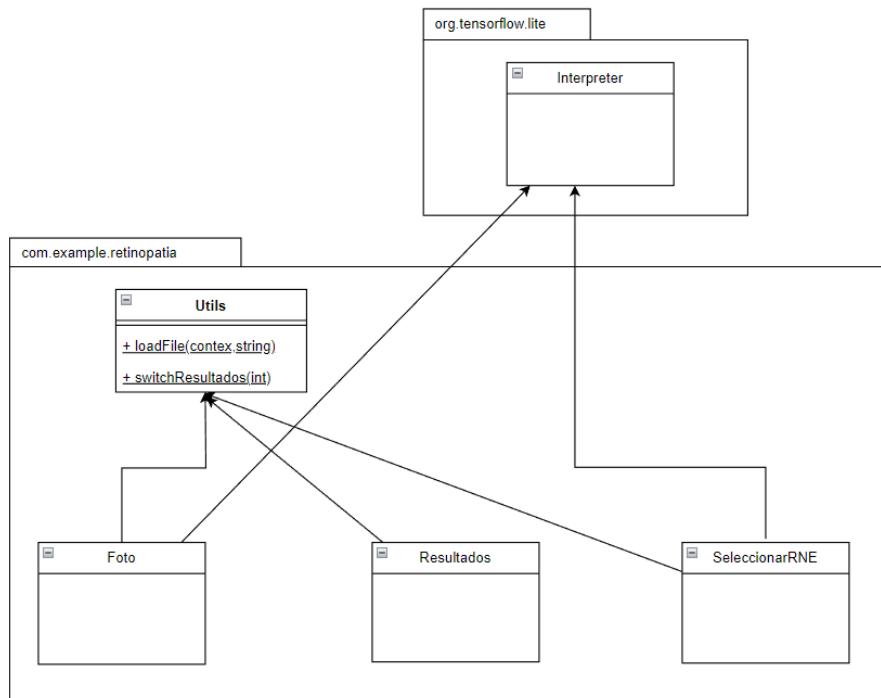


Figura C.5: Diagrama de clases de Utils e Interprete

En el segundo diagrama de clases, C.6, en el paquete “`com.example.retinopatia`”, no se incluyen las clases `Foto` y `Utils`. Sabiendo eso, las demás clases importan las clases `SQLiteDatabase`, `Cursor` y `BaseDeDatosHelper`. Por ello, para que no se liasen las líneas, se han introducido estas en los paquetes.

El motivo de la relación es que la clase `BaseDeDatosHelper` carga la base de datos sqlite almacenada en el dispositivo, cuando se va a hacer un cambio en esta, se abre en formato lectura o escritura dependiendo de la acción,

haciendo uso de la clase SQLiteDatabase. Por último, cuando se hace una consulta a la base de datos, es necesario que haya un cursor que recorra las distintas sentencias obtenidas.

