Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN ENXEÑARÍA DO SOFTWARE
MENCIÓN EN COMPUTACIÓN

Dx-OPHTHA: Plataforma web para la gestión de pacientes y diagnóstico inteligente en imágenes oftalmológicas

Estudante: Iván Barrientos Lema
Dirección: José Joaquim de Moura Ramos
Jorge Novo Buján
Marcos Ortega Hortas

A Coruña, setembro de 2021.

A las personas que han hecho posible que llegase aquí.

Agradecimientos

Gracias a Joaquim, Jorge y Marcos por ser los directores de mi Trabajo de Fin de Grado y por darme la oportunidad de participar en este proyecto. Agradecer especialmente a Joaquim por guiarme y ayudarme con el trabajo del día a día. También dar las gracias a todos los compañeros que se han cruzado conmigo en la carrera y a mi familia, por apoyarme y hacer más llevadero el camino hasta aquí.

Resumen

En el ámbito oftalmológico, para llevar a cabo diferentes procesos de diagnóstico clínico, es habitual disponer de diferentes modalidades de imagen de fondo de ojo. Actualmente, tanto la tomografía de coherencia óptica (OCT) como la angiografía por tomografía de coherencia óptica (OCT-A) han logrado grandes avances en el estudio del polo posterior del ojo (mácula, retina y vítreo), y son imprescindibles para el diagnóstico y seguimiento del tratamiento de diferentes patologías oculares y sistémicas. Por otro lado, el desarrollo de sistemas de apoyo al diagnóstico es un campo emergente, en el que los avances clínicos combinados con los tecnológicos están permitiendo a los especialistas clínicos diagnosticar con mayor precisión diversas patologías, lo que se traduce en un tratamiento más adecuado y, en consecuencia, en una mejora de la calidad de vida de los pacientes.

Con esto en mente nace Dx-OPHTHA, un sistema de gestión clínica que integra diferentes módulos inteligentes que explotan técnicas de *Deep Learning* y que permiten apoyar a los especialistas clínicos en la tarea de diagnóstico y control de diferentes patologías a partir de distintos tipos de imágenes oftalmológicas. Por un lado, Dx-OPHTHA cuenta con 3 módulos inteligentes integrados para el análisis automático de imágenes oftalmológicas: (I) Módulo de análisis según el tipo de dispositivo de captura de imagen, (II) Módulo de análisis patológico en imágenes OCT y (III) Módulo de análisis patológico en imágenes OCT-A. Por otro lado, Dx-OPHTHA también dispone de diferentes módulos de gestión clínica que completan su funcionalidad. En concreto, consta de un gestor de pacientes, informes e imágenes, lo que lo convierte en una potente herramienta capaz de facilitar el trabajo rutinario de los oftalmólogos. Dx-OPHTHA también ofrece funcionalidades adicionales como la visualización intuitiva de los resultados, la exportación de informes a formato PDF o el envío automático de los mismos por correo electrónico. Dx-OPHTHA es una plataforma web totalmente escalable, que permite añadir fácilmente nuevos módulos inteligentes para el diagnóstico de nuevas patologías u otros tipos de imágenes médicas.

Abstract

In the field of ophthalmology, different fundus imaging modalities are commonly used to carry out different clinical diagnostic procedures. Currently, both optical coherence tomography (OCT) and optical coherence tomography angiography (OCT-A) have made great advances in the study of the posterior pole of the eye (macula, retina and vitreous), and are essential for the diagnosis and monitoring of the treatment of different ocular and systemic

pathologies. On the other hand, the development of diagnostic support systems is an emerging field, in which clinical advances combined with technological advances are allowing clinical specialists to diagnose various pathologies with greater precision, which translates into more appropriate treatment and, consequently, an improvement in the quality of life of patients.

With this in mind, Dx-OPHTHA was born, a clinical management system that integrates different intelligent modules that exploit Deep Learning techniques to support clinical specialists in the task of diagnosis and control of different pathologies based on different types of ophthalmological images. On the one hand, Dx-OPHTHA has 3 integrated intelligent modules for the automatic analysis of ophthalmological images: (I) Analysis module according to the type of image capture device, (II) Pathology analysis module on OCT images and (III) Pathology analysis module on OCT-A images. On the other hand, Dx-OPHTHA also has different clinical management modules that complete its functionality. Specifically, it consists of a patient, report and image manager, which makes it a powerful tool capable of facilitating the routine work of ophthalmologists. Dx-OPHTHA also offers additional functionalities such as the intuitive visualisation of results, the export of reports to PDF format or the automatic sending of reports by e-mail. Dx-OPHTHA is a fully scalable web platform, which allows to easily add new intelligent modules for the diagnosis of new pathologies or other types of medical images.

Palabras clave:

- Aprendizaje profundo
- Imagen oftalmológica
- Redes de neuronas convolucionales
- SCRUM
- Java
- Python
- JavaScript

Keywords:

- Deep learning
- Ophthalmological imaging
- Convolutional Neural Networks
- SCRUM
- Java
- Python
- JavaScript

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo se introducirá el contexto del proyecto así como las motivaciones y los objetivos del mismo. A continuación, se expondrá la forma de este documento con el fin de estructurar los apartados que lo conforman.

1.1 Descripción

La Visión Artificial es la disciplina encargada de procesar y analizar imágenes con el fin de obtener información útil. Esta disciplina abarca muchos campos ya que tiene una gran utilidad en el mundo actual. Gracias a esto, el reconocimiento de objetos, la detección de patrones o la generación de imágenes son posibles y la integración en aplicaciones permiten generar productos de alta utilidad [2]. En concreto, en el ámbito de la medicina, existen muchas aplicaciones en las que se utiliza la Visión Artificial. Normalmente, esta práctica está asociada al desarrollo de sistemas de apoyo al diagnóstico de diferentes patologías sobre imágenes médicas, como pueden ser tomografías, radiografías, fluoroscopías o resonancias magnéticas, entre otras.

En el campo de la oftalmología existen diferentes procedimientos de diagnóstico clínico, en los que se suele disponer de diferentes métodos de captura de imagen del fondo de ojo. Este tipo de imágenes pueden ser de dos tipos: invasivas, que implican una agresión a los órganos visuales (como la aplicación de contraste intravenoso para teñir y visualizar los tejidos oculares); no invasivas, que no implican ningún tipo de agresión y suelen ser más fáciles de obtener y muy cómodas para el paciente. Atendiendo a esto, en los últimos años, el uso de este último grupo de imágenes oftalmológicas para el diagnóstico se ha incrementado significativamente con el fin de identificar diferentes patologías sistémicas y/o oculares.

Dentro de este grupo de imágenes se encuentran 2 técnicas de captura de imágenes oftalmológicas que se utilizarán en este proyecto: la Tomografía de Coherencia Óptica (OCT) y la Angiografía por Tomografía de Coherencia Óptica (OCT-A). Por un lado, OCT es una técnica

de imagen tomográfica no invasiva que permite estudiar las principales estructuras internas de un tejido, en este caso, de la retina humana. Por otro lado, OCT-A es un método de imagen capaz de mostrar la circulación retiniana y coroidal en directo sin el uso de ningún material de contraste, convirtiendo así esta técnica en no invasiva.

Actualmente, estas técnicas de imagen son las más relevantes dentro de este campo y son además esenciales para el diagnóstico y el seguimiento de pacientes en las consultas oftalmológicas.

1.2 Motivación

Por un lado, el proceso de análisis y diagnóstico basado en estas imágenes es un procedimiento que normalmente realiza el experto clínico mediante la inspección visual de las imágenes, lo que hace que este proceso sea muy laborioso y además dificulta la comparación de los resultados entre diferentes estudios debido a la falta de repetibilidad entre expertos. Por otro lado, los algoritmos de aprendizaje máquina basados en técnicas de *Deep Learning* han demostrado un alto rendimiento en problemas similares, lo que los convierte en los más idóneos para resolver este tipo de tareas. Por ello, este proyecto explota todo el potencial de estos modelos para la extracción automática de características de la imagen, permitiendo desarrollar un sistema inteligente capaz de clasificar de forma totalmente automática las imágenes oftalmológicas (OCT/OCT-A) en función de la presencia o ausencia de patologías. Destacar el uso de este tipo de imágenes oftalmológicas debido a que son las más vanguardistas y las que tienen mayor potencial futuro en diagnóstico en esta especialidad. Además, los sistemas de apoyo al diagnóstico son un campo en rápida evolución, en el que los avances clínicos combinados con los tecnológicos están permitiendo a los especialistas realizar de manera más precisa el diagnóstico de diferentes patologías, lo que permite un tratamiento más adecuado y, en consecuencia, una mejora de la calidad de vida de los pacientes.

1.3 Objetivos

Con el fin de facilitar las tareas de diagnóstico clínico, se desarrollarán varios métodos automatizados para detectar la presencia de neovascularización coroidal (CNV), de edema macular diabético (DME) y de la presencia de drusas (DRUSEN) a partir de imágenes OCT. Además, se desarrollará un método automático para detectar la presencia de oclusión venosa de la retina (RVO), en este caso, mediante el análisis de imagen OCT-A. Decir que para la elección de estas afecciones se ha tenido en cuenta que son las más frecuentes en los pacientes que acceden a los servicios clínicos. Como el tipo de imágenes puede variar según el tipo de patologías a diagnosticar, se creará otro modelo capaz de diferenciar entre distintos tipos

de imágenes oftalmológicas. En este caso, entre OCT y OCT-A. Todos los métodos computacionales que se desarrollarán en este trabajo emplearán técnicas de *Deep Learning*. Una vez desarrollados estos modelos inteligentes, se integrarán en una plataforma capaz de gestionar la información del paciente, facilitando el diagnóstico y control de diferentes patologías oculares. Para ello, se diseñará una aplicación web con el fin de expandirse con nuevos modelos inteligentes, adaptándose así fácilmente a nuevas patologías descubiertas o a otros tipos de imágenes oftalmológicas.

Así, nuestra propuesta ofrecerá (entre otras) 4 ventajas principales derivadas de la automatización de estos procesos:

1. Reducción drástica de los tiempos de diagnóstico;
2. Aumento de la eficiencia/reducción de los costes directos e indirectos;
3. Repetibilidad/ausencia de subjetividad en los procesos de diagnóstico, haciendo que las evaluaciones sean consistentes y objetivas a lo largo del tiempo;
4. Plataforma de apoyo al diagnóstico pensada para la transferencia de estos resultados al contexto sanitario y su uso por parte de los especialistas.

1.4 Estructura de la memoria

- **Introducción**. Se describe la intención del proyecto, las motivaciones, los objetivos y la estructura del documento.
- **Contextualización**. Se explica el contexto clínico del proyecto así como la base tecnológica utilizada en el mismo.
- **Materiales**. Se detalla el conjunto de *datasets* empleados en el proyecto.
- **Tecnologías**. Se detallan las tecnologías empleadas en el proyecto.
- **Proceso de ingeniería**. Se presenta la metodología de desarrollo *software* empleada, así como la gestión del proyecto, recursos y costes asociados al proyecto.
- **Metodologías computacionales para el screening patológico**. Se presentan las metodologías computacionales empleadas para la producción de los modelos inteligentes, así como la arquitectura de red utilizada.
- **Resultados y análisis**. Se muestran y analizan los resultados obtenidos por las metodologías computacionales.

- **Desarrollo software**. Se detallan el análisis, el diseño, la implementación y las pruebas del sistema desarrollado en el proyecto.
- **Conclusiones**. Se presenta las principales conclusiones del proyecto.
- **Estado final**. Se explica el estado en el que queda el proyecto.
- **Trabajo futuro**. Se enumeran las posibles mejoras que tendrían cabida en el estado en el que queda el proyecto.
- **Apéndices**. Está compuesta por un glosario de términos y otro de acrónimos. También incluye otros contenidos complementarios relevantes.
  - **A. Casos de uso.** Se describen los diferentes casos de uso de todas las funciones de la aplicación web.
  - **B. Interfaz.** Se muestran diferentes imágenes de los aspectos más importantes de la interfaz de la aplicación web.
- **Bibliografía**. Se recoge toda la documentación empleada para el desarrollo del proyecto.

Capítulo 2

Contextualización

2.1 El ojo humano

2.1.1 Anatomía y fisiología

El ojo o globo ocular es una estructura esférica de aproximadamente 2.5 cm de diámetro con una protuberancia en la parte delantera. Las paredes del globo ocular están formadas por tres capas: externa, media e interna.

- La capa externa es la esclerótica. Esta tiene una composición dura con los laterales y el lado posterior que son de color blanco y opaco, mientras que la parte delantera recibe el nombre de córnea y es, por contrapartida, transparente.
- La capa media, conocida como úvea, es de color oscuro y tiene como función la de nutrir a la retina. En esta capa se pueden distinguir tres componentes principales: el iris, el cristalino y la coroides.
  - El iris está situado entre el cristalino y la córnea. Esta parte está caracterizada por poseer una serie de pigmentos que le dan color. Los pigmentos del iris impiden la entrada de luz a través del mismo, excepto en la apertura central, la pupila. El tamaño de la pupila es controlada por dos músculos (midriasis y miosis) que son los encargados de su dilatación y contracción. La pupila tiene un diámetro medio de 3 mm, pero esta se puede dilatar y alcanzar los 9 mm.
  - El cristalino está situado detrás del iris. Este tiene forma de lente biconvexa y su función principal es la de enfocar la imagen a diferentes distancias. Gracias a esta parte, los seres vivos que constan de ojos son capaces de percibir la profundidad.
  - La coroides está situada entre la retina y la esclerótica. Esta parte tiene como función la de irrigar los elementos del globo ocular puesto que es una membrana constituida por vasos sanguíneos.

- La capa interna, también conocida como retina, comprende desde los músculos ciliares hasta la parte posterior del ojo. Esta capa es la encargada de transformar la luz en impulsos eléctricos, que se envían al cerebro a través del nervio óptico. Esta estructura es la más compleja del ojo puesto que consta de las neuronas más especializadas de todo nuestro cuerpo. En la retina podemos destacar 3 estructuras: la mácula, la fóvea y el disco óptico.

  - La mácula está situada en la parte posterior de la retina. La principal función de la mácula es percibir los estímulos luminosos.

  - La fóvea es la zona central de la mácula, es muy característica en las imágenes ya que posee una zona avascular. La fóvea es la encargada de percibir los colores.

  - El disco óptico es el encargado de enviar las señales que se captan en el globo ocular al cerebro. Este se encuentra en el centro de la retina y tiene una forma circular.

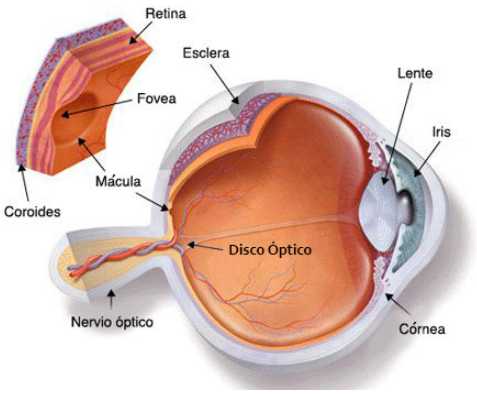


Figura 2.1: Anatomía del ojo humano.

El órgano visual está compuesto por cinco partes diferenciadas como ya hemos comentado. La pupila, el iris, la retina, la córnea y el cristalino. Todas estas partes funcionan conjuntamente para que se produzca la visión. La luz pasa por la pupila y atraviesa el cristalino, de esta forma se proyecta sobre la retina, donde, gracias a las células fotoreceptoras, esta luz es transformada en impulsos eléctricos que el nervio óptico hace llegar al cerebro [3, 4]. La Figura 2.1 muestra una imagen representativa de la anatomía del ojo humano, donde podemos apreciar las principales partes estructurales que lo componen.

2.2 Dispositivos de captura de imágenes oftalmológicas

2.2.1 Tomografía de coherencia óptica (OCT)

La tomografía de coherencia óptica (OCT) es una técnica no invasiva de obtención de imágenes que utiliza la luz de baja coherencia para capturar imágenes bidimensionales y tridimensionales de resolución micrométrica dentro de medios de dispersión óptica (por ejemplo, tejidos biológicos). Se utiliza para la obtención de imágenes médicas. La tomografía de coherencia óptica se basa en la interferometría de baja coherencia, que suele emplear una luz cercana al infrarrojo. El uso de luz de longitud de onda larga permite penetrar en el medio de dispersión [5]. Los sistemas de tomografía de coherencia óptica disponibles en el mercado se emplean en diversas aplicaciones, sobre todo en oftalmología y optometría, donde puede utilizarse para obtener imágenes detalladas del interior de la retina, permitiendo el diagnóstico, control y seguimiento para el estudio de cortes histológicos de la retina *in vivo*.

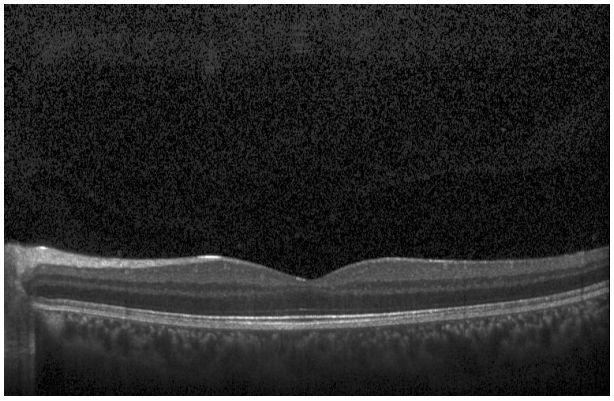


Figura 2.2: Imagen OCT de un ojo sano.

Las imágenes OCT son muy utilizadas por oftalmólogos y optometristas para obtener imágenes de alta resolución de la retina y del segmento anterior y posterior. Debido a la capacidad de la OCT de mostrar secciones transversales de las capas de tejido con una resolución micrométrica, la OCT proporciona un método sencillo para evaluar la organización celular, la integridad de los fotorreceptores, la degeneración macular (como puede ser la neovascularización coroidea, las drusas...), el edema macular diabético y otras enfermedades oculares o patologías sistémicas que presentan signos oculares [6]. En la Figura 2.2 podemos apreciar un ejemplo representativo de una imagen OCT de un ojo sano.

2.2.2 Angiografía por tomografía de coherencia óptica (OCT-A)

La Angiografía por tomografía de coherencia óptica (OCT-A, Optical coherence tomography angiography) es una nueva técnica de imagen de alta resolución de la circulación

coroidea y retinal. Esta prueba oftalmológica se lleva a cabo sin la inyección de un medio de contraste y por tanto es una técnica no invasiva. Al no ser invasiva, esta no causa ningún tipo de daño al globo ocular y por tanto se puede hacer de manera reiterada. Esta técnica es utilizada para el análisis del fondo de ojo y es capaz de generar imágenes de la estructura vascular retiniana a distintas profundidades, basándose en la detección de movimiento del flujo sanguíneo. Esto nos permite llegar más lejos que las imágenes OCT ya que se podría crear un mapa 3D del ojo. El funcionamiento de esta técnica es muy similar al de las imágenes OCT, pero cambiando el plano de vista. Esta prueba médica puede ayudar al diagnóstico precoz de la oclusión de la vena retiniana [7]. En la Figura 2.3 podemos apreciar un ejemplo representativo de una imagen OCT-A de un ojo sano.

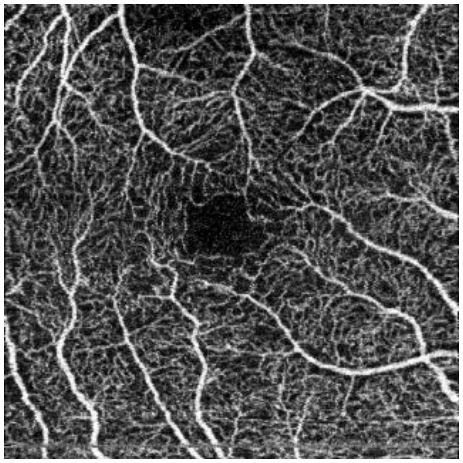


Figura 2.3: Imagen OCT-A de un ojo sano.

2.2.3 Patologías oculares

Una patología ocular es aquella que describe un problema localizado en el ojo y que impide su funcionamiento correcto. Entre estas patologías se pueden encontrar las siguientes: conjuntivitis, distrofias, cataratas, edema macular, drusas, neovascularización, oclusión de la vena retiniana, entre otras. Las patologías oculares de interés clínico que se abordarán en este proyecto se describirán con más detalle a continuación.

Edema macular diabético (DME)

El edema macular diabético (DME, *Diabetic macular edema*) es una patología que se produce en la retina debido a la diabetes. Es una de las principales causas de ceguera irreversible en los países desarrollados [8]. El DME afecta a los vasos sanguíneos periféricos, incluidos los de la retina, que pueden filtrar líquido, sangre y, en ocasiones, grasas en la retina, provocando su inflamación y dañando así la retina. En la Figura 2.4 podemos apreciar un ejemplo

representativo de una imagen OCT de un paciente diagnosticado con DME.

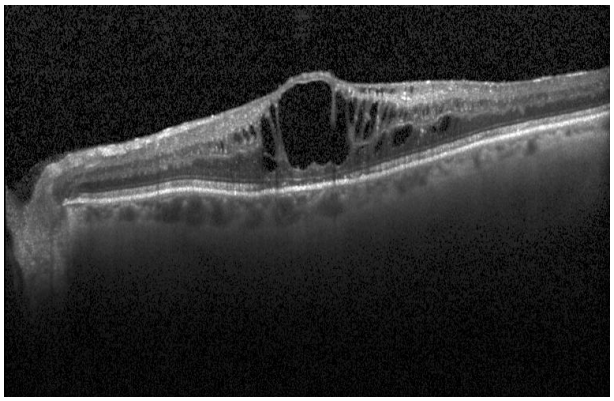


Figura 2.4: Imagen OCT de un paciente diagnosticado con DME.

Neovascularización coroidea (CNV)

La neovascularización coroidea (CNV, *Choroidal neovascularization*) es una patología que se caracteriza por la creación de nuevos vasos sanguíneos en la capa coroidea del ojo. La neovascularización coroidea es una causa común de la maculopatía y degeneración macular. Esta patología produce distorsión u ondulación de la visión central o manchas oscuras en la visión central. La CNV puede producir visión borrosa, visión distorsionada o puede crear un punto ciego o mancha oscura en el centro de la visión. La Figura 2.5 muestra una ejemplo representativo de imagen OCT de un paciente diagnosticado con neovascularización coroidea.

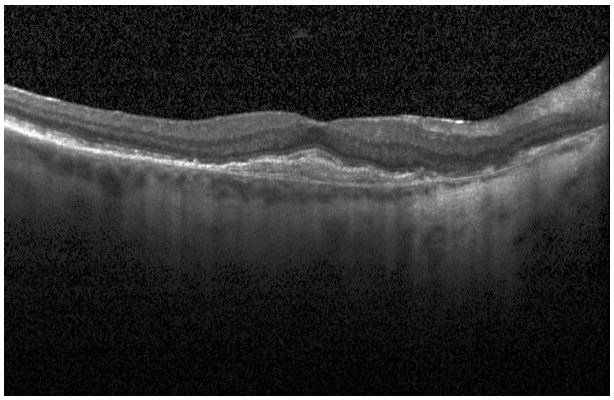


Figura 2.5: Imagen OCT de un paciente diagnosticado con CNV.

Drusas (DRUSEN)

Las drusas se caracterizan por ser acumulaciones amarillentas de poco tamaño y de un material extracelular que se acumulan entre la membrana de Bruch y el epitelio pigmentario

de la retina del ojo. La aparición de estas acumulaciones está asociada al avance de la edad. Por otro lado, si estas acumulaciones presentan un tamaño considerable, pueden significar un signo temprano de la degeneración macular. Esta patología puede ser asintomática, pero también puede producir visión borrosa, problemas en la visión con los cambios de luz, dificultad en la visión central, aparición de puntos o manchas oscuras y pérdida de agudeza visual. En la Figura 2.6 podemos apreciar un ejemplo representativo de una imagen OCT de un paciente diagnosticado con drusas.

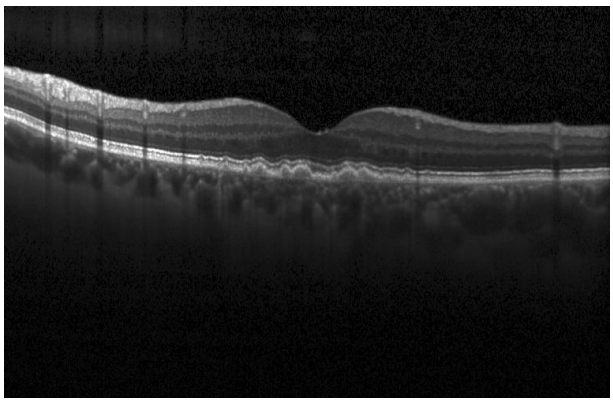


Figura 2.6: Imagen OCT de un paciente diagnosticado con drusas.

Oclusión de la vena retiniana (RVO)

La oclusión de la vena retiniana (RVO, *Retinal Vein Occlusion*) es un trastorno vascular común de la retina y la segunda causa más común de pérdida de visión en todo el mundo después de la DME. Esta patología es producida por la obstrucción de una arteria o vena. Este proceso se denomina oclusión o derrame. La oclusión de la vena retiniana suele ser causada por un coágulo de sangre que está bloqueando dicha vena y por tanto el flujo de sangre se bloquea [9]. La Figura 2.7 muestra una ejemplo representativo de imagen OCT-A de un paciente diagnosticado con oclusión de la vena retiniana.

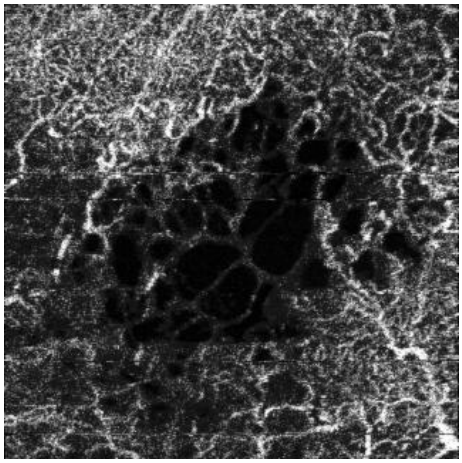


Figura 2.7: Imagen OCT-A de un paciente diagnosticado con RVO.

Estos tipos de imágenes servirán de entrada al módulo inteligente, que tendrá que distinguirlas para clasificarlas correctamente. Las principales diferencias visuales en las imágenes (OCT/OCT-A) de las citadas patologías pueden ser apreciadas en la Figura 2.8.

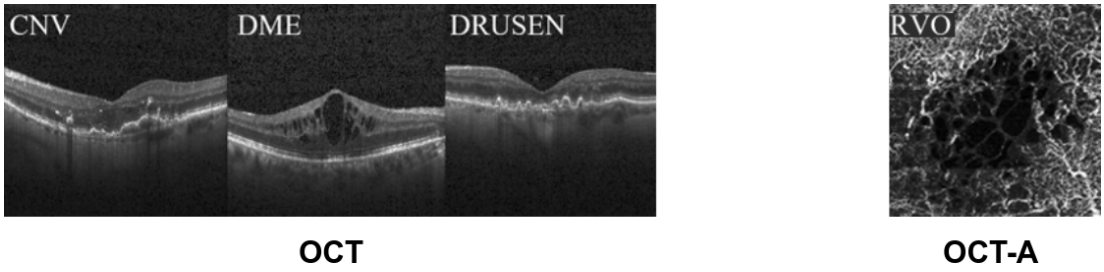


Figura 2.8: Imagen comparativa sobre las patologías abordadas en este proyecto.

