



Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

DED-AI: DETECCIÓN DE RETINOPATÍAS DIABÉTICAS A TRAVÉS
DE IA

DED-AI: DIABETIC RETINOPATHY DETECTION USING AI

Autor: Javier Gómez Martínez
Tutores: María Teresa García Ordás
Jose Luis Vidal de la Rosa

(Julio, 2023)

**UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y
Aeroespacial**

**GRADO EN INGENIERÍA INFORMÁTICA
Trabajo de Fin de Grado**

ALUMNO: Javier Gómez Martínez

TUTORES: María Teresa García Ordás/Jose Luis Vidal de la Rosa

TÍTULO: DED-AI: Detección de retinopatías diabéticas a través de IA

TITLE: DED-AI: Diabetetic retinopathy detection using AI

CONVOCATORIA: Julio, 2023

RESUMEN:

La retinopatía diabética, una de las múltiples complicaciones causadas por la diabetes, representa una de las principales causas de la ceguera. Su diagnóstico y control requiere de un instrumental costoso y personal cualificado, que en sistemas sanitarios como el de Castilla y León, con un alto porcentaje de población rural, es de difícil extensión a la totalidad de los ciudadanos. Como solución, se propone entrenar un modelo de aprendizaje automático, que, dada una captura de fondo de ojo, sea capaz de ofrecer una probabilidad de que ese sujeto padezca algún tipo de retinopatía. Así mismo, se ofrece un cliente móvil capaz de tomar capturas, enviarlas al servidor, y presentar los resultados de forma sencilla, ofreciendo así una solución de conjunto rápida y a un coste sumamente menor.

ABSTRACT:

Diabetic retinopathy is one of the multiple complications caused by diabetes, and also one of the main causes of blindness. Its diagnostic and control requires expensive equipment and qualified doctors, requirements that, in health systems with a high rate of rural population, like in the autonomous community of Castile and León, are difficult to extend to the whole territory. As a solution, we purpose a machine learning model, which, given a fundus image, will be able to offer a probability that the subject is affected by some type of retinopathy. A smartphone client is also offered, having the ability to take images, send them to a server and show the results in a clear fashion, achieving a global solution in a faster and much cheaper way.

Palabras clave: diabetes, retinopatía, salud, inteligencia artificial, aprendizaje automático.

Firma del alumno:

VºBº Tutor/es:

Índice de contenidos

INTRODUCCIÓN	14
Planteamiento del problema	14
Objetivos	15
Metodología empleada	16
Estructura del trabajo	16
1. ESTUDIO DEL PROBLEMA	18
1.1. Contexto del problema	18
1.2. Definición del problema	20
1.3. Estado del arte	21
1.4. Crítica al estado del arte	22
2. GESTIÓN DEL PROYECTO	23
2.1. Alcance del proyecto	23
2.2. Plan de trabajo	24
2.2.1. Identificación de fases	24
2.2.2. Plan de Iteración	24
2.2.3. Planificación de tareas	27
2.2.3.1. Definición del proyecto	27
2.2.3.2. Desarrollo del proyecto	28
2.2.3.3. Revisión y documentación	31
2.3. Gestión de recursos	31
2.3.1. Recursos humanos	32
2.3.1.1. Especificación	32
2.3.1.2. Asignación	33
2.3.2. Recursos materiales	33
2.3.2.1. Hardware	34
2.3.2.2. Software	35
2.3.2.3. Servicios	36
2.3.3. Costes totales	37
2.4. Gestión de riesgos	38

2.4.1. Identificación de riesgos	38
2.4.1.1. Riesgos genéricos	38
2.4.1.2. Riesgos específicos	39
2.4.2. Estimación de probabilidad e impacto	39
2.4.2.1. Probabilidad e impacto	39
2.4.2.2. Prioridad	40
3. SOLUCIÓN	41
3.1. Descripción de la solución	41
3.2. Análisis de la solución	42
3.2.1. Especificación de requisitos funcionales	42
3.2.2. Especificación de requisitos no funcionales	46
3.3. Tecnologías	47
3.4. Arquitectura	49
3.5. Diseño	51
3.5.1. Diseño de la aplicación móvil	51
3.5.1.1. Login	51
3.5.1.2. Welcome	52
3.5.1.3. Preview	53
3.5.1.4. Loading	54
3.5.1.5. Error	55
3.5.1.6. Results	56
3.5.1.7. Feedback	57
3.5.1.8. Thanks	58
3.5.2. Diseño del servidor REST Express	59
3.5.2.1. Parámetros generales	60
3.5.2.2. Parámetros de seguridad	60
3.5.2.3. Objetos	61
3.5.2.4. Rutas	62
3.5.3. Diseño de la base de datos	65
3.5.3.1. Colección Tokens	65
3.5.3.2. Colección Clients	66

3.5.3.3. Colección Samples	67
3.5.3.4. Colección Users	68
3.5.4. Diseño del modelo de aprendizaje automático	69
3.5.4.1. Descripción del dataset	69
3.5.4.2. Limpieza y división	70
3.5.4.3. Preprocesado	72
3.5.4.4. Aumentación	77
3.5.4.5. Arquitectura de la red neuronal	77
3.5.4.6. Parámetros de entrenamiento	81
3.5.5. Diseño del servidor Flask	83
3.6. Implementación	84
3.6.1. Estructura	84
3.6.1.1. Aplicación móvil	84
3.6.1.2. Servidor REST Express y base de datos	89
3.6.1.3. Servidor Flask y modelo	92
3.6.2. Despliegue	93
3.6.2.1. Servidor REST Express	93
3.6.2.2. Servidor Flask	94
4. EVALUACIÓN	
4.1. Aplicación cliente-servidor	95
4.1.1. Metodologías	95
4.1.2. Validación	96
4.1.2.1. Flujo de la aplicación	96
4.1.2.2. Casos de prueba	99
4.1.3. Cuestionario	110
4.1.3.1. Ficha técnica	110
4.1.3.2. Principales conclusiones	111
4.2. Modelo de aprendizaje automático	111
4.2.1. Métricas	112
4.2.1.1. Definiciones	112
4.2.1.2. Directas	112

4.2.1.3. Resumen	113
4.2.2. Resultados generales	114
4.2.3. Resultados específicos	115
4.2.3.1. Grado 1	115
4.2.3.2. Grado 2	115
4.2.3.3. Grado 3	116
4.2.3.4. Grado 4	116
CONCLUSIONES	117
Aportaciones realizadas	117
Evolución previsible	118
Problemas encontrados	118
AGRADECIMIENTOS	120
LISTA DE REFERENCIAS	122
ANEXOS	127
Anexo A: Control de Versiones	127
Anexo B: Historias	130
Anexo C: Documento Básico de Especificaciones	146
Anexo D: Resultados Completos del Cuestionario de Evaluación	155

Índice de cuadros y tablas

Tabla 1.1: Grados de retinopatía diabética	19
Tabla 2.1: Relación de salarios para cada perfil	33
Tabla 2.2: Fecha de comienzo, fin, y horas trabajadas para cada perfil	33
Tabla 2.3: Requisitos mínimos para las estaciones de desarrollo	34
Tabla 2.4: Coste de una estación de desarrollo	35
Tabla 2.5: Costes totales de hardware	35
Tabla 2.6. Costes totales de software	35
Tabla 2.7: Costes totales de servicios	36
Tabla 2.8: Costes totales del proyecto	37
Tabla 2.9: Riesgo genérico 1 – Falta de presupuesto	38
Tabla 2.10: Riesgo genérico 2 – Modificación de requisitos	38
Tabla 2.11: Riesgo genérico 3 – Superación del tiempo de desarrollo estimado	39
Tabla 2.12: Riesgo específico 1 – Brecha de seguridad	39
Tabla 2.13: Probabilidad e impacto para cada riesgo identificado	40
Tabla 2.14: Prioridad para cada riesgo identificado	40
Tabla 3.1: Requisito funcional 1 – Inicio de sesión	42
Tabla 3.2: Requisito funcional 2 – Alerta de credenciales incorrectas	43
Tabla 3.3: Requisito funcional 3 – Escena principal	43
Tabla 3.4: Requisito funcional 4 – Captura de nuevas imágenes	43
Tabla 3.5: Requisito funcional 5 – Selección de imágenes en memoria	43
Tabla 3.6: Requisito funcional 6 – Confirmación de imágenes	43
Tabla 3.7: Requisito funcional 7 – Regreso a la escena principal	44
Tabla 3.8: Requisito funcional 8 – Envío de imágenes	44
Tabla 3.9: Requisito funcional 9 – Obtención de probabilidades	44
Tabla 3.10: Requisito funcional 10 – Persistencia de resultados	44
Tabla 3.11: Requisito funcional 11 – Información de errores	44
Tabla 3.12: Requisito funcional 12 – Escena de espera	45
Tabla 3.13: Requisito funcional 13 – Visualización de resultados	45
Tabla 3.14: Requisito funcional 14 – Feedback	45

Tabla 3.15: Requisito funcional 15 – Persistencia de feedback	45
Tabla 3.16: Requisito funcional 16 – Escena de agradecimiento	46
Tabla 3.17: Requisito no funcional 1 – Tiempo de respuesta	46
Tabla 3.18: Requisito no funcional 2 – Acceso al sistema	46
Tabla 3.19: Requisito no funcional 3 – Autenticación mediante OAuth2 Password Grant	46
Tabla 3.20: Requisito no funcional 4 – Servidor API REST	46
Tabla 3.21: Requisito no funcional 5 – Servidor implementado en Golang o ExpressJS	47
Tabla 3.22: Requisito no funcional 6 – Aplicación móvil React Native	47
Tabla 3.23: Requisito no funcional 7 – Modelo de aprendizaje automático	47
Tabla 3.24: Requisito no funcional 8 – Accesibilidad	47
Tabla 3.25: Parámetros generales de la API REST	60
Tabla 3.26: Parámetros de seguridad de la API REST	61
Tabla 3.27: Ficha del objeto Image	61
Tabla 3.28: Ficha del objeto Sample	61
Tabla 3.29: Ficha del objeto User	62
Tabla 3.30: Ficha del objeto Token	62
Tabla 3.31: Ficha del método POST /auth	63
Tabla 3.32: Ficha del método POST /analysis	63
Tabla 3.33: Ficha del método PUT /analysis	64
Tabla 3.34: Ficha del método POST /register	65
Tabla 3.35: Resumen del documento Token	66
Tabla 3.36: Resumen del documento Client	67
Tabla 3.37: Resumen del documento Sample	68
Tabla 3.38: Resumen del documento User	68
Tabla 3.39: Recuento del dataset original según grado de retinopatía	71
Tabla 3.40: Recuento del dataset tras balancear	71
Tabla 3.41: Recuento entrenamiento	72
Tabla 3.42: Recuento test	72
Tabla 3.43: Recuento test grado 1	72
Tabla 3.44: Recuento test grado 2	72
Tabla 3.45: Recuento test grado 3	72

Tabla 3.46: Recuento test grado 4	72
Tabla 3.47: Ficha del método POST /predict	84
Tabla 4.1: Caso de prueba 1 – Usuario con caracteres no alfanuméricos	100
Tabla 4.2: Caso de prueba 2 – Inicio de sesión correcto	100
Tabla 4.3: Caso de prueba 3 – Usuario incorrecto	100
Tabla 4.4: Caso de prueba 4 – Contraseña incorrecta	101
Tabla 4.5: Caso de prueba 5 – Reintento con credenciales incorrectas	101
Tabla 4.6: Caso de prueba 6 – Vuelta a Login	102
Tabla 4.7: Caso de prueba 7 – Sin conexión al iniciar sesión	102
Tabla 4.8: Caso de prueba 8 – Reintento de inicio de sesión sin conexión	103
Tabla 4.9: Caso de prueba 9 – Reintento de inicio de sesión con conexión	103
Tabla 4.10: Caso de prueba 10 – Captura de fotografía mediante cámara	103
Tabla 4.11: Caso de prueba 11 – Selección de imagen mediante galería	104
Tabla 4.12: Caso de prueba 12 – Rechazo de vista previa	104
Tabla 4.13: Caso de prueba 13 – Envío de imagen correcto	105
Tabla 4.14: Caso de prueba 14 – Envío de imagen sin conexión	105
Tabla 4.15: Caso de prueba 15 – Reintento de envío sin conexión	105
Tabla 4.16: Caso de prueba 16 – Reintento de envío con conexión	106
Tabla 4.17: Caso de prueba 17 – Vuelta a Welcome tras envío de imagen fallido	106
Tabla 4.18: Caso de prueba 18 – Avance a vista Feedback	107
Tabla 4.19: Caso de prueba 19 – Envío de feedback correcto	107
Tabla 4.20: Caso de prueba 20 – Envío de feedback sin conexión	108
Tabla 4.21: Caso de prueba 21 – Reintento de envío de feedback sin conexión	108
Tabla 4.22: Caso de prueba 22 – Reintento de envío de feedback con conexión	108
Tabla 4.23: Caso de prueba 23 – Vuelta a Welcome tras envío feedback fallido	109
Tabla 4.24: Caso de prueba 24 – Finalización en Thanks	109
Tabla 4.25: Ficha técnica del cuestionario	111
Tabla 4.26: Tabla de verdad de un modelo de clasificación	112
Tabla 4.27: Resultados Generales – Conjunto de test completo	114
Tabla 4.28: Resultados Específicos 1 – Retinopatía de Grado 1	115
Tabla 4.29: Resultados Específicos 2 – Retinopatía de Grado 2	116

Tabla 4.30: Resultados Específicos 3 – Retinopatía de Grado 3	116
Tabla 4.31: Resultados Específicos 4 – Retinopatía de Grado 4	116
Tabla Anexo B.1: Historia 1 – Setup ExpressJS Environment	130
Tabla Anexo B.2: Historia 2 – Setup React Native CLI and dependencies	131
Tabla Anexo B.3: Historia 3 – Setup MongoDB local database	131
Tabla Anexo B.4: Historia 4 – Create endpoint for analysis request	131
Tabla Anexo B.5: Historia 5 – Welcome scene	132
Tabla Anexo B.6: Historia 6 – Preview scene	132
Tabla Anexo B.7: Historia 7 – Processing scene	133
Tabla Anexo B.8: Historia 8 – Error scene	133
Tabla Anexo B.9: Historia 9 – Results scene	134
Tabla Anexo B.10: Historia 10 – Deploy ExpressJS server	134
Tabla Anexo B.11: Historia 11 – Feedback scene	134
Tabla Anexo B.12: Historia 12 – Thanks scene	135
Tabla Anexo B.13: Historia 13 – Create endpoint to save feedback	135
Tabla Anexo B.14: Historia 14 – Define model architecture and training parameters	136
Tabla Anexo B.15: Historia 15 – Train and validate ML model	136
Tabla Anexo B.16: Historia 16 – Setup express-openapi and dependencies	136
Tabla Anexo B.17: Historia 17 – Create OpenAPI basic doc file	137
Tabla Anexo B.18: Historia 18 – Adapt analysis route to OpenAPI specs	137
Tabla Anexo B.19: Historia 19 – Create Token schema	137
Tabla Anexo B.20: Historia 20 – Create Sample schema	138
Tabla Anexo B.21: Historia 21 – Create User schema	138
Tabla Anexo B.22: Historia 22 – Create Client schema	139
Tabla Anexo B.23: Historia 23 – Setup MongoDB Atlas environment	139
Tabla Anexo B.24: Historia 24 – Setup OAuth2 server and helpers	139
Tabla Anexo B.25: Historia 25 – Create login endpoint	140
Tabla Anexo B.26: Historia 26 – Login scene	140
Tabla Anexo B.27: Historia 27 – Setup mongoose connection config	141
Tabla Anexo B.28: Historia 28 – Setup HTTP header protection using Helmet	141
Tabla Anexo B.29: Historia 29 – Update to UNet++ architecture	141

Tabla Anexo B.30: Historia 30 – Define separated train and test datasets for each retinopathy grade	142
Tabla Anexo B.31: Historia 31 – Equilibrate train datasets	142
Tabla Anexo B.32: Historia 32 – Implement blur and mean colour subtraction	142
Tabla Anexo B.33: Historia 33 – [Proof of Concept] Change training optimizer	143
Tabla Anexo B.34: Historia 34 – [Proof of Concept] Test different batch sizes	143
Tabla Anexo B.35: Historia 35 – Setup Flask environment	143
Tabla Anexo B.36: Historia 36 – Create POST endpoint on Flask	144
Tabla Anexo B.37: Historia 37 – Deploy Flask development	144
Tabla Anexo B.38: Historia 38 – Connect analysis endpoint to Flask server	145

Índice de figuras

Figura 1.1: Diagrama de flujo del diagnóstico de la retinopatía en CyL	20
Figura 2.1: Diagrama de Gantt de la planificación del proyecto	27
Figura 2.2: Calendario de seguimiento de la fase de desarrollo	28
Figura 2.3: Gráfico de distribución de gastos totales	37
Figura 3.1: Arquitectura de la solución desarrollada	50
Figura 3.2: Vista Login	52
Figura 3.3: Vista Welcome	53
Figura 3.4: Vista Preview	54
Figura 3.5: Vista Loading	55
Figura 3.6: Vista Error	56
Figura 3.7: Vista Results	57
Figura 3.8: Vista Feedback	58
Figura 3.9: Vista Thanks	59
Figura 3.10: Diagrama de colecciones de la base de datos	65
Figura 3.11: Varias fotografías pertenecientes a la muestra	70
Figura 3.12: Transformación de un píxel mediante convolución	73
Figura 3.13: Distribución de Gauss	74
Figura 3.14: Aplicación del filtro gaussiano a muestra	75
Figura 3.15: Sustracción de la media de color a muestra	76
Figura 3.16: Varias muestras aumentadas	77
Figura 3.17: Arquitectura UNet original	78
Figura 3.18: Arquitectura ResNet-34	79
Figura 3.19: Esquema de una red completamente conectada	80
Figura 3.20: Componente App.js	85
Figura 3.21: Área de importaciones en la vista Error	86
Figura 3.22: Funciones internas en la vista Error	87
Figura 3.23: Área de retorno en la vista Error	88
Figura 3.24: Código desarrollado para el servicio AnalysisService	89
Figura 3.25: Variables definidas en el archivo principal de Express	90

Figura 3.26: Inicializaciones en el servidor Express, parte 1	91
Figura 3.27: Inicializaciones en el servidor Express, parte 2	92
Figura 3.28: Código de la clase Net	93
Figura 4.1: Diagrama de flujo de la aplicación desarrollada, parte 1	97
Figura 4.2: Diagrama de flujo de la aplicación desarrollada, parte 2	98
Figura 4.3: Diagrama de flujo de la aplicación desarrollada, parte 3	99
Figura Anexo A.1: Tablero scrum al finalizar el proyecto	127
Figura Anexo A.2: Estadísticas de la rama main	128
Figura Anexo A.3: Gráfico de actividad por horas	129
Figura Anexo A.4: Gráfico de actividad por días de la semana	129
Figura Anexo D.1: Respuestas a la pregunta 1	155
Figura Anexo D.2: Respuestas a la pregunta 2	155
Figura Anexo D.3: Respuestas a la pregunta 3	156
Figura Anexo D.4: Respuestas a la pregunta 4	156
Figura Anexo D.5: Respuestas a la pregunta 5	156
Figura Anexo D.6: Respuestas a la pregunta 6	157
Figura Anexo D.7: Respuestas a la pregunta 7	157
Figura Anexo D.8: Respuestas a la pregunta 8	157
Figura Anexo D.9: Respuestas a la pregunta 9	158
Figura Anexo D.10: Respuestas a la pregunta 10	158
Figura Anexo D.11: Respuestas a la pregunta 11	158
Figura Anexo D.12: Respuestas a la pregunta 12	159
Figura Anexo D.13: Respuestas a la pregunta 13	159
Figura Anexo D.14: Respuestas a la pregunta 14	160
Figura Anexo D.15: Respuestas a la pregunta 15	160

Introducción

La presente memoria describe el proyecto DED-AI, desarrollado en colaboración con HP SCDS en el marco del Programa Observatorio [1]. Este proyecto tiene como objetivo la creación de un ecosistema basado en un algoritmo de clasificación de imágenes, el cual sea capaz de detectar si una fotografía de fondo de ojo presenta algún tipo de retinopatía diabética. Además del desarrollo del algoritmo, se ha diseñado e implementado una aplicación móvil capaz de tomar fotografías y enviarlas a un servidor, el cual procesa la petición y retorna el correspondiente resultado del análisis.

En esta introducción se describen brevemente el problema tratado y los objetivos del proyecto, la metodología y tecnologías empleadas, así como la estructura de este documento.

PLANTEAMIENTO DEL PROBLEMA

En España un total de 5.1 millones de adultos padecen diabetes [2]. Entre las múltiples complicaciones que puede ocasionar se encuentra la retinopatía diabética, enfermedad caracterizada por una alteración de los vasos sanguíneos retinales. Los grados de mayor desarrollo de una retinopatía acaban provocando una ceguera, lo que convierte en una obligación el muestreo ocular continuo de todo paciente con diabetes.

En Castilla y León, el diagnóstico de la retinopatía diabética se realiza utilizando retinógrafos, instrumental costoso que no está disponible en muchos centros de salud de la autonomía, dada la vasta extensión de su territorio. Además del retinógrafo, es necesario un especialista que interprete las imágenes capturadas, el cual puede que tampoco esté disponible. En el peor de los casos, se obliga al desplazamiento del paciente para realizarse una simple captura de fondo de ojo, y además, las imágenes no son evaluadas al momento, sino que son enviadas al Instituto de Oftalmología Aplicada de Valladolid [3], que recibe muestras de las nueve provincias de la comunidad.

Se generan por tanto una serie de ineficiencias en tiempo y forma, que derivan en un retraso a la hora de preescribir tratamientos o tomar otra serie de medidas para evitar el avance de la enfermedad.

La solución propuesta trata de dar respuesta a estas ineficiencias, combinando el entrenamiento de un modelo de aprendizaje automático con el desarrollo de una solución móvil, la cual permite el envío de imágenes para su evaluación por el modelo, retornando los resultados en margen de escasos segundos.

OBJETIVOS

La solución descrita en el párrafo anterior propone un sistema de detección de la diabetes usando técnicas de inteligencia artificial, a través del desarrollo de una serie de componentes diferenciados que permitan enviar y evaluar imágenes, así como mostrar los resultados de dicha evaluación. En definitiva, son objetivos del proyecto:

1. La creación de una aplicación móvil, la cual deberá:
 - a) Poder seleccionar una imagen existente en el dispositivo, o tomarla mediante la cámara.
 - b) Mostrar la imagen seleccionada al usuario para su confirmación.
 - c) Enviar la imagen al servidor para su evaluación.
 - d) Mostrar los resultados de dicha evaluación, o un posible error en el envío, si lo hubiere.
 - e) Ofrecer la posibilidad de registrar *feedback* sobre el resultado obtenido, de cara a futuras mejoras del modelo de aprendizaje.
2. Desarrollo de un servidor web API REST que canalice las solicitudes de análisis, así como el registro de *feedback*. Dado el caso, este servidor también debe servir como puerta de autenticación de los usuarios.
3. Entrenamiento de un modelo de aprendizaje automático, basado en redes neuronales convolucionales, capaz de tomar una imagen y evaluar la posibilidad de que esté afectada o no por una retinopatía diabética.
4. Construcción de una base de datos, donde se persistan las imágenes analizadas junto a un identificador único, así como el *feedback* enviado por el cliente. Estos objetivos serán desglosados más detalladamente en el capítulo reservado a explicar la solución.

METODOLOGÍA EMPLEADA

El proyecto ha sido elaborado empleando una metodología ágil, basada en un desarrollo iterativo e incremental. El marco de trabajo que se ha utilizado en todo momento ha sido Scrum, el cual se basa en los siguientes componentes:

- Una serie de iteraciones o *sprints*, en los cuales se implementan una serie de características. Estos *sprints* tuvieron una duración de entre 2 y 3 semanas, dependiendo de la carga de trabajo soportada en cada momento del curso.
- Una serie de historias de usuario o *issues*, los cuales desglosan los requisitos del producto, describiendo qué debe hacer en términos de cliente, así como una serie de claves para comprobar que la funcionalidad se ha implementado de acuerdo con dicha descripción (*acceptance criterio*).
- Una pila de producto o *backlog*, que contiene todas las historias de usuario del proyecto. En cada iteración, el desarrollador decide qué historias del *backlog* implementar, pudiendo modificarlo si se detectan nuevos requisitos, o si son exigidos por el cliente.
- Una serie de reuniones de retrospectiva, de media hora de duración, en las cuales se hace un repaso de la iteración, se realiza una demostración del producto desarrollado hasta la fecha, se planifica el siguiente *sprint*, y se resuelven dudas si las hubiese.
- Tres roles diferentes. Por un lado, el desarrollador, que implementa las diferentes historias de usuario, el *product owner*, que representa al cliente y el *scrum master*, que ofrece una serie de herramientas de apoyo al desarrollador para aumentar su rendimiento y garantizar el cumplimiento del marco *agile*. En este caso, las funciones de desarrollador recayeron en mi persona, mientras que José Luis Vidal, como tutor externo, desempeñó las labores de *product owner* y *scrum master*.

ESTRUCTURA DEL TRABAJO

La memoria del proyecto está estructurada en un total de cuatro capítulos, destinados a cubrir la totalidad del proceso de vida de éste:

- **Estudio del problema:** se ofrece una visión detallada del contexto del problema, así como una definición exhaustiva del mismo. Se realiza un análisis del estado del arte, destacando las razones que llevan a plantear la presente solución.
- **Gestión del proyecto:** definido y analizado el problema, se delimita el alcance del proyecto, y se desglosan todas las variables que influyen en el mismo, tales como la planificación, gestión de recursos y riesgos.
- **Solución:** se detalla la solución construida desde una perspectiva técnica, exponiendo la arquitectura del sistema, tecnologías empleadas, y otros aspectos relevantes de diseño e implementación.
- **Evaluación:** en base a los objetivos del proyecto así como el proceso de trabajo, se analiza el grado de satisfacción a la hora de ejecutar el proyecto, y en definitiva, si el producto desarrollado cumple con lo estipulado.

Posteriormente, y cerrando esta memoria, se detallan una serie de conclusiones personales, entre las cuales se incluyen un análisis sobre la evolución previsible del sistema, y los problemas afrontados a lo largo de la elaboración del proyecto.

1. Estudio del problema

En este capítulo se ofrece una explicación detallada del contexto del problema, así como de los sistemas utilizados en la actualidad para abordarlo. Así mismo, se analiza el estado del arte, entendido éste como las soluciones que se aproximan al enfoque empleado en el proyecto. Finalmente, se realiza una crítica en base a la información expuesta, destacando las razones que llevan a proponer el presente proyecto.

1.1. CONTEXTO DEL PROBLEMA

La diabetes es un conjunto de trastornos metabólicos caracterizados por una alteración en los niveles de glucosa en sangre. Una enfermedad que, pese a su escasa presencia mediática, es cada vez más frecuente. Según los últimos datos ofrecidos por la *Sociedad Española de Diabetes*, un total 5.1 millones de adultos en España son diabéticos, suponiendo un aumento del 42% desde 2019 [2].

Es causa de complicaciones de diversa índole, entre las que se pueden encontrar fallos del corazón, riñones, o de visión. Esta última familia de complicaciones son las retinopatías diabéticas.

En la Tabla 1.1. pueden observarse los 4 tipos de retinopatías diabéticas generalmente reconocidos por la comunidad científica. En las etapas más tempranas de la enfermedad, destaca la presencia de microaneurismas, los cuales son pequeñas dilataciones de los capilares de la retina. En estados más avanzados, éstos se entremezclan con hemorragias, al producirse la rotura los capilares, y exudados, en forma de pus. Finalmente, se producen hemorragias vítreas y formación de nuevos capilares (neovascularización), síntomas que pueden desembocar en una ceguera.

Grado	Síntomas	Retinopatía
1	Presencia de microaneurismas en uno de los cuatro cuadrantes	No proliferativa leve
2	Microaneurismas, exudados duros, blandos y algodonosos	No proliferativa moderada

3	Anomalías microvasculares en un cuadrante, arrosariamiento venoso en dos cuadrantes, exudados duros intrarretinales	No proliferativa severa
4	Neovascularización, hemorragias vítreas o prerretinales	Proliferativa

Tabla 1.1: grados de retinopatía diabética (Fuente: NAGPAL et al. [4])

Es por ello que constituye una obligación para todo sistema de salud el seguimiento periódico de los pacientes con diabetes, para evitar o mitigar el desarrollo de estas complicaciones. La Comunidad Autónoma de Castilla y León no es ajena a estos protocolos de prevención y seguimiento, los cuales presentan ciertas peculiaridades dadas las características geográficas y demográficas de su territorio.

Con una extensión de 94.223 km, Castilla y León es la tercera región más extensa de la Unión Europea, siendo a su vez la que presenta la menor densidad de población de España, con 25.34 habitantes/km² [5]. Nos encontramos por tanto ante un sistema de salud de gran complejidad, el cual cuenta, solo en la provincia de León, con un total de 651 establecimientos médicos públicos, entre consultorios, centros de salud y centros de guardia [6].

En el caso específico de la retinopatía diabética, Salud de Castilla y León (SACYL) viene desarrollando una estrategia basada en la compra de retinógrafos para la captura de fotografías de fondo de ojo. Estas imágenes permiten observar posibles alteraciones en la retina, tales como el deterioro de los vasos sanguíneos propio de las retinopatías diabéticas. El instrumental, cuyo coste oscila en magnitudes de varios miles de euros, solo se ha instalado en varios centros de salud, siendo un porcentaje significativo de población el que aún no tiene acceso *directo* a un dispositivo de este tipo.

A la falta de medios materiales se une la carencia de profesionales cualificados para la correcta interpretación de las imágenes. Debe ser un oftalmólogo, como especialista médico de la visión, quien determine finalmente si se observan alteraciones en la retina. En caso de que el establecimiento médico carezca de especialistas de este tipo, las imágenes son remitidas al Instituto de Oftalmobiología Aplicada (IOBA), institución perteneciente a la Universidad de Valladolid. Allí, un profesional específicamente

designado a esta cuestión analiza el fondo de ojo y retorna su juicio clínico al punto de toma de la fotografía.

1.2. DEFINICIÓN DEL PROBLEMA

Puede comprobarse que el sistema expuesto anteriormente presenta una serie de aspectos susceptibles de optimización. En el caso más favorable, el centro médico dispondrá tanto de un retinógrafo como de un oftalmólogo, por lo que el proceso completo, entre la toma del fondo de ojo y el diagnóstico, abarca escasos minutos. Sin embargo, en el peor de los casos, el establecimiento no dispondrá ni de medios materiales ni de personal cualificado, lo que obligará al traslado del paciente a un centro lejano, el cual podría contar con el instrumental, pero seguiría expuesto a la carga de trabajo que afronte el IOBA, pudiéndose demorar el diagnóstico varios días. El diagrama de flujo expuesto en la *Figura 1.1* refleja esta situación.

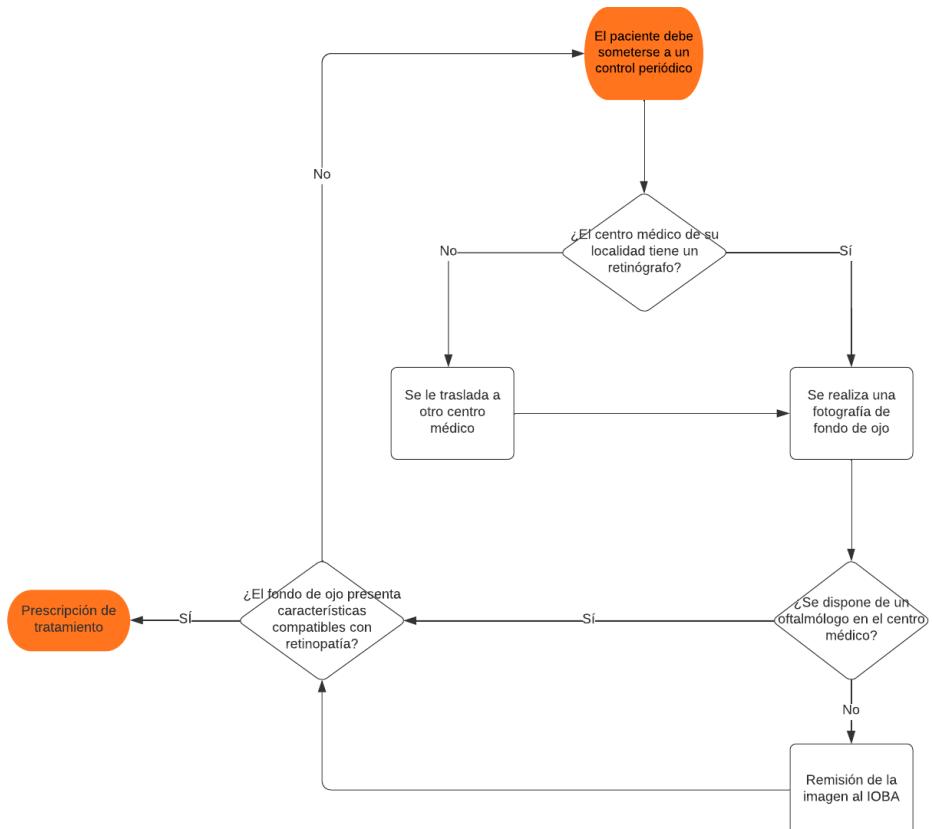


Figura 1.1. Diagrama de flujo del diagnóstico de la retinopatía en CyL (Elaboración propia)

Por ello, el proyecto actual persigue ofrecer una solución que, en primer lugar, agilice el proceso de diagnóstico de la retinopatía diabética, anticipando las etapas de tratamiento y prevención de futuras etapas de la enfermedad. Por otro lado, este proceso de diagnóstico trata de ser universal, consiguiendo que la totalidad de la población tenga acceso a un método sencillo y asequible económicamente, mejorando en definitiva, la calidad del sistema de salud pública de Castilla y León.

1.3. ESTADO DEL ARTE

La detección y clasificación de la retinopatía diabética mediante imágenes de fondo de ojo es un campo de investigación emergente, pudiéndose encontrar literatura abundante al respecto. Una reciente publicación de NAGPAL et al. [4] ofrece información sobre las técnicas existentes, así como un cuadro resumen de los estudios más destacados hasta la fecha.

Dentro de los estudios que utilizan aprendizaje automático, podemos distinguir dos grandes grupos: aquellos que utilizan algoritmos tradicionales de clasificación, como máquinas de vectores de soporte (SVM), regresión logística, o *Naive-Bayes*; y los que emplean las más recientes redes neuronales convolucionales (CNNs).

En este último grupo destaca el estudio de PRATT et al. [7], el cual entrena un modelo de CNN mediante el dataset ofrecido por la *California Healthcare Foundation* en la comunidad de aprendizaje automático *Kaggle*, obteniendo una exactitud del 75% en la clasificación de las diferentes clases de retinopatía.

Fuera de la recopilación ofrecida por NAGPAL et al., se encuentran otros estudios que demuestran la viabilidad de emplear imágenes tomadas por un dispositivo móvil para el diagnóstico de la retinopatía diabética. RAJALAKSHMI et al. utilizaron en 2018 un *smartphone* para capturar imágenes de fondo de ojo, y la solución propietaria EyeArt para la identificación de retinopatías, obteniendo una especificidad del 80.2% en el caso de clasificación binaria [8].

