E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Aplicación web para la recogida y consulta de información de lesiones colorrectales de pacientes

Grado en Ingeniería Informática

Trabajo Fin de Grado

Juana María Guerra Trapiella

Nicanor Humberto Bustince, Iosu Rodríguez Martínez

Pamplona, 6/09/2021

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Desarrollo de una aplicación web para la recogida y consulta de información de lesiones colorrectales de pacientes

Resumen

Este proyecto surgió tras una propuesta realizada por Navarrabiomed, en concreto por la unidad de investigación en endoscopia digestiva. Se buscaba obtener una herramienta de almacenamiento de información clínica, morfológica y patológica de sus pacientes en diferentes momentos: evaluación de lesiones, evaluación de cicatrices, evaluación de cicatrices si se presenta alguna complicación, seguimiento pasados 3-6 meses y seguimiento pasados 15-18 meses. Además de que posibilitase adjuntar imágenes endoscópicas asociadas a cada momento.

Este proyecto se centra en la creación de dicha aplicación web, así como en la creación de la base de datos MySQL. En esta se almacenará la información clínica clave de los pacientes en sus diferentes momentos junto a sus imágenes endoscópicas.

El objetivo es que esta aplicación sea el primer paso dentro de un proyecto a mayor escala que explotará la información recolectada tras su uso. En el proyecto a mayor escala se aplicarán técnicas de IA para mejorar la precisión de los diagnósticos y la eficacia y seguridad de los tratamientos endoscópicos que se aplican. Toda esta idea conjunta ha sido premiada con el primer premio del “Concurso Medtech Navarra” [1].

Se ha aplicado un desarrollo iterativo e incremental en el que han tenido un papel fundamental las entrevistas realizadas con los médicos del Equipo de Navarrabiomed al frente del proyecto. Además, se ha elegido su desarrollo en el Framework PHP Laravel ya que posee numerosas posibilidades en su uso, las cuales han sido empleadas siendo muy útiles.

Palabras clave: aplicación web, Laravel, medicina personalizada, endoscopia digestiva.

Tabla de contenidos

1. Introducción

1.1 Motivación

El centro de investigación biomédica Navarrabiomed busca mejorar la eficacia y seguridad de los tratamientos endoscópicos que aplican a sus pacientes que sufren neoplastias gastrointestinales. También busca obtener una predicción más precisa de posibles complicaciones después de haberles realizado un procedimiento llamado resección endoscópica mucosa (MRE). Además, ha realizado una investigación exhaustiva respecto a las posibles complicaciones que puede sufrir un paciente en función de ciertos parámetros y factores clínicos, morfológicos y patológicos [2, 3].

Navarrabiomed planteó esta necesidad al Grupo de Inteligencia Artificial y Razonamiento Aproximado (GIARA) de la UPNA. A partir de esta investigación previa realizada, surgió la idea de la creación de una aplicación que permitiese recopilar estos factores y parámetros. Si además de recopilar estos datos, se incluía la posibilidad de recolectar imágenes endoscópicas, toda esta información podría ser utilizada para realizar predicciones más precisas aplicando técnicas de Inteligencia Artificial.

La idea de este proyecto a gran escala ha sido premiada con el primer premio del "Concurso Medtech Navarra. Nuevas ideas de investigación en el ámbito de la salud 2021" [1], en el que se planteó el proyecto de investigación "Inteligencia Artificial (IA) y Neoplastias Gastrointestinales". En este se propuso la idea de creación de una aplicación de IA para la optimización del tratamiento endoscópico de las neoplastias gastrointestinales, así como la predicción y anticipación de potenciales complicaciones.

Para poder llevar a cabo esta idea, se acordó la realización de un primer proyecto. Este primer proyecto representa el primer paso para poder alcanzar el último objetivo de mejora de predicción de las complicaciones y de los tratamientos. En este se busca obtener una herramienta web que permita la recopilación anónima de datos de pacientes. También debe permitir recopilar imágenes endoscópicas, que permitirán ampliar de una manera más extensa la información.

De esta manera surgió el proyecto que a continuación se presenta. Consiste en la creación de una herramienta web que permita la recopilación, consulta y modificación de datos clínicos, patológicos y/o morfológicos pertenecientes a pacientes con posibles lesiones colorrectales. Además, debe permitir adjuntar imágenes de las posibles lesiones. Toda esta información debe ser recopilada y accesible por los especialistas de diferentes centros médicos.

1.2 Objetivos

El objetivo principal de este proyecto es la creación de una página web cuya funcionalidad principal es la recopilación de datos clínicos, morfológicos y/o patológicos de pacientes con

posibilidad de sufrir lesiones colorrectales. Además de ofrecer la posibilidad de adjuntar sus imágenes endoscópicas.

En la recopilación de la información se debe mantener constantemente la anonimidad de los pacientes. De manera que no se debe poder identificar a los pacientes dentro de la información almacenada. La Ley Orgánica de Protección de Datos [4] advierte de los riesgos de tratar con este tipo de datos médicos.

Además, los datos de los pacientes que hayan sido dados de alta en un centro hospitalario podrán ser accesibles por los médicos especialistas pertenecientes a ese centro. Posibilitando que diferentes centros puedan acceder a la aplicación. Cada médico especialista podrá modificar o añadir nueva información según el seguimiento del paciente en diferentes momentos. De esta forma esta herramienta debe posibilitar una gestión más sencilla y eficaz de la información recopilada de los pacientes.

Para poder cumplir este objetivo principal, se han planteado cinco subobjetivos que garantizan el total cumplimiento del objetivo principal de este proyecto:

- Crear una aplicación web que podrá ser utilizada por médicos de distintos centros hospitalarios. Su utilización debe ser de forma sencilla e intuitiva. De esta manera podrán realizar el seguimiento de sus pacientes en diferentes momentos en los que se desea almacenar información diferente: evaluación de lesiones, evaluación de cicatrices, evaluación de cicatrices si se presenta alguna complicación, seguimiento 3-6 meses después y seguimiento 15-18 meses después.

- Añadir a la aplicación web la funcionalidad de añadir y borrar, en cada momento, imágenes endoscópicas de los pacientes. En el futuro proyecto de Inteligencia Artificial, se podrán aplicar técnicas de tratamiento de imagen para ayudar a la predicción de posibles complicaciones del paciente.

- Diseñar la correspondiente base de datos MySQL a partir de la documentación recibida por parte del Equipo de Navarrabiomed. Esta documentación contiene las variables más importantes que son necesarias incluir en las tablas de la base de datos.

- Configurar un sistema de autenticación de forma que sólo los médicos especialistas que tengan permiso de acceso puedan emplear la aplicación web.

- Realizar la puesta en producción de la aplicación web, incluyéndola en la plataforma “GIMLI” (Giara IMage Labelling Infastructure) [5]. El objetivo es llevar a cabo un despliegue, como una aplicación más dentro de una colección de aplicaciones de asistencia a proyectos médicos. Esta plataforma permite recoger datos a tratar para su posterior análisis y explotación.

1.3 Estructura de la memoria

A continuación, se presenta a modo de resumen, los siguientes bloques en los que está dividida esta memoria para ofrecer una vista general y presentar los temas a tratar:

- Estado del arte: se mencionan algunas aplicaciones existentes y se indica la aportación de este TFG en relación con las aplicaciones existentes.

- Proceso de desarrollo software: se menciona el proceso de desarrollo software elegido. Así como la razón por la que se eligió, qué ventajas presenta frente a otros tipos de desarrollo y cómo se ha llevado a cabo.

- Análisis de requisitos: se menciona la forma en la que se recolectaron los requisitos. Cuáles fueron los requisitos que se obtuvieron clasificándolos en requisitos funcionales (RF) y no funcionales (RNF). Se incluye también su fase de modelado mediante diagramas de casos de uso y una explicación más extensa de los casos de uso obtenidos.

- Diseño: se presenta el modelo lógico de la base de datos creada, así como la explicación de cómo se relacionan las diferentes tablas. Se presenta una descripción del patrón arquitectónico utilizado y los motivos por los que se eligió.

- Tecnologías y herramientas: se citan las herramientas y tecnologías utilizadas en el desarrollo del proyecto software y qué ventajas presentan frente a otras opciones.

- Implementación: se presenta la estructura del proyecto Laravel creado para el desarrollo de la aplicación. Se explica, directorio por directorio, su contenido. Además, se mencionan las ventajas ofrecidas por el framework Laravel que han sido utilizadas para facilitar la fase de implementación de la aplicación.

- Aplicación final: se muestra el proyecto final mediante una serie de capturas donde se pueden observar las diferentes vistas del proyecto. Al mismo tiempo, se presentan las diferentes funcionalidades que ofrece.

- Conclusiones: se realiza una reflexión final sobre cómo ha sido el proceso de desarrollo del proyecto y qué asignaturas impartidas en este grado han sido totalmente necesarias para poder cumplir con los objetivos propuestos. También se mencionan las líneas de mejora y posibles proyectos que pueden surgir a partir de este.

- Referencias: se listan las referencias a los recursos consultados.

