Sistema para la detección de patologías médicas mediante el análisis de imágenes

Memoria

VNiVERSiDAD
D SALAMANCA

Trabajo de Fin de Grado

Grado de Ingeniería Informática

Julio de 2021

Tutores:

De Paz Santana, Juan Francisco

García Encinas, Francisco

Villarrubia González, Gabriel

Alumno:

Óscar Sánchez Barriga

Dr. Juan Francisco De Paz Santana, D. Francisco García Encinas y Dr. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

Hacen constar:

Que el trabajo titulado “Sistema para la detección de patologías médicas mediante el análisis de imágenes” ha sido realizado por D. Óscar Sánchez Barriga, con DNI 70918413Y y contribuye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 6 de julio de 2021

DE PAZ SANTANA JUAN FRANCISCO - 76125754D

Firmado digitalmente por DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-76125754D, givenName=JUAN FRANCISCO, sn=DE PAZ SANTANA, cn=DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Fecha: 2021.07.06 20:52:23 +02'00'

Dr. Juan Francisco De Paz Santana

VILLARRUBIA GONZALEZ GABRIEL - 71932845T

Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-71932845T, givenName=GABRIEL, sn=VILLARRUBIA GONZALEZ, cn=VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2021.07.07 11:50:17 +02'00'

Dr. Gabriel Villarrubia González

GARCIA ENCINAS FRANCISCO - 21120042Q

Digitally signed by GARCIA ENCINAS FRANCISCO - 21120042Q
Date: 2021.07.07 08:37:15 +02'00'

D. Francisco García Encinas

Resumen

Este trabajo de fin de grado, como su nombre indica, consta de la investigación y realización de un sistema para la detección de patologías médicas mediante el análisis de imágenes. Esta aplicación web ayuda a los especialistas médicos detectar el cáncer de mama y registrarlo en una base de datos.

La detección de esta enfermedad se centra en el análisis de imágenes previamente introducidas por un médico, estas se guardarán en un directorio con el nombre del DNI y un subdirectorio con la fecha de la realización del análisis. La inteligencia artificial que se ha creado con el propósito de la detección de esta enfermedad analizará las distintas imágenes introducidas y dará un resultado de la paciente. Además de esto, se podrá acceder a los distintos análisis que se han realizado a las pacientes, ver los resultados y las imágenes analizadas.

Para el desarrollo de la inteligencia artificial se ha usado el lenguaje de programación Python en Colaboratory, un servicio en la nube, basado en los Notebooks de Jupyter, que permite el uso gratuito de las GPUs y TPUs de Google.

Para el desarrollo de la aplicación web se ha decidido usar Flask, un entorno de trabajo escrito en Python que permite la creación de aplicaciones web, dando un servicio a los distintos usuarios. Flask está basado en la especificación WSGI de Werkzeug y el motor de templates Jinja2 y tiene licencia BSD.

En la elaboración de las páginas web se ha usado HTML, junto con SB Admin 2, un tema bootstrap creado con HTML, CSS y JavaScript que permite crear interfaces web.

En el almacenamiento de los datos de las pacientes y los médicos, se ha utilizado MySQL, un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual (licencia pública general y licencia comercial por Oracle Corporation) y está considerada como la base de datos de código abierto más popular del mundo.

A lo largo del proyecto se han seguido las prácticas del Proceso Unificado siguiendo lo aprendido en la asignatura de Ingeniería del Software. Además, se ha realizado una estimación de esfuerzo del proyecto, utilizando UCP y una planificación temporal del proyecto, gracias a los conocimientos adquiridos en la asignatura de Gestión de Proyectos.

Summary

This end-of-degree project, as its name suggests, consists of the research and development of a system for the detection of medical pathologies through image analysis. This web application helps medical specialists to detect breast cancer and register it in a database.

The detection of this disease focuses on the analysis of images previously entered by a doctor, these will be stored in a directory with the name of the DNI and a subdirectory with the date of the analysis. The artificial intelligence that has been created for the purpose of detecting this disease will analyze the different images entered and give a result of the patient. In addition to this, it will be possible to access the different analyses that have been performed on the patients, view the results and the analyzed images.

For the development of the artificial intelligence, the Python programming language has been used in Colaboratory, a cloud service, based on Jupyter Notebooks, which allows the free use of Google GPUs and TPUs.

For the development of the web application it has been decided to use Flask, a working environment written in Python that allows the creation of web applications, giving a service to different users. Flask is based on the WSGI specification of Werkzeug and the Jinja2 template engine and is BSD licensed.

HTML has been used in the development of the web pages, together with SB Admin 2, a bootstrap theme created with HTML, CSS and JavaScript to create web interfaces.

MySQL, a relational database management system developed under a dual license (general public license and commercial license by Oracle Corporation) and considered to be the most popular open source database in the world, was used to store patient and physician data.

Throughout the project, the practices of the Unified Process have been followed following what has been learned in the subject of Software Engineering. In addition, a project effort estimation has been made, using UCP and a time planning of the project, thanks to the knowledge acquired in the Project Management course.

Índice de contenidos

1. Introducción 14
2. Objetivos 17
2.1 Objetivos del sistema 17
2.2 Objetivos personales 17
3. Conceptos teóricos 18
4. Técnicas y herramientas usadas 23
4.1 Herramientas y entornos de desarrollo 23
4.2 Herramientas CASE 26
4.3 Framework 27
4.4 Lenguajes 27
4.5 Bibliotecas de Python para la IA 28
4.6 Bootstrap 29
4.7 Datasets 30
5. Aspectos relevantes del desarrollo 32
5.1 Marco de trabajo 32
5.2 Estimación de coste y esfuerzo 33
5.2.1 Cálculo de los UUCP 33
5.2.2 Cálculo de los TCF 35
5.2.3 Cálculo de los ECF 36
5.2.4 Resultados 37
5.3 Planificación temporal 37
5.4 Especificación de requisitos 39
5.4.1 Objetivos 39
5.4.2 Requisitos de información 40
5.4.3 Requisitos de restricción de información 41
5.4.4 Requisitos no funcionales 41
5.4.5 Requisitos funcionales 42
5.5 Análisis de sistema 44
5.5.1 Modelo de dominio 44
5.5.2 Paquete de análisis 45
5.5.3 Vista arquitectónica 46
5.5.4 Realización de los casos de uso 47
5.6 Diseño del sistema 47
5.6.1 Patrones arquitectónicos 47

5.6.2 Subsistemas de diseño 48
5.6.3 Clases de diseño 50
5.6.4 Vista arquitectónica 51
5.6.5 Realización de los casos de uso 52
5.6.6 Modelo de despliegue 52
5.7 Implementación 53
5.7.1 Implementación del modelo de red neuronal 53
5.7.2 Implementación de la aplicación web 62
5.8 Pruebas 62
5.8.1 Pruebas sobre el modelo de red neuronal 63
5.8.2 Pruebas sobre la aplicación web 65
5.9 Funcionalidad de la aplicación 65
6. Conclusión y líneas de trabajo futuras 80
6.1 Conclusión 80
6.2 Líneas de trabajo futuras 82
7. Referencias 84
8. Bibliografía 85

Tabla de Ilustraciones

Ilustración 1: Símbolo de la conciencia del cáncer de mama 14
Ilustración 2: Funcionamiento de una neurona artificial 19
Ilustración 3: Función Sigmoide 19
Ilustración 4: Función ReLU 20
Ilustración 5: Capas de una red neuronal 20
Ilustración 6: Visual Studio Code 23
Ilustración 7: PyCharm 24
Ilustración 8: Colaboratory 24
Ilustración 9: XAMPP 25
Ilustración 10: Visual Paradigm 26
Ilustración 11: EZ Estimate 26
Ilustración 12: Microsoft Project 27
Ilustración 13: SB Admin 2 29
Ilustración 14: Conjunto de entrenamiento, validación y prueba 31
Ilustración 15: Proceso unificado 32
Ilustración 16: Resultados de estimación de coste y esfuerzo 37
Ilustración 17: Diagrama de Gantt parte 1 38
Ilustración 18: Diagrama de Gantt parte 2 38
Ilustración 19: Diagrama de Gantt parte 3 39
Ilustración 20: Diagrama de actores 42
Ilustración 21: Diagrama de casos de uso 44
Ilustración 22: Modelo de dominio 45
Ilustración 23: Diagrama de paquetes de análisis 46
Ilustración 24: Vista de arquitectura del modelo de análisis 46
Ilustración 25: Realización del UC1 Iniciar sesión 47
Ilustración 26: Arquitectura MVC 48
Ilustración 27: Subsistema de vistas 49
Ilustración 28: Subsistema de controladores 50
Ilustración 29: Diagrama de clases del modelo de diseño 50
Ilustración 30: Diagrama de vista arquitectónica 51
Ilustración 31: Diagrama de secuencia del UC1 Iniciar sesión 52
Ilustración 32: Diagrama de despliegue 53
Ilustración 33: trainAug ImageDataGenerator 55
Ilustración 34: valAug ImageDataGenerator 55
Ilustración 35: testAug ImageDataGenerator 56
Ilustración 36: trainGen flow_from_directory 57
Ilustración 37: valGen flow_from_directory 57
Ilustración 38: testGen flow_from_directory 58
Ilustración 39: Modelo 58
Ilustración 40: Primer bloque 60
Ilustración 41: Segundo bloque 60
Ilustración 42: Tercer bloque 60
Ilustración 43: Cuarto bloque 61
Ilustración 44: Compilar y entrenar 61

Ilustración 45: Métricas de evaluación 63
Ilustración 46: Función de perdida y precisión 64
Ilustración 47: Inicio de sesión 66
Ilustración 48: Página principal 66
Ilustración 49: Contraseña olvidada 67
Ilustración 50: Correo electrónico para recuperar la contraseña 67
Ilustración 51: Solicitud de cuenta 68
Ilustración 52: Mostrar información del perfil 68
Ilustración 53: Editar información del perfil 69
Ilustración 54: Cerrar sesión 69
Ilustración 55: Index Gestión de pacientes 70
Ilustración 56: Añadir paciente 71
Ilustración 57: Seleccionar un paciente 71
Ilustración 58: Mostrar información de una paciente 72
Ilustración 59: Editar paciente 73
Ilustración 60: Eliminar paciente confirmación 73
Ilustración 61: Añadir imágenes 74
Ilustración 62: Diagnóstico de imágenes 75
Ilustración 63: Resultados del diagnóstico 76
Ilustración 64: Correo electrónico con el diagnóstico médico 76
Ilustración 65: Index Ver historial 77
Ilustración 66: Historial 77
Ilustración 67: Cambiar diagnóstico manualmente 78
Ilustración 68: Volver a realizar diagnóstico 79

Índice de tablas

Tabla 1: Complejidad de los actores del sistema 33
Tabla 2: Complejidad Gestión de médicos 34
Tabla 3: Complejidad Gestión de pacientes 34
Tabla 4: Complejidad Gestión de diagnósticos 34
Tabla 5: Complejidad Gestión de imágenes 34
Tabla 6: Factores de complejidad técnica 35
Tabla 7: Factores de complejidad de entorno 36
Tabla 8: Objetivo Gestión de pacientes 40
Tabla 9: Requisitos de información Datos de los médicos 40
Tabla 10: Requisitos de restricción Datos de los médicos 41
Tabla 11: Requisitos no funcionales Escalabilidad 42
Tabla 12: Actor médico 43
Tabla 13: Caso de uso Iniciar sesión 43

1. Introducción

En el año 2018 se diagnosticaron aproximadamente 2 millones de casos nuevos de cáncer de mama en todo el mundo. En la actualidad, es el tumor más frecuente en la población femenina y, aunque las tasas de cáncer de mama son más altas en países desarrollados, están aumentando en casi todas las regiones del mundo.

Según el Observatorio de Cáncer AECC, en España se diagnosticaron 33.000 nuevos casos en 2019, siendo estos el 30% de todos los tumores del sexo femenino en nuestro país.

Estos datos son estimados, pero si se puede concluir que, tanto el número de casos como las tasas de incidencia, aumentan lentamente en España y en el mundo, posiblemente debido al envejecimiento de la población y a un diagnóstico cada vez más temprano. Actualmente, según los datos del Observatorio de Cáncer AECC de 2018, se considera que el riesgo de padecer cáncer de mama a lo largo de la vida es de, aproximadamente, 1 de cada 8 mujeres.

En la "Ilustración 1: Símbolo de la conciencia del cáncer de mama" se puede ver el símbolo más representativo en la lucha contra el cáncer de mama. El lazo rosa representa a nivel internacional el compromiso y conciencia social de esta enfermedad.

Ilustración 1: Símbolo de la conciencia del cáncer de mama

Este tipo de enfermedad constituye la primera causa de muerte por cáncer entre las mujeres y en 2018, según los datos del Observatorio del Cáncer AECC, es de unos 28 fallecimientos por cada 100.000 habitantes. Además, en España el cáncer de mama supone la primera causa de mortalidad por cáncer en mujeres con 6.579 fallecimientos en 2018. Esto representa un 17% de todos los fallecimientos por cáncer del sexo femenino en nuestro país, y el 3.3% del total de muertes entre las mujeres. [1]

En esta aplicación creada se busca ayudar a predecir el cáncer de mama a través de imágenes de histología mamaria con una interfaz web que permite a los médicos introducir la información personal de la paciente y las imágenes, realizando así un análisis médico a través de la Inteligencia Artificial creada para esta función.

Con la implantación de este sistema, muchos de los profesionales sanitarios podrán obtener resultados de las pacientes mucho más rápido que de la forma convencional y así poder empezar cuanto antes con los tratamientos necesarios y la recuperación de la paciente. Además, este programa, permite llevar un historial de los análisis de la paciente y, de esta forma, poder tener una visión y un progreso de esta.

El sistema constará de una red neuronal convolucional (CNN) que analiza imágenes de histología mamaria en busca de factores de riesgo de cáncer y que se comunicará con el médico a través de la aplicación web. Esto pretende acercar la informática a la sanidad, haciendo así posible una evolución en este campo.

En esta memoria se explican los objetivos principales del proyecto, tanto desde un nivel personal, como desde los puntos que se persigue conseguir con este trabajo. También se expondrán las herramientas, recursos y técnicas utilizadas, junto con las metodologías elegidas, como el análisis, diseño, implementación, temporización y pruebas de la aplicación.

Para profundizar más en el desarrollo del trabajo, se han creado, junto a esta memoria, seis anexos que se dividen en:

- **Anexo I Planificación temporal:** En este documento se incluyen la planificación de tareas en el desarrollo de la aplicación, distribuidas sobre una línea temporal del calendario de trabajo.
- **Anexo II Especificación de requisitos:** En él se recogen los distintos requisitos que debe cumplir el sistema para la correcta realización del proyecto.
- **Anexo III Análisis del Sistema:** Se realiza un análisis de los requisitos encontrados en la fase de especificación. Entre estos se encuentran el modelo de dominio, paquetes de análisis, vista de la arquitectura y realización de caso de uso.
- **Anexo IV Diseño del Sistema:** En este anexo se comprende el diseño de la arquitectura del sistema, de la interfaz y las distintas pruebas realizadas en la aplicación.
- **Anexo V Manual del Programador:** Como su nombre indica, se trata de un manual enfocado a profesionales para conseguir que su trabajo sobre este

proyecto sea más eficiente, puesto que en él se detalla su funcionamiento y estructura referente a la programación del proyecto.

- **Anexo VI Manual de Usuario:** Se trata de un manual orientado al usuario final, en él se enseña la funcionalidad del sistema, y se explica de forma clara como utilizar las distintas opciones que ofrece la aplicación.

Además de estos anexos, en esta memoria se distinguen distintos apartados para una mejor comprensión de los pasos dados a lo largo del proyecto, divididos en los siguientes puntos:

- **Objetivos:** En este apartado se incluyen los puntos que se pretende conseguir en el desarrollo del trabajo. En él se verán tanto los personales como los del sistema.
- **Conceptos teóricos:** En él se abordan distintos términos y conceptos para tener en cuenta para la perfecta comprensión del trabajo realizado.
- **Técnicas y herramientas:** Se hablan de las distintas herramientas, lenguajes y formatos utilizados para el desarrollo del trabajo.
- **Aspectos relevantes del desarrollo:** En él se desarrollan las distintas etapas del proceso unificado por las que transcurre el desarrollo de la aplicación.
- **Conclusión y líneas de trabajo futuras:** Se contemplan los resultados vistos en el desarrollo y se plantean desarrollos futuros para incrementar la utilidad del proyecto.
- **Referencias y bibliografía:** Se recogen las referencias consultadas para el desarrollo tanto de la aplicación como de la memoria.

2. Objetivos

En este apartado se exponen los objetivos que cumplirá el sistema, así como los objetivos personales que se cumplen con este.

2.1 Objetivos del sistema

Los objetivos principales que se persiguen con la realización de este trabajo son:

- **Gestión de médicos:** Este objetivo del sistema busca el manejo de los datos de los médicos que son los usuarios de la aplicación, permitiendo así solicitar una cuenta, iniciar sesión, modificar distintos datos personales y recuperar la contraseña en caso de que se haya perdido.
- **Gestión de pacientes:** Este objetivo del sistema busca el manejo de los datos de los pacientes, así como la realización de diagnósticos sobre estos con la ayuda de la inteligencia artificial. También se busca el poder eliminarlos, modificarlos y añadir nuevos pacientes al sistema.
- **Gestión de imágenes:** La gestión de imágenes busca el poder seleccionarlas para realizar distintos diagnósticos, así como almacenarlas para futuras consultas y eliminarlas en caso de que ya no fueran necesarias.
- **Gestión de diagnósticos:** En el se busca poder realizar distintos diagnósticos, almacenarlos, así como modificarlos o consultarlos cuando el médico lo desee. También se busca el poder de eliminarlos y volver a realizar el análisis con la ayuda de la red neuronal.

2.2 Objetivos personales

La principal motivación del proyecto es profundizar en el uso, conocimiento y diseño de la inteligencia artificial en el análisis de imágenes, ya que es uno de los campos que me han resultado más interesantes. Además de esto, gracias a la convivencia con familiares que pertenecen al sector sanitario, se quiso juntar ambos campos para conseguir un objetivo común, el diagnostico de pacientes.

Extendiendo los conocimientos adquiridos durante la carrera, se creará una aplicación capaz de ofrecer una interfaz al usuario, en este caso un médico o profesional sanitario, que se comunicará con él y con la inteligencia artificial creada.

3. Conceptos teóricos

Inteligencia Artificial (IA):

La inteligencia artificial (*Artificial Intelligence* (AI) en inglés) es la simulación de tareas de inteligencia humana por parte de máquinas. Estos procesos incluyen el aprendizaje, razonamiento y la autocorrección. [2]

Machine Learning:

Machine Learning o aprendizaje automático es una técnica de análisis de datos que enseña a las computadoras. Esta dentro de la rama de la Inteligencia Artificial. [3]

Para diferenciar entre la Inteligencia Artificial y el *Machine Learning* es que la IA tiene la capacidad de mostrar un comportamiento "inteligente" y, para que esta se trate de *Machine Learning*, necesita usar una técnica para crear y mejorar dicho comportamiento, mediante entrenamientos con datos.

Deep Learning:

El *Deep Learning* es un subcampo del aprendizaje automático que estructura algoritmos en capas para crear una red neuronal artificial que puede aprender a tomar decisiones inteligentes por sí misma. En lugar de darle una lista de reglas para resolver un problema, se le dota de un modelo que pueda evaluar ejemplos y una pequeña colección de instrucciones para modificar el modelo cuando se produzcan errores. Después del entrenamiento, en la que se producirán esos pequeños cambios en el modelo, la IA será capaz de resolver problemas por si sola.

Neurona artificial:

Es una unidad de cálculo que intenta emular el comportamiento de una neurona del cerebro humano. La interconexión de un conjunto de estas unidades forma una red neuronal artificial.

Para el procesamiento de información, las neuronas tienen conexiones de entrada a través de los que reciben estímulos externos (valores de entrada Xn). Con ellos, realizará un cálculo interno y generará un valor de salida (Y). Este cálculo interno se trata de una suma ponderada de los valores de entradas con su peso asignado (Wn), es decir, cada conexión que llega a la neurona tendrá asociado un valor que servirá para definir con que intensidad cada variable de entrada afecta a la neurona, después pasará por una función de activación, dando así un valor de salida. Además de estas entradas, la neurona tendrá un valor llamado sesgo (*bias* en inglés) y se puede ver como otra conexión a la neurona, pero la variable siempre esta asignada a 1. [4]

En la "Ilustración 2: Funcionamiento de una neurona artificial" se puede observar el funcionamiento y procesamiento de información del a misma.

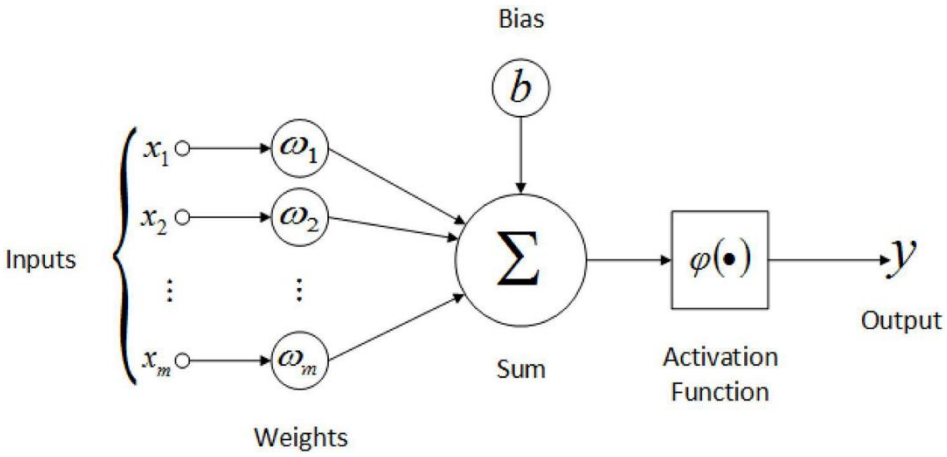


Ilustración 2: Funcionamiento de una neurona artificial

Función de activación

La función de activación utiliza la suma ponderada y la transforma como salida. Entre las diversas funciones de activación, destacan la sigmoide y ReLU.

Siendo la variable z la suma ponderada de entradas, la función sigmoide se define como en la "Ilustración 3: Función Sigmoide":

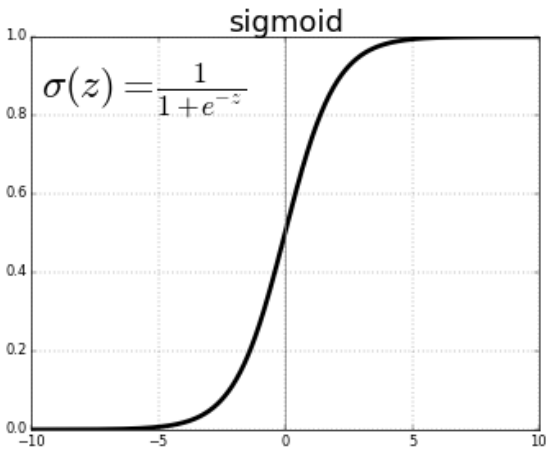


Ilustración 3: Función Sigmoide

Y la función ReLU como en la "Ilustración 4: Función ReLU":

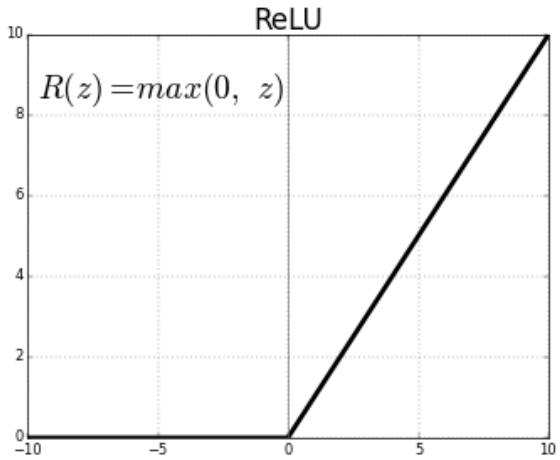


Ilustración 4: Función ReLU

Red neuronal artificial

Las redes neuronales artificiales (RNA) se basan en el funcionamiento del cerebro humano. Está compuesta por neuronas artificiales conectadas entre sí, agrupadas en diferentes niveles llamados capas.

Capas

Son los niveles en los que se estructura una red neuronal. Hay varios tipos, las de entrada que reciben los datos, las de salida que darán un resultado final al usuario y las capas ocultas, las cuales contienen unidades no observables. En la "Ilustración 5: Capas de una red neuronal", se puede ver como se estructuran en una red neuronal.