2.3 Técnicas de aprendizaje máquina

2.3.1 Aprendizaje máquina clásica

El aprendizaje máquina (ML por su nombre en inglés: *Machine Learning*) es una subdisciplina de la rama de la Inteligencia Artificial (AI por su nombre en inglés: *Artificial Intelligence*) dedicada al estudio de algoritmos y modelos computacionales que se utilizan para realizar un conjunto de tareas sin utilizar instrucciones explícitas. En particular, estos algoritmos de aprendizaje automático construyen un modelo matemático con el fin de hacer predicciones o tomar decisiones sin estar explícitamente programados con esa finalidad. Así, estos algoritmos de aprendizaje utilizan técnicas de aprendizaje inductiva, por lo que aprenden de los ejemplos presentados mediante un proceso de generalización. Más concretamente, el aprendizaje inductivo es una búsqueda heurística en un espacio de descripciones simbólicas, generado

por la aplicación de varias reglas de inferencia sobre las observaciones iniciales. Dentro del aprendizaje inductivo, tenemos diferentes tipos de problemas que se diferencian por el tipo de salida que el algoritmo debe predecir. Tenemos principalmente tres grupos: clasificación, regresión y ranking.

En el caso de la clasificación, se intenta predecir una etiqueta entre un conjunto de etiquetas predefinidas. En el caso de la regresión se intenta predecir un valor real según valores pasados. En el caso del ranking, se intenta predecir el orden de un conjunto de objetos según su relevancia. Conociendo estos problemas en el aprendizaje máquina clásico, tenemos diferentes algoritmos que se clasifican dependiendo del tipo de entrada, salida y del tratamiento de datos. Los más habituales son el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje por refuerzo.

- En el aprendizaje supervisado se genera una función que establece una correspondencia entre las entradas y las salidas deseadas del sistema, donde la base de conocimientos del sistema está formada, por ejemplo, por etiquetas previamente definidas. Este aprendizaje se adapta perfectamente a los problemas de clasificación.

- En el aprendizaje no supervisado, se genera una función en la que no existía previamente a la creación del modelo una correspondencia entre las entradas y salidas. El proceso de modelado se lleva a cabo sobre un conjunto de ejemplos formados únicamente por entradas al sistema, sin conocer su clasificación correcta. Los problemas de regresión son válidos para este tipo de aprendizaje.

- Por último, en el caso del aprendizaje por refuerzo, el algoritmo aprende continuamente y con un flujo constante de información. De esta forma, el algoritmo realiza un proceso de ensayo y error, reforzando aquellas acciones que reciben una respuesta positiva permitiendo así mejorar constantemente el algoritmo [10].

En la Figura 2.9, podemos ver un ejemplo representativo de un pipeline que usa técnicas clásicas de aprendizaje maquina. En particular, podemos apreciar que la etapa de extracción de características es externa a la etapa de aprendizaje máquina.

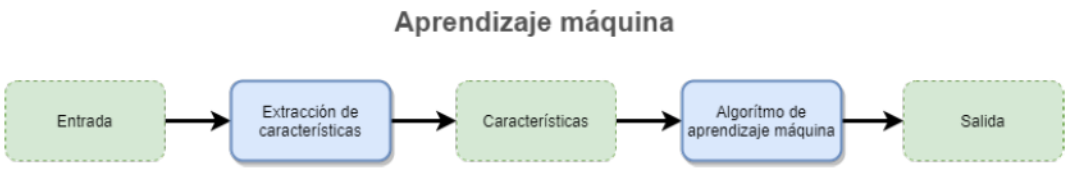


Figura 2.9: Ejemplo representativo de un pipeline que usa técnicas clásicas de aprendizaje maquina.

2.3.2 Deep Learning

El *Deep Learning* o aprendizaje profundo es una rama del *Machine Learning* que tiene como base la organización del conocimiento en un número superior de capas en comparación a otras ramas. Esta parte del ML supuso un importante cambio de paradigma ya que, hasta este punto, la tendencia era utilizar redes de neuronas con un número de neuronas elevado por cada capa, siendo el número de estas capas muy reducido. Por el contrario, esta tecnología busca cada vez más profundidad en vez de anchura. Esto quiere decir que el número de capas será mucho mayor y, al contrario de lo que se utilizaba hasta el momento, habrá un número de neuronas no tan elevado por capa. Estos cambios han producido mejoras en el rendimiento y en el comportamiento de las redes neuronales donde la solución al problema era mucho más compleja y las otras redes no podían llegar.

Este paradigma no habría sido posible sin los avances tecnológicos en las unidades de procesado de gráficos (GPU por *Graphics Processing Unit*) y es que, las principales operaciones matemáticas en el proceso de entrenado de una red neuronal son multiplicaciones de matrices, en las cuales las GPU son altamente eficientes. Esto ha permitido diseñar y acelerar el aprendizaje de redes mucho mayores.

Gracias a estos cambios, el punto de vista desde el cual se entrena una red neuronal también ha cambiado. Con *Deep Learning* ya no es necesario preprocesar las entradas para las redes y es por ello que son las propias redes las que extraen las características para realizar la adquisición de conocimiento. Esto ha originado que el proceso de entrenamiento se vea simplificado ya que, la fase de extracción de características, de la que se encargaba la ingeniería de características, ha desaparecido. Además, esto ha permitido que las propias redes neuronales se especialicen en base a los datos de entrada, creando ellas mismas una estructura óptima para realizar la tarea, mejorando así los resultados de las redes existentes [11]. En la Figura 2.10, podemos ver un ejemplo representativo de un pipeline que usa técnicas de aprendizaje profundo. En particular, podemos apreciar que la etapa de extracción de características es interna a la etapa de aprendizaje máquina.

Figura 2.10: Ejemplo representativo de un pipeline que usa técnicas de aprendizaje profundo.

2.3.3 Redes de neuronas artificiales

Una red de neuronas artificiales se basa en un conjunto de nodos conectados llamados neuronas artificiales. Estos intentan imitar el modelo de las neuronas de un cerebro biológico. Cada conexión, como las sinapsis de un cerebro biológico, puede transmitir una señal a otras neuronas. Una neurona artificial que recibe una señal la procesa y puede enviar señales a las neuronas conectadas con ella. Las señales que se transmiten entre las neuronas artificiales son un número real, mientras que la salida de cada neurona se calcula mediante una función que tiene como parámetros las entradas de esa misma neurona. Las neuronas suelen tener un peso que se ajusta a medida que avanza el aprendizaje. El peso modifica la fuerza de la señal de conexión entre las neuronas. A mayores, pueden tener un umbral, de manera que sólo se envía una señal si el valor agregado cruza ese mismo. Normalmente, las neuronas se agrupan en capas y cada una de ellas puede realizar diferentes transformaciones en sus entradas. En una red neuronal se pueden identificar tres tipos de capas, tal como podemos ver en la Figura 2.11. La primera capa (o capa de entrada), capas intermedias (u ocultas) y última capa (o capa de salida). Las señales viajan desde la capa de entrada hasta la capa de salida, pasando por las capas ocultas. Todas aquellas capas situadas entre la de entrada y la de salida son capas ocultas, pudiendo haber cero o más capas de este tipo [12].

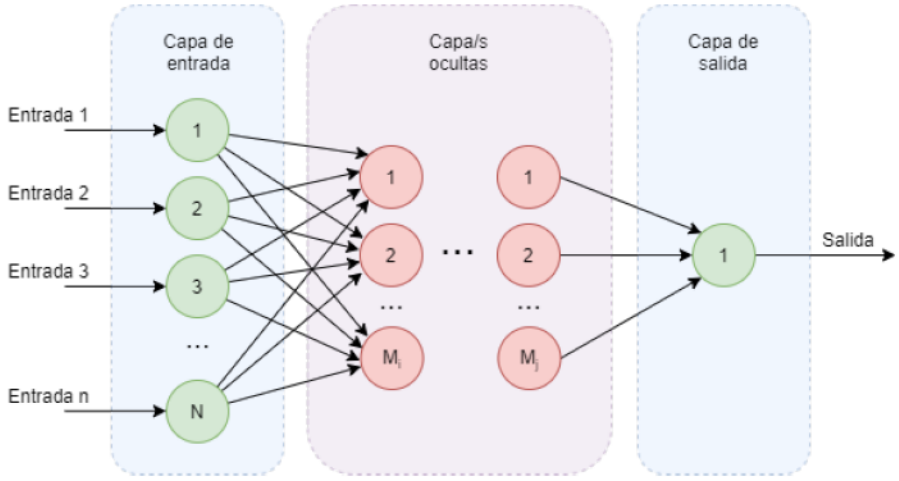


Figura 2.11: Estructura general de las redes neuronales.

2.3.4 Redes de neuronas convolucionales

En el caso de una red neuronal con una imagen como entrada, esta tendría que cumplir al menos una característica: la capa de entrada debe tener el mismo número de neuronas como de píxeles. Esto implica que introducir como entrada una imagen con una calidad estándar sea inviable, ya que el número de parámetros de la red sería demasiado grande produciendo

así una excesiva carga computacional. Es por ello que aparecen las redes neuronales convolucionales que se llamarán de aquí en adelante CNN (*Convolutional Neural Networks*) y que son el tipo de red neuronal más utilizado en aplicaciones de visión artificial. Las CNN son un tipo de red neuronal perteneciente al grupo de redes de *Deep Learning*. Este tipo de red tiene como principal operación las convoluciones para lograr llevar a cabo una tarea determinada. La principal función de este clase de estructuras es la de extraer las características necesarias para poder transformar la entrada correspondiente (una imagen) en una salida (una categoría o clase).

La convolución consiste en tomar grupos de píxeles cercanos de una imagen de entrada y operar mediante el producto escalar contra el *kernel* (pequeña matriz de valores). Este *kernel* es el encargado de recorrer las neuronas de entrada y generar una matriz de salida que pasará a ser la siguiente capa de neuronas. Típicamente, un *kernel* no opera de forma aislada si no que pertenece a un grupo de *kernels* (también conocidos como filtros). Estos filtros generan nuevas imágenes a partir de una sola de entrada. Las salidas corresponderán con características extraídas de esa primera imagen. Como se podría deducir, el crecimiento exponencial de neuronas al realizar esta tarea es excesivo y es por ello que se emplea el *Pooling* o *Subsampling*, con el cual reducimos el número de neuronas para, nuevamente, pasárselas como entrada a otra capa.

El *Pooling* es un proceso para reducir progresivamente el tamaño espacial de la representación y así reducir la cantidad de parámetros y cálculos en una red. En otras palabras el objetivo de esta función es reducir la muestra de una representación de entrada reduciendo su dimensionalidad y permitiendo que se hagan suposiciones sobre las características contenidas en las subregiones agrupadas. Después de este proceso, deberán prevalecer las características más importantes que ha detectado cada filtro.Este proceso de convolución y de *pooling* se repetirá las veces necesarias formando así la fase de extracción de características.

Después de esta fase, los datos llegan a la fase de clasificación. En este momento, los datos tienen un nivel de detalle lo suficientemente profundo como para ser capaz de diferenciar una serie de características únicas para cada imagen de entrada, y es por esto que la última fase, la de clasificación, tiene el poder de clasificar estas características hacia una clase u otra. Para esta fase se utiliza la técnica de redes neuronales tradicional [13]. En la Figura 2.12 se pueden ver las dos fases descritas con sus respectivos componentes.

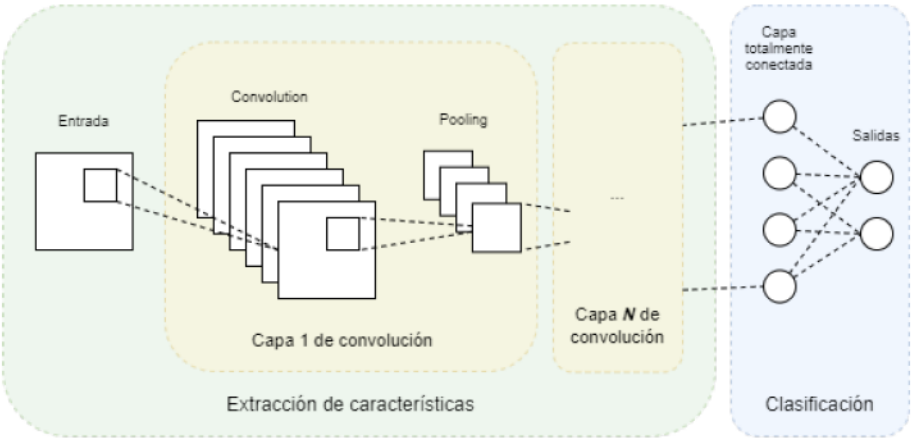


Figura 2.12: Arquitectura de una red neuronal convolucional.

2.3.5 Pre-entrenamiento

Durante la etapa de entrenamiento de una red neuronal se calculan los pesos óptimos en las conexiones de las neuronas para un correcto funcionamiento del modelo. Estos pesos son extraíbles de una red entrenada, permitiendo exportarlos a otras redes que aún no lo han sido, ya que, este proceso conlleva mucho tiempo y necesita una gran cantidad de recursos computacionales. Esta práctica es común en las redes neuronales de reconocimiento de imágenes. De esta manera aparecen las redes pre-entrenadas. Estas redes permiten un uso instantáneo de las mismas puesto que ya tienen un *conocimiento base* que les permite orientar diferentes problemas. Además, estos modelos pueden ser fácilmente adaptables para unas actividades concretas a través de lo que se denomina *fine-tuning* y *transfer learning*:

- El *fine-tuning* es el proceso por el cual una red se ajusta muy precisamente para que el funcionamiento sea óptimo a la hora de desarrollar una tarea.
- El *transfer learning* es el proceso por el cual el conocimiento de un modelo en un problema concreto es extraído para ser llevado a otro, el cual guarda cierta similitud con el primero. Lo habitual en este proceso es modificar la capa de salida para contemplar únicamente las clases interesadas. Con esto se parte de una base con unos pesos estabilizados permitiendo un entrenamiento más eficiente y con una convergencia más rápida. Esto permite además el uso de *datasets* más pequeños al tener un conocimiento previo.

En el caso de este proyecto, se hace uso de una arquitectura *DenseNet-161* pre-entrenada, que será explicada con mayor profundidad en el apartado 6.2.1. Concretamente, esta arquitectura ha sido pre-entrenada usando el *dataset ImageNet*, una base de datos diseñada para

el reconocimiento visual de objetos. Esta base de datos está compuesta por más de catorce millones de imágenes agrupadas en más de veinte mil categorías [14]. Esta variedad de información e imágenes en el entrenamiento de redes neuronales ha supuesto un incremento de la eficiencia en las CNN [15]. En la Figura 2.13 podemos ver el uso del transfer learning para adaptar un modelo pre-entrenado a nuestro proyecto.

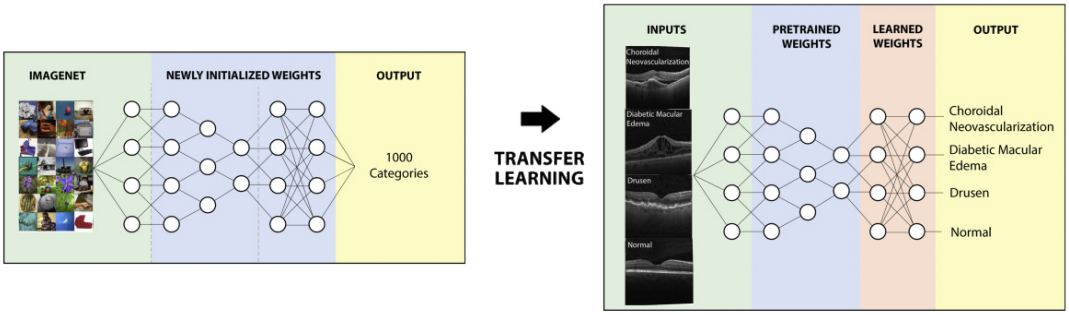


Figura 2.13: Ejemplo de uso de un modelo pre-entrenado con *ImageNet* (acotado con 1000 clases) y *transfer learning*.

2.3.6 DenseNet

La arquitectura DenseNet (*Dense Convolutional Nerwork*) fue presentada en 2016 en la prestigiosa *(Conference on Computer Vision and Pattern Recognition - CVPR)*, por Huang *et al.* [16], y es una arquitectura basada en neuronales convolucionales densamente conectadas que están dirigidas principalmente a la clasificación de imágenes.

Esta arquitectura se desarrolló en un intento de resolver varios problemas comunes en otras arquitecturas. En particular, en algunos casos, el recorrido de la información desde la capa de entrada hasta la capa de salida se hace tan grande, que pueden desaparecer antes de llegar al otro lado. Con la DenseNet, y a diferencia de otras arquitecturas CNN, se simplifica el patrón de conectividad entre capas, creando conexiones cortas entra las capas iniciales y finales. Además, esta arquitectura se caracteriza por utilizar menos parámetros que otras arquitecturas, ya que no es necesario aprender mapas de características redundantes. Adicionalmente, otra ventaja que presenta, es la diferenciación entre la información nueva y la información ya presente en la red, lo que permite una mayor gestión en los parámetros. Por otro lado, el acceso a los gradientes y a la función de *loss* por parte de cada capa permite una supervisión implícita que ayuda, regula y mejora la eficiencia en los entrenamientos en arquitecturas profundas, permitiendo también reducir el *overfitting* en entrenamientos donde el conjunto de datos es reducido [17].

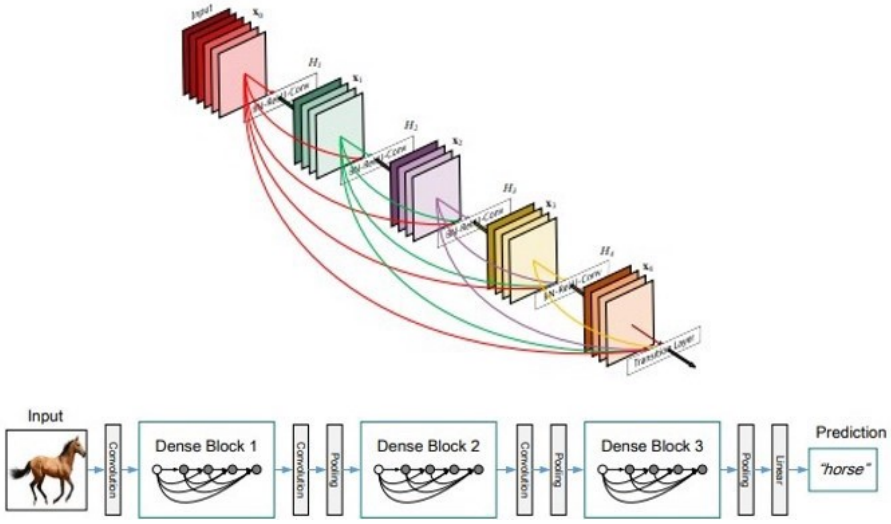


Figura 2.14: Esquema representativo de la arquitectura DenseNet.

A continuación se detallarán las distintas partes que componen la arquitectura DenseNet. Este tipo de redes están compuestas de bloques densos (*Dense Blocks*) que son capas densamente conectadas entre ellas, es decir, que cada capa recibe como entrada el mapa de características de salida de las capas previas. Eso se debe a que a cada bloque denso se le añade el mapa de características de la capa anterior y de todas las antecesoras, que se concatenan y permiten crear un mapa de gran profundidad, del mismo tamaño que cada capa. Estos bloques están compuestos por varias capas, teniendo cada uno su mapa de características. Cada capa está formada por una operación de *batch normalization*, una *ReLu* y una convolución de 3×3 con N filtros, donde N denota el ratio de crecimiento. Los bloques densos están unidos por capas de transición que se componen de capas de convolución y de *pooling*. La última capa de esta arquitectura corresponde con una capa de clasificación totalmente conectada con *softmax* [1, 17]. La Figura 2.14 muestra un esquema representativo de la arquitectura DenseNet y sus principales componentes.

2.4 Estado del arte

Dada la gran relevancia de esta temática, diversos autores han abordado el desarrollo de sistemas inteligentes para el diagnóstico de diferentes patologías oculares en imágenes OCT. Como referencia, en el trabajo de Yoon *et al.* [18], se describe un modelo computacional capaz de identificar la coriorretinopatía serosa central, una enfermedad caracterizada por el desprendimiento de la retina neurosensorial en el polo posterior del ojo. En particular, los autores de este trabajo proponen una metodología computacional que consigue alcanzar la precisión que tienen los expertos oftalmológicos con más de 10 años de experiencia. Para ello, se emplean

técnicas de *Deep Learning* en imágenes OCT. Asimismo, Vidal *et al.* [19] propone un análisis comparativo entre diferentes metodologías para la identificación de regiones quísticas de pacientes diagnosticados con EMD en imágenes OCT. Por tanto, este y otros estudios similares [20, 21, 22, 23] demuestran que el uso de las técnicas de *Deep Learning* en imágenes OCT para el apoyo al diagnóstico de patologías oculares es una realidad presente en el campo clínico.

Por otro lado, debido a la reciente aparición de la modalidad OCT-A y el uso generalizado de estos dispositivos de captura hacen que este campo esté en constante cambio y crecimiento. En este sentido, algunos autores propusieron trabajos utilizando técnicas de *Deep Learning* para identificar diferentes estructuras y patologías oculares. Como referencia, Le *et al.* [24] comparan el uso del aprendizaje máquina clásico con el *Deep Learning*. En este trabajo, los autores demostraron que el *Deep Learning* es una herramienta viable para la clasificación automática de arterias y venas, la segmentación de áreas avasculares y otras regiones de interés en las imágenes OCT-A. Propuesta similares fueron presentadas para la identificación de otras estructuras y patologías oculares [25, 26, 27, 28]. Por otra parte, estudios como el de Xi *et al.* [29] hacen uso del *Deep Learning* no para el diagnóstico, si no para el mejorado de las imágenes de fondo de ojo, ofreciéndolo como alternativa a los algoritmos de análisis convencionales. Concretamente, este estudio se lleva a cabo sobre imágenes OCT-A para extraer y analizar la variación de la señal OCT en diferentes puntos temporales.

En cuanto a la parte de gestión clínica, sí existen sistemas que permiten el control y supervisión de pacientes e informes, tanto aplicaciones privadas como públicas, véase MNProgram[1], Pabau[2] o Nubimed[3] entre otros. Sin embargo, todas estas herramientas de gestión clínica carecen de módulos inteligentes de apoyo al diagnóstico usando las imágenes (OCT/OCT-A).

Por lo tanto, el desarrollo de un sistema unificado de gestión de pacientes y de apoyo al diagnóstico clínico de patologías oculares a través de *Deep Learning* en imágenes OCT y OCT-A sería un proyecto único. Es por este motivo que se ha escogido como tema central para este trabajo. Como se ha comentado, se parte con la información de que el uso de modelos inteligentes para el diagnóstico de patologías en imágenes de fondo de ojo es acertado y eficaz. Además, las características extraídas por los modelos integrados no solo son útiles para distinguir patologías oculares, si no que se podrían diferenciar tipos de imágenes de fondo de ojo, permitiendo así analizar las patologías dependiendo del tipo de imagen que tenga como entrada. Por lo tanto, tendremos diferentes modelos inteligentes integrados en la herramienta por cada dispositivo de captura, facilitando el diagnóstico y control de diferentes patologías oculares. Para ello, se diseñará una aplicación web con el fin de expandirse con nuevos modelos inteligentes, adaptándose así fácilmente a nuevas patologías descubiertas o a otros tipos de imágenes oftalmológicas.

[1] mnprogram.com
[2] pabau.com
[3] nubimed.com

Capítulo 3

Materiales

En este capítulo se describen los recursos materiales utilizados en este proyecto.

3.1 *Datasets*

A continuación se detallan los conjuntos de datos que se utilizaron para el entrenamiento, la validación y el test de los modelos computacionales de apoyo al diagnóstico clínico.

3.1.1 Retinal Images - (OCT)

Este conjunto de datos está compuesto únicamente de imágenes OCT. En particular, consta de 84.484 imágenes OCT, pertenecientes a 26.315 pacientes sanos, 37.205 pacientes diagnosticados de CNV, 11.348 pacientes diagnosticados de DME y 8.616 pacientes diagnosticados de DRUSEN. Por tanto, este *dataset* contiene 4 clases (NORMAL, CNV, DME, DRUSEN). Además, en este conjunto de datos las imágenes se etiquetan de la siguiente manera: (enfermedad)-(ID de paciente aleatorio)-(número de imagen de este paciente). Las imágenes de tomografía de coherencia óptica (OCT) que componen este dataset se seleccionaron de cohortes retrospectivas de pacientes adultos del *Shiley Eye Institute de la Universidad de California San Diego, la California Retinal Research Foundation, Medical Center Ophthalmology Associates*, el Shanghai First People's Hospital y el Beijing Tongren Eye Center. Además, cada imagen ha pasado por un sistema de clasificación por niveles que consistía en múltiples capas de clasificadores para la verificación y corrección de las etiquetas de las imágenes por parte de dos especialistas en retina independientes de alto nivel [30]. Este conjunto de datos está disponible públicamente a la comunidad científica en el sitio web de Kaggle[1].

[1] https://www.kaggle.com/paultimothymooney/kermany2018

3.1.2 RVO Screening - (OCT-A)

El conjunto de datos *RVO Screening - (OCT-A)* consta de 1.551 imágenes obtenidas por un dispositivo de captura OCT-A, donde 870 son imágenes de pacientes diagnosticados con RVO y 681 son imágenes sin presencia de RVO. Por lo tanto, sólo hay dos clases (RVO y NO-RVO). Todas las imágenes están etiquetadas de 0 a N para cada clase, cumpliendo el tratado de Helsinki sobre la anonimización de los pacientes que participaron en este estudio. El dispositivo de captura por tomografía con el que se han tomado las imágenes ha sido un *DRI OCT Triton de TOPCON Corp.* Las imágenes fueron obtenidas en el *Complejo Hospitalario Universitario de Santiago (CHUS)* por diferentes especialistas clínicos que identificaron y clasificaron las imágenes según la patología. Actualmente, este conjunto de datos es privado y, por tanto, no está a disposición de la comunidad científica.

Capítulo 4

Tecnologías

En este capítulo se describirán las tecnologías utilizadas en este proyecto así como la decisión del empleo de cada una. En este apartado tiene cabida tanto los lenguajes de programación como las librerías y los *frameworks* empleados. Se diferencian en 8 principales grupos.

1. Primeramente se describirán todos los elementos utilizados para el proceso de entrenamiento de los modelos computacionales generados descritos más adelante en este documento.

2. Seguidamente se explicarán los componentes relacionados al *Backend* propietario de los modelos entrenados así como de la generación y envío de informes.

3. Posteriormente se describirán los elementos empleados en el *Backend* encargado de la gestión clínica.

4. Por otro lado, se explicarán los componentes que conforman la parte del *Frontend*, es decir, la parte cliente de la aplicación.

5. En cuanto a la persistencia de datos, se explicará el sistema de base de datos empleado y la decisión de escoger este mismo.

6. Después, se detallará la herramienta utilizada para el despliegue de esta aplicación.

7. Luego se esclarecerá cómo y qué se ha utilizado para el control de versiones en el desarrollo de la aplicación.

8. Por último, se explicará y aclarará qué herramienta se ha empleado para la gestión del proyecto, siendo este necesario para la planificación, organización y seguimiento del mismo.

4.1 Módulo de entrenamiento de los modelos computacionales

A continuación se describirán las tecnologías utilizadas para la creación del módulo de entrenamiento utilizado en las aproximaciones computacionales.

4.1.1 Lenguaje de programación: Python

Para este módulo se ha hecho uso del lenguaje de programación Python[1], un lenguaje de programación interpretado de alto nivel y de propósito general. Además, es de código abierto y soporta diferentes paradigmas como programación orientada a objetos, programación imperativa y programación funcional. Todo esto hace que Python sea el lenguaje de inteligencia artificial por excelencia.

4.1.2 *Framework* de Machine Learning: Pytorch

PyTorch[2] es una biblioteca de aprendizaje automático de código abierto basada en la biblioteca Torch, utilizada para aplicaciones como la visión artifical y el procesamiento del lenguaje natural. Esta biblioteca fue desarrollada principalmente por Facebook. Se ha escogido este *Framework* puesto que está ganando popularidad tanto por su simpleza como por su eficiente uso de la memoria.