2. Estado del arte

En este apartado se mencionan algunas aplicaciones existentes relacionadas con este proyecto y cuál es la aportación de este proyecto frente a aplicaciones que se estén utilizando en este momento.

2.1 Aplicaciones similares

La aplicación de la Inteligencia Artificial en la medicina es cada vez mayor. Las razones de este hecho son las numerosas ventajas que ofrece en el campo de la salud. Entre estas cabe destacar:

- Facilita el control y seguimiento de los pacientes ya que permite procesar y analizar datos médicos con los que mejorar la gestión sanitaria. De esta forma alivia la carga de trabajo a profesionales médicos.

- Permite obtener unos diagnósticos más precisos y rápidos. En general permite mejorar la calidad de vida de los pacientes.

- Permite agilizar los tiempos de investigación en el desarrollo de nuevos fármacos y sobre determinadas enfermedades o lesiones.

Dos ejemplos de aplicaciones existentes que emplean la IA para la mejora de diagnósticos en sus pacientes y que están estrechamente relacionadas con el objetivo a gran escala de este proyecto son:

- “Desarrollo y validación de un algoritmo de detección automática basado en aprendizaje profundo para nódulos pulmonares malignos en radiografías de tórax” [6]”. Este algoritmo analiza radiografías de tórax y detecta un crecimiento celular anormal, como posibles cánceres.

- “Detección de metástasis ganglionar del cáncer de mama basada en inteligencia artificial” [7]. Este algoritmo permite identificar tumores metastásicos de cáncer de mama a partir de biopsias de ganglios linfáticos. Permitiendo identificar regiones sospechosas indistinguibles para el ojo humano en las muestras de biopsia proporcionadas.

2.2 Aportación de este proyecto

Navarrabiomed hace uso en la actualidad de una web Redcap [8] en la que se almacena los datos clínicos, morfológicos y patológicos de sus pacientes. Esta es una aplicación web segura que permite la construcción y manejo de encuestas en línea y bases de datos. Está especializada en la captura de datos para estudios de investigación.

La web mencionada anteriormente se ha utilizado como punto de partida para el desarrollo de esta aplicación. Los parámetros y factores que se han establecido necesarios recopilar en este proyecto han sido obtenidos de esta web.

Este proyecto ampliará las funcionalidades de la web Redcap de la que disponen hasta el momento. Además de permitir almacenar los datos clínicos, morfológicos y patológicos de sus pacientes, posibilita que se adjunten todas las imágenes endoscópicas disponibles en cada momento. Pudiendo añadir nuevas imágenes siempre que se disponga de nuevas imágenes o eliminarlas en caso de que así se desee. El objetivo es que estas sean explotadas con técnicas de tratamiento de imágenes. Para poder obtener una predicción de posibles complicaciones más precisa y tratamientos más efectivos.

Además, posibilita que varios centros médicos a nivel nacional puedan almacenar toda su información. Actuando, por tanto, como un banco de datos médicos, a partir del cual podrán surgir futuros proyectos de Deep Learning y/o minería de datos.

3. Tecnologías y herramientas

En este apartado se presentan los lenguajes y herramientas empleadas en el desarrollo de la aplicación.

3.1 Lenguajes

- **HTML:** o Lenguaje de Marcado de Hipertexto. Este lenguaje se escribe en su totalidad con elementos que crean la estructura del contenido de una página web o aplicación. Estos elementos a su vez están constituidos por etiquetas, contenido y atributos. Es el lenguaje que interpreta el navegador web para poder mostrar los sitios o aplicaciones web [9].

- **PHP:** lenguaje de programación de código abierto y en constante perfeccionamiento destinado a la creación de páginas web o al desarrollo de aplicaciones para web. Favorece la conexión entre los servidores y el interfaz de usuario [10]. En este proyecto se ha utilizado PHP porque es el lenguaje principal de desarrollo en servidores.

- **CSS:** es el lenguaje para describir la presentación de las páginas web, incluidos los colores, el diseño y las fuentes. Permite adaptar los diseños para diferentes tipos de aparatos como grandes pantallas, pequeñas pantallas o impresoras. CSS es independiente de HTML. Su separación permite un mantenimiento más sencillo de las webs, compartir hojas de formato a través de estas y adaptar páginas a diferentes entornos [11]. En el proyecto se ha empleado CSS para crear los estilos de las vistas.

- **JavaScript:** es un lenguaje de programación o una secuencia de comandos que permite implementar funciones complejas en páginas web como actualizaciones dinámicas de contenido, animación de imágenes y mucho más [12]. En este proyecto se ha utilizado

JavaScript porque ha permitido añadir funcionalidades interactivas en la aplicación web, como permitir mostrar y ocultar campos en los formularios.

3.2 Entorno de desarrollo

- **Visual Studio Code:** es un editor de texto plano desarrollado por Microsoft de código abierto para ofrecer a los usuarios una herramienta de programación avanzada como alternativa al Bloc de Notas. Este editor está escrito en forma de aplicación de escritorio. No se caracteriza precisamente por un bajo consumo de memoria (ya que tiene que cargar todo el core de Chrome), pero es muy sencillo de programar, muy potente y flexible [13]. Se eligió este editor porque se puede utilizar con los principales lenguajes de programación web, además de contar con una terminal integrada. Además, ofrece la posibilidad de tener más de un código visible o carpetas al mismo tiempo.

- **XAMPP:** es una distribución de Apache fácil de instalar que contiene MariaDB, PHP y Perl [14]. XAMPP se ha utilizado como la principal plataforma de desarrollo a nivel local de la aplicación.

3.3 Tecnologías software

- **Framework:** o marco de trabajo software. Es una plataforma concreta o conceptual donde código común con funcionalidades genéricas puede ser especializado o sobrescrito por desarrolladores o usuarios. Los frameworks toman la forma de librerías, donde una interfaz de programación de aplicaciones (API) bien definida es reutilizada en cualquier fragmento del software que está siendo desarrollado.

- **Laravel:** es un framework PHP. Es uno de los frameworks más utilizados y de mayor comunidad en el mundo de internet. Como framework resulta bastante moderno y ofrece muchas utilidades potentes a los desarrolladores. Permitiendo agilizar el desarrollo de las aplicaciones web. Laravel ofrece un entorno de desarrollo altamente funcional, así como interfaces de línea de comandos intuitivas y expresivas. Los desarrolladores también pueden añadir funcionalidad a sus aplicaciones sin problemas, gracias su sistema de empaquetado modular y a la sólida gestión de dependencias [15]. En el apartado 7.1 se presentan las ventajas que ofrece este entorno de desarrollo y como se han utilizado.

- **MariaDB**: es uno de los servidores de bases de datos más populares a nivel mundial. Originalmente diseñado como una mejora, creada para sustituir a MySQL. MariaDB es usado porque es rápido, escalable y robusto. Posee un rico entorno de motores de almacenamiento, plugins y muchas otras herramientas que la hacen versátil para una amplia variedad de casos de uso. Ha sido desarrollado por un software libre, siendo una base de datos relacional que provee una interfaz SQL para el acceso a los datos. [16].

MariaDB ha sido utilizado como principal gestor de base de datos del proyecto, debido a que es una mejora frente a MySQL.

- **Eloquent:** es el ORM propio de Laravel. Un ORM es una librería que define el código necesario para la manipulación de los datos de una base de datos, mediante el lenguaje propio de una librería, sin tener que emplear el lenguaje SQL [17]. La forma en la que se ha trabajado con Eloquent en este proyecto queda explicada en el primer punto del apartado 7.1.

4. Proceso de desarrollo Software

4.1 Proceso software Iterativo e Incremental

Un ciclo de vida en cascada divide un proyecto en diferentes fases:

1. Especificación de requisitos.
2. Análisis.
3. Diseño.
4. Implementación.
5. Pruebas.
6. Instalación y mantenimiento.

Cada fase tiene un orden de realización y se realiza una única vez. Una fase no puede iniciarse hasta que no termine con éxito la que lo precede. La principal razón por la cual no se eligió este tipo de desarrollo en este proyecto fue que al principio del proyecto no se podía tener una visión completa de todos los requisitos. Además de que estos podrían necesitar alguna modificación en fases más avanzadas.

El ciclo de vida iterativo e incremental propone una solución a las limitaciones del ciclo de vida en cascada. Este ciclo de vida se puede describir de la siguiente manera: el proyecto se planifica en múltiples iteraciones (bloques temporales) y repetitivas.

En cada iteración se libera un prototipo (teniendo desde la primera iteración un producto funcional para el usuario final). Además de que en cada iteración se debe añadir nuevas funcionalidades, se debe aportar una mayor calidad con respecto a la iteración anterior. El último objetivo de este ciclo de vida es llegar a la última iteración en la que se realiza la entrega final del proyecto. La Figura 1 ilustra las distintas fases de este tipo de desarrollo.