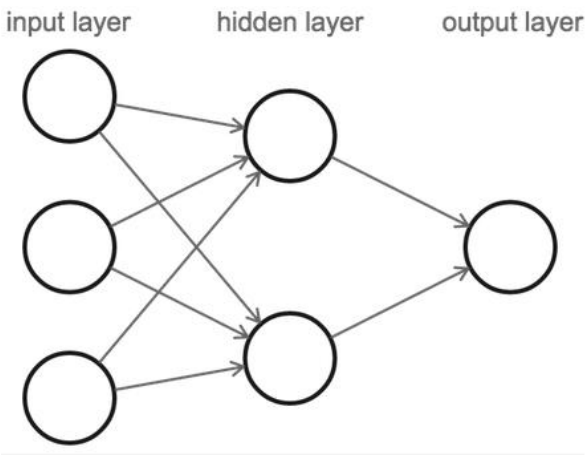


Ilustración 5: Capas de una red neuronal

CNN (*Convolutional neural network*)

La CNN es un tipo de Red Neuronal Artificial con aprendizaje supervisado que procesa sus capas imitando al ojo humano para identificar distintas características en las entradas que hace que puedan identificar figuras y objetos. [5]

Su funcionamiento consiste en aplicar filtros a cada imagen de entrenamiento con distintas resoluciones, y la salida de cada imagen convulsionada se emplea como entrada para la siguiente capa.

Image datasets

Image datasets es una colección de datos o imágenes que tiene la finalidad de entrenar a una red neuronal.

Underfitting

En español se puede traducir como “infrajuste”. Este se produce cuando el modelo no es capaz de identificar patrones, obteniendo resultados erróneos en las predicciones.

Overfitting

En español se puede traducir como “sobreajuste”. Este se produce cuando el modelo sobreentrena con unos ciertos datos para los que se reconoce el resultado (normalmente el conjunto de entrenamiento), de esta forma el algoritmo queda ajustado a unas características muy específicas de los datos con los que ha entrenado, obteniendo errores en el momento que intente predecir datos fuera de ese conjunto.

Frontend y Backend:

Frontend es la parte de la aplicación que interactúa con el usuario. Es todo aquello que se ve por pantalla, previamente creado por el programador, de esta forma se obtiene una interfaz que interactúa con el cliente

Backend es la parte que está en el servidor, a la cual no puede acceder el usuario final y está destinada a hacer diversos cálculos, operaciones o acciones con el fin de comunicarlo al Frontend y este al usuario.

Herramientas CASE

Las herramientas CASE (*Computer Aided Software Engineering*, o Ingeniería de Software Asistida por Computadoras) son aplicaciones informáticas destinadas a ayudar y aumentar la productividad en el Desarrollo Software reduciendo el coste de tiempo y

dinero de esta. Son de gran utilidad en todos los aspectos de ciclo de vida de desarrollo del software, en tareas como el diseño de proyectos, cálculo de costes, documentación o detección de errores entre otras.

Bootstrap

Son un conjunto de herramientas de código abierto para el diseño de sitios y aplicaciones web. Contiene diversas plantillas de diseño con diversos elementos basado en HTML y CSS.

4. Técnicas y herramientas usadas

4.1 Herramientas y entornos de desarrollo

Visual Studio Code:

Visual Studio Code es un editor de código desarrollado por Microsoft. Se uso este por la gran experiencia en el desarrollo de aplicaciones con él a lo largo de la carrera. Fue usado en la creación de las páginas HTML y pequeños cambios en la aplicación por la rapidez y fluidez de este, además de consumir menos recursos que PyCharm, por lo que estos cambios se hacían más rápidos. [6]

La interfaz de este programa se pude ver en la "Ilustración 6: Visual Studio Code".

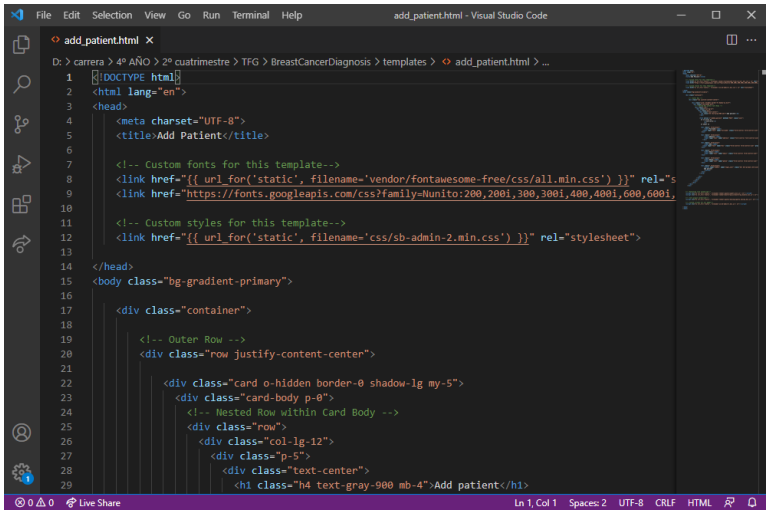


Ilustración 6: Visual Studio Code

PyCharm:

PyCharm es un entorno de desarrollo integrado (IDE) utilizado en programación, específicamente para el lenguaje Python. Está desarrollado por la empresa JetBrains. [7]

Se ha utilizado este IDE debido a la facilidad de uso de este, en la creación de la aplicación web, además de la creación de un programa para ordenar el banco de imágenes, puesto que estas no estaban ordenadas en las carpetas validation, training y test, usadas para el entrenamiento de la IA.

En la "Ilustración 7: PyCharm", se puede observar el aspecto de este IDE.

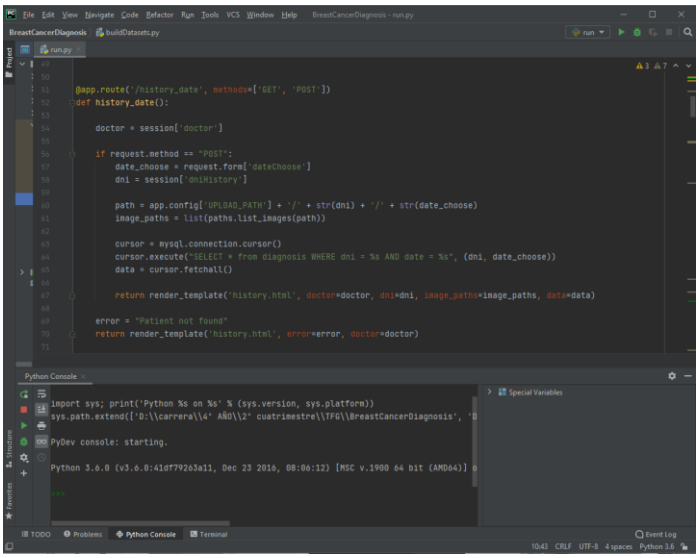


Ilustración 7: PyCharm

Google Colaboratory:

Colaboratory, también llamado “Colab”, permite ejecutar y programar en Python a través del navegador con las ventajas de que no se requiere una configuración previa, se tiene acceso gratuito a GPUs para el entrenamiento de la IA, además de permitir compartir contenido fácilmente. [8]

Ha sido usado en el desarrollo del proyecto para el entrenamiento de la Inteligencia Artificial, puesto que se requiere gran cantidad de recursos para ello y las limitaciones del equipo personal no hacían posible este entrenamiento, por ello, se vio necesario el uso de esta plataforma, ya que el acceso a GPUs hacía posible que la cantidad de tiempo requerido disminuyese drásticamente.

En la “Ilustración 8: Colaboratory” se ve el aspecto que presenta está página web.

Ilustración 8: Colaboratory

Google Drive:

Google Drive es un servicio de Google para el almacenamiento de ficheros en la nube. Además, tiene la posibilidad de compartir los archivos con Colaboratory para la ejecución de los distintos archivos. [9]

Durante el proyecto fue usado para el almacenamiento de versiones de los archivos del proyecto a modo de copias de seguridad y para compartir los archivos Python para el entrenamiento de la IA con Colaboratory.

XAMPP:

Es un paquete de software que sirve principalmente para la gestión de bases de datos MySQL y para el servidor web Apache. [10]

En el proyecto fue utilizado para crear y gestionar la base de datos de la aplicación, ya que ofrece la herramienta phpMyAdmin con la cual se puede administrar bases de datos MySQL. En la "Ilustración 9: XAMPP" se puede ver el aspecto de esta aplicación.

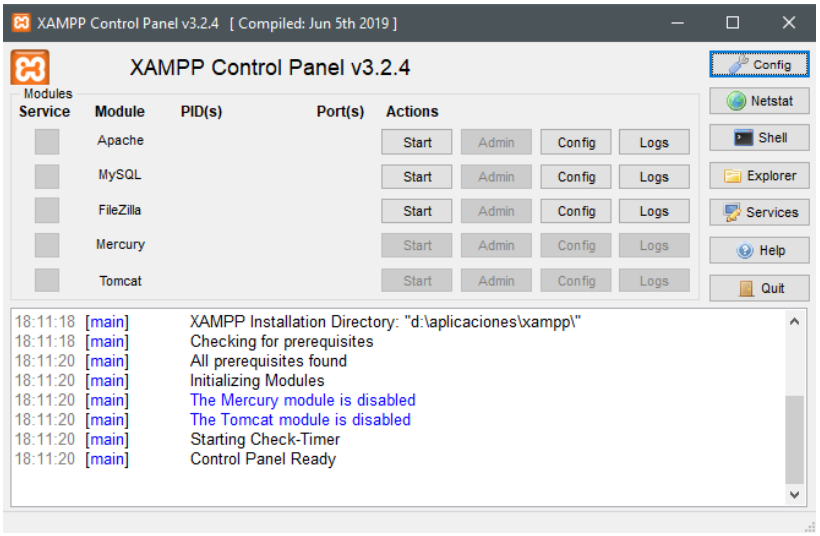


Ilustración 9: XAMPP

Pydoc:

Pydoc es un módulo de documentación estándar para el lenguaje de programación Python. Permite a los programadores acceder a los archivos de ayuda de la documentación de Python, generar texto y páginas HTML con información específica de la documentación. [11]

4.2 Herramientas CASE

Visual Paradigm:

Visual Paradigm, el cual se puede ver en la "Ilustración 10: Visual Paradigm", es un programa que proporciona un conjunto de ayudas para el desarrollo de programas informáticos, realizando distintos diagramas del sistema. [12]

Durante el proyecto ha acogido la tarea de generar todos los diagramas.

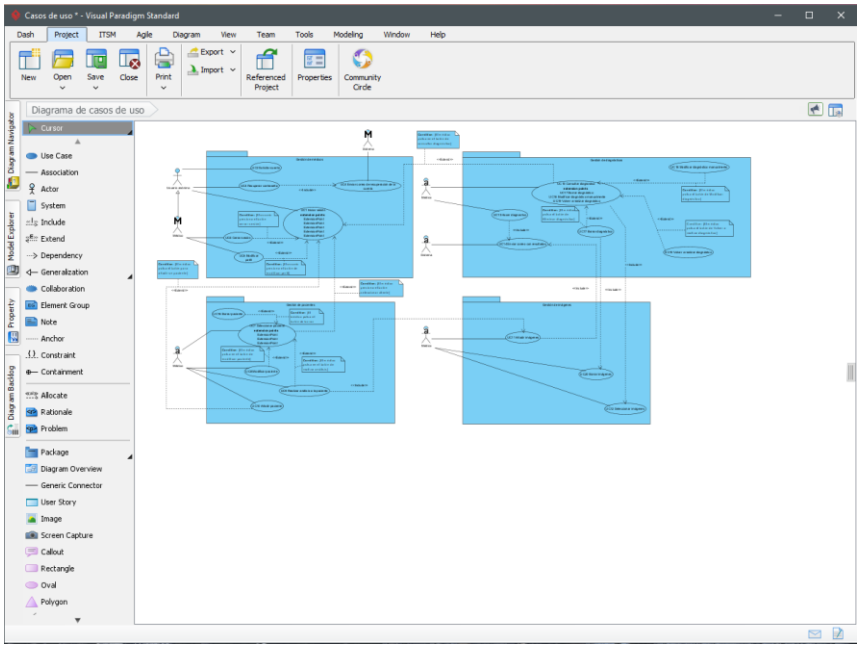


Ilustración 10: Visual Paradigm

EZ Estimate:

EZ Estimate, el cual se puede ver en la "Ilustración 11: EZ Estimate", es una herramienta de estimación del coste y esfuerzo del desarrollo del sistema usando el método UCP (Use Case Point) a partir de los casos de uso.

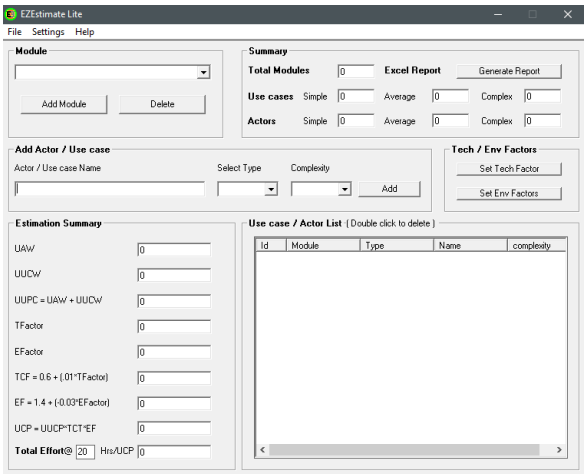


Ilustración 11: EZ Estimate

Microsoft Project:

Se trata de un software que ayuda a administrar proyectos en el desarrollo de planes, asignación de recursos a tareas, tener un seguimiento del progreso, administrar presupuestos y analizar cargas de trabajo. En la "Ilustración 12: Microsoft Project" se puede ver la interfaz de este software. [13]

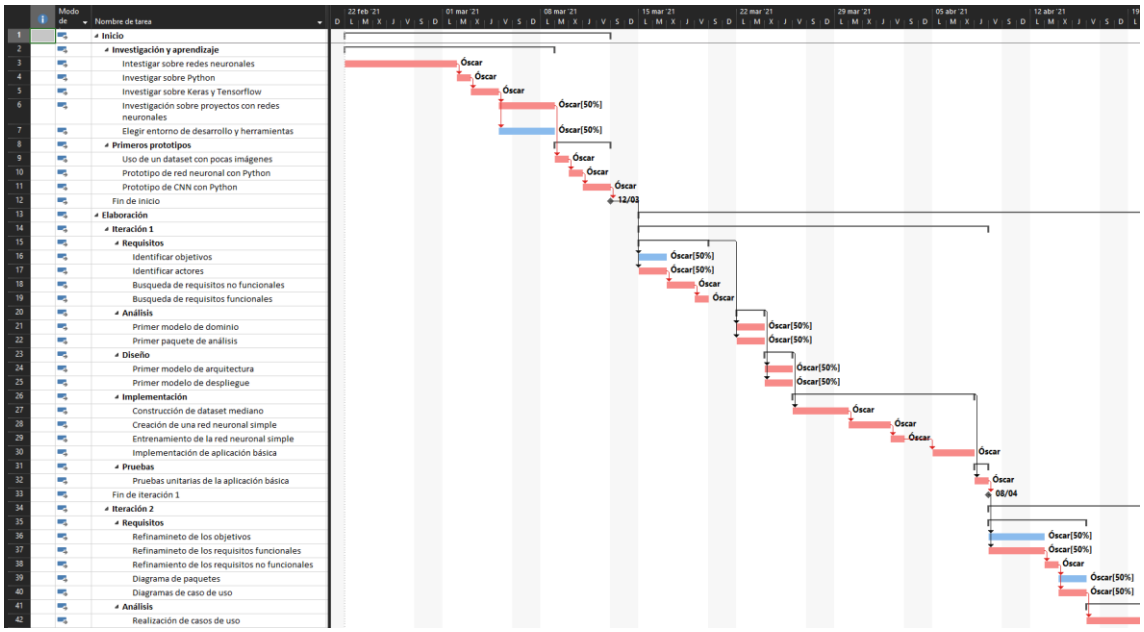


Ilustración 12: Microsoft Project

4.3 Framework

Flask:

Flask es un framework de Python que permite crear páginas web y servicios REST de una manera cómoda y sencilla. [14]

Se ha utilizado este framework en el desarrollo del proyecto para la creación de la aplicación web.

4.4 Lenguajes

Python:

Python es un lenguaje de programación de código abierto, cuya filosofía hace hincapié en la legibilidad del código. Tiene una sintaxis sencilla, pero con un gran potencial para el diseño de aplicaciones. [15]

Los principales usos para los que está destinado son:

- Inteligencia artificial
- Big Data
- Data Science
- Desarrollo de aplicaciones web

El lenguaje Python ha sido utilizado durante el proyecto para la ceración de la Inteligencia Artificial, de la aplicación web, así como para el manejo de las imágenes del Dataset, ya que el banco de imágenes no estaba ordenado, puesto que no tenía las carpetas validation, training y test, necesarias para entrenar a la inteligencia artificial.

HTML:

HTML (HyperText Markup Language o Lenguaje de marcado de hipertexto en español) es un lenguaje de marcado para la creación de páginas web.

Se ha usado para estructurar la vista de las páginas que pertenecen a la aplicación web.

CSS:

CSS (Cascading Style Sheets o Hojas de estilo en cascada en español) es un lenguaje de diseño gráfico que aporta el diseño a las páginas de HTML.

JavaScript:

JavaScript es un lenguaje de programación que funciona en los navegadores de forma nativa, es decir, sin necesidad de compilación. Por tanto, se utiliza como complemento a HTML y CSS para la creación de las páginas web.

Se ha usado en las páginas web, ya que el Bootstrap utilizado, que se comentará más adelante, tiene implementada funcionalidad con este lenguaje. De esta forma las páginas por las que navega el usuario poseen elementos, como ventanas modales y desplegables entre otros, que sin este lenguaje no habría sido posible.

Además de en el Bootstrap, se utilizó JavaScript para recoger las imágenes que introduce el médico. Esto es gracias a dropzone.js, una librería para subir imágenes a un servidor.

4.5 Bibliotecas de Python para la IA

Las bibliotecas más destacables utilizadas en el proyecto están destinadas a la implementación de la inteligencia artificial creada para la detección del cáncer de mama.

A continuación, se comentan cada una de ellas.

Tensorflow:

Es una biblioteca de código abierto para el aprendizaje automático, capaz de construir y entrenar redes neuronales para detectar patrones y correlaciones, de una manera análoga a como lo haría el razonamiento humano. [16]

Keras:

Es una biblioteca de Redes Neuronales de código abierto escrita en Python. La gran ventaja de su uso es que es capaz de ejecutarse sobre TensorFlow, entre otros. Su objetivo es el desarrollo y experimentación con redes de *Deep Learning* o Aprendizaje profundo. [17]

Scikit-learn:

Es una herramienta destinada a el aprendizaje automático. Está programado para el lenguaje de programación en Python. Incluye algoritmos para la clasificación, regresión y análisis de grupos.

Imutils:

Es un conjunto de funciones con el fin de realizar tareas en el procesamiento de imágenes, de una manera sencilla. Usado normalmente para el entrenamiento de una Inteligencia Artificial a partir de imágenes.

4.6 Bootstrap

SB Admin 2:

SB Admin es una plantilla de administración de Bootstrap gratuita, de código abierto y con licencia del MIT. Esta plantilla usa estilos predeterminados de Bootstrap 5 que junto con complementos permite crear aplicaciones web. [18]

Fue utilizado para ofrecer el diseño de la aplicación web del proyecto, se puede visualizar su aspecto en “Ilustración 13: SB Admin 2”.

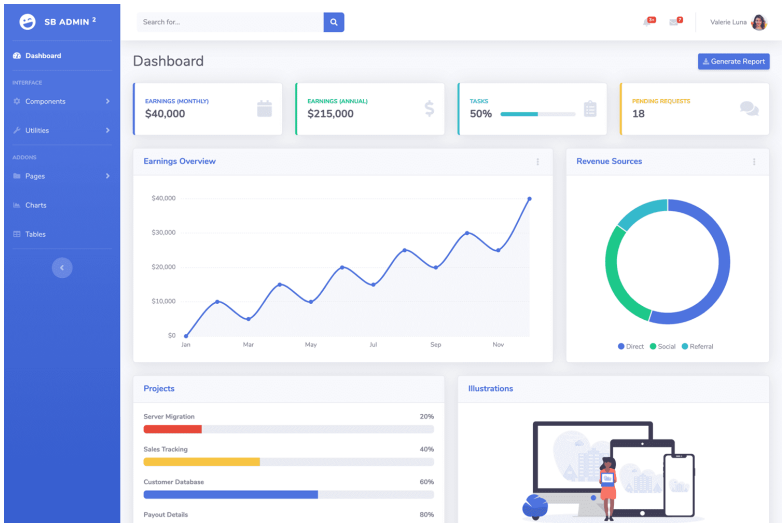


Ilustración 13: SB Admin 2

4.7 Datasets

Como ya se sabe, un datasets es una colección de datos o imágenes que tiene la finalidad de entrenar a una red neuronal. Por lo tanto, para el correcto funcionamiento de la Inteligencia Artificial del proyecto es necesario usar un banco de imágenes.

Por ello se investigó en la página web Kaggle, una comunidad en línea de científicos de datos y profesionales de aprendizaje automático. En ella se encuentran diversos datasets, entre ellos un banco de imágenes de cáncer de mama. [19]

Este dataset contenía diversos directorios con un id y dentro de estas un subdirectorio con el nombre de 0 para imágenes que no tenían esta enfermedad y 1 para las imágenes que si tenían. Se creó un programa en Python (buildDatasets.py) para organizar estos directorios y subdirectorios en tres carpetas (test, training, validation), dentro de estas a su vez los mencionados subdirectorios 0 y 1.

De esta forma la Inteligencia artificial podía entrenar de la siguiente forma:

Test (Prueba):

Conjunto de imágenes usado para evaluar el rendimiento de la red neuronal. No se usa para entrenar o para ajustar ningún parámetro.

Training (Entrenamiento):

Conjunto de imágenes usadas de ejemplo durante el proceso de entrenamiento. Utilizado para ajustar los parámetros de la red neuronal. Además, suele ser el conjunto más grande de los tres.

Validation (Validación):

Conjunto de datos que se utilizan para ajustar los hiperparámetros.

Para el entrenamiento de la red neuronal, se ha elegido usar el 80% de los datos para el entrenamiento, el 18% de los datos para validación y el 2% para las pruebas. Estos porcentajes pueden variar conforme se quiera que dar un mayor énfasis al entrenamiento, ajustando de esta forma los parámetros de la red neuronal, o a la validación, ajustando así los hiperparámetros o tener un conjunto más grande de pruebas, dándole así un mayor porcentaje al conjunto de *test*. Por regla general se debe respetar que el conjunto de entrenamiento sea el más grande de los tres, ya que gracias a él se configura la red neuronal.

Como se puede observar en la “Ilustración 14: Conjunto de entrenamiento, validación y prueba”. El conjunto de entrenamiento suele ser el más grande de los tres, mientras que

el de validación y pruebas son más pequeños. La elección de estos tamaños queda en manos del programador.

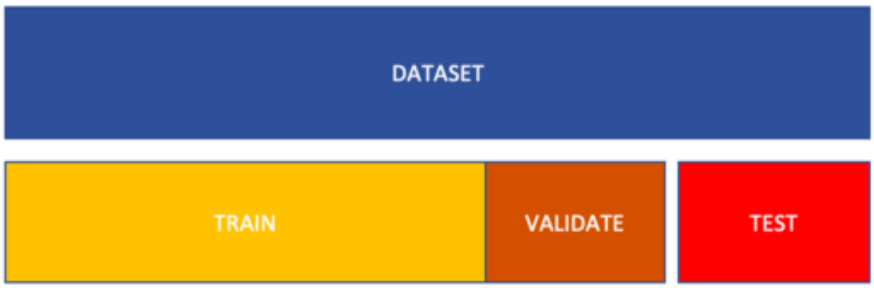


Ilustración 14: Conjunto de entrenamiento, validación y prueba

5. Aspectos relevantes del desarrollo

En este apartado se detallan los aspectos más relevantes de las distintas fases del desarrollo del proyecto.

5.1 Marco de trabajo

Para el desarrollo de este trabajo de fin de grado se ha utilizado el Proceso Unificado como marco de trabajo. El cuál tiene las siguientes características:

- **Dirigido por casos de uso**: Estos son utilizados para describir la funcionalidad el sistema y sirven como marco para realizar las iteraciones. Además, los casos de uso se utilizan para analizar los requisitos del sistema y para obtener una base sobre la que se desarrollará todo el proceso unificado.

- **Centrado en la arquitectura**: Esta define como se estructura el sistema, ofreciendo así una realización de todos los casos de uso. Está puede variar conforme se desarrolla el proyecto para adaptarse a los casos de uso.

- **Iterativo e incremental:** Como el desarrollo de un proyecto conlleva mucho tiempo, siendo una tarea larga y costosa, es necesario dividirlo en distintas iteraciones. En cada una de estas se desarrolla parte del sistema, cumpliendo de esta forma unos objetivos y unos casos de uso. Al finalizar cada iteración se revisa el proyecto, marcando un hito con una versión entregable del producto.

El Proceso Unificado tiene distintas fases (inicio, elaboración, construcción y transición), en las cuales se puede tener varias iteraciones. En cada iteración se tendrán distintas tareas centradas en las disciplinas que ofrece el Proceso Unificado (modelado de negocio, requisitos, análisis, diseño, implementación y pruebas). Todo esto se puede ver en la “Ilustración 15: Proceso unificado”.