Concluyeron que “*un algoritmo de clasificación basado en IA, en combinación con el uso de smartphones validados para la captura de imágenes de pacientes diabéticos, puede ser utilizado para el cribado de pacientes con retinopatía diabética, los cuales serían remitidos*

después a un especialista para una evaluación más amplia y tratamiento”.

1.4. CRÍTICA AL ESTADO DEL ARTE

La información ofrecida anteriormente sustenta la viabilidad de utilizar un algoritmo de aprendizaje automático para el diagnóstico de la retinopatía diabética. En algunos casos, es posible descargar el producto de estas investigaciones y utilizarlo para fines no comerciales.

Sin embargo, los estudios que evalúan la eficacia de emplear un enfoque móvil hacen uso de tecnologías propietarias, no estando disponibles de forma libre. Ésta es la principal debilidad del estado del arte, al suponer una barrera que puede desincentivar optar por una solución de *machine learning* frente al actual enfoque basado en la adquisición y uso de retinógrafos.

En conclusión, no existe una solución pública que combine un modelo de aprendizaje automático junto con la toma de fotografías mediante *smartphone*, por lo que se aconseja la elaboración de una alternativa que elimine esta barrera de acceso y solucione los problemas existentes en el sistema de diagnóstico actual.

2. Gestión del proyecto

A continuación se detallan los aspectos más relevantes de la gestión del proyecto, mediante una simulación empresarial del mismo. Se definirá y describirá el alcance del proyecto, plan de trabajo, gestión de tareas, recursos y riesgos, así como el presupuesto de ejecución.

2.1. ALCANCE DEL PROYECTO

El proyecto consiste en el desarrollo de una solución *full-stack* para la detección de la retinopatía diabética. Para ello, contará con una aplicación móvil para el envío de capturas de fondo de ojo, un servidor para canalizar las peticiones de los usuarios, un modelo de aprendizaje automático que determine la probabilidad de afección por una retinopatía, y una base de datos para persistir información sobre usuarios y análisis.

El público objetivo de la aplicación serán profesionales de la salud, que desarrollem su actividad tanto en centros de atención primaria como especializados. El usuario no necesitará conocimientos avanzados de oftalmología, sin embargo, se recomienda utilizar la solución desarrollada como una herramienta de cribado, contrastándose posteriormente los resultados ofrecidos con un especialista de la visión.

Los usuarios únicamente tendrán contacto directo con la aplicación móvil, la cual será desarrollada de manera que sea sencilla e intuitiva. Cualquier persona, independientemente del nivel de conocimiento informático que posea, debe ser capaz de utilizar la aplicación sin ninguna dificultad. Esta sencillez también se traducirá en el empleo de estándares de accesibilidad que garanticen el uso de la aplicación por parte de colectivos con alguna discapacidad.

Como límite, la aplicación solo estará disponible en idioma español. Así mismo, en su versión inicial, solo se garantiza su funcionamiento bajo sistema operativo Android, aunque las tecnologías empleadas permitan una fácil portabilidad a otras plataformas. No se requerirá de ninguna otra aplicación complementaria para el uso de la solución.

2.2. PLAN DE TRABAJO

A continuación se ofrecerá una visión general del proceso de desarrollo del proyecto, agrupado en tres fases diferenciadas. Posteriormente, se ofrecerá un mayor detalle de las distintas iteraciones del proyecto, siguiendo un enfoque más técnico y desglosando las tareas realizadas como historias de usuario.

2.2.1. IDENTIFICACIÓN DE FASES

El proyecto ha sido dividido en tres fases de alto nivel, ofreciéndose una breve descripción de cada una de ellas desde un punto de vista empresarial.

- **Definición del proyecto:** etapa en la que cliente y desarrollador fijan los objetivos del producto, definen la funcionalidad demandada y los requisitos necesarios para implementar la solución, incluyendo el estudio de las tecnologías más adecuadas. Todo queda reflejado en la pila de producto o *producto backlog*.
- **Desarrollo del proyecto:** implementación de los requisitos, en base a una serie de historias de usuario, organizando el trabajo por iteraciones o *sprints* de entre 2 y 4 semanas de duración. Al inicio de cada *sprint*, se realiza una reunión de planificación, y al concluir la iteración, una de retrospectiva.
- **Revisión y documentación:** finalizada la implementación de la funcionalidad, se documenta el producto desarrollado, y se revisa que se ha desarrollado siguiendo las directrices del cliente, empleando pruebas de caja negra y casos de uso.

2.2.2. PLAN DE ITERACIÓN

Identificadas las fases del proyecto, se procede a definir con mayor detalle las tareas a llevar a cabo en cada una de ellas, a modo de estimación.

- **Definición del proyecto:** en esta fase, se desarrollarán una serie de reuniones con el cliente, para en primer lugar, determinar la arquitectura básica del sistema y las tecnologías a emplear. También debe definirse el documento básico de especificaciones, donde se incluyan los objetivos del proyecto, funcionalidad

deseada y metodología de trabajo. De dicho documento se extraerán los requisitos, reflejados como historias de usuario, realizando una definición inicial de la pila de producto. Siguiendo el marco *scrum*, también se realizará una reunión de planificación de iteraciones.

- **Desarrollo del proyecto:** se estructura en 7 iteraciones o *sprints*, de duración variable en función de la carga de trabajo y disponibilidad del desarrollador y el cliente:

- **Primer sprint:** reunión *kick-off* con el cliente y *scrum master*, repasando la pila de requisitos. Instalación y configuración de las distintas herramientas en el equipo de trabajo, e inicio de desarrollo de la parte servidora. Formación y aprendizaje de las tecnologías empleadas, así como sobre el dominio del proyecto. Reunión de retrospectiva, repasando el grado de cumplimiento de los objetivos, y ajustando la velocidad de ejecución.
- **Segundo sprint:** reunión de planificación *follow-up*, extrayendo de la pila de producto las historias de usuario deseadas para el final de la iteración. Desarrollo de las distintas vistas de usuario, formando un *end to end* básico desde la pantalla de inicio hasta la de resultados. Despliegue del servidor en una máquina accesible desde Internet. Reunión de retrospectiva, incluyendo, además del repaso al gráfico *burn-down*, una demostración del cliente móvil inicial.
- **Tercer sprint:** reunión *follow-up*, fijando los objetivos para la presente iteración. Desarrollo de la vista de *feedback* de la aplicación, así como del correspondiente punto de comunicación en el servidor. Definición de la arquitectura de la red neuronal, así como de los parámetros de entrenamiento iniciales. Creación de un modelo inicial de aprendizaje automático. Reunión de retrospectiva, incluyendo una nueva demostración de la funcionalidad implementada, así como repasando los resultados iniciales del modelo de aprendizaje.
- **Cuarto sprint:** reunión *follow-up*, estableciendo las historias de usuario a implementar en esta iteración. Adaptación del servidor desarrollado a la especificación OpenAPI [9]. Definición de las colecciones de la base de datos,

e implementación de los esquemas que permitan la inserción y modificación de documentos en dichas colecciones. Despliegue de la base de datos en un servidor en la nube. Implementación de control de acceso, mediante autenticación por usuario y contraseña de acuerdo con el estándar OAuth2 [10]. Reunión de retrospectiva, realizando una demostración de la nueva funcionalidad incorporada, así como repasando el grado de ejecución de los objetivos planteados.

- **Quinto sprint:** *follow-up* con el cliente, escogiendo las historias de usuario a implementar. Creación de la vista de *Login*. Conexión del servidor a base de datos. Revisión de la seguridad del servidor. Reunión de retrospectiva, mostrando el nuevo *end-to-end* actualizado, así como revisando el ritmo de ejecución y gráfico *burn-down*.
- **Sexto sprint:** *follow-up* del proyecto, estableciendo las metas de la nueva iteración. Revisión de la arquitectura del modelo de aprendizaje automático. Revisión de las etapas de limpieza y preprocesado de la muestra. Creación de *scripts* para el tratamiento automatizado de la muestra. Prueba con nuevos parámetros de entrenamiento. Reunión de retrospectiva, revisando los resultados ofrecidos por el modelo actualizado y trabajo restante en el proyecto.
- **Séptimo sprint:** reunión de planificación o *follow-up*, revisando y escogiendo las historias de usuario para esta iteración. Diseño del servidor que servirá el modelo de aprendizaje automático. Implementación de dicho servidor. Despliegue en una máquina accesible desde Internet. Conexión de los dos servidores desarrollados. Reunión de retrospectiva, incluyendo demostración de la solución desarrollada.
- **Revisión y documentación:** en la última fase, se realiza una evaluación del producto desarrollado, comprobando que cumple los requisitos demandados por el cliente en el documento básico de especificaciones. Para ello, se emplean pruebas de caja negra. Por último, se preparan los entregables finales, y se documenta exhaustivamente todo el producto final, incluyendo todos los apartados desarrollados en la presente memoria.

2.2.3. PLANIFICACIÓN DE TAREAS

Con el plan de fases e iteraciones expuesto en los apartados anteriores, comenzó la elaboración del proyecto, el cual tuvo como fecha de inicio el 30 de septiembre de 2022, y finalización el 12 de junio de 2023. Téngase en cuenta que las iteraciones tuvieron una duración variable de entre 2 y 4 semanas, en función de festivos y vacaciones, calendario académico y carga de trabajo por parte del desarrollador, así como la planificación propia del tutor, José Luis Vidal, y la empresa HP SCDS, para la cual trabaja y que ha proporcionado espacios y herramientas para el desarrollo. En la *Figura 2.1* se observa un diagrama de Gantt con la planificación temporal de las distintas fases y *sprints*.

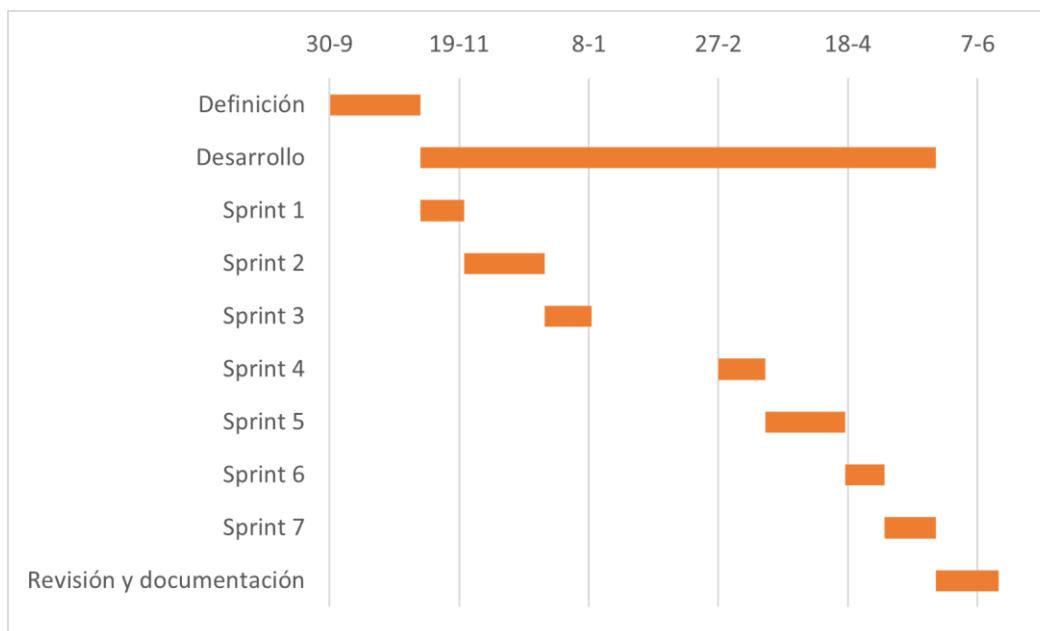


Figura 2.1: Diagrama de Gantt de la planificación del proyecto

Se detallan a continuación las fechas de cada fase e iteración, así como la correspondencia entre las tareas mencionadas en el apartado 2.2.2 y las historias de usuario implementadas.

2.2.3.1. DEFINICIÓN DEL PROYECTO

Tiene una duración de 35 días. A lo largo de esta fase, tienen lugar los siguientes hitos:

- **Reunión de presentación:** realizada el 30 de septiembre de 2022, de forma

telemática, repasando los antecedentes, objetivos y tecnologías propuestas, así como la metodología de trabajo.

- **Definición de documento de requisitos:** entre el 30 de septiembre y el 7 de octubre. Los asuntos tratados en la reunión de presentación son volcados a un documento, el cual se ofrece en el *Anexo C*.
- **Ánalysis de requisitos y arquitectura:** entre el 7 de octubre y el 4 de noviembre. Se identifican los componentes que conforman el sistema propuesto, así como los requisitos en forma de historias de usuario, conformando la pila de producto inicial.
- **Reunión de planificación de iteraciones:** se realiza el 4 de noviembre, como fin de fase y antesala al inicio de la fase de desarrollo.

2.2.3.2. DESARROLLO DEL PROYECTO

A lo largo de 199 días, se sigue una planificación basada en *scrum*, con 7 iteraciones diferenciadas. En la *Figura 2.2* puede observarse el calendario de seguimiento de la fase, incluyendo las fechas de inicio, fin y duración de cada iteración. Nótese que existe un vacío entre la tercera y la cuarta iteración, motivado por el espacio entre los semestres académicos del curso.

Tarea	Inicio	Fin	Duración
Sprint 1	04/11/2022	21/11/2022	17
Sprint 2	21/11/2022	22/12/2022	31
Sprint 3	22/12/2022	09/01/2023	18
Sprint 4	27/02/2023	17/03/2023	18
Sprint 5	17/03/2023	17/04/2023	31
Sprint 6	17/04/2023	02/05/2023	15
Sprint 7	02/05/2023	22/05/2023	20
Desarrollo	04/11/2022	22/05/2023	199

Figura 2.2: Calendario de seguimiento de la fase de desarrollo

Se detalla a continuación cada *sprint*, incluidos sus hitos más relevantes y las historias de usuario implementadas, descritas con mayor detalle en el *Anexo B* (en inglés).

- **Sprint 1:** tiene una duración de 17 días. Comienza con la reunión *kick-off*, el día 4 de noviembre, y finaliza con la reunión de retrospectiva, el 21 de noviembre.

Durante esta iteración, se desarrollan las siguientes tareas:

- **Instalación y configuración de las herramientas de trabajo:** abarca las siguientes historias:
 - Setup ExpressJS environment
 - Setup React Native CLI and dependencies
 - Setup MongoDB local database
- **Inicio de desarrollo de la parte servidora:** abarca la historia:
 - Create endpoint for analysis request
- **Sprint 2:** tiene una duración de 31 días. Comienza con la reunión *follow-up*, el 21 de noviembre, y finaliza con la reunión de retrospectiva, el 22 de diciembre. Esta iteración aglutina las siguientes tareas:
 - **Desarrollo de las vistas de usuario:** abarca las historias:
 - Welcome scene
 - Preview scene
 - Processing scene
 - Error scene
 - Results scene
 - **Despliegue del servidor en una máquina accesible en internet:** abarca la historia:
 - Deploy ExpressJS server
- **Sprint 3:** dura 18 días. Comienza con el *follow-up* realizado el 22 de diciembre de 2022 y termina con la reunión de retrospectiva, el 9 de enero de 2023. En esta iteración se desarrollan las siguientes tareas:
 - **Desarrollo de la vista de *feedback*:** abarca las siguientes historias:
 - Feedback scene
 - Thanks scene
 - Create endpoint to save feedback
 - **Definición y entrenamiento del modelo inicial de aprendizaje automático:** abarca las historias:
 - Define model architecture and training parameters
 - Train and validate ML model

- **Sprint 4:** dura 18 días. Comienza tras el parón entre semestres, con la reunión *follow-up* del 27 de febrero, y finaliza el 17 de marzo al realizarse la reunión de retrospectiva. Se realizaron las siguientes tareas:
 - o **Adaptación del servidor a la especificación OpenAPI:** abarca las siguientes historias:
 - Setup express-openapi and dependencias
 - Create OpenAPI basic doc file
 - Adapt analysis route to OpenAPI specs
 - o **Definición de las colecciones de la base de datos:** abarca las historias:
 - Create Token schema
 - Create Sample schema
 - Create User schema
 - Create Client schema
 - o **Despliegue de la base de datos en un entorno cloud:** abarca la historia:
 - Setup MongoDB Atlas environment
 - o **Implementación de control de acceso mediante OAuth2:** abarca las historias:
 - Create login endpoint
 - Setup OAuth2 server and helpers
- **Sprint 5:** dura 31 días. Da comienzo al *sprint* la reunión *follow-up* celebrada el 17 de marzo, y concluye con la reunión de retrospectiva, el 17 de abril. Se desarrollan las siguientes tareas:
 - o **Creación de la vista de Login:** abarca la historia:
 - Login scene
 - o **Conexión del servidor a base de datos:** abarca la historia:
 - Setup mongoose connection config
 - o **Revisión de la seguridad del servidor:** abarca la historia:
 - Setup HTTP header protections using Helmet
- **Sprint 6:** tiene una duración de 15 días. Comienza con el *follow-up* del 17 de abril y finaliza con la reunión de retrospectiva del 2 de mayo. Se desarrollan las siguientes tareas:

- **Revisión de la arquitectura del modelo de aprendizaje:** abarca la historia:
 - Update to Unet++ architecture
- **Revisión de las etapas de preprocesamiento y limpieza:** abarca las historias:
 - Define separated train and test datasets for each retinopathy grade
 - Equilibrate train datasets
 - Implement blur and mean color subtraction preprocess
- **Prueba con nuevos parámetros de entrenamiento:** abarca las historias:
 - [Proof of Concept] Change training optimizer
 - [Proof of Concept] Test different batch sizes and epochs
- **Sprint 7:** última iteración, de 20 días de duración. Se inicia el 2 de mayo con la reunión de *follow-up* y finaliza el 22 de mayo con la reunión de retrospectiva. Se desarrolla la tarea:
 - **Diseño, implementación y despliegue del servidor dedicado para el modelo de aprendizaje:** abarca las tareas:
 - Setup Flask environment
 - Create POST endpoint on Flask
 - Deploy Flask environment
 - **Conexión de los dos servidores desarrollados:** abarca la tarea:
 - Connect analysis endpoint to Flask server

2.2.3.3. REVISIÓN Y DOCUMENTACIÓN

La última fase del proyecto abarca 24 días, desde el 22 de mayo, cuando se realiza la última reunión de retrospectiva, hasta el 15 de junio, cuando se finalizan los distintos entregables. Las tareas, que incluyen la realización de pruebas de caja negra, documentación y finalización de entregables, no se organizan de una forma diferenciada, sino que se realizan en paralelo durante toda la duración de la fase.

2.3. GESTIÓN DE RECURSOS

Este subapartado realiza una estimación de los distintos recursos, tanto humanos como

Javier Gómez Martínez

materiales, necesarios para ejecutar el proyecto, incluyendo una estimación de costes económicos y de tiempo empleado por cada trabajador.

2.3.1. RECURSOS HUMANOS

En primer lugar se especificarán los perfiles necesarios para conformar el equipo de desarrollo, junto con su salario medio. Después, se realizará una asignación de recursos en función de cada perfil y la relevancia de éste en cada fase del proyecto.

2.3.1.1. ESPECIFICACIÓN

Para desarrollar la solución planteada, se estima que son necesarios 5 perfiles distintos:

- Un **desarrollador web**, familiarizado en el uso de tecnologías y *frameworks* basados en JavaScript. Implementará la aplicación móvil.
- Un **analista de sistemas** para diseñar y verificar la arquitectura y requisitos de la solución.
- Un **técnico de sistemas** encargado de la implementación y despliegue de los servidores, así como de la base de datos.
- Un **científico de datos** encargado de todos los aspectos relacionados con el modelo de aprendizaje automático: limpieza y preprocesado de los datos, determinación de la arquitectura de aprendizaje más adecuada, y entrenamiento y validación de modelos.
- Un **scrum master** que instruya al equipo de desarrollo en el uso de buenas prácticas software, así como en la utilización de herramientas y metodologías de desarrollo ágil.

En la *Tabla 2.1* figura el salario bruto anual de cada uno de estos perfiles, junto a su salario por horas.

Perfil	Bruto anual (€)	Salario/hora (€/hora)
Desarrollador web	29837 [11]	14,86
Analista de sistemas	29845 [12]	14,86

Técnico de sistemas	24492 [13]	12,19
Científico de datos	38175 [14]	19,01
Scrum master	42195 [15]	21,01

Tabla 2.1: Relación de salarios para cada perfil (Fuente: Indeed)

El salario por hora ha sido calculado según la *Ecuación 2.1*:

$$\text{Salario/hora} = \frac{\text{Bruto anual}}{251 \text{ días laborables} * 8 \text{ horas diarias}} \quad (2.1)$$

2.3.1.2. ASIGNACIÓN

La asignación de trabajo a cada uno de los perfiles ha sido realizada basándose en las iteraciones y/o fases en las que mayor relevancia tiene dicho trabajador. Como puede observarse en el apartado 2.2.3, las tareas de desarrollo de cada componente no se realizan en paralelo, existiendo periodos en los que no se trabaja en un componente determinado. Por ello, la *Tabla 2.2* refleja las fechas de comienzo y fin del trabajo de cada perfil, no existiendo, sin embargo, relación directa entre estas fechas y el número de horas trabajadas, pues el trabajo realizado no se desempeña de forma continua.

Perfil	Comienzo	Fin	H. trabajadas	Salario (€)
Desarrollador web	04/11/2022	17/04/2023	664	9867,04
Analista de sistemas	30/09/2022	15/06/2023	1208	17950,88
Técnico de sistemas	04/11/2022	22/05/2023	864	10532,16
Científico de datos	22/12/2022	02/05/2023	200	3802
Scrum master	04/11/2022	22/05/2023	864	18152,64

Tabla 2.2: Fecha de comienzo, fin, y horas trabajadas para cada perfil

2.3.2. RECURSOS MATERIALES

Analizados los medios humanos necesarios para llevar a cabo el proyecto, a continuación

se detallan los costes producidos por la adquisición de servicios, así como de material hardware y software.

2.3.2.1. HARDWARE

Para alcanzar un nivel óptimo de productividad, se ve conveniente dotar a cada miembro del equipo de una estación de trabajo dedicada. Éstas no requieren de una capacidad de cómputo excesivamente elevada, pues únicamente se utilizarán para labores de desarrollo, pruebas y elaboración de documentación. Por tanto, se propone adquirir cinco equipos de sobremesa que cumplan con los requisitos mínimos estipulados en la *Tabla 2.3*.

Requisito	Valor
CPU	Intel Core i3/AMD Ryzen 3 o equivalente
Memoria RAM	8 GB
Almacenamiento	500 GB SSD
GPU	2 GB VRAM
Monitor	LED 19"
Teclado	QWERTY Mecánico
Ratón	Óptico

Tabla 2.3: Requisitos mínimos para las estaciones de desarrollo (Elaboración propia)

En base a los requisitos especificados, una posible configuración sería la indicada en la siguiente tabla:

Componente	Precio (€)
CPU AMD Ryzen 3 4100 3.8 Ghz	64,99
Placa MSI B450M PRO-VDH Max	73,99
Kingston FURY Beast DDR4 2666 Mhz 8 GB	21,99
Caja Nox LITE010 + Fuente de Alimentación 500 W	42,99
SSD Crucial MX500 500GB SATA	42,99

Gráfica Asus GeForce GT 730 2GB GDDR5	73,99
Monitor Asus VS197DE 19"	82,99
Ratón + Teclado NGS Cocoa Kit	10,99
TOTAL	414,92

Tabla 2.4: Coste de una estación de desarrollo (Fuente: PCComponentes [16])

Durante las pruebas de la aplicación móvil, también se empleará un dispositivo Google Pixel 4a 5G, cuyo coste asciende a 286,99 euros, según Amazon [17]. En la *Tabla 2.5* se recapitulan todos los costes de hardware.

Recurso	Cantidad	Precio unitario (€)	Subtotal (€)
Estación de trabajo	5	414,92	2074,60
Google Pixel 4ª 5G	1	286,99	286,99
TOTAL			2361,59

Tabla 2.5: Costes totales de hardware

2.3.2.2. SOFTWARE

A lo largo del desarrollo no son necesarias herramientas de pago, por lo que únicamente tendrá que afrontarse el coste de adquisición de las licencias del sistema operativo Windows 11 Pro, para cada una de las cinco estaciones de trabajo planteadas. El coste unitario de la licencia asciende a 259 euros, según el sitio web oficial de Microsoft [18].

Recurso	Cantidad	Precio unitario (€)	Subtotal (€)
Licencia Windows 11 Pro	5	259	1295
TOTAL			1295

Tabla 2.6: Costes totales de software

2.3.2.3. SERVICIOS

Para el entrenamiento del modelo de aprendizaje automático, se estima que serán necesarios, al menos, 40 GB de memoria RAM dedicada a gráficos (VRAM). La primera opción para llevar a cabo esta tarea consiste en adquirir un equipo para trabajar en local. Una búsqueda en PCComponentes arroja un único resultado de GPU que cumple con la especificación anterior, una Nvidia Quadro RTX A6000, con un precio de 7491 euros [19].

Dado el elevado coste que supondría adquirir hardware para realizar un entrenamiento local, se propone recurrir a servicios basados en la nube. Éstos suelen ofrecer acceso a *clusters* dedicados, facturando en función del número de horas empleadas en el entrenamiento. Una opción destacable es Google Colab, servicio consistente en un entorno de ejecución basado en Python, respaldado por los centros de datos de la compañía norteamericana.

Los servicios de Colab se facturan según el número de *unidades informáticas* consumidas [20]. El ritmo de consumo de las unidades informáticas varía en función de los recursos empleados en cada momento (cantidad de RAM, cantidad de VRAM, y consumo de CPU). Para entrenar el modelo de DED-AI, se espera un consumo aproximado de 12.5 unidades informáticas a la hora, durante un total de 150 horas, arrojando un consumo total aproximado de 1875 unidades informáticas.

Actualmente, no existe la posibilidad de adquirir unidades informáticas a un coste unitario, debiéndose comprar en paquetes de 500. En definitiva, el gasto en servicios se reduciría a la adquisición de 4 paquetes de 500 unidades informáticas, situación reflejada en la *Tabla 2.7*.

Recurso	Cantidad	Precio unitario (€)	Subtotal (€)
500 unidades informáticas Google Colab	4	51,99	207,96
TOTAL			207,96

Tabla 2.7: Costes totales de servicios

2.3.3. COSTES TOTALES

El coste total estimado resulta de la suma de cada uno de los costes desglosados en los dos apartados anteriores. Este asciende a un total de SESENTA Y CUATRO MIL CIENTO SESENTA Y NUEVE CON VENITISIETE EUROS (**64169,27€**).

Coste	Subtotal (€)
Recursos humanos	60304,72
Hardware	2361,59
Software	1295
Servicios	207,96
TOTAL	64169,27

Tabla 2.8: Costes totales del proyecto

En la *Figura 2.3* puede observarse la distribución de gastos, apreciándose que la partida destinada a recursos humanos es la más significativa, representando un 94% del coste total.

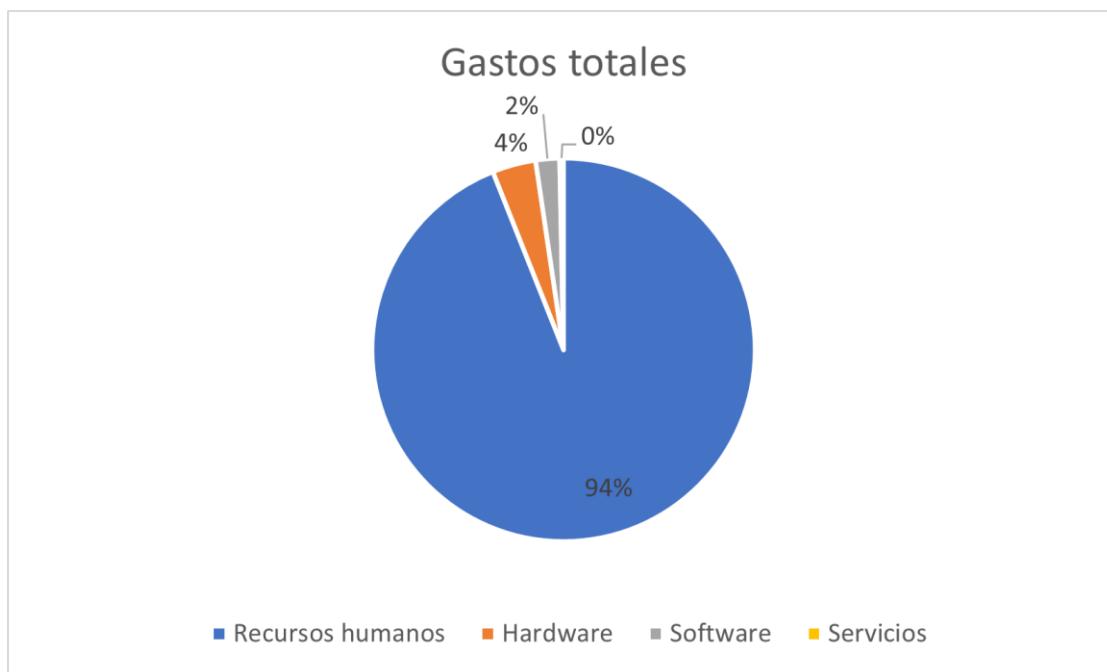


Figura 2.3: Gráfico de distribución de gastos totales

2.4. GESTIÓN DE RIESGOS

En este punto se identificarán los posibles riesgos que afectan al proyecto, incluyendo una estimación de su probabilidad e impacto, conforme a los criterios del Instituto de Ingeniería del Software (SEI), y su guía *Taxonomy-Based Risk Identification* [21].

2.4.1. IDENTIFICACIÓN DE RIESGOS

Los riesgos identificados de acuerdo a la taxonomía mencionada se dividirán en dos tipos, genéricos y específicos. Dentro de los primeros se agruparán todos los relacionados con fallos en la estimación o cambios en los requisitos, mientras que la segunda categoría se centra en los fallos técnicos.

2.4.1.1. RIESGOS GENÉRICOS

[RG-1]: Falta de presupuesto	
Categoría	Estimación
Precondición	El equipo de proyecto agota los recursos económicos disponibles.
Consecuencia	El producto no puede ser implementado en su totalidad, llevando, en el caso más favorable, a un aumento en el coste final así como un retraso en la entrega, y en el más desfavorable, a la cancelación del proyecto.

Tabla 2.9: Riesgo genérico 1 – Falta de presupuesto

[RG-2]: Modificación de requisitos	
Categoría	Requisitos
Precondición	El cliente modifica la especificación de requisitos en la fase de desarrollo.
Consecuencia	El tiempo de entrega final aumenta para cumplir con la nueva especificación. El presupuesto también podría verse afectado al alza.

Tabla 2.10: Riesgo genérico 2 – Modificación de requisitos

[RG-3]: Superación del tiempo de desarrollo estimado

Categoría	Estimación
Precondición	El equipo de proyecto incumple la planificación establecida.
Consecuencia	Los costes finales se incrementan al aumentar el número de horas trabajadas. El cliente no puede llevar a producción el producto en tiempo y forma.

Tabla 2.11: Riesgo genérico 3 – Superación del tiempo de desarrollo estimado

2.4.1.2. RIESGOS ESPECÍFICOS

[RE-1]: Brecha de seguridad	
Categoría	Técnico
Precondición	El producto presenta fallos que permiten el acceso o alteración no autorizados de la información manejada.
Consecuencia	Los datos quedan expuestos a individuos u organizaciones malintencionadas. Posibles contenciosos legales, y daño reputacional. Retraso en la entrega final, si las brechas se dan en dependencias de terceros.

Tabla 2.12: Riesgo específico 1 – Brecha de seguridad

2.4.2. ESTIMACIÓN DE PROBABILIDAD E IMPACTO

Expuestos los riesgos asociados al proyecto, se realiza una estimación de la probabilidad de que se presenten, así como del grado de impacto que causarían. En función de ambas estimaciones, se determinará la prioridad de cada riesgo.

2.4.2.1. PROBABILIDAD E IMPACTO

Para ambas métricas se emplea una escala de 0 a 1, donde 0 implica que el riesgo evaluado es completamente improbable o de nulo impacto, mientras que 1 significa que sucederá con total seguridad, o causará un impacto irreversible al proyecto. En la Tabla 2.12 se enumeran los riesgos, junto a sus evaluaciones correspondientes.

Riesgo	Probabilidad	Impacto
--------	--------------	---------

RG-1	0,2	0,7
RG-2	0,7	0,6
RG-3	0,3	0,6
RE-1	0,6	0,8

Tabla 2.13: Probabilidad e impacto para cada riesgo identificado

2.4.2.2. PRIORIDAD

Hallados los valores de probabilidad e impacto, se procede a establecer la prioridad con la que se debe atender para cada riesgo. Ésta es hallada mediante la fórmula:

$$\text{Prioridad} = \text{Probabilidad} * \text{Impacto} \quad (2.2)$$

Riesgo	Prioridad	Orden
RE-1	0,48	1
RG-2	0,41	2
RG-3	0,18	3
RG-1	0,14	4

Tabla 2.14: Prioridad para cada riesgo identificado

3. Solución

En este capítulo se detallará la solución desarrollada. En primer lugar se realizará una descripción a alto nivel de la funcionalidad del sistema, realizando posteriormente un análisis del que se extraerán los requisitos, tanto funcionales como no funcionales. Se enumerarán las tecnologías empleadas y se expondrá la arquitectura. También se tratarán asuntos de implementación, tales como la estructuración en componentes, y otros aspectos de relevancia en el código.

3.1. DESCRIPCIÓN DE LA SOLUCIÓN

Al emplear la solución, el usuario partirá de una aplicación móvil implementada en React Native [22]. Como control de acceso, se mostrará una pantalla de inicio de sesión. Esta autenticación se realizará mediante un usuario y una contraseña, siguiendo la especificación OAuth2 en su modalidad *password grant* [10]. Si el usuario introduce unas credenciales incorrectas, o caracteres no permitidos, se le advertirá, impidiéndosele continuar a la siguiente escena.

Tras iniciar sesión con su usuario y contraseña, el usuario encontrará la escena principal, en la que dispondrá de dos botones: uno para realizar una nueva fotografía, y otro para seleccionarla desde la memoria del dispositivo. Si se opta por la primera opción, se lanzará la aplicación cámara para realizar la captura. En caso de optar por la segunda, se abrirá un explorador de archivos o galería.

Cuando la captura haya sido realizada (o bien se haya seleccionado en memoria), ésta se mostrará en forma de previsualización, junto a un botón que permita confirmar la imagen. Si la imagen se rechaza, se regresará a la escena principal.

Si se confirma la imagen, ésta será enviada a un servidor API REST implementado en ExpressJS [23]. Éste debe obtener la probabilidad de que la imagen esté afectada por una retinopatía utilizando un modelo entrenado de aprendizaje profundo. La respuesta, que debe poder obtenerse en un tiempo inferior a 10s, será retornada al usuario. Así mismo, será persistida en base de datos, junto a un identificador único y asociada al usuario que realizó la petición.