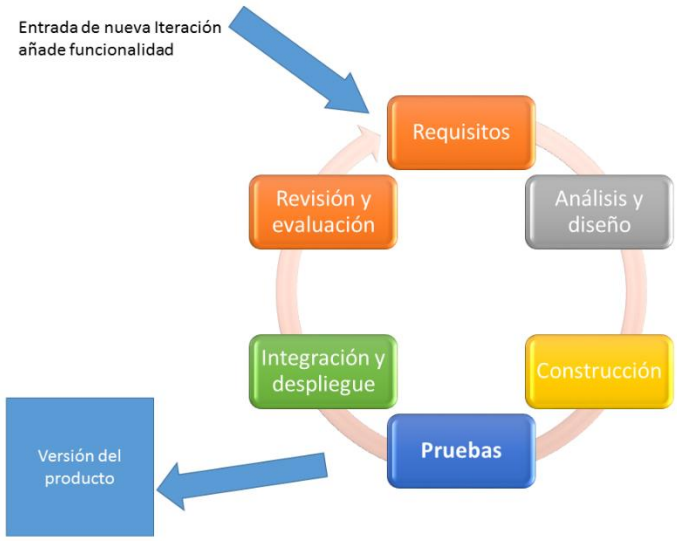


Fig. 1 Etapas del Desarrollo Iterativo e Incremental

Se eligió un desarrollo iterativo e incremental porque nos permite entregar valor al usuario antes de acabar todo el desarrollo. En cada demo del incremento el usuario final devuelve sus impresiones, respuestas o sugerencias. Dependiendo de estas se añadirán nuevos requisitos o se modificarán los ya existentes.

En cada iteración se realiza una revisión del prototipo entregado. Los errores detectados pueden solucionarse en la próxima iteración y los nuevos requisitos serán reorganizados según el valor que aporte al usuario.

4.2 Ventajas

En este apartado se mencionan aquellas ventajas que ofrece un desarrollo iterativo e incremental que han sido utilizadas en este proyecto:

- Permite gestionar las expectativas del cliente: los médicos al frente del proyecto informático tenían una idea general de lo que querían al comienzo del proyecto, que fijamos como nuestros “requisitos de alto nivel”. En cada iteración se pudieron ir refinando los requisitos, apareciendo nuevos requisitos que hasta el momento no se habían planteado. Siempre realizando una validación de los requisitos que se tienen en cada momento con los médicos. Por lo tanto, utilizando el proceso iterativo e incremental no es necesario tener una lista completa de todos los requisitos del proyecto antes de empezar el desarrollo.

- Permite que el Equipo de Navarrabiomed pueda ver resultados con funcionalidad desde la primera iteración.

- Permite gestionar los cambios en el proyecto: al final de cada iteración se recogían sus impresiones y sugerencias, a través de una demo del prototipo, en una entrevista. Estas

eran utilizadas para planificar los siguientes requisitos que se necesitaban completar en función del valor que le aportaban.

- Permite gestionar la complejidad del proyecto: en cada iteración sólo se trabajó con los requisitos que aportaban mayor valor al equipo de Navarrabiomed, dividiendo la complejidad en diferentes iteraciones.

- Minimiza el número de errores, aumentado la calidad del proyecto y mitigando los riesgos.

4.3 Desventajas

- Requiere un cliente involucrado durante todo el curso del proyecto ya que participa de manera continua: dada la carga de trabajo de los médicos, no siempre se pudo tener una reunión con ellos en los mismos plazos.

- Implica una estrecha colaboración entre el cliente y el equipo de desarrollo. A diferencia del desarrollo en cascada donde se trabaja de forma independiente. En nuestro caso, contamos con la ventaja de disponer de un cliente que ya había trabajado en proyectos informáticos previamente. Esto facilitó esta estrecha colaboración siempre que era posible.

- Requiere que cada iteración aporte valor al cliente y debe dar como resultado requisitos terminados.

- Requiere disponer de técnicas o herramientas que permitan hacer cambios fácilmente en el producto.

Estas desventajas se tuvieron en cuenta en la fase de planificación del proyecto, en el momento de elegir el tipo de desarrollo. Se concluyó que eran mayores las ventajas que presentaba que las desventajas.

4.4 Estructura del proyecto

En la etapa de planificación inicial del proyecto se realizó una primera reunión con los médicos al frente del proyecto. En esta se planteó su necesidad, el objetivo principal del proyecto, la forma en la que había surgido y unos primeros requisitos que debía cumplir.

A partir de esta reunión se puso en marcha todo el equipo informático necesario y comenzó una primera iteración. En esta se realizó un análisis de los primeros requisitos que se debían implementar teniendo en cuenta la importancia que se les atribuyó en la reunión. Así mismo, se anotaron cuáles eran las tareas que eran necesarias realizar para cumplirlos.

A partir de esto comenzó la etapa de implementación que terminó con un primer prototipo. Este fue presentado en una segunda reunión en la que se realizó una demo y se obtuvieron sus primeras impresiones, utilizadas para añadir nuevos requisitos al proyecto. Siempre se expresaron los requisitos que se tenían en cada momento, incluyendo los nuevos obtenidos, y se validaron por parte del cliente.

Se realizaron más iteraciones en las que se reorganizaban los requisitos a implementar después de obtener las impresiones del cliente. Según el valor que le aportase, se anotaban las tareas que harían cumplir los requisitos. A continuación, se pasaba a la fase de implementación y después se realizaba una demo del prototipo a través de una entrevista.

5. Especificación de requisitos

5.1 Introducción

Una de las fases más críticas en el ciclo de vida del desarrollo de software es la fase de especificación de requisitos. De esta depende el éxito en la implementación del sistema. La aplicación de buenas prácticas en el proceso de obtención de éstos ayudará a mantener siempre una idea centrada en lo que el proyecto necesita. De esta manera se minimizan posibles fallos en el entendimiento de la funcionalidad de la aplicación. Además de permitir la adaptación a nuevos requisitos o mejoras.

A pesar de que la especificación de requisitos software (ERS) según el estándar IEEE 830 [18] tiene como finalidad la creación de un documento de especificación de requisitos y en este trabajo la documentación no ha sido tan exhaustiva, nos basamos en aquellas fases previas a la creación de la documentación definidas en este estándar:

- Una primera etapa de análisis de requisitos.
- Una segunda etapa de clasificación. En este caso tratamos una clasificación en requisitos funcionales (RF) y requisitos no funcionales (RNF).
- Una fase de modelado de requisitos mediante casos de uso.
- Una última fase de negociación, expresando y validando los requisitos con el cliente.

Como ha sido mencionado con anterioridad, el proceso de desarrollo software elegido ha sido el iterativo e incremental, que se caracteriza por la participación de los propios usuarios finales. En este caso participaron los médicos al frente del proyecto informático, que además de ser el cliente, son también el usuario final de la aplicación. Ayudaron en la determinación de los requisitos, empleando la técnica de entrevistas.

En cada iteración de la fase de prototipado iterativo se realizaba una entrevista con el cliente. En cada entrevista se llevaban a cabo demos/pruebas del incremento del que se disponía en ese momento. El objetivo de realizar estas entrevistas fue el de recopilar requisitos que podrían no haberse planteado, refinar algunos de los ya existentes e incluso modificarlos, según las reacciones recibidas. Siempre expresando y validando los requisitos con el cliente.

5.2 Requisitos Funcionales

Los requisitos funcionales (RF) son aquellos que describen el sistema o las funciones de los elementos del sistema o las funciones que deben ser realizadas [7].

La lista de los requisitos funcionales (RF) que se capturaron a través de múltiples entrevistas con los médicos de Navarrabiomed se especifica en la Tabla 1, en el orden en el que aparecieron y se fueron implementando.

RF-01	Registro de pacientes.	El médico especialista desea poder dar de alta a un nuevo paciente a través de su número de la seguridad social para iniciar su seguimiento. Una vez se haya clicado en el botón de registrar paciente, aparecerá un formulario donde se rellenará, además del número de la seguridad social del paciente, sus características basales.
RF-02	Búsqueda de pacientes.	El médico especialista desea poder buscar los pacientes que hayan sido registrados en su hospital a través de su número de la seguridad social. En caso de que el paciente no esté registrado o no se pueda acceder a él debido a que éste ha sido dado de alta en otro hospital, debe aparecer un mensaje de error en la pantalla.
RF-03	Consultar, añadir, y modificar características basales de pacientes.	Una vez el médico especialista haya, o bien dado de alta a un paciente, o bien buscado a un paciente, se mostrará una vista de sus características basales guardadas. Pudiendo añadir o modificar la información guardada.
RF-04	Consultar, añadir y modificar información de los diferentes momentos correspondientes al seguimiento del paciente.	Una vez el médico especialista haya, o bien dado de alta a un paciente, o bien buscado a un paciente, en su vista de características basales guardadas debe aparecer botones que llevarán a los diferentes momentos del paciente. Cuando se clique por primera vez en uno de esos botones, se podrá añadir información. Las siguientes veces que se clique, se podrá consultar la información, pudiendo modificarla.