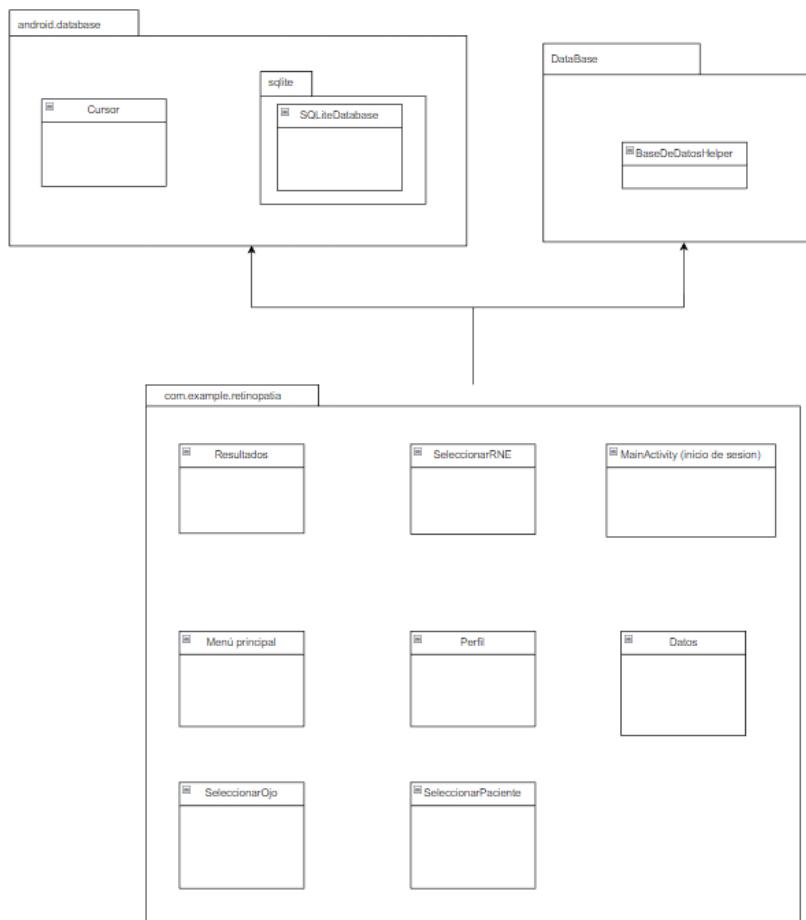


Figura C.6: Diagrama de clases relativos a la base de datos

C.5. Diseño de pruebas

Como se ha comentado, no se han implementado pruebas automáticas, sino pruebas manuales, aun así, se hizo un conjunto de pruebas, para comprobar el funcionamiento de la aplicación.

El conjunto de pruebas se puede ver en las tablas C.1, C.2, C.3, C.4

ID de la prueba	CP1	ID del requisito	RF-3, RF-3.1
Descripción	Inicio de sesión	Prioridad	3
Precondición		Postcondición	
Paso	Inputs	Resultado esperado	Resultado obtenido
1	[null,null]	mensaje de error	mensaje de error
2	[medico1@gmail.com,null]	mensaje de error	mensaje de error
3	[null, contrasena]	mensaje de error	mensaje de error
4	[medico1@gmail.com,a]	mensaje de error	mensaje de error
5	[usuario1@gmail.com,contrasena]	mensaje de error	mensaje de error
6	[medico1@gmail.com,contrasena]	inicio de sesión	inicio de sesión

Tabla C.1: Pruebas CP-1.

ID de la prueba	CP2	ID del requisito	RF-4
Descripción	Escoger paciente	Prioridad	5
Precondición	Iniciar como médico	Postcondición	
Paso	Inputs	Resultado esperado	Resultado obtenido
1	null	mensaje de error	mensaje de error
2	12345678	Mensaje paciente1	mensaje paciente1
3	11111111	Mensaje médico1	mensaje médico1
4	12341234	mensaje de error	mensaje de error
5	12345678A	mensaje de error	mensaje de error
6	[medico1@gmail.com,contrasena]	inicio de sesión	inicio de sesión

Tabla C.2: Pruebas CP-2.

C.6. Diseño de la aplicación

Inicialmente se hizo un boceto de como serían las distintas ventanas de la aplicación [C.7](#).

Posteriormente, se diseñó el logo de la aplicación, para ello, se utilizó el *prompt* “Un ícono que combine un signo de diabetes con un ojo: esta opción combina el símbolo de la diabetes (un círculo con un punto en el medio) con un ojo para sugerir la relación entre la diabetes y la salud visual. Android app style” en la herramienta DALL · E, y entre las opciones elegidas, se obtuvo [C.8](#), y se modificó para obtener [C.9](#).

Y, por último, se diseñó la aplicación final con las interfaces [C.10](#)

ID de la prueba	CP3	ID del requisito	RF-2
Descripción	Creación y comprobación de los informes invitado	Prioridad	4
Precondición	Postcondición		
Paso	Inputs	Resultado esperado	Resultado obtenido
1	iniciar sesión como invitado	ir menú principal	ir menú principal
2	botón resultados del paciente	0 informes	0 informes
3	volver	ir menú principal	ir menú principal
4	botón sacar foto al paciente	ir menú elegir ojo	ir menú elegir ojo
5	botón ojo derecho	ir menú sacar foto	ir menú sacar foto
6	sacar nueva foto	calidad no es apta	calidad no es apta
7	escoger foto no apta	calidad no es apta	calidad no es apta
8	escoger foto apta	calidad es apta	calidad es apta
9	seleccionar redes y mandar	ir menú elegir ojo	ir menú elegir ojo
10	volver	ir menú principal	ir menú principal
11	botón resultados del paciente	1 informe	1 informe

Tabla C.3: Pruebas CP-3.

ID de la prueba	CP4	ID del requisito	RF-2
Descripción	Creación y comprobación de los informes médico	Prioridad	5
Precondición	Postcondición		
Paso	Inputs	Resultado esperado	Resultado obtenido
1	[medico1@gmail.com,contraseña]	ir seleccionar DNI	ir seleccionar DNI
2	12345678	ir menú principal	ir menú principal
2	botón resultados del paciente	0 informes	0 informes
3	volver	ir menú principal	ir menú principal
4	botón sacar foto al paciente	ir menú elegir ojo	ir menú elegir ojo
5	botón ojo derecho	ir menú sacar foto	ir menú sacar foto
6	sacar nueva foto	calidad no es apta	calidad no es apta
7	escoger foto no apta	calidad no es apta	calidad no es apta
8	escoger foto apta	calidad es apta	calidad es apta
9	seleccionar redes y mandar	ir menú elegir ojo	ir menú elegir ojo
10	volver	ir menú principal	ir menú principal
11	botón resultados del paciente	1 informe	1 informe

Tabla C.4: Pruebas CP-4.