4.1.3 Librería de entrenamiento supervisado y no supervisado: Scikit-learn

Scikit-learn[3] es una biblioteca de aprendizaje automático de software libre para Python. Esta biblioteca incluye una gran variedad de algoritmos de clasificación y de regresión, entre otros. Será útil en este proyecto para el empleo de los algoritmos de clasificación complementando su uso con Pytorch.

4.1.4 Librería para la generación de gráficos: Matplotlib

Matplotlib[4] es una biblioteca multiplataforma de visualización de datos y gráficos para Python. Esta biblioteca ofrece una alternativa de código abierto a MATLAB. Se empleará para la representación de los datos de entrenamiento.

[1] python.org
[2] pytorch.org
[3] scikit-learn.org
[4] matplotlib.org

4.2 *Backend* de procesado

A continuación se describirán las tecnologías utilizadas para el desarrollo de una *API REST* encargada de procesar y clasificar las imágenes, así como de generar informes clínicos y enviarlos por email.

4.2.1 Lenguaje de programación: Python

Una vez más se ha utilizado Python[5] para esta tarea. Esta vez, los motivos que nos han llevado a tomar esta decisión es la cantidad de librerías disponibles para llevar acabo las tareas anteriormente comentadas. Además, el uso de las mismas es muy sencillo puesto que es un lenguaje de alto nivel.

4.2.2 *Micro-Framework* para la creación de APIs: Flask

Flask[6] es un Micro-framework web desarrollado en Python. En concreto se ha empleado Flask-RESTful[7], una biblioteca adicional para Flask que permite desplegar rápidamente APIs REST. Se ha escogido este framework debido a su simpleza, puesto que no es necesaria ninguna tarea compleja de configuración.

4.2.3 *Framework* para la creación de PDFs: Pyfpdf

Pyfpdf[8] es una librería escrita en Python bajo el paradigma de orientación a objetos. La principal función de esta librería es la generación de documentos PDF. La elección de esta librería ha sido casi obligada debido a la ausencia de diversidad de librerías que puedan cumplir con los requisitos necesarios para llevar a cabo las tareas esperadas.

4.3 *Backend* de gestión clínica

Seguidamente, se describirán las tecnologías utilizadas para el desarrollo de una API REST encargada de gestionar pacientes y clínicos, así como de las operaciones básicas para el seguimiento y diagnóstico de patologías en imágenes oftalmológicas.

[5] python.org
[6] flask.palletsprojects.com
[7] flask-restful.readthedocs.io
[8] pyfpdf.readthedocs.io

4.3.1 Lenguaje de programación: Java

Para desarrollar esta parte, se ha empleado el lenguaje de programación de propósito general, orientado a objetos y basado en clases Java[9]. Este lenguaje ha sido desarrollado con la finalidad de que el dispositivo donde se vaya a ejecutar sea transparente para el desarrollador, sin necesidad de que el programa sea recompilado. Esto se debe a que los programas de Java se ejecutan en la JVM (*Java Virtual Machine*), una máquina virtual encargada de interpretar el código binario Java compilado. Se ha escogido este lenguaje debido a su uso extendido en el sector, así como la cantidad de *frameworks* y bibliotecas que ofrece. Por otro lado, gracias al *Java Garbage Collection* o recolector de basura Java, este presenta una alta eficiencia y un reducido consumo de recursos, extrayendo al desarrollador de cualquier tipo de creación o borrado de objetos.

4.3.2 Herramienta de compilación: Maven

Maven[10] es una herramienta desarrollada en Java para la automatización de la compilación en proyectos principalmente Java. Maven se basa en el concepto del modelo de objetos del proyecto (POM) y utiliza una base de repositorios para administrar archivos Java comprimidos (*jar*). Además, es la encargada de gestionar las dependencias del proyecto y tiene la capacidad de automatizar la compilación y la ejecución de pruebas. Como se ha comentado, los archivos Java están comprimidos, de esta función también se encarga Maven, que, a través del popular formato de compresión ZIP, los convierte en archivos *jar* o *war*.

4.3.3 *Framework* para el desarrollo de aplicaciones web: Spring Boot

Spring Boot[11] es un *framework* de código abierto para el desarrollo de aplicaciones web en Java. Este *framework* facilita el desarrollo de aplicaciones web extrayendo al desarrollador de tareas repetitivas y aumentando la productividad, reduciendo también la fricción. Las tareas que facilitan van desde la inyección de dependencias hasta las pruebas unitarias de código, pasando por acceso a datos e inversión de control. Proporciona, a mayores, herramientas para implementar de forma sencilla la autenticación y el control de acceso a la aplicación. Por otro lado, este módulo proporciona una infraestructura ligera que permite eliminar la mayor parte del trabajo de configuración de las aplicaciones basadas en Spring. En el caso de Spring Boot ya cuenta con un servidor integrado haciendo más fácil la tarea de distribución de aplicaciones permitiendo que el escalamiento horizontal sea más sencillo. Se ha escogido este *framework* debido a que el alumno ya tenía conocimientos previos y además la curva de aprendizaje es la apropiada para este proyecto.

[9] java.com
[10] maven.apache.org
[11] spring.io/projects/spring-boot

4.3.4 *Framework* para la creación de pruebas unitarias: JUnit

JUnit[12] es un marco de pruebas unitarias de código abierto para Java. Es uno de los *frameworks* más utilizados para las pruebas de regresión. Se ha escogido para este proyecto puesto que es una opción muy popular y su integración en IDEs como Eclipse viene instalada y configurada por defecto.

4.3.5 *Framework* para la creación de *mocks* en pruebas unitarias: Mockito

Mockito[13] es un *framework* de código abierto para Java destinado a la creación de pruebas. Esta herramienta permite crear objetos *mock*, que son objetos que simulan el comportamiento de objetos reales, para pruebas unitarias, facilitando así el desarrollo de las mismas.

4.3.6 Herramienta de creación y prueba de APIs: Postman

Postman[14] es una plataforma de colaboración para el desarrollo de APIs. Permite el envío de peticiones REST, SOAP y GraphQL sin la necesidad de un cliente. También permite simular un *backend* para probar clientes. Además, permite la creación de pruebas de integración y de requisitos no funcionales ya que muestra el rendimiento y el tiempo de respuesta de las peticiones.

4.4 *Frontend* de aplicación web

Posteriormente, se describirán las tecnologías utilizadas para el desarrollo de un cliente encargado de mostrar una interfaz para los clínicos donde podrán gestionar pacientes e imágenes oftalmológicas. Este *frontend* se comunicará con el *Backend* de gestión clínica.

4.4.1 Lenguaje de programación: Javascript con ReactJs

Para desarrollar el *Frontend* se ha empleado el lenguaje de programación JavaScript[15], de alto nivel, comúnmente compilado *just-in-time* y multiparadigma. JavaScript, junto con HTML Y CSS, es una de las tecnologías más utilizadas de la *World Wide Web* [31]. Para el desarrollo del *Frontend*, nos hemos ayudado de la biblioteca ReactJs[16]. Con vistas declarativas y basada en componentes ReactJs es una biblioteca de código abierto desarrollada por Facebook multiplataforma y con la finalidad de construir *Single-page applications (SPA)*. React se ocupa

[12] junit.org
[13] site.mockito.org
[14] postman.com
[15] developer.mozilla.org/es/docs/Web/JavaScript
[16] es.reactjs.org

de la gestión del estado y de la representación de ese estado en el DOM. Se ha empleado esta tecnología debido a su popularidad, eficiencia, escalabilidad y simpleza.

4.4.2 Librería de gestión de estado: Redux

ReactJs no gestiona el estado, para ello haremos uso de la librería Redux[17]. Está desarrollada en JavaScript y es un código abierto. Su uso principal es el manejo del estado de las aplicaciones.

4.4.3 Librería de generación de gráficos: Recharts

Para representar gráficas dentro de la interfaz de usuario se emplea Recharts[18], una librería de código abierto para React.

4.4.4 Librería de internacionalización: Format.JS

Para internacionalizar la aplicación en la parte *Frontend* se hará uso de la librería de JavaScript Format.JS[19].

4.4.5 Librería de diseño: Material UI

Material UI[20] es una librería de componentes predefinidos para un desarrollo web más rápido y sencillo. Gracias a esta librería, el desempeño en el diseño de la aplicación se reduce en gran cantidad. Se empleará por su popularidad y su integración con ReactJs.

4.5 Persistencia de datos

En este apartado se comentará la solución empleada para la persistencia de datos en este proyecto. Ha sido únicamente necesario almacenar datos en el *Backend* de gestión clínica, puesto que para el *Backend* de los modelos computacionales una vez utilizadas y procesadas, las imágenes son eliminadas del sistema para mantener la privacidad de los pacientes.

4.5.1 Lenguaje de programación: MySQL

MySQL[21] es un sistema de gestión de bases de datos relacional de código abierto y propiedad de Oracle. MySQL ha sido la opción escogida para la persistencia de datos. Se almacenarán

[17] es.redux.js.org
[18] recharts.org
[19] formatjs.io
[20] material-ui.com
[21] mysql.com

en esta base de datos los detalles de los clínicos, pacientes, informes e imágenes. El motivo de esta decisión ha sido el conocimiento previo de esta tecnología y su popularidad.

4.6 Control de versiones

Un sistema de control de versiones es aquel que nos permite hacer un seguimiento de los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de forma que podamos acceder a diferentes versiones de ellos. Esto es muy útil en el desarrollo software ya que podemos hacer un seguimiento del estado del producto pudiendo llevar un registro y control de todas las versiones por las que pasa el proyecto.

4.6.1 Herramienta de control de versiones: Git

Git[22] es un software para el seguimiento de cambios en archivos. Es ampliamente utilizado para coordinar el trabajo entre los desarrolladores que trabajan conjuntamente en un mismo desarrollo software. Git se caracteriza por su eficiencia, fiabilidad y el soporte de múltiples flujos de trabajo distribuidos y no lineales. Se ha escogido Git frente a Subversion[23] porque es distribuido y más eficiente, además de contar con más experiencia por parte del alumno.

4.7 Gestión de proyecto

Para desarrollar un proyecto software es esencial utilizar una herramienta de gestión de proyectos para mantener un control y registro de las actividades y de los progresos que se van llevando a cabo.

4.7.1 Herramienta de gestión: Trello

La herramienta escogida ha sido Trello. Aunque en principio Trello[24] proporciona menos funcionalidades que Taiga o Jira, esta elección se debe a su simpleza y flexibilidad a la hora de adaptarlo a cualquier tipo de proyecto. Con la ayuda de *Power-Ups* ha sido posible adaptar esta herramienta tanto a proyectos con metodología Scrum como a proyectos con metodología más tradicionales.

[22] git-scm.com
[23] subversion.apache.org
[24] trello.com

4.8 Calidad de código

Un pilar fundamental en el desarrollo software es la calidad del producto, esto se consigue entre otras cosas con la calidad del código, por ello es importante seguir buenas prácticas de programación y mantener el código comentado y mantenible.

4.8.1 SonarQube

Para el control de calidad del código desarrollado en Java se ha usado SonarQube[25]. Esta plataforma de software libre permite evaluar el código fuente y obtener métricas que pueden ayudar a mejorar la calidad del código del programa.

4.8.2 ESLint

Para controlar la calidad del código desarrollado en JavaScript se ha usado ESLint[26]. Esta herramienta de linting para JavaScript es la encargada de revisar el código y de encontrar errores que podrían provocar problemas de compilación, *bugs* o malas prácticas.

4.9 Despliegue

El despliegue de software es el conjunto de actividades que hacen que un sistema de software esté disponible para su uso. En el caso de este proyecto se desplegará la aplicación para simular un entorno de producción, completando así el paso por todas las etapas del desarrollo y dejando a un lado únicamente el mantenimiento.

4.9.1 Herramienta para la creación de contenedores: Docker

Docker[27] es un herramienta de código abierto que automatiza el despliegue de aplicaciones software dentro de contenedores. Esto proporciona un patrón *fachada* para los diferentes módulos y una manera de conseguir automatizar la virtualización de aplicaciones independientemente del sistema operativo. Por otro lado, para desplegar la aplicación al completo se ha hecho uso del Compose, una herramienta perteneciente a Docker para definir y ejecutar varios contenedores con un solo comando. Esta herramienta utiliza un archivo YAML para la configuración de dichos contenedores. Gracias a esto, logramos desplegar varios contenedores en un mismo contenedor que correspondería con la aplicación en sí.

[25] sonarqube.org
[26] eslint.org
[27] docker.com

4.10 Entornos de desarrollo

Una vez terminado con las tecnologías, comentaremos el entorno de desarrollo que se ha utilizado para este proyecto. A la hora de elegir las herramientas a utilizar para el desarrollo de un proyecto, es fundamental conocer las necesidades del mismo, puesto que, un buen entorno de desarrollo puede mejorar en gran medida la productividad y la calidad del producto software. Un entorno de desarrollo integrado o *Integrated Development Environment (IDE)* es una aplicación software que proporciona facilidades a los desarrolladores para la creación de productos software. Las partes fundamentales de un IDE son un editor de código fuente, herramientas de automatización de la construcción (compilación, empaquetado y demás) y un depurador.

4.10.1 Eclipse

Para el desarrollo en Java se ha escogido el IDE Eclipse[28] debido a su popularidad, a que está altamente probado, a que cuenta con un sistema de *plugins*, a que permite la personalización del entorno y a que se integra con herramientas de control y versionado. Cuenta además con funciones básicas muy completas como puede ser el sistema de depurador o el compilador en tiempo real. Estas ventajas permiten que el desarrollador incremente su productividad y se centre en el desarrollo en sí.

4.10.2 Visual Studio Code

Para el desarrollo tanto de JavaScript como de Python se ha empleado Visual Studio Code[29] que no es un IDE en sí, pero es un editor de texto muy completo. Esta herramienta ha sido desarrollada por Microsoft y cuenta con una tienda de *plugins* muy variada que lo convierte en uno de los editores de código más utilizados hoy en día. A través de *plugins*, este editor cuenta con depurador, control de versiones, autocompleado y resaltador de sintaxis.

[28] eclipse.org
[29] code.visualstudio.com

Capítulo 5

Proceso de ingeniería

En este capítulo se describe la metodología utilizada en este proyecto. Como el sistema a desarrollar es una aplicación web típica se opta por una metodología ágil. Además se comentará la planificación y los costes del proyecto.

5.1 Metodología de desarrollo

En este apartado se describe la metodología empleada en el desarrollo de este proyecto *software* así como las características que han justificado la decisión tomada. El proceso de ingeniería explicado más adelante se ha elaborado utilizando una metodología que se apoya en la base teórica de Scrum, empleando su misma estructura y elementos, pero haciéndola más pragmática. Asimismo, se describen todos los componentes que forman el proceso de ingeniería aledaños a esta metodología.

5.1.1 Decisión

Se ha utilizado Scrum como metodología en este proyecto ya que los requisitos de la aplicación no estaban estrictamente definidos. Lo único establecido era una idea general de la funcionalidad principal y unos problemas que se resolverían en un primer momento. Siempre teniendo en mente que sería un sistema escalable al que no debería afectar la volatilidad de los requisitos, puesto que, en el campo de la oftalmología los avances tecnológicos producen continuamente nuevos métodos de diagnóstico de patologías.

Por otro lado, es una metodología que tiene una rápida detección de riesgos y permite solucionarlos lo antes posible. Además, gracias a las entregas parciales del producto durante todo el proceso de desarrollo, se puede obtener *feedback* de los usuarios lo que favorece la detección de riesgos antes mencionada y a mayores pueden presentar posibles mejoras al *software*.

Por estos motivos Scrum, al ser una metodología ágil, iterativa e incremental, se adapta al proyecto de tal manera que los requisitos se pueden ir refinando a medida que el desarrollo se lleva a cabo. La aparición de nuevos requerimientos no sería un problema para la aplicación. Todo esto permitiendo crear un producto tangible, valioso y útil en cada iteración [32].

5.1.2 Scrum

Scrum fue concebido como un estilo de gestión de productos en empresas de fabricación por Ikujiro Nonaka e Hirotaka Takeuchi en los años 80. Esta metodología es el resultado de observar que los proyectos que utilizaban equipos pequeños e interfuncionales producían los mejores resultados. La palabra que nombra esta metodología fue escogida por Nonaka y Takeuchi al comparar este proceso a la formación Scrum que se utiliza en el Rugby y que da nombre al avance de varios jugadores de forma conjunto [33]. Esta metodología se consolida con la llegada de Jeff Sutherland, John Scumniotales y Jeff McKenna que la diseñaron, documentaron e implementaron en los años 90 basándose en los principios observados por Nonaka y Takeuchi. De este trabajo sale como resultado Scrum Development Process en el año 95 por parte de Ken Schwaber y Jeff Sutherland. A continuación, se describirán las diferentes partes de Scrum: los roles, los artefactos y los eventos, tal como lo recoge la Guía de Scrum, desarrollada al igual que la del 95, y titulada *The Definitive Guide to Scrum: The Rules of the Game*. Se explicará también el flujo utilizado sobre esta metodología que se ha empleado en este proyecto.

Roles

En esta metodología existen unos roles definidos que representan la totalidad de los perfiles que participan en el proyecto. El equipo de Scrum tiene que ser auto-organizado y multifuncional. Esto quiere decir que el propio equipo es el encargado de elegir la mejor forma de llevar a cabo su trabajo. Estos no son dirigidos por nadie, por lo que no dependen de otras personas que no formen parte del equipo de Scrum. De este modo, la metodología se vuelve muy flexible pudiendo aplicarla a prácticamente cualquier marco de desarrollo. Los roles definidos por esta metodología son el equipo de desarrollo, el Scrum Master y el Product Owner.

- El **equipo de desarrollo** debe ser un equipo de profesionales lo bastante pequeño como para mantener esa estructura permitiéndole ser ágil, pero a su vez, lo bastante grande para que lleve a cabo una cantidad suficiente de trabajo que permita entregar un producto útil al final de cada incremento. Por estas razones, participarán entre 4 y 9 profesionales. Entre las tareas del equipo de desarrollo también se encuentra la descomposición de las historias de usuario en tareas así como la estimación de las mismas [33].

Cabe destacar que para este proyecto no se cumple con la cantidad de personas recomendadas para conformar un equipo de desarrollo y es por ese motivo que se utilizará, como se ha indicado anteriormente, una metodología basada en Scrum y no dicha metodología en sí misma. Por último indicar que este perfil será asignado al alumno del Trabajo de Fin de Grado.

- El **Scrum Master**, según Ken Schwaber y Heff Sutherland, es el responsable de asegurar que Scrum se entienda y se adopte. Este perfil toma la figura de líder dentro del equipo de desarrollo y es el intermediario entre este equipo y el Product Owner. Por otro lado, es el encargado de evitar impedimentos en el proceso de elaboración del producto dentro de ese mismo equipo eludiendo también impedimentos externos, haciendo así de este perfil un filtro que evita que el equipo pierda su objetivo. El Scrum Master, aunque forme parte del equipo de desarrollo, nunca lo gestionará si no que tendrá la función opuesta; ayudarlo a ser autoorganizado y multifuncional. Este perfil, además de todas las tareas explicadas, al ser parte del equipo, debe cumplir sus funciones dentro de él [33].

  Dentro de este proyecto es difícil establecer un figura de Scrum Master ya que, según su versión teórica, ninguno de los perfiles encaja con este rol. Es por esto que el propio alumno del Trabajo de Fin de Grado será el que ocupe este perfil ya que cumplirá la función de supervisar y será el encargado del cumplimiento de la metodología; impidiendo que el equipo de desarrollo, en este caso el mismo, se vea interferido por obstáculos externos al desarrollo del proyecto.

- El **Product Owner** es el responsable de maximizar el valor del producto. Entre sus principales funciones para poder hacerlo se encuentra la gestión del Product Backlog que comprende la definición, priorización y comprensión de las historias de usuario. De esta manera, se maximiza el valor del trabajo realizado por el equipo de Scrum. Esta tarea debe realizarse no solo al principio del proyecto si no tras cada iteración, ya que, las prioridades o estimaciones sobre las historias de usuario han podido cambiar según las necesidades del cliente. Además, este rol tiene la responsabilidad de valorar si el trabajo entregado en cada incremento es aceptado por el cliente. Todas estas tareas requieren que el Product Owner sea un experto en la materia, puesto que, debe conocer tanto las necesidades del negocio como la manera de optimizar el proceso para que el producto incremental sea útil para el cliente. Por estos motivos el Product Owner es la interfaz entre el cliente y el equipo de Scrum [33].

  En el caso de este proyecto, y dado que se ajustan perfectamente al perfil, el Product Owner será asignado a José Joaquim de Moura Ramos sabiendo que, el perfil de cliente está asociado al resto de directores (Jorge Novo Buján y Marcos Ortega Hortas) ya que el Product Owner es el encargado de que se lleven a cabo las tareas requeridas por el

cliente.

Artefactos

- El **Product Backlog** es una lista que representa la única fuente de requisitos y que debe estar priorizada y actualizada. Como ya se ha comentado anteriormente, el responsable de que esto suceda es el Product Owner. Además, cada entrada debe proporcionar necesariamente valor al cliente. Esta lista cambia constantemente debido a las necesidades del mercado, a los cambios tecnológicos o gracias a la retroalimentación que proporciona Scrum con cada iteración, puesto que tanto la lista como el producto en sí evolucionan. El Product Backlog debe recoger todas las características, funcionalidades y correcciones necesarias para la elaboración del producto, ya que, es sobre la que se centrará la estimación y el refinamiento de las diferentes tareas que constituyen el proyecto. El refinamiento se entiende que es el acto de detallar, clarificar y estimar cada elemento del Product Backlog por parte del equipo de Scrum [33].

  En el caso de este proyecto el Product Backlog se ha elaborado prácticamente en su totalidad al inicio, obligados en parte, por la creación del anteproyecto. Si bien es cierto, con el tiempo se fueron detallando a medida que el desarrollo progresaba.

- El **Sprint Backlog** es la lista compuesta por los elementos extraídos del Product Backlog que serán desarrollados en ese mismo Sprint [33]. Además de estas entradas, el Sprint Backlog contendrá el plan a seguir para entregar el incremento del producto. Estos elementos deben constituir una o varias tareas que serán estimadas en puntos de historia. Si una tarea es muy grande debe ser descompuesta en dos o más tareas, ya que, puede aumentar su complejidad a la hora de detectar problemas.

Eventos

- El **Sprint** es la base sobre la que se sustenta Scrum. Representa un bloque de tiempo, *time-box*, de cuatro semanas o menos durante el que se desarrolla un incremento del producto, aportando valor y siendo útil al cliente ya que este puede ser desplegado al finalizar dicho Sprint. La duración de cada Sprint puede no ser fija durante todo el proyecto, aunque esto no sea recomendable, pero sí es cierto que se debe establecer antes del comienzo del Sprint y no puede cambiar durante el mismo. Los Sprints se distribuyen de manera continua, es decir, al finalizar uno da comienzo otro. No está compuesto únicamente por el desarrollo del incremento, también forman parte de este evento la Sprint Planning Meeting, que dan comienzo al Sprint con la organización del mismo, las Daily Scrums, que se celebran diariamente con el fin de actualizar el progreso

de dicho Sprint, el Sprint Review, que se celebra al final del Sprint y por último, el Sprint Retrospective, situado al igual que el anterior, al final del Sprint [33].

Una vez estudiado el caso de este proyecto, se ha decidido que la duración de los Sprints serían de 15 días.

- La reunión de **Sprint Planning**, como se ha mencionado anteriormente, es una reunión que se celebra al principio del Sprint y da comienzo al mismo. En él, se analiza el Product Backlog y se determinan qué historias de usuario pasan al Sprint Backlog determinando así el alcance del Sprint. Las entradas candidatas que se van a llevar a cabo en dicho Sprint deben ser refinadas para producir tareas específicas y que así puedan ser estimadas. Durante esta reunión participa el Product Owner, que será el encargado de presentar el Product Backlog al equipo de desarrollo incluyendo el Scrum Master. Esta reunión no debe superar las 8 horas [33].

- La **Daily Meeting** es una reunión diaria cuya principal función es mantener actualizado a todo el equipo de desarrollo del avance del Sprint. En ella, cada miembro informa al resto del equipo sobre su progreso en las tareas asignadas. Durante este informe de situación se responde normalmente a tres preguntas *¿Qué hice ayer?*, *¿Qué voy a hacer hoy?* y *¿Tengo algún impedimento que me está retrasando?*. La duración estándar de esta reunión es de unos 15 minutos [33].

- La reunión de **Sprint Review** se celebra al final del Sprint. Su funcionalidad es la de presentar el producto conseguido en la iteración y recibir retroalimentación. Participan tanto el Product Owner como el equipo de desarrollo y, además, el cliente, al que se le hará una demostración de los objetivos conseguidos durante el Sprint. La duración de esta reunión no debe sobrepasar las 4 horas [33].

- El **Sprint Retrospective** al igual que el Sprint Review, se celebra al final del Sprint. En esta reunión se inspeccionará cómo fue el Sprint en cuanto a personas, relaciones, procesos y herramientas. Además, se identificarán qué elementos salieron bien y en cuáles es posible una mejora. Todo lo conseguido en esta reunión se pondrá en marcha para el siguiente incremento, haciendo que, Sprint a Sprint, el equipo mejore. La duración recomendada es de un máximo de 3 horas para esta reunión [33]. En el caso de este proyecto se han realizado con la participación del alumno y los directores.

A modo de resumen, en la Figura 5.1.3 ilustra un diagrama del funcionamiento de la metodología ágil SCRUM y sus principales componentes.

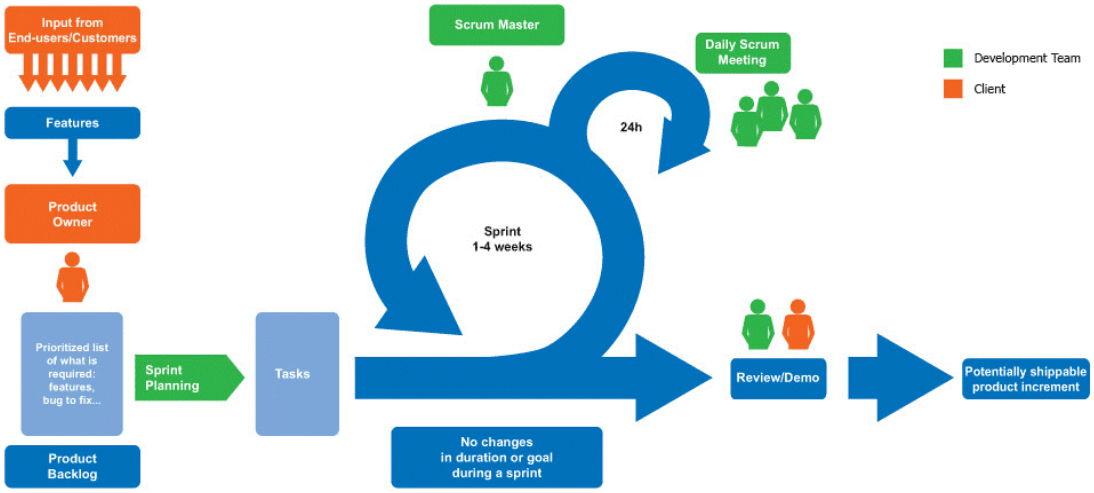


Figura 5.1: Diagrama del funcionamiento de SCRUM.

5.1.3 Gestión de proyecto

Product Backlog

A continuación se indicarán las entradas que han conformado el Product Backlog.

- Fase de análisis:
  1. Análisis del dominio.
  2. Análisis de metodologías y tecnologías.
  3. Análisis de las herramientas de gestión y desarrollo.
- Fase de instalación y configuración:
  1. Instalación y configuración de tecnologías para la gestión del proyecto.
  2. Instalación y configuración de *frameworks* y de tecnologías para el desarrollo del proyecto.
- Fase de requisitos e implementación:
  1. Modelos inteligentes:
     (a) Implementación de un modelo capaz de distinguir imágenes OCT y OCT-A.
     (b) Implementación de un modelo capaz de distinguir entre imágenes sanas, imágenes con CNV, imágenes con DME e imágenes con DRUSEN en imágenes oftalmológicas de tipo OCT.