RF-05	Consultar, añadir y modificar imágenes endoscópicas de los pacientes en cada uno de los momentos.	Cada vez se acceda a un momento del paciente, se podrán consultar las imágenes adjuntadas a éste. Pudiendo eliminarlas y añadir nuevas.
RF-06	Permitir almacenar formularios incompletos.	Una vez se haya registrado un nuevo paciente, es obligatorio rellenar todas sus características basales. En caso de que haya algún campo sin rellenar, aparecerá un mensaje de error en rojo, indicando aquellos campos que faltan por rellenar. Sin embargo, para los momentos de seguimiento del paciente, esto es distinto. Ya que se podrán ir rellenando los formularios según su avance clínico. Es el usuario quien debe rellenar si un formulario de un momento está completo o no. En caso de que lo señale como completo, sólo deberá poder enviar el formulario si como mínimo se ha añadido una imagen y todos los campos están completos.
RF-07	Informar mediante texto en rojo los campos que queden sin completar.	Para aquellos formularios de los momentos que estén guardados como incompletos, se debe poder ver en texto rojo aquellos campos que queden por rellenar.
RF-08	Indicar a través de diferentes colores el estado de los formularios de cada uno de los momentos.	Los botones de acceso a los momentos deberán tener diferentes colores en función del estado de sus formularios. Rojo si todavía no se ha accedido, amarillo si se encuentra incompleto o verde si está completo.
RF-09	Entrada en el sistema.	El médico especialista desea tener un sistema de autenticación para acceder a la página web introduciendo su código de doctor y contraseña. El sistema sólo debe permitir el acceso a la aplicación a los usuarios registrados correctamente, en caso de fallo deberá aparecer un mensaje de error en rojo. Para su cumplimiento se ha utilizado el sistema de Login del que dispone Laravel.
RF-10	Salida del sistema.	El médico especialista desea poder cerrar su sesión en la página web.

Tabla 1. Requisitos funcionales

5.3 Requisitos No Funcionales

Los requisitos no funcionales (RNF) son aquellos que imponen restricciones sobre los que el sistema debe operar o existir, o propiedades del sistema. Definen como debe ser un sistema. Requisitos en la calidad y factores humanos son ejemplos de este tipo de requisitos [19].

La lista de los requisitos no funcionales (RNF) que se capturaron a través de múltiples entrevistas con los médicos de Navarrabiomed se especifica en la Tabla 2, en el orden en el que aparecieron y se fueron implementando.

RNF-01	Restricciones de acceso de pacientes.	Un médico sólo puede acceder a aquellos pacientes que hayan sido dados de alta en el hospital en el que trabaja. Esto se tuvo en cuenta en la búsqueda de pacientes.
RNF-02	Anonimidad de los datos.	Los datos de los pacientes deben ser anónimos, no pudiendo relacionar los datos guardados con el paciente al que corresponde. Para su cumplimiento, se ha empleado el sistema hash que proporciona Laravel sobre el número de la seguridad social del paciente.
RNF-03	Restricción de acceso a la herramienta web.	El acceso a los datos debe estar protegido frente a accesos no autorizados. Para su cumplimiento se ha utilizado un middleware del que dispone Laravel llamado “auth middleware”. Este impide que acceda a la aplicación toda persona que no haya podido autenticarse correctamente.
RNF-04	Validaciones de formularios.	En cada momento, si la información almacenada se selecciona como completa, no debe faltar ningún campo por completar y debe haber una imagen como mínimo asociada a ese momento. Esto se comprueba mediante las validaciones que dispone Laravel.

RNF-05	Uso intuitivo y sencillo de la aplicación.	Las interfaces de búsquedas y de formularios son de uso sencillo y guiadas mediante el uso de valores predeterminados.
RNF-06	Restricciones de registro de pacientes.	Un médico no puede registrar un paciente que ya haya sido dado de alta en otro centro médico.

Tabla 2. Requisitos no funcionales

5.4 Diagrama de casos de uso

En la fase de análisis de requisitos en el desarrollo software, UML [20] permite representar las definiciones de los requerimientos de software. En este caso utilizaremos un modelo de casos de uso. En la Figura 2 se representa como se llevará a cabo la interacción entre el sistema y el actor.

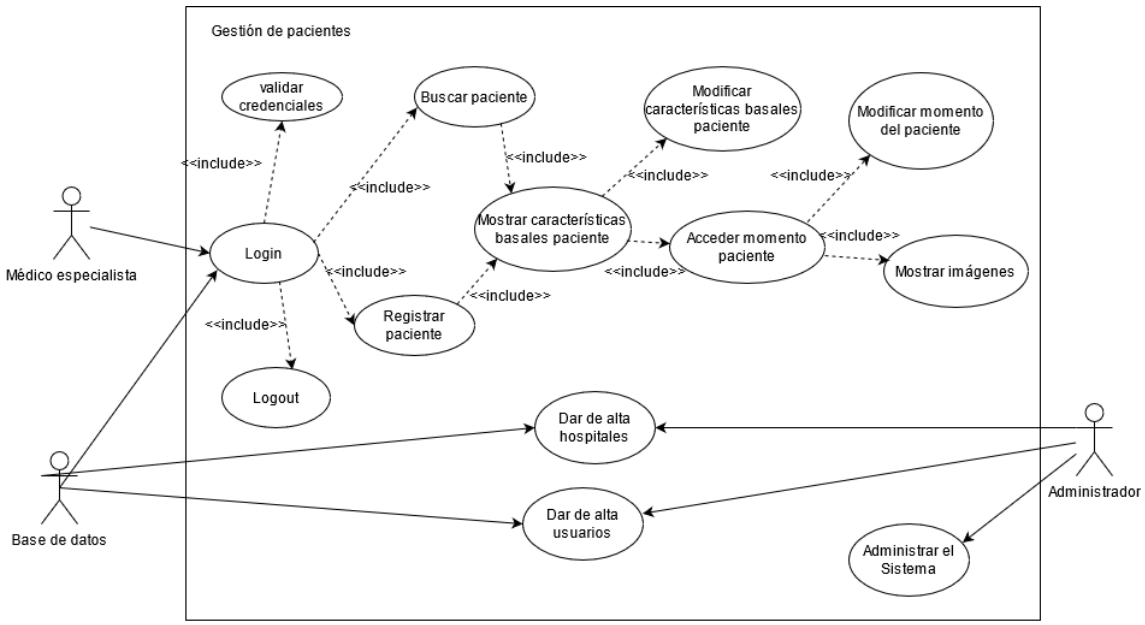


Fig. 2 Diagrama de casos de uso

5.5 Descripción de los casos de uso

En las siguientes Tablas [3,12] se muestra una explicación detallada de cada uno de los casos de uso representados en la Figura 2.

Caso de uso	**Login**

Descripción	Inicio de sesión para los usuarios.
Actores	Médico especialista y Base de datos.
Precondiciones	-
Flujo normal	1. El usuario introduce su código de doctor y su contraseña y pulsa el botón de acceder.
Postcondiciones	Usuario ha iniciado sesión.
Requisitos funcionales	RF-09
Requisitos no funcionales	-

Tabla 3. Caso de uso Login

Caso de uso	**Logout**
Descripción	El usuario finaliza su sesión.
Actores	Médico especialista y Base de datos.
Precondiciones	Haber iniciado sesión.
Flujo normal	1. El usuario pulsa el botón de cerrar sesión.
Postcondiciones	Usuario ha cerrado su sesión.
Requisitos funcionales	RF-10
Requisitos no funcionales	-

Tabla 4. Caso de uso Logout

Caso de uso	**Validar credenciales**
Descripción	Se comprueban las credenciales dadas con las almacenadas en la base de datos.
Actores	Médico especialista y Base de datos.
Precondiciones	El usuario ha realizado el proceso de iniciar sesión.
Flujo normal	1. El sistema comprueba en la base de datos que existe ese código de doctor con esa contraseña. 2. En caso de que haya error en las credenciales, aparece un mensaje de error en rojo.
Postcondiciones	Se han verificado las credenciales.
Requisitos funcionales	-
Requisitos no funcionales	RNF-01

Tabla 5. Caso de uso Validar credenciales

Caso de uso	**Registrar paciente**
Descripción	El usuario registra un nuevo paciente en la base de datos.

Actores	Médico especialista y Base de datos.
Precondiciones	Usuario ha iniciado sesión correctamente.
Flujo normal	1. El usuario pulsa el botón de registrar paciente, rellena su formulario de características basales y lo envía. 2. A continuación, se muestra una vista con la información guardada del paciente.
Postcondiciones	En caso de que el paciente ya exista, se informa de este hecho. En caso contrario, se almacena en la base de datos.
Requisitos funcionales	RF-01 y RF-03
Requisitos no funcionales	RNF-02, RNF-04, RNF-05, RNF-06

Tabla 6. Caso de uso Registrar paciente

Caso de uso	**Buscar paciente**
Descripción	El usuario busca un paciente a través de su número de seguridad social.
Actores	Médico especialista y Base de datos.
Precondiciones	Usuario ha iniciado sesión.
Flujo normal	1. El médico introduce en el buscador el número de la seguridad social y clica en buscar. 2. En caso de que el paciente no esté dado de alta o no se tenga acceso a éste, se indicará por pantalla.
Postcondiciones	Si el médico tiene acceso al paciente, se muestra su información. En caso contrario, se le indica que no tiene acceso.
Requisitos funcionales	RF-02
Requisitos no funcionales	RNF-01, RNF-05

Tabla 7. Caso de uso Buscar paciente

Caso de uso	**Mostrar características basales paciente**
Descripción	El usuario consulta las características basales del paciente almacenadas.
Actores	Médico especialista y Base de datos.
Precondiciones	El usuario ha iniciado sesión y, o bien ha registrado un nuevo paciente, o ha buscado un paciente.