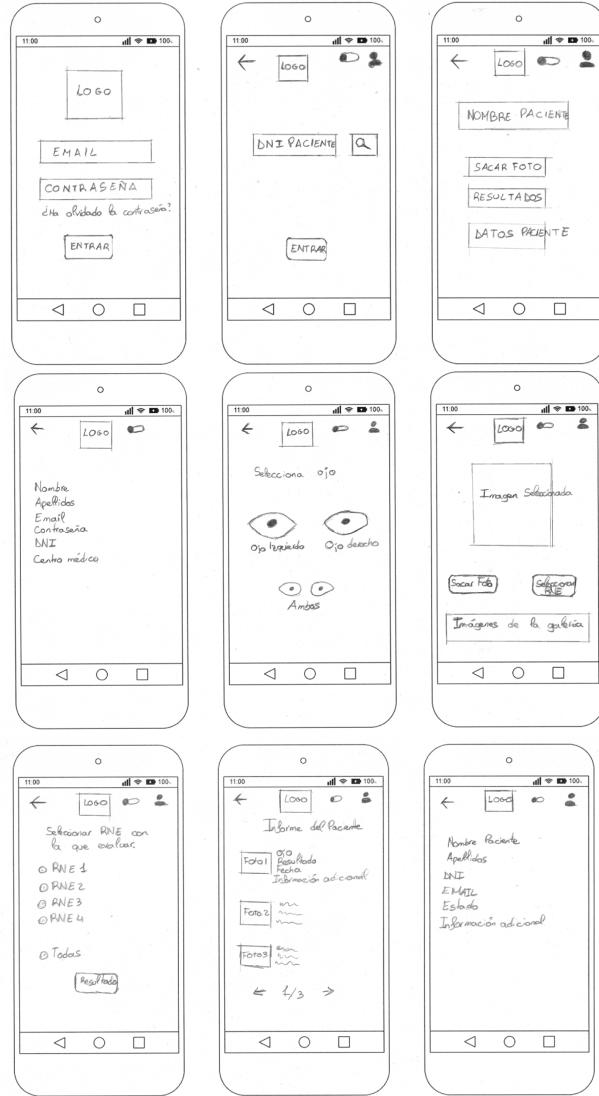


Figura C.7: Interfaz inicial de la aplicación



Figura C.8: Logo obtenido de Dalle



Figura C.9: Logo final

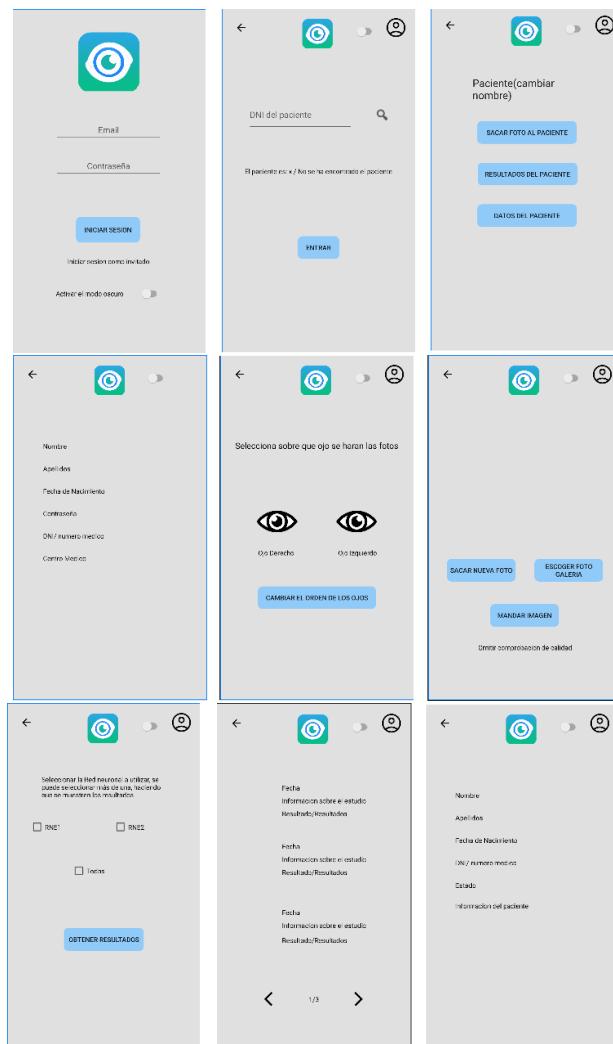


Figura C.10: Interfaz final de la aplicación

La navegación de la aplicación se vería como se puede observar en [C.11](#)

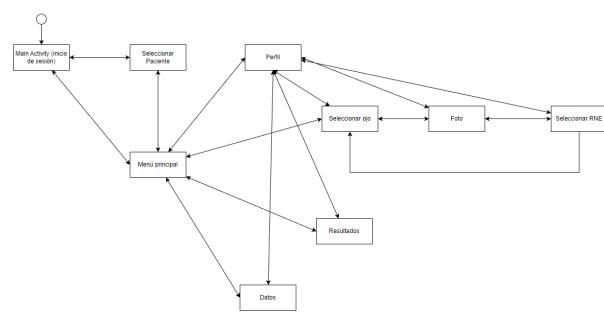


Figura C.11: Diagrama de navegación de la aplicación

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apartado se describen los aspectos técnicos de programación.