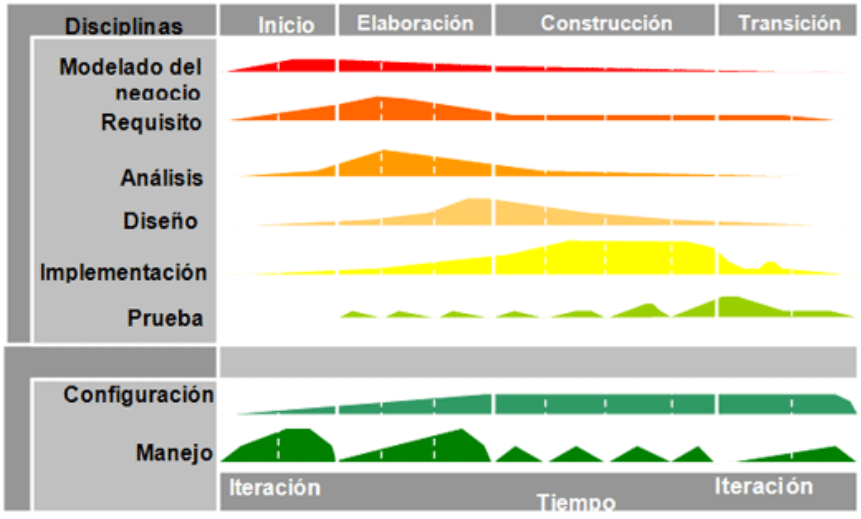


Ilustración 15: Proceso unificado

5.2 Estimación de coste y esfuerzo

Gracias a la estimación de coste y esfuerzo se puede calcular una predicción del tamaño del software, así como del tiempo que se tardará en desarrollarlo. Es por ello por lo que se ha usado el método UCP (puntos de caso de uso) ya que es con el que se ha trabajado en distintas asignaturas a lo largo de la carrera. Este método mide la funcionalidad representada por los casos de uso. Al finalizar con la estimación, se obtendrá de una manera aproximada el esfuerzo total del desarrollo expresado en horas de persona.

Para calcular los puntos de caso de uso (UCP), es necesario haber calculado primero los puntos de caso de uso sin ajustar (UUCP), el factor de complejidad técnica (TCF) y el factor de complejidad de entorno (ECF). Este cálculo se consigue haciendo uso de la siguiente fórmula:

$$UCP = UUCP * TCF * ECF$$

5.2.1 Cálculo de los UUCP

Los puntos de caso de uso sin ajustar (UUCP) se calculan a partir del peso de los casos de uso sin ajustar (UUCW) y de los pesos de los actores sin ajustar (UAW) con el uso de la siguiente fórmula:

$$UUCP = UAW * UUCW$$

Los UUCW son el sumatorio de los pesos asignados a cada caso de uso en función de su complejidad:

$$UUCW = \sum peso_i$$

Igual que para los UUCW, los UAW se calculan con el sumatorio de los pesos asignados a cada actor en función de la complejidad que desempeña el sistema:

$$UAW = \sum peso_i$$

Actores del sistema:

Los actores del sistema se pueden consultar en la "Tabla 1: Complejidad de los actores del sistema".

Actor	Complejidad	Observaciones
ACT-0001 Usuario anónimo	Complejo	Contempla un uso con la interfaz gráfica de la aplicación.
ACT-0002 Médico	Complejo	Al igual que el anónimo, usa la interfaz gráfica de la aplicación.
ACT-0003 Sistema	Medio	Se trata de un actor encargado de enviar correos electrónicos.

Tabla 1: Complejidad de los actores del sistema

Los distintos casos de uso se pueden consultar en la “Tabla 2: Complejidad Gestión de médicos”, la “Tabla 3: Complejidad Gestión de pacientes”, la “Tabla 4: Complejidad Gestión de diagnósticos” y la “Tabla 5: Complejidad Gestión de imágenes”.

Gestión de médicos

Caso de uso	Número de transacciones	Complejidad
UC-0001 Iniciar sesión	1	Simple
UC-0002 Solicitar cuenta	1	Simple
UC-0003 Recuperar contraseña	1	Simple
UC-0004 Enviar correo electrónico de recuperación de la cuenta	1	Simple
UC-0005 Cerrar sesión	2	Simple
UC-0006 Modificar perfil	2	Simple

Tabla 2: Complejidad Gestión de médicos

Gestión de pacientes

Caso de uso	Número de transacciones	Complejidad
UC-0007 Seleccionar paciente	3	Simple
UC-0008 Modificar paciente	2	Simple
UC-0009 Realizar análisis a la paciente	1	Simple
UC-0010 Añadir paciente	2	Simple
UC-0016 Borrar paciente	2	Simple

Tabla 3: Complejidad Gestión de pacientes

Gestión de diagnósticos

Caso de uso	Número de transacciones	Complejidad
UC-0013 Hacer diagnóstico	2	Simple
UC-0014 Enviar correo con resultados	1	Simple
UC-0015 Consultar diagnóstico	2	Simple
UC-0017 Borrar diagnóstico	2	Simple
UC-0018 Modificar diagnóstico manualmente	2	Simple
UC-0019 Volver a realizar diagnóstico	3	Simple

Tabla 4: Complejidad Gestión de diagnósticos

Gestión de imágenes

Caso de uso	Número de transacciones	Complejidad
UC-0011 Añadir imágenes	2	Simple
UC-0012 Seleccionar imágenes	1	Simple
UC-0020 Borrar imágenes	1	Simple

Tabla 5: Complejidad Gestión de imágenes

5.2.2 Cálculo de los TCF

Para calcular los factores de complejidad técnica (TCF) se hace uso de la siguiente fórmula:

$$TCF = C_1 + C_2 * \sum (W_i * F_i)$$

En esta fórmula, C_1 y C_2 son constantes, W_i es el peso i-ésimo del factor de complejidad técnica y F_i es el factor de complejidad calculado en una escala de 0 a 5, indicando mayor irrelevancia cuanto menor es el número. Estos factores se pueden consultar en la "Tabla 6: Factores de complejidad técnica".

Factor	Complejidad	Explicación
E1 Familiaridad con UML	2	Los conocimientos en UML son limitados. La experiencia se debe a las clases y trabajos en las asignaturas de Ingeniería del Software.
E2 Trabajadores a tiempo parcial	3	Se compagina el desarrollo del trabajo final con el último año de carrera y con tareas fuera de la vida académica.
E3 Capacidad de los analistas	2	Similar a las razones de *E1: Familiaridad con UML.*
E4 Experiencia en la aplicación	3	Se han desarrollado pocas aplicaciones web antes, sumado al uso de nuevas tecnologías en las que no se tiene experiencia como Python, Keras y Tensorflow.
E5 Experiencia en orientación a objetos	4	La experiencia adquirida se debe a las distintas asignaturas de la carrera que tratan este tipo de programación.
E6 Motivación	5	Existe una gran motivación con respecto a este trabajo de fin de grado.
E7 Dificultad de lenguaje de programación	4	Python no es un lenguaje especialmente complejo.
E8 Estabilidad de los requisitos	4	Los requisitos son bastante estables y no se espera que cambien.

Tabla 6: Factores de complejidad técnica

5.2.3 Cálculo de los ECF

Para calcular los factores de complejidad de entorno (ECF) se haces uso de la siguiente fórmula:

$$ECF = C_1 + C_2 * \sum (W_i * F_i)$$

En esta fórmula, C_1 y C_2 son constantes, W_i es el peso i-ésimo del factor de complejidad de entrono, y F_i es el factor de complejidad calculado en una escala de 0 a 5, indicando mayor irrelevancia cuanto menor es el número. Estos factores se pueden consultar en la "Tabla 7: Factores de complejidad de entorno".

Factor	Complejidad	Explicación
E1 Familiaridad con UML	2	Los conocimientos en UML son limitados. La experiencia se debe a las clases y trabajos en las asignaturas de Ingeniería del Software.
E2 Trabajadores a tiempo parcial	3	Se compagina el desarrollo del trabajo final con el último año de carrera y con tareas fuera de la vida académica.
E3 Capacidad de los analistas	2	Similar a las razones de *E1: Familiaridad con UML.*
E4 Experiencia en la aplicación	3	Se han desarrollado pocas aplicaciones web antes, sumado al uso de nuevas tecnologías en las que no se tiene experiencia como Python, Keras y Tensorflow.
E5 Experiencia en orientación a objetos	4	La experiencia adquirida se debe a las distintas asignaturas de la carrera que tratan este tipo de programación.
E6 Motivación	5	Existe una gran motivación con respecto a este trabajo de fin de grado.
E7 Dificultad de lenguaje de programación	4	Python no es un lenguaje especialmente complejo.
E8 Estabilidad de los requisitos	4	Los requisitos son bastante estables y no se espera que cambien.

Tabla 7: Factores de complejidad de entorno

5.2.4 Resultados

Utilizando el programa EZEstimate, junto con los datos recopilados en los apartados anteriores, se han realizado los cálculos para obtener una estimación, que se puede ver en la "Ilustración 16: Resultados de estimación de coste y esfuerzo":

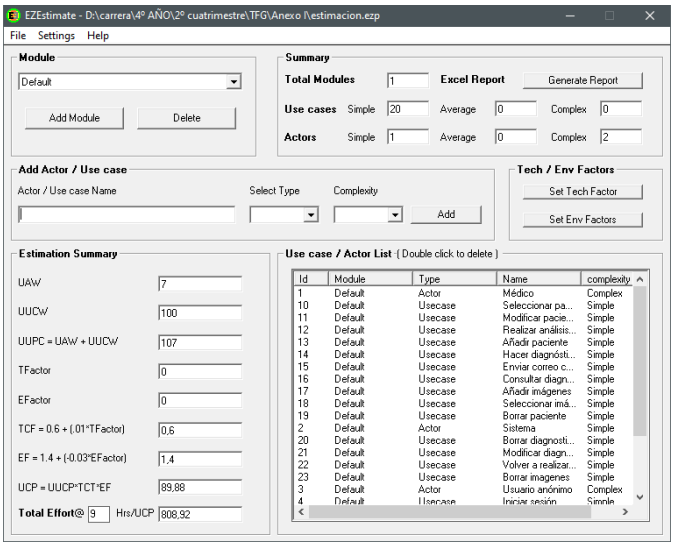


Ilustración 16: Resultados de estimación de coste y esfuerzo

5.3 Planificación temporal

En este apartado se detalla la división de las tareas del proyecto, así como de su estimación de los tiempos de inicio y fin de estas. Para realizar esta planificación se ha usado el software Microsoft Project.

En el *Anexo I – Planificación temporal* se recoge este apartado con un mayor detalle.

A continuación, se muestran las tareas del proyecto, junto a sus dependencias y duración en el diagrama de Gantt en la "Ilustración 17: Diagrama de Gantt parte 1", la "Ilustración 18: Diagrama de Gantt parte 2" y la "Ilustración 19: Diagrama de Gantt parte 3":

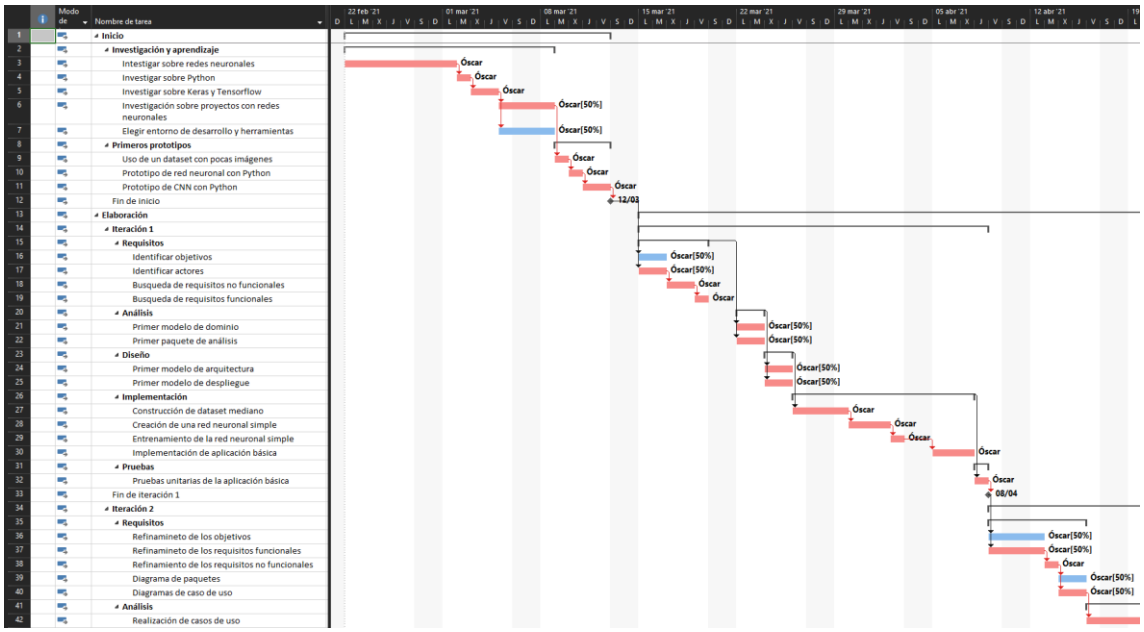


Ilustración 17: Diagrama de Gantt parte 1

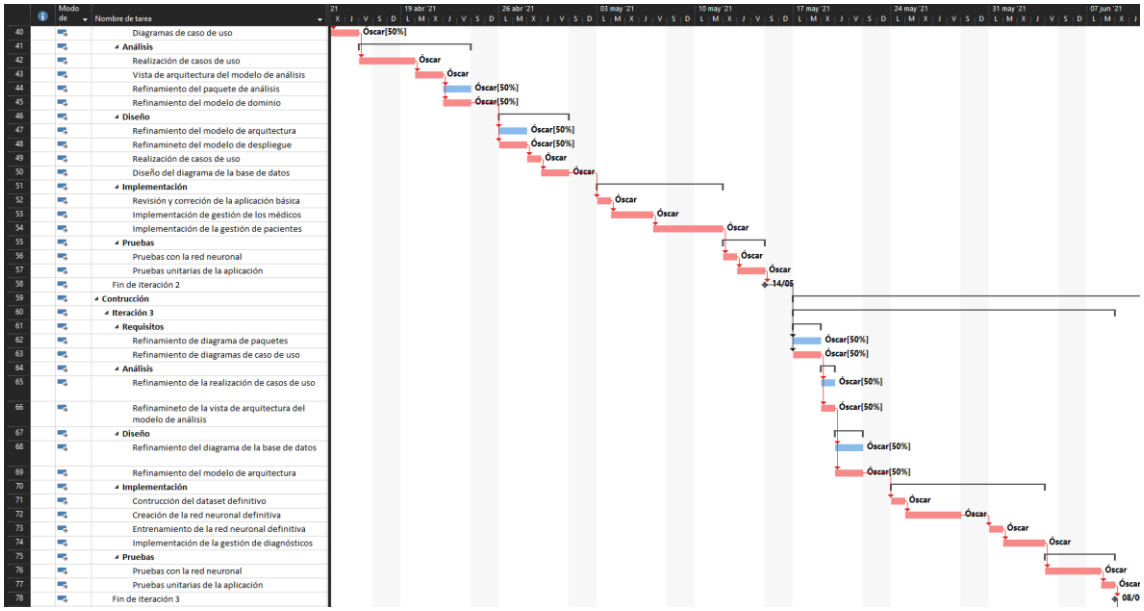


Ilustración 18: Diagrama de Gantt parte 2

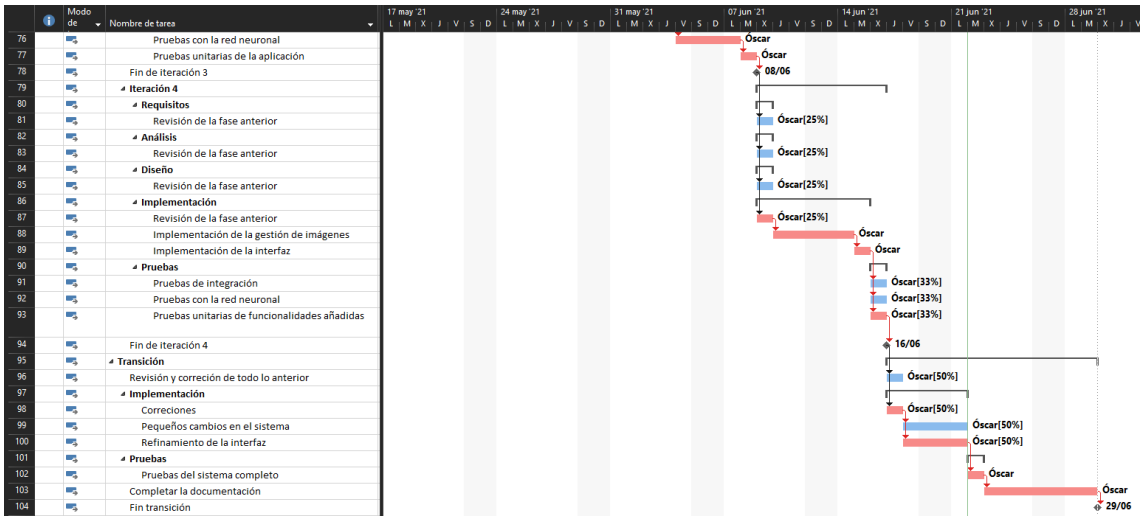


Ilustración 19: Diagrama de Gantt parte 3

5.4 Especificación de requisitos

En este apartado se van a exponer las especificaciones de requisitos software del proyecto.

Mediante la metodología de Duran y Bernárdez, se desarrolla la especificación de cada uno de los componentes del sistema, obteniendo así los objetivos, requisitos de información, requisitos de restricción de información, requisitos no funcionales, actores y casos de uso necesarios para la implementación del sistema.

En el *Anexo II – Especificación de requisitos* se puede consultar este apartado más al detalle.

5.4.1 Objetivos

Los objetivos que se deben implementar son los siguientes:

- Gestión de médicos
- Gestión de pacientes
- Gestión de diagnósticos
- Gestión de imágenes

En el *Anexo II – Especificación de requisitos* se encuentran desarrollados los distintos objetivos en tablas como la “Tabla 8: Objetivo Gestión de pacientes”.

OBJ-0002	Gestión de pacientes
Versión	1.0
Autores	Óscar Sánchez Barriga
Fuentes	

Descripción	El sistema deberá ser capaz de registrar pacientes en una base de datos y permitir editar esta información.
Subobjetivos	Ninguno
Importancia	Alta
Urgencia	Alta
Estado	Activo
Estabilidad	Alta
Comentarios	Ninguno

Tabla 8: Objetivo Gestión de pacientes

5.4.2 Requisitos de información

El almacenamiento de información es una tarea vital dentro de un proyecto, ya que se debe saber en todo momento donde y como se va a guardar determinada información referente a los médicos y pacientes implicados en el sistema.

El sistema debe seguir los siguientes requisitos de información:

- Datos de los médicos
- Datos de los pacientes
- Datos de los diagnósticos
- Datos de las imágenes

Estos requisitos están desarrollados en tablas como la "Tabla 9: Requisitos de información Datos de los médicos":

IRQ-0001	**Datos de los médicos**
Versión	1.0
Autores	Óscar Sánchez Barriga
Fuentes	
Dependencias	[OBJ-0001] Gestión de Médicos
Descripción	El sistema deberá almacenar la información correspondiente a los datos de cada médico.
Datos específicos	• Id • Contraseña • Nombre y apellidos • Nombre de usuario • Correo electrónico
Importancia	Vital
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 9: Requisitos de información Datos de los médicos

5.4.3 Requisitos de restricción de información

En ellos se explican las condiciones que debe cumplir la información almacenada.

El sistema tiene restricciones en los siguientes datos:

- Médicos
- Pacientes
- Diagnósticos
- Imágenes

Los requisitos de restricción de información se describen en tablas como la "Tabla 10: Requisitos de restricción Datos de los médicos"

CRQ-0001	Restricción de datos de los médicos
Versión	1.0
Autores	Óscar Sánchez Barriga
Fuentes	
Dependencias	[IRQ-0001] Datos de los Médicos
Descripción	La información almacenada en el sistema debe satisfacer la siguiente restricción: No puede haber dos médicos con el mismo ID, con el mismo nombre de usuario o con el mismo correo electrónico.
Importancia	Vital
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 10: Requisitos de restricción Datos de los médicos

5.4.4 Requisitos no funcionales

Los requisitos funcionales suelen estar ligados a la estructura de la aplicación o a la compatibilidad de esta con otros sistemas externos.

Los requisitos no funcionales son los siguientes:

- Escalabilidad
- Sistema multiplataforma
- Privacidad de los datos
- Seguridad de los datos
- Eficiencia
- Accesibilidad
- Interfaz gráfica
- Concurrencia
- Mantenibilidad

Los requisitos no funcionales se describen en tablas como la “Tabla 11: Requisitos no funcionales Escalabilidad".

NFR-0001	Escalabilidad
Versión	1.0
Autores	Óscar Sánchez Barriga
Dependencias	General
Descripción	El sistema debe ser capaz de adaptarse a la infraestructura de cualquier tamaño. Además, se podrán añadir nuevas funcionalidades con el crecimiento del sistema.
Importancia	Vital
Estado	Validado
Estabilidad	Alta

Tabla 11: Requisitos no funcionales Escalabilidad

5.4.5 Requisitos funcionales

Estos requisitos son los referentes a la funcionalidad y objetivos que la aplicación debe cumplir y tener implementado para cubrir todas las necesidades.

Actores

Los actores representan los usuarios que interactúan en el sistema se pueden observar en la “Ilustración 20: Diagrama de actores”.

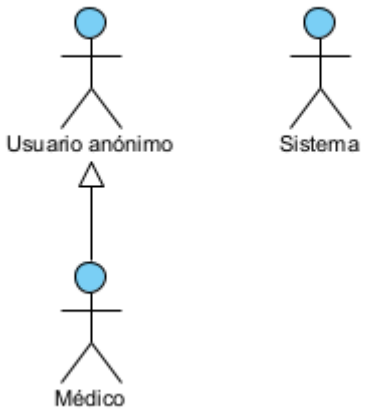


Ilustración 20: Diagrama de actores

Los actores del sistema se describen en tablas como la “Tabla 12: Actor médico".

ACT-0001	Usuario anónimo
Versión	1.0
Autores	Óscar Sánchez Barriga
Descripción	Este actor representa usuarios que no han iniciado sesión.
Comentarios	Ninguno

Tabla 12: Actor médico

Casos de uso

Los casos de uso recogen los objetivos funcionales del sistema, junto con los pasos a desarrollar por cada funcionalidad.

En la “Tabla 13: Caso de uso Iniciar sesión” se muestra la plantilla utilizada para la especificación de todos los casos de uso.

UC-0001	Iniciar sesión	
Versión	1.0	
Autores	Óscar Sánchez Barriga	
Fuentes		
Dependencias	Ninguna	
Descripción	El sistema deberá dar la opción de iniciar sesión a usuarios que todavía no esté identificados.	
Precondición	El usuario anónimo todavía no ha iniciado sesión	
Secuencia normal	**Paso**	**Acción**
	1	Se identifica al usuario anónimo.
Postcondición	Ninguna.	
Excepciones	**Paso**	**Acción**
	1	Si el usuario no existe, se realizará el caso de uso Solicitar cuenta (UC-0002)
Rendimiento	**Paso**	**Tiempo máximo**
	1	Indefinido
Frecuencia esperada	Alta	
Importancia	Alta	
Urgencia	Alta	
Estado	Activo	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 13: Caso de uso Iniciar sesión

En la “Ilustración 21: Diagrama de casos de uso” se puede observar el diagrama de casos de uso del sistema.

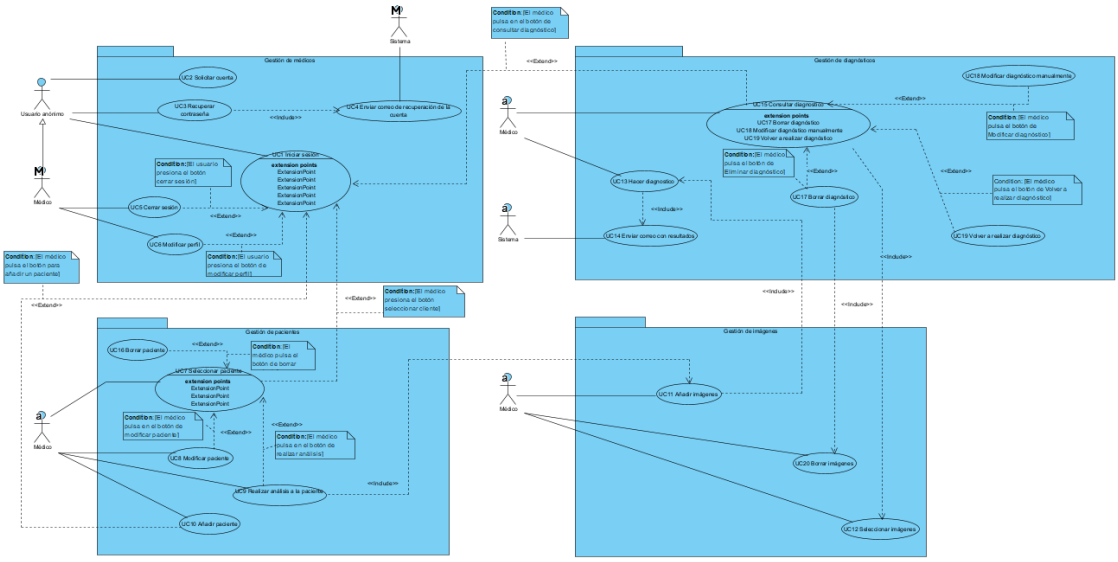


Ilustración 21: Diagrama de casos de uso

5.5 Análisis de sistema

A partir de la descripción de objetivos y requisitos del *Anexo II – Especificación de requisitos* se ha realizado el análisis de los requisitos del sistema.