Mientras se produzca el proceso descrito anteriormente, la aplicación móvil mostrará una escena de espera al usuario, indicando que la imagen se está procesando. En caso de que se produzca algún error en el proceso de análisis, se mostrará una nueva escena informando del fallo, y ofreciendo la posibilidad de reintentar el envío, o comenzar de cero desde la escena principal.

Si todo el procesamiento transcurre de forma correcta, la siguiente escena mostrará el resultado del análisis al usuario, indicando si la muestra es positiva en retinopatía o no. Un botón permitirá avanzar a la escena de *feedback*, donde se pedirá al usuario que indique si el algoritmo ha sido certero. Esta información se enviará de vuelta al servidor REST, quien la persistirá en base de datos, asociada a la imagen anteriormente registrada.

Finalmente, el usuario visualizará una escena de agradecimiento, junto a un botón para regresar a la escena principal.

3.2. ANÁLISIS DE LA SOLUCIÓN

En esta sección se describen los requisitos del proyecto, en base a la descripción anterior. Se incluyen tanto los requisitos funcionales como los no funcionales, y en definitiva, todos aquellos necesarios para que el producto final implemente el comportamiento deseado.

3.2.1. ESPECIFICACIÓN DE REQUISITOS FUNCIONALES

[RF-1]: Inicio de sesión	
Descripción	El sistema permitirá a los usuarios autenticarse mediante un usuario y una contraseña
Prioridad	Alta
Estado	Desarrollada

Tabla 3.1: Requisito funcional 1 – Inicio de sesión

[RF-2] Alerta de credenciales incorrectas	
Descripción	El sistema alertará a los usuarios si introducen credenciales incorrectas, o caracteres no permitidos

Prioridad	Alta
Estado	Desarrollada

Tabla 3.2: Requisito funcional 2 – Alerta de credenciales incorrectas

[RF-3]: Escena principal	
Descripción	El sistema mostrará una escena principal o <i>home</i> , donde podrá seleccionar una captura
Prioridad	Alta
Estado	Desarrollada

Tabla 3.3: Requisito funcional 3 – Escena principal

[RF-4]: Captura de nuevas imágenes	
Descripción	El sistema permitirá la toma de nuevas imágenes, utilizando la cámara del dispositivo
Prioridad	Alta
Estado	Desarrollada

Tabla 3.4: Requisito funcional 4 – Captura de nuevas imágenes

[RF-5]: Selección de imágenes en memoria	
Descripción	El sistema permitirá seleccionar imágenes desde la memoria del dispositivo, lanzando el explorador de archivos o galería.
Prioridad	Alta
Estado	Desarrollada

Tabla 3.5: Requisito funcional 5 – Selección de imágenes en memoria

[RF-6]: Confirmación de imágenes	
Descripción	El sistema mostrará una vista previa de la imagen seleccionada, permitiendo su confirmación por el usuario
Prioridad	Alta
Estado	Desarrollada

Tabla 3.6: Requisito funcional 6 – Confirmación de imágenes

[RF-7]: Regreso a la escena principal	
Descripción	El sistema siempre permitirá regresar a la escena principal
Prioridad	Alta
Estado	Desarrollada

Tabla 3.7: Requisito funcional 7 – Regreso a la escena principal

[RF-8]: Envío de imágenes	
Descripción	El sistema permitirá el envío de imágenes a un servidor de cara a su análisis
Prioridad	Alta
Estado	Desarrollada

Tabla 3.8: Requisito funcional 8 – Envío de imágenes

[RF-9]: Obtención de probabilidades	
Descripción	El sistema ofrecerá la probabilidad de que una imagen esté afectada por retinopatía, o no lo esté
Prioridad	Alta
Estado	Desarrollada

Tabla 3.9: Requisito funcional 9 – Obtención de probabilidades

[RF-10]: Persistencia de resultados	
Descripción	El sistema guardará los resultados del análisis de una imagen, junto a un identificador único
Prioridad	Alta
Estado	Desarrollada

Tabla 3.10: Requisito funcional 10 – Persistencia de resultados

[RF-11]: Información de errores	
Descripción	El sistema informará sobre errores en el proceso de análisis, si los hubiere
Prioridad	Alta
Estado	Desarrollada

Tabla 3.11: Requisito funcional 11 – Información de errores

[RF-12]: Escena de espera	
Descripción	El sistema mostrará una escena de espera mientras se procesa una imagen
Prioridad	Alta
Estado	Desarrollada

Tabla 3.12: Requisito funcional 12 – Escena de espera

[RF-13]: Visualización de resultados	
Descripción	El sistema mostrará al usuario el resultado de analizar una imagen, indicando si es positiva en retinopatía o no
Prioridad	Alta
Estado	Desarrollada

Tabla 3.13: Requisito funcional 13 – Visualización de resultados

[RF-14]: Feedback	
Descripción	El sistema permitirá a los usuarios indicar si el resultado de un análisis ha sido certero
Prioridad	Alta
Estado	Desarrollada

Tabla 3.14: Requisito funcional 14 – Feedback

[RF-15]: Persistencia de feedback	
Descripción	El sistema guardará las opiniones ofrecidas por los usuarios en el <i>feedback</i> , asociándolas a las imágenes analizadas
Prioridad	Alta
Estado	Desarrollada

Tabla 3.15: Requisito funcional 15 – Persistencia de feedback

[RF-16]: Escena de agradecimiento	
Descripción	El sistema agradecerá a los usuarios su utilización al finalizar el proceso completo de análisis
Prioridad	Alta
Estado	Desarrollada

Tabla 3.16: Requisito funcional 16 – Escena de agradecimiento

3.2.2. ESPECIFICACIÓN DE REQUISITOS NO FUNCIONALES

[RNF-1]: Tiempo de respuesta	
Descripción	El sistema debe responder en un tiempo razonable a una petición de análisis, no superando en ningún caso los 10 segundos
Prioridad	Alta
Estado	Desarrollada

Tabla 3.17: Requisito no funcional 1 – Tiempo de respuesta

[RNF-2]: Acceso al sistema	
Descripción	El sistema únicamente permitirá el acceso al servicio de análisis a los usuarios que presenten unas credenciales válidas
Prioridad	Alta
Estado	Desarrollada

Tabla 3.18: Requisito no funcional 2– Acceso al sistema

[RNF-3]: Autenticación mediante OAuth2 Password Grant	
Descripción	La autenticación de usuarios debe cumplir con las especificaciones de OAuth2, en su modalidad <i>password grant</i> .
Prioridad	Alta
Estado	Desarrollada

Tabla 3.19: Requisito no funcional 3 – Autenticación mediante OAuth2 Password Grant

[RNF-4]: Servidor API REST	
Descripción	El servidor debe cumplir con la arquitectura REST
Prioridad	Alta
Estado	Desarrollada

Tabla 3.20: Requisito no funcional 4 – Servidor API REST

[RNF-5]: Servidor implementado en Golang o ExpressJS	
Descripción	El servidor debe estar implementado usando, o bien el lenguaje Go, o bien el <i>framework</i> ExpressJS
Prioridad	Alta
Estado	Desarrollada

Tabla 3.21: Requisito no funcional 5 – Servidor implementado en Golang o ExpressJS

[RNF-5]: Aplicación móvil React Native	
Descripción	La aplicación móvil debe ser desarrollada empleando el <i>framework</i> React Native
Prioridad	Alta
Estado	Desarrollada

Tabla 3.22: Requisito no funcional 6 – Aplicación móvil React Native

[RNF-7]: Modelo de aprendizaje profundo	
Descripción	El modelo predictivo para el análisis de imágenes debe desarrollarse utilizando arquitecturas de aprendizaje profundo
Prioridad	Alta
Estado	Desarrollada

Tabla 3.23: Requisito no funcional 7 – Modelo de aprendizaje profundo

[RNF-8]: Accesibilidad	
Descripción	La interfaz de usuario del sistema será desarrollada conforme a patrones de diseño accesible
Prioridad	Media
Estado	Desarrollada

Tabla 3.24: Requisito no funcional 8 – Accesibilidad

3.3. TECNOLOGÍAS

La especificación de requisitos anterior establece el uso de varias tecnologías, tanto en la parte de desarrollo móvil como en la implementación del servidor. Entre ellas, destacan:

- **React Native:** adaptación a plataformas móviles del conocido *framework* de desarrollo de aplicaciones de una sola página React. Basado en Javascript, permite programar aplicaciones con una gran independencia de la plataforma destino, permitiendo por tanto, que sean compatibles tanto con iOS como Android [22].
- **Expo:** *framework* que extiende React Native, facilitando en gran medida la labor de desarrollo de aplicaciones universales, también para entornos web. Ha sido de especial utilidad **Expo Go**, complemento que permite probar desarrollos basados en Expo a través de un servidor web, sin necesidad de recurrir a un emulador [24].
- **React Native Paper:** librería de componentes para construir interfaces de usuario basadas en el lenguaje de diseño Material Design de Google [25].
- **NodeJS:** entorno de ejecución JavaScript fuera de navegador. Altamente utilizado como base para construir servidores web. Así mismo, su gestor de paquetes, **npm**, es utilizado para la gestión de dependencias en proyectos React Native y Express [26].
- **ExpressJS:** marco de trabajo para NodeJS que facilita la construcción de servidores RESTful [23].
- **Axios:** cliente HTTP basado en promesas para NodeJS. Imprescindible para coordinar las acciones entre la aplicación móvil y el servidor, que consumirá un tiempo determinado hasta ofrecer respuesta [27].
- **Express-OpenAPI:** *framework* que permite construir servicios web que cumplan con la especificación OpenAPI, anteriormente conocida como swagger. Un servicio web implementado en base a OpenAPI permite la generación automática de documentación sobre métodos y parámetros [28].
- **OAuth2Server:** módulo para NodeJS que permite implementar un servidor de autenticación de acuerdo con la especificación OAuth2. Soporta diversos flujos de autenticación, incluyendo el *password grant* utilizado en este proyecto [29].
- **Sendgrid:** proveedor para el envío de correos electrónicos, más concretamente utilizado para confirmar el registro de un nuevo usuario [30].
- **Flask:** marco de trabajo minimalista que permite crear servidores web en lenguaje Python. En él se ha desarrollado un segundo servidor, que hospeda y ejecuta el modelo de aprendizaje automático [31].

- **MongoDB:** base de datos no relacional, basada en documentos, utilizada para persistir tanto la información relacionada con los análisis de imágenes, como las credenciales de los usuarios. Para facilitar el intercambio de información entre el servidor REST y la base de datos, se ha utilizado el ORM **mongoose** [32] [33].

En el proceso de elaboración del modelo de inteligencia artificial, cabe destacar las siguientes tecnologías:

- **PyTorch:** librería de código abierto para Python, utilizada para construir, entrenar y validar modelos de aprendizaje profundo. Permite el cómputo de tensores utilizando unidades de procesamiento gráfico (GPUs) [34].
- **Pandas:** librería para la lectura y manipulación de datos. Utilizada para la limpieza y balanceado del *dataset* empleado, así como para la creación de los *datasets* de entrenamiento y validación [35].
- **Segmentation Models:** librería que implementa en PyTorch varias arquitecturas de redes neuronales para la segmentación de imágenes, incluyendo la arquitectura UNet utilizada en este proyecto. También ofrece métodos para el cálculo de diversas métricas [36].
- **OpenCV:** librería de visión artificial. Abarca una amplia variedad de casos de uso, desde el reconocimiento facial, de personas u objetos, hasta diversos algoritmos de manipulación imágenes. En el presente proyecto, se ha empleado para el preprocesado de las capturas de fondo de ojo [37].
- **Google Colab:** entorno de ejecución de Jupyter Notebooks que utiliza los recursos computacionales de Google Cloud. Ha sido empleado para entrenar el modelo, dada la carencia de una GPU lo suficientemente potente en mi entorno de trabajo [38].

3.4. ARQUITECTURA

La solución desarrollada sigue un modelo de arquitectura cliente-servidor con una serie de componentes diferenciados, tal y como se aprecia en la Figura 3.1. Del lado del cliente, se cuenta con una aplicación móvil, desarrollada en React Native. La aplicación cuenta con una serie de vistas (*views*), las cuales representan las distintas escenas por las que puede

navegar el usuario. La navegación por las vistas se controla mediante el *router*, que apila el historial del usuario.

En algunas vistas, como en las que se envía una imagen para analizar o se inicia sesión, el cliente se comunica con un servidor REST, programado en ExpressJS. Esta conexión se desarrolla mediante el cliente *axios*. REST es una arquitectura que permite a dos aplicaciones interactuar entre sí utilizando los métodos propios del protocolo HTTP, tales como POST, GET, PUT y DELETE, con total independencia de aspectos relacionados con la implementación.

El servidor Express puede tener que recuperar información de la base de datos, la cual ha sido implementada usando MongoDB. Cada usuario o imagen es almacenado como un documento en formato BSON, con una serie de atributos. Estos documentos son validados y mapeados mediante el ODM *mongoose*.

Finalmente, para cumplir la funcionalidad básica de la aplicación, la cual es analizar si una imagen posee algún tipo de retinopatía, se ha optado por hospedar el modelo de aprendizaje automático en un servidor separado, desarrollado mediante *Flask*. Este servidor también cumple con la arquitectura REST, y se comunica con el servidor principal Express cuando este último lo requiere, de nuevo, utilizando un cliente *axios*. La Figura 3.1 refleja la totalidad de la arquitectura expuesta.

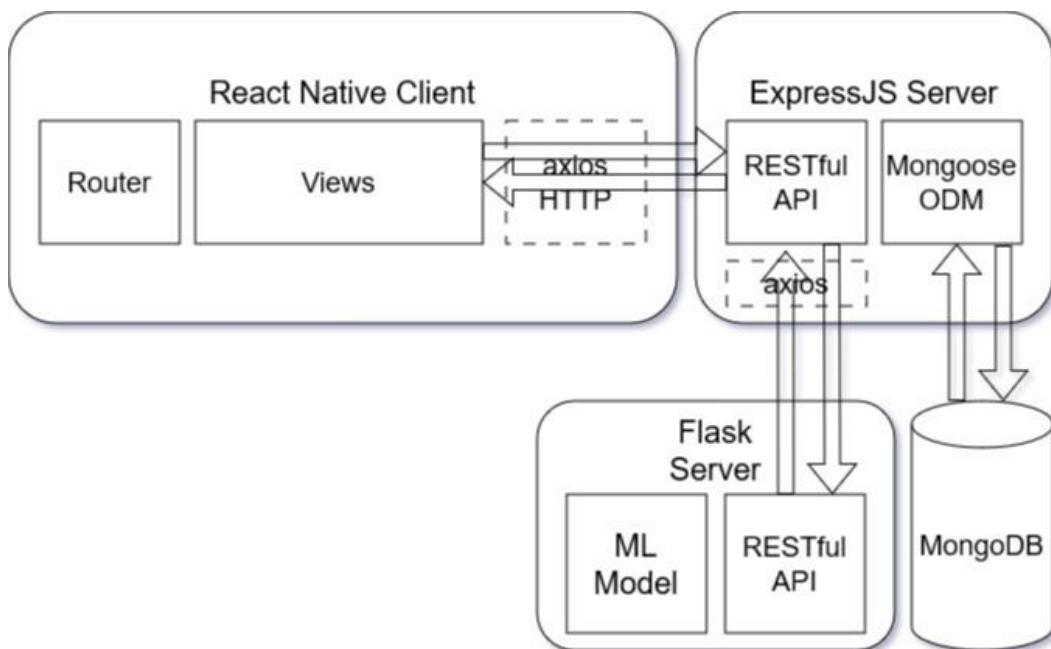


Figura 3.1: Arquitectura de la solución desarrollada (Elaboración propia)

3.5. DISEÑO

Presentada la arquitectura en el apartado anterior, a continuación se ofrece un análisis pormenorizado de los componentes descritos. Además de ofrecer información sobre los dos servidores, el cliente móvil, y la base de datos, también se expondrán varios aspectos relacionados con el modelo de aprendizaje automático, tales como la limpieza de datos, arquitectura de la red neuronal, o parámetros de entrenamiento, entre otros detalles.

3.5.1. DISEÑO DE LA APLICACIÓN MÓVIL

La aplicación móvil consta de un total de ocho escenas que cubren el flujo de navegación y uso descrito en el punto 3.1. Éstas son *Login*, *Welcome*, *Preview*, *Loading*, *Error*, *Results*, *Feedback* y *Thanks*.

3.5.1.1. LOGIN

Al abrir la aplicación móvil, el usuario debe iniciar sesión, mediante su usuario y contraseña, en la escena *Login*. En ella pueden apreciarse un título descriptivo, dos cuadros de entrada para introducir los datos anteriormente mencionados, y un botón de *siguiente*, el cual acciona el envío de las credenciales al servidor, para comprobar su existencia en base de datos. Si la autenticación se realiza con éxito, se mostrará la vista *Welcome*, tras mostrar previamente la vista de espera *Loading*. Si se da algún fallo de comunicación, o bien las credenciales son incorrectas, se muestra la vista *Error*. En la Figura 3.2 se muestra una captura de pantalla de la escena.



Figura 3.2: Vista Login

3.5.1.2. WELCOME

Referida a lo largo de la descripción de la solución como *escena principal*, es el punto de partida donde, tras iniciar sesión, el usuario puede comenzar un nuevo análisis. Al título que ya tenía la vista anterior se une un breve párrafo de ayuda donde se indica al usuario qué puede hacer: o bien elegir una captura desde la memoria del dispositivo, o realizar una nueva fotografía. A continuación, se ofrecen dos botones que accionan la funcionalidad descrita (ver Figura 3.3).

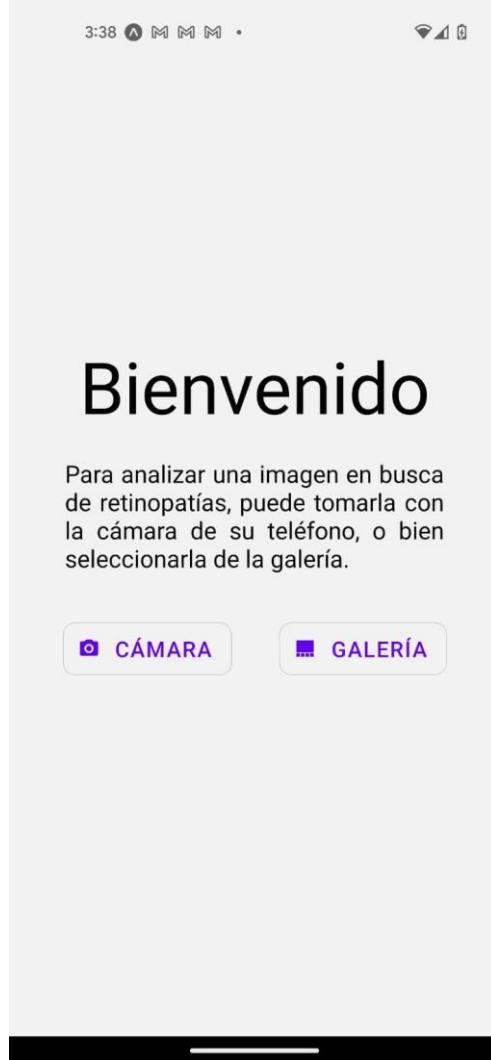


Figura 3.3: Vista Welcome

3.5.1.3. PREVIEW

Seleccionada la imagen desde el método de preferencia, ésta se muestra en un cuadro de vista previa, de dimensiones 350x350. El título de la vista se sitúa sobre dicho recuadro. Debajo, dos botones identificados en verde y rojo permiten al usuario confirmar o descartar la imagen. Si la imagen es descartada, se regresa a la escena *Welcome*, mientras que si se confirma, se ordena el envío de la imagen al servidor para su procesamiento, mostrándose mientras tanto la escena *Loading*. La Figura 3.4 ilustra la vista *Preview*.



Figura 3.4: Vista Preview

3.5.1.4. LOADING

Escena de espera mostrada al usuario cuando se está a la espera de una respuesta por parte del servidor. Puede ser visualizada en tres ocasiones: tras un envío de credenciales para iniciar sesión, mientras se espera el resultado de analizar una imagen, o cuando se remite el *feedback* para su almacenamiento en base de datos. Consta de un círculo de carga animado, y el texto “*Procesando. Espere, por favor*” a su pie inferior (ver Figura 3.5).



Figura 3.5: Vista Loading

3.5.1.5. ERROR

Escena mostrada al ocurrir un error de comunicación del servidor, o un envío de datos incorrectos. Concretamente, esta escena puede aparecer tras fallar el inicio de sesión, el envío de una imagen, o la remisión del *feedback*. Encabezada por un título, ofrece un párrafo descriptivo, variable según dónde y cómo se ha producido el fallo, así como dos botones al pie de esta descripción: *Reintentar*, para repetir la acción que ha ocasionado el fallo, y *Volver*, que permite ir a la escena *Welcome* para comenzar de cero el proceso. En la Figura 3.6 se muestra una captura de la vista *Error*.

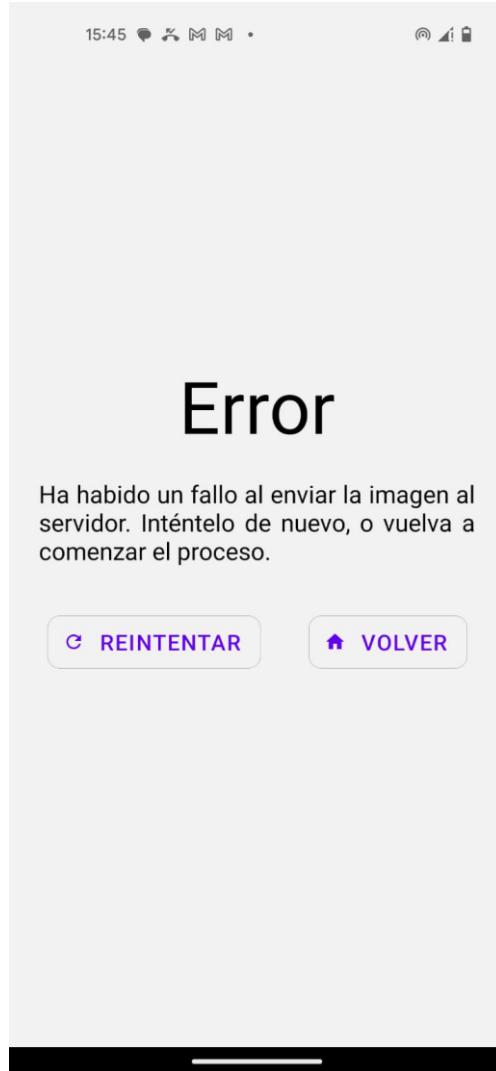


Figura 3.6: Vista Error

3.5.1.6. RESULTS

Tras un procesamiento correcto de la imagen, los resultados del análisis son retornados a la aplicación. La escena *Results* está encabezada por un título variable según el resultado devuelto: *Positivo*, en caso de que se haya detectado alguna retinopatía en la muestra, y *Negativo*, si la imagen no presenta anomalías detectadas. Un texto en párrafo muestra el identificador asociado a la imagen, y un botón en la parte inferior permite avanzar hacia la escena *Feedback*. La Figura 3.7 muestra los distintos elementos de la vista *Results*.

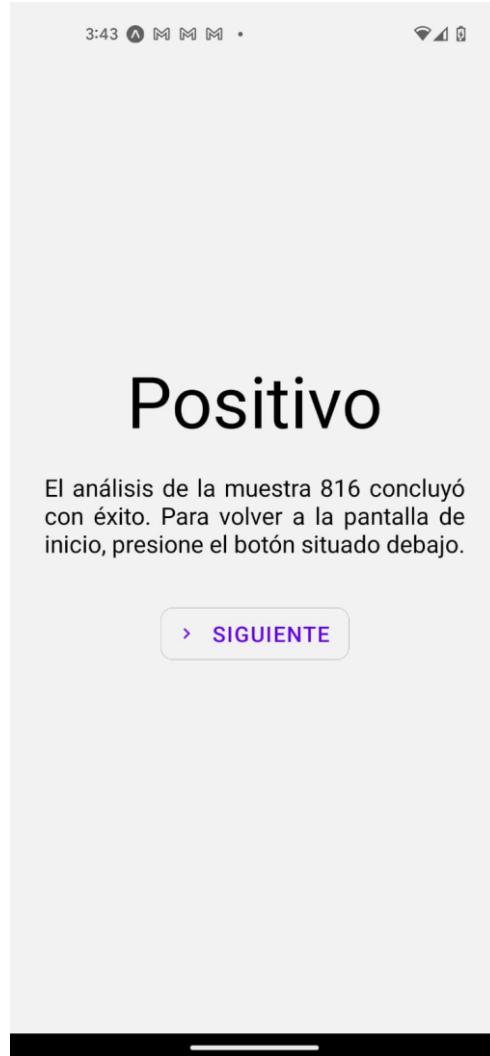


Figura 3.7: Vista Results

3.5.1.7. FEEDBACK

Mostrado el resultado del análisis, el usuario puede indicar si éste coincide con el diagnóstico realizado por un profesional cualificado. Debajo del título de la escena se sitúa un párrafo que formula la pregunta, y dos botones, en colores verde y rojo: *sí*, cuando se ha ofrecido un resultado coincidente, y *no*, en caso contrario. Cualquiera de los dos botones acciona el envío de la retroalimentación al servidor, dándose paso posteriormente a la escena *Thanks*, o a *Error*, si se produce algún fallo en la conexión. La Figura 3.8 ofrece una captura de pantalla de la vista *Feedback*.



Fuente 3.8: Vista Feedback

3.5.1.8. THANKS

Si todo el flujo de la aplicación, desde que se inicia sesión, hasta que se envía el *feedback*, transcurre correctamente, se muestra la escena de agradecimiento *Thanks*. Un título acompaña a un párrafo donde se agradece al usuario el uso de la aplicación, y un botón le permite regresar a la escena *Welcome* para realizar otro análisis (ver Figura 3.9).

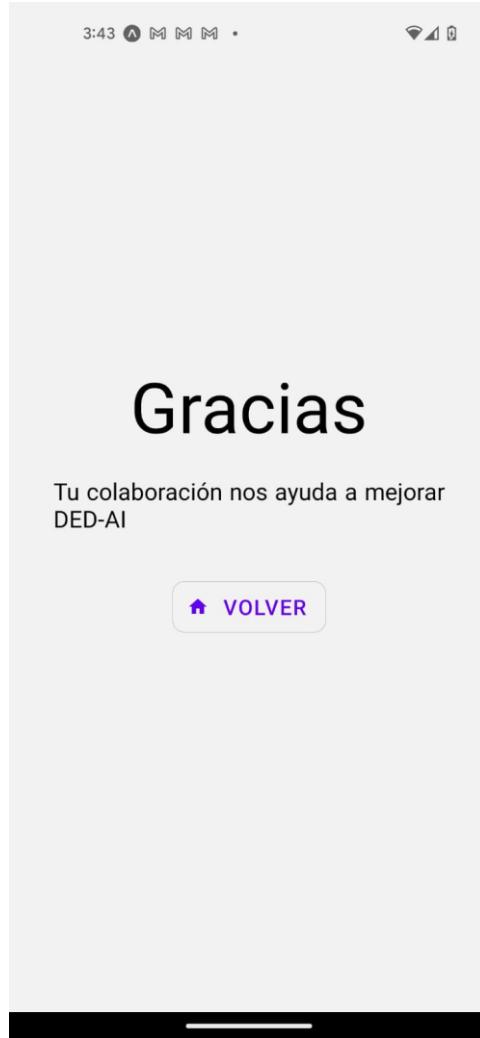


Figura 3.9: Vista Thanks

3.5.2. DISEÑO DEL SERVIDOR REST EXPRESS

Siempre que un usuario necesita autenticarse, remitir una imagen para analizar, o enviar *feedback*, la aplicación móvil establece comunicación con el servidor Express. Éste ha sido diseñado de acuerdo con la especificación OpenAPI, la cual tiene como objetivo desarrollar APIs HTTP comprensibles tanto para humanos como máquinas.

Toda API diseñada siguiendo la especificación debe definir un esquema, donde se recoge diversa metainformación, incluyendo las rutas, requerimientos de seguridad, o definición de los objetos recibidos y retornados por la API. A continuación, se detallan los parámetros definidos en el esquema.

3.5.2.1. PARÁMETROS GENERALES

Se incluyen aquí los campos obligatorios en todo esquema OpenAPI, como la versión de la especificación empleada, el directorio raíz, así como el título y descripción de la API.

Parámetros Generales		
Campo	Valor	Descripción
<i>swagger</i>	“2.0”	Versión de la especificación OpenAPI empleada
<i>basepath</i>	“/”	Directorio raíz
<i>info</i>	<i>title</i>	DED_AI API
	<i>version</i>	1.0.0
		Versión de la API

Tabla 3.25: Parámetros generales de la API REST

3.5.2.2. PARÁMETROS DE SEGURIDAD

Incluye todos los parámetros que rigen el acceso a los puntos de acceso de la API. En este caso particular, solo se cuenta con un único parámetro, *passwordAuthentication*, objeto que define el flujo de autenticación por usuario y contraseña.

Parámetros de Seguridad		
Campo	Valor	Descripción
<i>passwordAuthentication</i>	<i>Type</i>	“ouath2”
	<i>flow</i>	“password”
	<i>tokenUrl</i>	“https://ded-ai-api-
		URL que

		server.onrender.com/auth"	proporciona un token de sesión
	<i>scopes</i>	[“read”, “write”]	Privilegios de acceso una vez autenticado

Tabla 3.26: Parámetros de seguridad de la API REST

3.5.2.3. OBJETOS

Definen los datos que pueden ser recibidos y retornados por los métodos de la API. DED_AI API utiliza 4 tipos de objetos: *Image*, *Sample*, *User* y *Token*.

Image			
Propiedad	Tipo	Formato	Requerida
<i>raw</i>	String	Binario	Sí

Tabla 3.27: Ficha del objeto *Image*

Sample		
Propiedad	Tipo	Requerida
<i>_id</i>	String	Sí
<i>fileName</i>	String	Sí
<i>analysisResult</i>	String	Sí
<i>analysisDate</i>	String	Sí
<i>feedbackResult</i>	String	No

Tabla 3.28: Ficha del objeto *Sample*

User		
Propiedad	Tipo	Requerida
<i>_id</i>	String	No
<i>username</i>	String	Sí
<i>password</i>	String	Sí

<i>email</i>	String	No
--------------	--------	----

Tabla 3.29: Ficha del objeto User

Token			
Propiedad	Tipo	Formato	Requerida
<i>accessToken</i>	String		Sí
<i>accessTokenExpiresAt</i>	String	Date	Sí
<i>client</i>	Objeto		Sí
<i>user</i>	Objeto		Sí

Tabla 3.30: Ficha del objeto Token

3.5.2.4. RUTAS

Cada ruta, definida por una URI, define uno o varios de los puntos de comunicación o *endpoints* que pueden ser llamados por el cliente. Se definen tres rutas: *analysis*, *auth* y *register*.

/AUTH

Ruta empleada para el inicio de sesión, únicamente define un método POST, al cual se envían el usuario y contraseña, encapsulados en un objeto *User*. Si la autenticación tiene éxito, retorna un token de sesión.

POST /auth	
Descripción	POST Login Credentials
OperationId	postLogin
Consume	application/json
Parámetros	<i>User</i> en <i>body</i>
Respuestas	200 OK Autenticación correcta. Se devuelve un objeto <i>Token</i> .

	401 Unauthorized	Credenciales incorrectas.
	400 Bad Request	Petición mal formada.
	500 Internal Server Error	Fallo en el procesamiento.

Tabla 3.31: Ficha del método POST /auth

/ANALYSIS

Bajo esta ruta se definen un método POST y otro PUT. El primero de ellos es empleado al enviar una imagen desde el dispositivo, la cual se encapsulará en un objeto *Image*. Al concluirse el análisis, se retorna un objeto *Sample*.

POST /analysis		
Descripción	POST Image	
OperationId	postImage	
Consume	application/json	
Autenticación requerida	Sí, <i>passwordAuthentication</i>	
Parámetros	<i>Image</i> en body	
Respuestas	200 OK 400 Bad Request 401 Unauthorized 500 Internal Server Error	Análisis completado con éxito. Se devuelve un objeto <i>Sample</i> . Petición mal formada. Autenticación requerida. Fallo en el procesamiento.

Tabla 3.32: Ficha del método POST /analysis

El objeto Sample es reutilizado por el método PUT. Éste se emplea para insertar el *feedback* del usuario en base de datos, retornándose el objeto actualizado.

PUT /analysis

Descripción	PUT Feedback	
OperationId	putFeedback	
Consume	application/json	
Autenticación requerida	Sí, <i>passwordAuthentication</i>	
Parámetros	<i>Sample</i> en <i>body</i>	
Respuestas	200 OK	Feedback insertado. Se devuelve el objeto <i>Sample</i> actualizado.
	400 Bad Request	Petición mal formada.
	401 Unauthorized	Autenticación requerida.
	500 Internal Server Error	Fallo en el procesamiento.

Tabla 3.33: Ficha del método PUT /analysis

/REGISTER

Ruta reservada para la creación de nuevos usuarios en el sistema. Se define un único método POST, el cual recibe un objeto *User* completo, incluyendo nombre de usuario, contraseña y dirección email.

POST /register		
Descripción	POST Register Data	
OperationId	postRegister	
Consume	application/json	
Autenticación requerida	No	
Parámetros	<i>User</i> en <i>body</i>	
Respuestas	200 OK	Registro completado con éxito.
	400 Bad Request	Petición mal formada.

	500 Internal Server Error	Fallo en el procesamiento.
--	---------------------------	----------------------------

Tabla 3.34: Ficha del método POST /register

3.5.3. DISEÑO DE LA BASE DE DATOS

La base de datos ha sido diseñada utilizando un modelo no relacional, basado en documentos, utilizando MongoDB. Estos documentos se agrupan en colecciones, las cuales son construidas siguiendo un esquema. Este esquema recoge los distintos atributos del documento BSON, así como los tipos de estos argumentos, su obligatoriedad, unicidad, y otro tipo de restricciones en los valores.

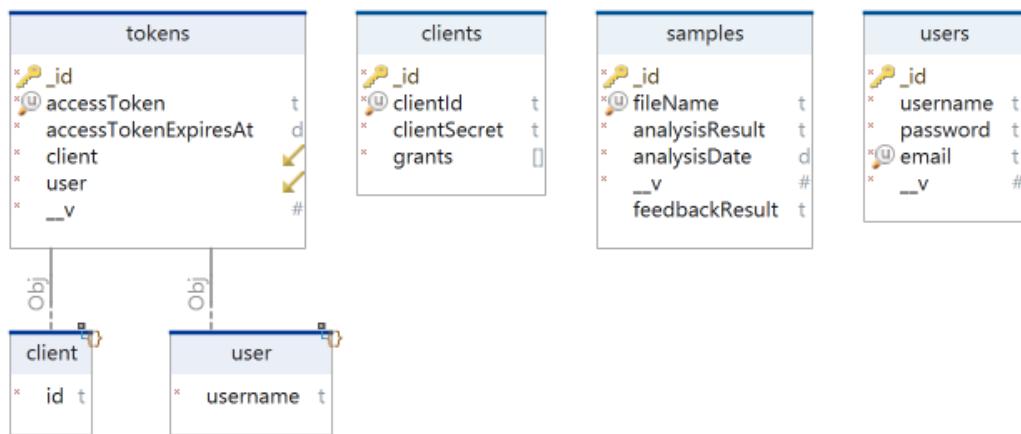


Figura 3.10: Diagrama de colecciones de la base de datos

La Figura 3.10 muestra más claramente la configuración de las distintas colecciones, así como los atributos de los documentos. De igual forma, a continuación se ofrece una explicación más detallada de cada colección.