D.2. Estructura de directorios

- /: En este directorio se incluyen todos los documentos utilizados en el desarrollo del proyecto.
- /CNN y converter: En este directorio se incluyen los archivos Python con los que se ha obtenido la red neuronal convolucional y el convertidor de keras a TensorFlow Lite.
- /CNN y converter/dataset: En este directorio se incluyen las fotos para entrenar la red neuronal convolucional.
- /app/retinopatia: En este directorio se encuentra el proyecto Android Studio. Incluyendo los archivos Gradle.
- /app/retinopatia/app: Directorio correspondiente a la aplicación.
- /app/retinopatia/app/src: Directorio donde se encuentra el código fuente.
- /app/retinopatia/app/src/androidTest: Directorio con los test de interfaz gráfica de la aplicación.

- /app/retinopatia/app/src/main: Directorio donde se encuentran los archivos principales de la aplicación.
- /app/retinopatia/app/src/main/assets: Directorio donde se almacenan recursos que se utilizan durante la ejecución de la aplicación.
- /app/retinopatia/app/src/main/java: Directorio donde se encuentran las clases creadas para la aplicación.
- /app/retinopatia/app/src/res: Directorio donde se encuentran más recursos, como son layouts, imágenes (drawables), strings, etc.
- /app/retinopatia/app/src/test: Directorio donde se encuentran los tests unitarios.
- /doc: Directorio que incluye el documento LaTeX.

D.3. Manual del programador

En esta sección se explicará lo que tiene que realizar futuros programadores para trabajar con la aplicación.

Al igual que en apartados anteriores, se va a separar el proyecto Android, del proyecto de Python de la red neuronal, para una mayor comprensión.

En el proyecto Android se necesita instalar los siguientes softwares:

- Android Studio.
- Java JDK.
- SDK de Android.
- Emulador de Android.

Mientras que, para el proyecto de la red neuronal, se necesitan instalar:

- Python 3.10.8.
- Un IDE, en mi caso Visual Studio Code, con la extensión de Python.
- NumPy
- Pandas

- matplotlib
- scikit-learn
- tensorflow

Además, es necesario la herramienta Git, para descargar el repositorio y navegar entre las distintas ramas, haciendo uso de un desarrollo continuo.

Android Studio

Android Studio es el IDE oficial de Android, diseñado por Google basandose en IntelliJ IDEA. Android Studio incluye la SDK de Android y Android Virtual Device, para la simulación de la aplicación. Ademas, con el soporte de Gradle se incluyen varios Java JDK para su utilización en el proyecto Android.

En D.1, se puede observar la interfaz de Android Studio, y la emulación del dispositivo virtual creado con AVD, se pueden crear más AVD en D.3. Para la versión Android del dispositivo, se hace uso de la versión del SDK D.4, donde Android ofrece varios según la orientación del proyecto. Y por último, se puede ver en D.2 como no es necesario la instalación de un JDK, puesto que lo incluye Gradle.

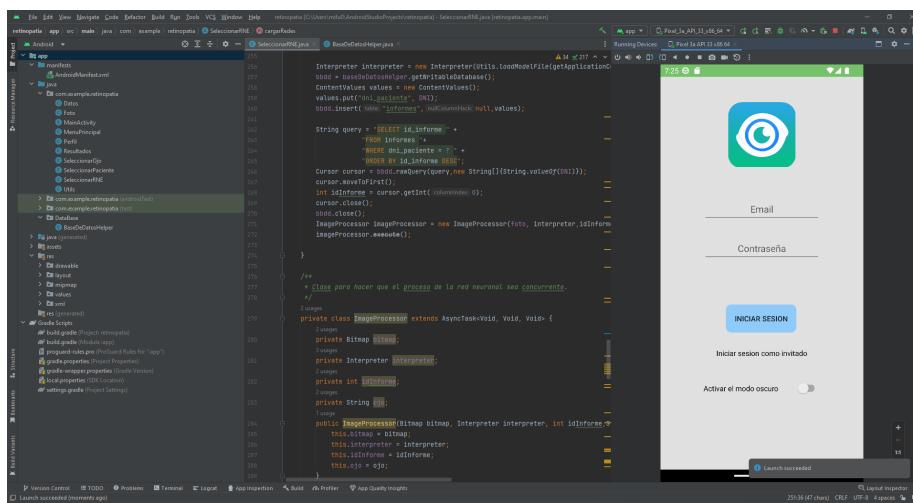


Figura D.1: Android Studio con un dispositivo virtual.