(c) Implementación de un modelo capaz de distinguir entre imágenes sanas e imágenes con RVO en imágenes oftalmológicas de tipo OCT-A.

2. Aplicación web:
   (a) Registro y acceso a usuarios clínicos.
   (b) Gestión de pacientes con capacidad de dar de alta, listar, actualizar y borrar.
   (c) Gestión de informes con capacidad de crear, listar, actualizar y borrar.
   (d) Gestión de imágenes oftalmológicas con capacidad de subir, listar, procesar y borrar.
   (e) Generación de informes en PDF para descargar y enviar por correo electrónico al clínico responsable del paciente.
3. Despliegue:
   (a) Integración de la aplicación en Docker para el posterior despliegue.

5.1.4 Planificación

En la Figura 5.2 se puede ver la planificación que se ha realizado de las diferentes tareas del proyecto. En una primera fase se estudia el dominio tanto clínico como tecnológico, donde se abordan los conceptos técnicos del problema. Seguidamente se llevará a cabo la creación de los modelos computacionales y el estudio de sus resultados. A continuación se abordará el desarrollo la aplicación web. La creación de esta parte se dividirá en 5 Sprints donde se desarrollarán las distintas partes de la aplicación. Por último se cierra el proyecto con la recolección de los resultados y su análisis, plasmando todo este proceso en este documento.

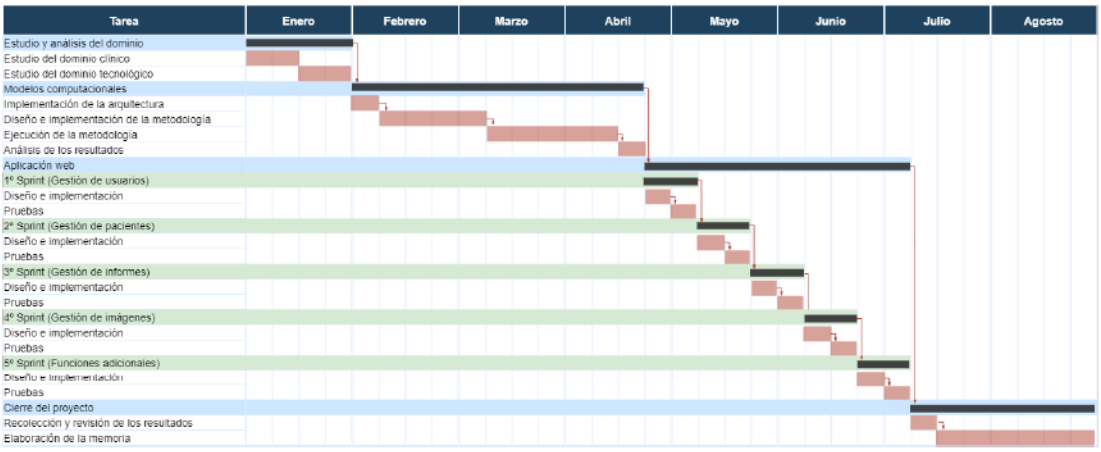


Figura 5.2: Diagrama de Gantt de la planificación prevista para el proyecto.

Destacar que la planificación inicial no ha coincidido con la ejecución del proyecto, pero con las correcciones tomadas durante su transcurso se ha logrado adaptar la ejecución a la

planificación de la Figura 5.2. La principal corrección tomada ha sido el incremento de las horas por día de trabajo; esto nos permite mantener la planificación en tiempo a expensas del aumento de los costes de recursos humanos del proyecto. Se puede ver la repercusión en el proyecto de esta medida en la Sección 5.1.6.

5.1.5 Recursos

En este apartado se detallan los recursos empleados en este proyecto, se diferencian entre recursos materiales y recursos humanos.

Recursos materiales

En cuanto a materiales, el único gasto a tener en cuenta es el coste del equipo donde se han ejecutado los entrenamientos y el desarrollo así como su mantenimiento. Como el equipo no se ha adquirido para el proyecto si no que ya se contaba con él y su mantenimiento es mínimo al ser un ordenador de sobremesa el coste será despreciado. En cuanto a recursos materiales empleados tenemos los recursos *software*, ya explicados en el apartado de tecnologías, y los recursos *hardware*. Para el desarrollo de este proyecto se ha empleado únicamente un ordenador de sobremesa. Las especificaciones del equipo son las siguientes:

- Ordenador de sobremesa
  - Procesador: Intel Core i5-6600K 3.50GHz
  - Memoria RAM: G.Skill Ripjaws V Red DDR4 2133 PC4-17000 4x4GB CL15
  - Almacenamiento: 1TB HDD y 250GB SSD
  - Tarjeta gráfica: Asus ROG GeForce GTX 1060 Strix Gaming OC 6GB GDDR5
  - Sistema operativo: Windows 10 Home

Recursos humanos

En cuanto a los recursos humanos para este proyecto es necesario que intervengan cuatro roles diferentes: Un jefe de proyecto que planifique, ejecute y monitorice las acciones que forman parte del proceso de elaboración del producto *software*. Un analista que analice el problema y lo describa con el propósito de solucionarlo mediante un proceso *software*. Un diseñador que defina la arquitectura, componentes, interfaces y otras características de un sistema *software*. Un programador que elabora y mantiene el código del producto *software* en su nivel más bajo.

5.1.6 Costes

En este apartado se comentarán los costes del proyecto, tanto materiales como humanos.

Costes estimados de *software*

Primeramente tenemos los costes de los materiales *software*, como se han empleado siempre tecnologías de código abierto en esta parte no tendremos coste alguno.

Costes estimados de *hardware*

Por otro lado en cuanto a los costes de los materiales *hardware* el único gasto real es el coste del equipo donde se han ejecutado los entrenamientos y el desarrollo así como su mantenimiento. Como el equipo no se ha adquirido para el proyecto, si no que ya se contaba con él y, además, su mantenimiento es mínimo al ser un ordenador de sobremesa, el coste será despreciado.

Costes estimados de recursos humanos

Se estima que el tiempo destinado por el alumno a este proyecto es de 6 horas al día con 5 días a la semana durante 32 semanas (8 meses). Por lo que tenemos una estimación final de 960 horas $\times$ persona. En el caso de los directores de proyecto se estima que le dedicarán 4 horas a la semana, siendo esta estimación una media de todas las semanas, entre contextualización del dominio, consultas y reuniones. Por tanto la estimación total es de 128 horas $\times$ persona. Una vez estimadas las horas de todos los perfiles que intervienen en el proyecto el coste se calcula en base al salario medio del país y a costes tanto laborales (Seguridad Social, bonificaciones...) como de empresa (local, electricidad...). Suponiendo que el salario de un Ingeniero Informático sin experiencia es de 20.000€ anuales limpios y le asociamos unos gastos a mayores de 18.400€ el salario es de 20€ $\times$ hora [34]. Para el caso de los directores del proyecto el salario medio supuesto es de 30.000€ anuales limpios y se le asocian unos gastos a mayores de 27.600€, siendo por tanto 30€ $\times$ hora [35].

Rol	**Coste €/h**	**Horas $\times$ persona**	**Coste total €/h**
Alumno	20	960	19.200
Director de proyecto 1	30	128	3.840
Director de proyecto 2	30	128	3.840
Director de proyecto 3	30	128	3.840
		Total €:	30.720

Tabla 5.1: Estimación de los costes del proyecto de los recursos humanos.

Como se puede ver en la Tabla 5.1, el coste estimado del proyecto asciende a 30.720 €. Debido al desconocimiento del dominio por parte del alumno y a problemas encontrados du-

rante el desarrollo del proyecto tanto el tiempo como el coste ha sido subestimado. Durante los últimos 3 meses el tiempo dedicado por parte del alumno ha sido incrementado en 1 hora llegando este a 1.020 horas $\times$ persona y un incremento del coste en 1.200€, alcanzando el coste del proyecto a 31.920 €.

Capítulo 6

Metodologías computacionales para el screening patológico

En este capítulo se describen las metodologías computacionales para el screening patológico, las aproximaciones que las componen y el *pipeline* que unifica todas estas metodologías. El objetivo de estas metodologías es el de proporcionar modelos útiles para el apoyo al diagnóstico de patologías en imágenes oftalmológicas (OCT/OCT-A). Además, se detalla la arquitectura de red empleada y sus principales características.

6.1 Metodologías computacionales

La Figura 6.1 muestra un representación esquemática de las diferentes metodologías computacionales que fueron desarrolladas para el screening patológico. En concreto, podemos ver cómo las imágenes al entrar en el sistema se clasifican según el tipo de dispositivo (OCT y OCT-A) y una vez realizado este proceso, se clasifican según las patologías diferenciadas en cada tipo de dispositivo. Este *pipeline* nos deja 3 metodologías diferenciadas. Una primera metodología de *clasificación automática de la imagen según el tipo de dispositivo*, una segunda metodología de *screening patológico en OCT* y una tercera metodología de *screening patológico en OCT-A*. A continuación se describen las metodologías mencionadas y los enfoques que las componen.

6.1.1 Clasificación automática de la imagen según el tipo de dispositivo: OCT vs OCT-A

Este módulo inteligente distingue automáticamente entre imágenes de tipo OCT y OCT-A. A la hora de diagnosticar patologías en imágenes oftalmológicas, es útil diferenciar el tipo de imagen con el que estamos trabajando para un diagnóstico preciso y eficaz. Con esto en mente, hemos diseñado un módulo que clasificará en la clase *OCT* aquellas imágenes de tomografía de

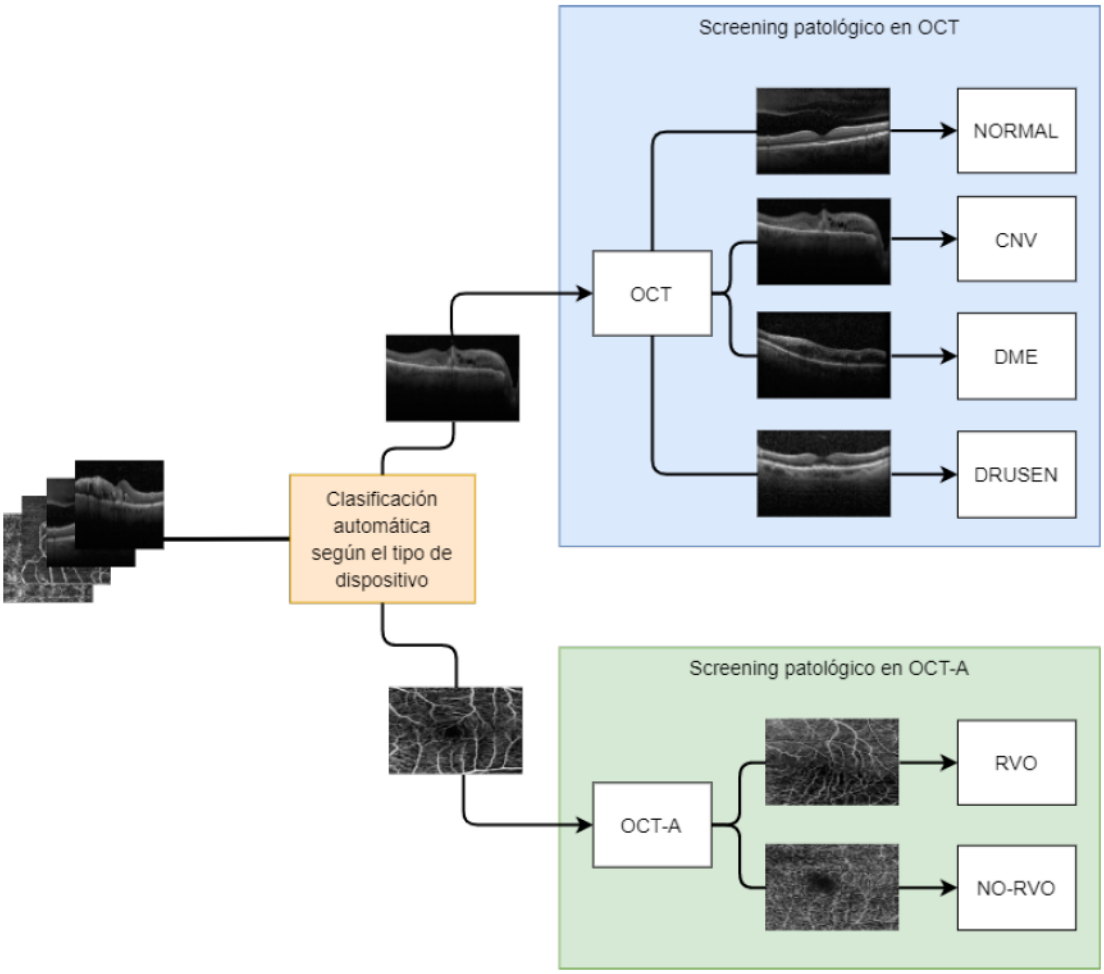


Figura 6.1: Estructura de la metodología computacional para el screening patológico.

coherencia óptica y en la clase *OCT-A* aquellas imágenes que correspondan con angiografías por tomografía de coherencia óptica.

6.1.2 Screening patológico en OCT: NORMAL vs CNV vs DME vs DRUSEN

Este módulo inteligente es capaz de diferenciar a los pacientes sanos de los pacientes con alguna de las patologías presentes en las imágenes OCT. De este modo, las imágenes se clasificarán en *CNV* para pacientes con neovascularización coroidea, *DME* para pacientes con edema macular diabético, *DRUSEN* para pacientes con drusas entre las capas de la retina y *NORMAL* para pacientes sin patología.

6.1.3 Screening patológico en OCT-A: RVO vs NO-RVO

En este módulo, el sistema es capaz de clasificar automáticamente las imágenes de tipo OCT-A en patológicas o sanas. Concretamente, la patología a la que se hace referencia es la oclusión de la vena retiniana (RVO). En la clase *RVO* se clasificarán las imágenes que presenten signos de oclusión de la vena retiniana, mientras que en la clase *NO-RVO* se clasificarán las imágenes de pacientes que no presenten ningún tipo de patología ocular.

6.2 Arquitectura de red

6.2.1 DenseNet-161

En este proyecto se utiliza una arquitectura DenseNet-161 [36], una especificación de la original con profundidad 161 y un ratio de crecimiento de 48. Además, cabe destacar que en la última capa (*Classification Layer*) el número de salidas está adaptada al tipo de problema, siendo esta de dos salidas para el problema de clasificación de OCT vs OCT-A, de dos también para el problema de clasificación de RVO vs NO-RVO y de cuatro para el problema de clasificación de CNV vs DME vs DRUSAS vs NORMAL. Por último, resaltamos que el uso de esta arquitectura de red está motivado por su robustez, sencillez y los resultados satisfactorios obtenidos en problemas similares. La estructura completa de la arquitectura DenseNet-161 se muestra en la Figura 6.1.

Layers	Output Size	DenseNet-161
Convolution	112 × 112	conv. 7 × 7, stride 2
Pooling	56 × 56	max pool 3 × 3, stride 2
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 conv \\ 3 \times 3 conv \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	conv. 1 × 1
	28 × 28	2 × 2 average pool, stride 2
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 conv \\ 3 \times 3 conv \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	conv. 1 × 1
	14 × 14	2 × 2 average pool, stride 2
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 conv \\ 3 \times 3 conv \end{bmatrix} \times 36$
Transition Layer (3)	14 × 14	conv. 1 × 1
	7 × 7	2 × 2 average pool, stride 2
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 conv \\ 3 \times 3 conv \end{bmatrix} \times 24$
Classification Layer	1 × 1	7 × 7 global average pool
		2 o 4 fully-connected, softmax

Tabla 6.1: Estructura de la arquitectura DenseNet-161 [1].

6.2.2 Detalles del entrenamiento

Para el proceso de entrenamiento se han divido los *datasets* en subconjuntos mutuamente excluyentes de entrenamiento (60%), validación (20%) y prueba (20%). En cuanto a la función de *loss*, se emplea la de entropía cruzada o *Cross Entropy Loss*. Para problemas no binarios, la entropía cruzada corresponde a la Ecuación 6.1.

$$Loss = -\sum_{b=1}^{N} y_{a,b} \log(p_{a,b}) \tag{6.1}$$

Donde N es el número de clases, y el indicador binario (0 o 1) si la etiqueta de clase b es la clasificación correcta para la observación a y p. Por otro lado, en lo que respecta a la optimización del modelo, durante el entrenamiento se emplea *Stochastic Gradient Descent* (SGD) con una constante de ratio de aprendizaje de 0,01, un tamaño de *mini-batch size* de 17 y un *momentum* de primera orden de 0,9. Este método es muy efectivo para los problemas de optimización de alta dimensión, reduciendo la carga computacional y logrando así iteraciones

más rápidas, pero logrando una menor tasa de convergencia. Todos los valores mencionados anteriormente han sido obtenidos y seleccionados mediante experimentación exhaustiva.

Por otro lado, denominaremos *epoch* a un ciclo completo de entrenamiento, entendiendo esto como una pasada por todos los *frames* del conjunto de entrenamiento. El número de *epochs* es variable según el modelo a entrenar. Los valores que han tomado han sido de 50 y 200. Esto se debe a que en ciertos entrenamientos, ya sea porque existe una cantidad suficiente de imágenes o porque el aprendizaje es más rápido (debido a la simpleza de la clasificación) no es necesario que superen los 50 *epochs*. En otros casos ha sido necesario llegar a las 200 *epochs* para alcanzar un resultado válido y estable, siendo un número mayor de ciclos innecesario puesto que no podemos apreciar una mejora significativa ni en la precisión ni en la función de *loss*.

Por último, para conseguir resultados consistentes, el entrenamiento se ha repetido 5 veces de forma independiente y con muestras aleatorias, permitiendo extraer una generalización a partir de las medias de los resultados obtenidos y así evaluar el funcionamiento global de todas las aproximaciones computacionales propuestas. Para todos los enfoques, se analizarán los resultados obtenidos con un modelo *"scratch"* (sin preentrenamiento) y con un modelo preentrenado (con ImageNet), con el fin de analizar el comportamiento de los modelos en ambos casos.

Capítulo 7

Resultados y análisis

A continuación se comentan los resultados obtenidos en la experimentación para cada una de las metodologías computacionales. A modo de resumen:

- En la primera metodología tenemos dos aproximaciones. La primera se basa en la creación de un modelo sin la utilización de pre-entrenamiento y con la finalidad de que pueda clasificar imágenes OCT-A y imágenes OCT. La segunda aproximación es *idem* que la anterior a excepción de que esta segunda sí cuenta con pre-entrenamiento.
- En la segunda metodología tenemos otras dos aproximaciones. La primera se basa en la creación de un modelo sin la utilización de pre-entrenamiento y con la finalidad de que pueda clasificar imágenes OCT sanas, con CNV, con DME o con DRUSEN. La segunda aproximación es *idem* que la anterior a excepción de que esta segunda sí cuenta con pre-entrenamiento.
- En la tercera metodología también tenemos dos aproximaciones. La primera se basa en la creación de un modelo sin la utilización de pre-entrenamiento y con la finalidad de que pueda clasificar imágenes OCT-A sanas o con RVO. La segunda y última aproximación es *idem* que la anterior a excepción de que esta última sí cuenta con pre-entrenamiento.

7.1 Métricas

Para analizar los resultados se usarán las siguientes métricas: Precisión (*Accuracy*), Exactitud (*Precision*), Sensibilidad (*Recall*) y *F1-score*. En particular, para calcular las métricas se emplearán los valores obtenidos a partir de la matriz de confusión (véase Figura 7.1), que nos permite la visualización del desempeño de un algoritmo, y que son las siguientes: *True Positive (TP)*, *False Positive (FP)*, *False Negative (FN)* y *True Negative (TN)*.

		Valores predichos	
		Positivo	Negativo
Valores reales	Positivo	TP	FN
	Negativo	FP	TN

Figura 7.1: Matriz de confusión.

- **Precisión (*Accuracy*).** Índice que representa el grado de acierto. Número de aciertos entre número de intentos. Su formulación matemática corresponde con la Ecuación 7.1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.1}$$

- **Exactitud (*Precision*).** Índice que representa el número de positivos acertados sobre todos los positivos. Su formulación matemática corresponde con la Ecuación 7.2.

$$Precision = \frac{TP}{TP + FP} \tag{7.2}$$

- **Sensibilidad (*Recall*).** Índice que representa el número de positivos acertados sobre todos los no positivos. Su formulación matemática corresponde con la Ecuación 7.3.

$$Recall = \frac{TP}{TP + FN} \tag{7.3}$$

- ***F1-score.*** Índice que representa la media armónica de la exactitud y la sensibilidad. Es útil para identificar desbalanceo en la clasificación. Su valor máximo es 1, indicando que tanto la exactitud como la sensibilidad son perfectos y el mínimo es 0 cuando al menos una de las medidas es 0. Su formulación matemática corresponde con la Ecuación 7.4.

$$F1 - score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{7.4}$$

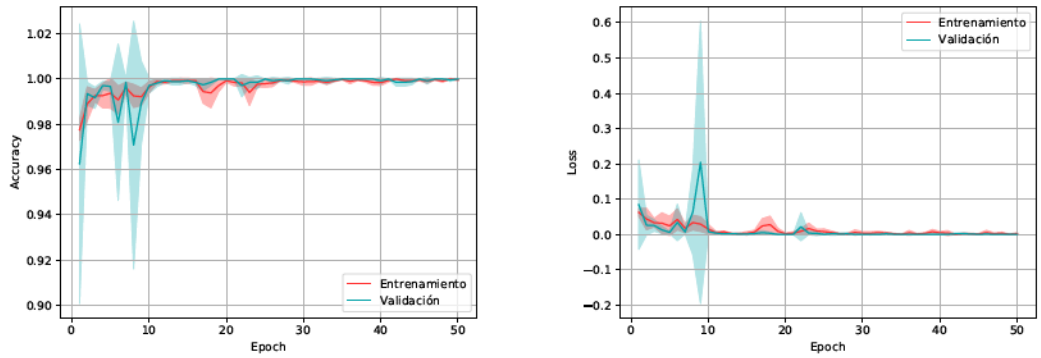
7.2 Resultados

En este apartado detallaremos los resultados de cada experimento así como su análisis. Para cada aproximación tendremos una gráfica de la evolución del *Accuracy* tanto en la etapa de entrenamiento como en la de validación. De forma complementaria, se presentará otra gráfica con la evolución del *Loss*, una vez más, en las etapas de entrenamiento y validación. Por último, cada aproximación tendrá una tabla resumen con los resultados obtenidos utilizando el conjunto de datos de prueba.

7.2.1 Resultados y análisis de los modelos de clasificación según el tipo de dispositivo: OCT vs OCT-A

Aproximación sin pre-entrenamiento

A continuación, podemos ver el resumen del proceso de entrenamiento de cinco modelos sin pre-entrenamiento para esta aproximación. La Figura 7.2a nos muestra la progresión del *Accuracy* y la Figura 7.2b la progresión del *Loss*, tanto para el entrenamiento como para la validación de los modelos. Como podemos ver, el entrenamiento ha concluido cerca de los valores óptimos, aunque los primeros valores de variación son muy altos, se estabilizan rápidamente con valores cercanos al 1 y al 0, tanto para el *Accuracy* como para el *Loss*, respectivamente.



(a) (b)

Figura 7.2: Progresión de la aproximación sin pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT y OCT-A. (a) Representación de la evolución del *Accuracy* en el entrenamiento y en la validación. (b) Representación de la evolución del *Loss* en el entrenamiento y en la validación.

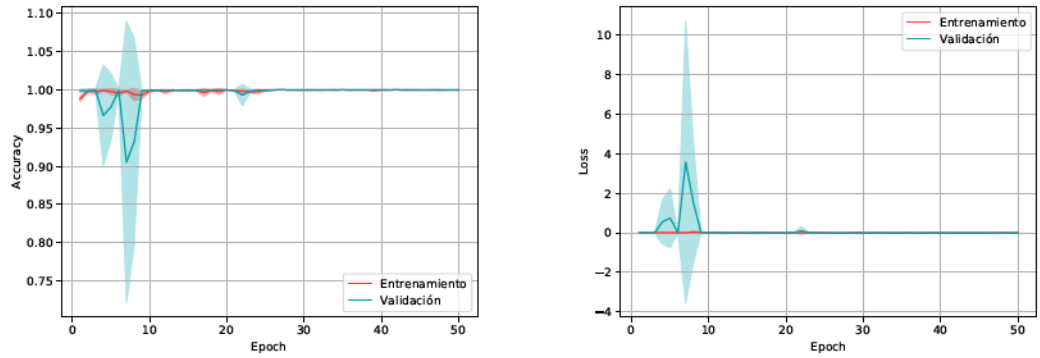
A continuación, presentamos los resultados obtenidos utilizando el conjunto de datos de test (ver Tabla 7.1), donde podemos apreciar la media y la desviación típica de las métricas (*Recall*, *Precision* y *F1-score*) para cada clase. Los valores presentados corresponden a la evaluación de la etapa de test con 5 modelos sin pre-entrenamiento (*scratch*). Como podemos ver, los resultados obtenidos son satisfactorios para ambas clases, alcanzando un *Accuracy* de $0{,}9986 \pm 0{,}0013$.

Clase	*Recall*	*Precision*	*F1-score*
OCT	$0{,}9993 \pm 0{,}0015$	$0{,}9978 \pm 0{,}0029$	$0{,}9985 \pm 0{,}0014$
OCT-A	$0{,}9979 \pm 0{,}0028$	$0{,}9993 \pm 0{,}0014$	$0{,}9986 \pm 0{,}0013$

Tabla 7.1: *Recall*, *Precision* y *F1-score* por clase de la aproximación sin pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT y OCT-A.

Aproximación con pre-entrenamiento

A continuación, podemos observar dos representaciones gráficas con el resumen del proceso de entrenamiento y validación de cinco modelos a partir de una red preentrenada con ImageNet. En particular, la Figura 7.3a nos muestra la progresión del *Accuracy* y la Figura 7.3b la progresión del *Loss*, tanto para el entrenamiento como para la validación de los modelos. Como podemos ver, el entrenamiento ha concluido cerca de los valores óptimos, aunque los primeros valores de variación son muy altos, se estabilizan rápidamente con valores cercanos al 1 y al 0, tanto para el *Accuracy* como para el *Loss*, respectivamente. Por último, observamos que el crecimiento es más acelerado al principio de la curva y se estabiliza a partir del *epoch* 10, lo que demuestra la utilidad del pre-entrenamiento en esta aproximación.



(a) (b)

Figura 7.3: Progresión de la aproximación con pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT y OCT-A. (a) Representación de la evolución del *Accuracy* en el entrenamiento y en la validación. (b) Representación de la evolución del *Loss* en el entrenamiento y en la validación.

A continuación, presentamos en la Tabla 7.2 los resultados obtenidos utilizando el conjunto de datos de test, donde podemos apreciar tanto la media como la desviación típica de las métricas (*Recall*, *Precision* y *F1-score*) para cada clase. Como podemos ver, los resultados obtenidos son muy satisfactorios en todas las métricas, alcanzando un *Accuracy* de 1,0000 $\pm$ 0,0000. De este modo, podemos concluir que el modelo ha sido capaz de diferenciar con éxito las imágenes entre las 2 categorías (OCT/OCT-A). También hay que destacar que la mejora de los resultados en comparación con los modelos sin pre-entrenamiento de este mismo enfoque se debe a la capacidad de la red para adquirir y transferir conocimiento (en este caso concreto, a partir del conjunto de datos ImageNet). Esto permite a la red utilizar el conocimiento de ImageNet y, en consecuencia, hacer que la curva de aprendizaje parta de un de valor alto (más cercano a 1) y se estabilice con menos *epochs*, mejorando la eficiencia del proceso.