Flujo normal	1. Se muestra al médico la información del paciente.
Postcondiciones	El usuario ve en pantalla las características basales del paciente.
Requisitos funcionales	RF-03
Requisitos no funcionales	RNF-02

Tabla 8. Caso de uso Mostrar características paciente

Caso de uso	**Modificar características basales paciente**
Descripción	El usuario modifica la información almacenada del paciente.
Actores	El médico especialista y Base de datos.
Precondiciones	El usuario ha iniciado sesión y, o bien, ha registrado un paciente o lo busca para poder modificarlo.
Flujo normal	1. El médico pulsa el botón de modificar las características basales del paciente. 2. Se muestra una vista modificable con los datos del paciente guardados hasta el momento. 3. El médico modifica aquellos datos que necesite y se pulsa el botón de enviar.
Postcondiciones	Se modifican los datos del paciente.
Requisitos funcionales	RF-03
Requisitos no funcionales	RNF-02

Tabla 9. Caso de uso Modificar características paciente

Caso de uso	**Acceder momento paciente**
Descripción	El usuario accede un momento del seguimiento del paciente.
Actores	El médico especialista y Base de datos.
Precondiciones	El médico se encuentra en la vista de las características basales del paciente.
Flujo normal	1. El médico pulsa el botón que acceder al momento. 2. Se muestra la información almacenada del momento, en caso de que falte información, aparecerá en rojo.

Postcondiciones	El médico accederá a la información del momento del paciente, incluyendo los campos que no se hayan completado.
Requisitos funcionales	RF-04, RF-07, RF-08
Requisitos no funcionales	-

Tabla 10. Caso de uso Acceder momento paciente

Caso de uso	**Modificar momento paciente**
Descripción	El usuario modifica la información almacenada del momento del paciente.
Actores	Médico especialista y Base de datos.
Precondiciones	El médico se encuentra en la vista que muestra los datos del momento del paciente.
Flujo normal	1. El médico pulsa el botón de modificar momento. 2. Se muestra una vista modificable con los datos almacenados en el momento. 3. El médico modifica aquellos datos que necesite, incluyendo las imágenes, y pulsa el botón de enviar.
Postcondiciones	Se guardan los datos modificados del paciente.
Requisitos funcionales	RF-04, RF-07
Requisitos no funcionales	RNF-04

Tabla 11. Caso de uso Modificar momento paciente

Caso de uso	**Mostrar imágenes**
Descripción	El usuario consulta las imágenes endoscópicas almacenadas en cada momento.
Actores	El médico especialista y Base de datos.
Precondiciones	El médico se encuentra en la vista que muestra la información almacenada del momento del paciente.
Flujo normal	1. El médico accede al momento. 2. Puede observar las imágenes al final del formulario, junto a su nombre, en caso de que no haya

	ninguna imagen adjunta, se mostrará un mensaje en rojo.
Postcondiciones	-
Requisitos funcionales	RF-05, RF-07
Requisitos no funcionales	-

Tabla 12. Caso de uso Mostrar imágenes

De esta manera quedan totalmente especificados todos los requisitos del usuario y del sistema.

6. Diseño

6.1 Introducción

En esta etapa del proceso de desarrollo de software se establecen las estructuras de datos, la arquitectura general del software y la representación de la interfaz y algoritmos. El diseño traduce requisitos en una representación de software. Además, un buen diseño implicará una calidad en el producto que se implemente.

En esta etapa se utilizará la información recolectada en la anterior etapa de Especificación de requisitos. El diseño en desarrollo software se divide en diferentes etapas, pero nos centraremos en dos: diseño de los datos y diseño arquitectónico.

6.2 Diseño y modelado de la Base de Datos

Empleamos el modelo relacional que aparece en la Figura 3 para representar la base de datos SQL que se creó para el diseño de los datos. Está formado por las siguientes tablas y relaciones entre ellas:

- **hospitals**: cuya clave primaria es el código del hospital “hosp_code”, posee además el nombre del hospital “hosp_name”.

- **users**: cuya clave primaria es el código del doctor “doc_code”. Posee además una contraseña con la que acceder a la página web, “password” y el código “hosp_code” del hospital al que pertenece.

  La relación entre la tabla **hospitals** y la tabla **users** es de 1 a M, por lo tanto, un doctor pertenece a un hospital y en un hospital pueden trabajar muchos médicos.

- **patient_baseline_characteristics**: cuya clave principal es un identificador de paciente “id_patient” de tipo incremental que se incrementa en uno cada vez que se registre un nuevo paciente. Posee además un campo único, “soc_sec” correspondiente con su número de la seguridad social que será utilizado para su búsqueda. Se almacena también la información basal del paciente.

- **registry**: en esta tabla se guardan los registros de los pacientes, dados de alta a través de un médico en un hospital. Contiene "id_patient" como clave primaria, siendo además una clave extranjera que hace referencia a la tabla **patient_baseline_characteristics**. Su campo "hosp_code" es clave extranjera en la tabla **hospitals** y su campo "doc_code" es clave extranjera en la tabla **users**.

  La tabla **registry** está relacionada con la tabla **hospitals** mediante una relación 1 a N. De manera que en un registro sólo puede haber un hospital asociado pero un hospital puede aparecer en más de un registro.

  De la misma manera está relacionada la tabla **registry** con la tabla **users**. La tabla **registry** posee una relación 1 a 1 con la tabla **patient_baseline_characteristics** ya que un paciente es dado de alta únicamente en un registro y en un registro sólo puede darse de alta un paciente.

- **images**: en esta tabla se guardan las imágenes colonoscópicas de los pacientes, asociadas a cada uno de sus momentos. Consta de un identificador de imagen, "id", como clave primaria y es de tipo incremental. También posee el identificador del paciente, "id_patient" al que está asociada. El momento "moment" al que pertenece la imagen. Este momento se guarda en tipo texto ("lesion", "histological", "procedure", "delayed"," 36MonthsFollowUp", "1518MonthsFollowUp"). Un nombre único, "image" formado por la intersección de la fecha de subida y el índice de la imagen de subida. Por último, un nombre público que utiliza el médico para subir la imagen "name_image" que aparece al lado de cada imagen.

  La relación entre la tabla **patient_baseline_characteristics** e **images** es de 1 a N. De manera que un paciente puede tener asociadas varias imágenes en diferentes momentos, pero una imagen pertenece a un único paciente.

- Para las siguientes tablas correspondientes a los momentos con los que se desea seguir el seguimiento del paciente: **lesion_characteristics**, **procedure**, **histological_assessment**, **delayed_complication**, **3-6months_follow_up**, **15-18months_follow_up**, se ha seguido la siguiente estructura:

  - Siempre poseen el identificador del paciente al que pertenecen, "id_patient". Así como un atributo que aparece en cada uno de los momentos llamado "State_form". A través de este atributo el médico guarda de manera manual el estado del formulario, que puede ser completo o incompleto.

  - Para el almacenamiento de las variables de cada tabla se ha seguido la siguiente estructura:
    - Para las variables multivaluadas se ha creado una columna en la tabla para cada posible valor.

- Para el resto de las variables, se ha creado una única columna.