Para ver más al detalle el análisis de requisitos del sistema se puede consultar el *Anexo III – Análisis de requisitos*.

5.5.1 Modelo de dominio

El modelo de dominio, que se puede ver en la “Ilustración 22: Modelo de dominio”, trata de mostrar la representación de los conceptos del mundo real que manejará la aplicación, junto a sus relaciones.

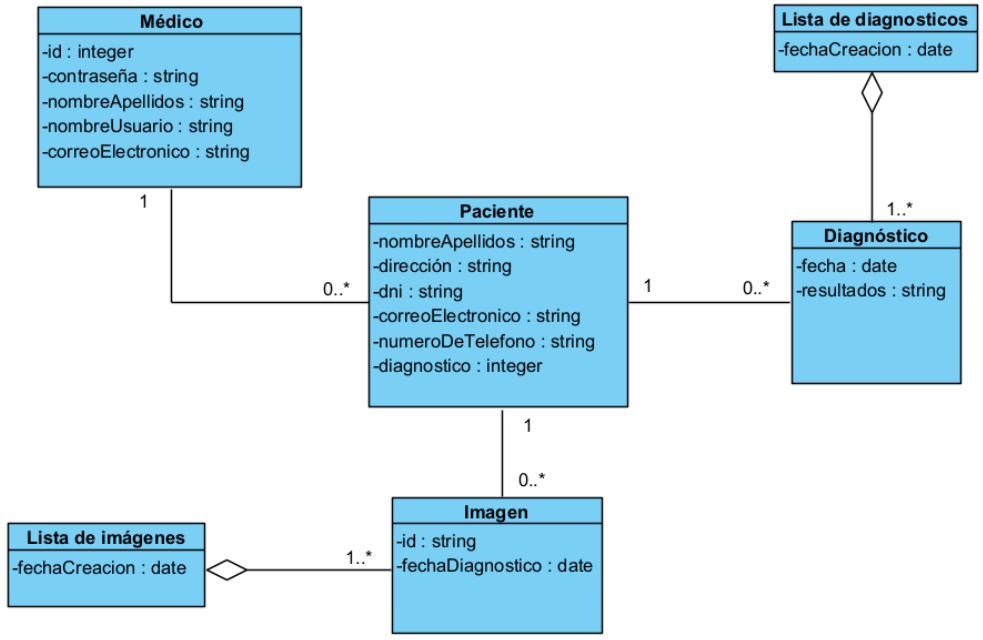


Ilustración 22: Modelo de dominio

5.5.2 Paquete de análisis

En este apartado se muestra una división de la aplicación en distintos paquetes.

- Gestión de médicos: Recoge toda la funcionalidad asociada al inicio de sesión y modificación de datos de los médicos, que son los usuarios en este sistema.
- Gestión de pacientes: Realiza las tareas relacionadas con el manejo de información de los pacientes.
- Gestión de diagnósticos: Realiza las tareas relacionadas con el manejo y creación de los distintos diagnósticos que realizará la Inteligencia Artificial.
- Gestión de imágenes: Recoge toda la funcionalidad asociada a la administración de las imágenes de histología mamaria.

En la “Ilustración 23: Diagrama de paquetes de análisis” se puede observar el diagrama que tendrá el sistema.

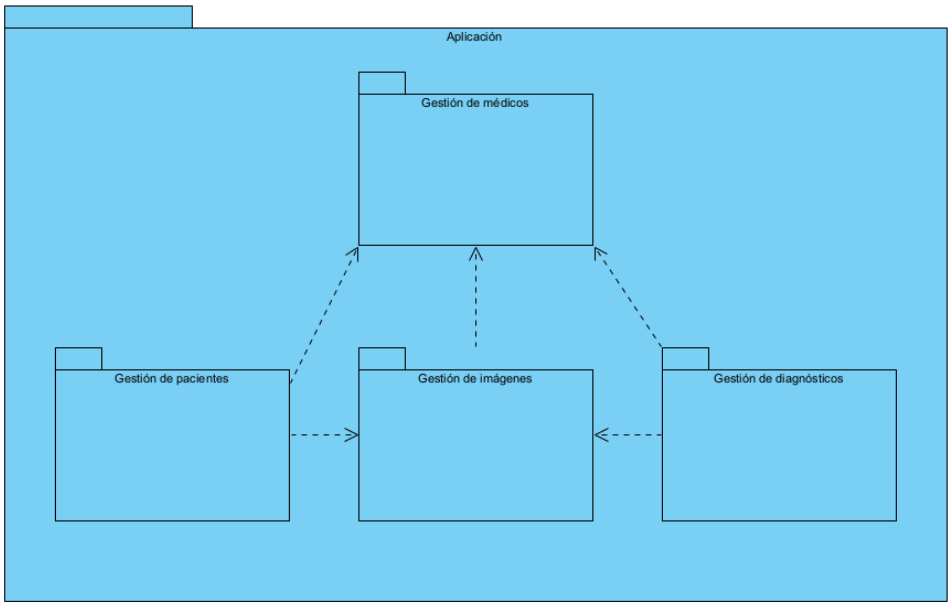


Ilustración 23: Diagrama de paquetes de análisis

5.5.3 Vista arquitectónica

A continuación, en la Ilustración 24: Vista de arquitectura del modelo de análisis", se detalla una primera versión de la arquitectura del sistema, que se detallará en mayor profundidad en el *Anexo IV – Diseño del sistema.*

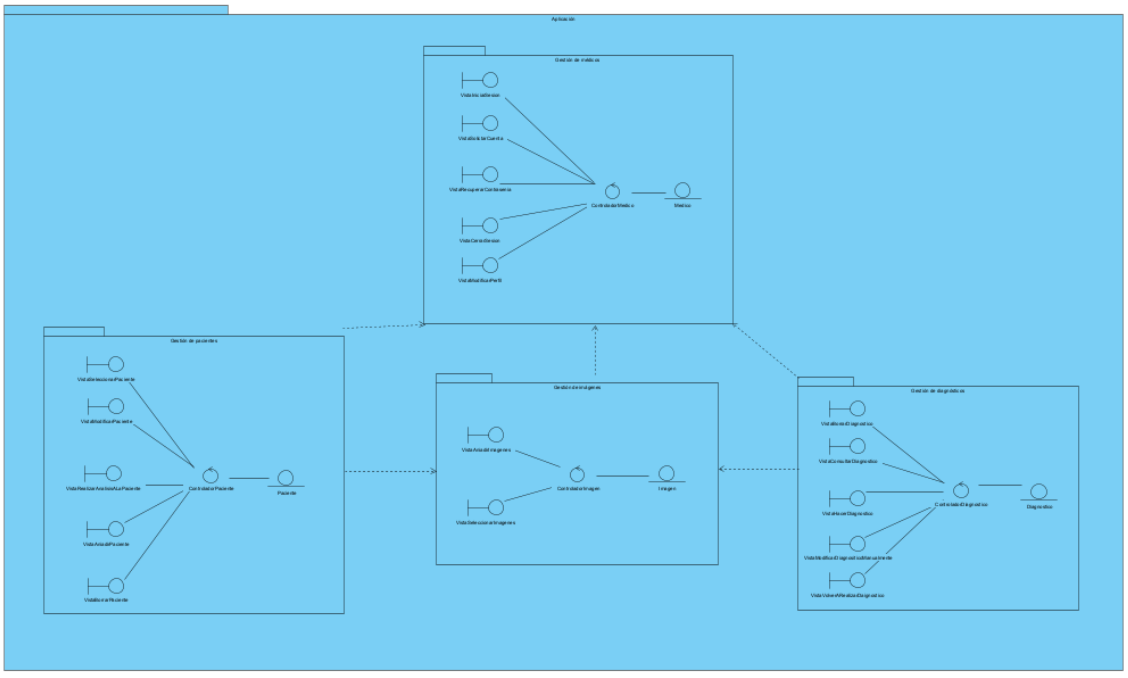


Ilustración 24: Vista de arquitectura del modelo de análisis

5.5.4 Realización de los casos de uso

En los diagramas de secuencia se puede ver como interactúan los objetivos entre sí, indicando los mensajes entre los componentes del sistema. En la "Ilustración 25: Realización del UC1 Iniciar sesión" se puede ver un ejemplo de los diagramas de secuencia de la aplicación.

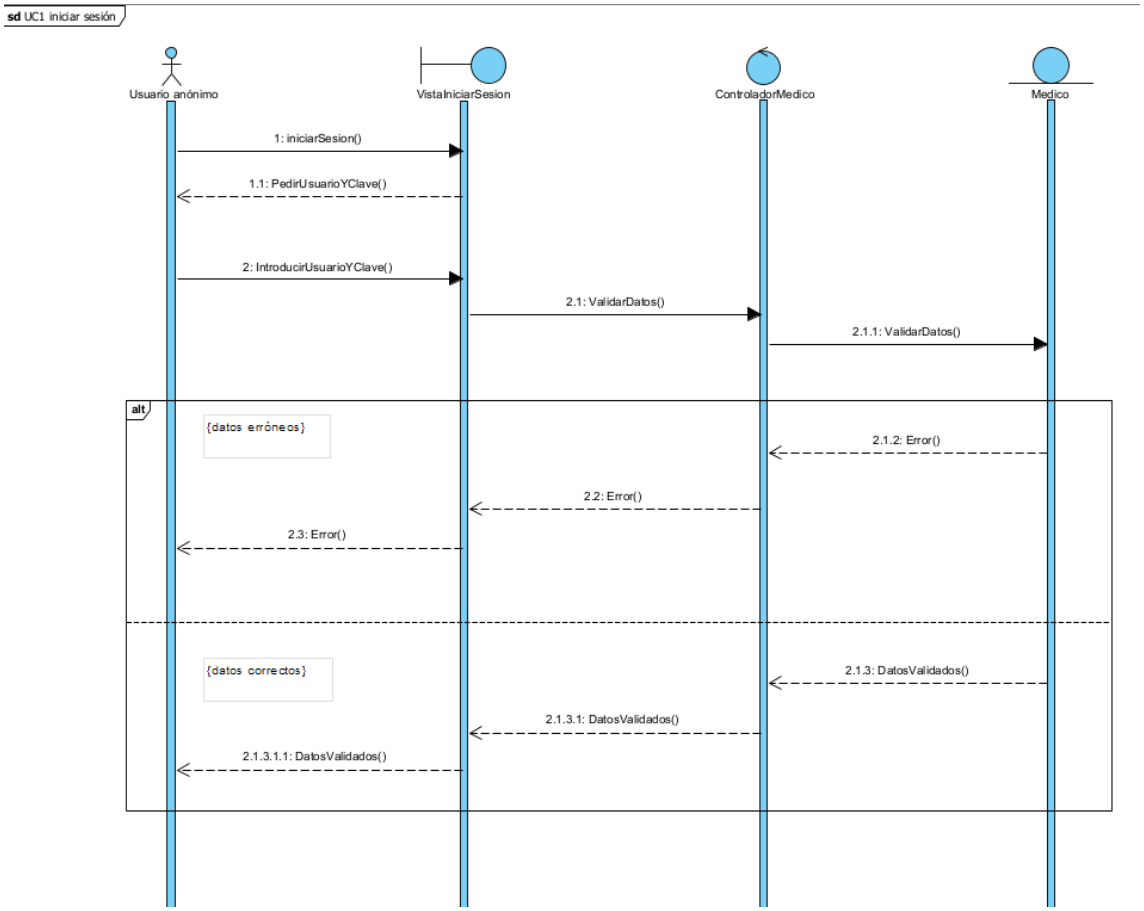


Ilustración 25: Realización del UC1 Iniciar sesión

5.6 Diseño del sistema

En la fase de diseño del sistema se pretende buscar una solución para implementar el proyecto. Es por ello por lo que en este apartado se detalla una posible solución para el sistema.

Para ver más al detalle el diseño del sistema se puede consultar el *Anexo IV – Diseño del sistema.*

5.6.1 Patrones arquitectónicos

Para el desarrollo del proyecto se ha utilizado el patrón MVC (Modelo Vista Controlador). Este es un tipo de patrón de arquitectura software que se basa en las ideas

de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

Los componentes de este patrón son los siguientes:

- **Modelo**: Representa la información con la cual el sistema trabaja. Envía a la 'Vista' aquella información que se solicita en cada momento. Las peticiones de acceso y manipulación de la información del 'Modelo' por la 'Vista' se hacen a través del 'Controlador'.
- **Controlador**: Responde a eventos y acciones del usuario e invoca peticiones al 'Modelo' cuando se solicita cierta información que este posee.
- **Vista**: Se encarga de representar y estructurar la información que el usuario verá por pantalla.

En la "Ilustración 26: Arquitectura MVC" se puede apreciar cómo trabajan y se relacionan estos tres componentes entre ellos. Gracias a esta arquitectura se facilita el manejo de errores, permite al sistema ser escalable y es posible agregar múltiples representaciones de datos.

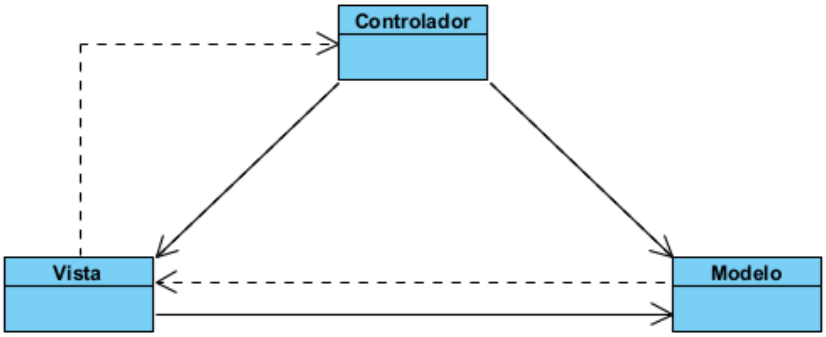


Ilustración 26: Arquitectura MVC

5.6.2 Subsistemas de diseño

En los diagramas que se muestran en la "Ilustración 27: Subsistema de vistas" y en la "Ilustración 28: Subsistema de controladores" se muestra la división de los diferentes componentes y la relación entre ellos.

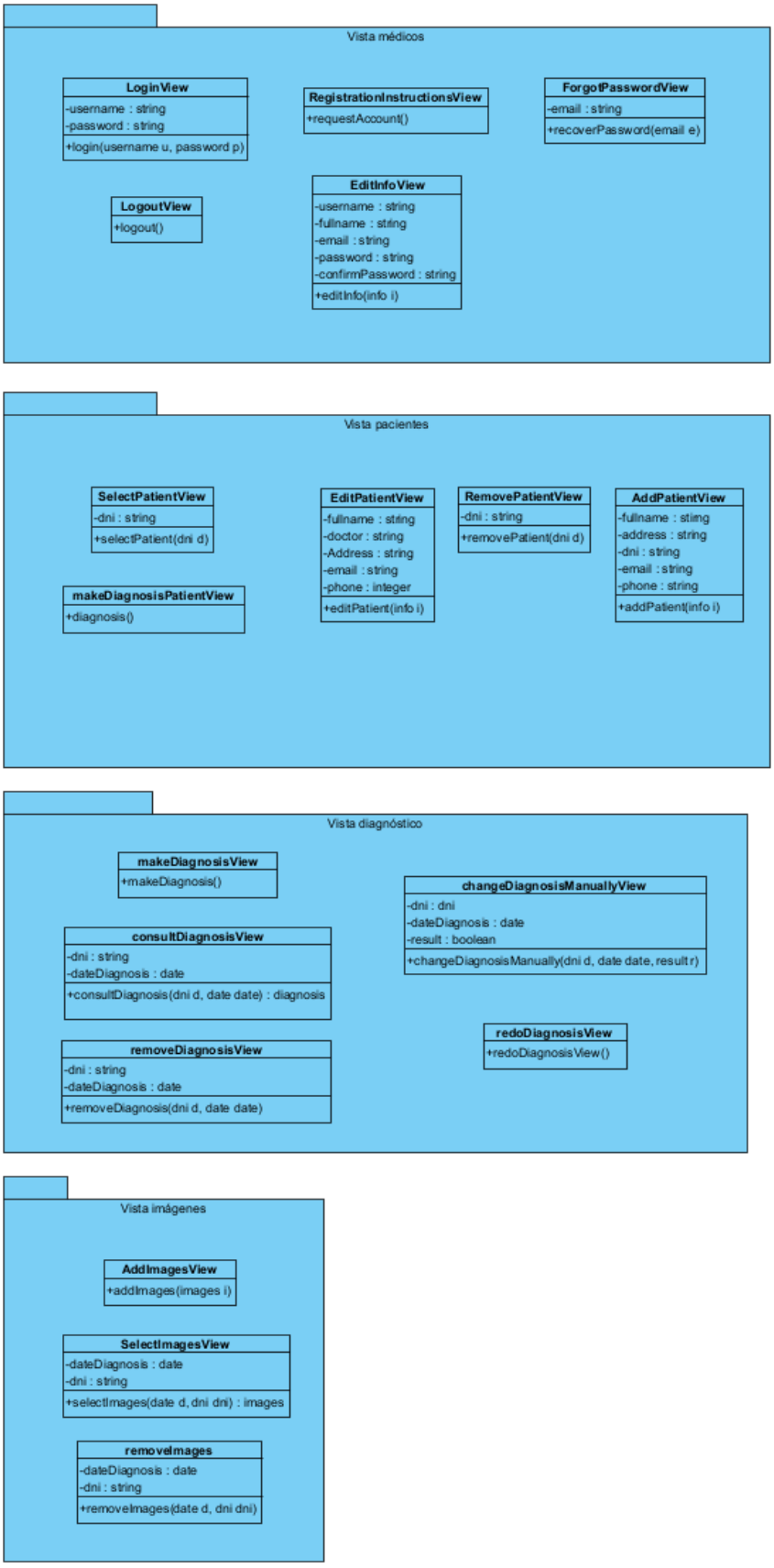


Ilustración 27: Subsistema de vistas

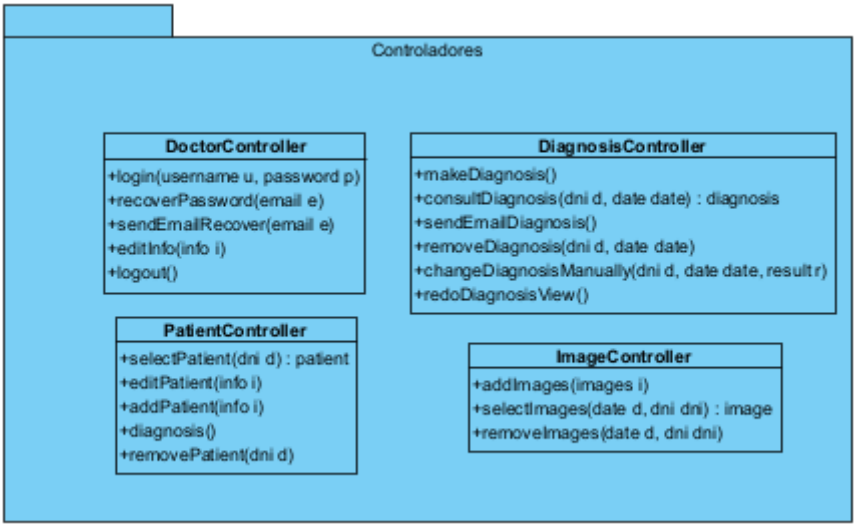


Ilustración 28: Subsistema de controladores

5.6.3 Clases de diseño

Se trata de la evolución del modelo de dominio del *Anexo III – Análisis de requisitos*. Este cambio se debe a la inclusión de atributos y operaciones que tendrán las clases en el sistema.

En la "Ilustración 29: Diagrama de clases del modelo de diseño" se puede ver la estructura que tendrá el sistema.

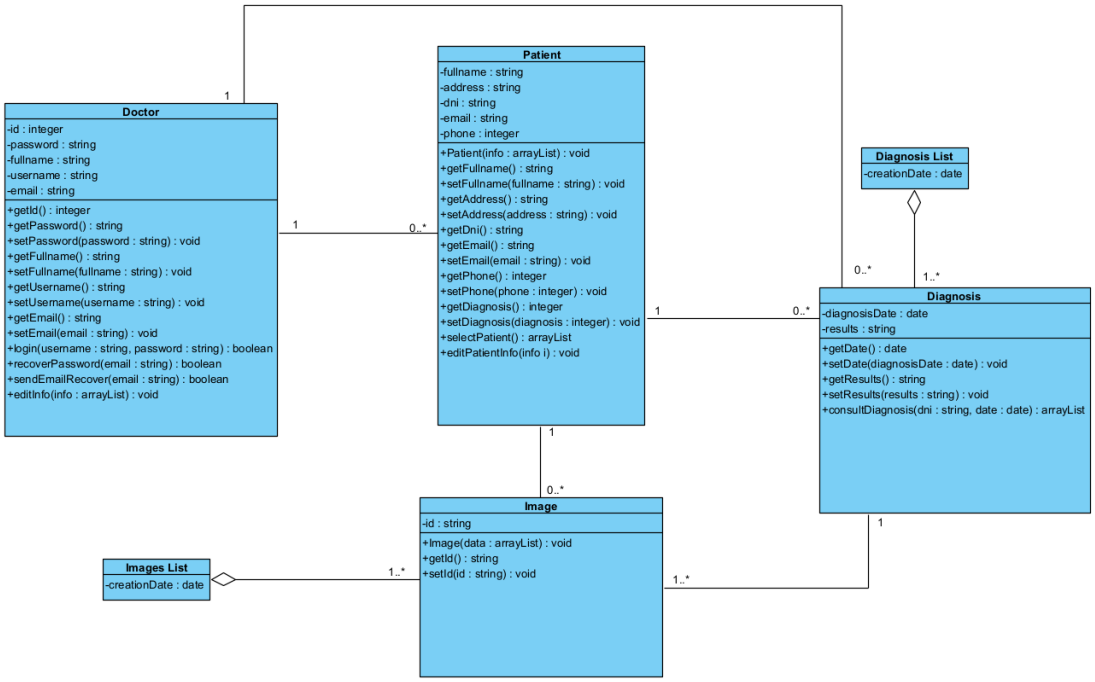


Ilustración 29: Diagrama de clases del modelo de diseño

5.6.4 Vista arquitectónica

Esta vista evoluciona de la vista representada en el *Anexo III – Análisis de requisitos,* pero siguiendo el patrón MVC comentado anteriormente.

En la "Ilustración 30: Diagrama de vista arquitectónica" se puede ver la estructura que tendrá el sistema.

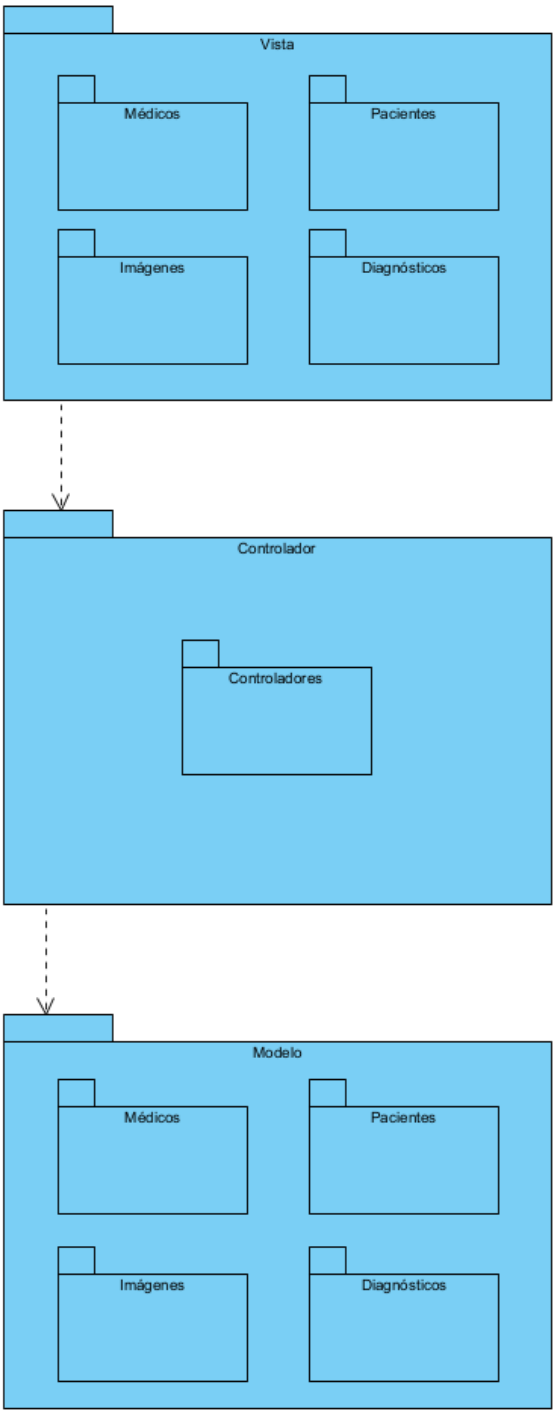


Ilustración 30: Diagrama de vista arquitectónica

5.6.5 Realización de los casos de uso

La realización de los casos de uso se representa con diagramas de secuencia indicando los mensajes ente los distintos componentes del sistema.