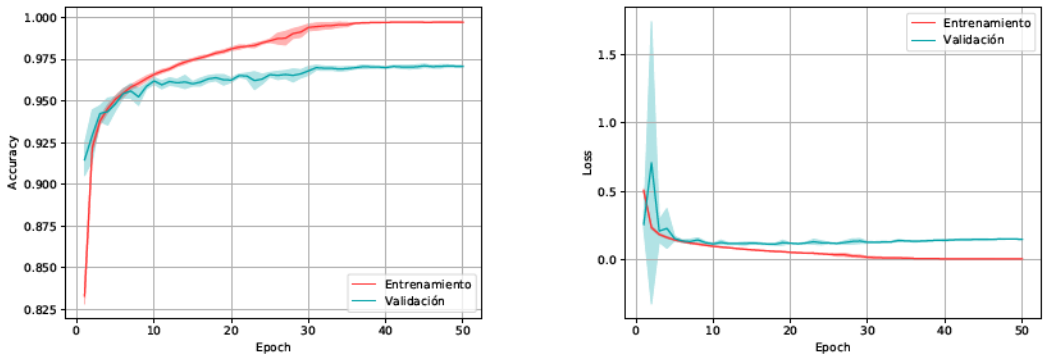
Clase	*Recall*	*Precision*	*F1-score*
OCT	1,0000 ± 0,0000	1,0000 ± 0,0000	1,0000 ± 0,0000
OCT-A	1,0000 ± 0,0000	1,0000 ± 0,0000	1,0000 ± 0,0000

Tabla 7.2: *Recall*, *Precision* y *F1-score* por clase de la aproximación con pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT y OCT-A.

7.2.2 Resultados y análisis de los modelos de screening patológico en OCT: NORMAL vs CNV vs DME vs DRUSEN

Aproximación sin pre-entrenamiento

A continuación, podemos ver el resumen del proceso de entrenamiento de cinco modelos sin pre-entrenamiento para esta aproximación. La Figura 7.4a nos muestra la progresión del *Accuracy* y la Figura 7.4b la progresión del *Loss*, tanto para el entrenamiento como para la validación de los modelos. Como podemos ver, el entrenamiento ha concluido tras estabilizarse en valores óptimos para el *Accuracy*, cercanos al 1 para el entrenamiento y a 0.975 para la validación. En cuanto al *Loss* podemos ver una variación en los *epochs* iniciales pero que, a medida que el *Accuracy* se estabiliza, el *Loss* disminuye considerablemente.



(a) (b)

Figura 7.4: Progresión de la aproximación sin pre-entrenamiento del screening patológico para la diferenciación de imágenes NORMALES, CNV, DME o DRUSEN. (a) Representación de la evolución del *Accuracy* en el entrenamiento y en la validación. (b) Representación de la evolución del *Loss* en el entrenamiento y en la validación.

A continuación, tenemos la Tabla 7.3 con las métricas de *Recall*, *Precision* y *F1-score* para cada clase. Los valores presentados corresponden a la media y la desviación típica de 5 modelos sin entrenamiento previo (scratch) para esta aproximación. Como podemos ver, los resultados obtenidos son generalmente satisfactorios para todas las clases, alcanzando un *Accuracy* de

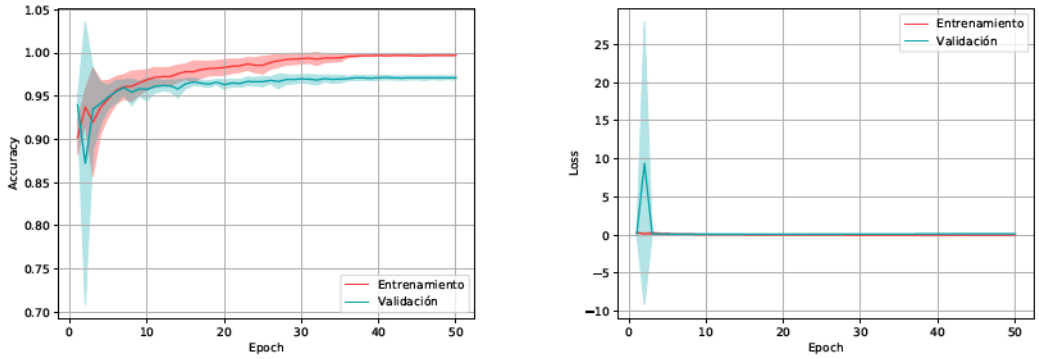
0,9701 $\pm$ 0,0008. En particular, los mejores resultados son obtenidos para la clase NORMAL, que corresponde a imágenes OCT sin patología ocular. Estos resultados son muy cercanos a los obtenidos para la clase CNV. En contrapartida, los peores resultados se obtienen para la clase DRUSEN, aproximadamente 5% menos que la clase DME.

Clase	*Recall*	*Precision*	*F1-score*
NORMAL	0,9831 $\pm$ 0,0019	0,9784 $\pm$ 0,0014	0,9808 $\pm$ 0,0014
CNV	0,9800 $\pm$ 0,0012	0,9800 $\pm$ 0,0014	0,9800 $\pm$ 0,0004
DME	0,9547 $\pm$ 0,0041	0,9654 $\pm$ 0,0007	0,9600 $\pm$ 0,0019
DRUSEN	0,9079 $\pm$ 0,0058	0,9078 $\pm$ 0,0056	0,9079 $\pm$ 0,0030

Tabla 7.3: *Recall*, *Precision* y *F1-score* por clase para la aproximación sin pre-entrenamiento del screening patológico para la diferenciación de imágenes NORMALES, CNV, DME o DRUSEN.

Aproximación con pre-entrenamiento

A continuación podemos ver el resumen del proceso de entrenamiento de cinco modelos con pre-entrenamiento para esta aproximación. La Figura 7.5a nos muestra la progresión del *Accuracy* y la Figura 7.5b la progresión del *Loss*, tanto para el entrenamiento como para la validación de los modelos. Una vez más, como podemos ver, el entrenamiento ha concluido después de estabilizarse en valores óptimos para el *Accuracy*, cercanos al 1 para el entrenamiento y a 0.975 para la validación. En cuanto al *Loss* podemos ver una variación inicial pero que, rápidamente se estabiliza. Los resultados obtenidos corresponden con utilizar un modelo pre-entrenado, ya que el crecimiento es más acelerado al principio de la curva y además se estabiliza antes.



(a) (b)

Figura 7.5: Progresión de la aproximación con pre-entrenamiento del screening patológico para la diferenciación de imágenes NORMALES, CNV, DME o DRUSEN. (a) Representación de la evolución del *Accuracy* en el entrenamiento y en la validación. (b) Representación de la evolución del *Loss* en el entrenamiento y en la validación.

A continuación se presentan los resultados obtenidos con el conjunto de datos de test. En la Tabla 7.4, podemos apreciar la media y la desviación típica de *Recall*, *Precision* y *F1-score* para cada clase, que corresponde a los valores obtenidos con un modelo pre-entrenado con ImageNet para esta aproximación.

Como podemos ver en la tabla y después de analizar las gráficas, los resultados obtenidos son satisfactorios para las 4 clases, alcanzando un *Accuracy* de 0,9718 $\pm$ 0,0011. Al igual que la aproximación sin pre-entrenamiento los mejores resultados son obtenidos para la clase NORMAL, seguida de la CNV. Los peores resultados se obtienen para la clase DRUSEN, aproximadamente 5% menos que la clase DME. Además, también observamos una ligera mejora en el rendimiento global del sistema gracias a la transferencia de conocimiento derivado del conjunto de datos ImageNet.

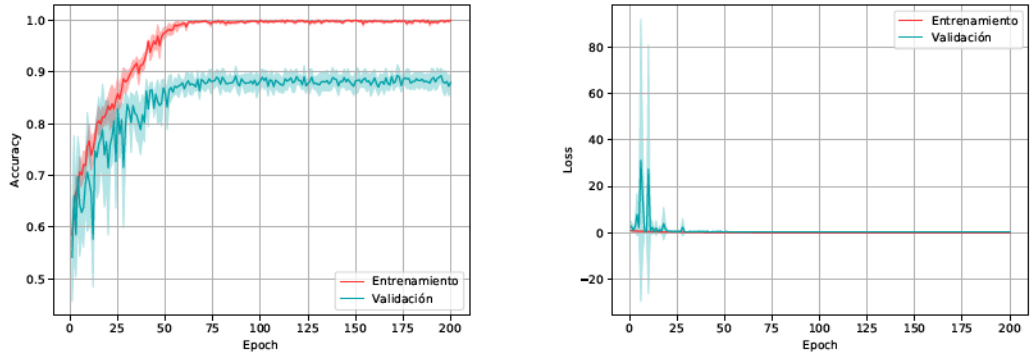
Clase	*Recall*	*Precision*	*F1-score*
NORMAL	0,9838 $\pm$ 0,0013	0,9814 $\pm$ 0,0037	0,9826 $\pm$ 0,0018
CNV	0,9807 $\pm$ 0,0020	0,9808 $\pm$ 0,0024	0,9808 $\pm$ 0,0019
DME	0,9598 $\pm$ 0,0059	0,9680 $\pm$ 0,0029	0,9639 $\pm$ 0,0034
DRUSEN	0,9122 $\pm$ 0,0059	0,9085 $\pm$ 0,0069	0,9103 $\pm$ 0,0041

Tabla 7.4: *Recall*, *Precision* y *F1-score* por clase de la aproximación con pre-entrenamiento del screening patológico para la diferenciación de imágenes NORMALES, CNV, DME o DRUSEN.

7.2.3 Resultados y análisis de los modelos screening patológico en OCT-A: RVO vs NO-RVO

Aproximación sin pre-entrenamiento

A continuación, presentamos la evolución del proceso de entrenamiento y validación de cinco modelos sin pre-entrenamiento (scratch) para esta aproximación. En particular, la Figura 7.6a nos muestra la progresión del *Accuracy* y la Figura 7.6b la progresión del *Loss*. Como podemos ver, los valores de *Accuracy* se han estabilizado a partir del *epoch* 75. En cuanto al *Loss* podemos ver una pequeña variación en los *epochs* iniciales pero que, a medida que se estabiliza el *Accuracy*, el *Loss* disminuye considerablemente, alcanzando valores cercanos a 0.



(a) (b)

Figura 7.6: Progresión de la aproximación sin pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT-A con RVO. (a) RRepresentación de la evolución del *Accuracy* en el entrenamiento y en la validación. (b) Representación de la evolución del *Loss* en el entrenamiento y en la validación.

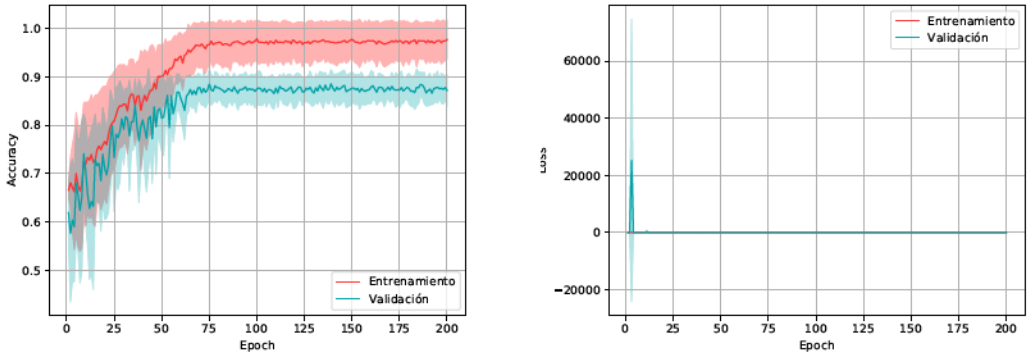
A continuación, presentamos la Tabla 7.5 con las métricas de *Recall*, *Precision* y *F1-score* para cada clase, sin pre-entrenamiento. Como podemos ver, los resultados obtenidos son satisfactorios para ambas clases (RVO/NO-RVO), alcanzando un *Accuracy* de 0,8637 $\pm$ 0,0111. En particular, también observamos un mejor rendimiento del sistema en la identificación de la clase patológica, con valores de F1-score superiores a 87%.

Clase	*Recall*	*Precision*	*F1-score*
RVO	0,8623 $\pm$ 0,0171	0,8890 $\pm$ 0,0096	0,8754 $\pm$ 0,0110
NO-RVO	0,8654 $\pm$ 0,0105	0,8340 $\pm$ 0,0180	0,8494 $\pm$ 0,0110

Tabla 7.5: *Recall*, *Precision* y *F1-score* por clase de la aproximación sin pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT-A con RVO.

Aproximación con pre-entrenamiento

A continuación, podemos ver el resumen del proceso de entrenamiento y validación de cinco modelos pre-entrenados con ImageNet para esta aproximación. La Figura 7.7a nos muestra la progresión del *Accuracy* y la Figura 7.7b la progresión del *Loss*. Una vez más, como podemos ver, los valores de *Accuracy* se han estabilizado a partir del *epoch* 75, tanto para el entrenamiento como para la validación. En cuanto al *Loss* podemos ver una variación en los *epochs* iniciales muy grande pero que, a medida que se estabiliza el *Accuracy*, el *Loss* disminuye considerablemente, alcanzando valores cercanos a 0.



(a) (b)

Figura 7.7: Progresión de la aproximación con pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT-A con RVO. (a) Representación de la evolución del *Accuracy* en el entrenamiento y en la validación. (b) Representación de la evolución del *Loss* en el entrenamiento y en la validación.

A continuación, presentamos la Tabla 7.6 con los valores obtenidos para cada clase y representados mediante las métricas de *Recall*, *Precision* y *F1-score*. En modo general, los resultados obtenidos por el sistema con el dataset de test son satisfactorios, dado que alcanza un valor de Accuracy igual a 0,8695 $\pm$ 0,0414. Al igual que el enfoque anterior, también observamos resultados superiores en la identificación de la clase (NO-RVO). Además, también observamos una pequeña mejora en el rendimiento general del sistema debido al uso de la estrategia de aprendizaje por transferencia a través del conjunto de datos ImageNet.

Clase	*Recall*	*Precision*	*F1-score*
RVO	0,8757 $\pm$ 0,0239	0,8890 $\pm$ 0,0519	0,8823 $\pm$ 0,0360
NO-RVO	0,8615 $\pm$ 0,0681	0,8455 $\pm$ 0,0323	0,8534 $\pm$ 0,0486

Tabla 7.6: *Recall*, *Precision* y *F1-score* por clase de la aproximación con pre-entrenamiento del screening patológico para la diferenciación de imágenes OCT-A con RVO.

Capítulo 8

Desarrollo software

En este capítulo se expondrá todo lo relativo al proceso de desarrollo *software* del proyecto. Se pasarán por las fases propias de este proceso, que son el análisis de requisitos, el diseño software, la implementación y las pruebas. También se explicará cual ha sido el proceso realizado para llevar a cabo estas tareas.

8.1 Análisis de requisitos

El análisis de requisitos es el proceso de estudio y refinamiento de las necesidades de una empresa, organización o usuarios para llegar a una definición de los requisitos *software*. Una buena identificación y análisis de los requisitos es esencial para llevar a cabo un proyecto *software* satisfactorio y conseguir un producto con las características deseadas. Por lo tanto, se detallarán a continuación los requisitos funcionales y no funcionales que se han identificado para este proyecto, a través del estudio del dominio y entrevistas con los directores del proyecto, cumpliendo ellos el rol de clientes.

8.1.1 Requisitos funcionales

Los requisitos funcionales son aquellos que describen el comportamiento del sistema. En otras palabras, los requisitos funcionales deben indicar qué debe hacer el sistema. A continuación, se detalla la lista de requisitos para este proyecto.

1. Gestión de clínicos.
    (a) *Los clínicos deben poder autenticarse en el sistema con sus credenciales.*
    (b) *Los clínicos deben poder cambiar sus datos de usuario (nombre, apellidos, contraseña...).*
2. Gestión de pacientes.

   (a) *Los clínicos deben poder dar de alta nuevos pacientes.*
   (b) *Los clínicos deben poder eliminar sus pacientes.*
   (c) *Los clínicos deben poder editar los datos de sus pacientes.*
   (d) *Los clínicos deben poder listar todos sus pacientes.*
   (e) *Los clínicos deben poder listar sus pacientes con filtros.*

3. Gestión de informes.
   (a) *Los clínicos deben poder crear nuevos informes para sus pacientes.*
   (b) *Los clínicos deben poder eliminar los informes de sus pacientes.*
   (c) *Los clínicos deben poder editar los informes de sus pacientes.*
   (d) *Los clínicos deben poder listar todos los informes de sus pacientes.*
   (e) *Los clínicos deben poder listar los informes de sus pacientes con filtros.*

4. Gestión de imágenes oftalmológicas.
   (a) *Los clínicos deben poder subir imágenes a los informes de sus pacientes.*
   (b) *Los clínicos deben poder eliminar las imágenes de los informes de sus pacientes.*
   (c) *Los clínicos deben poder procesar las imágenes de los informes de sus pacientes.*
   (d) *Los clínicos deben poder visualizar el resultado de procesar las imágenes de los informes de sus pacientes.*

5. Procesado de imágenes oftalmológicas.
   (a) *Los clínicos deben poder procesar las imágenes de los informes de sus pacientes.*

6. Representado y visualización del proceso de imágenes.
   (a) *Los clínicos deben poder visualizar el resultado de procesar las imágenes de los informes de sus pacientes.*

7. Generación automática de informes en PDF.
   (a) *Los clínicos deben poder descargar el informe de sus pacientes en formato PDF con todos los datos relativos a dicho informe e imágenes asociadas a él.*

8. Envío de informes por email.
   (a) *Los clínicos deben poder recibir en el email asociado a su cuenta cualquier informe de sus pacientes en formato PDF con todos los datos relativos a dicho informe e imágenes asociadas a él.*

8.1.2 Requisitos no funcionales

Los requisitos no funcionales son aquellos que determinan características generales y restricciones del sistema para que su uso y manejo sea el esperado. En otras palabras, los requisitos no funcionales deben indicar como lo debe hacer el sistema. A continuación se detalla la lista de los requerimientos no funcionales encontrados en este proyecto, así como una pequeña justificación de cada requisito escogido.

1. Privacidad: Seguridad en el tratamiento de datos sensibles.

2. Accesibilidad: Internacionalización de toda la aplicación web.

3. Escalabilidad: Implementación de forma modular para permitir escalabilidad del sistema.

4. Conformidad: Implementación de una interfaz eficaz y cómoda para un perfil clínico.

5. Robustez: No debe tener fallos de seguridad y debe ser resistente a posibles fallos.

6. Disponibilidad: La aplicación debe estar disponible el mayor tiempo posible para poder considerarse un sistema útil.

8.1.3 Casos de uso

Un *caso de uso* es una acción o conjunto de acciones que definen típicamente las interacciones entre un actor y un sistema para llevar a cabo una tarea con un objetivo. Un *actor* especifica el rol que juega un usuario u otro sistema que interactúa con el sujeto. En el apéndice A se enumerarán y detallarán los casos de uso del sistema. En la Figura A.1 se puede ver el actor clínico interaccionando con el sistema completo.

8.2 Diseño

8.2.1 Arquitectura del sistema

Una vez finalizada la etapa de análisis de requisitos, se abordará el diseño del proyecto. Primeramente se describirá la arquitectura escogida para esta aplicación. En la Figura 8.1 se presenta la arquitectura global de la aplicación y a continuación se explicará detalladamente todos sus componentes.

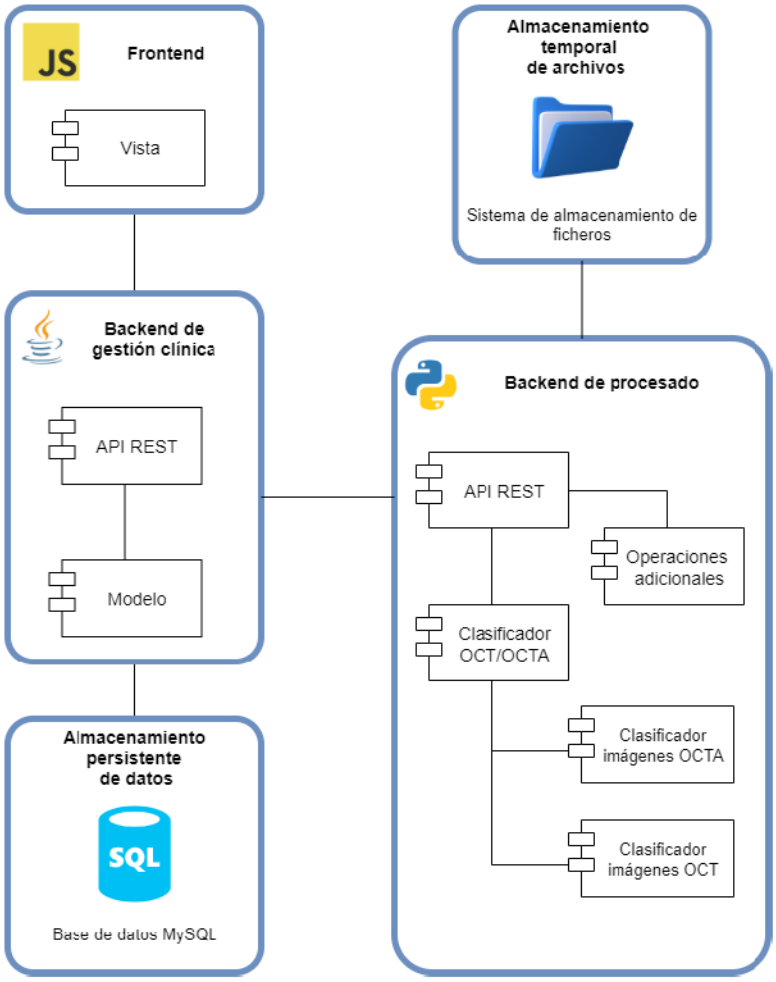


Figura 8.1: Arquitectura del sistema.

8.2.2 Frontend

Este componente se encargará de transformar los datos procedentes de las API en una interfaz limpia, clara y funcional para los clínicos.

Estructura de componentes

A continuación se describe la estructura de componentes que conforma este módulo.

- **src**
  - **backend**: Servicios de comunicación con el *backend*.
    - **patientService.js**: Servicio con las operaciones relacionadas con los pacientes.

        * **pictureService.js**: Servicio con las operaciones relacionadas con las imágenes.
        * **reportservice.js**: Servicio con las operaciones relacionadas con los informes.
        * **userService.js**: Servicio con las operaciones relacionadas con los usuarios clínicos.
    - **i18n**: Ficheros de internacionalización.
    - **img**: Carpeta con las imágenes utilizadas en la interfaz.
    - **modules**: Conjunto de módulos que forman la aplicación.
        * **app**: Conjunto de componentes principales: *header*, *footer* y *body*.
        * **common**: Conjunto de componentes comunes empleados en otros módulos.
        * **patients**: Conjunto de componentes relacionados con los pacientes.
        * **pictures**: Conjunto de componentes relacionados con las imágenes.
        * **reports**: Conjunto de componentes relacionados con los informes.
        * **users**: Conjunto de componentes relacionados con los usuarios clínicos.
    - **store**: Contiene el objeto que mantiene el árbol de estado de la aplicación.

8.2.3 Backend de gestión clínica

Por otro lado, este componente gestionará una API de operaciones referentes a la gestión de los clínicos, pacientes, informes e imágenes.

Estructura de componentes

A continuación se describe la estructura de componentes que conforma este módulo.

- **main**
    - **java/es/udc/tfg/backend**: Carpeta con la lógica de la aplicación.
        * **model**: Carpeta con todas las funcionalidades referentes al modelo.
            · **common**: Contiene los elementos comunes del modelo, como pueden ser las excepciones o ciertas estructuras.
            · **entities**: Contiene las entidades y *DAOs* de la aplicación.
            · **services**: Contiene la implementación de los servicios de la aplicación y sus interfaces.
        * **rest**: Carpeta con todas las funcionalidades referentes al controlador REST.
            · **common**: Contiene los elementos comunes del controlador, como pueden ser las excepciones.

            - **controllers**: Contiene los controladores de la aplicación.
            - **dtos**: Contiene los *DTOs* de la aplicación.
        * **Application.java**: Fichero principal de ejecución de la aplicación.
    - **resources**: Contiene los ficheros de configuración y de internacionalización.

- **sql**: Carpeta con los ficheros SQL.
    - **1-MySQLCreateTables.sql**: Fichero SQL para la creación de las tablas.
    - **2-MySQLCreateData**: Fichero SQL para crear datos iniciales.

- **test**
    - **java/es/udc/tfg/backend**: Carpeta con todos los ficheros de prueba.
    - **resources**: Contiene los ficheros de configuración de las pruebas.

8.2.4 Backend de procesado

Contamos además con otro *Backend*, este componente gestionará una API para el procesado de imágenes, es decir, tendrá cargados los modelos entrenados y los ejecutará con las imágenes que se le faciliten. Tiene a mayores operaciones adicionales como la generación de informes y su envío por email.

Estructura de componentes

A continuación se describe la estructura de componentes que conforma este módulo.

- **static**: Contiene todos los elementos estáticos, como pueden ser las imágenes del PDF.
- **models**: Carpeta con todos los modelos de procesado de imágenes.
    - **model_oct**: Modelo encargado de procesar las imágenes OCT.
    - **model_octa**: Modelo encargado de procesar las imágenes OCT-A.
    - **model_type**: Modelo encargado de clasificar imágenes según el dispositivo de captura, en este caso OCT o OCT-A.
- **api.py**: Contiene la *API* con las funciones y procedimientos que ofrece este *backend*.
- **mail_module.py**: Módulo encargado de enviar los emails.
- **models_module.py**: Módulo encargado de gestionar las llamadas y salidas de los modelos inteligentes.

- **pdf_module.py**: Módulo encargado de crear el informe en PDF.

- **plots_module.py**: Módulo encargado de crear las gráficas correspondientes para posteriormente añadirlas al PDF.

- **util_processing.py**: Módulo encargado de las operaciones de preprocesado y post procesado de las imágenes de los otros módulos. Este preprocesamiento consiste en la creación de una estructura de carpetas, del guardado de imágenes, de codificar y descodificar las imágenes de la llamada del *API* y del borrado, tanto de las imágenes como de los informes generados. Este almacenaje se explicará posteriormente y nos referiremos a el como *Almacenamiento temporal de archivos* 8.2.5.

8.2.5 Almacenamiento temporal de archivos

El componente de almacenamiento temporal de archivos corresponde con un sistema de almacenamiento de ficheros estándar necesario para almacenar los archivos temporales durante el procesado de las imágenes y la generación del informe en PDF. Durante estas operaciones se guardarán las imágenes antes de ser procesadas para poder transformarlas y pasarlas como entrada a los clasificadores. Además, para generar el informe, tanto las imágenes como los gráficos que se agregarán al fichero, son previamente guardados. Todos estos archivos generados son eliminados una vez acabe el procesado.

8.2.6 Almacenamiento persistente de datos

Por último el componente de almacenamiento persistente de datos corresponde con una base de datos MySQL donde se guardarán todos los datos referentes a los clínicos, pacientes, informes e imágenes.

Modelado de datos

En la Figura 8.2 se puede ver el esquema de la base de datos implementada para la aplicación. Se pueden apreciar alguna de las decisiones de implementación escogida, como puede ser el tamaño de los campos de texto o el formato escogido para el almacenamiento de las imágenes.