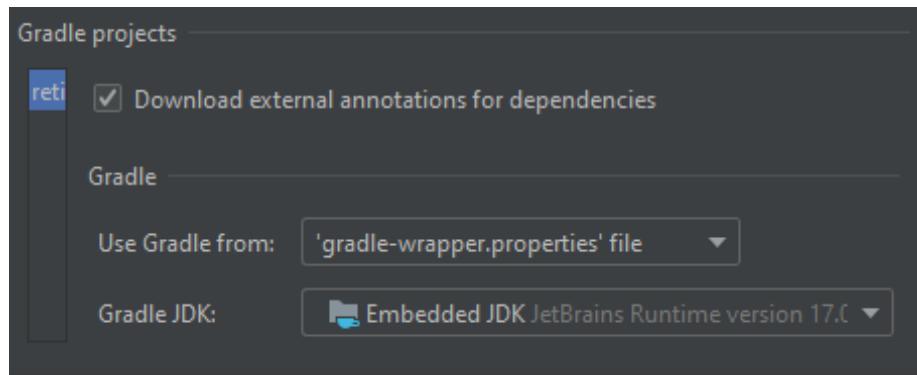


Figura D.2: Gradle incluye el JDK.

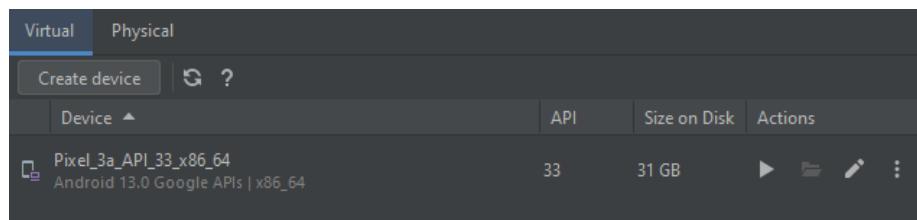


Figura D.3: Dispositivos virtuales Android.

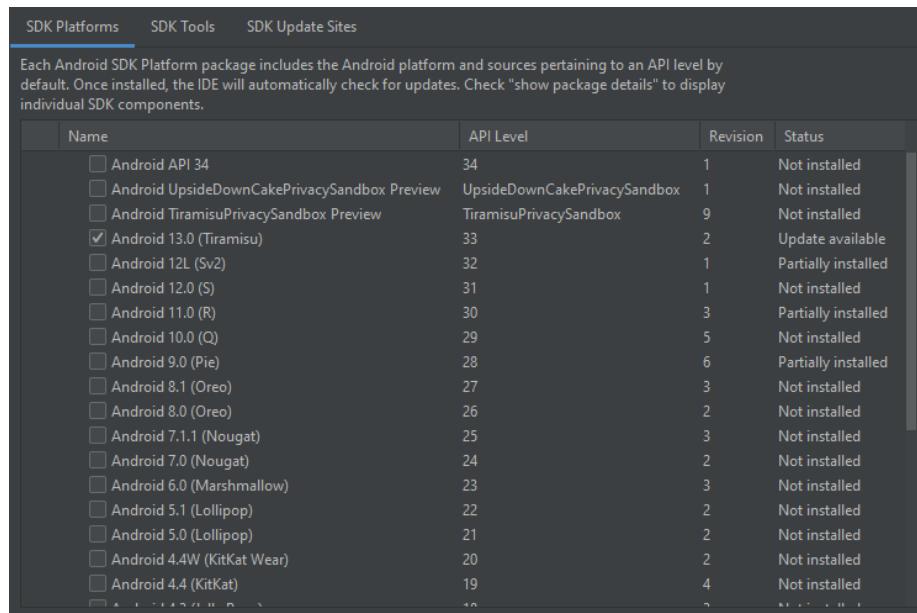


Figura D.4: SDKs ofrecidos.

D.4. Compilación, instalación y ejecución del proyecto

Instalación

Para instalar Android Studio, desde la página oficial se obtiene el ejecutable, puesto que es posible que el proyecto no sea compatible con futuras versiones de Android Studio, se ha utilizado la versión “Flamingo 2022.2.1”, en caso de incompatibilidad, la página ofrece un apartado para descargar **versiones anteriores**.

Al descargar el ejecutable, se siguen las instrucciones, y el instalador instala automáticamente el SDK de Android y el emulador. Y el JDK se incluye en el Gradle del proyecto.