En la "Ilustración 31: Diagrama de secuencia del UC1 Iniciar sesión" se puede ver un ejemplo de realización de caso de uso.

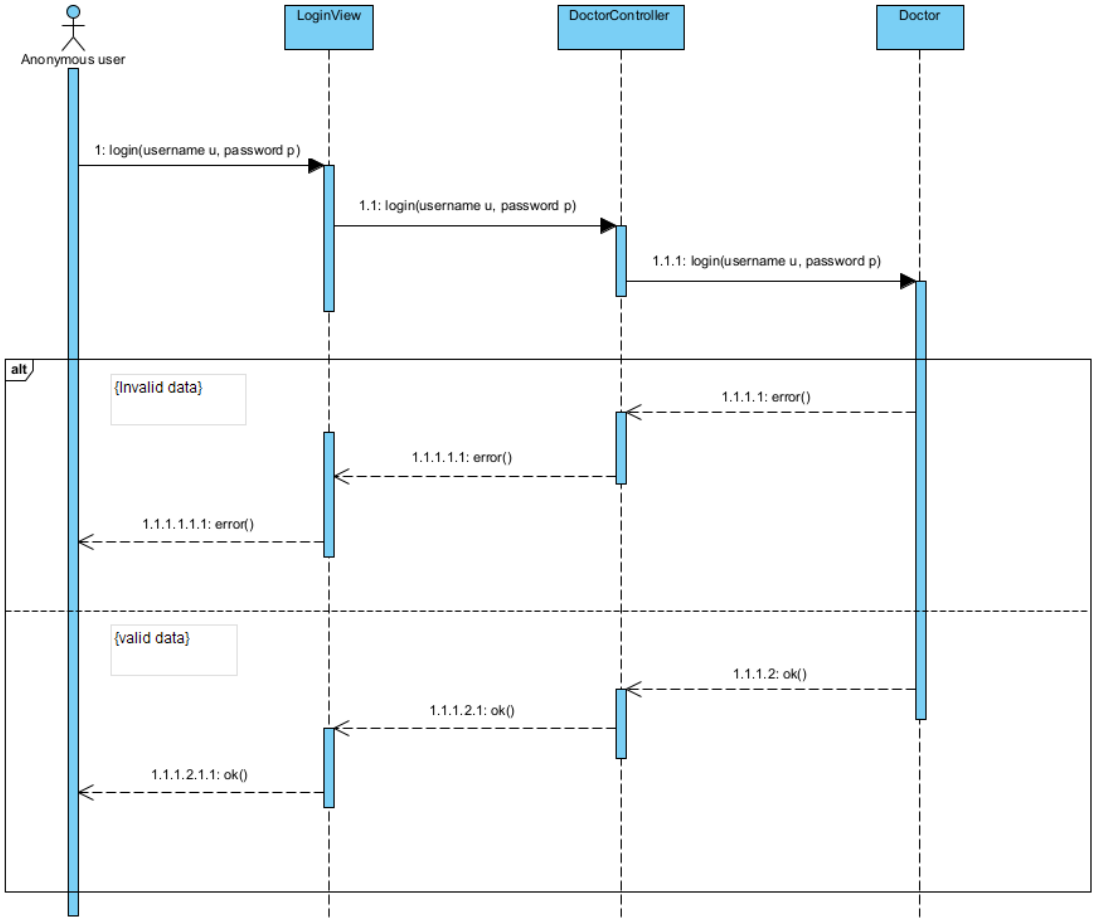


Ilustración 31: Diagrama de secuencia del UC1 Iniciar sesión

5.6.6 Modelo de despliegue

Todas las comunicaciones de los nodos y dispositivos se realizan a través de internet.

El diagrama de despliegue representa los distintos elementos del sistema. Cada nodo del diagrama representa un elemento hardware o software. Este despliegue se puede observar en la "Ilustración 32: Diagrama de despliegue".

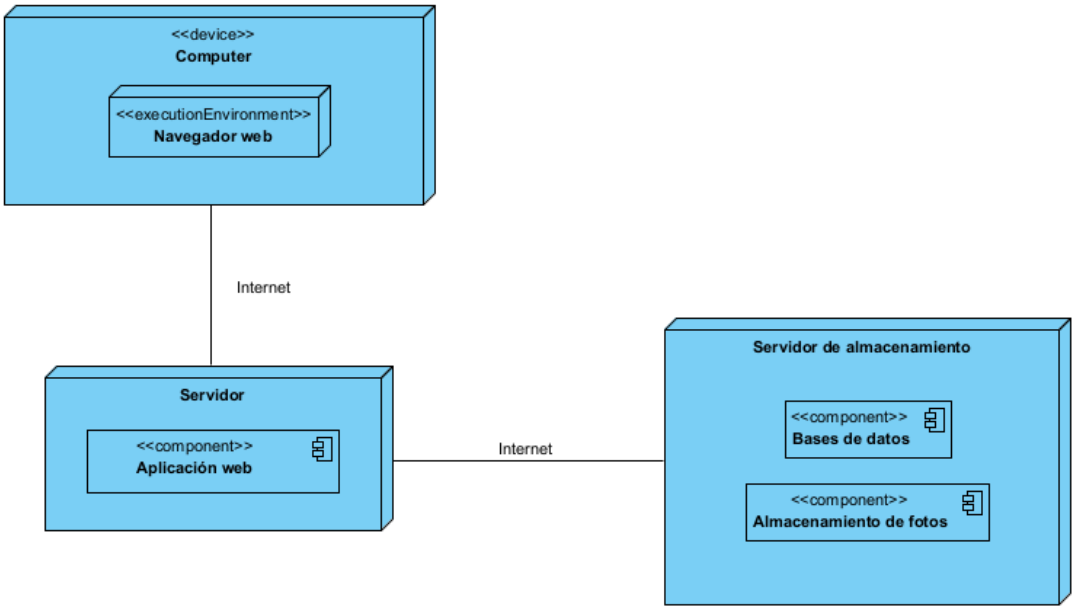


Ilustración 32: Diagrama de despliegue

5.7 Implementación

Para la implementación se puede hacer una diferencia entre la implementación del modelo de red neuronal y la implementación de la aplicación web la cual se ha desarrollado siguiendo los resultados de la fase de diseño descrita en el *Anexo IV – Diseño del sistema.*

Durante la implementación se ha utilizado las herramientas y técnicas comentadas a lo largo de la memoria. Además, se puede ver más al detalle la implementación del sistema consultando el *Anexo V – Manual del programador.*

5.7.1 Implementación del modelo de red neuronal

En este apartado se comentarán los fragmentos de código más importantes de la inteligencia artificial para ver como se ha utilizado el conjunto de datos que se ha comentado anteriormente en la memoria y como esta red neuronal funciona, dando así un diagnóstico a las imágenes. El tipo de modelo de red neuronal usado es CNN (*Convolutional neural network*) con aprendizaje supervisado que procesa sus distintas capas imitando al ojo humano, identificando las distintas características en las entradas que hace que puedan identificar figuras. Su funcionamiento consiste en aplicar filtros a cada imagen de entrada con distintas resoluciones, y la salida de cada imagen convulsionada se emplea como entrada para la siguiente.

Para el entrenamiento de la red neuronal se han utilizado distintas constantes entre las que destacan las siguientes:

- **NUM_EPOCHS = 50:** Especifica el número de veces que se ejecuta el algoritmo de *forwardpropagation* y *backpropagation*, en otras palabras, especifica el número de pases completos a través del conjunto de datos de entrenamiento. No existe una regla para elegir el valor concreto de este, sino que será el programador quien debe probar diferentes valores hasta encontrar el que mejor se comporte para su red neuronal.
- **INIT_LR = 1e-3:** *Learning rate* o tasa de aprendizaje es un parámetro de ajuste para un algoritmo de optimización que determina el tamaño de paso en cada iteración mientras se mueve hacia un mínimo de función de pérdida. En la práctica suele tener un valor muy pequeño.
- **BS = 64:** *Batch size* o tamaño de lote es el número de ejemplos que se introducen en la red neuronal para entrenar en cada iteración. Es recomendable que tenga un valor grande, puesto que así la red neuronal tendrá en cuenta los casos más importantes a la hora de aprender, pero el cálculo computacional será mayor, con lo que se obtendrá una mayor lentitud en el aprendizaje.
- **WIDTH = 48:** El ancho de las imágenes a analizar (en pixeles).
- **HEIGHT = 48:** El largo de las imágenes a analizar (en pixeles).

En primer lugar, se definen los generadores de imágenes de entrenamiento, validación y test usando la clase *ImageDataGenerator* para inicializar el objeto de aumento de datos y después se usa *flow_from_directory* para inicializar los generadores que proporcionarán lote de imágenes bajo demanda, tomando la ruta a un directorio. Estos permiten generar lotes de datos de imágenes de tensores con aumento de datos a tiempo real, haciendo posible la creación de una red neuronal sin necesidad de aumentar más el número de imágenes del conjunto de datos, debido a que esta función modifica con pequeños cambios las fotografías consiguiendo un mayor número de imágenes.

Como se puede observar en la “Ilustración 33: trainAug ImageDataGenerator”, la “Ilustración 34: valAug ImageDataGenerator” y la “Ilustración 35: testAug ImageDataGenerator” se han usado distintos argumentos los cuales se detallan debajo de las ilustraciones.

```
trainAug = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    brightness_range=None,
    shear_range=0.05,
    zoom_range=0.05,
    channel_shift_range=0.0,
    fill_mode='nearest',
    cval=0.0,
    horizontal_flip=True,
    vertical_flip=True,
    rescale=1 / 255.0,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None

)
```

Ilustración 33: trainAug ImageDataGenerator

```
valAug = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    brightness_range=None,
    shear_range=0.0,
    zoom_range=0.0,
    channel_shift_range=0.0,
    fill_mode='nearest',
    cval=0.0,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=1 / 255.0,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None
)
```

Ilustración 34: valAug ImageDataGenerator

```
testAug = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    zca_epsilon=1e-06,
    rotation_range=0,
    width_shift_range=0.0,
    height_shift_range=0.0,
    brightness_range=None,
    shear_range=0.0,
    zoom_range=0.0,
    channel_shift_range=0.0,
    fill_mode='nearest',
    cval=0.0,
    horizontal_flip=False,
    vertical_flip=False,
    rescale=1 / 255.0,
    preprocessing_function=None,
    data_format=None,
    validation_split=0.0,
    dtype=None
)
```

Ilustración 35: testAug ImageDataGenerator

Los argumentos utilizados para los distintos conjuntos son los siguientes:

- *featurewise_center:* Para establecer la media de entrada en 0 sobre el conjunto de datos, en función de las características.
- *samplewise_center:* Establecer la media de cada muestra en 0.
- *featurewise_std_normalization:* Divide las entradas por estándares del conjunto de datos, según características.
- *samplewise_std_normalization:* Divide cada entrada por su estándar.
- *zca_whitening:* Épsilon para blanquear ZCA.
- *zca-epsilon:* Aplica blanqueamiento ZCA.
- *rotation_range:* Rango de grados para rotaciones aleatorias.
- *width_shift_range:* Realiza desplazamientos de las imágenes hacia la izquierda o la derecha (desplazamientos horizontales).
- *height_shift_range:* Realiza desplazamientos de las imágenes hacia arriba y hacia abajo (desplazamientos verticales).
- *brightness_range:* Rango para elegir un valor de cambio de brillo.
- *shear_range:* Intensidad (ángulo) de corte.
- *zoom_range:* Rango para zoom aleatorio.
- *channel_shift_range:* Rango para cambios de canal aleatorios.
- *fill_mode:* Los puntos de fuera de los límites de la entrada se llenan de acuerdo con un modo dado.
- *cval:* Valor utilizado para puntos fuera de los límites.

- *horizontal_flip:* Voltea las entradas de forma aleatoria horizontalmente.
- *vertical_flip:* Voltea las entradas verticalmente al azar.
- *rescale:* Factor de cambio de escala.
- *preprocessing_function:* Función que se aplicará en cada entrada.
- *data_format:* Formato de datos de imagen.
- *validation_split:* Fracción de imágenes reservada para la validación
- *dtype:* Se utilizará para las matrices generadas.

Como se ha podido apreciar en la "Ilustración 34: valAug ImageDataGenerator" e "Ilustración 35: testAug ImageDataGenerator" generan los lotes de datos de imágenes de la misma forma, pero si se comparan con la "Ilustración 33: trainAug ImageDataGenerator" se puede observar que este último en sus argumentos tiene bastantes diferencias con respecto a los otros dos, esto es así porque es necesario que las imágenes de este conjunto tengan mayores diferencias entre ellas y la variedad sea más grande, consiguiendo de esta forma que la acción de entrenar la red neuronal sea más efectiva.

En la "Ilustración 36: trainGen flow_from_directory", la "Ilustración 37: valGen flow_from_directory" y la "Ilustración 38: testGen flow_from_directory" se pueden observar los distintos argumentos usados para generar los lotes de datos aumentados tomando la ruta a un directorio.

```
trainGen = trainAug.flow_from_directory(
    TRAIN_PATH,
    class_mode="categorical",
    target_size=(WIDTH, HEIGHT),
    color_mode="rgb",
    shuffle=True,
    seed=42,
    batch_size=BS
)
```

Ilustración 36: trainGen flow_from_directory

```
valGen = valAug.flow_from_directory(
    VAL_PATH,
    class_mode="categorical",
    target_size=(WIDTH, HEIGHT),
    color_mode="rgb",
    shuffle=False,
    seed=False,
    batch_size=BS
)
```

Ilustración 37: valGen flow_from_directory

```
testGen = testAug.flow_from_directory(
    TEST_PATH,
    class_mode="categorical",
    target_size=(WIDTH, HEIGHT),
    color_mode="rgb",
    shuffle=False,
    seed=False,
    batch_size=BS
)
```

Ilustración 38: testGen flow_from_directory

Los argumentos utilizados para los distintos conjuntos son los siguientes:

- *directorio*: Ruta al directorio de destino. Debe contener un subdirectorio por clase (0 ,1).
- *class_mode*: Determina el tipo de matrices de etiquetas que se devuelven.
- *target_size*: Las dimensiones a las que se cambiarán todas las imágenes encontradas
- *color_mode*: Se usa para determinar si las imágenes se convertirán para tener 1, 3 o 4 canales.
- *shuffle*: Si se deben mezclar los datos o si por el contrario se ordenan en orden alfanumérico.
- *seed*: Semilla aleatoria opcional para barajar el conjunto de datos.
- *batch_size*: Tamaño de los lotes de datos.

El siguiente punto es la creación del modelo, para este se eligió el tipo secuencial (*Sequencial()*), ya que se buscaba crear una red neuronal formado por capas, teniendo cada una de estas un tensor de entrada y otro tensor de salida. Junto con el tipo de modelo, se define el *input shape* (forma de entrada). En la "Ilustración 39: Modelo" se puede observar tanto el modelo elegido como el formato de entrada.

```
model = Sequential()
inputShape = (height, width, depth)
```

Ilustración 39: Modelo

Una vez decidido el tipo de modelo, se añaden las distintas capas que lo compondrán. Por ello se mencionan aquí los tipos usados y luego la arquitectura que se creó con las mismas.

Tipos de capas usadas:

SeparableConv2D: Las convoluciones separables consisten en realizar primero una convolución espacial en profundidad, seguida de una convolución puntual que mezcla los canales de salida resultantes.

Argumentos**:**

- Filtros: son el número de filtros de salida de la convolución
- *Kernel_size*: Especifica la altura y el ancho de la ventana de convolución 2D.
- *Padding*: Da como resultado un relleno, de modo que la salida tiene la misma dimensión que la entrada.
- *Input_shape:* Dimensionalidad de la entrada.

Activation ("relu")*:* Se trata de una función de activación. Su uso se debe a que transforma los valores introducidos anulando los valores negativos y dejando los positivos sin modificar.

BatchNormalization: La normalización por lotes aplica una transformación que mantiene la salida cercana a 0 y la desviación estándar de salida cercana a 1.

MaxPooling2D: Reduce la muestra de entrada a lo largo de sus dimensiones espaciales (ancho y alto) tomando el valor máximo sobre una ventana de entrada (*pool_size*)

Argumentos:

- *Pool_size*: Tamaño de ventana sobre el que tomar el valor máximo.

Dropout: La capa de abandono establece aleatoriamente las unidades de entrada en 0 con una determinada frecuencia (*rate*). De esta forma se evita que el modelo tenga un sobreajuste sobre el conjunto de datos, ya que este es de un tamaño limitado.

Flatten: Esta capa es utilizada para convertir una entrada multidimensional en una unidimensional.

Dense: Esta capa de cálculo conecta cada neurona en una capa con todas las salidas de la capa anterior.

Activation ("softmax"): Este tipo de capas suele colocarse la última en una red neuronal, esto se debe a que el resultado que ofrece puede interpretarse como una distribución de probabilidad.

Para la arquitectura se decidió crear 3 grandes bloques, cada uno con un mayor número de capas por bloque. Además de esta variación, también según se avanza en los bloques el número de filtros de las capas *SeparableConv2D* va en aumento (de 32 a 64 y a 128).

Como se puede observar en la "Ilustración 40: Primer bloque", la "Ilustración 41: Segundo bloque" y la "Ilustración 42: Tercer bloque" se puede ver la arquitectura de las distintas capas que forman el modelo de la red neuronal.

```
model.add(SeparableConv2D(32, (3, 3), padding="same", input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(SeparableConv2D(32, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

Ilustración 40: Primer bloque

```
model.add(SeparableConv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(SeparableConv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(SeparableConv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

Ilustración 41: Segundo bloque

```
model.add(SeparableConv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(SeparableConv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(SeparableConv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(SeparableConv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

Ilustración 42: Tercer bloque

Por último, se añadió un último bloque que se puede observar en la "Ilustración 43: Cuarto bloque" el cual contiene distintas capas, destacando la última llamada *Activation("softmax")* la cual dará como salida el porcentaje de predicción de la imagen analizada. De esta forma se consigue que la red neuronal creada ofrezca un diagnóstico por cada imagen a analizar.

```
model.add(Flatten())
model.add(Dense(256))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))

# Softmax classifier
model.add(Dense(classes))
# The output of softmax will be a percentage of prediction
model.add(Activation("softmax"))
```

Ilustración 43: Cuarto bloque

Una vez creada la arquitectura que tendrá la red neuronal, es necesario compilar y entrenar el modelo con las entradas dadas. Por ello, como se puede observar en la "Ilustración 44: Compilar y entrenar", se hace uso de las funciones *compile* y *fit*.

```
opt = Adagrad(learning_rate=INIT_LR, decay=INIT_LR/NUM_EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=["accuracy"])


# H -> history.history record of training loss values and metrics values at successive epochs
H = model.fit(x=trainGen, steps_per_epoch=totalTrain // BS,
              validation_data=valGen,
              validation_steps=totalVal // BS,
              class_weight=classWeight,
              epochs=NUM_EPOCHS)