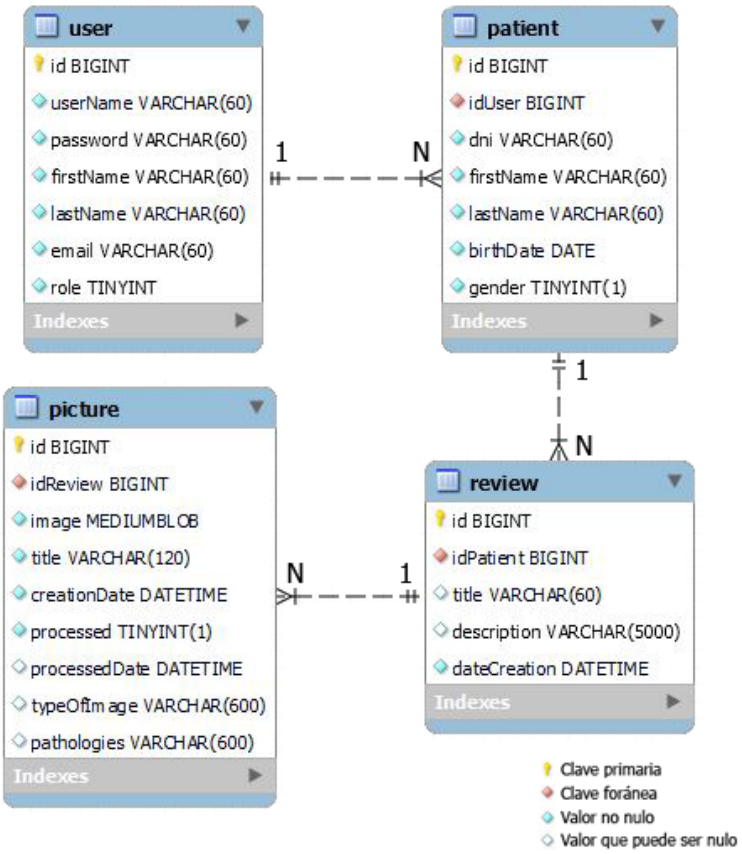


Figura 8.2: Esquema de la base de datos.

8.2.7 Diagrama de clases

Para más detalle del *backend* de gestión clínica se muestran las Figuras 8.3 y 8.4, donde podemos ver el diagrama de clases separado en dos imágenes para su mejor visualización. Se muestra únicamente el diagrama de este *backend* porque es por donde pasan todas las interacciones del usuario, además de ser el núcleo de Dx-OPHTHA, donde concentra toda la lógica de la aplicación.

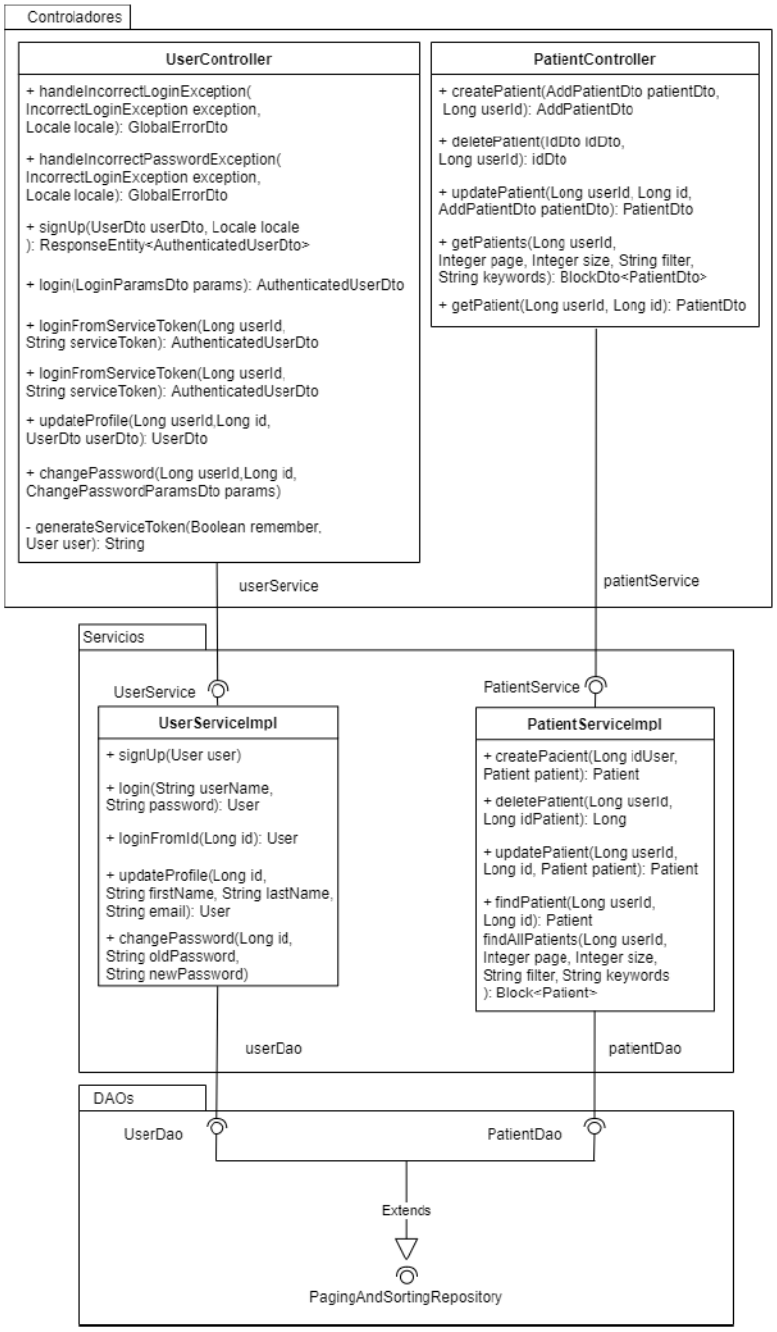


Figura 8.3: Primera parte del diagrama de clases.

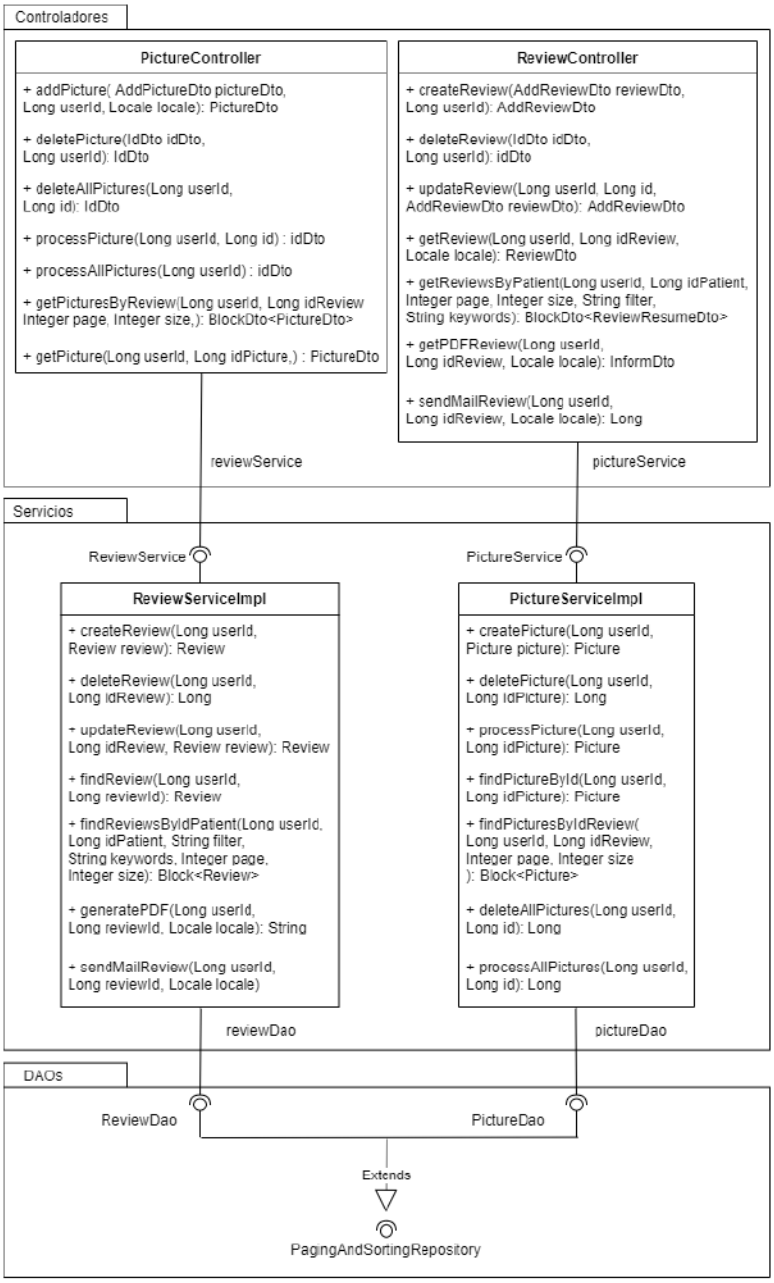


Figura 8.4: Segunda parte del diagrama de clases.

8.2.8 Patrones de diseño

DAO

El patrón *DAO* o *Data Access Object* es un patrón que proporciona una interfaz abstracta a un tipo de base de datos u otro mecanismo de persistencia. En esta aplicación los *DAO*s se utilizan para comunicar el servicio con la base de datos. Además, se apoya en la implementación

de *Spring* permitiendo utilizarlos de manera más rápida y eficiente. Concretamente todos los *DAO*s extienden *PagingAndSortingRepository* que es una clase que permite obtener los datos de forma ordenada y paginada. Se puede ver su uso en las Figuras 8.3 y 8.4.

DTO

Un *DTO* o *Data Transfer Object* es un objeto que transporta datos entre diferentes procesos. Son habituales en los servicios web. En este proyecto se emplean los *DTO*s en la comunicación entre la API REST del *backend* de gestión y el *frontend*, de esta manera se controla la información que llega al cliente del servicio.

Factory

El patrón *Factory* es un patrón de creación que utiliza métodos de fábrica para resolver el problema de la creación de objetos sin tener que especificar la clase exacta del objeto que se va a crear. Un ejemplo concreto en el contexto de esta aplicación es la declaración e implementación de los servicios. Se ha creado una interfaz con los métodos de los servicios y una clase que los sobre-escriba e implemente. Se puede ver su estructura en las Figuras 8.3 y 8.4.

Singleton

El patrón *Singleton* es un patrón de diseño que restringe la instanciación de una clase a una única instancia. Esto es útil cuando se necesita exactamente un objeto para coordinar acciones en todo el sistema. Para el desarrollo de esta aplicación este patrón se ha emplea para la instanciación de los servicios y controladores, es decir, todo aquello que le indiquemos a *Spring* que es un *bean*. Esta indicación le permite a *Spring* realizar correctamente la inyección de dependencias.

Facade

El patrón *Facade* es un patrón de diseño que funciona de forma análoga a una fachada en arquitectura, siendo este un objeto que sirve de interfaz que enmascara un código más complejo. En esta aplicación este patrón se encuentra tanto en la *API REST* del *backend* de procesado como en el *backend* de gestión. De forma trivial una *API REST* utiliza el patrón *Facade*.

State

El patrón *State* es un patrón de diseño de comportamiento que permite que un objeto altere su comportamiento cuando su estado interno cambia. Se ha empleado en el *frontend* con la propia implementación del *State* de *React*.

Observer

El patrón *Observer* es un patrón de diseño en el que un objeto mantiene una lista de sus dependientes, llamados observadores, y les notifica automáticamente cualquier cambio de estado. Este patrón está implementado de manera interna en *React* y se puede ver en el caso en el que los elementos de un componente cambian, en ese momento el componente se actualiza. Por lo que se puede decir que su uso ha sido transparente para el alumno.

8.3 Implementación

Una vez terminado con el diseño se detallarán algunas de las decisiones tomadas en la etapa de implementación.

8.3.1 Interfaz del usuario

A continuación se muestran puntos destacables de la interfaz del usuario desarrollada. En el apéndice B se presentan y explican algunos de los diseños escogidos para la interfaz.

8.3.2 Puntos destacables

En este apartado se enumerarán aspectos destacables y decisiones tomadas en la implementación de la aplicación.

Enfoque de diseño y desarrollo *Mobile first*

En este proyecto se sigue el enfoque de diseño y desarrollo *Mobile First*. Esta es una estrategia de diseño donde se modela la interfaz primeramente pensando en pantallas pequeñas (*SmartPhones*) y desde ahí hacer crecer el diseño para adaptarse a pantallas de mayor tamaño, como tablets y/o ordenadores. También se ha adaptado el diseño a pantallas ultra grandes, que correspondería con los formatos ultra panorámicos.

Paginación con elementos por página variable

Las diferentes listas que se muestran en la aplicación están paginadas permitiendo que la información se visualice de manera ordenada. Dependiendo del uso que el clínico quiera hacer de esa lista la cantidad de elementos por página podría variar, por ello se ha desarrollado la paginación de manera que sea posible seleccionar la cantidad de elementos por página. En la Figura 8.5 se puede ver el componente descrito.

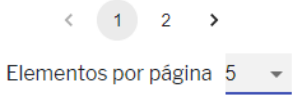


Figura 8.5: Componente de paginación.

JSON Web Token *(JWT)*

JSON Web Token *(JWT)* es un estándar abierto *(RFC 7519)* para la creación de tokens de acceso que permiten la propagación de identidad y privilegios. En el caso de este proyecto, el servidor genera un token indicando la autenticidad del usuario y se lo proporciona al cliente. El cliente podría utilizar el token para realizar ciertas acciones a las que típicamente tendría acceso un usuario. El token está firmado por la clave del servidor, así que el cliente y el servidor son ambos capaces de verificar que el token es legítimo. De esta forma y sabiendo que es una aplicación médica, este sistema de seguridad es válido y muy útil para este proyecto.

Gestión de errores

En cuanto a la gestión de errores, cualquier error producido en uno de los *backends* es procesado y mostrado en el *frontend*. De esta manera se hace llegar al usuario cualquier error producido en la aplicación, tanto de un campo específico como de un error general, mostrando además información específica sobre él. Por ejemplo, se mostrarán errores cuando un campo no cumple con los prerrequisitos, también lo hará cuando uno de los servicios no esté disponible o cuando se produzca un error interno en el servidor. Estos son algunos de los posibles errores que se gestionan.

i18n

Toda la aplicación, exceptuando los informes generados, está internacionalizada, permitiendo el uso del programa en cualquier región. Durante este proyecto se han considerado únicamente 3 idiomas: castellano, gallego e inglés; pero está desarrollada para permitir la inclusión de más idiomas.

L10n

Una vez más, toda la aplicación, exceptuando los informes generados, utiliza la localización para su uso en otras regiones. En este proyecto la localización se centra sobre todo en el formato de las fechas. Típicamente la localización junto a la internacionalización forman el término globalización o *g11n*.

Lógica de negocio

En cuanto a la lógica de negocio, únicamente se mencionará la restricción de creación y visualización de pacientes. Un clínico no puede tener pacientes repetidos, sí con el mismo nombre pero no con el mismo número identificativo: DNI, NIE o Pasaporte. Además, un mismo paciente puede tener varios clínicos asociados pero con distintos informes e imágenes, es decir, un clínico no puede ver los informes que tiene su paciente con otro clínico. Por otro lado un clínico solo puede ver sus pacientes, informes e imágenes asociadas a dicho paciente.

Escalabilidad en el formato de salida del procesamiento de imágenes

Para conseguir escalabilidad en el forma de salida del procesamiento de imágenes se ha creado un formato de salida presentado en la Figura 8.6 y que se detalla a continuación. Para el modelo de clasificación de dispositivos de captura se ha empleado una lista de listas con todas las clases posibles y un porcentaje que indica la precisión con la que determina que es esa misma clase. Como se puede ver la suma de estos porcentajes debe dar como resultado la unidad. Para los modelos de clasificación de patologías la estructura es idéntica, se indica en la primera posición de la lista la patología de la que se trata seguida del porcentaje de seguridad de la misma.

Gracias a esta estructura, en un futuro, se pueden crear modelos inteligentes capaces de diagnosticar más patologías y con información extra, de ahí que el formato escogido sea una lista de listas. Además, se ha empleado el mismo sistema para la clasificación del dispositivo de captura, por lo que sería posible introducir nuevos tipos de imágenes y añadir información extra para complementar la información del dispositivo de captura. Por tanto, el código existente sería completamente válido y funcional para cualquier nuevo modelo, siendo necesario únicamente, el procesando de la salida del modelo computacional en el formato indicado.

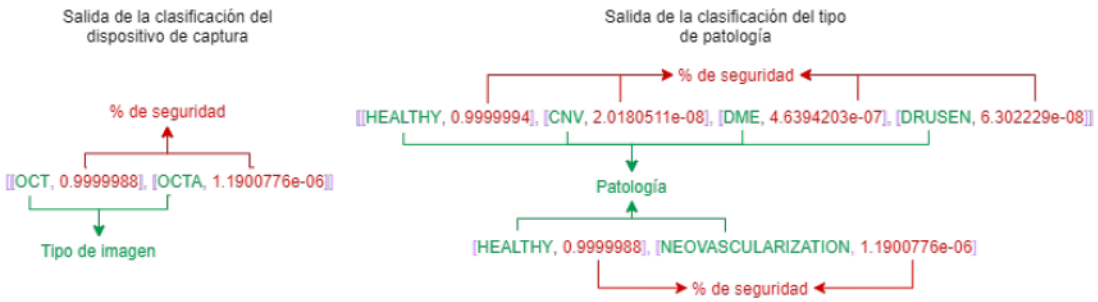


Figura 8.6: Formato de salida del procesado de imágenes.

Envío de emails

A través de la aplicación se pueden enviar los informes al email del usuario clínico. Para ello, se ha creado una cuenta en el proveedor de servicios de correo electrónico de Gmail y todos los emails enviados serán a través de esa cuenta. Cabe mencionar que se emplea *SSL* para evitar filtración de datos sensibles proveniente de los informes.

8.4 Pruebas

Como todo proyecto *software* es necesario que el código esté verificado y validado para que sea un producto válido para el cliente. Para esto es necesario diferenciar la verificación y la validación. A continuación podemos ver las definiciones por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) [37].

- ***Validación.*** *"La garantía de que un producto, servicio o sistema satisface las necesidades del cliente y de otras partes interesadas identificadas. Suele implicar la aceptación y adecuación con clientes externos."*

- ***Verificación.*** *"La evaluación de si un producto, servicio o sistema cumple o no con una normativa, requisito, especificación o condición impuesta. Suele ser un proceso interno."*

En cuanto a la verificación se tienen un conjunto de pruebas unitarias, de integración y de calidad que comprueban que el *software* cumple los requisitos funcionales y no funcionales de su especificación. Por otro lado, en cuanto a la validación, esta se ha llevado a cabo de manera manual a través de demostraciones del producto al cliente. Seguidamente se detallarán las pruebas tangibles que se han llevado a cabo en este proyecto:

8.4.1 Pruebas unitarias

Las pruebas unitarias sirven para probar el funcionamiento del código de manera aislada para cada componente. Esta tarea se ha empleado exclusivamente en el *backend* de gestión clínica, donde se concentra toda la lógica de negocio y que es necesaria probar. Para realizar las pruebas nos hemos ayudado de *JUnit* y *Mockito*. Este último especialmente útil para probar los controladores de manera aislada, sin depender del funcionamiento de los servicios. La métrica empleada para el control del *testing* ha sido la cobertura, que nos indica el porcentaje de lineas cubiertas por las pruebas pero que no es un indicador fiable ya que no nos indica que estén bien probadas.

8.4.2 Pruebas de integración

Las pruebas de integración sirven para comprobar el funcionamiento conjunto de varios componentes juntos. Estas pruebas se han llevado a cabo con *Postman*. Esta herramienta nos permite hacer consultas sin la necesidad de desarrollar un cliente, de esta manera podemos probar los *backends*. Además cuenta con una característica que nos permite crear servidores *mock*, es decir, servidores ficticios que devuelven los datos que nosotros le pidamos. De esta manera se ha probado el *frontend*. Gracias a esta herramienta y uniendo estas dos funciones se han realizado las pruebas de integración de la aplicación al completo.

8.4.3 Pruebas de calidad

Las pruebas de calidad sirven para detectar *bugs* o malas conductas en el código fuente. Además nos permite mantener el código legible y no redundante. Para esta tarea en el *frontend* se ha empleado *ESLint*, explicado anteriormente. Por otro lado, también se ha empleado *SonarQube* para la parte del *backend*. Esta última herramienta es especialmente útil ya que nos permite visualizar diferentes métricas de pruebas, asignación de resolución de tareas o incluso una estimación del tiempo que llevaría corregir los errores en el código. Se muestra en la Figura 8.7 un resumen de las métricas empleadas en el proyecto.

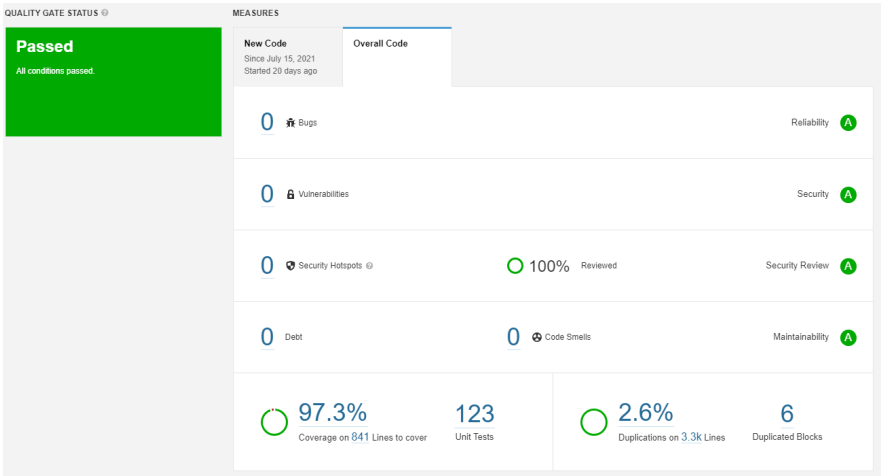


Figura 8.7: Resumen de las métricas de calidad de SonarQube.

Capítulo 9

Conclusiones

En este último capítulo del documento se expone el estado final del proyecto, las lecciones aprendidas y el trabajo futuro, cerrando con esto todo el trabajo incluido en este documento.

9.1 Estado final

Se han desarrollado diferentes modelos para el diagnóstico de patologías en base a distintos dispositivos de captura (OCT/OCT-A), los cuales son actualmente los tipos de imágenes de vanguardia y que tienen una progresión futura mayor. Para la elaboración de estos modelos se ha conseguido llevar a cabo un análisis patológico en el cual se han escogido las afecciones más comunes y representativas en el ámbito oftalmológico, haciendo del proyecto un producto útil y de interés en la actualidad. En este trabajo se ha logrado, a través de la explotación del *Deep Learning*, producir modelos simples que de manera totalmente convolucional procesen imágenes de entrada y, de manera interna y a través de diferentes módulos, se consiga extraer un diagnóstico útil para un perfil clínico. Estos módulos inteligentes finalmente han sido 3: (I) Módulo de análisis según el tipo de dispositivo de captura de imagen, (II) Módulo de análisis patológico en imágenes OCT y (III) Módulo de análisis patológico en imágenes OCT-A. El porcentaje de éxito de estos modelos es satisfactorio, como se puede ver en el capítulo 7. Por otro lado, se ha conseguido desarrollar una aplicación eficaz, completa y probada de un gestor de pacientes e integrarla con los modelos computacionales para realizar un diagnóstico inteligente sobre las imágenes clínicas. Asimismo, se han creado más funcionalidades para completar el servicio ofrecido por la aplicación, como puede ser el envío de correos o la exportación de informes.

A mayores y con el objetivo de mejorar la calidad del proyecto, se han integrado funcionalidades no previstas en el anteproyecto, como pueden ser la internacionalización y la localización o incluso el despliegue en contenedores con Docker. También cabe destacar que

se han encontrado problemas en el transcurso del proyecto. El principal problema ha sido el desconocimiento del dominio por parte del alumno, que supuso un retrasado en el desarrollo de la aplicación. Otro problema ha sido el entrenamiento de los modelos ya que con el *hardware* del alumno supuso más de un 15% (5 semanas) del tiempo total del proyecto.

Habiendo conseguido llevar a cabo el proyecto al completo y logrando todos los objetivos marcados al principio del mismo, se puede cerrar el apartado comentando que el estado final del proyecto es incluso un poco mayor del previsto, por lo que se puede decir que es satisfactorio.

9.2 Trabajo futuro

En esta sección se describen posibles tareas futuras que podrían incluirse en el desarrollo de este proyecto.

9.2.1 Conjunto de datos

Las conclusiones extraídas de los modelos creados en este proyecto están ligadas a los *datasets* empleados, por lo que sería interesante utilizar conjuntos de datos, más amplios y más variados. De esta forma, las conclusiones podrían variar y se podría extraer una idea más general del *screening* de este tipo de patologías. Cabe destacar también que, cuanto mayores y más variados sean los *datasets* más eficaz serán los modelos, proporcionándonos un mejor funcionamiento de la aplicación. Por otro lado, si la ampliación de los *datasets* no fuese posible, gracias al *data augmentation*, se podrían mejorar los conjuntos de datos ya existentes, proporcionándonos más cantidad de imágenes diferentes con el mismo *dataset*. El *data augmentation* es un término acuñado a una estrategia en el mundo de análisis de datos que permite aumentar significativamente la diversidad de datos disponibles para el entrenamiento de los modelos, sin necesidad de recoger nuevos datos.

9.2.2 Arquitecturas

Los datos de los entrenamientos de los modelos extraídos y analizados se basan en una arquitectura específica, la *DenseNet-161*. Como afirman Khan *et al.* [38] en su trabajo, actualmente existen numerosas arquitecturas de *CNN*s y con el paso del tiempo, el crecimiento de éstas será exponencial. Todas estas nuevas arquitecturas podrían aumentar la eficiencia de los modelos y, en consecuencia, mejorar el rendimiento de nuestra herramienta web.

9.2.3 Entorno cambiante

Como se ha mencionado anteriormente, con la aparición de nuevas tecnologías y nuevas formas de diagnóstico, el campo de la oftalmología está cambiando muy rápidamente. Este proyecto se ha limitado a dos tipos de dispositivos de captura, OCT y OCT-A, y cuatro patologías, CNV, DME, drusas y RVO. Por ello, la aplicación se ha desarrollado para incluir nuevos tipos de dispositivos de captura de imagen y nuevas patologías. Por lo tanto, como trabajo futuro, se podrían añadir nuevos dispositivos para diagnosticar patologías existentes dentro del modelo o incluso nuevas patologías e integrarlas en el herramienta. De este modo, las funcionalidades de la plataforma web podrían ampliarse progresivamente hasta alcanzar un nivel de diagnóstico igual o superior al de los expertos humanos, pudiendo utilizar conjuntamente diferentes tipos de imágenes oftalmológicas para realizar un diagnóstico multimodal de la misma patología.

9.2.4 Nuevas aplicaciones y *marketing*

Como se comenta en el sección 2.4, este proyecto es único hasta la fecha y surge de una idea innovadora. Además, con el crecimiento de los modelos de Dx-OPHTHA, se podrían desarrollar otras aplicaciones externas para crear así un ecosistema clínico mucho más completo y que se adapte a la metodología de trabajo de cada centro clínico. Como se ha mencionado en esta sección, esta aplicación tiene características únicas que podrían comercializarse fácilmente y ponerse a disposición de los centros clínicos interesados.

9.2.5 Rendimiento

Uno de los aspectos que no se han tenido en cuenta en este proyecto ha sido el rendimiento de la aplicación en un servidor. Sería interesante monitorizarlo y optimizarlo, ya que, al utilizar inteligencia artificial, los recursos computacionales utilizados por las *GPUs* son elevados. Por otro lado y con vistas al anterior apartado, para que un número elevado de tráfico no afectase al rendimiento de la aplicación, también sería necesario hacer un seguimiento tanto del rendimiento de los *backends* como de la base de datos con el fin de optimizarlos.

9.3 Conclusiones del proyecto

En conclusión, podemos hacer una valoración muy positiva del trabajo realizado en este proyecto, ya que se han cumplido todos los objetivos establecidos. Además, al tratarse de una propuesta interdisciplinar, el alumno ha adquirido nuevos conocimientos, tanto específicos como generales. En cuanto a los específicos, podemos destacar la adquisición de conocimientos específicos sobre el campo clínico o sobre las tecnologías empleadas. En cuanto a

los genéricos, podemos mencionar la introducción a un proyecto real y la adquisición de conocimientos necesarios tanto en el mercado laboral actual como en un ámbito especializado como es el de la oftalmología.