3.5.3.1. COLECCIÓN TOKENS

Representación en base de datos de los tokens de sesión generados por el servidor OAuth2.

Un documento *Token* contiene los siguientes atributos:

- **_id**: identificador único del documento, asociado por MongoDB.
- **accessToken**: clave única hexadecimal, representando la autorización de acceso concedida.
- **accessTokenExpiresAt**: fecha en la cual la autorización deja de tener validez.
- **Client**: objeto que representa el cliente (aplicación) que ha realizado la autenticación. Dicho objeto contiene el ID del cliente, el cual debe haberse registrado previamente en la base de datos.
- **User**: objeto que representa al usuario que se ha autenticado. Este objeto, al igual que en el caso anterior, contiene el ID en base de datos del usuario.
- **_v**: versión del documento, con contador inicial en 0. Atributo creado por la base de datos.

Se ofrece a continuación una tabla resumen de las características de cada atributo:

Token			
Atributo	Tipo	Único	Requerido
<i>_id</i>	ObjectId	Sí	Sí
<i>accessToken</i>	Strn	Sí	Sí
<i>accessTokenExpiresAt</i>	Date	No	Sí
<i>client</i>	Schema.Types.Mixed	No	Sí
<i>user</i>	Schema.Types.Mixed	No	Sí
<i>_v</i>	Int32	No	Sí

Tabla 3.35: Resumen del documento Token

3.5.3.2. COLECCIÓN CLIENTS

Almacena los clientes (aplicaciones) autorizados a acceder a la funcionalidad ofrecida por el servidor. Conta de los siguientes atributos:

- **_id**: identificador único del documento.
- **clientId**: cadena de caracteres que identifica al cliente, escogida por el desarrollador.

- **clientSecret**: contraseña requerida para confirmar la identidad del cliente, únicamente conocida por ambos extremos de la comunicación.
- **grants**: array de permisos autorizados a la aplicación.
- **redirectUris**: direcciones a las que se debe dirigir la navegación tras un proceso de autenticación exitoso.

Se muestra a continuación un cuadro resumen de los atributos descritos:

Client			
Atributo	Tipo	Único	Requerido
<code>_id</code>	ObjectId	Sí	Sí
<code>clientId</code>	String	Sí	Sí
<code>clientSecret</code>	String	No	Sí
<code>grants</code>	Array	No	Sí
<code>redirectUris</code>	Array	No	No

Tabla 3.36: Resumen del documento Client

3.5.3.3. COLECCIÓN SAMPLES

Un documento *Sample* es creado para almacenar los resultados de un análisis, así como el *feedback* ofrecido por el usuario. Puede contener los siguientes atributos:

- **_id**: identificador único asociado por MongoDB.
- **fileName**: nombre del fichero subido al servidor, como cadena de caracteres.
- **analysisResult**: booleano que indica si la imagen ha sido marcada como positiva en retinopatía, o negativa.
- **analysisDate**: fecha en la que se realizó el análisis, codificada de acuerdo con el objeto Date de JavaScript (ISO 8601 [39]).
- **feedbackResult**: almacena la retroalimentación del usuario. *True* si el resultado ofrecido coincide con el diagnóstico profesional, falso en caso contrario.
- **userId**: referencia al identificador en base de datos del usuario que solicitó el análisis.
- **_v**: versión del documento, autogenerada por la base de datos.

En el siguiente cuadro resumen se recogen los distintos atributos del documento *Sample*:

Sample			
Atributo	Tipo	Único	Requerido
<i>_id</i>	ObjectId	Sí	Sí
<i>fileName</i>	String	Sí	Sí
<i>analysisResult</i>	Boolean	No	Sí
<i>analysisDate</i>	Date	No	Sí
<i>feedbackResult</i>	Boolean	No	No
<i>userId</i>	ObjectId	No	Sí

Tabla 3.37: Resumen del documento Sample

3.5.3.4. COLECCIÓN USERS

Almacena cada uno de los usuarios de la aplicación, incluyendo sus credenciales y email.

Contiene los siguientes atributos:

- ***_id***: identificador único creado por MongoDB.
- ***username***: nombre de usuario, único y en formato cadena de caracteres. Debe almacenarse correctamente formateado, sin espacios.
- ***password***: contraseña, almacenada como hash.
- ***email***: dirección de correo electrónico, almacenada como String. Debe ser única para cada usuario, almacenarse sin espacios y en minúsculas.

A continuación se ofrece un cuadro resumen de los atributos de *User*:

User					
Atributo	Tipo	Único	Requerido	Sin espacios	Minúsculas
<i>_id</i>	ObjectId	Sí	Sí	Sí	No
<i>username</i>	String	Sí	Sí	Sí	No
<i>password</i>	String	No	Sí	No	No
<i>email</i>	String	Sí	Sí	Sí	Sí

Tabla 3.38: Resumen del documento User

3.5.4. DISEÑO DEL MODELO DE APRENDIZAJE AUTOMÁTICO

El modelo de detección de retinopatías es la pieza imprescindible para ofrecer la funcionalidad del producto. Este desarrollo tiene una naturaleza distinta a la del resto de artefactos software descritos anteriormente. Por ello, en este apartado se comenzará detallando la muestra de datos empleada y la limpieza y preprocesamiento de éstos. Después, se analizará la arquitectura de red neuronal empleada, y finalmente, las funciones que permiten evaluar y optimizar el entrenamiento.

3.5.4.1. DESCRIPCIÓN DEL DATASET

La muestra de datos utilizada se ha extraído de la comunidad de aprendizaje automático Kaggle [40]. En esta plataforma, numerosas instituciones y empresas organizan competiciones para mejorar la calidad de sus modelos de inteligencia artificial. En el 2015, la *California Health Foundation* [41] y la compañía de exámenes clínicos *EyePACS* [42] abrieron un concurso para desarrollar algoritmos que permitieran detectar la retinopatía diabética. Para ello, pusieron a disposición un total de 88702 imágenes de fondo de ojo a color, pertenecientes a pacientes tanto positivos como negativos en retinopatía. De ellas, 35108 correspondían a la muestra de entrenamiento, y 53594 a la muestra de *test*. Así mismo, se ofreció un archivo de valores separados por comas (CSV) con las etiquetas de cada imagen. El etiquetado sigue los grados de retinopatía mostrados en la *Tabla 1.1*. La *Figura 3.11* muestra varias imágenes pertenecientes al conjunto.

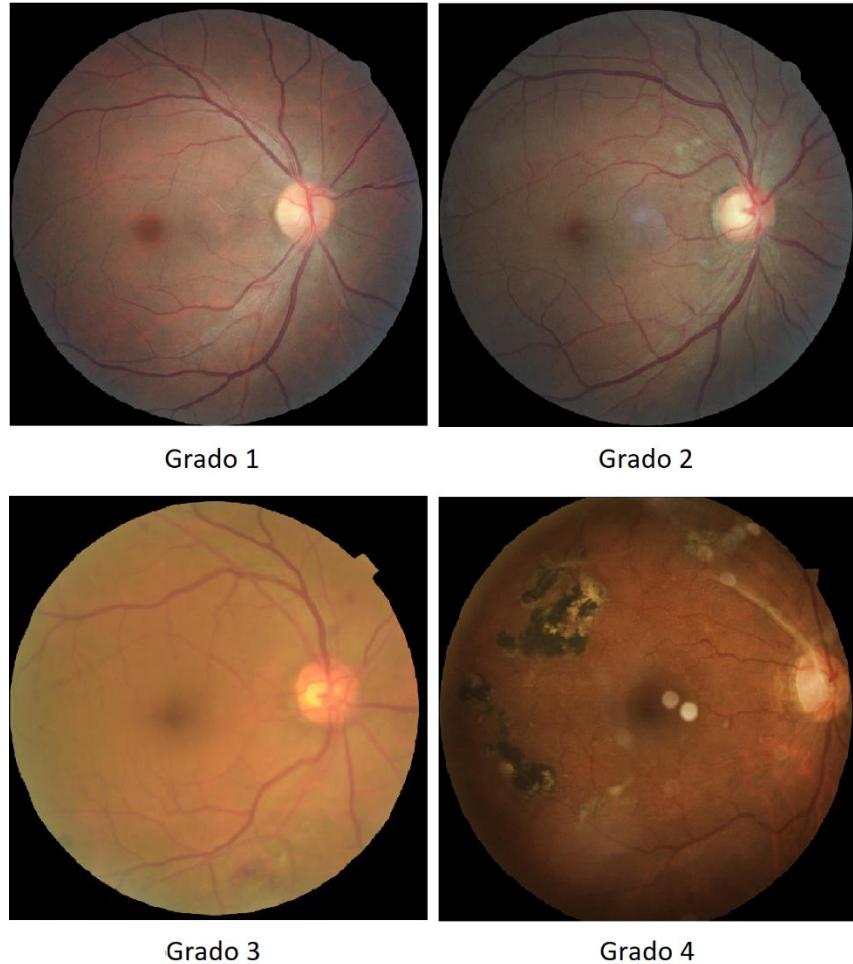


Figura 3.11: Varias fotografías pertenecientes a la muestra

3.5.4.2. LIMPIEZA Y DIVISIÓN

Anteriormente se ofrecieron detalles sobre el tamaño de la muestra, sin embargo, no se analizó si la misma estaba equilibrada, o la distribución de las muestras según cada grado de retinopatía. De forma previa a cualquier proceso, se ofrece a continuación la *Tabla 3.39*, que muestra las distintas clases de retinopatías, así como el número de capturas asociadas a dicho grado.

Grado	%	Recuento
0	73,49	25802
1	6,95	2438
2	15,06	5288

3	2,48	872
4	2,02	708

Tabla 3.39: Recuento del dataset original según grado de retinopatía

Puede comprobarse que el número de muestras que presentan algún tipo de retinopatía (etiquetas 1, 2, 3 y 4) es sensiblemente inferior a las muestras negativas. Por tanto, se hace necesario equilibrar el número de entradas positivas y negativas para optimizar el resultado del entrenamiento.

Es aquí donde se aplica la primera operación. Se obtiene el número de muestras positivas, y se calcula su diferencia con el número de negativos. Esta diferencia es el sobrante a eliminar, descartando tantas muestras negativas como se haya determinado. Así, se obtiene un subconjunto donde positivos y negativos mantienen una proporción 50%-50%.

Grado	%	Recuento
0	50	9306
1	13,10	2438
2	28,41	5288
3	4,69	872
4	3,80	708

Tabla 3.40: Recuento del dataset tras balancear

Balanceado el conjunto de datos, se procede a su división en dos partes, reservando el 85% de las muestras para el entrenamiento y validación, y el 15% restante para *test*. Los dos subconjuntos resultantes presentan el reparto observable en la *Tabla 3.41* y *Tabla 3.42*.

Grado	%	Recuento
0	50,02	7914
1	13,07	2067
2	28,45	4501
3	4,65	735

Grado	%	Recuento
0	49,85	1392
1	13,29	371
2	28,19	787
3	4,91	137

4	3,81	603
----------	------	-----

Tabla 3.41: Recuento entrenamiento

4	3,76	105
----------	------	-----

Tabla 3.42: Recuento test

De cara a evaluar la calidad del modelo frente a cada tipo de retinopatía, el conjunto de datos de test es partido en cuatro, consiguiendo un subconjunto separado para cada grado positivo. Estos subconjuntos son equilibrados siguiendo la misma metodología empleada al inicio de este capítulo.

Grado	%	Recuento
0	50	371
1	50	371

Tabla 3.43: Recuento test grado 1

Grado	%	Recuento
0	50	787
2	50	787

Tabla 3.44: Recuento test grado 2

Grado	%	Recuento
0	50	137
3	50	137

Tabla 3.45: Recuento test grado 3

Grado	%	Recuento
0	50	105
4	50	105

Tabla 3.46: Recuento test grado 4

3.5.4.3. PREPROCESADO

Al cerrarse el plazo de participación en el concurso, el *dataset* completo no está disponible en Kaggle. Sin embargo, la usuaria *Maria Herrero* subió posteriormente el conjunto de datos de entrenamiento [43]. En este nuevo conjunto, la resolución de las imágenes fue recortada a 1024x1024 píxeles, y también se centraron las capturas para eliminar la mayoría de los espacios en blanco.

Ya en el proyecto, se realizó otro preprocesamiento a mayores. Como se puede extraer del apartado 1.1. *Definición del problema*, muchos de los síntomas de una retinopatía diabética se manifiestan en forma de alteraciones en los capilares sanguíneos, incluyendo la formación de nuevos capilares en retinopatías de grado 4. Por ello, es necesario aplicar filtros de imagen que resalten la vascularización de la retina, así como eliminan otra serie de características no relevantes para estudiar la fotografía.

DESEFOQUE GAUSSIANO

Para comprender el funcionamiento de una capa de desenfoque gaussiano, en primer lugar es necesario entender la estructura de una imagen digital, así como el concepto de convolución. Una imagen está compuesta de píxeles, que representan la unidad más pequeña de información de la fotografía. Cada pixel puede ser concebido como un arreglo $[x, y, [R, G, B]]$, donde x es la ubicación del pixel en anchura (o en el eje horizontal), y la ubicación en altura (o en el eje vertical), mientras que $[R, G, B]$ es otro arreglo utilizado para representar el color, mediante la composición de los colores primarios rojo, verde y azul.

Una convolución es el resultado de tomar un fragmento de fotografía, al cual se denominará matriz A , y multiplicarlo por otra matriz de idéntica dimensión, llamada *kernel*, obteniendo un nuevo píxel que vendrá influido por los valores de los píxeles circundantes, como puede observarse en la *Figura 3.12*.

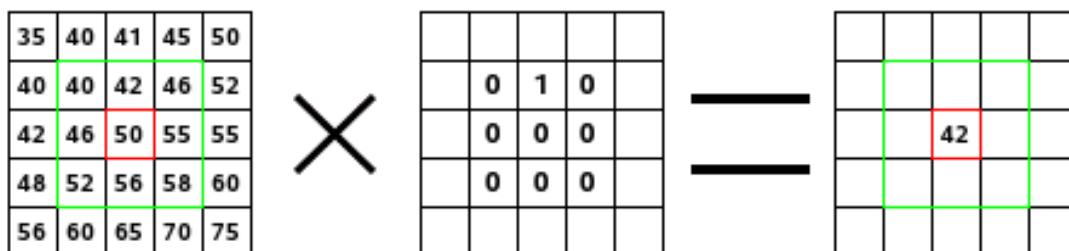


Figura 3.12: transformación de un pixel mediante convolución (Fuente: GIMP [44])

La selección de los valores de la matriz *kernel*, así como su dimensión, determinará el resultado del filtro. En el caso expuesto anteriormente, para calcular el píxel marcado en rojo, solo intervendrá aquel situado inmediatamente encima, pues el resto de los valores de la matriz *kernel* son iguales a 0.

Así pues, un filtro gaussiano es aquel que calcula los valores de la matriz *kernel* aplicando una distribución normal o gaussiana. La *campana de Gauss*, ilustrada en la *Figura 3.13*, es una de las distribuciones más conocidas en estadística, donde, estudiado un fenómeno

determinado, los valores situados a la mitad de rango son los que aparecen con mayor frecuencia.

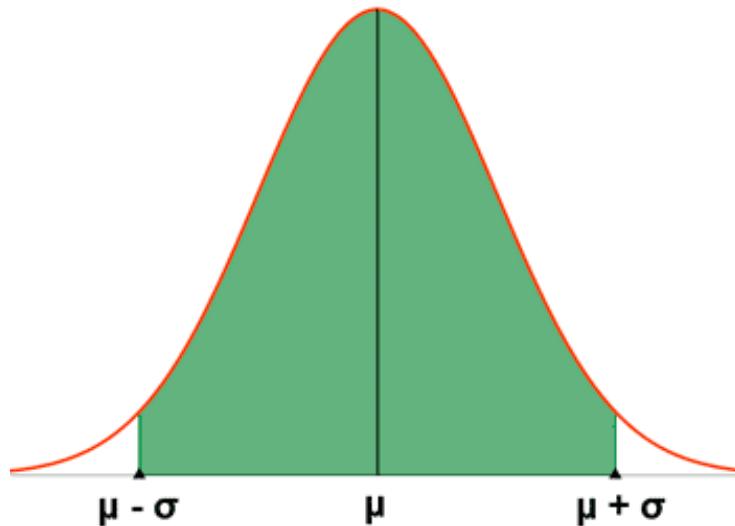


Figura 3.13: distribución de Gauss (Fuente: calculo.cc [45])

El valor central de una distribución de Gauss es la media, representada como μ , mientras que σ es la desviación típica de la muestra. Por tanto, un filtro gaussiano calculará los valores del *kernel* en función de la media de los píxeles así como su desviación típica. Los píxeles más cercanos a la media serán los más decisivos a la hora de computar la nueva imagen, mientras que se obviarán las características de aquellos píxeles que presenten valores muy alejados de la desviación típica.

En definitiva, dado un píxel ubicado en la posición (x,y) , la transformación a aplicar (o valor en la matriz *kernel*) viene determinada por la función gaussiana:

$$f(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.1)$$

En el caso que nos ocupa, la desviación típica es fijada en un valor de 10, mientras que las dimensiones de la matriz *kernel* son ajustadas automáticamente por OpenCV según dicho valor [46]. Se ha fijado 10 como desviación típica en un intento de conseguir una eliminación efectiva de características innecesarias en la imagen. En la Figura 3.14 puede

observarse el resultado de aplicar el filtro gaussiano con los valores indicados.

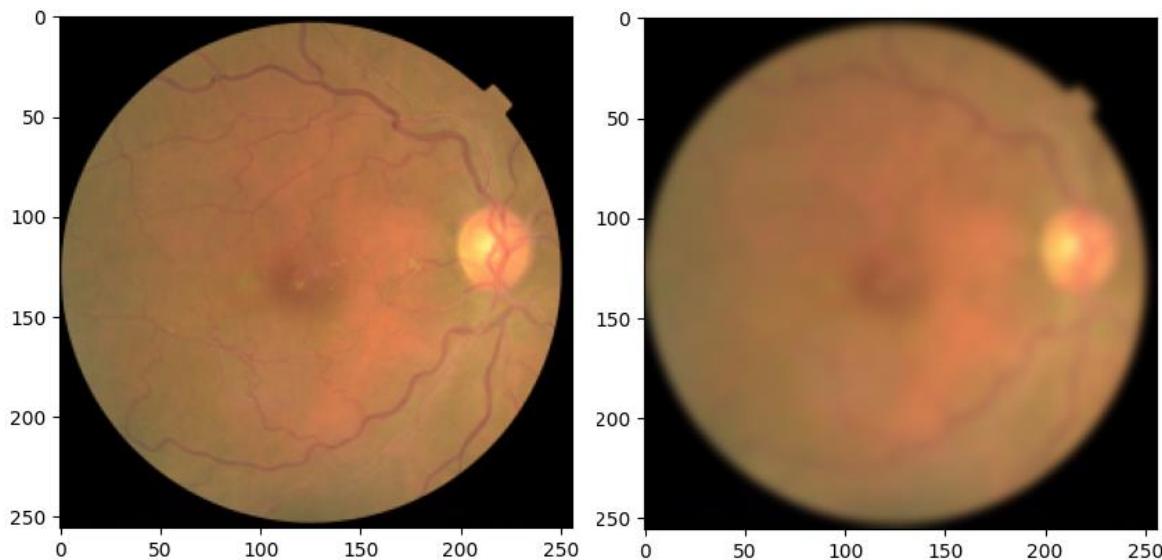


Figura 3.14: Aplicación del filtro gaussiano a muestra (Fuente: Elaboración propia)

SUSTRACCIÓN DE LA MEDIA DE COLOR

Desenfocada la imagen, y eliminada así una gran cantidad de información irrelevante, el objetivo se centra en conseguir el resaltado de los capilares y la vascularización del ojo. Puede comprobarse que éstos presentan una morfología distinta al resto de la retina, también a nivel de color. Por ello, se opta por eliminar las tonalidades uniformes al resto de la retina.

Este efecto es conseguido sustrayendo la media de color de cada píxel, para lo cual se utilizará la imagen resultante de aplicar el filtro gaussiano. Mediante la combinación de ambas imágenes, pre-filtro y post-filtro, se obtiene el resultado observable en la *Figura 3.15*.

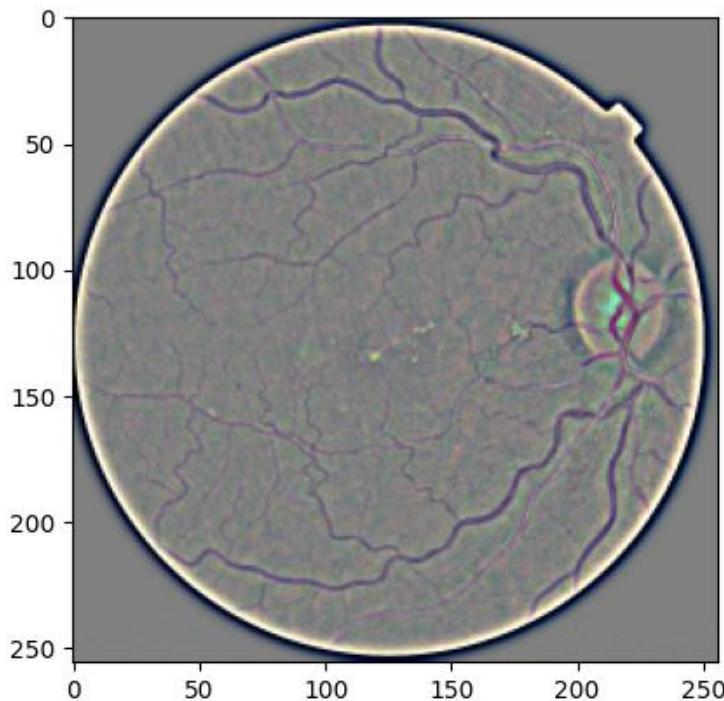


Figura 3.15: Sustracción de la media de color a muestra (Fuente: Elaboración propia)

OpenCV [47] permite la adición de dos capas de imagen, estableciendo un peso para cada capa, mediante la función expresada en la *Ecuación 3.2*.

$$dst = \alpha * src1 + \beta * src2 + \gamma \quad (3.2)$$

Donde dst es la matriz de píxeles de la imagen resultante, $src1$ y $src2$ el mapa de píxeles de las dos imágenes a combinar, α y β los pesos, mientras que γ es una constante. A mayores pesos, mayor será el resaltado de los capilares. En este proyecto, se han fijado como valor de los pesos 4 y -4, mientras que la constante γ ha quedado fijada en 128. Durante el ajuste del entrenamiento, se probaron valores superiores, comprobándose que conducían a una *sobrepredicción* por parte del modelo resultante.

REDIMENSIONAMIENTO

En último lugar, las imágenes fueron redimensionadas de nuevo, a 256x256, con el fin de acelerar el entrenamiento y evitar problemas de memoria en la unidad de procesamiento

gráfico. Este redimensionamiento fue llevado a cabo mediante interpolación por el vecino más cercano. Para el cálculo de los píxeles de la imagen reducida se utiliza una interpolación de área [48].

3.5.4.4. AUMENTACIÓN

La aumentación consiste en insertar en el conjunto de datos una o varias copias modificadas de la muestra, aplicando una serie de técnicas. Permite que el modelo entrenado generalice mejor, reduciendo así el *overfitting* o sobreajuste, es decir, que el modelo esté excesivamente adaptado a las peculiaridades de la muestra de entrenamiento, presentando niveles de exactitud sumamente inferiores al presentarse nuevos datos.

En este caso, por cada imagen se ha insertado una copia transformada, la cual ha sido rotada aleatoriamente entre 0 y 360 grados, además de escalarse $\pm 10\%$ respecto a su tamaño original. En la *Figura 3.16* se muestran varios ejemplos de las fotografías resultantes.

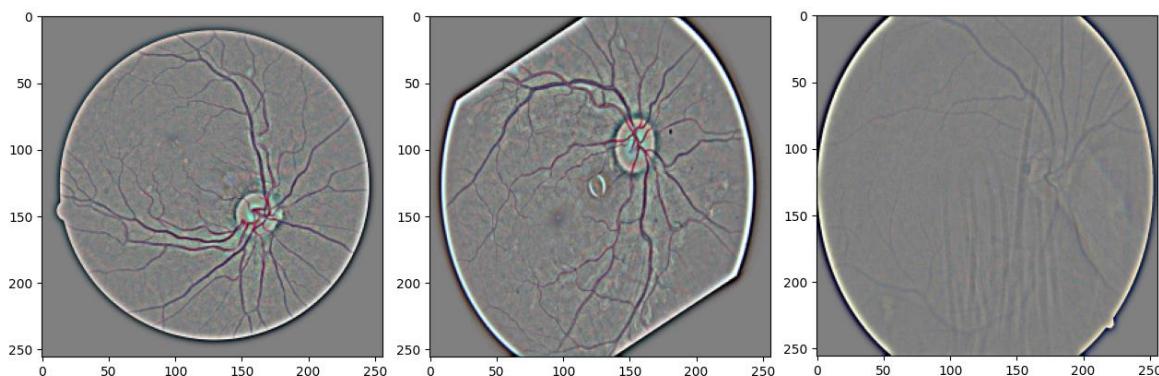


Figura 3.16: Varias muestras aumentadas

3.5.4.5. ARQUITECTURA DE LA RED NEURONAL

Este proyecto emplea una arquitectura de red convolucional conocida como ***U-Net***. Desarrollada por la Universidad de Freiburg para la segmentación de imágenes biomédicas, se basa en la aplicación de sucesivas operaciones de convolución, tal y como se explicaron en el punto anterior.

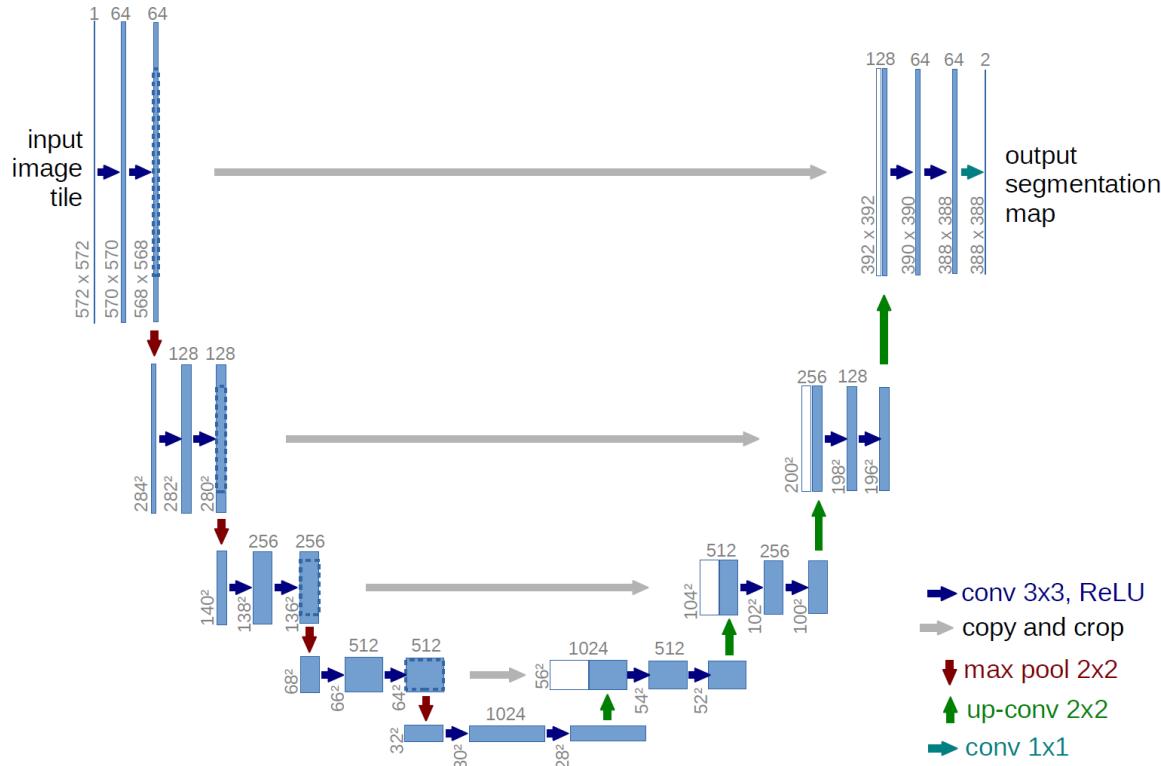


Figura 3.17: Arquitectura UNet original (Fuente: LMB – University of Freiburg [49])

U-Net aplica segmentación semántica, es decir, otorga a cada píxel de la imagen una etiqueta o conjunto de etiquetas. En el caso particular del *paper* original de la arquitectura, reflejado en la Figura 3.17, se toma una imagen de 572x572 píxeles, y se obtiene como salida otra captura de 388x388 píxeles, cada uno con dos etiquetas. Este escenario es similar al planteado para la solución propuesta, donde se quiere obtener un arreglo con dos probabilidades, la de que la muestra sea positiva y la de que no lo sea.

Su forma de U viene definida por las fases de *downsampling* y *upsampling*. En la primera fase, el conjunto de capas de red, conocidas como **encoder**, funcionan como extractores de características. En el cuello de botella que separa ambas fases, cada píxel ha recibido un total de 1024 etiquetas. A la par, el tamaño de la imagen se reduce. En la fase *upsampling*, o **decoder**, muchas de estas características de bajo nivel son descartadas en favor de una serie de rasgos más generales, a la par que se reamplía el mapa de píxeles.

Uno de los principales defectos de esta arquitectura reside en la pérdida de características a medida que aumenta el número de capas. En imágenes médicas, donde cada píxel puede

añadir un detalle de gran importancia, es de especial relevancia minimizar este fenómeno. Por ello, se introducen los ***skip connectors*** (nombrados en la *Figura 3.17* como *copy and crop*). En U-Net, cada nivel del encoder reintroduce las características de bajo nivel a su homólogo del decoder. Los skip connectors adquirieron gran popularidad a raíz del desarrollo de HE et al., *ResNet* [50].

Dados los condicionantes del problema tratado, se no ha utilizado la misma U-Net original. La solución propuesta toma como entrada una imagen de 256x256, y devuelve una salida del mismo tamaño, con un vector de dos características por píxel. En el decoder se utilizan matrices de convolución para reescalar la imagen, similares a las usadas en la arquitectura primigenia. Sin embargo, como encoder se ha usado **ResNet-34**.

ResNet-34 consiste en una red neuronal de 34 capas, en las que se aplican sucesivas capas de convolución mediante kernels de dimensión 3x3. Por cada dos operaciones, se localiza un skip connector (ver *Figura 3.18*).

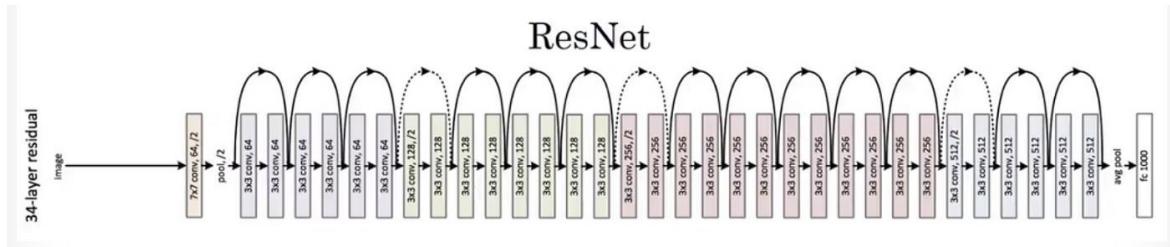


Figura 3.18: Arquitectura ResNet-34 (Fuente: Medium [51])

Además de las operaciones descritas, ResNet-34 utiliza dos operaciones de ***pooling*** para redimensionar la imagen: *max pooling* y *average pooling*. Un *pool* es una agrupación de píxeles de idéntica dimensión tanto en altura como en anchura. Se tomará como ejemplo una matriz de dimensiones 2x2, con cuatro píxeles de valores $[a, b, c, d]$. Si se realizase max pooling, se agruparían en un solo píxel p , que recibiría como valor:

$$p = \max(a, b, c, d) \quad (3.3)$$

En otro caso, si se hiciera average pooling, p recibiría el valor:

$$p = \text{avg}(a, b, c, d) \quad (3.4)$$

La arquitectura desarrollada para este proyecto no concluye con la red convolucional UNet explicada, faltando un último componente. En su salida, obtenemos dos características por cada píxel de la imagen es decir, un total de $2 \times 256 \times 256 = 131072$ características. Se desea obtener un solo vector de dos valores (probabilidad de positivo, probabilidad de negativo) para el conjunto de la fotografía. Se utiliza para ello una red de neuronas completamente conectadas (*fully-connected*).

Sean dos capas de neuronas a y b , con un total de j y k neuronas respectivamente. Se denotarán $a = [n_1, n_2, \dots, n_j]$ y $b = [p_1, p_2, \dots, p_k]$, siendo n una neurona de a , y p una neurona de b . Una red completamente conectada es aquella en la que existen conexiones desde n_1 , o cualquier otra neurona de a , hasta $[p_1, p_2, \dots, p_k]$. La Figura 3.19 ilustra esta situación.