Para instalar Python, se descarga desde la página oficial, en este caso como ya es una versión anterior, se va al apartado de versiones anteriores y se descarga la versión 3.10.8.

Una vez instalado, se comprueba que la variable de entorno está configurada correctamente, en caso de no estarlo, el “Path” por defecto en Windows es: *C:\Users\\$username\$\AppData\Local\Programs\Python\Python310*.

Para instalar las librerías de Python, se insertar 1 a 1 las siguientes líneas:

```
cd C:\Users\$username$\AppData\Local\Programs\Python\Python310\  
Scripts\  
pip install numpy  
pip install pandas  
pip install matplotlib  
pip install scikit-learn  
pip install tensorflow
```

Con las librerías instaladas, se procede a instalar el IDE, aunque también se puede usar un editor de texto y utilizar la consola para su ejecución. Como no es relevante el IDE que se utilice, no se comentará su instalación.

Obtención del proyecto

Para descargarse el proyecto, se recomienda hacer uso de git. Para ello, se prepara el directorio donde se va a realizar el proyecto, y posteriormente, se introduce el comando “git clone <https://github.com/mfg1014/Retinopatia-diabetica.git>”.

Con el proyecto descargado, se pueden importar los archivos Python de la carpeta “CNN y converter” en el IDE que se use, en abrir carpeta.

Para importar el proyecto Android, sería File > Open, y se selecciona la carpeta /app/retinopatía. Con ello, Android Studio detectará que es un proyecto Android, abriendo el proyecto para trabajar en él.

Ejecución del proyecto

Para el proyecto Python, la aplicación se ejecuta con el comando Python “nombre del archivo”.py. Es importante tener en cuenta que el archivo de las redes neuronales tarda bastante, debido a que tiene que crear cada ejecución el modelo VGG-16, además, se crean 30 modelos, con sus respectivas matrices de confusión.

Para el proyecto Android se puede comprobar su ejecución de distintas formas:

Emulador

Android Studio ofrece un emulador incluido, donde se pueden probar las funcionalidades de la aplicación.

Para usar un emulador, se abre el Device manager, añadiendo un nuevo dispositivo, donde se selecciona entre varios dispositivos, en la realización de la práctica, se ha utilizado el dispositivo Pixel 3a con la versión Tiramisu, también se pueden seleccionar características como los núcleos de la CPU, la memoria RAM, el espacio del dispositivo,... Una vez creado, se abre el menú “Run”, y se selecciona “Run app”.

Dispositivo físico

Para usar un dispositivo físico es necesario habilitar las opciones de desarrollador, una vez hecho, se tiene que habilitar la opción de depuración ya sea por WIFI o por USB. Cuando se vincula con Android Studio, el dispositivo informa que se quiere añadir una aplicación mediante depuración, y una vez instalada se puede usar como una aplicación normal.

Errores encontrados y nuevas funcionalidades

En caso de encontrar errores en la aplicación, se recomienda añadir una Issue al repositorio, donde se trabajará para solucionar el problema.

Si se quiere añadir nuevas funcionalidades, se recomienda crear una nueva rama por cada funcionalidad que se esté implementando concurrentemente.

Crear una nueva release

Para crear una nueva release, es necesario obtener la apk para ello, en Android Studio, se hace lo siguiente:

- Build > Build Bundle(s) /APK(s) > BUild APK(s).
- De esta forma, se genera el archivo .apk en el directorio “app/builde/outputs/apk/debug/”

Posteriormente, con el archivo APK, se comprime el directorio de la aplicación en un .zip y se va al repositorio al apartado releases, y se añade una nueva con estos archivos.

D.5. Pruebas del sistema

Como ya se ha comentado en el Diseño de pruebas, no se han realizado pruebas automáticas, pero cada vez que se realizaba un cambio de la aplicación, se comprobaba los casos de prueba relativos a las pruebas. Es importante aclarar que algunas funcionalidades no tienen pruebas específicas, pero de forma aleatoria se introducían en otros casos de prueba para comprobar el correcto funcionamiento.

Por otro lado, se han hecho pruebas en el emulador con una API Android 33 y en el dispositivo físico con una API Android 31. Por tanto, puede a ver errores en la visualización de los datos para versiones inferiores.

Mientras que en los archivos Python no es necesario el uso de pruebas.

Al finalizar el proyecto, todos los casos de prueba se pasaban correctamente.

Apéndice E

Documentación de usuario

E.1. Introducción

En este apéndice se explican los conocimientos y requisitos que debe saber el usuario para la utilización del proyecto.

En este apartado no se mencionará el proyecto realizado en Python, puesto que no está orientado al uso por un usuario que no sea desarrollador.