Apéndices

Apéndice A

Casos de uso

En este anexo se detallan los casos de uso del sistema. Cada tabla corresponde con un caso de uso y se describe en ella su finalidad, las precondiciones necesarias para ejecutarlo y las postcondiciones del mismo. Además, se describe el flujo normal del caso de uso y un flujo secundario.

En estos casos de uso no tienen cabida los usuarios no registrados, por tanto, no se incluye la precondición de estar autenticado. Esto se debe a que el actor Clínico que se puede ver en la Figura A.1 ya es un usuario registrado y autenticado.

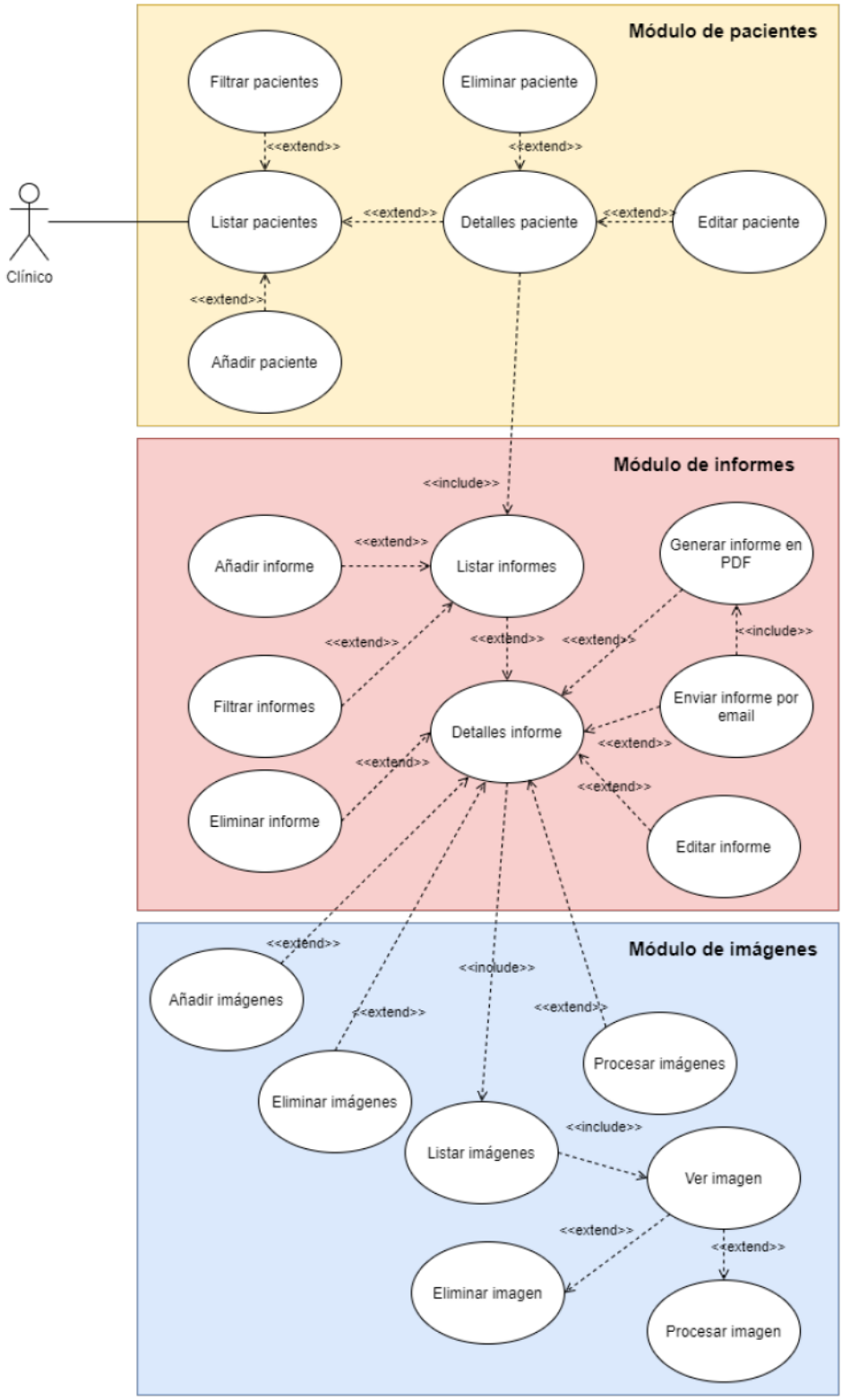


Figura A.1: Casos de uso.

Caso de uso	Listar pacientes
Actor	Clínico
Descripción	Se visualizan los pacientes asociados al clínico.
Precondiciones	-
Postcondiciones	-
Flujo principal	1. El clínico debe acceder a la vista que corresponda con listar pacientes. 2. Automáticamente aparecerán sus pacientes de forma paginada.
Flujo secundario	En caso de que no tenga pacientes asociados aparecerá un aviso de que no existen pacientes.

Tabla A.1: Caso de uso: Listar pacientes.

Caso de uso	Filtrar pacientes
Actor	Clínico
Descripción	Se listan los pacientes asociados al clínico con diferentes filtros.
Precondiciones	Se debe estar en la vista de listar pacientes.
Postcondiciones	Se mostrarán solo los pacientes filtrados.
Flujo principal	1. El clínico debe seleccionar el filtro que desee. 2. Automáticamente se mostrarán los pacientes filtrados.
Flujo secundario	Para volver al estado sin filtros bastaría con quitar el filtro seleccionado o volver manualmente al filtro por defecto.

Tabla A.2: Caso de uso: Filtrar pacientes.

Caso de uso	Añadir paciente
Actor	Clínico
Descripción	Se añade un paciente al sistema.
Precondiciones	El clínico debe estar en la vista de listar pacientes y el DNI/NIE o PASS asociado a ese paciente no debe estar dado de alta para ese mismo clínico.
Postcondiciones	-
Flujo principal	1. El clínico debe pulsar el botón de añadir paciente. 2. El clínico debe rellenar todos los datos asociados al paciente. 3. El clínico debe pulsar en añadir paciente.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.3: Caso de uso: Añadir paciente.

Caso de uso	Ver detalles paciente
Actor	Clínico
Descripción	Se detallan los datos del paciente.
Precondiciones	Estar en la vista de listar pacientes y que el paciente exista.
Postcondiciones	-
Flujo principal	1. El clínico debe pulsar el botón de ver paciente. 2. Se visualizarán los datos de ese mismo paciente.
Flujo secundario	-

Tabla A.4: Caso de uso: Ver detalles paciente.

Caso de uso	Eliminar paciente
Actor	Clínico
Descripción	Se elimina un paciente del sistema.
Precondiciones	Se debe estar en la vista de detalles de paciente y este debe existir.
Postcondiciones	El paciente y la información relacionada con él, tanto informes como imágenes, deja de estar disponible en el sistema.
Flujo principal	1. El clínico debe pulsar el botón de borrar paciente. 2. El clínico debe confirmar la eliminación.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.5: Caso de uso: Eliminar paciente.

Caso de uso	Editar paciente.
Actor	Clínico.
Descripción	Se edita un paciente del sistema.
Precondiciones	Se debe estar en la vista de detalles de paciente y el paciente tiene que existir.
Postcondiciones	El paciente se guardará en el sistema con los nuevos datos.
Flujo principal	1. El clínico debe pulsar el botón de editar paciente. 2. El clínico debe cubrir la información actualizada del paciente. 3. El clínico debe pulsar el botón de guardar.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.6: Caso de uso: Editar paciente.

Caso de uso	Listar informes.
Actor	Clínico.
Descripción	Se visualizan los informes de un paciente asociado al clínico.
Precondiciones	Se debe estar en la vista de detalles de paciente y el paciente tiene que existir.
Postcondiciones	-
Flujo principal	1. Automáticamente aparecerán sus informes de forma paginada.
Flujo secundario	En caso de que no tenga informes asociados aparecerá un aviso de que no existen informes.

Tabla A.7: Caso de uso: Listar informes.

Caso de uso	Añadir informe.
Actor	Clínico.
Descripción	Se añade un informe al sistema.
Precondiciones	El clínico debe estar en la vista de detalles del paciente.
Postcondiciones	-
Flujo principal	1. El clínico debe pulsar el botón de añadir informe. 2. El clínico debe rellenar todos los datos asociados al informe. 3. El clínico debe pulsar en guardar informe.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.8: Caso de uso: Añadir informe.

Caso de uso	Ver detalles informe.
Actor	Clínico.
Descripción	Se detallan los datos del informe.
Precondiciones	Que el informe exista y se debe estar en la vista de detalles del paciente.
Postcondiciones	-
Flujo principal	1. El clínico debe pulsar el botón de ver informe. 2. Se visualizarán los datos de ese mismo informe.
Flujo secundario	-

Tabla A.9: Caso de uso: Ver detalles informe.

Caso de uso	Eliminar informe.
Actor	Clínico.
Descripción	Se elimina un informe del sistema.
Precondiciones	Que el informe exista y encontrarse en la vista de detalles de informe.
Postcondiciones	El informe y las imágenes asociadas dejan de estar disponibles en el sistema.
Flujo principal	1. El clínico debe pulsar el botón de borrar informe. 2. El clínico debe confirmar la eliminación.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.10: Caso de uso: Eliminar informe.

Caso de uso	Editar informe.
Actor	Clínico.
Descripción	Se edita un paciente del sistema.
Precondiciones	Que el informe exista y encontrarse en la vista de detalles de informe.
Postcondiciones	El informe se guardará en el sistema con los nuevos datos.
Flujo principal	1. El clínico puede editar el informe directamente en la vista de detalles de informe. 2. El clínico debe pulsar el botón de guardar cambios.
Flujo secundario	En caso de que el clínico quiera cancelar la operación simplemente no deberá guardar los cambios.

Tabla A.11: Caso de uso: Editar informe.

Caso de uso	Filtrar informes.
Actor	Clínico.
Descripción	Se listan los informes con diferentes filtros.
Precondiciones	-
Postcondiciones	Se mostrarán solo los informes filtrados.
Flujo principal	1. El clínico debe seleccionar el filtro que desee. 2. Automáticamente se mostrarán los informes filtrados.
Flujo secundario	Bastaría con quitar el filtro seleccionado o volver manualmente al por defecto.

Tabla A.12: Caso de uso: Filtrar informe.

Caso de uso	Guardar informes en PDF.
Actor	Clínico
Descripción	Se listan los informes con diferentes filtros.
Precondiciones	-
Postcondiciones	Se mostrarán solo los informes filtrados.
Flujo principal	1. El clínico debe seleccionar el filtro que desee. 2. Automáticamente se mostrarán los informes filtrados.
Flujo secundario	Bastaría con quitar el filtro seleccionado o volver manualmente al por defecto.

Tabla A.13: Caso de uso: Guardar informes en PDF.

Caso de uso	Generar informe en PDF.
Actor	Clínico.
Descripción	Se genera el informe en PDF para exportar.
Precondiciones	Se tiene que estar en la vista de detalle de informe y el informe tiene que existir.
Postcondiciones	-
Flujo principal	1. El clínico debe pulsar el botón de generar informe. 2. Automáticamente se abrirá el informe en PDF.
Flujo secundario	-

Tabla A.14: Caso de uso: Generar informe en PDF.

Caso de uso	Enviar informe por email.
Actor	Clínico.
Descripción	Se enviará el informe al email del clínico.
Precondiciones	Se tiene que estar en la vista de detalle de informe y el informe tiene que existir.
Postcondiciones	-
Flujo principal	1. El clínico debe pulsar el botón de enviar informe al email. 2. Automáticamente se recibirá un email con el informe en PDF adjunto.
Flujo secundario	-

Tabla A.15: Caso de uso: Enviar informe por email.

Caso de uso	Añadir imágenes
Actor	Clínico
Descripción	Se añade una o varias imágenes al sistema.
Precondiciones	El clínico debe estar en la vista de detalle de informe y el informe tiene que existir.
Postcondiciones	-
Flujo principal	1. El clínico debe arrastrar las imágenes o seleccionarlas de alguna manera y pulsar en el botón de subir imágenes.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.16: Caso de uso: Añadir imágenes.

Caso de uso	Listar imágenes.
Actor	Clínico.
Descripción	Se podrán visualizar todas las imágenes del informe.
Precondiciones	Se debe estar en la vista de detalles del informe y el informe debe existir.
Postcondiciones	-
Flujo principal	1. Automáticamente aparecerán las imágenes de forma paginada.
Flujo secundario	En caso de que no tenga imágenes asociadas aparecerá un aviso de que no existen imágenes.

Tabla A.17: Caso de uso: Listar imágenes.

Caso de uso	Eliminar imágenes.
Actor	Clínico.
Descripción	Se eliminan todas las imágenes asociadas al informe del sistema.
Precondiciones	Que el informe exista y encontrarse en la vista de detalles de informe. Si no existen imágenes en el informe no se podrá llevar a cabo esta acción.
Postcondiciones	El informe quedará sin ninguna imagen.
Flujo principal	1. El clínico debe pulsar el botón de borrar todas las imágenes. 2. El clínico debe confirmar la eliminación.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.18: Caso de uso: Eliminar imágenes.

Caso de uso	Procesar imágenes.
Actor	Clínico.
Descripción	Se podrán procesar por lotes todas las imágenes del informe.
Precondiciones	Se debe estar en la vista de detalles del informe y el informe debe existir. Si existen imágenes ya procesadas, estas no se procesarán.
Postcondiciones	Los resultados del procesado de las imágenes se irán mostrando a medida que avance el diagnóstico inteligente.
Flujo principal	1. El clínico debe pulsar en el botón de procesar todas las imágenes. 2. Se irán mostrando los resultados en la lista de imágenes.
Flujo secundario	-

Tabla A.19: Caso de uso: Procesar imágenes.

Caso de uso	Ver imagen.
Actor	Clínico.
Descripción	Se podrán ver los detalles de una imagen concreta.
Precondiciones	Se debe estar en la vista de detalles del informe y tanto el informe como la imagen deben existir.
Postcondiciones	-
Flujo principal	1. Automáticamente aparecerán los detalles de la imagen.
Flujo secundario	-

Tabla A.20: Caso de uso: Ver imagen.

Caso de uso	Eliminar imagen.
Actor	Clínico.
Descripción	Se podrá eliminar una imagen concreta del informe.
Precondiciones	Se debe estar en la vista de detalles del informe y tanto el informe como la imagen deben existir.
Postcondiciones	La imagen eliminada deja de estar disponible en el sistema.
Flujo principal	1. El clínico debe pulsar el botón de borrar imagen. 2. El clínico debe confirmar la eliminación.
Flujo secundario	En caso de que el clínico quiera cancelar la operación pulsará cancelar o retroceder en la aplicación.

Tabla A.21: Caso de uso: Eliminar imagen.

Caso de uso	Procesar imagen.
Actor	Clínico.
Descripción	Se podrá procesar una imagen concreta de un informe.
Precondiciones	Se debe estar en la vista de detalles del informe y el informe debe existir. Si la imagen ya está procesada no se podrá volver a procesar.
Postcondiciones	Los resultados del procesado de las imágenes se irán mostrando a medida que avance el diagnóstico inteligente.
Flujo principal	1. El clínico debe pulsar en el botón de procesar todas las imágenes. 2. Se mostrarán progresivamente los resultados en la lista de imágenes.
Flujo secundario	-

Tabla A.22: Caso de uso: Procesar imagen.

Apéndice B

Interfaz

En este anexo se muestran diferentes imágenes sobre la interfaz elegida para la aplicación web, acompañada de una pequeña justificación.

B.1 Diseño general

En las Figuras B.1, B.2, B.3 y B.4 se muestra el diseño general de la aplicación, incluyendo la portada, una mención al equipo de trabajo y los servicios ofertados. Además, se puede ver la cabecera y el píe de página.

Figura B.1: Portada de la aplicación web.

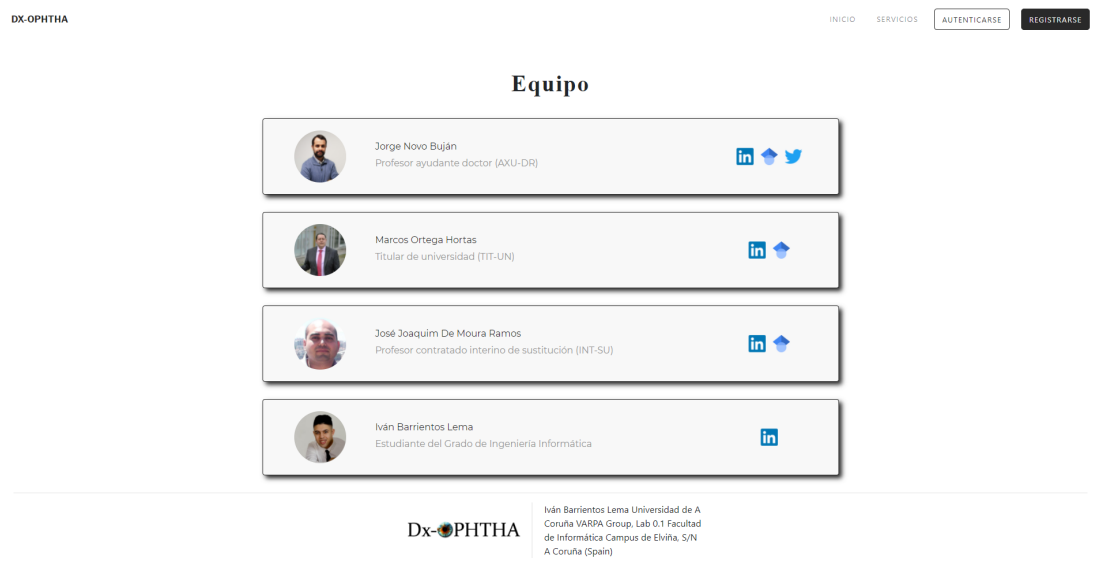


Figura B.2: Mención al equipo de trabajo.

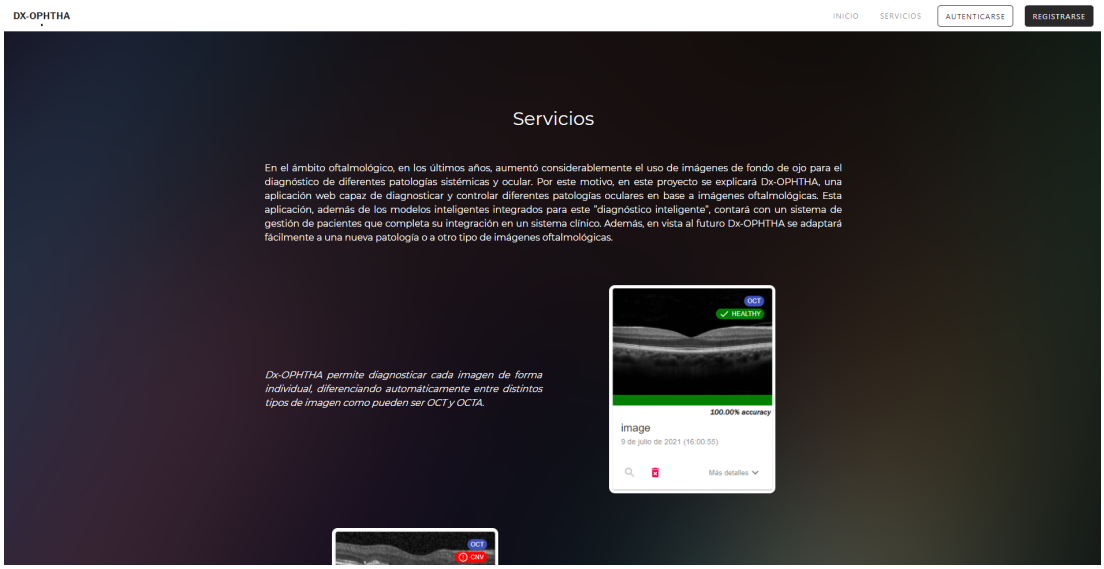


Figura B.3: Descripción de los servicios ofrecidos en la aplicación (1/2).

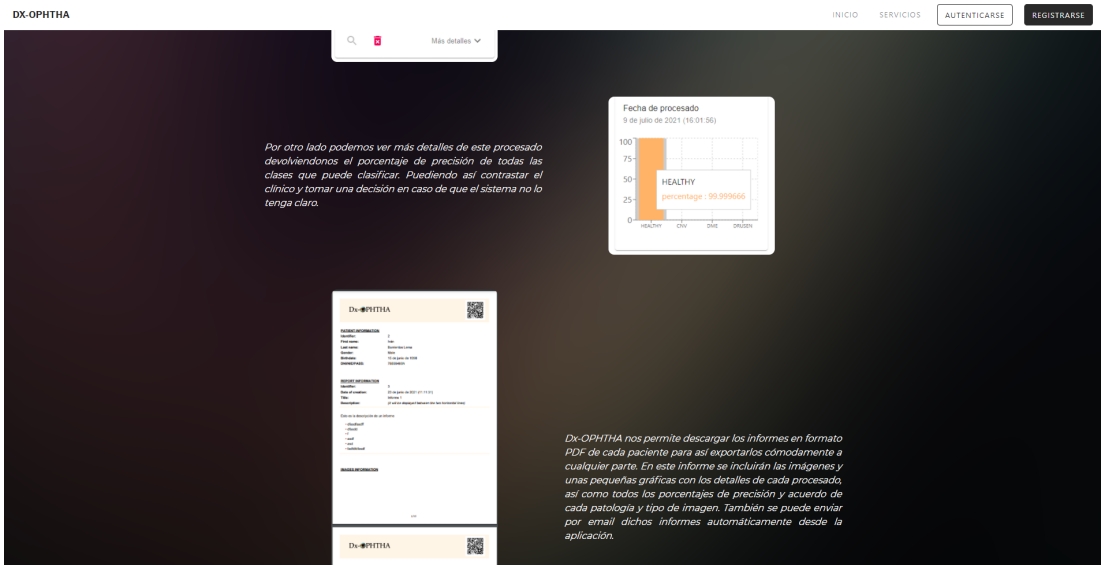


Figura B.4: Descripción de los servicios ofrecidos en la aplicación (2/2).

B.2 Autenticación y registro

En la Figura B.5 se muestra la pantalla para la autenticación donde se pedirá el usuario y la contraseña con la posibilidad de recordar sesión, para así, que quede abierta por más tiempo. El tiempo de expiración de la sesión por defecto es de 24 horas. El tiempo de expiración de la sesión extendida es de 10 días.

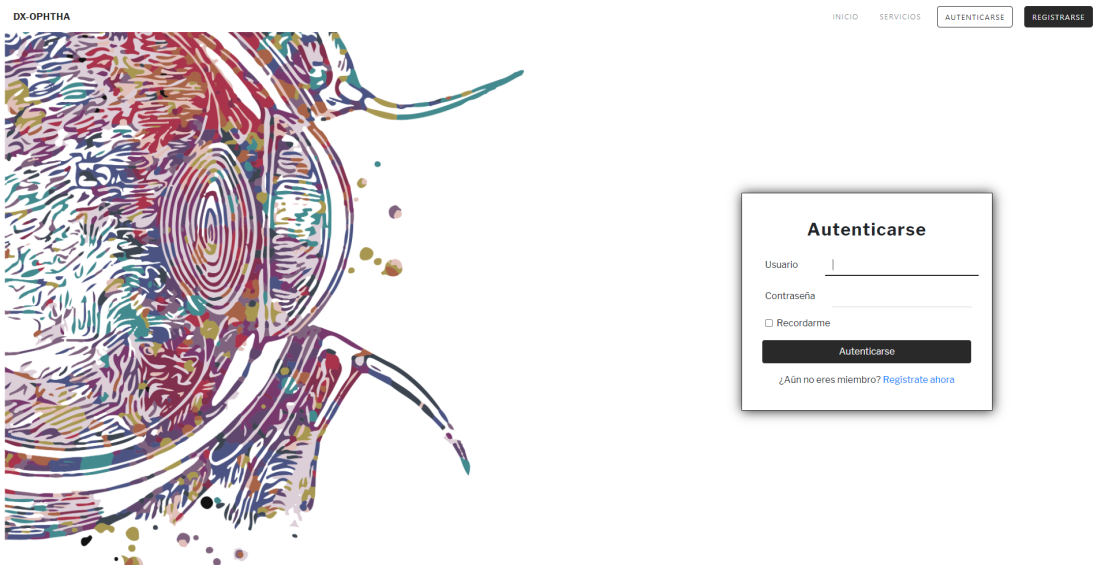


Figura B.5: Pantalla de inicio de sesión.

En la Figura B.6 se muestra la pantalla de registro donde se piden los campos necesarios para la creación del perfil. Destacar que el campo usuario es el necesario para el inicio de sesión y debe ser único.

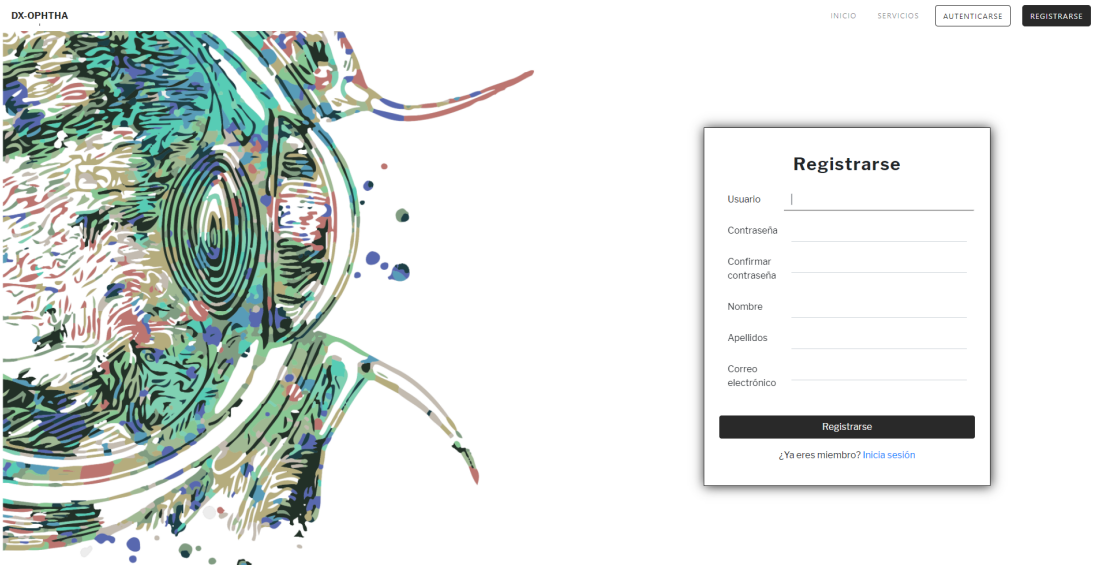


Figura B.6: Pantalla de registro.

En la Figura B.7 se muestra el cambio de formato de la cabecera una vez se inicia sesión. También se puede ver todas las nuevas opciones disponibles para un usuario autenticado.

Figura B.7: Cabecera con la sesión iniciada.

B.3 Pacientes

En la Figura B.8 se muestra el diseño de la lista de pacientes disponibles para un clínico. Esta vista está compuesta de una tabla paginada con varios datos relevantes de los pacientes, un buscador de pacientes y un botón para añadir un nuevo paciente.