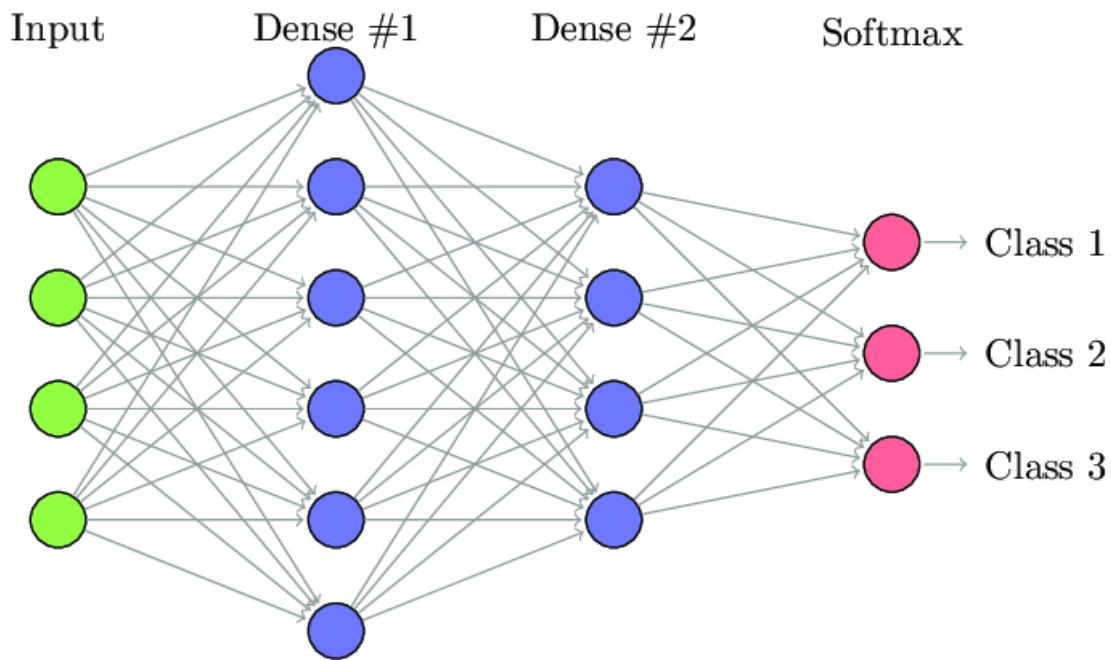


Figura 3.19: Esquema de una red completamente conectada (Fuente: ResearchGate [52])

Se distinguen una serie de entradas, *Input*, las neuronas, denotadas como *Dense*, y la salida. La entrada es un arreglo unidimensional de tamaño 131072, mientras que la salida es otro vector de tamaño 2, con las probabilidades ya mencionadas. Para que la salida esté delimitada en el rango $[0,1]$, se aplica la función *softmax*, la cual se define según la siguiente

ecuación.

$$\sigma(v)_j = \frac{e^{v_j}}{\sum_{k=1}^K e^{v_k}} \quad (3.5)$$

Donde $e = 2,7182$, y v^j es cada el valor que ocupa la posición j en el vector v .

3.5.4.6. PARÁMETROS DE ENTRENAMIENTO

Entrenar un modelo significa ajustar los pesos de sus neuronas/matrices de convolución, de manera que, presentada una entrada (en este caso, una imagen de fondo de retina) desconocida hasta el momento, sea capaz de predecir la etiqueta asociada a dicha entrada (positividad o negatividad de la muestra).

El entrenamiento comienza con los pesos en un determinado valor inicial. A continuación, el modelo procesa cada una de las entradas (imágenes) pertenecientes a la muestra de entrenamiento, anotando las salidas obtenidas. Las imágenes pueden ser procesadas una por una o en pequeños lotes (*mini-batches*), actualizando los valores de los pesos al finalizar el procesado de cada lote.

Las salidas producidas pueden coincidir o no con las esperadas. En caso de no coincidencia, se produce un error, o lo que es lo mismo, una desviación entre el valor obtenido y el esperado. Esta desviación es calculada conforme a una **función de pérdida**. El objetivo de todo entrenamiento es reducir el valor de desviación, para lo cual los pesos son actualizados según una **tasa de aprendizaje**. Esta tasa de aprendizaje, así mismo, puede variar a lo largo del entrenamiento, según determine el **optimizador**.

Este proceso completo se puede repetir un determinado número de iteraciones o **epochs**, hasta que las métricas evaluadas (serán tratadas en el punto 4) alcancen los valores deseados, o bien cuando se compruebe que se está dando *over-fitting*. Para detectar cuando se presenta esta situación, además de la muestra de entrenamiento, se debe consignar una muestra de validación. Ésta sirve para hacer *test* en tiempo real. Si de una iteración a otra, las métricas de validación empeoran significativamente, se está dando sobreajuste u overfitting.

HIPERPARÁMETROS

Descripto este escenario, encontramos una serie de **hiperparámetros** que definirán el entrenamiento:

- Los valores de **inicialización** de ResNet-34 se corresponden con los resultantes de entrenar un modelo de clasificación para *ImageNet* [53], un conjunto de imágenes provenientes de más de 20000 categorías.
- Como **tamaño de los lotes** o *batch size* se ha optado por 128. Existen dos condicionantes que llevan a elegir este tamaño. Por un lado, la velocidad de entrenamiento, siendo esta mayor cuanto más grandes sean los lotes, asunto de especial relevancia al trabajar con una muestra tan sumamente grande, de tal resolución, y en una plataforma de computación en la nube. Por otro, la capacidad de procesamiento gráfico, ya que debido a los factores anteriormente mencionados, no es posible trabajar con lotes mayores sin agotar los recursos disponibles. Se han probado tamaños inferiores, siguiendo una escala de múltiplos de 2, sin embargo, los modelos resultantes eran ligeramente peores a los conseguidos con la configuración final.
- La **tasa de aprendizaje** quedó fijada en 0.00008. El **optimizador** empleado, Adam [54], utiliza el momento de entrenamiento para readaptar la tasa, lo que lleva al modelo a converger (adaptarse al caso de entrenamiento) a una mayor velocidad, y en cierta medida, resta importancia al valor inicial.
- El **número de iteraciones** configurado fue 35. A lo largo del desarrollo, se observó que un número de iteraciones superior no mejoraba la calidad del modelo, derivando en over-fitting. Además, se combinó este valor con un mecanismo de *stop*, por el cual, el modelo finalmente guardado en memoria fue aquel que alcanzara un mayor valor de *F1 Score* (ver capítulo de métricas), independientemente de la iteración en la que se hubiera presentado.

FUNCIÓN DE PÉRDIDA: ENTROPÍA CRUZADA

Dada una variable de estudio X, la entropía se define como la incertidumbre que presenta

esa variable respecto a sus posibles valores. La entropía puede calcularse mediante la ecuación:

$$H(X) = -\sum_x p(x)\log(p(x)) \quad (3.6)$$

Donde x es uno de los posibles valores que puede presentar la variable de estudio X. Puede observarse que cuanto más similares sean las probabilidades asociadas a los posibles valores de X, mayor será la entropía.

Extrapolado al aprendizaje automático, un clasificador no estará bien entrenado si emite salidas al azar. Por tanto, se debe penalizar este tipo de comportamientos, aplicando una penalización mayor cuanto más grande sea la desviación entre los valores obtenidos y los esperados.

Recogiendo este objetivo, se define la función de entropía cruzada, o pérdida logarítmica, para un clasificador binario como:

$$L_{CE} = -\sum_{i=1}^2 t_i \log(p_i) \quad (3.7)$$

Donde t_i es el valor de verdad (0 o 1), y p_i la probabilidad softmax para la clase 0 o 1.

3.5.5. DISEÑO DEL SERVIDOR FLASK

Para servir el modelo de aprendizaje automático, se utiliza un servidor escrito en Python, apoyándose en el *framework* Flask. Este servidor expone un único método POST, encargado de evaluar una imagen.

POST /predict		
Descripción	POST to Predict	
OperationId	postPredict	
Consume	application/json	
Parámetros	JSON en body	
Respuestas	200 OK	Predicción obtenida con

		éxito. Se devuelve un arreglo como atributo de un objeto JavaScript.
	400 Bad Request	Petición mal formada.
	500 Internal Server Error	Fallo en el procesamiento.

Tabla 3.47: Ficha del método POST /predict

3.6. IMPLEMENTACIÓN

En este punto se recogerá, por un lado, la estructura de la aplicación, a través de los diferentes directorios y ficheros de cada uno de sus componentes. Por otro, se hará una breve descripción del despliegue.

3.6.1. ESTRUCTURA

En la raíz del repositorio, disponible en GitLab [55], se encuentran cuatro carpetas: **DED_AI_Client**, para la aplicación móvil, **DED_AI_Server**, para el servidor REST principal desarrollado en Express, y **DED_AI_Model_Server**, agrupando el modelo de aprendizaje automático así como el servidor dedicado a servir dicho modelo. A mayores, existe una carpeta **Training and Data**, que incluye contenido generado como apoyo durante el proceso de aprendizaje.

3.6.1.1. APLICACIÓN MÓVIL

El desarrollo móvil se estructura en torno a dos directorios básicos:

- **assets**: recoge el contenido multimedia empleado en la interfaz de usuario. Ya que solo se han utilizado los componentes nativos ofrecidos por Expo y React Native Paper, las imágenes que se pueden localizar en esta carpeta provienen de estas dos librerías.
- **src**: aglutina todo el código implementado. En esta carpeta se encuentra el fichero

theme.js, objeto JavaScript que abarca los distintos estilos CSS empleados en la aplicación. Además, se localizan los siguientes subdirectorios:

- **views:** contiene los distintos ficheros JSX nativos de React. Cada fichero representa una vista de las descritas en el apartado 3.5.1.
- **services:** componentes JavaScript escritos con la finalidad de separar la funcionalidad de conexión al servidor REST. Cada fichero contiene los métodos necesarios para enviar información hacia los puntos de comunicación detallados en el apartado 3.5.2.

La estructura interna de las vistas y servicios será discutida en apartados sucesivos. En lo sucesivo, cabe destacar que todos los componentes han sido implementados de forma funcional, por lo que contarán con un área de declaraciones/importaciones, y posteriormente, la función propiamente definida. Así mismo, en el directorio raíz también se encuentran los archivos:

- **App.js:** punto de entrada de la aplicación *React Native*. Devuelve una pila de *React Navigation*, conteniendo cada una de las vistas almacenadas en *views*. Su función es posibilitar el avance y retroceso entre vistas. La *Figura 3.20* muestra el código de este fichero.

```
import Welcome from './src/views/Welcome';
import Preview from './src/views/Preview';
import Loading from './src/views>Loading';
import Error from './src/views/Error';
import Results from './src/views/Results';
import Feedback from './src/views/Feedback';
import Thanks from './src/views/Thanks';
import Login from './src/views/Login';
import {NavigationContainer} from '@react-navigation/native';
import {createNativeStackNavigator} from '@react-navigation/native-stack';

const Stack = createNativeStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Login" component={Login} options={{headerShown: false}}/>
        <Stack.Screen name="Welcome" component={Welcome} options={{headerShown: false}}/>
        <Stack.Screen name="Preview" component={Preview} options={{headerShown: false}}/>
        <Stack.Screen name="Loading" component={Loading} options={{headerShown: false}}/>
        <Stack.Screen name="Error" component={Error} options={{headerShown: false}}/>
        <Stack.Screen name="Results" component={Results} options={{headerShown: false}}/>
        <Stack.Screen name="Feedback" component={Feedback} options={{headerShown: false}}/>
        <Stack.Screen name="Thanks" component={Thanks} options={{headerShown: false}}/>
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Figura 3.20: Componente App.js

- **package.json**: archivo creado por Node.js y su gestor de paquetes npm. Recoge las dependencias del proyecto, es decir, las librerías de terceros utilizadas para implementar la funcionalidad desarrollada.
- **package-lock.json**: fichero complementario al anterior, recogiendo las versiones instaladas de cada librería, así como otros atributos como la firma de integridad del paquete, la URL utilizada para la descarga, y subdependencias.
- **app.json**: objeto JavaScript que recoge metainformación sobre el proyecto. Así mismo, también es empleado por Expo para detallar dependencias o parámetros especiales para alguna de las plataformas soportadas.
- **.gitignore**: empleado por el controlador git para especificar los ficheros y directorios que no serán tenidos en cuenta la hora de crear el histórico de versiones.
- **babel.config.js**: Babel es el compilador de JavaScript utilizado por defecto en React Native. Este archivo especifica una serie de parámetros de su configuración.

VISTAS

Como se ha explicado, cada una de las vistas se describe en un fichero JSX. Ya que todos siguen la misma estructura, se realizará una descripción a modo de ejemplo del código de la vista *Error*.

Como en todo componente funcional de JavaScript, en primer lugar se observa un área de declaraciones. En ella se importan todos los componentes empleados, tanto para la parte lógica como para la visual. En la *Figura 3.21* pueden localizarse, entre otras, la importación de varios componentes de interfaz ofrecidos por React Native Paper, o del objeto *theme* de hoja de estilos.

```
import React from "react";
import {View, StyleSheet} from "react-native";
import {Text, Button} from 'react-native-paper';
import {StackActions} from '@react-navigation/native';
import theme from "../theme";
```

Figura 3.21: Área de importaciones en la vista Error

Posteriormente, encontramos la función, junto con su configuración de exportación, que determina su visibilidad y reutilización en otras funciones o componentes. Para todas las vistas, se ha utilizado la configuración por defecto *export default*. La función recibe como parámetro un objeto con dos atributos: *route* y *navigation*, empleados para recuperar los datos pasados desde otras vistas, así como la pila de navegación.

Dentro de la función, primero se definen las constantes empleadas como estados de la vista. En la mayor parte de las ocasiones, los valores de estas constantes son recuperados desde *route*. Después, se declaran otra serie de funciones internas, llamadas generalmente al pulsar un botón. En este caso particular, se observa la función *retry*, al pulsar el botón *Reintentar*, y la función *goBack*, al realizar la misma acción en *Volver*. Ya que la vista *Error* puede mostrarse en tres casos distintos (fallo al iniciar sesión, analizar, o dar feedback), se comprueba qué situación se presenta para dar paso a una u otra acción. El código de ambas funciones se puede observar en la *Figura 3.22*.

```
function retry(){
  if(analysisToRetry !== undefined){
    navigation.dispatch(StackActions.replace('Loading', {toAnalyze: analysisToRetry, accessToken: token}));
  }else if(feedbackToRetry !== undefined){
    navigation.dispatch(StackActions.replace('Loading', {feedback: feedbackToRetry, sample: aSample, accessToken: token}));
  }else if(credentialsToRetry !== undefined){
    navigation.dispatch(StackActions.replace('Loading', {credentials: credentialsToRetry}));
  }
}

function goBack(){
  if(credentialsToRetry !== undefined){
    navigation.dispatch(StackActions.replace('Login'));
  }else{
    navigation.dispatch(StackActions.replace('Welcome', {accessToken: token}));
  }
}
```

Figura 3.22: Funciones internas en la vista Error

Finalmente se encuentra el área de retorno, conteniendo el código de estilo HTML que será renderizado. En *React Native*, se usa *View* como componente básico, haciendo las veces de contenedor para otros componentes. En la *Figura 3.23* puede observarse como, en este ejemplo, se utiliza para anidar otros tres bloques *View*, los cuales contienen el título, el párrafo de descripción, y la zona de botones, respectivamente. Para introducir el título y los párrafos, se utilizan componentes *Text*, mientras que para los botones se utiliza *Button*.

```

return(
  <View style={styles.mainContainer}>
    <View styles={styles.titleParagraphContainers}>
      <Text style={styles.title}>
        | Error
      </Text>
    </View>

    <View style={styles.titleParagraphContainers}>
      <Text style={styles.paragraph}>{message}</Text>
    </View>

    <View style={styles.buttonsCotainer}>
      <Button style={styles.button} labelStyle={styles.buttonFont} icon="refresh" mode="outlined" onPress={() => {retry()}}>
        | Reintentar
      </Button>
      <Button style={styles.button} labelStyle={styles.buttonFont} icon="home" mode="outlined" onPress={() => {goBack()}}>
        | Volver
      </Button>
    </View>
  </View>
)

```

Figura 3.23: Área de retorno en la vista Error

Muchos de los componentes tienen definido un valor para el parámetro *style*, tomando de atributos de la constante *styles*. La definición de esta constante, un objeto fuera de la función cierra el archivo JSX. La mayoría de atributos hacen referencia a su vez al tema definido en *theme.js*.

SERVICIOS

Dentro de esta categoría recaen dos componentes diseñados para establecer conexión con el servidor REST: *AuthService*, para el proceso de autenticación, y *AnalysisService*, para enviar imágenes y *feedback*. Ambos cuentan con una única importación, *axios*, que permite realizar las peticiones HTTP. De nuevo, se exportan siguiendo la configuración por defecto. En *AuthService* se declara un único método *getSessionToken*, el cual recibe un objeto JSON, con el usuario y contraseña recogidos en la vista *View*, y retorna el resultado de la petición POST enviada al endpoint */auth*. *AnalysisService* define los métodos *sendImage* y *sendFeedback*, para las peticiones POST y PUT a */analysis*. Junto a la información la petición HTTP siempre contiene un encabezado de autenticación, que envía el token obtenido al iniciar sesión. El código de *AnalysisService* se muestra en la Figura 3.24.

```

import axios from 'axios';

export default {
    sendImage(pic, token) {
        return axios.post('https://ded-ai-api-server.onrender.com/analysis', {raw: pic}, {
            headers: {
                'Authorization' : `Bearer ${token}`
            }
        })
    },
    sendFeedback(sample, feedback, token){
        sample.feedbackResult = feedback;

        return axios.put('https://ded-ai-api-server.onrender.com/analysis', sample, {
            headers: {
                'Authorization' : `Bearer ${token}`
            }
        })
    }
};

```

Figura 3.24: Código desarrollado para el servicio AnalysisService

3.6.1.2. SERVIDOR REST EXPRESS Y BASE DE DATOS

El directorio **DED_AI_Server** se estructura en cinco subcarpetas:

- **doc:** contiene el fichero de documentación de la API, *api-doc.js*, elaborado siguiendo la especificación OpenAPI. Su contenido ha sido descrito en los apartados 3.5.2.1, 3.5.2.2 y 3.5.2.3, sobre los parámetros generales, de seguridad y objetos del servidor.
- **config:** incluye el fichero *db.js*, con los parámetros y función de conexión a la base de datos.
- **models:** agrupa los esquemas de cada colección de la base de datos, descritos en el apartado 3.5.3, sobre diseño de la base de datos.
- **helpers:** contiene los ficheros *OAuth2.js*, para la gestión del proceso de inicio de sesión, y *email.js*, que implementa una función de envío de correo a los usuarios cuando se registran por primera vez en el sistema.
- **paths:** raíz de tres subdirectorios, *analysis*, *auth*, y *register*, que representan los puntos de comunicación homónimos descritos en el apartado 3.5.2.

Además, también pueden encontrarse los ficheros **package.json**, **package-lock.json** y

.gitignore, que realizan las mismas funciones que en el caso de la aplicación móvil. Por último, se encuentra el fichero principal ***App.js***.

FICHERO PRINCIPAL

El archivo *App.js* comienza requiriendo todos los módulos/librerías necesarias para ejecutar el servidor. Entre ellos encontramos el propio ExpressJS, el *framework* OpenAPI (*express-openapi*), la función de conexión a base de datos, *OAuth2Server* para el proceso de autenticación, *swagger* para arrancar un servidor de documentación basado en OpenAPI, *helmet* [56] como buena práctica de seguridad, y diversos manejadores de peticiones y rutas, como *body-parser* y *path*. También se encuentran las variables para definir el puerto, o para almacenar una nueva instancia de Express, *port* y *app* (ver Figura 3.25).

```
var express = require('express');
var path = require('path');
var logger = require('morgan');
var helmet = require('helmet');
var parser = require('body-parser')
var openAPI = require("express-openapi");
var swagger = require("swagger-ui-express");
var OAuth2Server = require("@node-oauth/oauth2-server");
var connectDB = require("./config/db");
var port = process.env.PORT || 3001;
var app = express();
```

Figura 3.25: Variables definidas en el archivo principal de Express

Instanciado el servidor, se le pasan todos los módulos anteriormente referenciados, utilizando el método *use*. También se conecta con la base de datos, llamando a la función *connectDB*, y el servidor se abre a peticiones a través del método *listen*.

Así mismo, se inicia el servidor de autenticación, una nueva instancia de *OAuth2Server*. Recibe como parámetros el fichero donde se define el manejo de tokens (*OAuth2.js*) y el tiempo de vida para los mismos. También se pueden activar una serie de opciones, como el paso de tokens como un parámetro más en la URL. Las inicializaciones expuestas se

muestran en la captura de código correspondiente (ver *Figura 3.26*).

```
connectDB();

app.listen(port, () => {
    console.log(`DED_AI API Server running on ${port}`);
    console.log(`OpenAPI documentation available in http://localhost:\${port}/api-documentation`);
});

app.use(logger('dev'));
app.use(express.json({limit: '20mb'}));
app.use(parser.urlencoded({limit: '20mb', extended: true}));
app.disable('x-powered-by');
app.use(helmet());

app.oauth = new OAuth2Server({
    model: require('./helpers/OAuth2.js'),
    accessTokenLifetime: 60 * 60,
    allowBearerTokensInQueryString: true
});
```

Figura 3.26: Inicializaciones en el servidor Express, parte 1

Se inicializa el *framework* OpenAPI, indicando la instancia del servidor Express, el fichero básico de documentación (*api-doc.js*), el directorio donde se encuentran las rutas del servidor y la implementación de los patrones de seguridad, utilizando para ello los métodos proveídos por *OAuth2Server*. Finalmente, una última llamada a *use* pone en marcha en servidor de documentación basado en *swagger* (ver *Figura 3.27*).

```

openAPI.initialize({
  app,
  securityHandlers: {
    passwordAuthentication: function(req, scopes, schema){
      var request = new OAuth2Server.Request(req);
      var response = new OAuth2Server.Response(req.res);

      return app.oauth.authenticate(request, response)
        .catch(function(err){
          req.res.status(err.code || 500).json(err);
        });
    },
    apiDoc: require("./doc/api-doc"),
    paths: path.resolve(__dirname, 'paths')
  });
}

app.use(
  "/api-documentation",
  swagger.serve,
  swagger.setup(null, {
    swaggerOptions: {
      url: `http://localhost:${port}/api-docs`
    }
  })
);

```

Figura 3.27: Inicializaciones en el servidor Express, parte 2

3.6.1.3. SERVIDOR FLASK Y MODELO

En último lugar, en **DED_AI_Server_Model** encontramos cuatro ficheros:

- **App.py**: es el punto de entrada del servidor Flask. Define la función *predict*, la cual representa al *endpoint* POST homónimo descrito en capítulos anteriores. Así mismo, define otras dos funciones adicionales: *preprocess_image*, que aplica los procesos de conversión de la imagen a un tensor modificado según las técnicas del apartado 3.5.4.3, y *run_inference*, que carga el modelo contenido en *Net.py* con los pesos contenidos en *Model.pt*.
- **Net.py**: extiende la clase *nn.Module* de PyTorch. En su constructor define la arquitectura de la red neuronal, consistente, como se ha mencionado, en una *UNet* y una capa de neuronas *fully-connected*. El método *forward* es llamado en cada aplicación del modelo, y define los pasos a ejecutar cuando se evalúa una imagen

(ver *Figura 3.28*).

```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.unet = smp.UnetPlusPlus(classes=2)
        self.fc1 = nn.Linear(2 * 256 * 256, 2)

    def forward(self, x):
        unet_output = self.unet(x)
        unet_output = unet_output.contiguous()

        conv_flat = unet_output.view(-1, 2 * 256 * 256)
        out = self.fc1(conv_flat)

    return out
```

Figura 3.28: código de la clase Net

- **Model.pt**: archivo diccionario de PyTorch que contiene los pesos entrenados para la red neuronal.
- **Requirements.txt**: fichero que lista los paquetes necesarios para ejecutar el servidor, de cara su instalación por el gestor pip en el entorno de despliegue.

3.6.2. DESPLIEGUE

Los dos servidores desarrollados se han desplegado en la plataforma de servicios web Render [57] [58]. Se sigue una estrategia de despliegue continuo, donde cada actualización del repositorio genera una nueva tarea de compilación y ejecución en la máquina huésped.

3.6.2.1. SERVIDOR REST EXPRESS

En el servidor Express, desde la ruta dedicada DED_AI_Server, npm puede instalar todas las dependencias leyendo el fichero *package.json*, tras ejecutar la orden:

```
npm install
```

Desde el mismo directorio, para la ejecución, se usa el comando:

```
node app.js
```

3.6.2.2. SERVIDOR FLASK

Desde la ruta DED_AI_Server_Model, pip instala las dependencias mediante una lectura al fichero *requirements.txt*, mediante el comando:

```
pip install -r requirements.txt
```

El servidor web de referencia para ejecutar una aplicación Flask es Gunicorn [59]. Para lanzar el servidor, por tanto, se ejecuta la orden:

```
gunicorn app:app
```

4. Evaluación

A la hora de valorar la solución desarrollada, debemos distinguir, por un lado, la evaluación de la aplicación cliente-servidor, comprobando que cumple con los requisitos y es usable. Por otro lado, se detallarán los resultados de validar el modelo de aprendizaje automático, a través de diversas métricas.

4.1. APLICACIÓN CLIENTE-SERVIDOR

En este apartado se describirán, en primer lugar, las metodologías de evaluación empleadas para comprobar que la aplicación desarrollada satisface con éxito los objetivos del proyecto. Posteriormente, se ofrecerá información más detallada de cada evaluación por separado, incluyendo las conclusiones obtenidas.

4.1.1. METODOLOGÍAS

Se utilizan dos tipos de evaluación: validación y cuestionarios:

- **Validación:** tiene como objetivo verificar que la aplicación funciona tal y como el cliente desea, siguiendo las directrices establecidas en la Especificación de Requisitos del Software. Para ello, se trazan una serie de pruebas de caja negra que demuestran la conformidad con los requisitos, o revelan una desviación de las especificaciones.
- **Cuestionario:** herramienta ampliamente utilizada en el campo de accesibilidad, plantea una serie de preguntas a un grupo de usuarios de la aplicación. Su objetivo es valorar la usabilidad del sistema, definida en la ISO/IEC 9241 [60] como la *“medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado”*.

4.1.2. VALIDACIÓN

Para configurar un proceso de validación exitoso, es necesario seleccionar de forma adecuada el conjunto de casos de prueba. Para ello, se utilizará el diagrama de flujo de la aplicación, descrito de forma esquemática, y posteriormente, se detallarán los casos.

4.1.2.1. FLUJO DE LA APLICACIÓN

El flujo de la aplicación ya fue detallado a lo largo del punto 3, sin embargo, las *Figuras 4.1, 4.2 y 4.3* lo recogen de forma esquemática y a alto nivel. Se resalta en naranja el punto de partida, así como en verde cada cambio de vista en el cliente móvil.

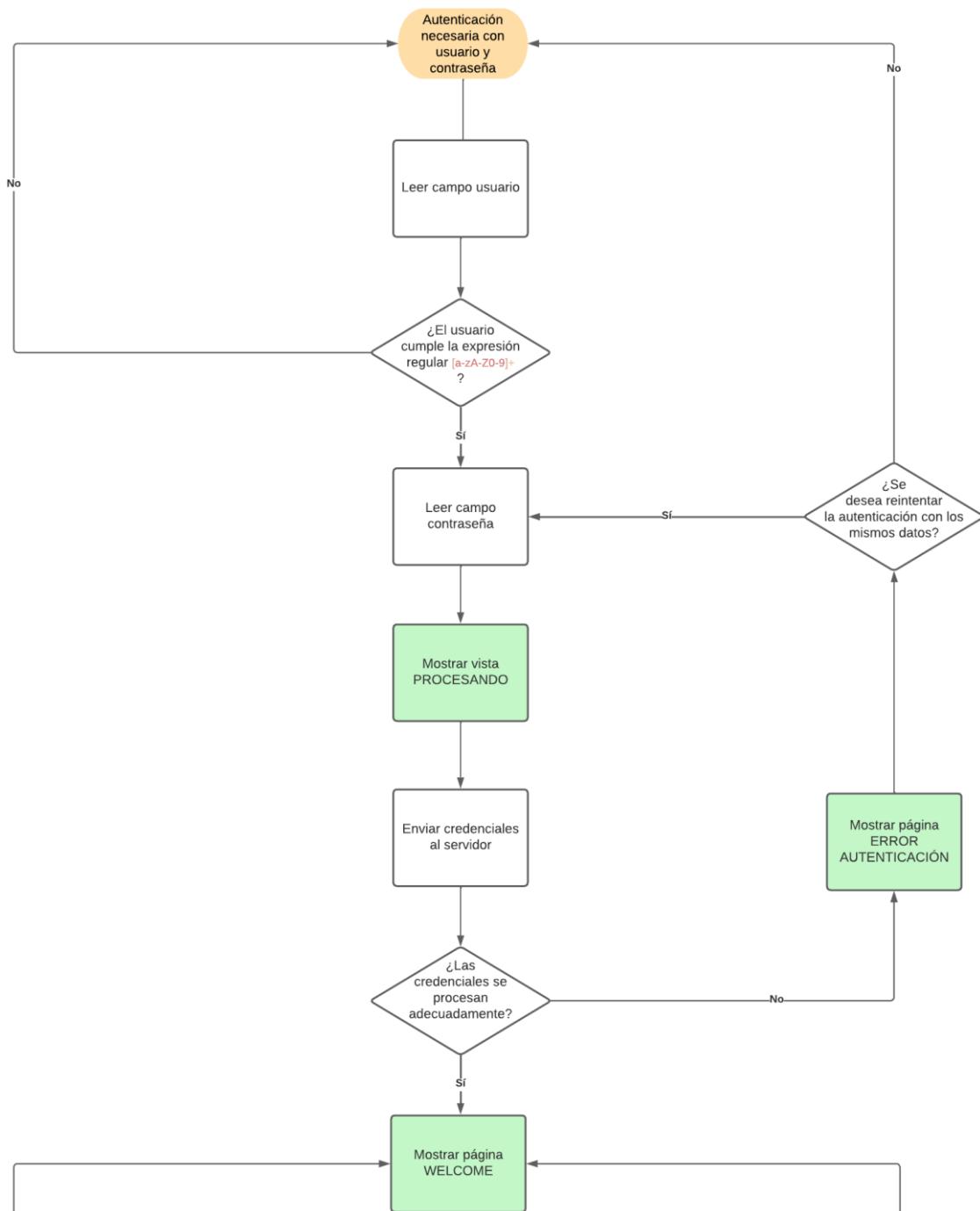


Figura 4.1: Diagrama de flujo de la aplicación desarrollada, parte 1

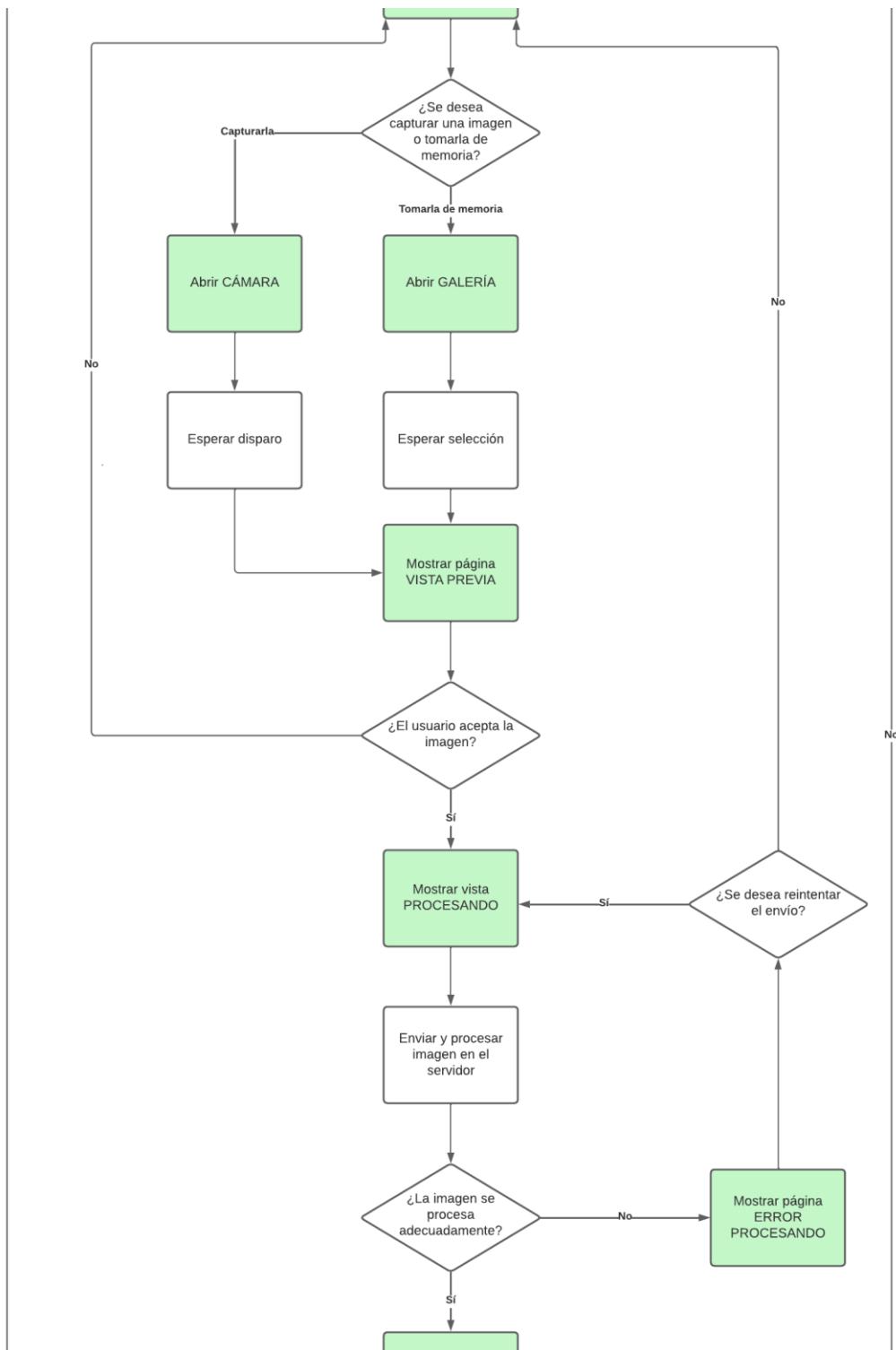


Figura 4.2: Diagrama de flujo de la aplicación desarrollada, parte 2

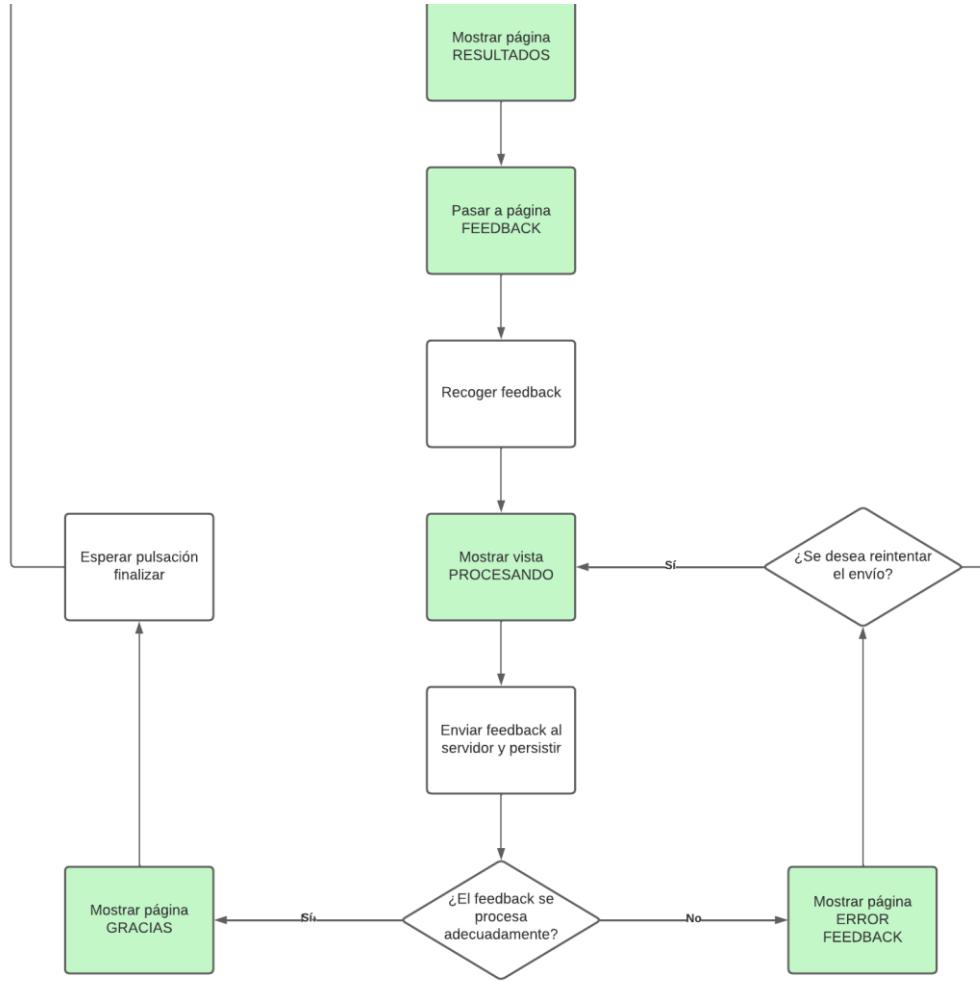


Figura 4.3: Diagrama de flujo de la aplicación desarrollada, parte 3

4.1.2.2. CASOS DE PRUEBA

De cada posible itinerario del diagrama de flujo puede extraerse un caso de prueba. Se recogen a continuación individualmente, indicando una serie de precondiciones, acciones realizadas por el usuario, y resultado esperado. Además, se recoge el resultado obtenido al realizar la prueba.

[CP-1]: Usuario con caracteres no alfanuméricos	
Precondiciones	El cliente móvil se ha iniciado
Flujo de acciones	<ol style="list-style-type: none"> 1. Se pulsa sobre el campo <i>Usuario</i> 2. Se introducen caracteres no alfanuméricos 3. Se pulsa sobre el botón <i>Continuar</i>

Resultado esperado	El campo <i>Usuario</i> se resalta en rojo
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.1 – Caso de prueba 1 – Usuario con caracteres no alfanuméricos

[CP-2]: Inicio de sesión correcto	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet
Flujo de acciones	<ol style="list-style-type: none"> 1. Se pulsa sobre el campo <i>Usuario</i> 2. Se introduce un usuario registrado en la aplicación 3. Se pulsa sobre el campo <i>Contraseña</i> 4. Se introduce la contraseña asignada al usuario 5. Se pulsa sobre el botón <i>Continuar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Welcome</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.2 – Caso de prueba 2 – Inicio de sesión correcto

[CP-3]: Usuario incorrecto	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet
Flujo de acciones	<ol style="list-style-type: none"> 1. Se pulsa sobre el campo <i>Usuario</i> 2. Se introduce un usuario incorrecto 3. Se pulsa sobre el campo <i>Contraseña</i> 4. Se introduce una contraseña 5. Se pulsa sobre el botón <i>Continuar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Error</i> , mostrándose el mensaje “ <i>Ha habido un fallo al iniciar sesión</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.3 – Caso de prueba 3 – Usuario incorrecto

[CP-4]: Contraseña incorrecta	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre el campo <i>Usuario</i> 2. Se introduce un usuario registrado en la aplicación 3. Se pulsa sobre el campo <i>Contraseña</i> 4. Se introduce una contraseña incorrecta 5. Se pulsa sobre el botón <i>Continuar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Error</i> , mostrándose el mensaje “ <i>Ha habido un fallo al iniciar sesión</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.4 – Caso de prueba 4 – Contraseña incorrecta

[CP-5]: Reintentó con credenciales incorrectas	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet Se ha intentado iniciar sesión con credenciales incorrectas Se muestra la vista <i>Error</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Reintentar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Error</i> , mostrándose el mensaje “ <i>Ha habido un fallo al iniciar sesión</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.5 – Caso de prueba 5 – Reintentó con credenciales incorrectas

[CP-6]: Vuelta a Login	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet Se ha intentado iniciar sesión con credenciales incorrectas Se muestra la vista <i>Error</i>

Flujo de acciones	1. Se pulsa sobre el botón <i>Volver</i>
Resultado esperado	Se muestra la pantalla <i>Login</i> , con los campos <i>Usuario</i> y <i>Contraseña</i> en blanco
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.6 – Caso de prueba 6 –Vuelta a Login

[CP-7]: Sin conexión al iniciar sesión	
Precondiciones	El cliente móvil se ha iniciado No existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre el campo <i>Usuario</i> 2. Se introduce un usuario registrado en la aplicación 3. Se pulsa sobre el campo <i>Contraseña</i> 4. Se introduce la contraseña asignada al usuario 5. Se pulsa sobre el botón <i>Continuar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Error</i> , mostrándose el mensaje “ <i>Ha habido un fallo al iniciar sesión</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.7 – Caso de prueba 7 – Sin conexión al iniciar sesión

[CP-8]: Reintentó de inicio de sesión sin conexión	
Precondiciones	El cliente móvil se ha iniciado Se ha intentado iniciar sesión Se muestra la vista <i>Error</i> No existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre botón <i>Reintentar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Error</i> , mostrándose el mensaje “ <i>Ha habido un fallo al iniciar sesión</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.8 – Caso de prueba 8 – Reintentar inicio de sesión sin conexión

[CP-9]: Reintentar inicio de sesión con conexión	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet Se ha intentado iniciar sesión con credenciales correctas Se muestra la vista <i>Error</i>
Flujo de acciones	1. Se pulsa sobre botón <i>Reintentar</i>
Resultado esperado	Se muestra la pantalla <i>Loading</i> . Después, se pasa a la vista <i>Welcome</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.9 – Caso de prueba 9 – Reintentar inicio de sesión con conexión

[CP-10]: Captura de fotografía mediante cámara	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se muestra la vista <i>Welcome</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Cámara</i> 2. Se lanza la aplicación de cámara 3. Se pulsa sobre el disparador 4. Se confirma la muestra en la aplicación de cámara
Resultado esperado	Se muestra la pantalla <i>Preview</i> con la imagen capturada
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.10 – Caso de prueba 10 – Captura de fotografía mediante cámara

[CP-11]: Selección de imagen mediante galería	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se muestra la vista <i>Welcome</i>

Flujo de acciones	1. Se pulsa sobre el botón <i>Galería</i> 2. Se lanza un explorador de fotografías 3. Se pulsa sobre la imagen deseada
Resultado esperado	Se muestra la pantalla <i>Preview</i> con la imagen seleccionada
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.11 – Caso de prueba 11 – Selección de imagen mediante galería

[CP-12]: Rechazo de vista previa	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha seleccionado una imagen Se muestra la pantalla <i>Preview</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Cancelar</i>
Resultado esperado	Se muestra la pantalla <i>Welcome</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.12 – Caso de prueba 12 – Rechazo de vista previa

[CP-13]: Envío de imagen correcto	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet Se ha iniciado sesión en la aplicación Se ha seleccionado una imagen Se muestra la pantalla <i>Preview</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Continuar</i>
Resultado esperado	Se muestra la vista <i>Loading</i> . Tras unos segundos, se pasa a la pantalla <i>Results</i> , donde se ofrece el resultado <i>Positivo/Negativo</i> .
Resultado	Se obtiene el resultado esperado

obtenido	
-----------------	--

Tabla 4.13 – Caso de prueba 13 – Envío de imagen correcto

[CP-14]: Envío de imagen sin conexión	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha seleccionado una imagen Se muestra la pantalla <i>Preview</i> No existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre el botón <i>Confirmar</i>
Resultado esperado	Se muestra la vista <i>Loading</i> , pasando inmediatamente a la vista <i>Error</i> con el mensaje “ <i>Ha habido un fallo al enviar la imagen al servidor</i> ”
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.14 – Caso de prueba 14 – Envío de imagen sin conexión

[CP-15]: Reintentó de envío sin conexión	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha intentado enviar una imagen al servidor Se muestra la pantalla <i>Error</i> No existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre el botón <i>Reintentar</i>
Resultado esperado	Se muestra la vista <i>Loading</i> , pasando inmediatamente a la vista <i>Error</i> con el mensaje “ <i>Ha habido un fallo al enviar la imagen al servidor</i> ”
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.15 – Caso de prueba 15 – Reintentó de envío sin conexión

[CP-16]: Reintentó de envío con conexión	
Precondiciones	El cliente móvil se ha iniciado

	Existe conectividad a Internet Se ha iniciado sesión en la aplicación Se ha intentado enviar una imagen al servidor Se muestra la pantalla <i>Error</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Reintentar</i>
Resultado esperado	Se muestra la vista <i>Loading</i> . Tras unos segundos, se pasa a la pantalla <i>Results</i> , donde se ofrece el resultado <i>Positivo/Negativo</i> .
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.16 – Caso de prueba 16 – Reintento de envío con conexión

[CP-17]: Vuelta a Welcome tras envío de imagen fallido	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha intentado enviar una imagen al servidor Se muestra la pantalla <i>Error</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Volver</i>
Resultado esperado	Se muestra la vista <i>Welcome</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.17 – Caso de prueba 17 – Vuelta a Welcome tras envío de imagen fallido

[CP-18]: Avance a vista Feedback	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se visualiza la página <i>Results</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Siguiente</i>
Resultado	Se muestra la vista <i>Feedback</i>

esperado	
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.18 – Caso de prueba 18 – Avance a vista Feedback

[CP-19]: Envío de feedback correcto	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se han obtenido los resultados del análisis Se visualiza la vista <i>Feedback</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Sí/No</i>
Resultado esperado	Se muestra la vista <i>Loading</i> , avanzando inmediatamente a la pantalla <i>Thanks</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.19 – Caso de prueba 19 – Envío de feedback correcto

[CP-20]: Envío de feedback sin conexión	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se han obtenido los resultados del análisis Se visualiza la vista <i>Feedback</i> No existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre el botón <i>Sí/No</i>
Resultado esperado	Se muestra la vista <i>Loading</i> , pasándose a la vista <i>Error</i> donde se muestra el mensaje “ <i>Ha habido un fallo al enviar el feedback</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.20 – Caso de prueba 20 – Envío de feedback sin conexión

[CP-21]: Reintentó de envío de feedback sin conexión	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se han obtenido los resultados del análisis Se ha intentado enviar feedback al servidor Se visualiza la vista <i>Error</i> No existe conectividad a Internet
Flujo de acciones	1. Se pulsa sobre el botón <i>Reintentar</i>
Resultado esperado	Se muestra la vista <i>Loading</i> , pasándose a la vista <i>Error</i> donde se muestra el mensaje “ <i>Ha habido un fallo al enviar el feedback</i> ”.
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.21 – Caso de prueba 21 – Reintentó de envío de feedback sin conexión

[CP-22]: Reintentó de envío de feedback con conexión	
Precondiciones	El cliente móvil se ha iniciado Existe conectividad a Internet Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se han obtenido los resultados del análisis Se ha intentado enviar feedback al servidor Se visualiza la vista <i>Error</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Reintentar</i>
Resultado esperado	Se muestra la vista <i>Loading</i> , avanzando inmediatamente a la pantalla <i>Thanks</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.22 – Caso de prueba 22 – Reintentó de envío de feedback con conexión

[CP-23]: Vuelta a Welcome tras envío de feedback fallido	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se han obtenido los resultados del análisis Se ha intentado enviar feedback al servidor Se visualiza la vista <i>Error</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Volver</i>
Resultado esperado	Se muestra la vista <i>Welcome</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.23 – Caso de prueba 23 – Vuelta a Welcome tras envío de feedback fallido

[CP-24]: Finalización en Thanks	
Precondiciones	El cliente móvil se ha iniciado Se ha iniciado sesión en la aplicación Se ha enviado una imagen para analizar Se han obtenido los resultados del análisis Se ha enviado feedback al servidor Se visualiza la vista <i>Thanks</i>
Flujo de acciones	1. Se pulsa sobre el botón <i>Volver</i>
Resultado esperado	Se muestra la vista <i>Welcome</i>
Resultado obtenido	Se obtiene el resultado esperado

Tabla 4.24 – Caso de prueba 24 – Finalización en Thanks

4.1.3. CUESTIONARIO

En esta metodología se ha recurrido a personas del entorno de desarrollo, facilitándoles el acceso a la aplicación, a través de las estaciones de trabajo empleadas, y posteriormente un enlace a Google Forms para evaluar la experiencia [61]. Debido a esta limitación, la muestra alcanzada ha sido escasa (11 personas), pero ha permitido conocer las fortalezas y debilidades del producto. A continuación se ofrece la ficha técnica del cuestionario, así como las principales conclusiones. Los resultados completos pueden ser consultados en el Anexo D.

4.1.3.1. FICHA TÉCNICA

Nombre	Evaluación DED-AI
Autor	Javier Gómez Martínez
Objetivo	Conocer el grado de satisfacción en el manejo de la interfaz del cliente móvil de DED-AI
Tipo de cuestionario	Post-test
Área de aplicación	Usabilidad, accesibilidad, diseño de interfaz de usuario
Número de participantes	11
Edad de los participantes	Entre 18 y 70 años
Duración	Entre 3 y 5 minutos
Número de ítems	15
Estructura	<p>Consta de tres partes diferenciadas:</p> <ul style="list-style-type: none"> - Información del participante: 4 preguntas en las que se recogen la edad, sexo, y posibles discapacidades. - Funcionamiento general de la app: 4 preguntas escalares sobre la sencillez, velocidad y aspecto de la aplicación, con respuestas posibles del 1 (Muy difícil/lento/inadecuado) al 5 (Muy fácil/rápido/adecuado). - Impresiones de cada vista: 4 preguntas escalares sobre la sencillez y aspecto de las vistas individuales de la aplicación, con respuestas posibles del 1 (Muy difícil/inadecuado) al 5

	(Muy fácil/adecuado), - Dificultades y puntos fuertes y débiles de la aplicación, como preguntas abiertas.
--	---

Tabla 4.25 – Ficha técnica del cuestionario

4.1.3.2. PRINCIPALES CONCLUSIONES

La mayoría de los usuarios coincidieron en que la aplicación es simple y sencilla de utilizar, siendo a su vez éste el principal punto fuerte destacado por 5 de los 11 participantes. Ninguno indicó haber sufrido algún tipo de dificultad probando el producto. En el apartado *Funcionamiento general de la app*, todos los participantes ofrecieron una calificación igual superior a 4, salvo en la sencillez de lectura de los textos, donde un usuario dio una valoración de 3.

En cuanto a las vistas individualizadas, la única que generó mayor discrepancia fue la de feedback. Como se puede extraer a posteriori, varios usuarios apreciaron como insuficiente la oportunidad de participación que se les ofrece, sugiriendo poder añadir comentarios acerca de la localización del problema en la retina, o en definitiva, un cuadro de entrada de texto.

Otras sugerencias destacables incluyen implementar la memorización de las credenciales de usuario por parte de la app, para que éstas no tengan que ser introducidas en cada reinicio. Por último, un participante sugirió incorporar una guía para ayudar en el reconocimiento de posibles problemas visuales.

4.2. MODELO DE APRENDIZAJE AUTOMÁTICO

Para valorar los resultados del modelo de aprendizaje automático, en primer lugar se detallarán las métricas empleadas. Posteriormente, se detallarán los valores obtenidos para cada una de las métricas, tanto para el conjunto de datos de validación completo, como para cada uno de los grados de retinopatía definidos en la *Tabla 1.1*.

4.2.1. MÉTRICAS

La presente evaluación distingue dos tipos de métricas, directas y resumen.

4.2.1.1. DEFINICIONES

Como se ha explicado anteriormente, por cada imagen analizada, el modelo entrenado provee un arreglo con dos salidas: la probabilidad de que la imagen presente algún tipo de retinopatía, y la probabilidad del suceso contrario. Se considera, por tanto, que la imagen es **positiva** cuando la primera probabilidad es superior a la segunda, y **negativa** en caso contrario.

En lo sucesivo, se asignará la etiqueta *1* a las muestras positivas, y *0* a las negativas. Se considerará un arreglo *t*, donde *t* representa las etiquetas reales de todas las muestras analizadas. Por otro lado, *t'* será otro arreglo representando las etiquetas predichas por el modelo de aprendizaje.

Comparando ambos arreglos, *t* y *t'*, podemos distinguir las siguientes clases:

Clase	Etiqueta en <i>t</i>	Etiqueta en <i>t'</i>	Operación lógica
Verdadero positivo	1	1	$t \wedge t'$
Verdadero negativo	0	0	$\overline{t} \vee \overline{t'}$
Falso positivo	0	1	$\overline{t} \wedge t'$
Falso negativo	1	0	$t \wedge \overline{t'}$

Tabla 4.26: Tabla de verdad de un modelo de clasificación

4.2.1.2. DIRECTAS

- **Error (E):** número de discrepancias de *t* respecto a *t'*, en relación al tamaño de la muestra estudiada, *n*.

$$E = \frac{\sum_{i=1}^n t_i \oplus t'_{i}}{n} \quad (4.1)$$

- **Exactitud (A):** cantidad de aciertos respecto al tamaño de la muestra, *n*. También puede

ser definido como el complementario de E , es decir:

$$A = 1 - E \quad (4.2)$$

- **Tasa de verdaderos positivos (TPR):** número de verdaderos positivos en relación al tamaño de la muestra, n :

$$TPR = \frac{\text{Verdaderos positivos}}{n} \quad (4.3)$$

- **Tasa de verdaderos negativos (TNR):** total de verdaderos negativos entre n :

$$TPN = \frac{\text{Verdaderos negativos}}{n} \quad (4.4)$$

- **Tasa de falsos positivos (FPR):** cantidad de falsos positivos dividida entre n :

$$FPR = \frac{\text{Falsos positivos}}{n} \quad (4.5)$$

- **Tasa de falsos negativos (FNR):** número de falsos positivos entre

$$FNR = \frac{\text{Falsos negativos}}{n} \quad (4.6)$$

4.2.1.3. RESUMEN

- **Sensibilidad:** se define como la relación entre el número de verdaderos positivos, VP , y el total de positivos presentes en la muestra (es decir, verdaderos positivos y falsos negativos, FN). En el problema tratado, representa la capacidad del modelo de detectar todo caso positivo en retinopatía.

$$\text{Sensibilidad} = \frac{VP}{VP+FN} \quad (4.7)$$

- **Precisión:** indica la probabilidad de que un positivo detectado por el modelo, efectivamente, lo sea. Es decir, es la relación entre los verdaderos positivos, y el total de positivos detectados (verdaderos positivos más falsos positivos, FP).

$$\text{Precisión} = \frac{VP}{VP+FP} \quad (4.8)$$

- **F_1 Score:** combinación de las dos medidas anteriores. Ha sido tomada como referencia para evaluar la calidad del modelo, dadas las debilidades que presenta la tradicional métrica de exactitud. Si la precisión y sensibilidad presentan valores altos, el valor de F_1 Score lo será igualmente.

$$F_1 \text{ Score} = \frac{2 \times \text{Precisión} \times \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \quad (4.9)$$

4.2.2. RESULTADOS GENERALES

Como se detalla en la *Tabla 3.42*, el conjunto de datos de test presenta un total de 2792 muestras, balanceadas al 50% entre positivas y negativas. Las etiquetas 1, 2 y 3 observables en la tabla fueron unidas en una única categoría, *positivo*, tal y como se explicó en apartados anteriores.

De las 1396 muestras positivas, 1049 fueron detectadas como tal (verdaderos positivos), mientras que del mismo número de imágenes negativas, 1066 fueron correctamente clasificados, obteniendo así un F_1 Score del 75,60%. La ficha para el conjunto de test completo puede ser observada en la *Tabla 4.27*.

[RG] Conjunto de test completo	
Exactitud	75,75%
Sensibilidad	75,14%
Precisión	76,06%
F_1 Score	75,60%

Tabla 4.27: Resultados Generales – Conjunto de test completo

4.2.3. RESULTADOS ESPECÍFICOS

Las métricas anteriores serán desglosadas para cada uno de los grados de retinopatía existentes.

4.2.3.1. GRADO 1

El conjunto de test de retinopatía grado 1 cuenta con 742 imágenes, como puede observarse en la *Tabla 3.43*. De las 371 muestras positivas, 234 fueron detectadas por el modelo, mientras que de la misma cantidad de negativas, 208 fueron clasificadas como tal. Estas cifras arrojan un F_1 Score del 60,94%.

[RE-1] Retinopatía de Grado 1	
Exactitud	59,57%
Sensibilidad	63,07%
Precisión	58,95%
F_1 Score	60,94%

Tabla 4.28: Resultados Específicos 1 – Retinopatía de Grado 1

4.2.3.2. GRADO 2

En el caso del conjunto de retinopatía grado 2, se reservaron para test 1574 muestras (*Tabla 3.44*), balanceadas como en anteriores ocasiones. El modelo arroja 626 verdaderos positivos, y 620 verdaderos negativos, alcanzando un F_1 Score del 79,24%.

[RE-2] Retinopatía de Grado 2	
Exactitud	79,16%
Sensibilidad	79,54%
Precisión	78,94%
F_1 Score	79,24%

Tabla 4.29: Resultados Específicos 2 – Retinopatía de Grado 2

4.2.3.3. GRADO 3

274 imágenes se reservaron para test en el caso de grado 3. De los 137 positivos presentes (*Tabla 3.45*), se lograron detectar 125, y con el mismo número de negativos, 124 se clasificaron de forma correcta, dando lugar a un F_1 Score del 90,90%.

[RE-3] Retinopatía de Grado 3	
Exactitud	90,87%
Sensibilidad	91,24%
Precisión	90,56%
F_1 Score	90,90%

Tabla 4.30: Resultados Específicos 3 – Retinopatía de Grado 3

4.2.3.4. GRADO 4

En la última categoría, de acuerdo con la *Tabla 3.46*, existen 210 imágenes con esta etiqueta. 12 de los 105 positivos presentes fueron clasificados erróneamente, así como 13 de los negativos. En definitiva, se obtuvo un F_1 Score del 88,04%.

[RE-4] Retinopatía de Grado 4	
Exactitud	88,09%
Sensibilidad	87,61%
Precisión	88,47%
F_1 Score	88,04%

Tabla 4.31: Resultados Específicos 4 – Retinopatía de Grado 4

Conclusiones

Como cierre a este documento, se extraerán una serie de conclusiones sobre el proyecto desarrollado. Se repasarán las aportaciones realizadas, revisando el cumplimiento de los objetivos marcados. Se explorarán varias vías de evolución previsibles, tanto en expansión de la funcionalidad ofrecida, como en la mejora de la ya existente. Finalmente, se analizarán los problemas encontrados.

APORTACIONES REALIZADAS

El proyecto comenzó con el objetivo de construir un sistema de detección de la retinopatía diabética mediante inteligencia artificial. A esa idea primigenia pronto se le unió la posibilidad de construir una sencilla aplicación móvil que permitiese la evaluación de imágenes de fondo de ojo.

Se articuló así un conjunto de componentes cliente-servidor, que junto al modelo de aprendizaje automático desarrollado, han permitido crear una solución de cribado barata, rápida y accesible, suponiendo una alternativa a considerar respecto a las ya existentes en el mercado.

El proyecto ha sido implementado en su gran mayoría mediante tecnologías web, generalmente basadas en JavaScript, lo que permite su portabilidad y/o despliegue en una amplia variedad de plataformas, además de contar con un importante soporte tanto comercial como sin ánimo de lucro, en un ecosistema en constante mantenimiento y evolución.

Se puede afirmar sin ningún género de duda que se han cumplido todos los objetivos establecidos en el Documento Básico de Especificaciones, superándose incluso, al introducir mecanismos de seguridad como el flujo de autenticación por usuario y contraseña, así como al implementar mecanismos de interoperabilidad en la parte servidora, como la especificación OpenAPI.

EVOLUCIÓN PREVISIBLE

El cliente móvil ofrece desde sus versiones iniciales la posibilidad de tomar una captura de ojo mediante la propia cámara del dispositivo móvil. En este proyecto no se ha entrado a valorar cual es el método óptimo para la toma de estas imágenes. Al inicio, se hizo referencia al estudio en el que RAJALAKSHMI et al. [8] acoplaban cámaras de dispositivos móviles a material oftalmológico especializado para realizar fotografías de fondo de ojo. Para abaratar más el coste, una vía sería el estudio de lentes portátiles, que, acopladas mediante una carcasa u otro soporte, permitieran capturar el fondo de retina.

Otro asunto de interés sería conocer el grado de retinopatía detectado. Actualmente, la aplicación solo ofrece la probabilidad de que el ojo presente alguna retinopatía, sin especificar el estado de avance de la enfermedad. Esta situación obliga a realizar una revisión posterior por un profesional, no solo para confirmar los resultados, sino para poder prescribir el tratamiento más adecuado a las circunstancias del paciente.

Por último, no se ha hecho mención a la delicadeza de los datos tratados, de carácter sanitario, y recogidos en la categoría de *sensibles* según el Reglamento de Protección de Datos de la Unión Europea. Ya que está entre los objetivos del proyecto mejorar el modelo servido, utilizando la retroalimentación de los usuarios, pueden plantearse métodos de entrenamiento descentralizados, como extra de privacidad y seguridad. El aprendizaje federado [62] se presenta como la forma más viable de implementar esta *privacidad desde el diseño*, permitiendo un entrenamiento desde el propio dispositivo móvil, sin necesidad de que los datos tengan que ser almacenados en el servidor.

PROBLEMAS ENCONTRADOS

Los principales obstáculos se han encontrado a la hora de lograr un mejor resultado de aprendizaje. Existen modelos privativos con mayores niveles de sensibilidad y precisión, sin embargo, hay que tener en cuenta que muchos de ellos han sido entrenados con conjuntos de datos propios, con más trabajo de preparación y etiquetado detrás. Para aprovechar todo el potencial de los modelos de red convolucional, es casi una obligación contar con un *dataset* clasificado píxel por píxel. Se requeriría de la colaboración adicional de algún ente

de salud, público o privado, que accediese a compartir una muestra de capturas, para que posteriormente fuesen etiquetadas por un equipo multidisciplinar, resaltando las zonas de especial afección en la retina.

Este enfoque, junto al entrenamiento de modelos separados para cada grado de la enfermedad propuesto en el apartado anterior, conseguiría incrementar sustancialmente los resultados en los estados de menor avance, los más interesantes precisamente para alcanzar la deseada detección precoz que impida el avance de los síntomas.

Los recursos computacionales disponibles también han supuesto una barrera en ocasiones, con periodos de entrenamiento de más de 24 horas para el conjunto de datos completo y las 35 iteraciones finalmente fijadas. Gracias a las políticas de stop y guardado implementadas para evitar *over-fitting*, se consiguió mitigar el problema sin afectar al ritmo de trabajo planificado, sin embargo, si se generó cierto coste económico evitable si se hubiesen empleado recursos propios.

Una inicialización de pesos distinta también podría haber llevado a un mejor modelo de aprendizaje. Se optó por no utilizar el mecanismo de inicialización por defecto de PyTorch, del cual existe numeroso debate en torno a si respeta las buenas prácticas actuales en aprendizaje automático [63]. Se hace necesaria una mayor investigación por mi parte en este asunto que me permita desarrollar una solución más adecuada.

Agradecimientos

Al comenzar a redactar esta sección no puedo pensar en otra cosa distinta de la Universidad. Ya no solo como institución, sino como comunidad. Por todo lo que me ha dado en estos cuatro años de mi vida, los cuales han sido una auténtica montaña rusa de emociones y vivencias. Comenzando en un 2019 que parece de otro siglo, atravesando una pandemia que me hizo poner en valor la rutina del día a día, y desembocando en un 2023 donde la palabra *disrupción*, precisamente asociada a la inteligencia artificial, inunda cada rincón de los medios.

Al aterrizar en la Universidad tenía la sensación de empezar de cero, a pesar de seguir en mi ciudad, en León, la otra gran culpable de que yo sea como soy, y disfrute con ello. En ese *reset* particular conocí a Miguel y Pepe, los *gallegos* (aunque uno sea de Ponfe). Y junto a ellos, a Josema, Luisfer, Sergio, Rober, *Kariku* y Josemi. Es imposible olvidar todos los momentos en ese piso, de fiesta, o de casa rural intentando representar a un taxidermista en el *Party* a las 4 de la mañana.

Lo cierto es que de cero no empecé, y de mi etapa preuniversitaria también vino a hacerme compañía Hugo. A nosotros rápidamente se nos unió Rodrigo, formando *el trío calavera*. Y por si fuéramos pocos, también apareció Abel, sin el cual no podría haber lucido en varias ocasiones un nuevo *look* de pelo. Cierra el grupo Samu, quien me ha salvado en más de una ocasión, y de quien estoy seguro le espera un futuro impresionante.

Esa montaña rusa de velocidad variable ha superado mis fuerzas en alguna que otra ocasión, y ahí, siempre ha estado mi madre, escuchándome y dándome su apoyo cuando trataba de controlar la dirección del vagón. También mi padre, para quien estaré siempre que lo necesite, aunque en ocasiones me tome el permiso de bromear con ello.