E.2. Requisitos de usuarios

Para que un usuario pueda utilizar la aplicación Android, es necesario:

- El usuario disponga con un dispositivo móvil Android con una versión Android 5.0 (Lollipop - API 21) o superior.
- El dispositivo debe tener cámara.
- El usuario cuente con el hardware externo para obtener fotos de la retina.
- La aplicación necesita el permiso para acceder a la cámara y al almacenamiento externo.

E.3. Instalación

Actualmente solo se dispone de instalación por apk. Para esta instalación, se debe seguir el siguiente procedimiento:

1. Se debe permitir la instalación de “origen desconocido”.
2. Se descarga la apk del repositorio desde el siguiente enlace: <https://github.com/mfg1014/Retinopatia-diabetica/releases>.
3. Una vez descargado, se ejecuta el fichero instalable.
4. Cuando termine su instalación ya se podrá hacer uso de la aplicación.

E.4. Manual del usuario

En este apartado, se busca explicar el uso que debería dar el usuario a la aplicación.

Navegación por la aplicación

Al abrir la aplicación, el usuario podrá observar la interfaz de iniciar sesión, donde podrá hacer las siguientes actividades:

- Iniciar sesión como invitado, al pulsar esta opción el usuario omite el inicio de sesión y la elección del paciente. Esta opción es la mejor para hacer pruebas o cuando no se poseen los datos del paciente o no se desea almacenar el informe en el paciente.
- Iniciar sesión, para ello, el usuario necesita introducir los datos del email y la contraseña; y que estos se encuentren en la base de datos. En caso de que no estén, se mostrará un mensaje de error.

Además, el usuario invitado no podrá seleccionar un paciente en específico, y tendrá deshabilitadas todas las opciones de ver datos del paciente y del médico.

Entonces en el diagrama de navegación, visto anteriormente, C.11, la diferencia se encontraría al iniciar sesión, y luego que el usuario invitado no tendría acceso a la interfaz Datos ni a Perfil.

Interfaz seleccionar paciente

En este caso, se va a considerar que el médico ha iniciado sesión con su email y contraseña, supongamos que estos son medico1@gmail.com y contrasena, respectivamente.

Entonces al iniciar sesión se le muestra la interfaz para seleccionar paciente; en esta interfaz el usuario deberá seguir este procedimiento:

1. Introducir un DNI, con solo los números, ejemplo 12345678.

2. Pulsar el botón “Lupa” el cual comprueba si el DNI existe en la base de datos.
3. Si es el caso, se habilita el botón para entrar al menú principal.
4. Si se ha introducido un DNI incorrecto, volver a hacer los pasos anteriores.
5. Pulsar el botón “Entrar”.

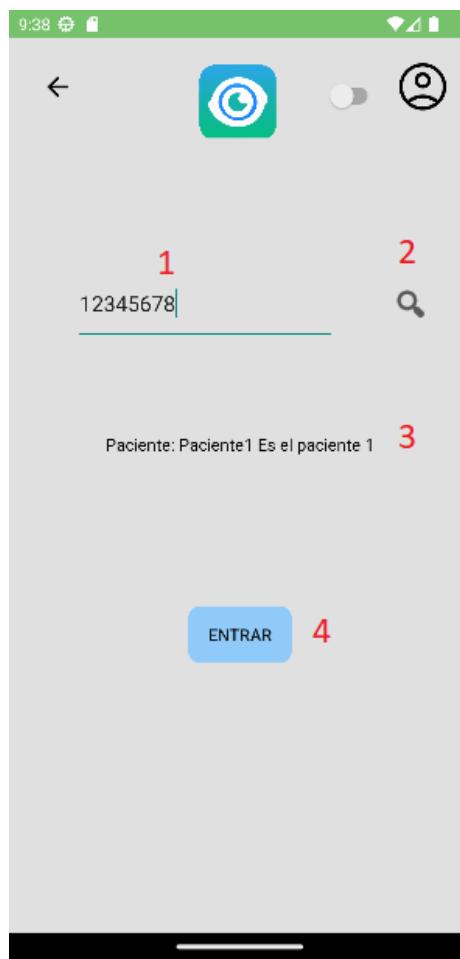


Figura E.1: Interfaz para seleccionar paciente.

Interfaz foto

Esta interfaz ya es común para ambos usuarios, su correcto funcionamiento sería:

1. El usuario escoge entre sacar una foto y elegirla desde la galería.

2. Una vez hecha la foto, se mostrará en pantalla, y al cabo de unos instantes, habilitará el botón para seleccionar la red neuronal.
3. Se recomienda al usuario sacar fotos hasta que se consideré de buena calidad, aunque si el médico lo considera buena, pulsar el botón para avanzar a la siguiente interfaz.