print("Save the model: " + MODEL_PATH)
model.save(MODEL_PATH, save_format="h5")
```

Ilustración 44: Compilar y entrenar

Los argumentos utilizados para el compilador son los siguientes:

- *loss*: Cadena (nombre de la función objetivo), función objetivo o instancia.
- *optimizer*: optimizador usado para compilar el modelo.
- *metrics*: Lista de métricas que el modelo evaluará durante el entrenamiento y las pruebas.

Los argumentos utilizados para el entrenamiento son los siguientes:

- *x:* Los datos de entrada.
- *steps_per_epoch*: Número total de pasos (lotes de muestras) antes de declarar finalizada una época y comenzar la siguiente.
- *validation_data*: Datos sobre los que se evalúa la pérdida y cualquier métrica del modelo al final de cada época.
- *validation_steps*: Número total de pasos (lotes de muestras) que se deben extraer antes de detenerse al realizar la validación al final de cada época.
- *class_weight*: Utilizado para ponderar la función de pérdida (solo durante el entrenamiento).
- *epochs*: Número de épocas para el entrenamiento del modelo.

- *callbacks*: Lista de *callbacks* para aplicar durante el entrenamiento.

Además, se puede ver el uso del optimizador *Adagrad* (*Adaptive Gradient Algorithm*), este tiene unas tasas de aprendizaje específicas de parámetros, que se adaptan en relación con la frecuencia con la que se actualiza un parámetro durante el entrenamiento. Cuantas más actualizaciones reciba un parámetro, menores serán sus actualizaciones. Sus argumentos son los siguientes:

- *learning_rate:* Tasa de aprendizaje que usa el optimizador.
- *decay:* La tasa de aprendizaje que decae con cada actualización.

5.7.2 Implementación de la aplicación web

La implementación de la aplicación web se realizó de una manera iterativa e incremental, de esta forma se consiguió desarrollar el sistema cumpliendo todos los objetivos y casos de uso que se propusieron en la fase de especificación de requisitos.

Una de las primeras tareas realizadas fueron la implementación de una aplicación web básica con la cual el programador se pudiera familiarizar con Python y Flask. En ella se podía predecir distintos diagnósticos, pero no guardaba ningún tipo de dato, simplemente mostraba resultados para un conjunto de imágenes.

Tras esta aplicación en la cual ya se podía usar la red neuronal, el siguiente paso era la implementación de la gestión de los médicos y pacientes. De esta forma se conseguía que los profesionales sanitarios fueran capaces de iniciar sesión en el sistema pudiendo así crear pacientes, modificarlos, eliminarlos y analizarlos.

Puesto que se necesitaba tener un historial de diagnósticos en el sistema se implementó en la aplicación las distintas funcionalidades para obtener este objetivo. Así los médicos eran capaces de guardar los diagnósticos de los pacientes, pudiéndolos consultar, modificar.

En la última etapa, se implementó la gestión de imágenes, pudiéndolas guardar en la aplicación para futuras consultas y análisis. También se mejoró la interfaz de esta con la ayuda del tema bootstrap (SB Admin 2) comentado anteriormente en la memoria, de esta forma, se intentó que la aplicación quedará mucho más clara de cara al usuario.

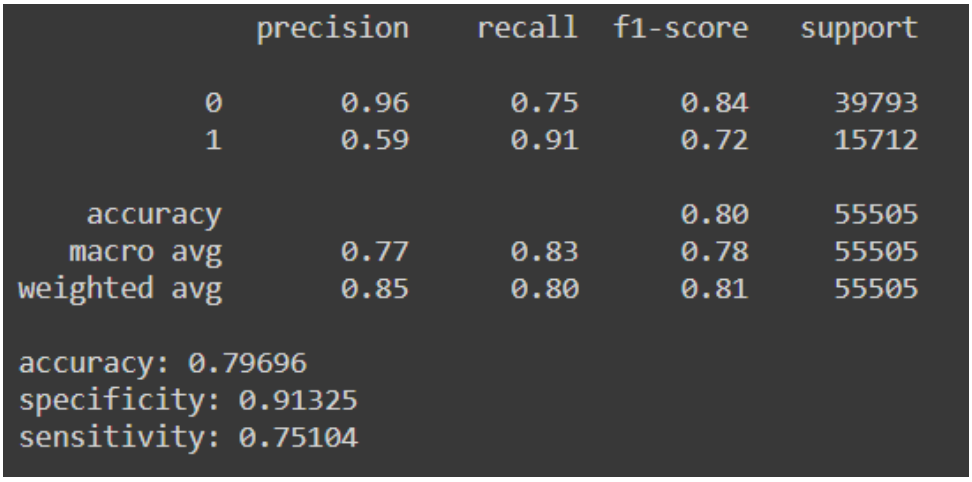
5.8 Pruebas

Las pruebas realizadas durante el desarrollo del trabajo de fin de grado pueden dividirse en pruebas realizadas sobre el entrenamiento del modelo de red neuronal para asegurar que esta tiene las cualidades necesarias para poder predecir imágenes de una forma correcta y pruebas sobre la aplicación web, centrándose en ver que el sistema funcione de manera correcta con la interacción del usuario.

5.8.1 Pruebas sobre el modelo de red neuronal

En este apartado de exponen las diversas variables obtenidas a partir del entrenamiento y testeo de la red neuronal.

Una vez se entrenó la inteligencia artificial, se recopilaron métricas de evaluación en el mismo fichero en el que se entrena la red neuronal llamado “trainModel.py”. Estas métricas se pueden observar en la “Ilustración 45: Métricas de evaluación” de las cuales se comentarán las más importantes.



	precision	recall	f1-score	support
0	0.96	0.75	0.84	39793
1	0.59	0.91	0.72	15712
accuracy			0.80	55505
macro avg	0.77	0.83	0.78	55505
weighted avg	0.85	0.80	0.81	55505

accuracy: 0.79696
specificity: 0.91325
sensitivity: 0.75104

Ilustración 45: Métricas de evaluación

En el entrenamiento de la red neuronal se ha obtenido una precisión del 80%. La clasificación para los casos negativos de cáncer (“benigno / sin cáncer”) se clasifica correctamente un 96% de las veces, sin embargo, para los casos positivos de cáncer (“maligno / cáncer”) se clasifica correctamente un 59% de las veces.

Además, se puede observar la sensibilidad y la especificidad obtenida para comprender mejor el rendimiento del modelo a un nivel más profundo:

- **Sensibilidad (sensitivity):** Mide la proporción de verdaderos positivos que también se predijeron como positivos, es decir, representa el número de personas que no son saludables y se predicen que no lo son. El modelo tiene un 75,10% de sensibilidad.
- **Especificidad (specificity):** Mide los verdaderos negativos, es decir, representa el número de personas que están sanas y se predicen que están sanas. El modelo tiene un 91,13% de especificidad.

Una vez visto el porcentaje de acierto que tiene el modelo para los verdaderos positivos y los verdaderos negativos, hay que tener en cuenta que en el diagnóstico del cáncer de mama se debe que tener mucho cuidado con los falsos negativos, puesto que no se quiere clasificar a una paciente como sana cuando en realidad padece la enfermedad. Esto el modelo lo solventa con grandes resultados puesto que el 75,10% de las pacientes que padecieron la enfermedad fueron diagnosticadas por la red neuronal como enfermas.

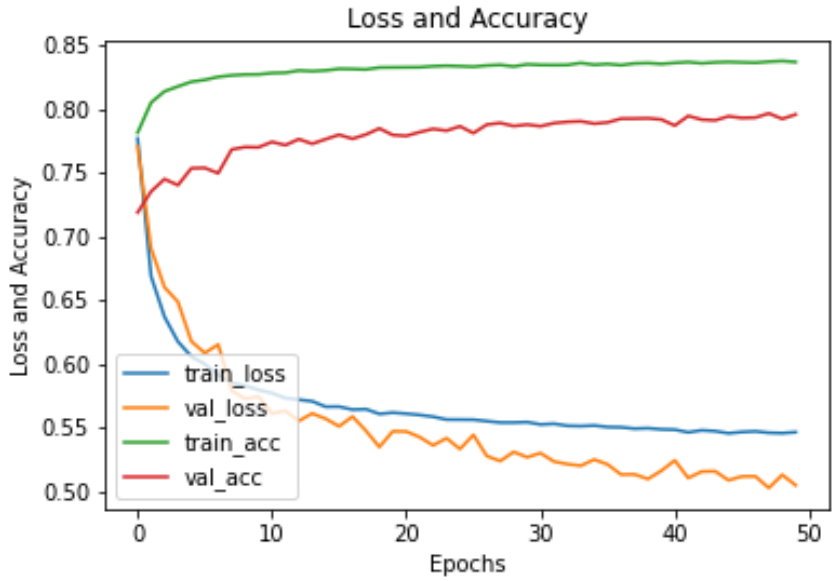


Ilustración 46: Función de perdida y precisión

La precisión es un método que sirve para medir el rendimiento en un modelo de clasificación. Suele expresarse en forma de porcentaje e informa de las predicciones en las que el valor predicho es correcto. Como se puede observar en la "Ilustración 46: Función de perdida y precisión", según crecían las *epochs* aumentaba el porcentaje de aciertos, tanto para el entrenamiento como para la validación, afirmando de esta forma que el modelo iba aprendiendo satisfactoriamente.

Una función de pérdida (también conocida como función de coste) trata de determinar el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros de una red neuronal. En la "Ilustración 46: Función de perdida y precisión" se puede observar cómo evoluciona la pérdida de validación (*val_loss*) junto con la perdida de entrenamiento (*tain_loss*). De esto se pueden sacar las distintas conclusiones:

- Si la pérdida de validación es mucho más grande que la pérdida de entrenamiento el modelo tiene *overfitting*.
- Si la pérdida de validación es mucho menor que la pérdida de entrenamiento el modelo tiene *underfitting*.

Como se ha podido ver en la "Ilustración 45: Función de perdida y precisión" existe una pequeña diferencia entre la pérdida de validación y la pérdida de entrenamiento. Si esta diferencia fuera mucho mayor se debería de tener en cuenta, modificando el modelo o el conjunto de datos utilizado para minimizar ese *underfitting* que presenta el modelo, pero al ser una diferencia tan pequeña es irrelevante, obteniendo así un modelo con unas buenas características para el análisis de imágenes.

5.8.2 Pruebas sobre la aplicación web