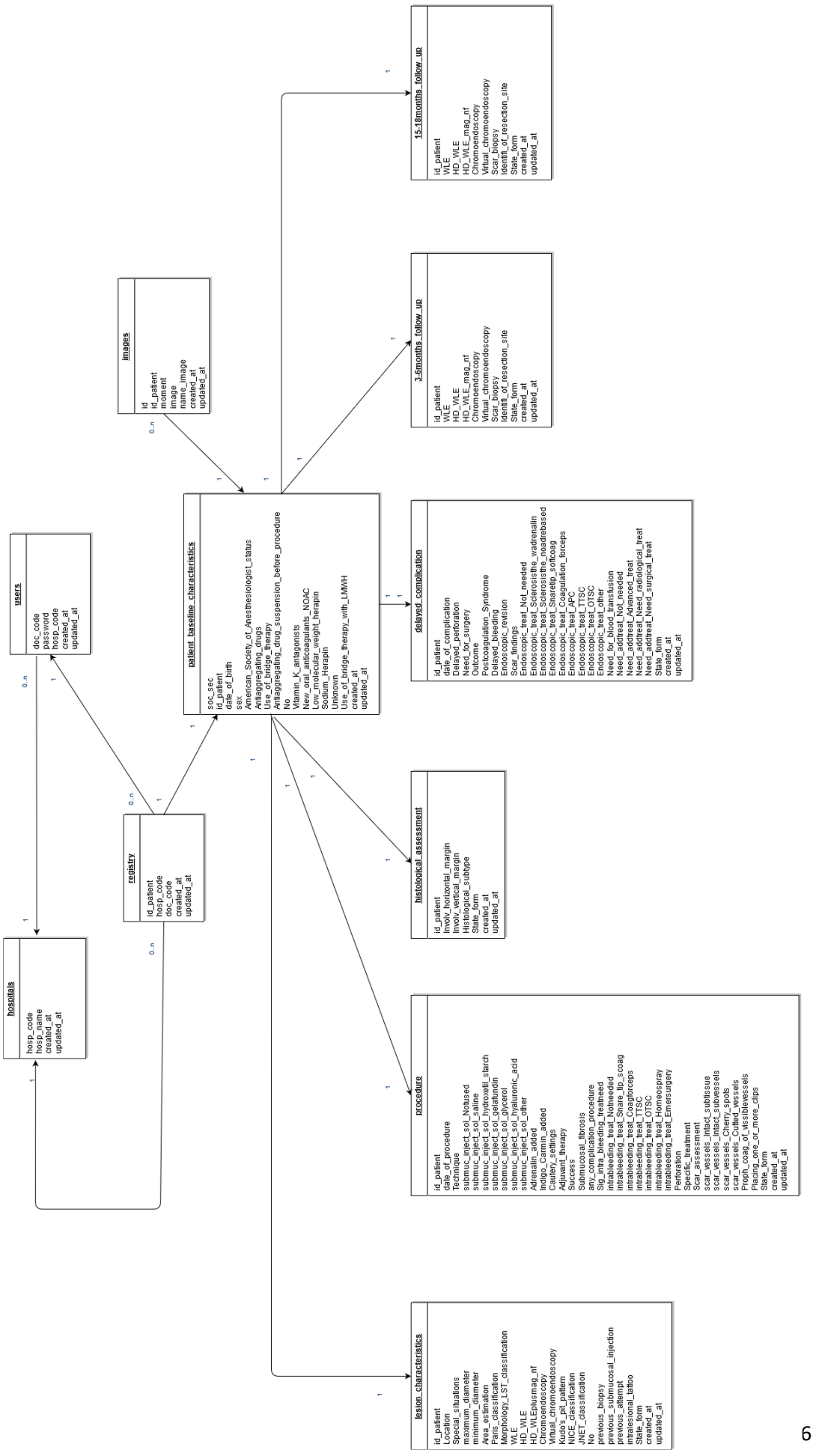

users
doc_code
password
hosp_code
created_at
updated_at
hospitals
hosp_code
hosp_name
created_at
updated_at
registry
id_patient
hosp_code
doc_code
created_at
updated_at
images
id
id_patient
moment
image
name_image
created_at
updated_at
patient_baseline_characteristics
soc_sec
id_patient
date_of_birth
sex
American_Society_of_Anesthesiologist_status
Antiaggregating_drugs
Use_of_bridge_therapy
Antiaggregating_drug_suspension_before_procedure
No
Vitamin_K_antagonists
New_oral_anticoagulants_NOAC
Low_molecular_weigth_Heparin
Sodium_Heparin
Unknown
Use_of_bridge_therapy_with_LMWH
created_at
updated_at
delayed_complication
id_patient
date_of_complication
Delayed_perforation
Need_for_surgery
Outcome
Postcoagulation_Syndrome
Delayed_bleeding
Endoscopic_prevision
Scar_findings
Endoscopic_treat_Not_needed
Endoscopic_treat_Sclerosistha_ivadrenalin
Endoscopic_treat_Sclerosistha_noadrebased
Endoscopic_treat_Snaretip_softcoag
Endoscopic_treat_Coagulation_forceps
Endoscopic_treat_APC
Endoscopic_treat_TTSC
Endoscopic_treat_OTSC
Endoscopic_treat_other
Need_for_blood_transfusion
Need_addtreat_Not_needed
Need_addtreat_Advanced_treat
Need_addtreat_Need_radiological_treat
Need_addtreat_Need_surgical_treat
State_form
created_at
updated_at
histological_assessment
id_patient
Involv_horizontal_margin
Involv_vertical_margin
Histological_subtype
State_form
created_at
updated_at
procedure
id_patient
date_of_procedure
Technique
submuc_inject_sol_NoUsed
submuc_inject_sol_saline
submuc_inject_sol_hydroxell_starch
submuc_inject_sol_glycerin
submuc_inject_sol_dextrandm
submuc_inject_sol_hyaluronic_acid
submuc_inject_sol_other
Adrenalin_added
Indigo_Carmin_added
Cautery_settings
Adjuvant_therapy
Success
Submucosal_fibrosis
any_complication_procedure
Sig_Intra_bleeding_treatneed
intrableeding_treat_Notneeded
intrableeding_treat_Sclerosis_scoag
intrableeding_treat_Coagforceps
intrableeding_treat_TTSC
intrableeding_treat_OTSC
intrableeding_treat_Homeospray
intrableeding_treat_Emersurgery
Perforation
Specific_treatment
Scar_assessment
scar_vessels_Intact_subtissue
scar_vessels_Intact_subvessels
scar_vessels_Cherry_spots
scar_vessels_Other_vessels
Prophylaxis_coag_of_vesiblevessels
Placing_one_or_more_clips
State_form
created_at
updated_at
lesion_characteristics
id_patient
Location
Special_situations
maximum_diameter
minimum_diameter
Area_estimation
Paris_classification
Morphology_LST_classification
WLE
HD_WLE
HD_WLEplusmag_nf
Chromoendoscopy
Virtual_chromoendoscopy
Kudo's_pit_pattern
NICE_classification
JNET_classification
No
prewous_biopsy
prewous_submucosal_injection
previous_attempt
Intralesional_tatoo
State_form
created_at
updated_at
15-18months_follow_up
id_patient
WLE
HD_WLE
HD_WLE_mag_nf
Chromoendoscopy
Virtual_chromoendoscopy
Scar_biopsy
Ident_of_resection_site
State_form
created_at
updated_at
3-6months_follow_up
id_patient
WLE
HD_WLE
HD_WLE_mag_nf
Chromoendoscopy
Virtual_chromoendoscopy
Scar_biopsy
Ident_of_resection_site
State_form
created_at
updated_at

Fig. 3 Modelo relacional de la base de datos

6.3 Arquitectura Software

La arquitectura software abarca diferentes patrones que ayudan a la construcción de un programa o aplicación, sirviendo de guía para lograr cumplir aquellos requerimientos de la aplicación.

Dentro de los patrones de diseño, encontramos el patrón de arquitectura MVC (Modelo, Vista, Controlador). Este es uno de los más empleados para poder crear proyectos modulares y escalables. Laravel está diseñado para desarrollar bajo este patrón: la aplicación se separa en tres componentes lógicos, donde cada uno realiza una función específica. Las relaciones entre los componentes permiten que las aplicaciones puedan ser entendidas, reutilizadas y mantenidas de una forma más sencilla.

6.3.1 Descripción del patrón Arquitectónico

A continuación, se describen los componentes del patrón MVC. La forma en la que se relacionan queda reflejada en la Figura 4.

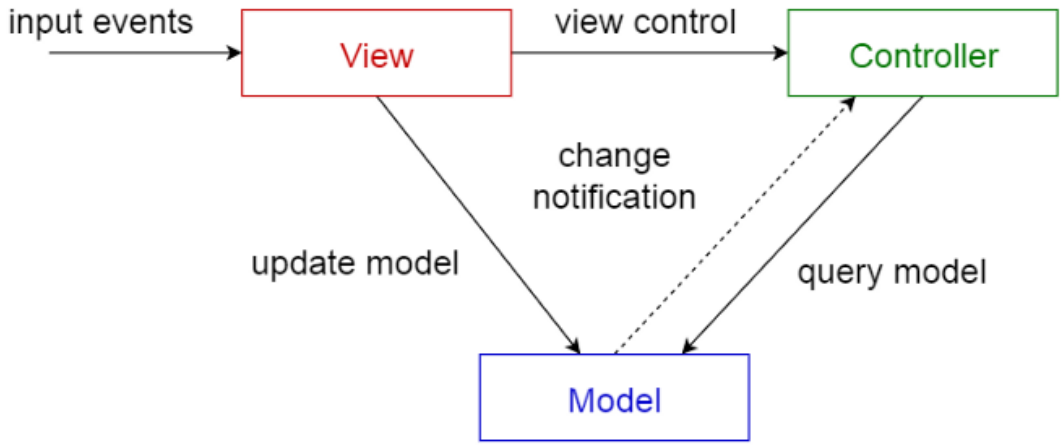


Fig. 4 Modelo, Vista, Controlador

Modelo: es la representación de la información con la cual la aplicación va a operar. Gestiona el acceso a dicha información: consultas, actualizaciones, creación de información y eliminación. Estas peticiones le llegan a través del controlador. Se conoce como la Lógica del negocio.