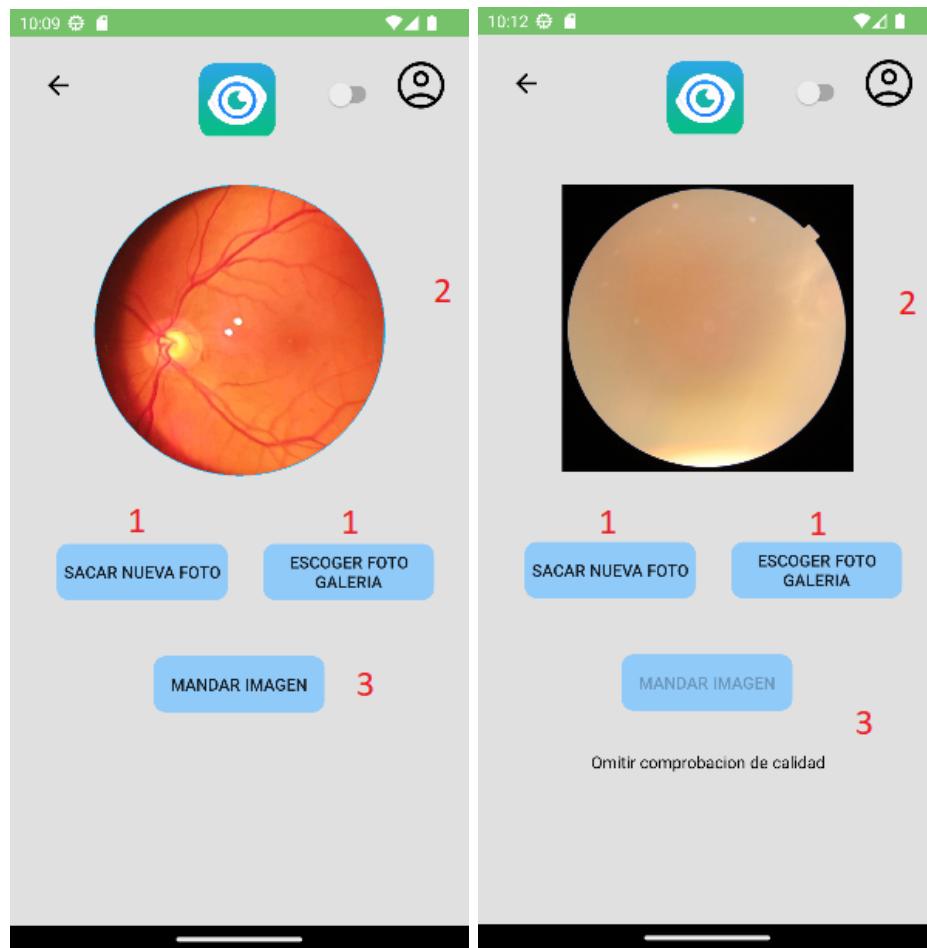


Figura E.2: Interfaces para obtener la foto.

Interfaz seleccionar red neuronal

En este caso, esta interfaz no está orientada para el entendimiento del médico puesto que no tiene conocimientos sobre las redes neuronales, pero el objetivo es facilitar estas redes de forma que a lo mejor en una marca de dispositivos una red neuronal obtiene mejor resultados y por tanto, avisar al usuario de cual se recomienda en cada caso:

1. El usuario escoge las redes neuronales con las que quiera analizar la foto realizada anteriormente
2. El usuario pulsa el botón “obtener resultados”.

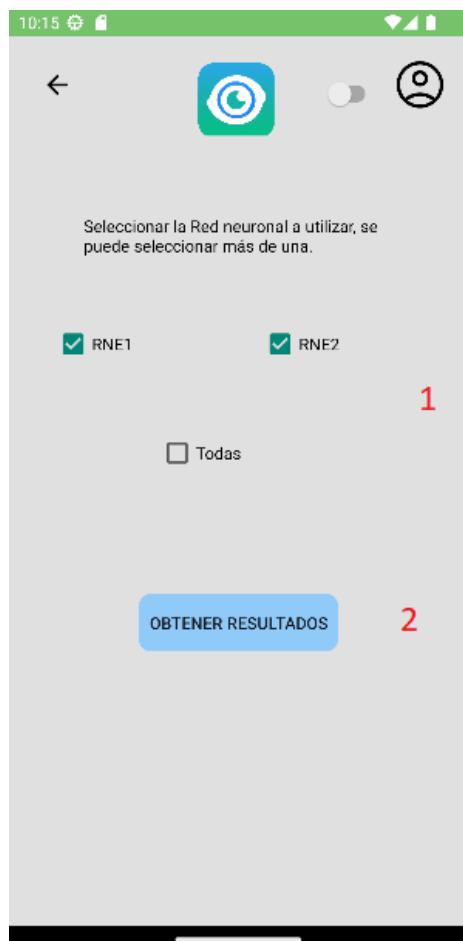


Figura E.3: Interfaz para seleccionar la red.