Se han realizado pruebas durante la implementación de la aplicación web.

Cada vez que se desarrollaba una parte del sistema se realizaban pruebas unitarias sobre esta. Gracias a ello, se obtenía una cantidad menor de fallos y así se podía asegurar que a la hora de realizar la integración de diversas partes, la cantidad de errores disminuyera drásticamente.

Además de las pruebas unitarias, se han realizado pruebas de integración con las cuales se asegura que todos los elementos unitarios que componen el software funcionen juntos correctamente, estas pruebas están centradas principalmente en probar la comunicación entre componentes.

5.9 Funcionalidad de la aplicación

La funcionalidad de la aplicación se centra en los siguientes ámbitos:

- Gestión de médicos
- Gestión de pacientes
- Gestión de diagnósticos
- Gestión de imágenes

Para ver más al detalle el diseño del sistema se puede consultar el *Anexo VI – Manual de usuario.*

5.9.1 Médicos

La aplicación provee diversas herramientas para que los médicos puedan realizar su trabajo en la aplicación.

A continuación, se adjuntan diferentes ilustraciones:

Para iniciar sesión, tan solo hace falta introducir el nombre de usuario que posee el médico y su contraseña, de esta forma se puede acceder a la aplicación, tal y como se observa en la “Ilustración 47: Inicio de sesión” y la “Ilustración 48: Página principal”.

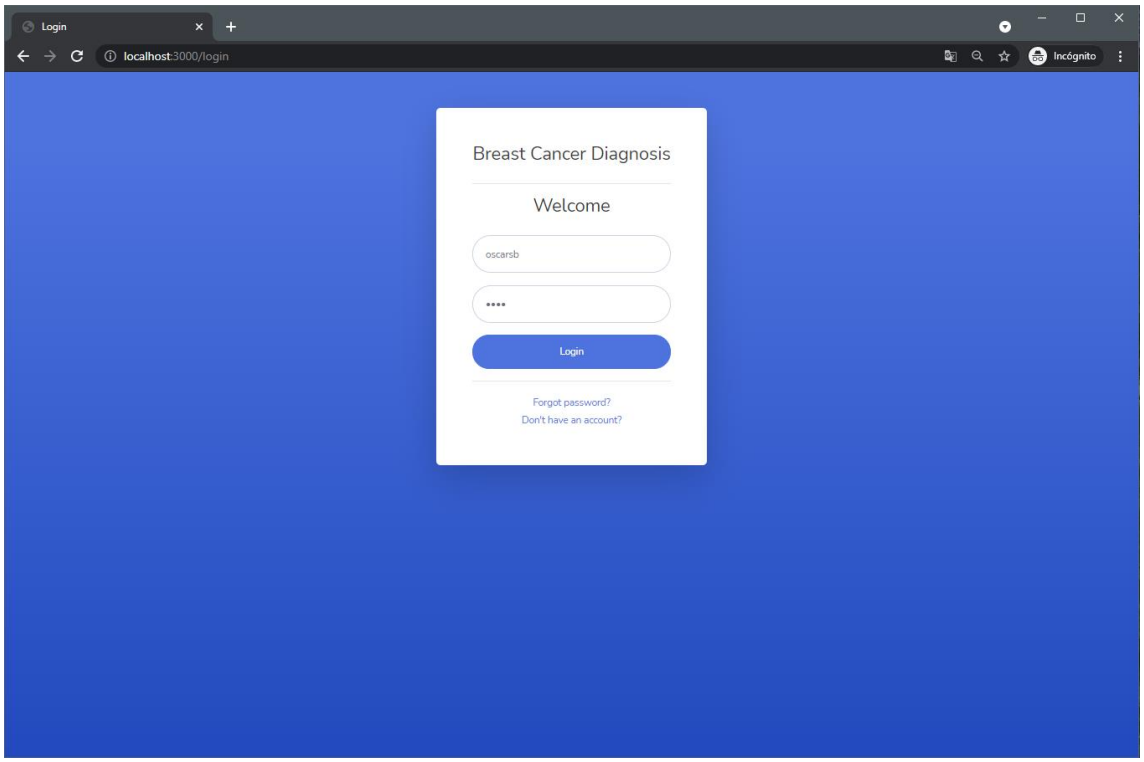


Ilustración 47: Inicio de sesión

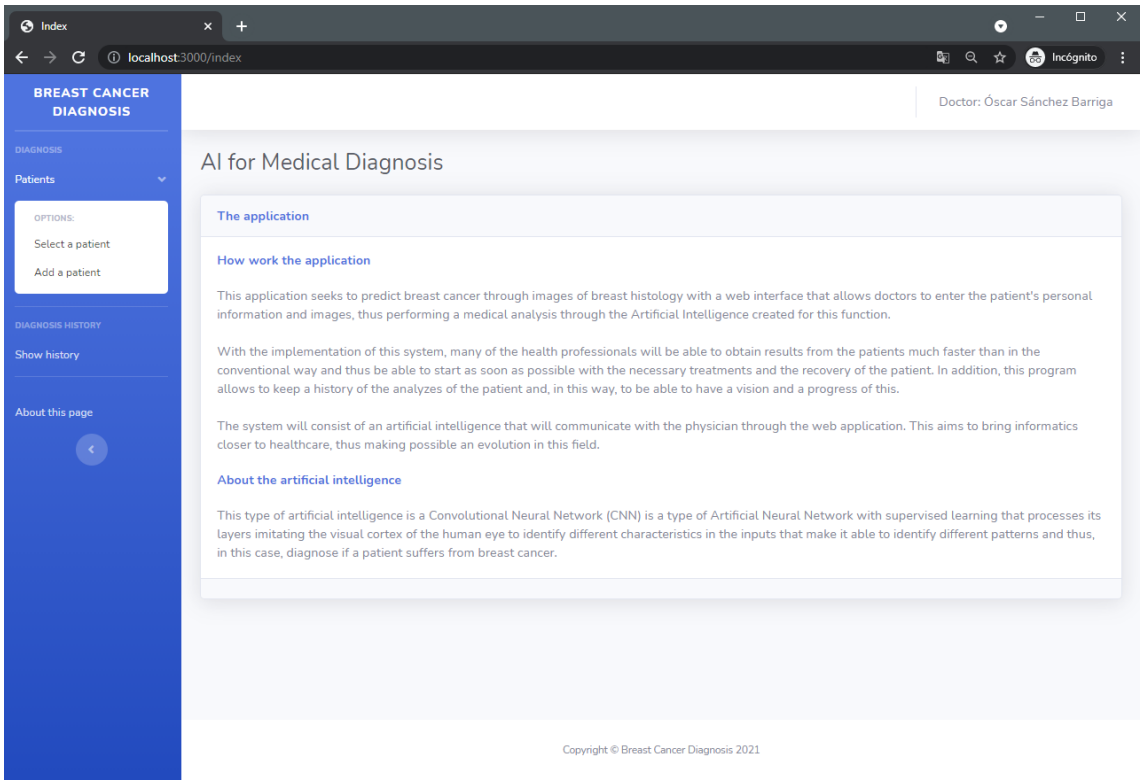


Ilustración 48: Página principal

En el caso de que el médico haya perdido la contraseña, podrá solicitar una nueva a través de la aplicación, como se puede observar en la "Ilustración 49: Contraseña olvidada".

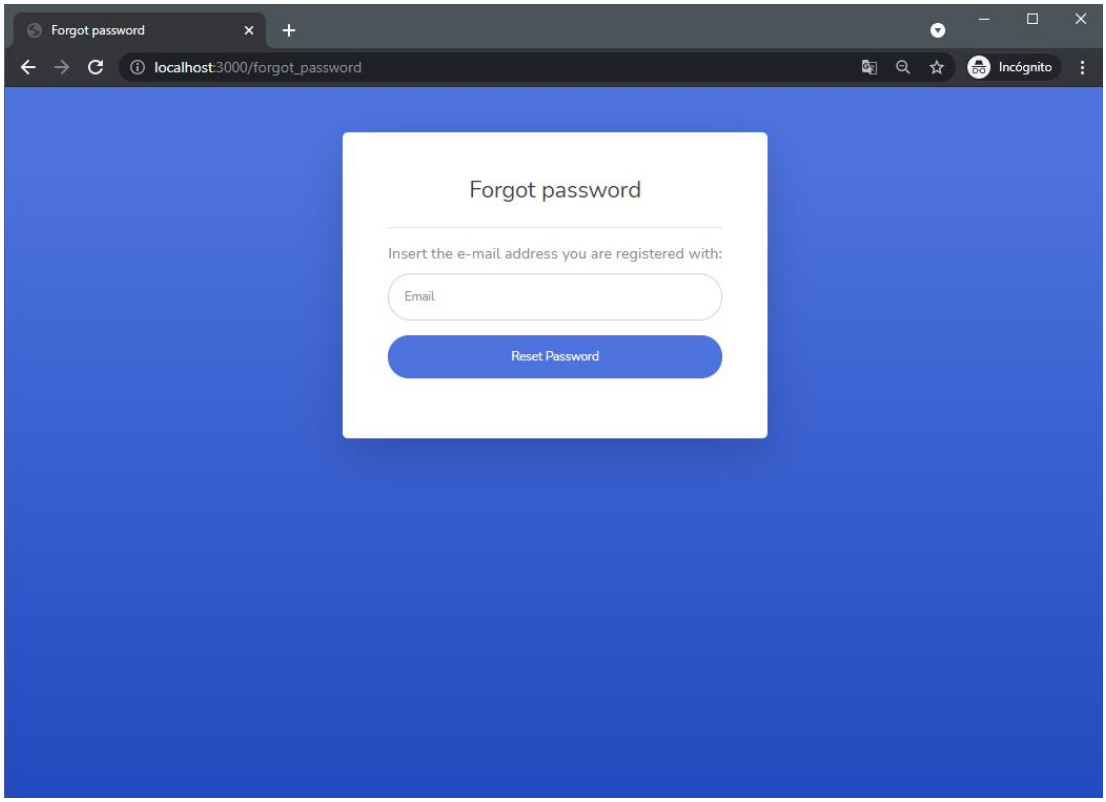


Ilustración 49: Contraseña olvidada

De esta forma, el médico recibirá un correo electrónico como el que se muestra a continuación en la "Ilustración 50: Correo electrónico para recuperar la contraseña".

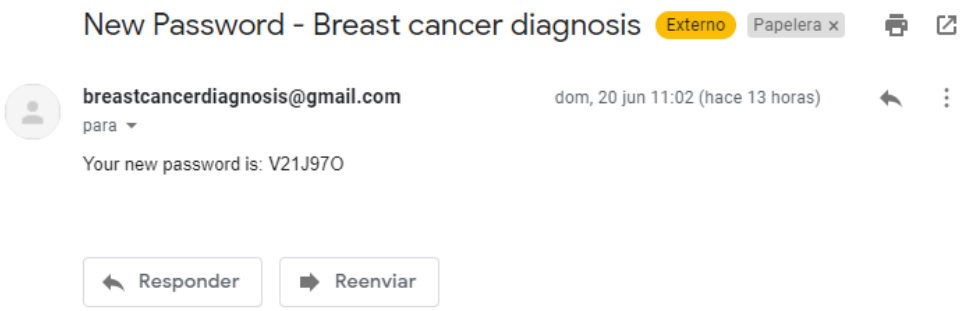


Ilustración 50: Correo electrónico para recuperar la contraseña

Si, por el contrario, todavía no tiene una cuenta en la aplicación, esta permitirá ponerse en contacto con el administrador para que le dé de alta en la plataforma, como se muestra a continuación en la "Ilustración 51: Solicitud de cuenta".

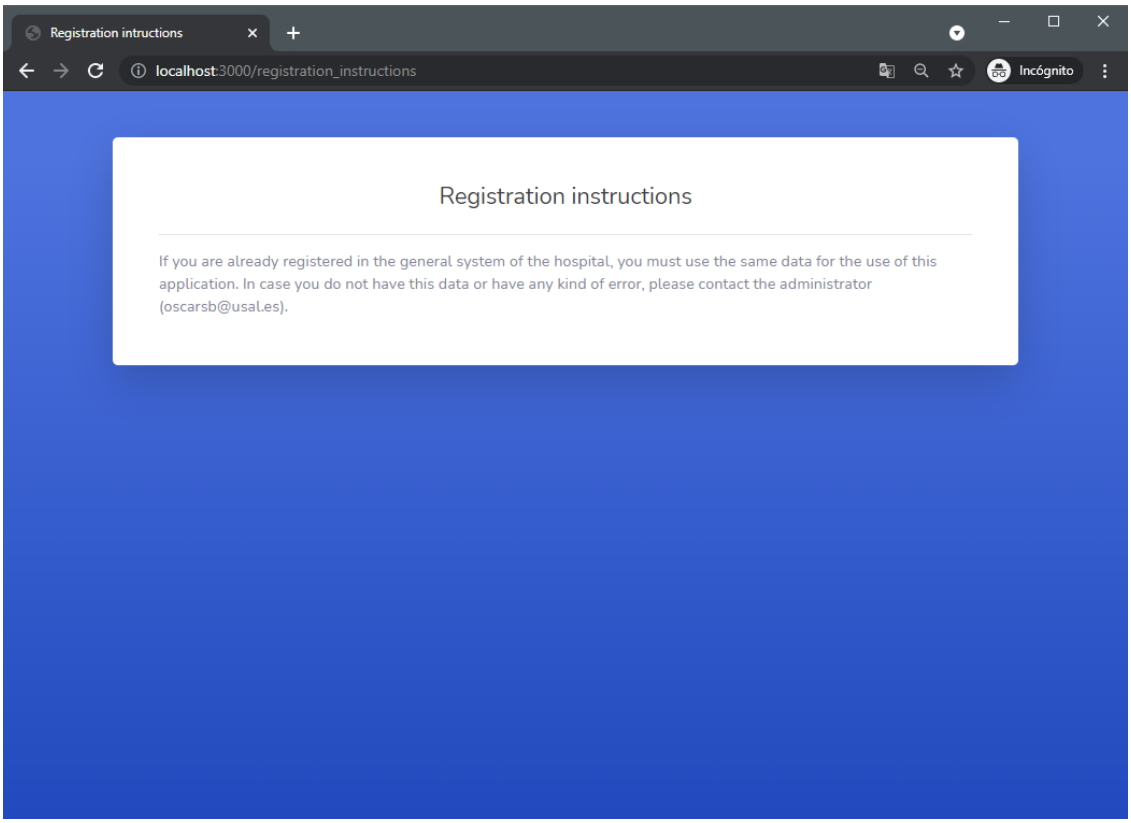


Ilustración 51: Solicitud de cuenta

Además, se podrá mostrar la información del perfil, así como editarla, tal y como se puede observar en la "Ilustración 52: Mostrar información del perfil" y en la "Ilustración 53: Editar información del perfil".

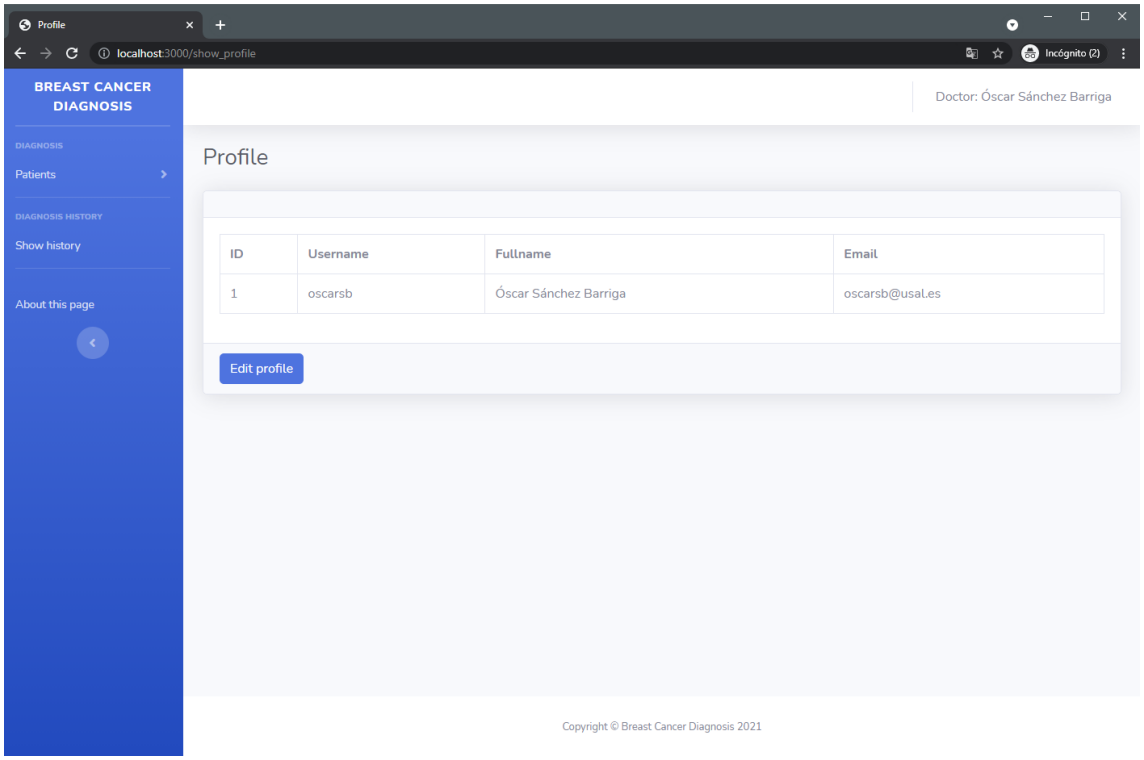


Ilustración 52: Mostrar información del perfil

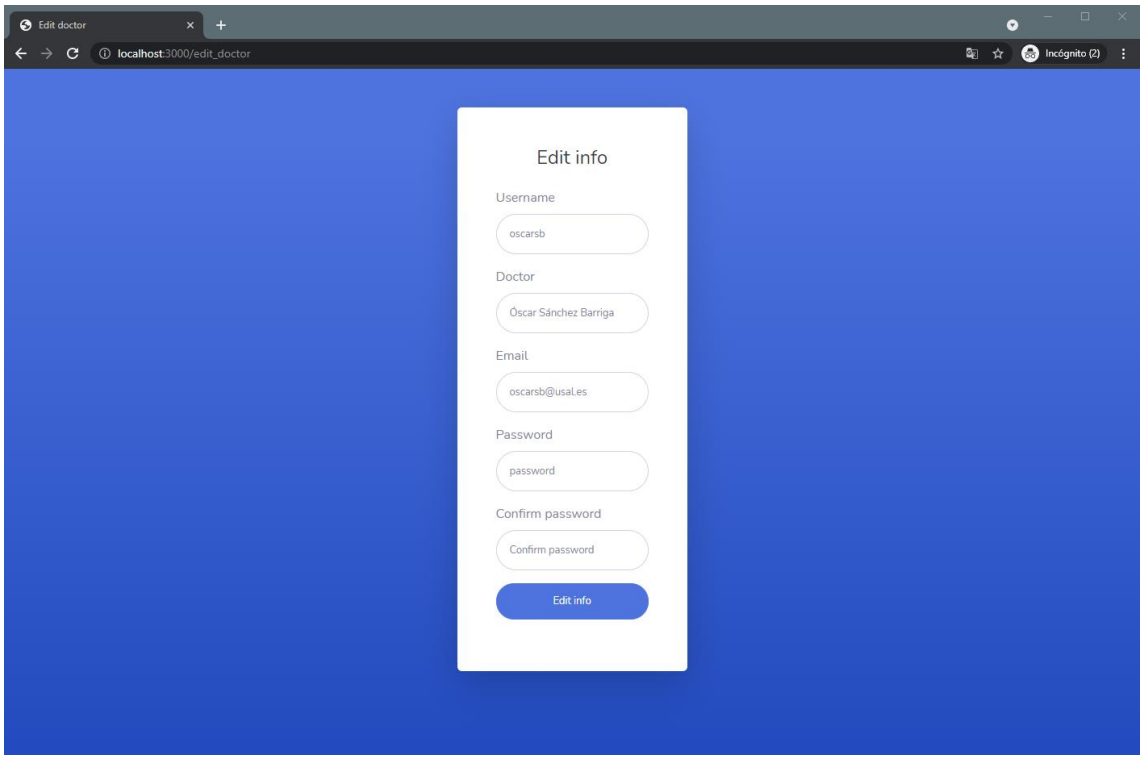


Ilustración 53: Editar información del perfil

El médico siempre podrá cerrar sesión en el momento que haya terminado de realizar las tareas que desea en la plataforma, tal y como se observa en la "Ilustración 54: Cerrar sesión".

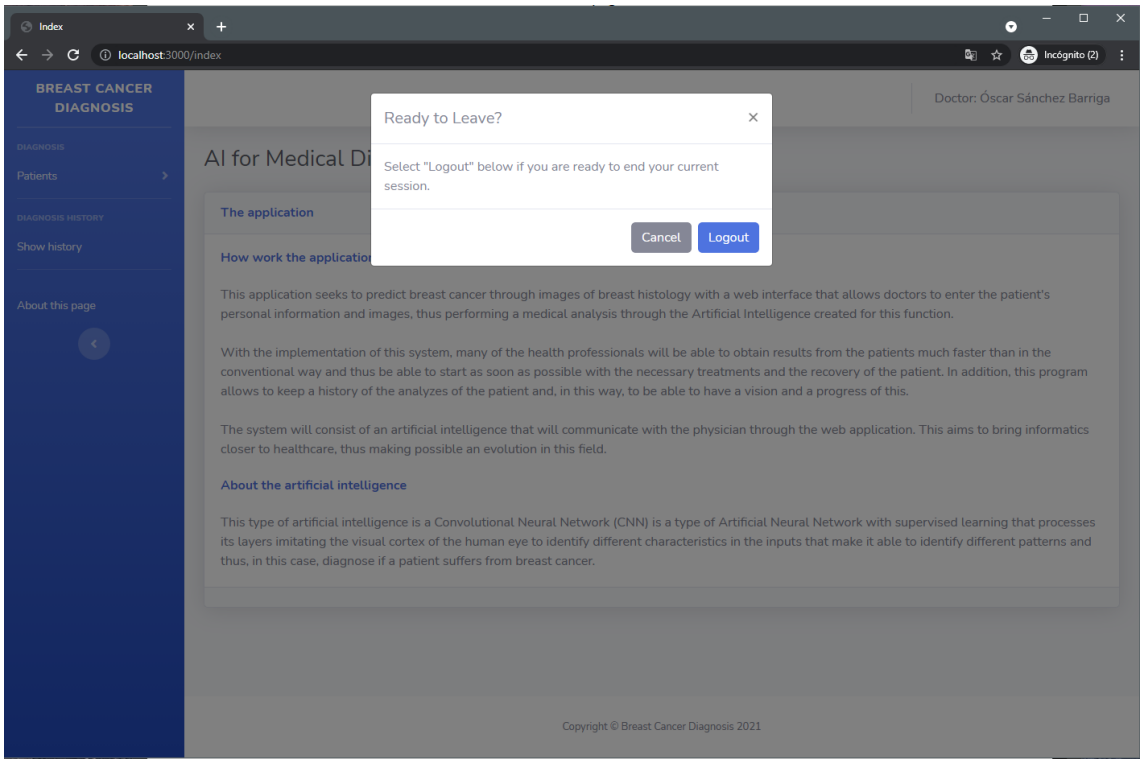


Ilustración 54: Cerrar sesión

5.9.2 Gestión de pacientes

Uno de los aspectos principales de la aplicación es la creación y diagnóstico de los pacientes. Por ello, el sistema permite la creación, modificación y eliminación de los pacientes, además de analizarlos con la red neuronal.

A continuación, se adjuntan diferentes ilustraciones:

En la página principal, se muestra un desplegable tras pinchar en 'Patients', como se observa en la "Ilustración 55: Index Gestión de pacientes", ofreciendo la posibilidad de añadir una paciente nueva o de seleccionar una de la base de datos.

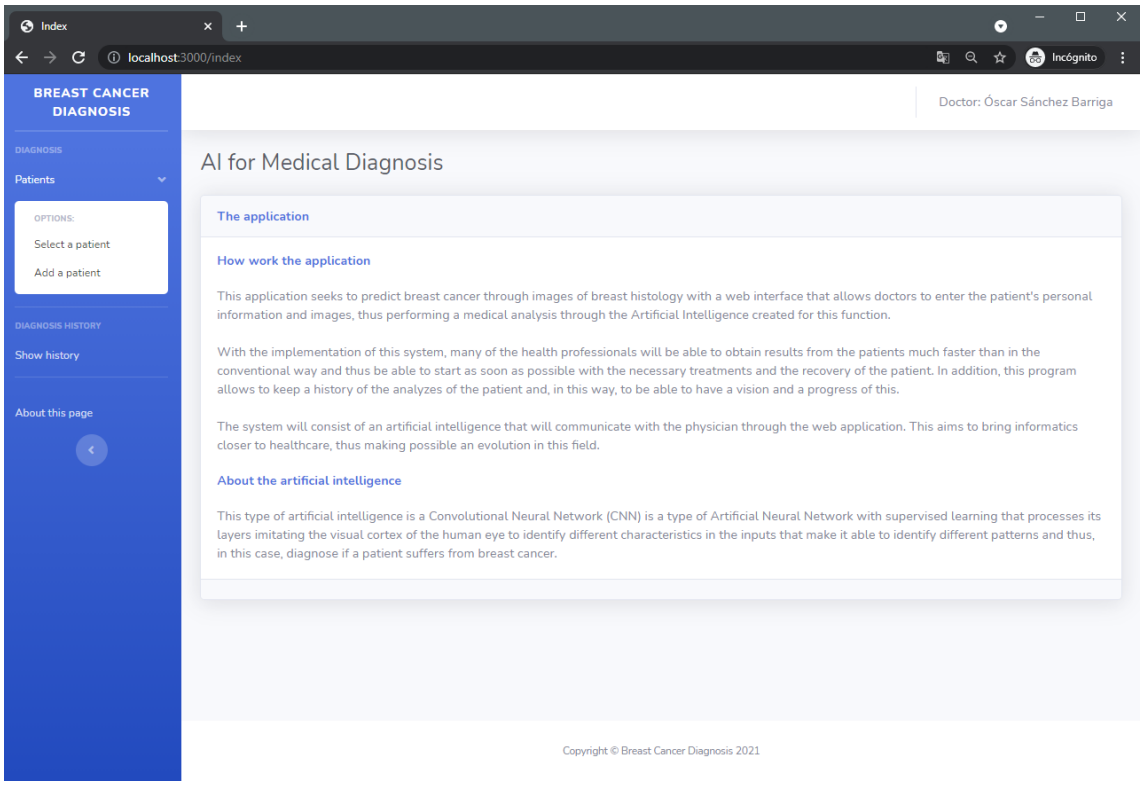


Ilustración 55: Index Gestión de pacientes

Si el médico desea crear un perfil de paciente en la aplicación, lo hará a través de la página que se observa en la "Ilustración 56: Añadir paciente", habiendo pulsado previamente en el botón 'Add a patient' de la "Ilustración 55: Index Gestión de pacientes".

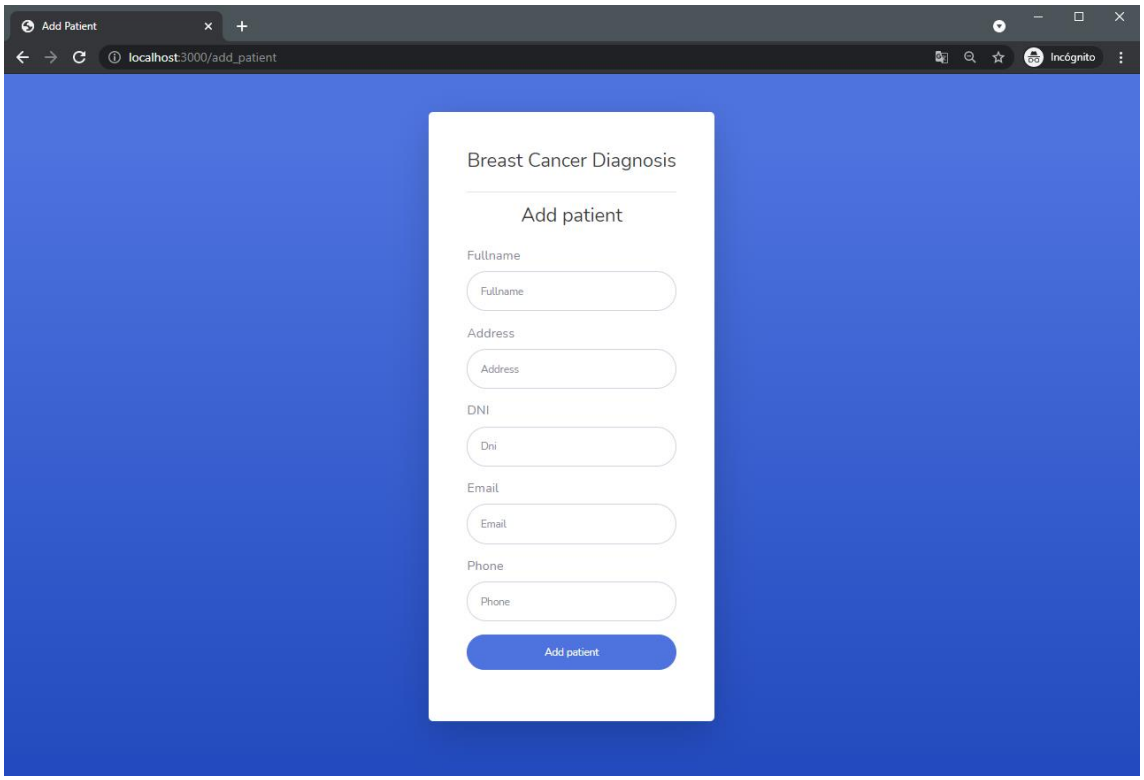


Ilustración 56: Añadir paciente

Si se pulsa sobre el botón 'Select patient' de la "Ilustración 55: Index Gestión de pacientes", la aplicación permite seleccionar un paciente y mostrar su información, buscándolo previamente por el DNI, como se muestra en la "Ilustración 57: Seleccionar un paciente" y esta mostrará la información que tiene en la base de datos como ocurre en la "Ilustración 58: Mostrar información de una paciente".

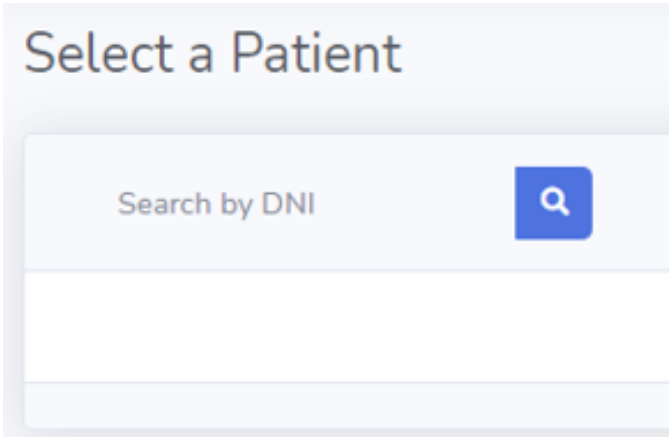


Ilustración 57: Seleccionar un paciente

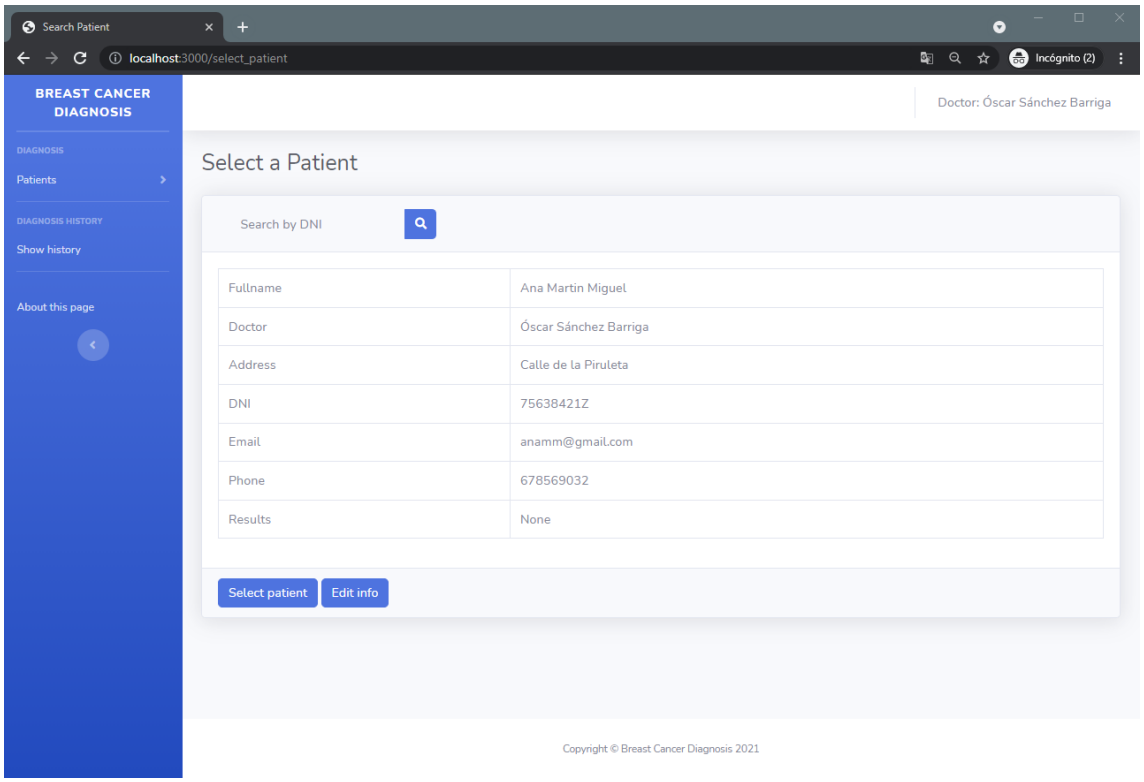


Ilustración 58: Mostrar información de una paciente

Por el contrario, si el médico desea editar un paciente ya existente en la aplicación, pulsará sobre el botón de la "Ilustración 58: Mostrar información de una paciente" llamado 'Edit info', mostrándose la "Ilustración 59: Editar paciente".

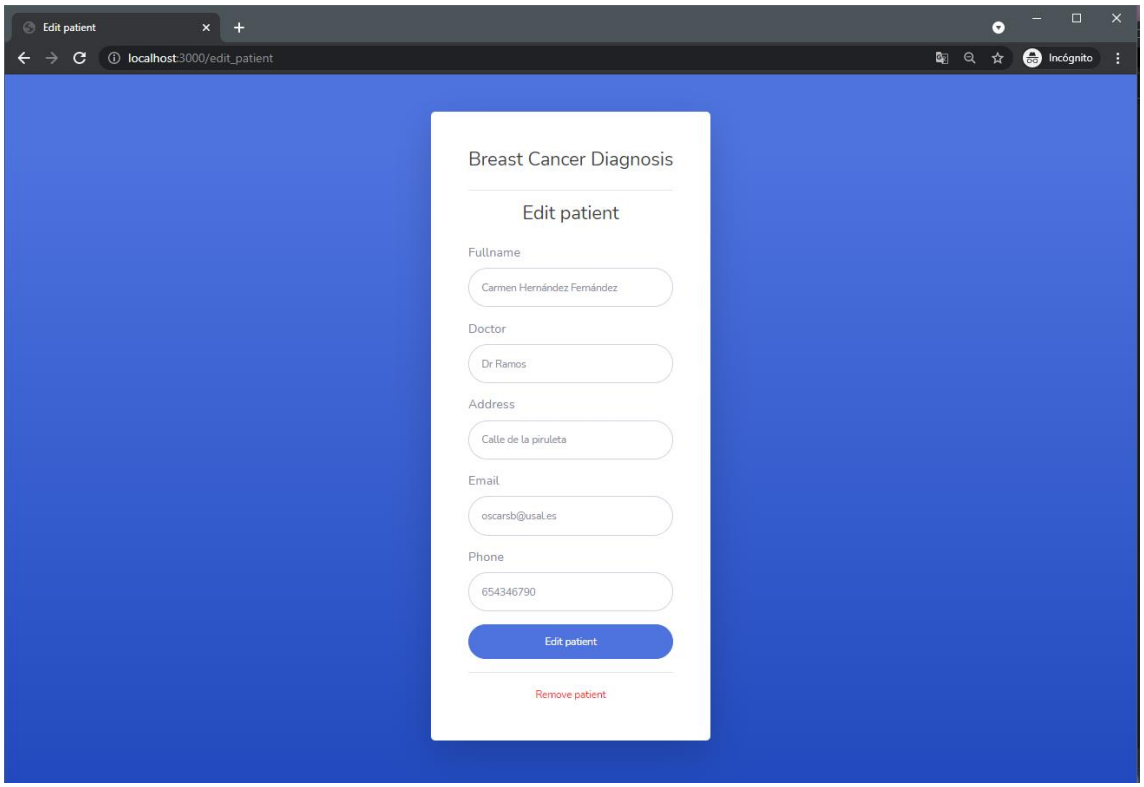


Ilustración 59: Editar paciente

Además, en esta misma página de la "Ilustración 59: Editar paciente" se puede eliminar de la aplicación, mostrando un mensaje de confirmación como el de la "Ilustración 60: Eliminar paciente confirmación".

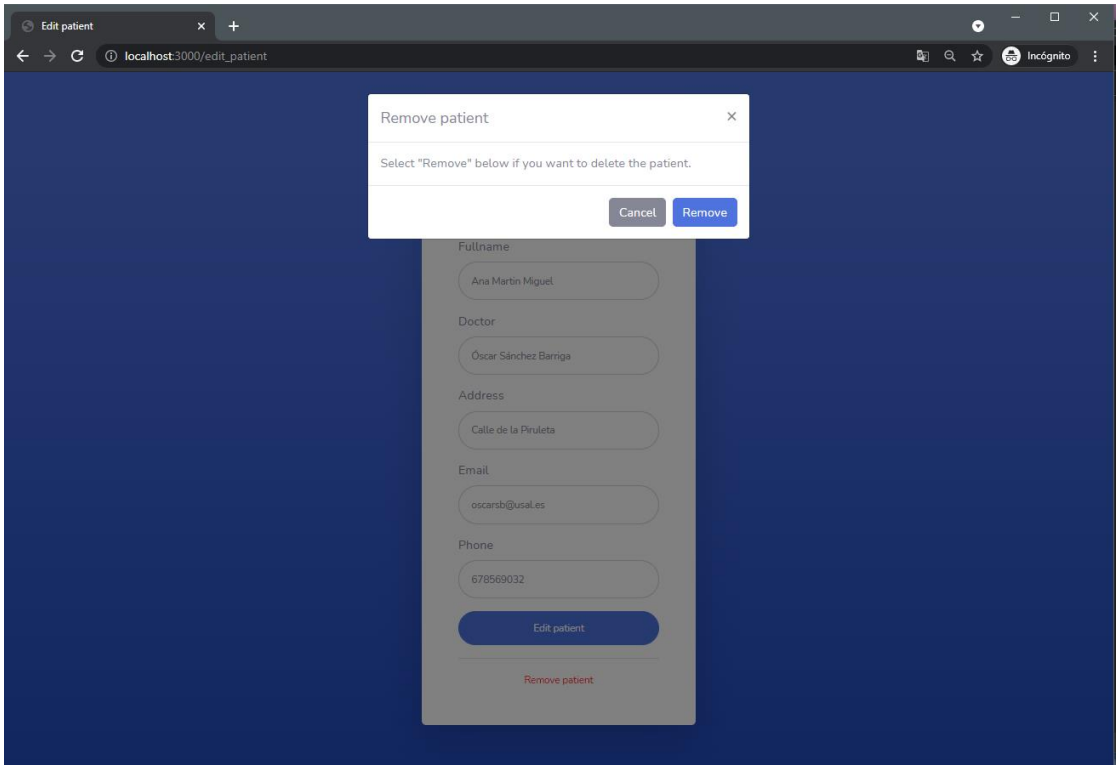


Ilustración 60: Eliminar paciente confirmación

Para realizar un análisis a una paciente se debe haber seleccionado previamente como se ha visto en la “Ilustración 57: Seleccionar un paciente”. Una vez este seleccionada se pulsa sobre el botón ‘Select patient’ y se le realizará el análisis adjuntando imágenes de la paciente, como se puede ver en la “Ilustración 61: Añadir imágenes”.

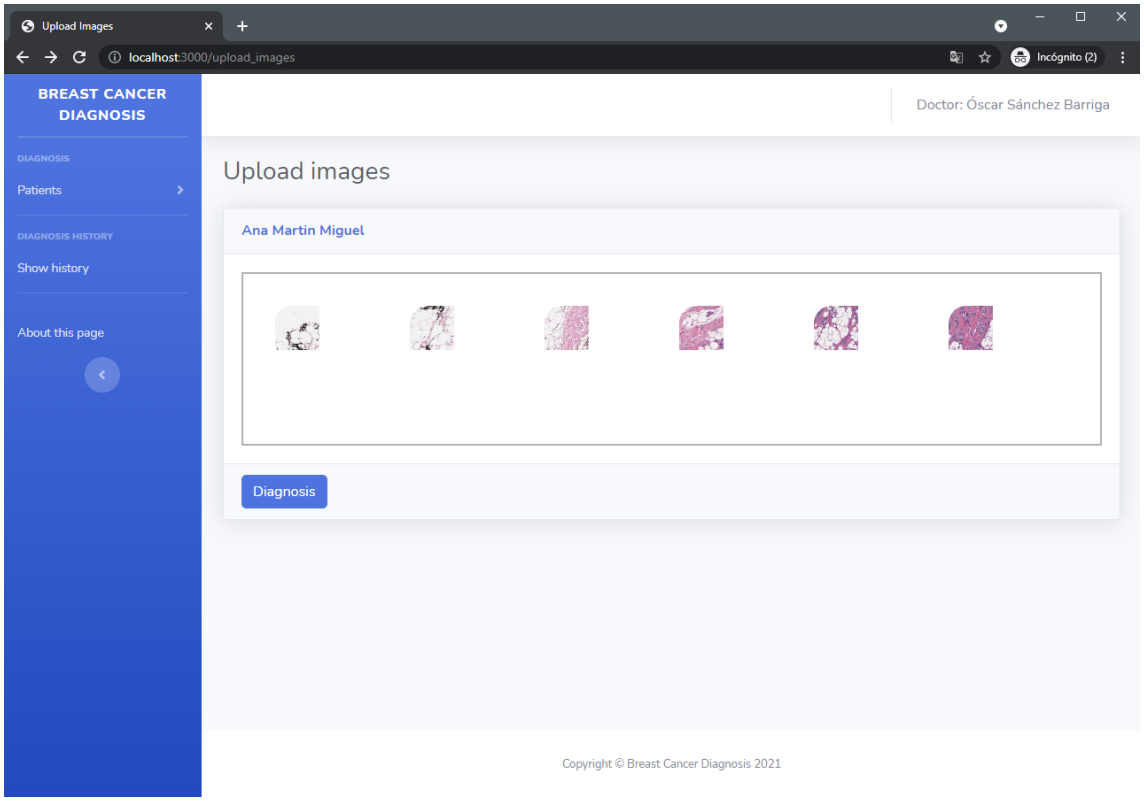


Ilustración 61: Añadir imágenes

Cuando se tengan añadidas todas las imágenes que se desean, se pulsa en el botón ‘Diagnosis’ y la red neuronal las realizará el diagnóstico, mostrando el resultado de cada una de ellas, como se puede observar en la “Ilustración 62: Diagnóstico de imágenes”.

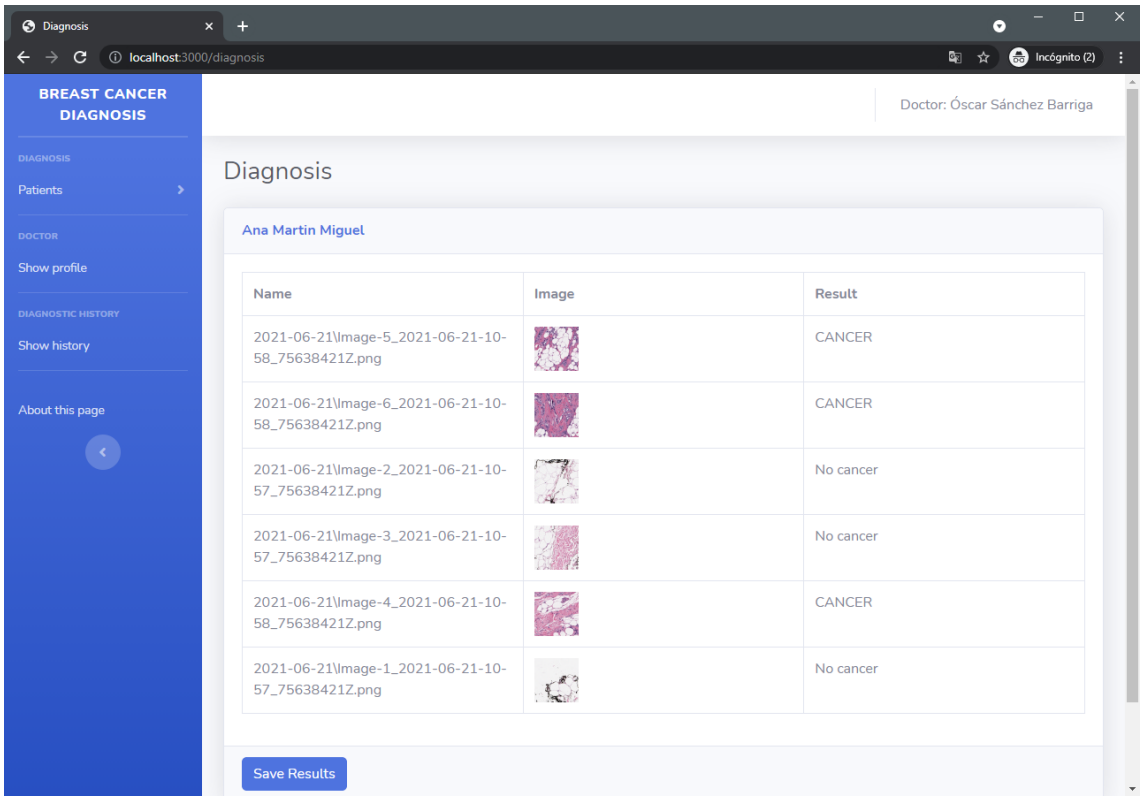


Ilustración 62: Diagnóstico de imágenes

Por último, pulsando en el botón 'Save Results' de la "Ilustración 62: Diagnóstico de imágenes", la aplicación mostrará los resultados de la paciente, dando la opción de enviar un correo electrónico a esta o volver a la página principal de la aplicación. Esto se puede visualizar en la "Ilustración 63: Resultados del diagnóstico".

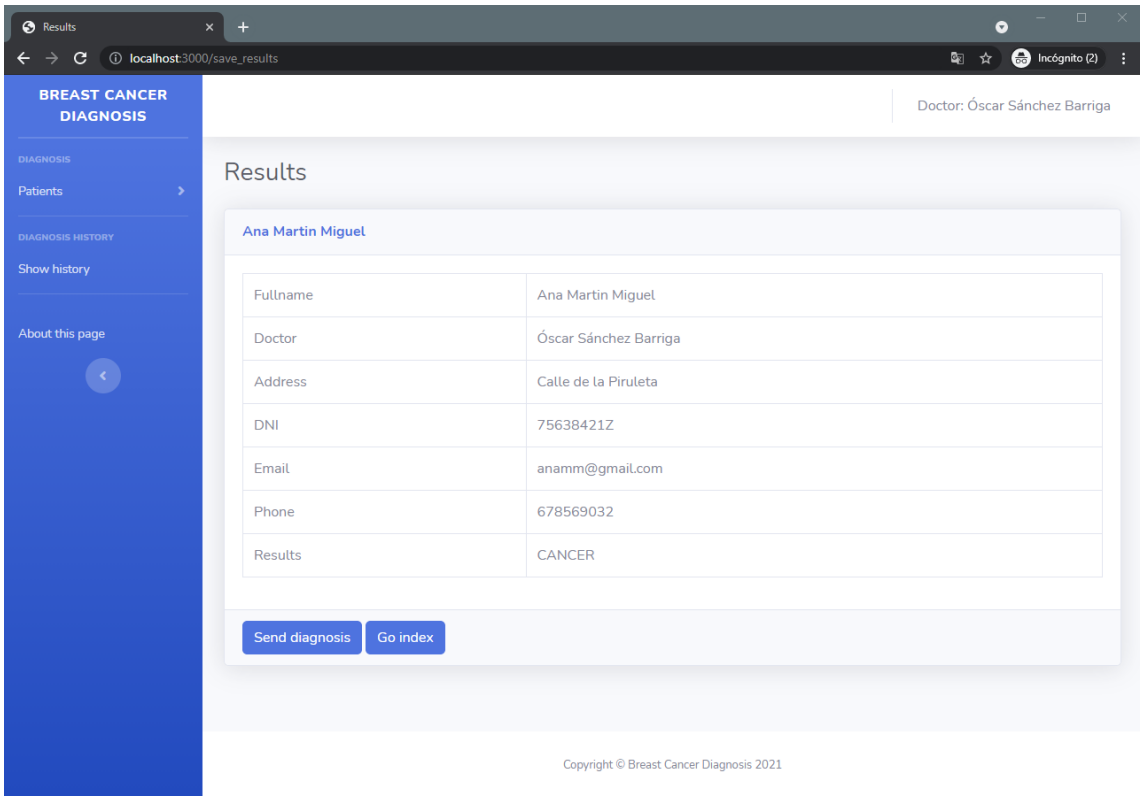


Ilustración 63: Resultados del diagnóstico

El correo electrónico que le llegará a la paciente tendrá un aspecto como el de la "Ilustración 64: Correo electrónico con el diagnóstico médico".

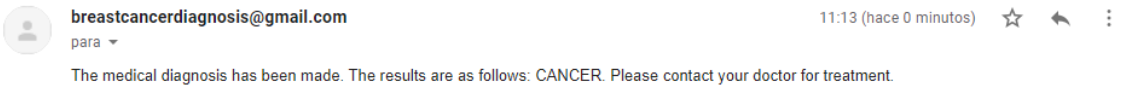


Ilustración 64: Correo electrónico con el diagnóstico médico

5.9.3 Diagnósticos

La aplicación permite diversas acciones con los diagnósticos permitiendo así consultar, modificar manualmente, eliminar y volver a realizar un diagnóstico con la IA.

A continuación, se adjuntan diferentes ilustraciones:

En la página principal, se muestra un botón llamado 'Show history', como se observa en la "Ilustración 65: Index Ver historial", ofreciendo la posibilidad de consultar un historial de diagnósticos.

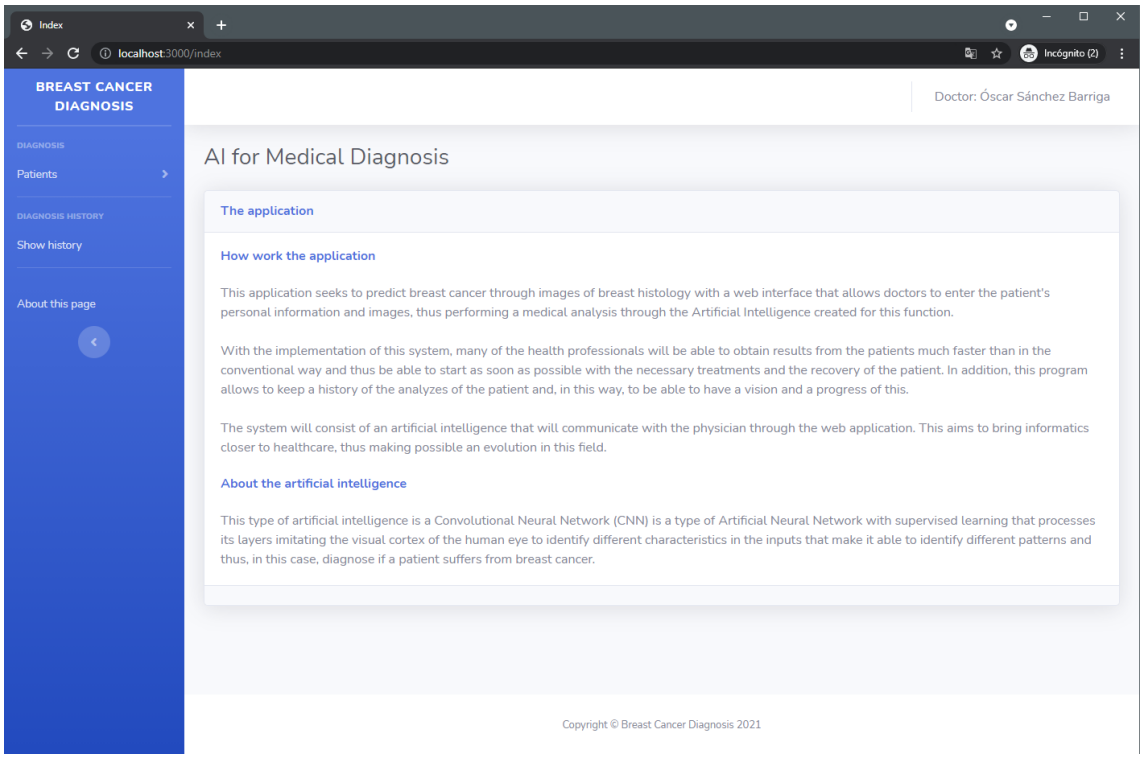


Ilustración 65: Index Ver historial

A continuación de haber pinchado, se muestra una pantalla como la de la "Ilustración 66: Historial", en la que se debe introducir el DNI del diagnóstico de la paciente que se desea consultar, así como seleccionar una de las fechas que ofrece la aplicación.

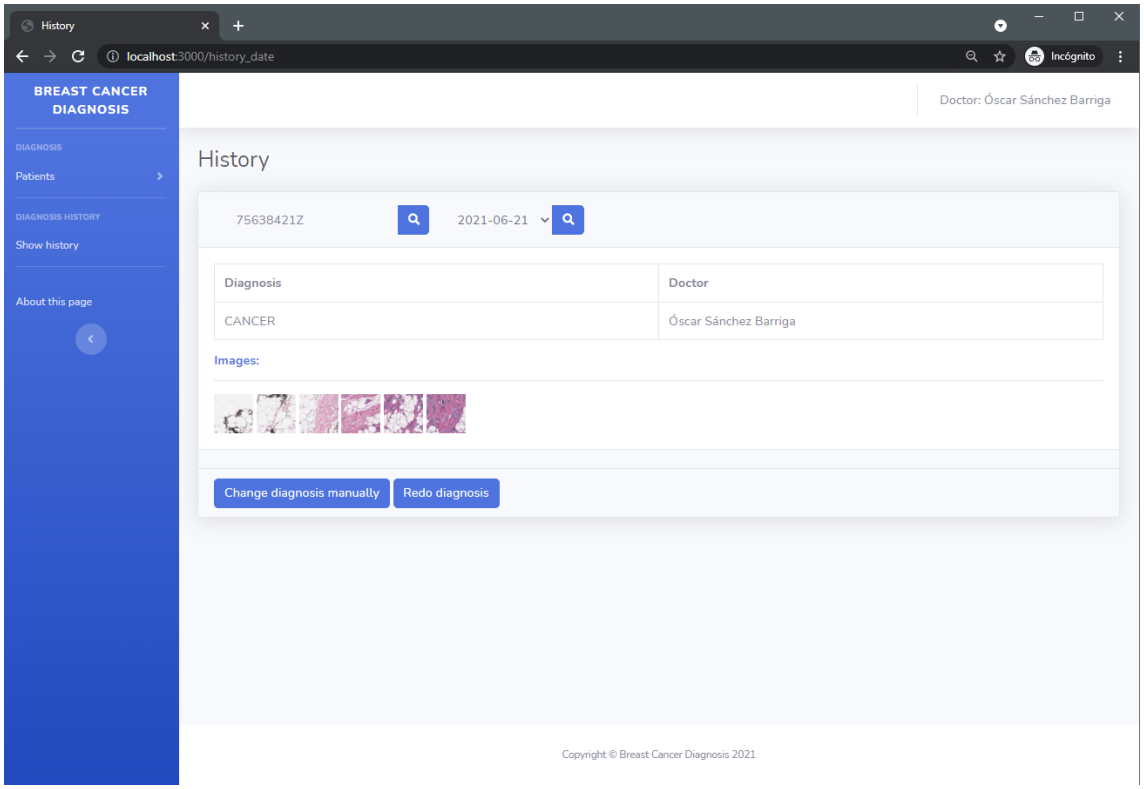


Ilustración 66: Historial