En el mismo parque de atracciones, pero en una montaña *eléctrica*, ha estado Jimmy. Ha habido de todo: éxitos, fracasos, viajes al otro lado del charco... Pero siempre ha habido dos cosas, apoyo y lealtad. Decir esto de un amigo es complicado para mucha gente.

Comisión, Club de Debate, Delegación, Radio Universitaria... es difícil recapitular todos los sitios por los que he pasado en estos cuatro años. En el último, un día se me ocurrió prestar unos altavoces, y que me los devolvieran a la una de la mañana. Ahí conocí a un loco llamado Pablo, el *socio corporativo*. Y junto a Jorge, es increíble lo que hemos logrado en

este último año. Espero que no desistamos, y que como uno de vosotros dijo, que *los sueños cada día estén un poquito más cerca.*

En definitiva, GRACIAS a la VIDA, porque sin duda, acabo el Grado con las manos llenas de todo lo que uno desea: BUENA GENTE.

Lista de referencias

- [1] Observatorio HP 2022-2023 - HP SCDS [en línea], (sin fecha). HP SCDS. Disponible en: <https://hpscdis.com/observatorio-tecnologico/observatorio-hp-22-23/>
- [2] Montoto, R., (2022). Más de 1,5 millones de españoles padecen diabetes sin saberlo: qué hacer para evitarlo [en línea]. The Objective. Disponible en: <https://theobjective.com/sociedad/2022-08-08/diabetes-espanoles/>
- [3] Vítreo y Retina - IOBA | Instituto Universitario de Oftalmobiología Aplicada [en línea], (sin fecha). IOBA | Instituto Universitario de Oftalmobiología Aplicada. Disponible en: <https://www.ioba.es/especialidades/vitreo-y-retina/>
- [4] Nagpal, D., Panda, S. N., Malarvel, M., Pattanaik, P. A. y Zubair Khan, M., (2021). A Review of Diabetic Retinopathy: datasets, approaches, evaluation metrics and future trends. Journal of King Saud University - Computer and Information Sciences [en línea]. Disponible en: doi: 10.1016/j.jksuci.2021.06.006
- [5] Castilla Y León 2023 [en línea], (sin fecha). Datosmacro.com. Disponible en: <https://datosmacro.expansion.com/ccaa/castilla-leon>
- [6] Calvo, J., (2023). Médicos y sanitarios de Primaria exigen más medios para atender 600 consultorios locales [en línea]. leonoticias.com. Disponible en: <https://www.leonoticias.com/leon/medicos-sanitarios-primaria-exigen-medios-atender-600-20230412171827-nt.html>
- [7] Pratt, H., Coenen, F., Broadbent, D. M., Harding, S. P. y Zheng, Y., (2016). Convolutional Neural Networks for Diabetic Retinopathy. Procedia Computer Science [en línea]. 90, 200–205. Disponible en: doi: 10.1016/j.procs.2016.07.014
- [8] Rajalakshmi, R., Subashini, R., Anjana, R. M. y Mohan, V., (2018). Automated diabetic retinopathy detection in smartphone-based fundus photography using artificial intelligence. Eye [en línea]. 32(6), 1138–1144. Disponible en: doi: 10.1038/s41433-018-0064-9
- [9] OpenAPI Specification v3.1.0 | Introduction, Definitions, & More [en línea], (sin fecha). Home | OpenAPI Initiative Registry. Disponible en: <https://spec.openapis.org/oas/v3.1.0>
- [10] OAuth 2.0 Password Grant Type [en línea], (sin fecha). OAuth Community Site.

Disponible en <https://oauth.net/2/grant-types/password/>

[11] Sueldo de Desarrollador/a de software en España [en línea], (sin fecha). Indeed.com. Disponible en: <https://es.indeed.com/career/desarrollador-de-software/salaries>

[12] Sueldo de Analista de sistemas en España [en línea], (sin fecha). Indeed.com. Disponible en: <https://es.indeed.com/career/analista-de-sistema/salaries>

[13] Sueldo de Técnico de sistemas en España [en línea], (sin fecha). Indeed.com. Disponible en: <https://es.indeed.com/career/t%C3%A9cnico-de-sistemas/salaries>

[14] Sueldo de Data scientist en España [en línea], (sin fecha). Indeed.com. Disponible en: https://es.indeed.com/career/data-scientist/salaries?from=top_sb

[15] Sueldo de Scrum master en España [en línea], (sin fecha). Indeed.com. Disponible en: <https://es.indeed.com/career/scrum-master/salaries>

[16] Configurador PC | Tu por piezas [en línea], (sin fecha). PCComponentes. Disponible en <https://www.pccomponentes.com/configurador>

[17] Google Pixel 4a 5G Factory Unlocked Smartphone [en línea], (sin fecha). Amazon. Disponible en <https://www.amazon.es/Google-Pixel-Factory-Unlocked-Smartphone/dp/B08JLW6VK8>

[18] Comprar Windows 11 Pro - Microsoft Store es-ES [en línea], (sin fecha). Microsoft Store. Disponible en: <https://www.microsoft.com/es-es/d/windows-11-pro/dg7gmgf0d8h4>

[19] Nvidia Quadro RTX A6000 48 GB [en línea], (sin fecha). PCComponentes. Disponible en <https://www.pccomponentes.com/qty-nvidia-quadro-rtx-a6000-48gb-gddr6>

[20] Google Colab [en línea], (sin fecha). Google Colab. Disponible en: <https://colab.research.google.com/signup>

[21] Taxonomy-Based Risk Identification [en línea], (sin fecha). SEI Digital Library. Disponible en: <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=11847>

[22] React Native · Learn once, write anywhere [en línea], (sin fecha). React Native · Learn once, write anywhere. Disponible en: <https://reactnative.dev/>

[23] Express - Node.js web application framework [en línea], (sin fecha). Express - Node.js web application framework. Disponible en: <https://expressjs.com/>

[24] Expo [en línea], (sin fecha). Expo. Disponible en: <https://expo.dev/>

- [25] React Native Paper [en línea], (sin fecha). React Native Paper. Disponible en: <https://reactnativepaper.com/>
- [26] Node.js [en línea], (sin fecha). Node.js. Disponible en: <https://nodejs.org/en>
- [27] Axios [en línea], (sin fecha). Axios. Disponible en: <https://axios-http.com/es/>
- [28] express-openapi [en línea], (sin fecha). npm. Disponible en: <https://www.npmjs.com/package/express-openapi>
- [29] GitHub - node-oauth/node-oauth2-server: The unofficial successor to oauthjs/oauth2-server. Complete, compliant, maintained and well tested OAuth2 Server for node.js. Includes native async await and PKCE. (s.f.). GitHub. <https://github.com/node-oauth/node-oauth2-server>
- [30] Email Delivery, API, Marketing Service. (s.f.). SendGrid. <https://sendgrid.com/>
- [31] Welcome to Flask — Flask Documentation (2.3.x). (s.f.). Flask Documentation (2.3.x). <https://flask.palletsprojects.com/en/2.3.x/>
- [32] MongoDB: The Developer Data Platform. (s.f.). MongoDB. <https://www.mongodb.com/>
- [33] Mongoose ODM v7.3.0. (s.f.). Mongoose ODM v7.3.0. <https://mongoosejs.com/>
- [34] PyTorch. (s.f.). PyTorch. <https://pytorch.org/>
- [35] pandas - Python Data Analysis Library. (s.f.). pandas. <https://pandas.pydata.org/>
- [36] Welcome to Segmentation Models's documentation! — Segmentation Models documentation. (s.f.). Segmentation Models documentation. <https://smp.readthedocs.io/en/latest/>
- [37] Home. (s.f.). OpenCV. <https://opencv.org/>
- [38] Google Colaboratory. (s.f.). Google Colab. <https://colab.research.google.com/>
- [39] ISO 8601 — Date and time format. (s.f.). ISO. <https://www.iso.org/iso-8601-date-and-time-format.html>
- [40] Diabetic Retinopathy Detection | Kaggle. (s.f.). Kaggle: Your Machine Learning and Data Science Community. <https://www.kaggle.com/competitions/diabetic-retinopathy-detection/overview>
- [41] California Health Care Foundation - Health Care That Works for All Californians. (s.f.). California Health Care Foundation. <https://www.chcf.org/>
- [42] Diabetic Retinopathy Screening - EyePACS. (s.f.). EyePACS.

<https://www.eyepacs.com/>

[43] Kaggle DR dataset (EyePACS). (s.f.). Kaggle: Your Machine Learning and Data Science Community. <https://www.kaggle.com/datasets/mariaherrerot/eyepacspreprocess>

[44] 8.2. Matriz de convolución. (s.f.). GIMP Documentation. <https://docs.gimp.org/2.6/es/plug-in-convmatrix.html>

[45] Campana de Gauss. Distribución normal o de Gauss. Ejemplos de distribución normal. Características de una distribución normal y de la curva normal. Propiedades de la función densidad. (s.f.). Calculo.cc. https://calculo.cc/temas/temas_estadistica/binomial_normal/teoria/normal.html

[46] OpenCV: Smoothing Images. (s.f.). OpenCV documentation index. https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

[47] OpenCV: Operations on arrays. (s.f.). OpenCV documentation index. [https://docs.opencv.org/3.4/d2/de8/group_core_array.html#gafafb2513349db3bcff51f54ee5592a19](https://docs.opencv.org/3.4/d2/de8/group__core__array.html#gafafb2513349db3bcff51f54ee5592a19)

[48] OpenCV: Geometric Image Transformations [en línea], (sin fecha). OpenCV documentation index. Disponible en: https://docs.opencv.org/3.4/da/d54/group_imgproc_transform.html

[49] U-Net: Convolutional Networks for Biomedical Image Segmentation [en línea], (sin fecha). Computer Vision Group, Freiburg. Disponible en: <https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/>

[50] He, K., Zhang, X., Ren, S., Sun, J., (2015). Deep Residual Learning for Image Recognition. arXiv. Disponible en: <https://doi.org/10.48550/ARXIV.1512.03385>.

[51] Mahajan, P., (2020). Understanding Residual Network (ResNet) Architecture [en línea]. Medium. Disponible en: <https://medium.com/analytics-vidhya/understanding-resnet-architecture-869915cc2a98>

[52] Pelletier, C., Webb, G. y Petitjean, F., (2019). Temporal Convolutional Neural Network for the Classification of Satellite Image Time Series. Remote Sensing [en línea]. 11(5), 523. Disponible en: doi: 10.3390/rs11050523

[53] ImageNet [en línea], (sin fecha). ImageNet. Disponible en: <https://www.image-net.org/>

[54] Kingma, D.P. y Ba, J. (2014) Adam: A Method for Stochastic Optimization. arXiv.

Disponible en: <https://doi.org/10.48550/ARXIV.1412.6980>.

[55] HP SCDS – Observatorio 22/23 – ULE-DED-AI [en línea], (sin fecha). Disponible en <https://gitlab.com/HP-SCDS/Observatorio/2022-2023/ded-ai/ule-ded-ai/> Copia disponible en https://github.com/igomem04/DED_AI

[56] Helmet.js [en línea], (sin fecha). Helmet.js. Disponible en: <https://helmetjs.github.io/>

[57] Swagger UI [en línea], (sin fecha). Disponible en: <https://ded-ai-api-server.onrender.com/api-documentation/>

[58] DED-AI-Model-Server [en línea], (sin fecha). Disponible en: <https://dedaimodel.onrender.com/>

[59] Gunicorn - Python WSGI HTTP Server for UNIX [en línea], (sin fecha). Gunicorn. [Consultado el 16 de junio de 2023]. Disponible en: <https://gunicorn.org/>

[60] ISO 9241-210:2010 [en línea], (sin fecha). ISO. Disponible en: <https://www.iso.org/standard/52075.html>

[61] Cuestionario de Evaluación DED-AI [en línea], (sin fecha). Google Docs. Disponible en: <https://forms.gle/iXpj9jPS5jmoEkMg6>

[62] Federated Learning [en línea], (sin fecha). Google. Disponible en: <https://federated.withgoogle.com/>

[63] Update weight initialisations to current best practices · Issue #18182 · pytorch/pytorch [en línea], (sin fecha). GitHub. Disponible en: <https://github.com/pytorch/pytorch/issues/18182>

Anexo A: Control de Versiones

Tanto en la fase de desarrollo como en la de pruebas y documentación, se ha empleado un controlador de versiones para llevar un registro de los cambios efectuados en el proyecto. El controlador utilizado ha sido Git, y para hospedar el repositorio remoto, se ha empleado GitLab, al ser esta la plataforma ofrecida desde HP SCDS. Cabe destacar que GitLab no es únicamente un repositorio remoto, sino que ofrece diversas herramientas para la gestión de las operaciones de desarrollo (*DevOps*). Entre otras funciones de utilidad, GitLab implementa un tablero *scrum* (ver *Figura Anexo A.1*) para seguir el estado de las historias de usuario, permitiendo una rápida asociación entre actualizaciones (*commits*) del repositorio y funcionalidad implementada.

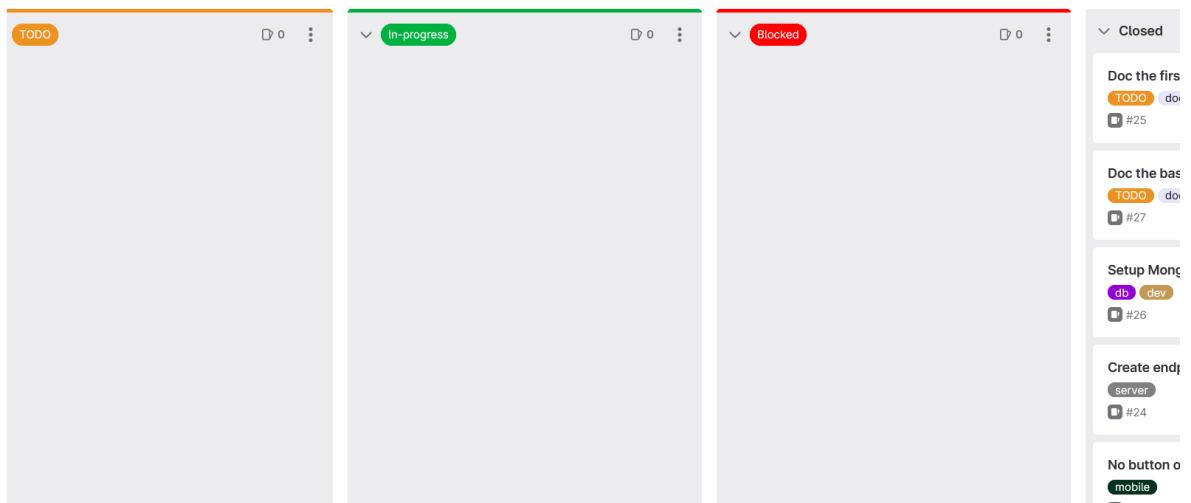


Figura Anexo A.1: Tablero scrum al finalizar el proyecto (Fuente: GitLab)

Git permite el manejo de diferentes ramas, característica especialmente útil cuando se trabaja en diferentes etapas de desarrollo, o el equipo de proyecto está formado por varias personas. Sin embargo, ya que el proyecto ha sido desarrollado por una única persona, no se ha considerado que tenga la suficiente complejidad como para requerir ramificaciones, trabajándose en una única rama principal. A continuación, en la *Figura Anexo A.2* se ofrece la estadística de commits de la rama *main*.

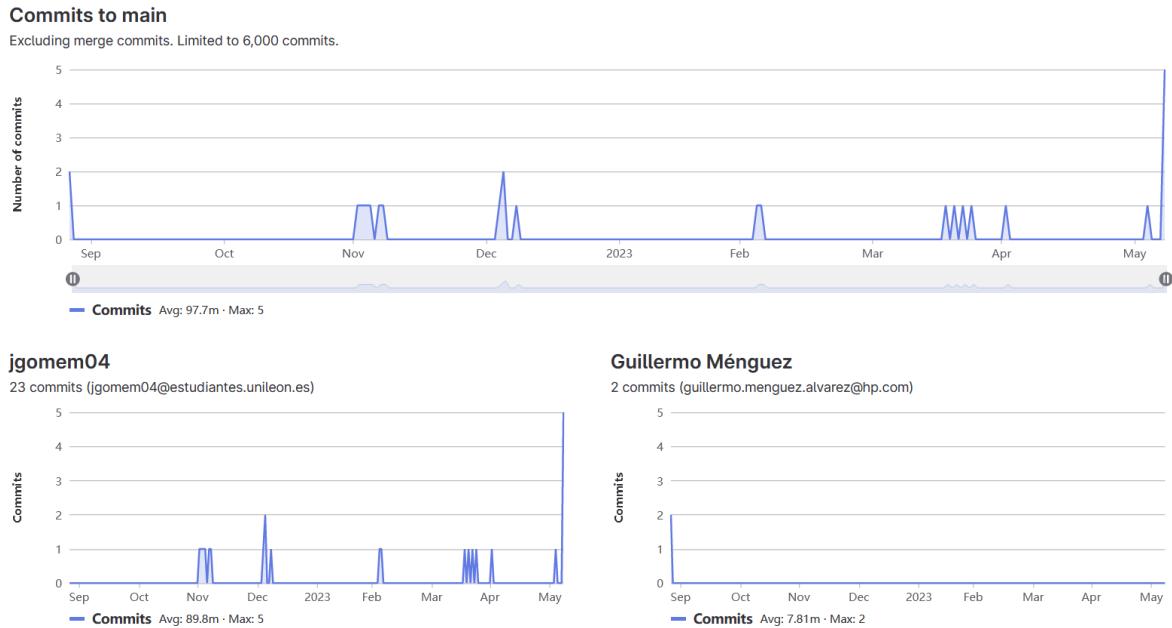
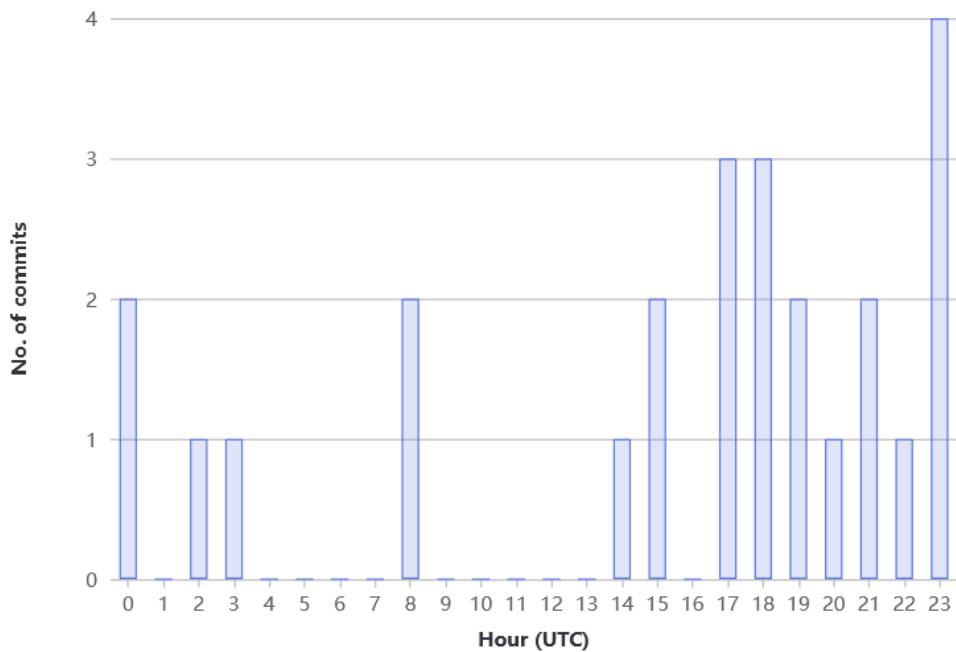


Figura Anexo A.2: Estadísticas de la rama main (Fuente: GitLab)

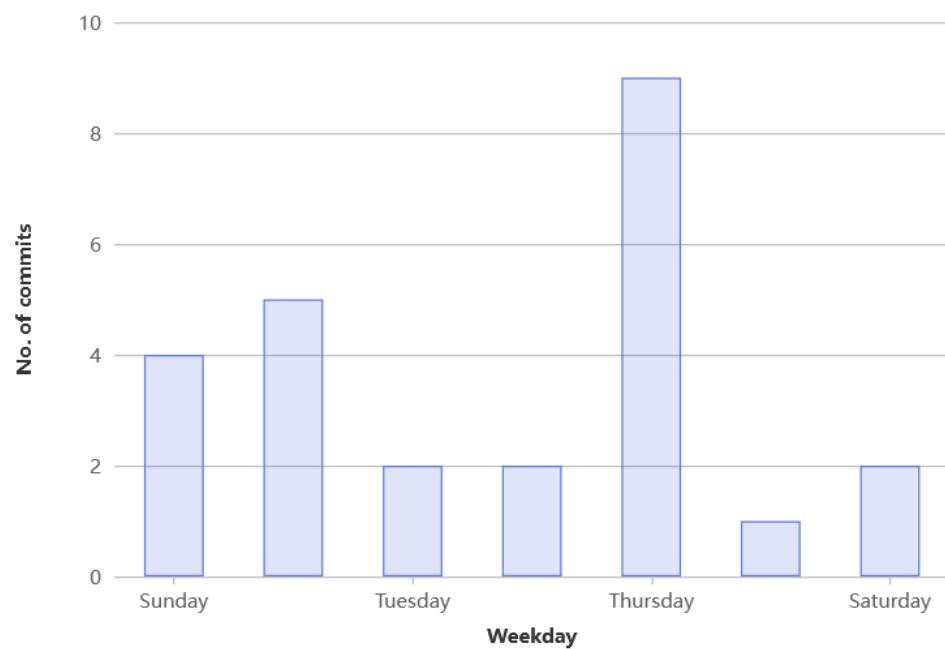
Puede observarse un contribuyente adicional, Guillermo Ménquez, Director del Programa Observatorio de HP SCDS, y que únicamente realizó los dos *commits* iniciales para configurar el repositorio. La actividad se concentra entre noviembre y junio, coincidente con las fases de desarrollo y documentación/pruebas. Al final del gráfico se observa un día de gran actividad, en el que se actualizaron varios ficheros de cara al entregable final.

A continuación, en las *Figuras Anexo A.3* y *Anexo A.4*, se ofrecen dos gráficas adicionales, con las estadísticas por horas y días. En el primero de los casos, se aprecia una mayor actividad durante la tarde y primeras horas de la noche, de acuerdo con mi estructura habitual de trabajo. En el segundo gráfico, se recogen como días más activos los jueves, sábados, domingos, y lunes, usualmente los días previos a las reuniones de retrospectiva.

Commits per day hour (UTC)

*Figura Anexo A.3: Gráfico de actividad por horas (Fuente: GitLab)*

Commits per weekday

*Figura Anexo A.4: Gráfico de actividad por días de la semana (Fuente: GitLab)*

Anexo B: Historias

En este anexo se describen los requisitos de la solución en forma de historia, siguiendo las recomendaciones de *scrum*. Téngase en cuenta que las historias han sido escritas en lenguaje inglés por requerimiento del tutor, José Luís Vidal. En todas ellas pueden distinguirse las siguientes partes:

- **Título:** breve resumen en una sola frase de lo que se desea implementar.
- **Descripción:** detalle de las características que el sistema debe poseer, en un lenguaje claro y próximo al utilizado por el cliente.
- **Criterios de aceptación:** elementos que sirvan para comprobar que la historia ha sido correctamente implementada.
- **Etiquetas:** *server* si la historia está relacionada con el servidor REST, *mobile* para las historias correspondientes a la aplicación móvil, *db* si se tratan asuntos en relación a la base de datos, *ml* para el modelo de aprendizaje automático, y *poc* para las pruebas de concepto.

SPRINT 1

[H-1]: Setup ExpressJS Environment	
Descripción	As a developer, I need to setup the ExpressJS environment, so I am able to develop and run the backend server
Criterios de aceptación	- Node JS package.json file is created - Main Express app is created - Main exposes GET endpoint with a <i>Hello World</i> response
Etiquetas	server

Tabla Anexo B.1: Historia 1 - Setup ExpressJS Environment

[H-2]: Setup React Native CLI and dependencies	
Descripción	As a developer, I need to setup React Native CLI and all of its dependencies, so I am able to

	create and run the mobile app
Criterios de aceptación	<ul style="list-style-type: none"> - Node JS is installed on the development machine - JDK is installed on the dev machine - Android Studio is installed on the dev machine - Environment variables of Android's SDK are set up - React Native project is created
Etiquetas	mobile

Tabla Anexo B.2: Historia 2 - Setup React Native CLI and dependencies

[H-3]: Setup MongoDB local database	
Descripción	As a developer, I need to setup a MongoDB database on my local machine, so I can store all data related to this project
Criterios de aceptación	<ul style="list-style-type: none"> - MongoDB is installed on dev machine - A connection to the database can be established
Etiquetas	db

Tabla Anexo B.3: Historia 3 - Setup MongoDB local database

[H-4]: Create endpoint for analysis request	
Descripción	As a user, I want to POST an image to the server, so I can know if retinopathy exists or not
Criterios de aceptación	<ul style="list-style-type: none"> - An endpoint is created in Express server - The endpoint receives the image data sent - The endpoint returns a JSON object - The endpoint returns HTTP 200 OK code if process goes OK - The endpoint returns HTTP 400 Bad Request if the request is not properly formed - The endpoint returns HTTP 500 Internal Server Error if other type of exception is thrown
Etiquetas	server

Tabla Anexo B.4: Historia 4 - Create endpoint for analysis request

SPRINT 2

[H-5]: Welcome scene	
Descripción	As a user, I want to see a welcome page on the mobile app, so I can easily find all the features of the app gathered in one place
Criterios de aceptación	- A page with a <i>Welcome</i> title is shown when opening the app - A brief description of what the user can do next is shown under the title - A camera button is shown under the description - Camera functionality is launched when clicking the button - A gallery button is shown behind the description of the welcome page - Image finder is opened when clicking on the button
Etiquetas	mobile

Tabla Anexo B.5: Historia 5 - Welcome scene

[H-6]: Preview scene	
Descripción	As a user, I want to preview the selected sample, so I can be sure it is the image I wanted to select
Criterios de aceptación	- A page with a <i>Preview</i> title is shown after selecting/shooting an image - The selected shot is shown behind the title - A <i>Discard</i> button is shown at left bottom of the screen - The button contains a cross icon - Welcome page is displayed when clicking <i>Discard</i> - A <i>Confirm</i> button is shown at right bottom of the screen - Button contains a <i>Check</i> icon
Etiquetas	mobile

Tabla Anexo B.6: Historia 6 - Preview scene

[H-7]: Processing scene	
Descripción	As a user,

	I want to notice when the image is being processed, so I am aware that a prediction result is not ready yet
Criterios de aceptación	<ul style="list-style-type: none"> - A new scene is opened after selecting Yes on <i>Preview</i> scene - A <i>Processing</i> title is shown at the top of the screen - An animated circle progress bar is shown behind the title - The scene is shown until server response is got
Etiquetas	mobile

Tabla Anexo B.7: Historia 7 - Processing scene

[H-8]: Error scene	
Descripción	<p>As a user,</p> <p>I want to notice when something goes wrong, so I am aware that a conclusive result won't be delivered</p>
Criterios de aceptación	<ul style="list-style-type: none"> - An <i>Error</i> title is shown at the top of the screen - The text “An error occurred while (<i>step where the error occurs</i>). Please try again or start from the beginning” is shown under the title - A <i>Retry</i> button is shown at the left bottom of the screen - The button contains a <i>refresh</i> icon - A <i>Cancel</i> button is shown at the right bottom of the screen - The button contains a <i>cross</i> icon - Welcome scene is displayed when clicking <i>Cancel</i>
Etiquetas	mobile

Tabla Anexo B.8: Historia 8 - Error scene

[H-9]: Results scene	
Descripción	<p>As a user,</p> <p>I want to see the result of processing the image, so I can see if the sample is potentially affected by a retinopathy or not</p>
Criterios de aceptación	<ul style="list-style-type: none"> - The view is opened after the <i>Processing</i> scene - A <i>Positive/Negative</i> title is shown on the screen - A brief description of the result is shown under the title - A <i>Next</i> button is shown at the bottom of the screen - The button contains a <i>right arrow</i> icon

Etiquetas	mobile
------------------	--------

Tabla Anexo B.9: Historia 9 – Results scene

[H-10]: Deploy ExpressJS server	
Descripción	As a developer, I need to have my backend server on a cloud machine, so I can access it whenever from the Internet
Criterios de aceptación	- A cloud machine is got from a hosting provider - The machine is identified by an URL, or a fixed IP, otherwise - A ping can be done from the dev machine to the cloud server - A request to the /analysis path returns HTTP 200 OK, together with a <i>Hello World</i> message
Etiquetas	server

Tabla Anexo B.10: Historia 10 – Deploy ExpressJS server

SPRINT 3

[H-11]: Feedback scene	
Descripción	As a user, I want to be offered the opportunity to give feedback, so the algorithm can be improved in the future
Criterios de aceptación	- A <i>Feedback</i> scene is shown clicking <i>Next</i> in <i>Results</i> scene - The text “Does the result coincide with the medical diagnosis?” is shown under the title - A <i>Yes</i> button is shown at the left bottom of the screen - The button contains a <i>check</i> icon - A <i>No</i> button is shown the right bottom of the screen - The button contains a <i>cross</i> icon - Both buttons launch the <i>Welcome</i> scene
Etiquetas	mobile

Tabla Anexo B.11: Historia 11 – Feedback scene

[H-12]: Thanks scene	
Descripción	As a user, I want to be thanked for the use of the app when all the workflow is finished
Criterios de aceptación	- A scene with <i>Thanks</i> title is shown after giving feedback - The text “Thank you. Your collaboration helps us improve DED-AI” is shown below the title. - A <i>Return</i> button is shown at the bottom - <i>Welcome</i> scene is launched when clicking the button
Etiquetas	mobile

Tabla Anexo B.12: Historia 12 – *Thanks scene*

[H-13]: Create endpoint to save feedback	
Descripción	As a user, I want to PUT my feedback on the received prediction, so the analysed imaged is properly classified for future improvements of the algorithm
Criterios de aceptación	- A PUT endpoint is created in Express server - The expects a JSON object in the request - The server classifies the image with the received ID in the DB - The endpoint returns HTTP 200 OK code if process goes OK - The endpoint returns HTTP 400 Bad Request if the request is not properly formed - The endpoint returns HTTP 500 Internal Server Error if other type of exception is thrown
Etiquetas	server

Tabla Anexo B.13: Historia 13 – *Create endpoint to save feedback*

[H-14]: Define model architecture and training parameters	
Descripción	As a user, I want to have a machine learning model architecture defined, so I can train it to detect retinopathy-affected images
Criterios de aceptación	- A Jupyter Notebook for the ML model is created - A <i>Dataset</i> class is created using PyTorch, representing the training data - A <i>Net</i> class is created using PyTorch, implementing a convolutional neural

	<p>network (CNN) architecture</p> <ul style="list-style-type: none"> - A loss function is defined - An optimizer is defined - Batch size, learning rate and number of epochs are established
Etiquetas	ml

Tabla Anexo B.14: Historia 14 – Define model architecture and training parameters

[H-15]: Train and validate ML model	
Descripción	As a user, I want to train a machine learning model, so I can use it to detect new patients affected by any retinopathy
Criterios de aceptación	<ul style="list-style-type: none"> - Loss and accuracy values have improved substantially after all epochs - A confusion matrix is obtained - Recall and F1 Score are obtained - A PyTorch .pt file containing the trained weights is saved
Etiquetas	ml

Tabla Anexo B.15: Historia 15 – Train and validate ML model

SPRINT 4

[H-16]: Setup express-openapi and dependencies	
Descripción	As a developer, I need to setup express-openapi and its dependencies, so I can build a compliant OpenAPI server
Criterios de aceptación	<ul style="list-style-type: none"> - Express-openapi is included in app.js requirements - Swagger is included in app.js requirements - OpenAPI initialization function is defined in app.js
Etiquetas	server

Tabla Anexo B.16: Historia 16 – Setup express-openapi and dependencies

[H-17]: Create OpenAPI basic doc file	
Descripción	As a user,

	I want to have the server properly documented under the OpenAPI spec, so it can be easier for both humans and machines to get info about it
Criterios de aceptación	<ul style="list-style-type: none"> - api-doc.js file is created on Express server - Doc file defines the different objects handled by the API - Doc file includes basic metadata about the API
Etiquetas	server

Tabla Anexo B.17: Historia 17 – Create OpenAPI basic doc file

[H-18]: Adapt analysis route to OpenAPI specs	
Descripción	As a user, I want the server routes to fulfil the OpenAPI specification, so it can be easier for both humans and machines to know how they work
Criterios de aceptación	<ul style="list-style-type: none"> - All endpoints are grouped inside a <i>paths</i> folder - The endpoints are named by their HTTP method (POST, PUT, GET, DELETE) - Each endpoint has an OpenAPI-styled doc parameter - Doc offers a brief description of the endpoints - Doc specifies which objects can be received and return by the endpoint - Doc lists all the HTTP responses of the method
Etiquetas	server

Tabla Anexo B.18: Historia 18 – Adapt analysis route to OpenAPI specs

[H-19]: Create Token schema	
Descripción	As a user, I want to store all the session tokens in the database, so I can manage the access to the application
Criterios de aceptación	<ul style="list-style-type: none"> - A new mongoose schema is created - The schema defines a String parameter to store the token - Schema defines a Date parameter to store the expiration date of the token - Schema allows to store a reference to clients and users implied in the authentication
Etiquetas	db

Tabla Anexo B.19: Historia 19 – Create Token schema

[H-20]: Create Sample schema	
Descripción	As a user, I want to store sample-related data on the database, so I can retrieve it in the future
Criterios de aceptación	<ul style="list-style-type: none"> - A new mongoose schema is created - The schema defines a unique key, a String representing the fileName - Schema defines a Date parameter to store the analysis date - Schema requires the result of the analysis, stored as a Boolean parameter - Schema allows storing the feedback offered by the user, as a Boolean parameter - A reference to the user that requested the analysis is defined
Etiquetas	db

Tabla Anexo B.20: Historia 20 – Create Sample schema

[H-21]: Create User schema	
Descripción	As a user, I want to store the info about who can access the application, so an access can be granted
Criterios de aceptación	<ul style="list-style-type: none"> - A new mongoose schema is created - The schema defines a String parameter for the username - The username must be trimmed - Schema defines a String parameter for the password - Schema defines a String parameter to store an email address - The email must be trimmed and in lowercase
Etiquetas	db

Tabla Anexo B.21: Historia 21 – Create User schema

[H-22]: Create Client schema	
Descripción	As a user, I want to store the info about the apps that can communicate with the server, so no other solution can be used to access
Criterios de aceptación	<ul style="list-style-type: none"> - A new mongoose schema is created - Schema defines a unique key, a String parameter representing the client ID

	<ul style="list-style-type: none"> - Schema defines a String parameter to store client's secret - Schema allows to store which grants can be given to a specific client - Schema allows to store redirect URLs
Etiquetas	db

Tabla Anexo B.22: Historia 22 – Create Client schema

[H-23]: Setup MongoDB Atlas environment	
Descripción	As a developer, I need to setup a MongoDB Atlas cluster, so I can store this db on cloud and access it from the Internet
Criterios de aceptación	<ul style="list-style-type: none"> - MongoDB cluster is created on Atlas platform - Cluster URI is got - Cluster can be accessed using a local Compass installation, or from the backend server
Etiquetas	db

Tabla Anexo B.23: Historia 23 – Setup MongoDB Atlas environment

[H-24]: Setup OAuth2 server and helpers	
Descripción	As a user, I want my server to implement an OAuth2 password flow, so I can control the access of the application using a username and a password.
Criterios de aceptación	<ul style="list-style-type: none"> - OAuth2Server is installed on Express - app.js requires OAuth2Server - Password authentication is defined on OpenAPI initialization - A helper to get or validate session tokens is implemented
Etiquetas	server

Tabla Anexo B.24: Historia 24 – Setup OAuth2 server and helpers

[H-25]: Create login endpoint	
Descripción	As a user, I want to POST my login credentials, so I can be granted access to the application
Criterios de	- An endpoint is created in Express server

Aceptación	<ul style="list-style-type: none"> - The endpoint receives an object containing the username and password - The endpoint returns a JSON object containing a session token - The endpoint returns HTTP 200 OK code if process goes OK - The endpoint returns HTTP 401 Unauthorized if the credentials don't match with any of the stored in the database - The endpoint returns HTTP 400 Bad Request if the request is not properly formed - The endpoint returns HTTP 500 Internal Server Error if other type of exception is thrown
Etiquetas	server

Tabla Anexo B.25: Historia 25 – Create login endpoint

SPRINT 5

[H-26]: Login scene	
Descripción	As a user, I want to have a login page on the app, so I can identify myself with my user and password
Criterios de aceptación	<ul style="list-style-type: none"> - A page with a <i>Login</i> title is shown when opening the app - A text input with for the username is shown under the title - A password field is shown afterwards - A <i>Continue</i> button is shown at the bottom of the screen - An authentication request is sent when pressing <i>Continue</i> - <i>Welcome</i> scene is shown after a successful authentication - Empty fields are marked in red
Etiquetas	mobile

Tabla Anexo B.26: Historia 26 – Login scene

[H-27]: Setup mongoose connection config	
Descripción	As a developer, I need to create the mongoose configuration file, so I can properly establish a connection between the server and the database
Criterios de	- A db.js file is created

Aceptación	<ul style="list-style-type: none"> - db.js is required on the main app.js file - A connection method and its parameters are defined - The method is called in app.js - If the connection is established, “MongoDB connected” is printed to the console log
Etiquetas	server, db

Tabla Anexo B.27: Historia 27 – Setup mongoose connection config

[H-28]: Setup HTTP header protections using Helmet	
Descripción	As a developer, I need to setup Helmet, so I can protect the Express server against different types of attacks
Criterios de aceptación	<ul style="list-style-type: none"> - Helmet is installed and appears in package.json - app.js requires and uses Helmet
Etiquetas	server

Tabla Anexo B.28: Historia 28 – Setup HTTP header protections using Helmet

SPRINT 6

[H-29]: Update to Unet++ architecture	
Descripción	As a developer, I want to update to Unet++ architecture, so I can improve the model’s detection of retinopathy
Criterios de aceptación	<ul style="list-style-type: none"> - A Net class is defined using PyTorch - Net implements the Unet++ architecture
Etiquetas	ml

Tabla Anexo B.29: Historia 29 – Update to Unet++ architecture

[H-30]: Define separated train and test datasets for each retinopathy grade	
Descripción	As a developer, I need to define separate train and test datasets for each retinopathy grade, so I can compare the usefulness of the model against each grade