Vista: es la parte de la aplicación encargada de la interfaz gráfica, cada elemento de esta interfaz interactúa con el usuario y permite realizar peticiones que se enviarán al controlador. Utiliza el modelo para representar la información.

Controlador: responde a eventos o acciones que realice el usuario a través de la vista para poder solicitar una operación de información. Además, es responsable de elegir qué vista es la que debe mostrar al usuario de acuerdo con la solicitud recibida. Por lo cual podemos decir que actúa como intermediario entre el modelo y la vista.

En este patrón cada componente actúa y está aislado con actividades definidas. El poder encapsularlas permite la reutilización de componentes, un mejor mantenimiento y poder escalar el tamaño de una aplicación sin afectar a los componentes ya creados.

En el proyecto se ha desarrollado bajo este patrón arquitectónico de la siguiente manera:

- El Modelo es el encargado de recuperar, añadir, borrar y actualizar cualquier dato solicitado por el Controlador en la base de datos. En el subapartado 7.3.2 se muestra una explicación más detallada de cómo ha sido utilizado.

- La Vista es la encargada de mostrar los formularios o información solicitadas por el Controlador en función de las acciones del usuario. En el subapartado 7.3.3 se muestran ejemplos de vistas creadas.

- El Controlador decide qué vistas mostrar dependiendo de la acción del usuario y se encarga de solicitar al Modelo la información necesaria. En el subapartado 7.3.1 se muestra una explicación más detallada de los métodos implementados en los controladores.

7. Implementación

7.1 Introducción

El uso del framework PHP Laravel posee múltiples ventajas que han sido utilizadas para desarrollar la aplicación de una manera más sencilla y eficiente:

- Integra un sistema ORM de mapeado de datos relacional llamado Eloquent, aunque también permite la construcción de consultas directas a la base de datos mediante su Query Builder.

  En el desarrollo de esta aplicación hemos utilizado este sistema ORM definiendo por cada tabla de la base de datos su correspondiente modelo. Una vez definido un modelo, hemos podido interactuar desde código con cada tabla. También hemos empleado su Query Builder para realizar consultas. En el subapartado 7.3.2 se explica de una manera

más extensa la forma en la que se han creado los modelos y cómo han sido utilizados, así como la utilización de Query Builder.

- Permite gestionar la base de datos y manipular sus tablas desde código manteniendo un sistema de control de versiones de estas desde su sistema de migraciones. Se explicará más detalladamente en el subapartado 7.2.2.

- Utiliza un sistema de plantillas para vistas llamado Blade, el cual facilita la creación de vistas a mediante el uso de layouts y secciones. Se explicará más detalladamente en el subapartado 7.2.4.

- Incorpora un intérprete de línea de comandos llamado Artisan que ayuda con muchas tareas rutinarias. Entre ellas está la creación de diferentes componentes de código, siendo utilizado para la creación de los modelos, vistas y controladores.

7.2 Estructura del proyecto Laravel

A continuación, se presenta la estructura de directorios que compone nuestro proyecto de Laravel, en la que destacamos los directorios app, database, public, resources y routes. Esta estructura queda reflejada en la Figura 5.

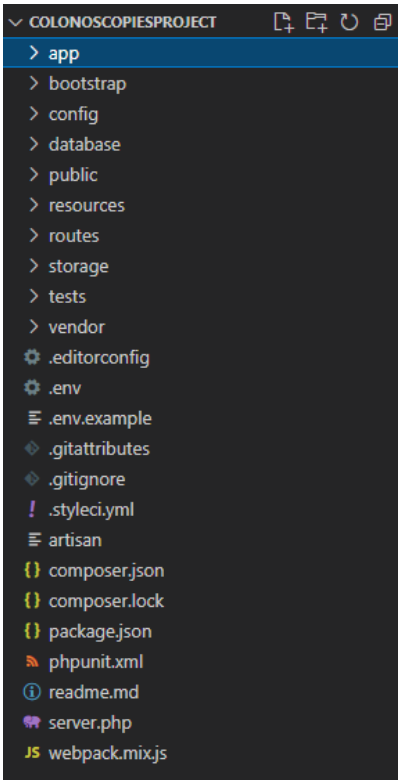


Fig. 5 Estructura del proyecto Laravel

7.2.1 Directorio App

Dentro de este directorio encontramos varios directorios importantes a los que se han accedido en el desarrollo de la aplicación:

1. Directorio App\Http\Controllers: en este directorio se encuentran los diferentes controladores que se han creado. Como se puede ver en la Figura 6 se creó un controlador para el paciente, y un controlador para cada momento del seguimiento. En el subapartado 7.3.1 se explica más detalladamente la forma en la que se crearon estos controladores, cuáles son sus principales funciones y algunas capturas de éstos.

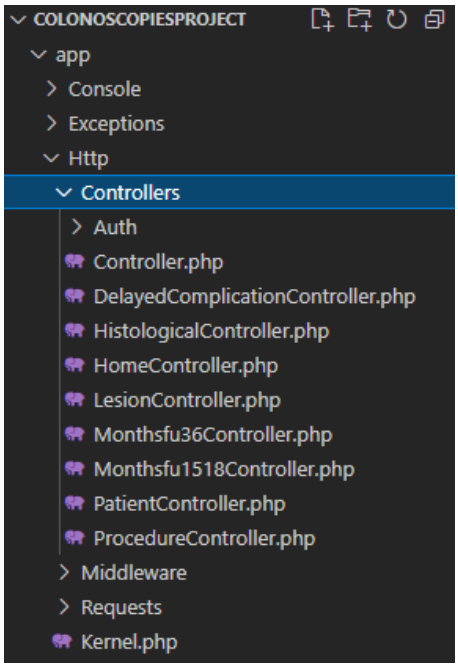


Fig. 6 Contenido directorio Controllers

1.1 Directorio App\Http\Controllers\Auth: en este se encuentra el controlador del Login, como se puede observar en la Figura 7. Este ha sido muy importante en este proyecto para el control de la autenticación de los médicos que acceden a la aplicación.

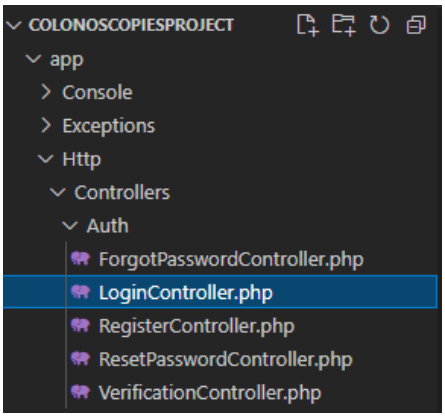
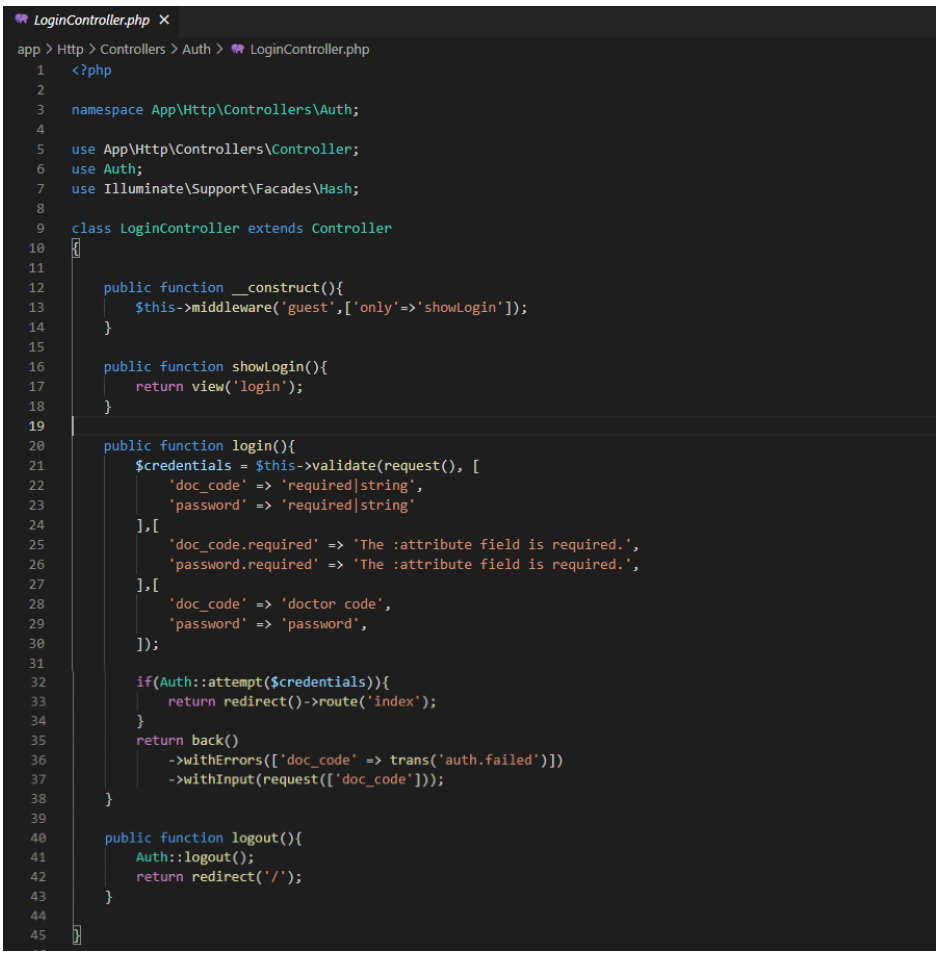


Fig. 7 Directorio Auth

Su contenido queda reflejado en la Figura 8.