Como se puede ver en la “Ilustración 66: Historial”, se puede volver a realizar el diagnóstico con la red neuronal pulsando en ‘Redo diagnosis’ o se puede cambiar manualmente los resultados para ese diagnóstico o borrarlo, pulsando sobre el botón ‘Change diagnosis manually’.

Si el médico desea cambiar el diagnóstico manualmente, en la página web de la “Ilustración 67: Cambiar diagnóstico manualmente”, podrá realizar la tarea.

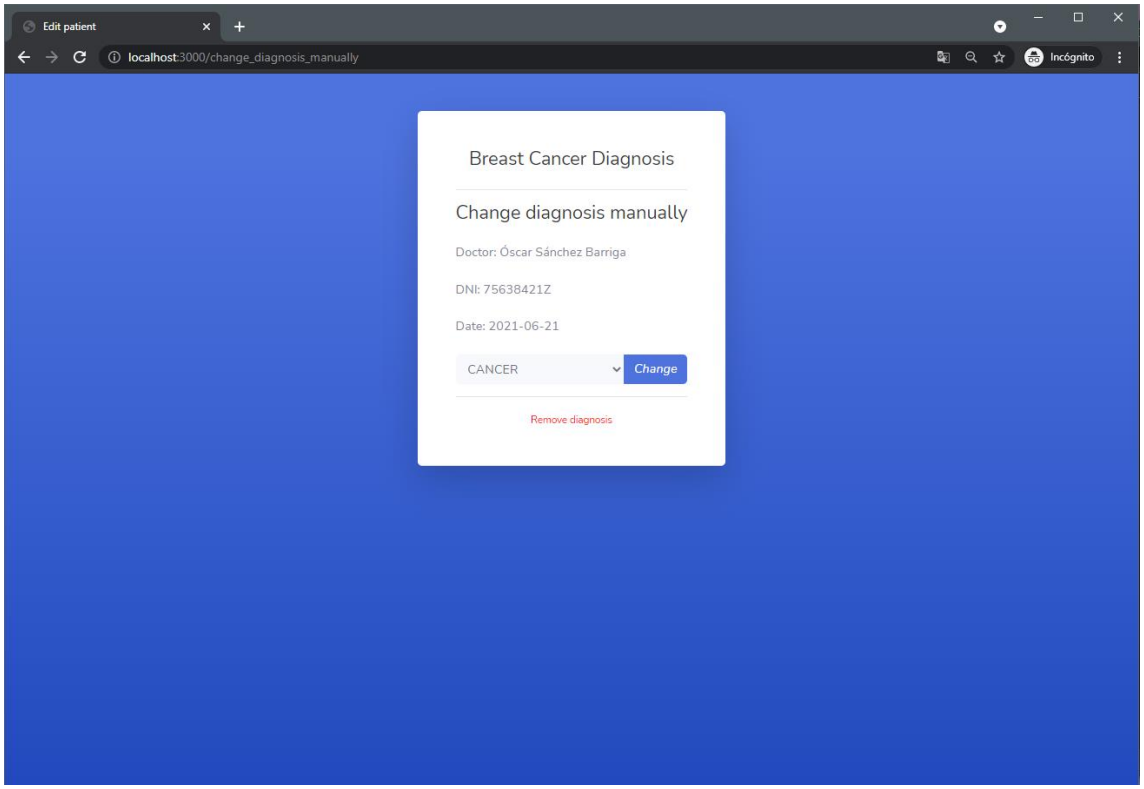


Ilustración 67: Cambiar diagnóstico manualmente

De igual forma, como se puede observar en la “Ilustración 67: Cambiar diagnóstico manualmente” se ofrece la posibilidad de borrar el diagnóstico, junto con las imágenes utilizadas para ese análisis.

Por el contrario, si el médico desea volver a realizar el análisis con la red neuronal como la de la “Ilustración 68: Volver a realizar diagnóstico”. En ella se pueden ver los distintos resultados por cada imagen del diagnóstico, pudiendo guardar los nuevos resultados.

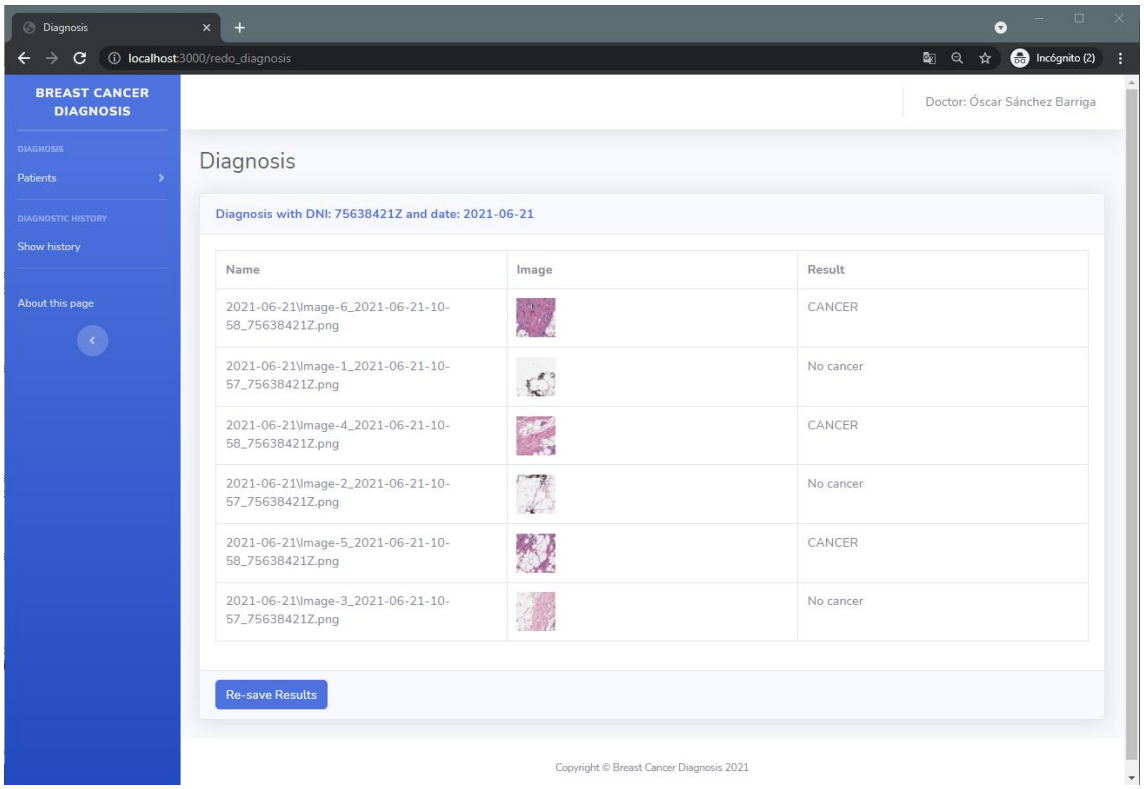


Ilustración 68: Volver a realizar diagnóstico

6. Conclusión y líneas de trabajo futuras

6.1 Conclusión

Del desarrollo de este proyecto se pueden extraer diversas conclusiones, tanto a nivel personal como a nivel técnico y profesional.

Los objetivos técnicos conseguidos:

- Se ha conseguido un banco de imágenes de histopatología mamaria con el cual, gracias a su gran número de elementos, ha servido para poder entrenar a la inteligencia artificial de una forma satisfactoria.
- Se ha conseguido desarrollar una red neuronal capaz de predecir de forma correcta si una paciente tiene cáncer de mama o, en cambio, está sana.
- Se ha conseguido desarrollar una aplicación que, al tratarse de una web, puede dar servicio a todos los profesionales sanitarios sin necesidad de que en sus equipos se gasten recursos innecesarios.
- Se ha implementado una base de datos para guardar los datos de los distintos médicos, pacientes y diagnósticos.
- Se ha desarrollado un sistema de búsqueda en la página web para poder encontrar a las pacientes y sus diagnósticos en la base de datos.
- Se ha creado un sistema de ficheros para guardar de forma correcta las distintas imágenes de histopatología mamaria de las pacientes.

En cuanto a los resultados de la predicción de la inteligencia artificial, se ha podido observar un gran progreso. Partiendo de un *dataset* pequeño con una red neuronal de pocas capas, se observaba como esta tenía fallos de predicción, pero a medida que se ha aumentado el número de imágenes con las que se entrena y la complejidad del modelo, se ha logrado obtener una inteligencia artificial capaz de predecir con gran exactitud si una paciente padece una enfermedad. Esto se debe al uso de las librerías de TensorFlow y Keras, siendo estas de las más extendidas en cuanto a la predicción de imágenes, haciendo posible así el desarrollo de la red neuronal. Aún así existe un margen de mejora para la sensibilidad, puesto que esta tiene un 75,10%. Se podría conseguir aumentar este con la ayuda de un conjunto de datos más complejo.

Cabe mencionar que durante el desarrollo se han adquirido conocimientos acerca del esfuerzo que conlleva el desarrollo de un proyecto y ver la utilidad de las metodologías y las diversas herramientas usadas en el transcurso del trabajo.

También hay que destacar las tecnologías aprendidas durante el transcurso del trabajo, como pueden ser Python, Flask, todo lo relacionado con la inteligencia artificial y el *Deep Learning*, entre otros. Conocer la gran comunidad software que hay, valorando así el gran trabajo que realizan los profesionales en este sector, que hacen posible que el desarrollo de proyectos tenga una mayor viabilidad y se obtengan mejores resultados.

Otra de las razones por las que la realización de este trabajo de fin de grado ha sido de gran utilidad es que se ha podido repasar todas las asignaturas de la carrera y profundizar en los conocimientos adquiridos de las mismas.

6.2 Líneas de trabajo futuras

Debido a que el proyecto permite extender su funcionalidad, se plantean líneas de investigación futuras, junto con nuevas ideas de desarrollo.

- Agregar más campos de información de las pacientes y diagnósticos.
- Reducir los falsos negativos e incluir otras métricas de evaluación de la red neuronal.
- Crear más inteligencias artificiales, capaces de predecir un mayor número de tipos de cáncer, así como de enfermedades. De esta forma se crearía una aplicación más completa en el diagnóstico de patologías médicas.
- Agregar la opción de elegir idioma, haciendo de esta forma una aplicación multilenguaje.
- Permitir a un paciente pedir cita o ponerse en contacto con su médico a través de la plataforma.
- Usar el historial de diagnósticos para poder obtener distintas estadísticas.

7. Referencias

[1] «AECC, cáncer de mama,» [En línea]. Available: https://www.aecc.es/es/todo-sobre-cancer/tipos-cancer/cancer-mama/mas-informacion/evolucion-cancer-mama.

[2] «Inteligencia artificial,» [En línea]. Available: https://searchdatacenter.techtarget.com/es/definicion/Inteligencia-artificial-o-AI.

[3] «Machine Learning,» [En línea]. Available: https://www.arimetrics.com/glosario-digital/machine-learning.

[4] «Redes neuronales Dot CSV,» [En línea]. Available: https://www.youtube.com/watch?v=MRIv2IwFTPg&list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ&ab_channel=DotCSV.

[5] «Convolutional Neural Network,» [En línea]. Available: https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/.

[6] «Visual Studio Code,» [En línea]. Available: https://code.visualstudio.com/.

[7] «PyCharm,» [En línea]. Available: https://www.jetbrains.com/es-es/pycharm/.

[8] «Google Colaboratory,» [En línea]. Available: https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index.

[9] «Google Drive,» [En línea]. Available: https://www.google.com/intl/es_es/drive/.

[10] «XAMPP,» [En línea]. Available: https://www.apachefriends.org/es/index.html.

[11] «Pydoc,» [En línea]. Available: https://docs.python.org/es/3/library/pydoc.html.

[12] «Visual Paradigm,» [En línea]. Available: https://www.visual-paradigm.com/.

[13] «Microsoft Project,» [En línea]. Available: https://support.microsoft.com/es-es/office/instalar-project-7059249b-d9fe-4d61-ab96-5c5bf435f281.

[14] «Flask,» [En línea]. Available: https://flask.palletsprojects.com/en/2.0.x/.

[15] «Python,» [En línea]. Available: https://www.python.org/.

[16] «TensorFLow,» [En línea]. Available: https://www.tensorflow.org/?hl=es-419.

[17] «Keras,» [En línea]. Available: https://keras.io/.

[18] «SB Admin 2,» [En línea]. Available: https://startbootstrap.com/theme/sb-admin-2.

[19] «Breast Histopathology Images,» [En línea]. Available: https://www.kaggle.com/paultimothymooney/breast-histopathology-images.

8. Bibliografía

Kaggle

https://www.kaggle.com/

Tutorial para detectar COVID-19

https://www.pyimagesearch.com/2020/03/16/detecting-covid-19-in-x-ray-images-with-keras-tensorflow-and-deep-learning/

Guardar y cargar un modelo – Keras

https://www.pyimagesearch.com/2018/12/10/keras-save-and-load-your-deep-learning-models/

Como hacer una red neuronal convolucional

https://www.youtube.com/watch?v=pEC4FtwUgLc

Explicación aprendizaje profundo

https://www.youtube.com/watch?v=aircAruvnKkç

Precisión y pérdida

https://docs.paperspace.com/machine-learning/wiki/accuracy-and-loss

Tutoriales para Python

https://www.youtube.com/watch?v=YYXdXT2l-Gg&list=PL-osiE80TeTt2d9bfVyTiXJA-UTHn6WwU

Tutorial para Flask

https://www.youtube.com/watch?v=MwZwr5Tvyxo&list=PL-osiE80TeTs4UjLw5MM6OjgkjFeUxCYH