Criterios de aceptación	<ul style="list-style-type: none"> - Python script files are created - Each file reads the complete dataset - The scripts separate each retinopathy grade - The separated sets are saved into new CSV files
Etiquetas	ml

Tabla Anexo B.30: Historia 30 – Define separated train and test datasets for each retinopathy grade

[H-31]: Equilibrate train datasets	
Descripción	As a developer, I need to equilibrate the train datasets, so I can get better results on training
Criterios de aceptación	<ul style="list-style-type: none"> - Python script files are created - Train datasets are read by the scripts - Negative samples are inserted to compensate as much as needed - New sets are saved into CSV files
Etiquetas	ml

Tabla Anexo B.31: Historia 31 – Equilibrate train datasets

[H-32]: Implement blur and mean colour subtraction preprocess	
Descripción	As a developer, I need to implement image pre-processing based on blurring and removing mean colour, so vascularization of the eye is highlighted
Criterios de aceptación	<ul style="list-style-type: none"> - OpenCV library is included in Jupyter Notebook - A preprocess method is called in the dataset - The method uses OpenCV to blur the image - The blur image is used to subtract local mean colour - The final image is returned to the dataset
Etiquetas	ml

Tabla Anexo B.32: Historia 32 – Implement blur and mean colour subtraction preprocess

[H-33]: [Proof of Concept] Change training optimizer	
Descripción	As a developer, I want to use Adam optimizer on training, so I can check if it leads to better

	models
Criterios de aceptación	<ul style="list-style-type: none"> - Adam optimizer is used while training - Convergence is achieved faster - Lower loss values are achieved - Higher accuracy and F1 Score values are achieved
Resultado	Éxito
Etiquetas	poc, ml

Tabla Anexo B.33: Historia 33 – [Proof of Concept] Change training optimizer

[H-34]: [Proof of Concept] Test different batch sizes and epochs	
Descripción	As a developer, I want to test different batch sizes and epochs, to see if better models are achieved
Criterios de aceptación	<ul style="list-style-type: none"> - Batch size value is modified - Number of epochs is modified - Training is run with different values - Higher accuracy and F1 Score values are achieved
Resultado	Deshechado
Etiquetas	poc, ml

Tabla Anexo B.34: Historia 34 – [Proof of Concept] Test different batch sizes and epochs

SPRINT 7

[H-35]: Setup Flask environment	
Descripción	As a developer, I need to setup a Flask server, so I can use it to serve the machine learning model
Criterios de aceptación	<ul style="list-style-type: none"> - An app.py script is created - The script requires Flask's library - A “Hello World” GET function is defined
Etiquetas	ml, server

Tabla Anexo B.35: Historia 35 – Setup Flask environment

[H-36]: Create POST endpoint on Flask	
Descripción	As a user, I want to call the ML model so I can get the probability that an image is affected by retinopathy or not
Criterios de aceptación	<ul style="list-style-type: none"> - A new POST method is defined inside app.py - The method expects a JSON containing the image encoded - The method uses PyTorch's <i>eval</i> to get the probabilities - A JSON containing the probabilities is returned - Method returns HTTP 200 OK code if process goes OK - Method returns HTTP 500 Internal Server Error if other type of exception is thrown
Etiquetas	ml, server

Tabla Anexo B.36: Historia 36 – Create POST endpoint on Flask

[H-37]: Deploy Flask environment	
Descripción	As a developer, I need to deploy my Flask server on a cloud computer, so I can offer uninterrupted service
Criterios de aceptación	<ul style="list-style-type: none"> - A cloud machine is got from a hosting provider - The machine is identified by an URL, or a fixed IP, otherwise - A ping can be done from the dev machine to the cloud server- - Requests are restricted to the main REST server
Etiquetas	ml, server

Tabla Anexo B.37: Historia 37 – Deploy Flask environment

[H-38]: Connect analysis endpoint to Flask server	
Descripción	As a developer, I need to connect the main REST server to the Flask server, so real prediction results can be returned to the users
Criterios de aceptación	<ul style="list-style-type: none"> - A request is launched from the analysis endpoint on Express - Request contains an image encoded - Flask server receives the request - Flask sends a JSON-formed response - The response is received by the Express server

	- Result is persisted on database and forwarded to the client that asked for it
Etiquetas	ml, server

Tabla Anexo B.38: Historia 38 – Connect analysis endpoint to Flask server

Anexo C: Documento Básico de Especificaciones

Technical proposal

DED-AI

Division—V 1.0.0

03/10/2022

[Jose Luis Vidal de la Rosa](#)



Javier Gómez Martínez



Contents

Introducción	3
Objetivo	3
Antecedentes	4
Requerimientos	6
Tecnologías.....	7
¿Y ahora qué?.....	7



Introducción

Durante las últimas décadas se ha estado observando un aumento considerable en el número de afectados por la diabetes. La previsión para 2035 coincide en que la cifra de diabéticos se duplicará hasta alcanzar los 5,1 millones de afectados. Para los afectados, supone un cambio importante en su estilo de vida y un seguimiento constante. Por eso es importante llevar un control de la enfermedad y conocerse a uno mismo.

Debido a estos cambios y los cuidados que se requieren, es posible que aparezcan enfermedades asociadas a la diabetes, y que pueden afectar a órganos como el corazón, o los riñones, así como también pueden afectar a los pies, o causar enfermedades oculares como retinopatías.

Objetivo

El objetivo final de DED-AI es crear un ecosistema basado en un algoritmo de clasificación de imágenes, utilizando Deep Learning, que sea capaz de identificar a partir de una imagen capturada mediante un retinógrafo, si la imagen indicada presenta alguna retinopatía o no.

Además, dado que el entrenamiento se realizará utilizando imágenes captadas mediante un retinógrafo, se estudiará la viabilidad de introducir imágenes tomadas con dispositivos móviles como parte del entrenamiento de la red neuronal para la posterior predicción.

Se utilizará una aplicación móvil que será la encargada de realizar la captura de la imagen, enviarla a un servidor donde se ejecutará el algoritmo y retornará una respuesta indicando la clasificación de la imagen capturada.



Antecedentes

Hoy en día ser diabético implica llevar un control en el estilo de vida de los diagnosticados, especialmente en la alimentación, actividad física que realiza, pero sobre todo en el índice de glucosa en sangre, o lo que es lo mismo, nivel de azúcar en sangre. Un nivel aceptable de azúcar en sangre está dentro del rango 70-120, donde cualquier nivel por debajo de 70 se considera hipoglucemia y cualquier valor por encima de 120 se considera hiperglucemia. Ambos casos implican riesgos para la salud:

- Hipoglucemia: pérdida de conocimiento impidiendo la ingesta de azúcares de absorción rápida, lo que podría llevar a un coma diabético. El cerebro necesita azúcar para funcionar, de ahí que ante la falta de azúcar se produzca la pérdida de conocimiento.
- Hiperglucemia: Sin el correcto tratamiento insulínico, el azúcar en sangre se dispara dado que el cuerpo humano obtiene el azúcar de la grasa corporal. En este proceso, al igual que en otros que implique actividad física de alta intensidad, genera cetona en el cuerpo humano. Este compuesto orgánico es tóxico y aunque es producido por el cuerpo humano, en alta cantidad puede provocar cetoacidosis, que se corrige con insulina, pero al carecer de ella, implicaría el ingreso inmediato de los pacientes diabéticos en la UCI por complicaciones cardíacas o cerebrales.

En la actualidad, la forma más común de controlar los niveles de azúcar en sangre es a través de glucómetros, donde los usuarios, a través de una gota de sangre obtenida de los capilares de los dedos, indicará al usuario el nivel de sangre que tenga en ese momento. Aunque es útil y extremadamente fiable, este sistema es caro, ya que se realiza a través de tiras reactivas de un solo uso. Los usuarios que utilizan este sistema realizan al menos 6 mediciones diarias, una antes de cada comida principal y otra, 2 horas después de haber comenzado cada comida principal. Además, siempre que tenga síntomas, deberá realizarse la prueba por sí mismo, y en caso de no poder, otra persona deberá hacérsela.

Hasta hora, este era el método tradicional, sin embargo, varios laboratorios han diseñado otros mecanismos igual de fiables y que aportan más información, permitiendo a los usuarios mayor control y libertad en su vida cotidiana. Se trata de sensores que se colocan en la parte trasera del brazo, donde un filamento se introduce en el líquido intersticial para calcular el nivel de azúcar en sangre en tiempo real. Para más información acerca de estos sensores: <https://www.freestylelibre.es/libre/>

El azúcar en sangre varía en función de la alimentación y de la actividad física. Así, la ingesta de hidratos de carbono proporciona azúcar en sangre, en la que se pueden distinguir tres tipos:

- De absorción lenta, como por ejemplo cereales, harinas, pastas, legumbres, arroces, etc.
- De absorción rápida: azúcar refinado, zumos, miel, etc.
- De absorción semi rápida: frutas.

Por lo tanto, un diabético deberá aprender a contar las raciones de hidratos de carbono que ingiere a lo largo del día para mantener un índice glucémico estable, y por otro lado aprender a calcular el número de unidades de insulina que necesitará para metabolizar esa ingesta y continuar manteniendo el azúcar en sangre de forma homogénea para así evitar llegar a hipoglucemias o hiperglucemias.

En diabetes de tipo 1, o mellitus, donde el diabético es insulinodependiente, deberá aplicarse a lo largo del día un número determinado de unidades de insulina. En concreto, se podrá aplicar las siguientes:

- Insulina lenta o basal, tiene una duración de aproximadamente 24 horas en el cuerpo humano, que se usa básicamente para controlar el azúcar en sangre durante la noche en períodos de sueño, mientras se está en ayunas y entre comidas. Normalmente se aplica un número determinado de unidades una vez al día.
- Insulina rápida, que comienza a hacer efecto a los 5-15 minutos de su aplicación y es capaz de corregir el pico máximo en el rango de 1 a 2 horas desde su aplicación. Se utiliza para corregir el azúcar en sangre que se produce en las digestiones de las comidas principales del día (desayuno, comida y cena) y cuando haya picos de azúcar en sangre descontrolados. También se podría aplicar a cualquier otra comida como la merienda o media mañana si la cantidad de



hidratos a ingerir es elevada, aunque no debería ser así. Esta insulina hace el trabajo que debería hacer el páncreas, de liberar insulina cuando hay demasiado nivel de azúcar en sangre. El usuario inyecta la insulina, por tanto, al menos 3 veces al día siempre justo en el instante antes de comer, evitando que pasen más de 15 minutos desde la aplicación hasta que comienza a ingerir alimentos.

- Insulina de acción intermedia: empieza a hacer efecto entre 1 a 2 horas después de su aplicación, y el pico se obtiene a partir de la cuarta a sexta hora tras su aplicación. Se utiliza de igual forma que la insulina lenta pero se necesitarían más aplicaciones al cabo del día.

Para más información: <https://dtc.ucsf.edu/es/tipos-de-diabetes/diabetes-tipo-2/tratamiento-de-la-diabetes-tipo-2/medicamentos-y-terapias-2/prescripcion-de-insulina-para-diabetes-tipo-2/tipos-de-insulina/>

A lo largo del día, cada persona puede ofrecer resistencia a la insulina, eso quiere decir que no siempre se necesita aplicar el mismo número de unidades para una misma ingesta de hidratos de carbono. Por ejemplo, si en la comida se han ingerido 4 raciones de hidratos de carbono y se han aplicado 5 unidades de insulina que han permitido mantener el azúcar estable desde que se empezó a comer hasta que se empezó a cenar, sin hipoglucemias ni hiperglucemias, no quiere decir que para una cena de también 4 raciones de hidratos de carbono se vayan a necesitar también 5 unidades de insulina, sino que puede ser que se necesiten más o menos, en función de la resistencia que se tenga a esa hora. Cada persona tiene momento en el día donde puede ofrecer más resistencia a la insulina y no tiene por qué ser igual entre personas, aunque sí que suele ser constante en la misma persona.

Además, hay que tener en cuenta lo que se consideran fases de luna de miel, en la que el páncreas de una persona diabética es capaz de seguir generando insulina, lo que hace que se produzcan hipoglucemias a consecuencia de la inyección de unidades de insulina artificial.



Requerimientos

Hoy en día en Castilla y León el protocolo para determinar si una persona diabética presenta alguna retinopatía o no consiste en realizar una prueba de fondo de ojo donde se toma una captura a través de un retinógrafo. Estos aparatos normalmente se encuentran en centros de salud y hospitales, aunque no todos disponen de ellos. Una vez se ha realizado la captura de la imagen, un oftalmólogo evaluará la imagen y determinará si existe o no retinopatía. En el mejor de los casos, el propio centro de salud u hospital contará con personal cualificado para la evaluación, aunque la gran mayoría de veces se recurre a un instituto especializado situado en Valladolid. Allí llegará la imagen, y nuevamente el oftalmólogo determinará la existencia de retinopatías.

Este proceso, como se deduce, puede realizarse en apenas 10 minutos, contando con que el centro de salud disponga de todo lo necesario, o puede dilatarse más en el tiempo en el caso en que no disponga ni de retinógrafo, ni de especialista capaz de evaluar la imagen obtenida.

Por ello, en colaboración con el Hospital de León y el Centro de Salud Río Órbigo, y partiendo de un conjunto de imágenes donde se ha determinado que existe retinopatía como otras tantas donde no, se pretende elaborar una red neuronal que identifique de forma automática cuándo una imagen presenta retinopatía y cuándo no.

Por lo tanto, una vez tengamos clasificadas nuestras imágenes, se procederá a la implementación del algoritmo de clasificación. Para ello, tomaremos el 80% de las imágenes para entrenar la red neuronal y el 20% restante se emplearán para validar el algoritmo, comprobar la tasa de error y readjustar la configuración de la arquitectura en caso de que sea necesario.

Una vez completado el algoritmo de clasificación, o bien en paralelo, se realizará un servidor web, con un API REST donde llegue una imagen en la petición, se procese en el algoritmo, y retorne como respuesta el resultado de la clasificación.

Para ello, el servidor expondrá al menos un endpoint, que, a través de una operación POST donde se incluya la imagen, invoque al algoritmo de predicción, espere el resultado, y lo devuelva al cliente que hizo la llamada indicando el resultado. En caso de ser necesarios más datos en el cuerpo de la llamada al endpoint, se enviará en formato JSON. La respuesta de la llamada será a su vez un objeto JSON, bien estructurado, junto con el código de respuesta HTTP oportuno en función de la ejecución (200 OK cuando haya respuesta por parte del algoritmo, 500 si algo ha fallado, 401 en caso de implementar seguridad y el cliente no esté autorizado, etc.).

Existirá un segundo endpoint en el API que permita indicar si la predicción sobre una imagen capturada fue certera o no. Para ello, necesitaremos no sólo almacenar la imagen en disco en el servidor en el primer endpoint, sino que deberemos tener una referencia a la misma mediante un ID único.

Además, se propone realizar una aplicación móvil que sea capaz de capturar una imagen además de ser cliente del servidor donde se ejecuta el algoritmo de clasificación. La versión mínima de la aplicación móvil puede implementar el siguiente flujo:

- Una vez el usuario selecciona la aplicación, abrirá la cámara directamente, donde habrá un botón para realizar la captura. La alternativa en caso de que exista demasiada complejidad es tener una vista principal, con un botón que, o bien abra la cámara desde la propia aplicación, o bien pueda abrir la aplicación Cámara existente en el dispositivo para poder realizar la captura.
- Cuando se haya realizado la captura, se mostrará al usuario junto con un botón que permita subir la imagen al servidor para su procesamiento.
- Mientras el servidor está procesando la imagen, la aplicación móvil mostrará una escena de espera al usuario, indicando que la imagen se está procesando.
- En caso de error tras el procesamiento, se mostrará una nueva escena indicando el motivo del error.
- Si el procesamiento ha sido correcto, se mostrará en una escena el resultado al usuario. En esta escena, se mostrarán dos botones al usuario para que indique si el algoritmo ha sido certero o no. Esta información se enviará de vuelta al servidor para clasificar la imagen de acuerdo al resultado.



Desde el punto de vista del servidor, durante el procesamiento de la imagen, ésta se almacenará en disco, se generará un ID único para dicha imagen y se persistirá de alguna manera para crear una relación. La imagen se guardará temporalmente en un directorio hasta completar todo el flujo.

Una vez que la predicción ha terminado y se ha enviado el resultado al cliente (en este caso la aplicación móvil), cuando el usuario seleccione en la aplicación si la predicción ha sido certera o no y se envíe de nuevo al servidor, éste moverá la imagen a la carpeta con las imágenes clasificadas como retinopatía en caso de que la predicción haya sido satisfactoria, y la moverá a una carpeta con las imágenes donde no se presentan retinopatías en caso contrario.

Las nuevas imágenes añadidas al dataset servirán para poder generar un nuevo modelo de predicción en caso necesario.

Tecnologías

Las tecnologías propuestas son:

- **Servidor:** deberá ser un servidor REST y se propone Golang como lenguaje de implementación. En su defecto, se propone NodeJS con ExpressJS, basado en JavaScript y utilizando TypeScript como lenguaje de desarrollo. Se propone Visual Studio Code para el desarrollo. Habrá un directorio específico en la raíz del repositorio para el desarrollo de la parte del Backend. En caso de optar por otras tecnologías, realizar un análisis con Pros & Cons, de cara a explicar la decisión en la memoria del proyecto
- **Aplicación Móvil:** Se propone Android Native con Kotlin como primera versión. Si se quiere expandir a más dispositivos, se propone utilizar React Native. Se necesitará en ambos casos Android Studio, aunque para desarrollar con React Native se puede utilizar Visual Studio Code. Si se opta por cualquier otro framework como Ionic, Flutter, NativeScript, etc., se realizará un análisis con Pros & Cons que se incluirá en la memoria del proyecto.
- **Red Neuronal:** En el caso de la red neuronal, se propone Pytorch sobre Python. La alternativa es utilizar TensorFlow o SciKit, por lo que en caso de optar por una de las alternativas se realizará un análisis con Pros & Cons para añadirlo a la memoria del proyecto.

¿Y ahora qué?

Los requerimientos podrán ir cambiando a lo largo del proyecto para incluir o excluir funcionalidad. Esta lista es una explicación a alto nivel de lo que debería incluir la aplicación, y no establece ningún orden, sino que será determinado por cada desarrollador cómo deberá gestionar los requerimientos. A partir de los requerimientos, se deberá poder empezar a crear issues en GitLab para que formen parte de nuestro backlog y tener identificado gran parte del trabajo del proyecto.

Antes de crear la lista de issues se recomienda analizar la tecnología a utilizar si no se conoce y a entender bien qué es lo que se pide. Por supuesto se pueden hacer propuestas, modificaciones y mejoras a los requerimientos dados para que os sintáis cómodos desarrollando. También es aconsejable analizar y entender la arquitectura del proyecto, realizando un diagrama a alto nivel donde aparezcan todos los componentes y la forma en que interactúan unos con otros.

Para el análisis de los requerimientos os recomiendo papel y lápiz antes de empezar a crear la lista de issues, ya que de ahí podréis sacar también cualquier otro tipo de diagrama que os sirva para llevar a cabo vuestra implementación. Este paso os lo aconsejo porque os ayudará a tener una visión muy amplia de los requisitos y a aportar el máximo de información posible a vuestros issues para que después os sea fácil implementarlo y completar la memoria del proyecto. Cuando penséis en la lista de issues a la hora de realizar el análisis, pensad en que sean lo suficientemente pequeños como para que podáis implementarlo en máximo 8 horas, que sería una jornada completa de trabajo. Esto se irá perfeccionando a lo largo de las iteraciones, pero cuanto más acertéis, más fácil será el curso de la iteración. Por ejemplo, sería conveniente que mostréis en un diagrama de flujo desde que el usuario realiza una acción en la aplicación móvil, indicar por qué otros componentes pasa la llamada hasta el retorno de la misma.

Si a la hora de desarrollar (en cuanto a implementación del código se refiere) un issue os lleva más tiempo de la cuenta, se hace pesado o no tiene sentido, pensad en dividirlo en uno o más issues para



que sean abarcables dentro de las iteraciones. Vosotros gestionaréis el backlog, por lo que podréis continuar creando issues como consideréis para llevar vuestro registro, al igual que eliminarlos o modificarlos siempre que lo necesitéis. En la revisión de la iteración que acaba y la planificación del siguiente, se comentarán estos cambios en los issues. En cada issue implementad única y exclusivamente lo que ponga en el issue dentro del acceptance criteria, cualquier otra cosa como por ejemplo una mejora, refactor del código, corrección de un issue que hayáis detectado, cread un nuevo issue para tenerlo en el backlog, pero no aprovechéis un issue para hacer cosas de otro. Si hay dependencia entre issues, entonces probad a resolverla antes reanalizando los issues que tenéis, puede ser que no quede más remedio que exista la dependencia, en ese caso, aseguraros de ir desarrollando issue por issue.

Si tenéis que hacer refactor del código, es decir, modificar la estructura de una clase, dividir el código en otras clases o métodos, un consejo: primero refactorizar el código existente y comprobar que todo sigue funcionando, y luego añadir la nueva funcionalidad, nunca mezcléis funcionalidad nueva mientras realizáis refactor.

Documentad en texto todo lo que podáis, tanto en comentarios en el código, como en el Readme del proyecto, en los detalles del issue o incluso la wiki que hay dentro de GitLab. Estos comentarios os servirán después para la memoria del proyecto.

El issue se creará con la siguiente información básica:

- Título: brevísima descripción de lo que vamos a implementar. Por ejemplo, en el caso del API REST: Crear el endpoint /path/whatever que devuelva siempre un valor estático en el cuerpo y el estado 200 OK.
- Descripción: se hará siguiendo esta plantilla,

As a <user, develop, other people>

I want to <brief explanation about what the person specified in the "As a" needs>

So that <why this will be useful for that person>

Casi todos los issues deberán ser As a user, ya que es a quien le vamos a entregar valor. En el caso del endpoint mencionado anteriormente:

As a user,

I want to POST an image to the server

So that I can know if exists retinopathy or not

Además, incluiremos en la descripción una sección que será el Acceptance Criteria para ese issue, que será el mínimo que se deba cumplir para que el issue se considere como realizado.

Como ejemplo del acceptance criteria:

Acceptance Criteria:

- o The endpoint will be /path/whatever.
- o The endpoint will return a JSON Object with the result of the prediction saying true or false.
- o The endpoint will return a 200 OK status code.
- o The endpoint will return a 500 Internal Server Error when an exception is thrown during the prediction
- o No header will be needed.
- Label: indicaremos si se trata desarrollo para el servidor (server), para la app móvil (mobile), para el algoritmo (ml machine learning), si se trata de un issue de investigación (con el término spike), si se trata de una prueba de concepto para ver si existe viabilidad de implementación



(con el término poc), si se trata de un issue para realizar tests de cualquier tipo (unit tests, integration tests, e2e tests) y podremos poner tantas etiquetas como necesitemos para explicar brevemente qué vamos a hacer.

Se puede crear una estrategia de milestones, donde cada uno de ellos será un MVP (minimum viable product). Bastará con indicar, de todas las funcionalidades detectadas en los requerimientos, cuántas contendrá cada milestone. Para ello, podemos seguir estrategias existentes como crear una versión Alpha, destinada a uso interno con parte de la funcionalidad pero que no es estable. Una versión Beta que sigue sin ser estable al 100% pero que ofrecerá una versión muy similar a la final para que pueda ser probada por equipos independientes o usuarios seleccionados. Una release candidate, que es la versión previa a la final, y que ya debería ser estable al 100%, y en caso de que no lo sea, se irá creando una nueva cada vez que se hagan correcciones al código. Estas correcciones deben ser mínimas. Y por último, una versión final, que será la que se lance al resto de usuarios finales.

Para llevar a cabo esta estrategia, necesitaremos tener claro cuál es el end 2 end que queremos implementar en cada milestone. Por ejemplo, en la versión Alpha 0.1 podemos implementar el algoritmo de predicción con todo lo que ello implica:

- Preparación del data set
- Implementación del algoritmo
- Entrenamiento de la red neuronal
- Validación de la red neuronal

Para la versión Alpha 0.2, además de tener funcional el algoritmo de predicción, crearemos el servidor con el REST API que sea capaz de invocar al algoritmo y obtener un resultado. Ello implica:

- Implementar el servidor para que corra en un puerto determinado
- Implementar el API REST con el endpoint inicial que necesitemos (definir el endpoint, los códigos HTTP de error posibles, los mensajes de respuesta, su estructura JSON, etc).
- Implementar la llamada desde el servidor al algoritmo y esperar a la respuesta.

Para la versión Alpha 0.3 se podría implementar la aplicación móvil con todo el UI necesario. Y así sucesivamente.

Desde un punto de vista de empresa o startup, el primer milestone debería incluir un end2end básico que involucre a todos los componentes necesarios para que un usuario pueda realizar una prueba de concepto. Esto implicaría que el Alpha 0.1 incluyese parte de implementación de los tres grandes componentes, el algoritmo, el servidor y la app móvil. Que sea un milestone y un end2end no significa que la información sea real, es decir, que el algoritmo por ejemplo no utilice el data set todavía y que para el end2end inicial retorne si/no aleatoriamente, que el servidor llame a ese algoritmo básico y que cuando haya una petición, que puede ser un POST al que no se envía nada temporalmente, reciba la respuesta, y una aplicación móvil con un botón que cuando se pulse se comunique con el servidor y reciba si/no y lo muestre en pantalla. Después de este milestone básico, se iría incrementando el valor, sin romper el end 2 end básico en cada nueva versión.



Anexo D: Resultados Completos del Cuestionario de Evaluación

A continuación se desglosan los resultados por preguntas del cuestionario llevado a cabo para evaluar la usabilidad de la aplicación, cuya ficha técnica es descrita en el capítulo 4.1.3.1. Cada pregunta es mostrada en un gráfico que agrupa las respuestas idénticas.

¿Cuál es tu edad?

11 respuestas

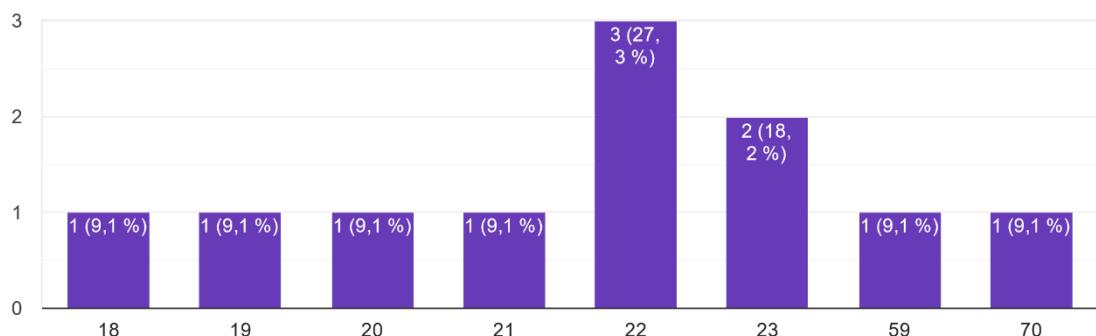


Figura Anexo D.1 – Respuestas a la pregunta 1

¿Y tu sexo?

11 respuestas

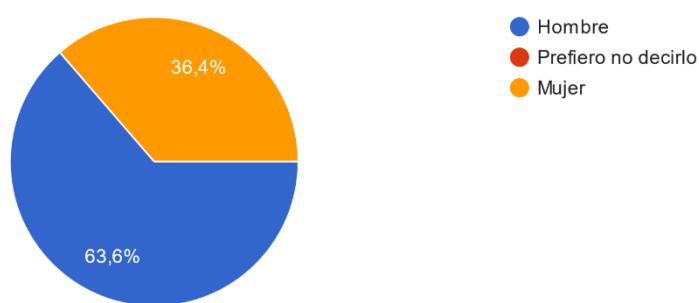


Figura Anexo D.2 – Respuestas a la pregunta 2

¿Tienes alguna discapacidad? (Especialmente visual)
11 respuestas

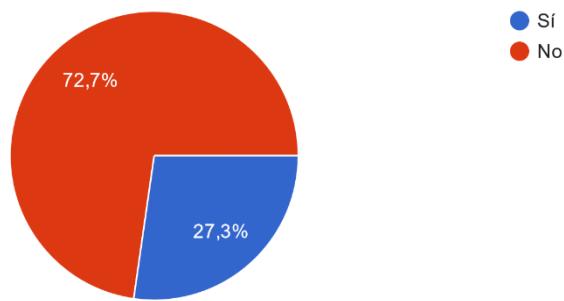


Figura Anexo D.3 – Respuestas a la pregunta 3

En caso afirmativo, indica cual

Vista de cerca y lejos

Miopia

Miopia

Figura Anexo D.4 – Respuestas a la pregunta 4

¿Cómo de fácil es usar esta app por primera vez?
11 respuestas

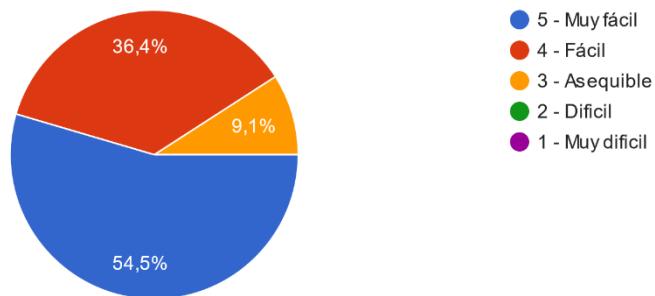


Figura Anexo D.5 – Respuestas a la pregunta 5

¿Te parece que la app es rápida?

11 respuestas

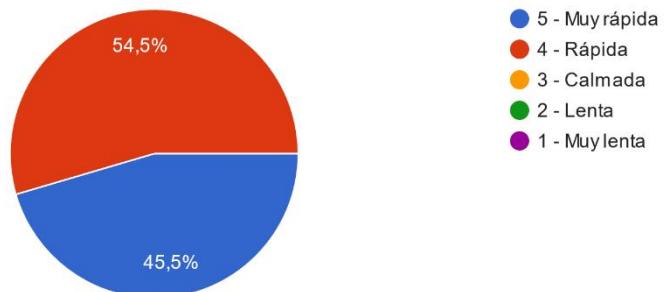


Figura Anexo D.6 – Respuestas a la pregunta 6

¿Te parece que los textos son fáciles de leer?

11 respuestas

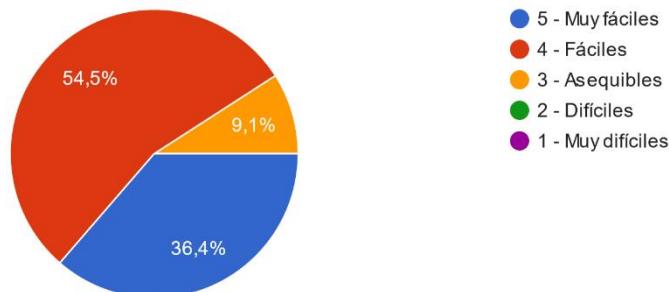


Figura Anexo D.7 – Respuestas a la pregunta 7

¿Te parecen que los colores empleados son los adecuados?

11 respuestas

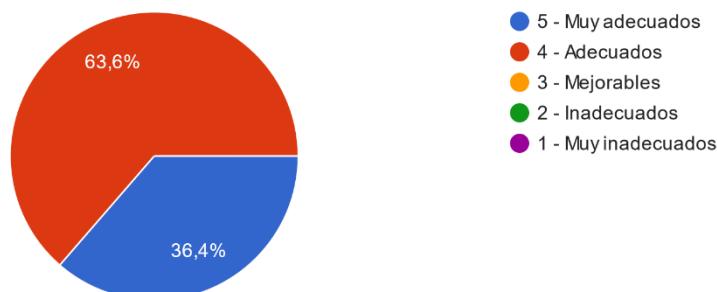


Figura Anexo D.8 – Respuestas a la pregunta 8

¿Cómo de sencillo te parece iniciar sesión?

11 respuestas

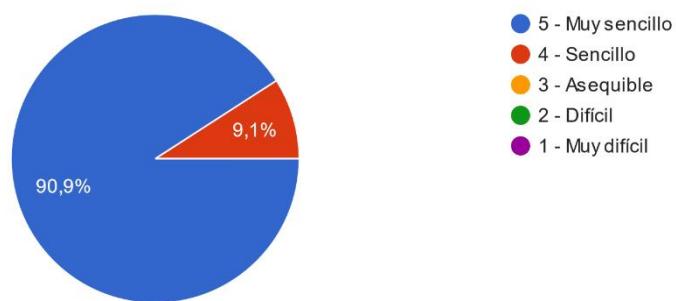


Figura Anexo D.9 – Respuestas a la pregunta 9

¿Cómo de sencillo te parece seleccionar y confirmar una imagen?

11 respuestas

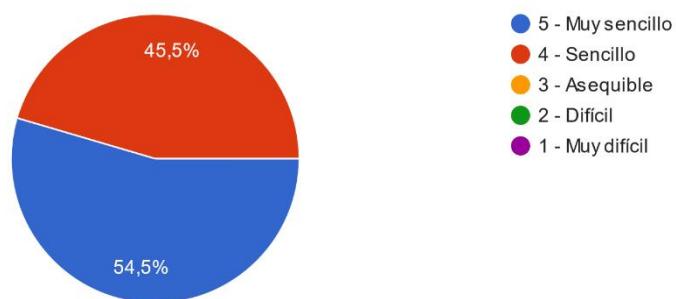


Figura Anexo D.10 – Respuestas a la pregunta 10

¿Crees que los resultados del análisis se muestran de forma sencilla?

11 respuestas

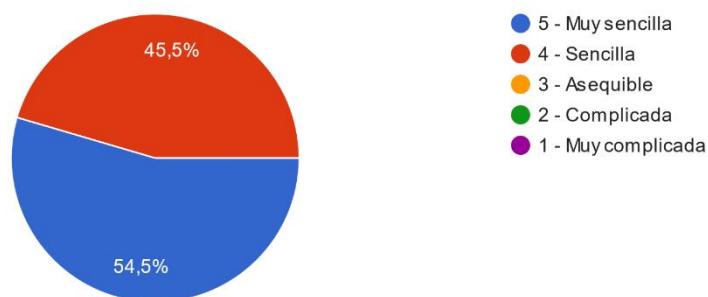


Figura Anexo D.11 – Respuestas a la pregunta 11

¿Cómo de sencillo te parece dar feedback?

11 respuestas

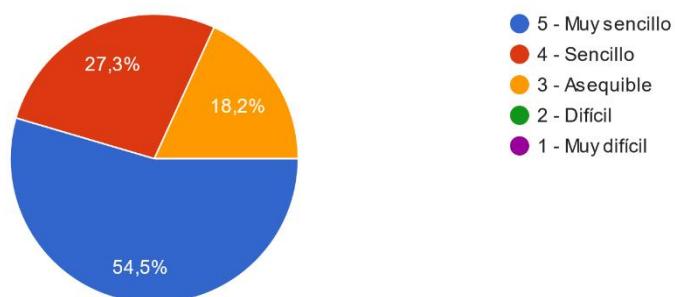


Figura Anexo D.12 – Respuestas a la pregunta 12

¿Has tenido alguna dificultad de manejo? (En caso afirmativo, indica cuál)

11 respuestas

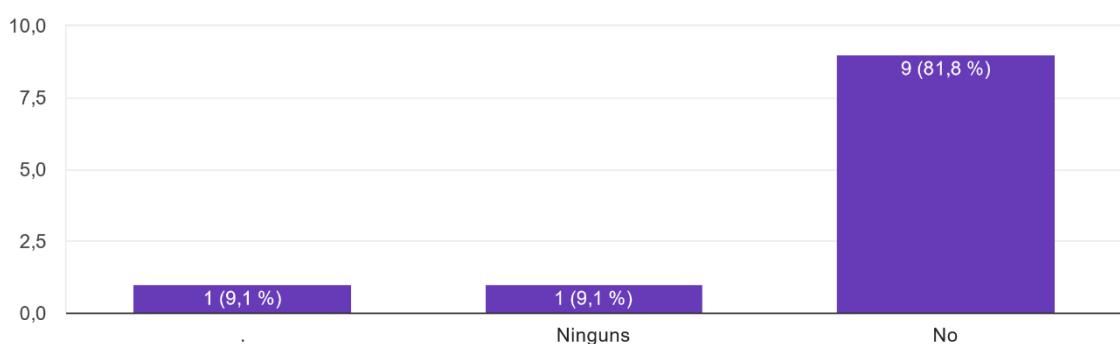


Figura Anexo D.13 – Respuestas a la pregunta 13

¿Qué puntos fuertes ves en la aplicación?

Eficacia a la hora del diagnóstico
Es muy sencilla
Tiene iconos muy grandes e intuitivos que pueden ayudar a cualquier persona que tengo problemas de visión o que le cueste identificarlos.
Creo que la rapidez es muy importante y el acortar caminos. Los profesionales pueden evitar pruebas que alarguen procesos.
La facilidad para todos los públicos, es decir, app intuitiva
muy útil y fácil de usar para cualquier tipo de persona tanto con conocimientos de informática como no
Seleccionar una imagen
La rapidez de respuesta
Simplez
La sencillez

Figura Anexo D.14 – Respuestas a la pregunta 14

¿Qué aspectos mejorarías?

Ninguno
Un formulario de registro
Tal vez cambiaría los tonos morados por otros en algunas situaciones, por lo demás parece perfecta.
Por decir algo, algún diseño para que llamase mas la atención
alguna indicación más para que no haya ninguna dificultad
Tal vez que el feedback fuese un poco más detallado si el usuario lo quisiera, como ofrecer un porcentaje de que sea afirmativo o negativo o tal vez que permita localizar un poco la zona donde la retina tiene el problema
Recordar el usuario al introducirlo una vez
Nada
Se podría añadir una guía de cómo reconocer si podrías tener algún problema visual

Figura Anexo D.15 – Respuestas a la pregunta 15