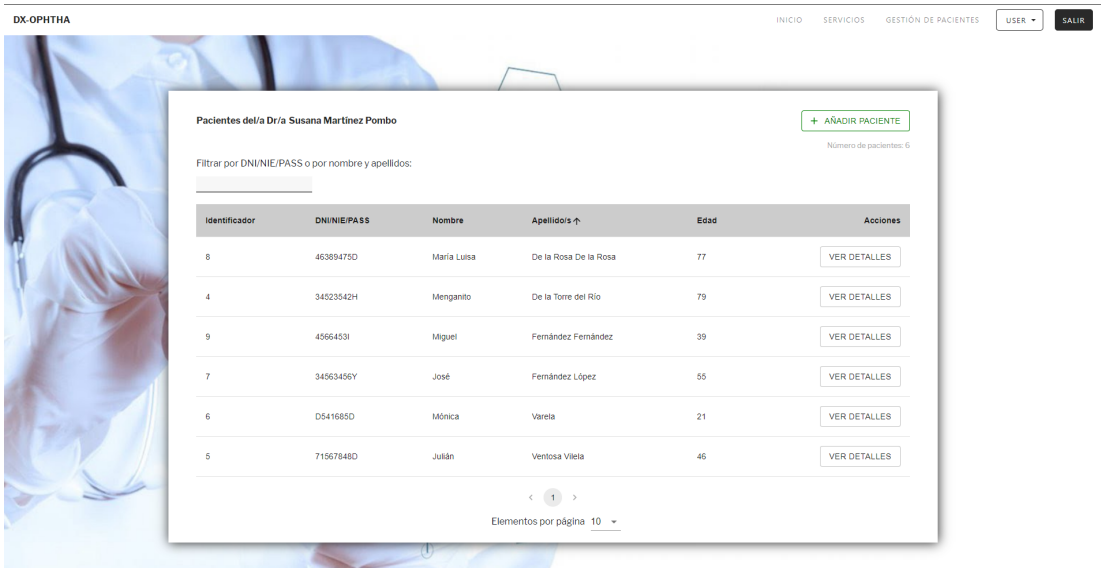


Figura B.8: Interfaz con la lista de pacientes.

En la Figura B.9 se muestra la pantalla para añadir un paciente nuevo.

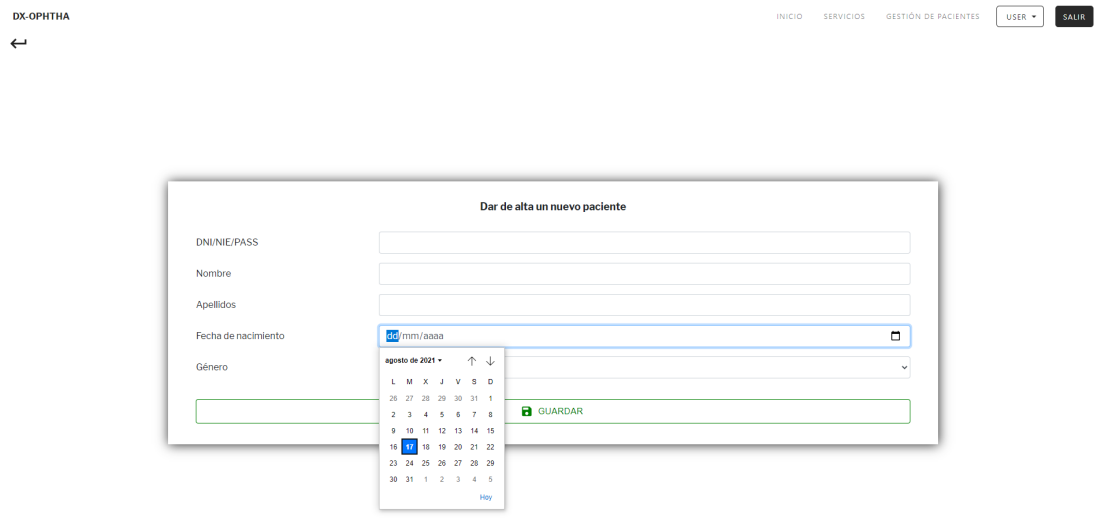


Figura B.9: Interfaz para añadir un paciente.

En la Figura B.10 se muestra la pantalla de detalles de un paciente. En esta pantalla tenemos a la izquierda los distintos datos almacenados del paciente. A la derecha tenemos todos los informes asociados a ese paciente, así como un botón para añadir nuevos informes.

Figura B.10: Pantalla de detalles de un paciente.

En la Figura B.11 se muestra la misma pantalla que la Figura B.10 pero una vez pulsado el botón actualizar, haciendo que los distintos datos del paciente ahora sean modificables.

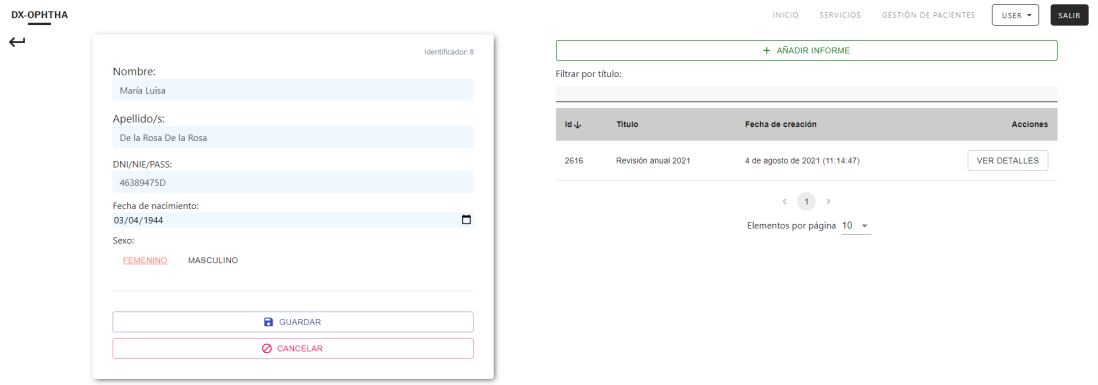


Figura B.11: Pantalla para actualizar detalles de un paciente.

B.4 Informes

En la Figura B.12 se muestra la pantalla para añadir un nuevo informe. Como se puede ver será necesario indicar un título y una descripción. La descripción tendrá formato y será

guardada con él.

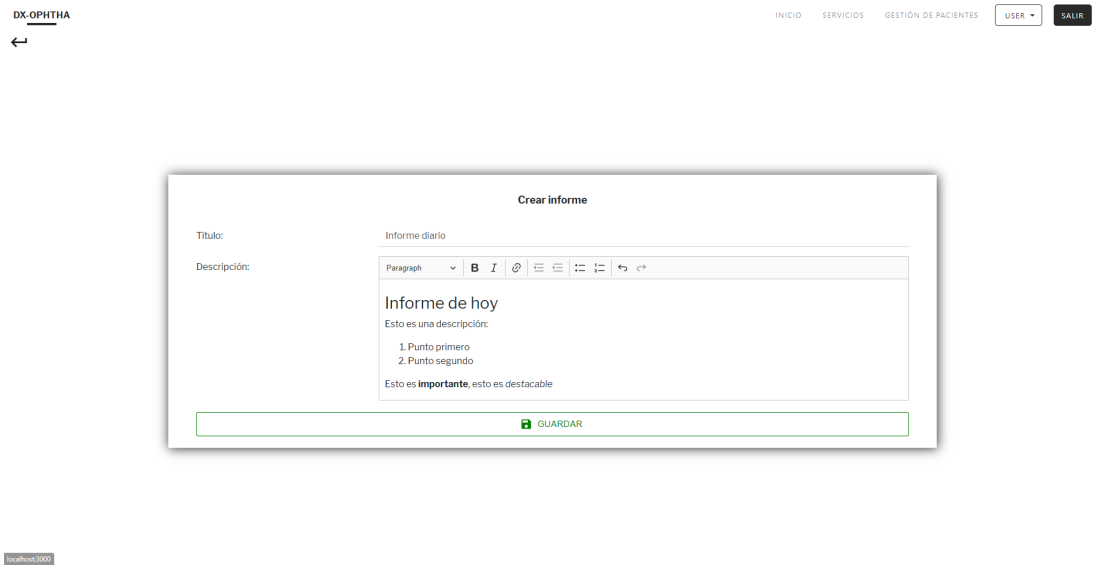


Figura B.12: Interfaz para crear un informe.

En la Figura B.13 y B.14 se muestra la pantalla de detalles del informe. La parte superior está compuesta por los datos del informe y seguidamente las imágenes asociadas a él. Además, cada imagen cuenta con el diagnóstico y estadísticas de su procesado, eso sí, si la imagen está procesada, si no, estará vacío. Por otro lado están las opciones de descargarlo en PDF y de enviarlo por email en la parte superior.

Para editar el informe solo será necesario escribir directamente en él y pulsar el botón de guardar cambios. Para añadir imágenes solo hace falta arrastrarlas a él y se añadirán automáticamente. Con estas dos funcionalidades lo que se busca es que la ampliación del informe por parte del clínico sea mucho más sencilla.

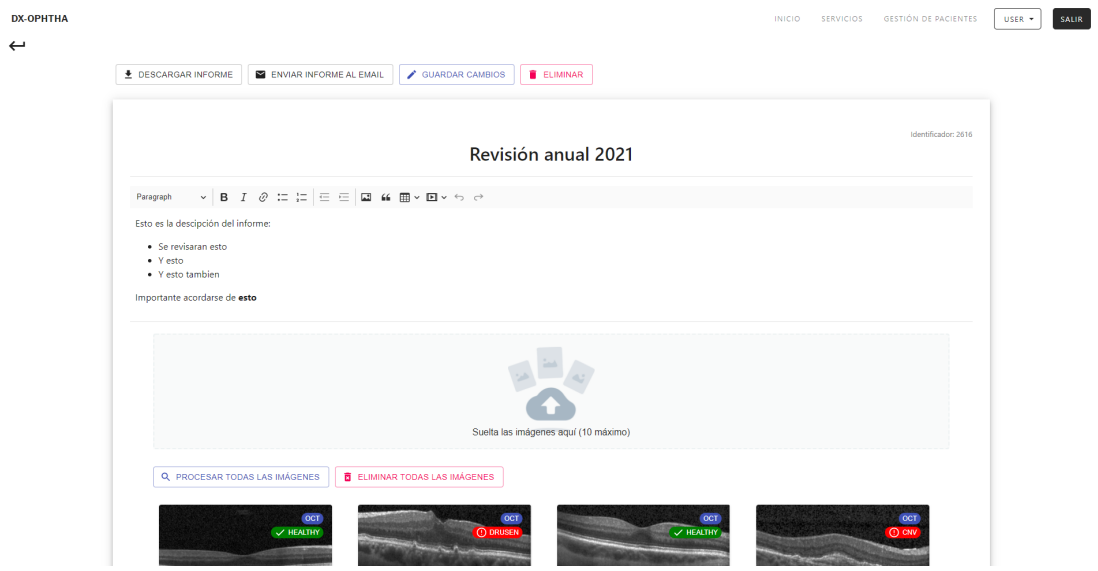


Figura B.13: Pantalla de detalles de un informe.

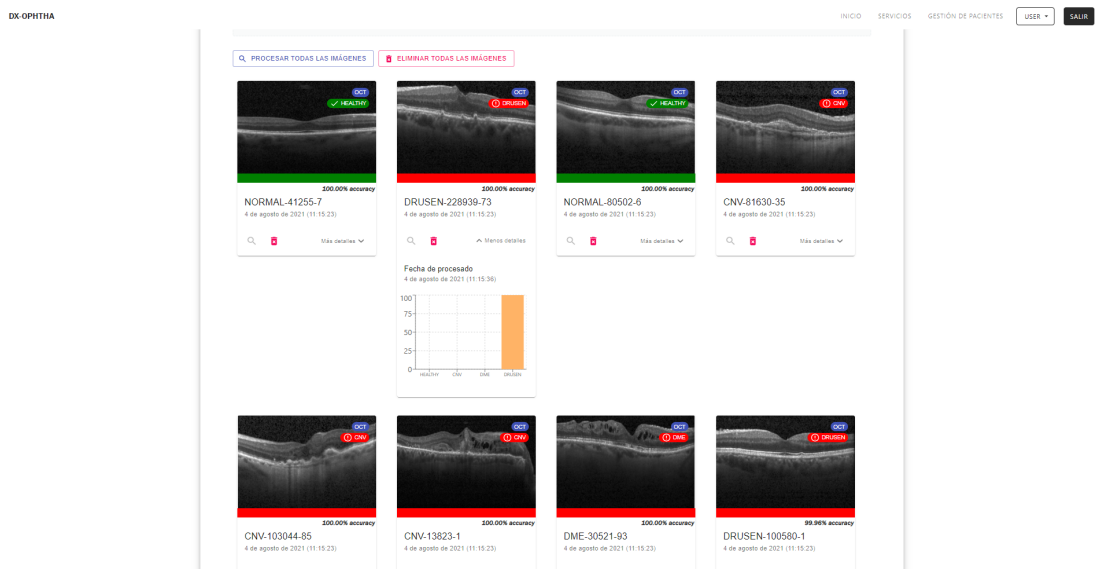


Figura B.14: Pantalla de detalles de un informe con imágenes.

B.5 Exportación de informes

Para la exportación de informes se emplea la información tanto del informe como de todos los datos asociados a él, imágenes y datos del paciente. Por otro lado se seguirá siempre la misma estructura para mayor legibilidad del informe. En una primera parte tendrá una cabecera con el nombre de la aplicación y un código QR para acceder rápidamente a su versión

on-line. Seguidamente estará la información del paciente y después la información principal del informe. Finalmente se mostrarán todas las imagen pertenecientes a este informe, acompañadas de gráficas resultantes de su procesado, donde se indicarán estadísticas referentes a su diagnóstico. Se puede apreciar lo explicado en las Figuras B.15 y B.16.

Dx-OPHTHA

PATIENT INFORMATION

Identifier:	8
First name:	María Luisa
Last name:	De la Rosa De la Rosa
Gender:	Female
Birthdate:	3 de abril de 1944
DNI/NIE/PASS:	46389475D

REPORT INFORMATION

Identifier:	2616
Date of creation:	4 de agosto de 2021 (11:14:47)
Title:	Revisión anual 2021
Description:	*(It will be displayed between the two horizontal lines)*

Esto es la descipción del informe:

- Se revisaran esto
- Y esto
- Y esto tambien

Importante acordarse de esto

IMAGES INFORMATION

Figura B.15: Informe exportado (1/2).

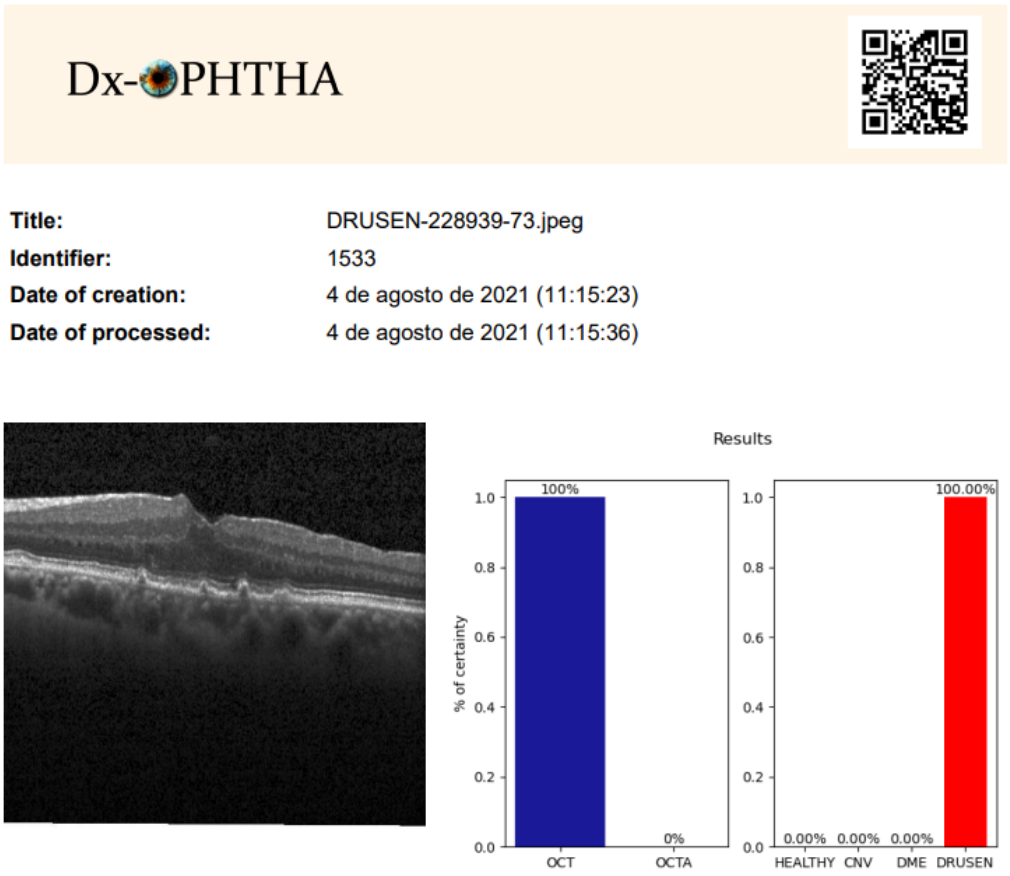


Figura B.16: Informe exportado (2/2).

B.6 Email

En la Figura B.17 se muestra el formato escogido para los emails enviados por la aplicación. Como se puede ver el asunto tiene el formato de ”Dx-OPHTHA: *Nombre del informe*”. El cuerpo del email es siempre el mismo a excepción del ID, que ira variando en función del propio ID del informe.

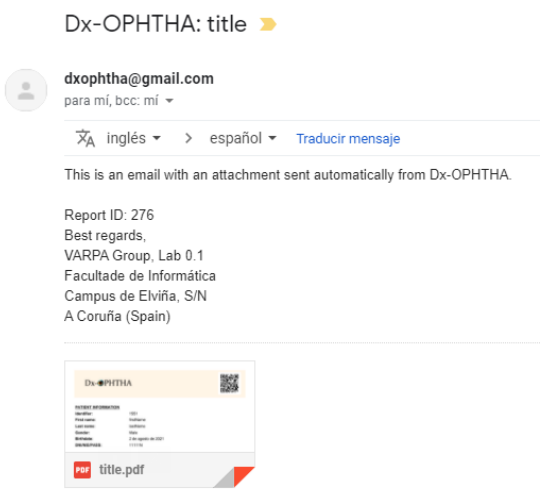


Figura B.17: Formato del email enviado por la aplicación.

Lista de acrónimos

API *Application Programming Interface*

CNN *Convolutional Neural Network*

CNV *Choroidal Neovascularization*

DenseNet *Dense Convolutional Network*

DME *Diabetic macular edema*

DRUSEN *Drusas*

FN *False Negative*

FP *False Positive*

GPU *Graphics Processing Unit*

HTML *HyperText Markup Language*

IDE *Integrated Development Environment*

JVM *Java Virtual Machine*

ML *Machine Learning*

NO-RVO *No Retinal Vein Occlusion*

OCT *Optical Coherence Tomography*

OCT-A *Optical Coherence Tomography Angiography*

PDF *Portable Document Format*

POM *Project Object Model*

ReLu *Rectified Linear Unit*

REST *Representational State Transfer*

RVO *Retinal Vein Occlusion*

SGD *Stochastic gradient descent*

SPA *Single Page Application*

SSL *Secure Sockets Layer*

TN *True Negative*

TP *True Positive*

Glosario

Accuracy Precisión.

Backend Parte de una aplicación encargada de la lógica.

Batch normalization Normalización por lotes.

Batch size Tamaño del lote.

Classical Machine Learning Aprendizaje máquina clásico.

Convolutional Neural Networks Red de neuronas convolucionales.

Datasets Conjunto de datos.

Data augmentation Aumento de datos añadiendo copias ligeramente modificadas a partir de datos ya existentes.

Deep Learning Aprendizaje profundo.

Dense Blocks Bloques densos.

Epoch Época, etapa, ciclo.

F1-score Puntuación F1.

Feedback Realimentación.

Fine-tuning Ajuste, perfección.

Frames Fotograma.

Frameworks Entorno de trabajo.

Frontend Parte de una aplicación encargada de la interfaz.

Graphics Processing Unit Unidad de procesado de gráficos.

Idem Lo mismo.

Kernel Matriz de convolución.

Loss Pérdida.

Machine Learning Aprendizaje máquina.

Mini-batch Mini-lote.

Momentum Impulso.

Overfitting Sobre-ajuste o sobre-entrenamiento.

Pipeline Mapa de etapas o flujo de un proceso.

Plugins Complementos.

Pooling Muestreo descendente de imágenes.

Precision Exactitud.

Recall Sensibilidad.

Screening Cribado.

Softmax Exponencial normalizada.

Stochastic Gradient Descent Descenso de gradiente estocástico.

Subsampling Submuestreo.

Time-box Caja o intervalo de tiempo.

Transfer Learning Transferencia de aprendizaje.

Bibliografía

[1] P. Ruiz. (2018) Densenets. [Online]. Available: http://www.pabloruizruiz10.com/resources/CNNs/DenseNets.pdf

[2] *1996 CERN School of Computing: Egmond aan Zee, The Netherlands 8 - 21 Sep 1996. 19th CERN School of Computing*, CERN. Geneva: CERN, 1996. [En línea]. Disponible en: http://cds.cern.ch/record/300250

[3] G. Tortora and B. Derrickson, *Principios de anatomía y fisiología.* Editorial Médica Panamericana, 2013. [En línea]. Disponible en: https://books.google.es/books?id=9nuzDAEACAAJ

[4] C. Starr and R. Taggart, *Biologia/ Biology: La unidad y diversidad de la vida*, ser. Books in the brooks / Cole biology series. Thomsom, 2008. [En línea]. Disponible en: https://books.google.es/books?id=iHDGTzWE-fcC

[5] H. G. Bezerra, M. A. Costa, G. Guagliumi, A. M. Rollins, and D. I. Simon, "Intracoronary optical coherence tomography: A comprehensive review: Clinical and research applications," *JACC: Cardiovascular Interventions*, 2009.

[6] D. Huang, E. A. Swanson, C. P. Lin, J. S. Schuman, W. G. Stinson, W. Chang, M. R. Hee, T. Flotte, K. Gregory, C. A. Puliafito, and J. G. Fujimoto, "Optical Coherence Tomography," *Science*, 1991.

[7] T. E. de Carlo, A. Romano, N. K. Waheed, and J. S. Duker, "A review of optical coherence tomography angiography (octa)," *International Journal of Retina and Vitreous*, vol. 1, no. 1, 2015.

[8] R. Varma, N. M. Bressler, Q. V. Doan, M. Gleeson, M. Danese, J. K. Bower, E. Selvin, C. Dolan, J. Fine, S. Colman, and A. Turpcu, *Prevalence of and risk factors for diabetic macular edema in the United States.* JAMA Ophthalmology, 2014.

[9] C. C. C De Graeve 1, W Van de Sompel, *Ocular ischemic syndrome: two case reports of bilateral involvement.* Bull Soc Belge Ophtalmol., 1999.

[10] T. Mitchell, *Machine Learning*, 1997.

[11] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015. [En línea]. Disponible en: https://www.sciencedirect.com/science/article/pii/S0893608014002135

[12] I. C. Education. (2020) Neural networks.

[13] *Flexible, High Performance Convolutional Neural Networks for Image Classification.* Twenty-Second International Joint Conference on Artificial Intelligence.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [En línea]. Disponible en: https://doi.org/10.1145/3065386

[16] G. Huang, Z. Liu, and L. van der Maaten, *Densely Connected Convolutional Networks.* Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017.

[17] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, 2016. [En línea]. Disponible en: http://arxiv.org/abs/1608.06993

[18] J. Yoon, J. Han, J. In Park, H. J. Seo, J. Mo Hanand Joonhong Sohn, K. Hyung Park, and D. Duck-Jin Hwang, "Optical coherence tomography-based deep-learning model for detecting central serous chorioretinopathy," *Scientific Reports*, 11 2020.

[19] P. L. Vidal, J. de Moura, M. Díaz, J. Novo, and M. Ortega, "Comparative and behavioural analysis of a diffuse paradigm for the evaluation of diabetic macular edema in oct images," in *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS).* IEEE, 2021, pp. 13–18.

[20] M. Pekala, N. Joshi, T. A. Liu, N. M. Bressler, D. C. DeBuc, and P. Burlina, "Deep learning based retinal oct segmentation," *Computers in biology and medicine*, vol. 114, p. 103445, 2019.

[21] T. Schlegl, S. M. Waldstein, H. Bogunovic, F. Endstraßer, A. Sadeghipour, A.-M. Philip, D. Podkowinski, B. S. Gerendas, G. Langs, and U. Schmidt-Erfurth, "Fully automated detection and quantification of macular fluid in oct using deep learning," *Ophthalmology*, vol. 125, no. 4, pp. 549–558, 2018.

[22] J. de Moura, J. Novo, M. Ortega, N. Barreira, and M. G. Penedo, "Automated segmentation of the central serous chorioretinopathy fluid regions using optical coherence tomography scans," in *2021 IEEE 34th International Symposium on Computer-Based Medical Systems (CBMS)*. IEEE, 2021, pp. 1–6.

[23] M. Gende, J. De Moura, J. Novo, P. Charlón, and M. Ortega, "Automatic segmentation and intuitive visualisation of the epiretinal membrane in 3d oct images using deep convolutional approaches," *IEEE Access*, vol. 9, pp. 75 993–76 004, 2021.

[24] D. Le, T. Son, and X. Yao, "Machine learning in optical coherence tomography angiography," *Experimental Biology and Medicine*, 07 2021.

[25] D. Le, M. N. Alam, J. I. Lim, R. Chan, and X. Yao, "Deep learning for objective octa detection of diabetic retinopathy," in *Ophthalmic Technologies XXX*, vol. 11218. International Society for Optics and Photonics, 2020, p. 112181P.

[26] Y. Xu, Y. Su, D. Hua, P. Heiduschka, W. Zhang, T. Cao, J. Liu, Z. Ji, and N. Eter, "Enhanced visualization of retinal microvasculature via deep learning on octa image quality," *Disease Markers*, vol. 2021, 2021.

[27] M. Gao, Y. Guo, T. T. Hormel, J. Sun, T. S. Hwang, and Y. Jia, "Reconstruction of high-resolution 6× 6-mm oct angiograms using deep learning," *Biomedical Optics Express*, vol. 11, no. 7, pp. 3585–3600, 2020.

[28] Z. Jiang, Z. Huang, B. Qiu, X. Meng, Y. You, X. Liu, M. Geng, G. Liu, C. Zhou, K. Yang *et al.*, "Weakly supervised deep learning-based optical coherence tomography angiography," *IEEE Transactions on Medical Imaging*, vol. 40, no. 2, pp. 688–698, 2020.

[29] L. Xi, Z. Huang, Z. Wang, C. Wen, Z. Jiang, Z. yu, J. Liu, G. Liu, X. Huang, A. Maier, Q. Ren, and Y. Lu, "A deep learning based pipeline for optical coherence tomography angiography," *Journal of Biophotonics*, vol. 12, 06 2019.

[30] D. S. Kermany, M. Goldbaum, W. Cai, C. C. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang. (2018) Identifying

medical diagnoses and treatable diseases by image-based deep learning. [En línea]. Disponible en: https://www.cell.com/cell/fulltext/S0092-8674(18)30154-5

[31] W. on. [En línea]. Disponible en: https://w3techs.com/technologies/details/cp-javascript

[32] J. S. Ken Schwaber, "The scrum guide," 2021. [En línea]. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[33] H. Takeuchi and I. Nonaka, "The new new product development game," *Harvard Business Review*, 1986.

[34] "Cuánto cobra un ingeniero informático," 2021. [En línea]. Disponible en: https://universidadeuropea.com/blog/cuanto-gana-un-ingeniero-informatico

[35] "Sueldo profesor universidad. ¿cuánto ganan los docentes?" 2021. [En línea]. Disponible en: https://oposiciones.es/noticias/sueldo-profesor-universidad/

[36] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. (2017) Densenet-161 pre-trained model for pytorch. [En línea]. Disponible en: https://www.kaggle.com/pytorch/densenet161

[37] "Ieee draft guide: Adoption of the project management institute (pmi) standard: A guide to the project management body of knowledge (pmbok guide)-2008 (4th edition)," *IEEE P1490/D1, May 2011*, 2011.

[38] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, 2020.