LoginController.php

app > Http > Controllers > Auth > LoginController.php

```php
<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;
use Auth;
use Illuminate\Support\Facades\Hash;

class LoginController extends Controller
{

    public function __construct(){
        $this->middleware('guest',['only'=>'showLogin']);
    }

    public function showLogin(){
        return view('login');
    }

    public function login(){
        $credentials = $this->validate(request(), [
            'doc_code' => 'required|string',
            'password' => 'required|string'
        ],[
            'doc_code.required' => 'The :attribute field is required.',
            'password.required' => 'The :attribute field is required.',
        ],[
            'doc_code' => 'doctor code',
            'password' => 'password',
        ]);

        if(Auth::attempt($credentials)){
            return redirect()->route('index');
        }
        return back()
            ->withErrors(['doc_code' => trans('auth.failed')])
            ->withInput(request(['doc_code']));
    }

    public function logout(){
        Auth::logout();
        return redirect('/');
    }

}
```

Fig. 8 Contenido LoginController

2. Directorio App\Models: en este directorio se encuentran todos los modelos creados, se pueden ver en la Figura 9. Cada modelo queda asociado a una tabla de la base de datos. En el subapartado 7.3.2 se

explica cómo se han creado los modelos, cómo se utilizan y dónde, junto a alguna captura de estos.

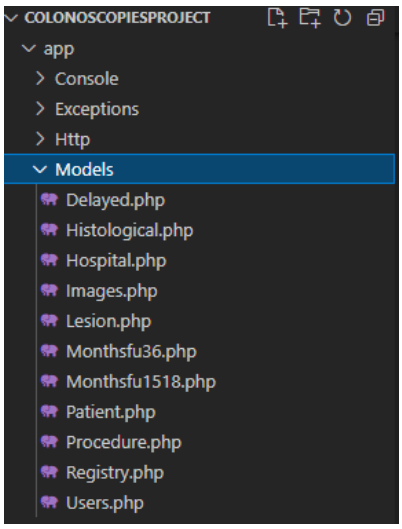


Fig. 9 Contenido directorio Models

7.2.2 Directorio Database

En este directorio se encuentran todos los archivos de migraciones. Las migraciones son un tipo de control de versiones de la base de datos. Estas permiten que un equipo modifique el esquema de la base de datos y se encuentre al día del estado actual del esquema. Las migraciones creadas para este proyecto se encuentran en la Figura 10.

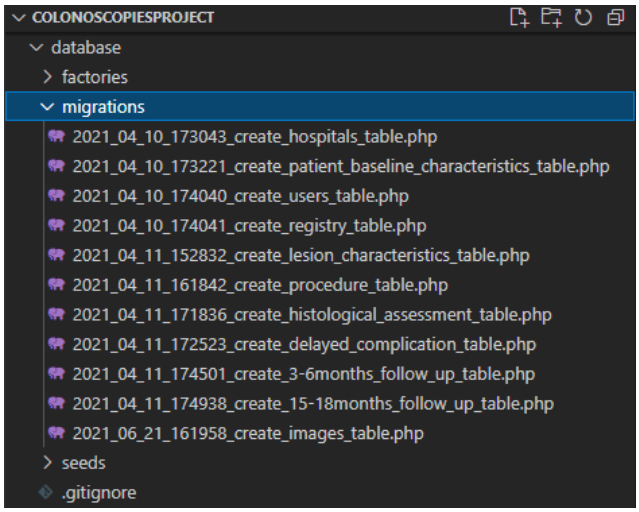


Fig. 10 Contenido directorio Database

Para la creación de una nueva migración se hizo uso del comando “make migration” de Artisan presente en la Figura 11.

```
PS C:\xampp2\htdocs\ColonoscopiesProject> php artisan make:migration create_patient_baseline_characteristics_table
--create=patient_baseline_characteristics
```

Fig. 11 Comando para la creación de migración

Los ficheros que se crean después de aplicar este comando están vacíos y es necesario definir los campos de la tabla que se desea crear de la forma en la que aparece en la Figura 12.

database > migrations > 2021_04_10_173221_create_patient_baseline_characteristics_table.php

```php
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreatePatientBaselineCharacteristicsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('patient_baseline_characteristics', function (Blueprint $table) {
            $table->string('soc_sec')->unique();
            $table->bigIncrements('id_patient');

            $table->date('date_of_birth');
            $table->integer('sex');
            $table->integer('American_Society_of_Anesthesiologist_status');
            $table->integer('Antiaggregating_drugs');
            $table->integer('Use_of_bridge_therapy')->nullable();
            $table->integer('Antiaggregating_drug_suspension_before_procedure')->nullable();

            /* multiple answers */
            $table->boolean('No');
            $table->boolean('Vitamin_K_antagonists');
            $table->boolean('New_oral_anticoagulants_NOAC');
            $table->boolean('Low_molecular_weight_heparin');
            $table->boolean('Sodium_Heparin');
            $table->boolean('Unknow');

            $table->integer('Use_of_bridge_therapy_with_LMWH')->nullable();


            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('patient_baseline_characteristics');
    }
}
```

Fig. 12 Contenido fichero de migración

Para ejecutar las migraciones se empleó el comando de la Figura 13.

```
PS C:\xampp2\htdocs\ColonoscopiesProject> php artisan migrate
```

Fig. 13 Comando para ejecutar migraciones

Cuando había más de una migración pendiente se utilizaba el anterior comando. De manera que se ejecutaban las migraciones en el orden en el que se habían creado a través de su método up(). La función down() de una migración define la operación

inversa. En este caso se utilizó para eliminar la tabla de la base de datos en algunos momentos.

Estos son dos comandos de migraciones que han sido muy utilizados:

- Cuando se deseaba deshacer los cambios realizados y volver a aplicar las migraciones se aplicaba el comando de la Figura 14.

```
PS C:\xampp2\htdocs\ColonoscopiesProject> php artisan migrate:refresh
```

Fig. 14 Comando deshacer cambios y aplicar migraciones

- Cuando se deseaba deshacer todas las migraciones se aplicaba el comando de la Figura 15.

```
PS C:\xampp2\htdocs\ColonoscopiesProject> php artisan migrate:reset
```

Fig. 15 Comando deshacer todas las migraciones

De esta manera se han ido creando las diferentes tablas que componen la base de datos de la aplicación.

7.2.3 Directorio Public

Esta es la única carpeta pública del proyecto, es visible en nuestro servidor web. En esta carpeta se alojan los archivos JavaScript y las imágenes que se suben a través de la página web. Su contenido queda reflejado en la Figura 16.

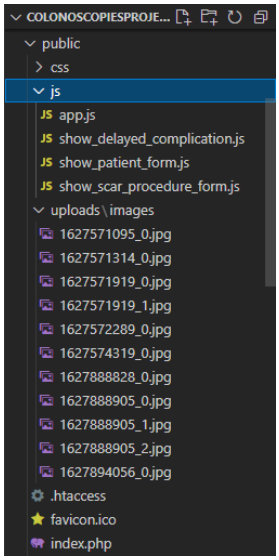


Fig. 16 Contenido archivo Public

Se crearon tres ficheros JavaScript. Estos ficheros JavaScript permiten que ciertos campos dentro de un formulario aparezcan o desaparezcan dependiendo de ciertas condiciones sobre los campos seleccionados.

La Figura 17 muestra el fichero JavaScript creado para la aparición y desaparición de campos dentro del formulario de las características basales del paciente.

```
$(document).ready(function(){
    var entero = document.getElementById("antiag_drugs_select").selectedIndex;
    if (entero === 2){
      $("#bridge_therapy").show();
      mostrarBridge = true;
    }else if(mostrarBridge){
      $("#bridge_therapy").hide();
      mostrarBridge = false;
    }
    if (entero !== 0 && entero !== 4){
      $("#antiag_drugs_susp").show();
      mostrarSusp = true;
    }else if(mostrarSusp){
      $("#antiag_drugs_susp").hide();
      mostrarSusp = false;
    }
});

$(document).ready(function(){
  $("#antiag_drugs_select").on('change',function(e){
    var entero = parseInt(e.target.value);
    if (entero === 2){
      $("#bridge_therapy").show();
      mostrarBridge = true;
    }else if(mostrarBridge){
      $("#bridge_therapy").hide();
      mostrarBridge = false;
    }
    if (entero !== 0 && entero !== 4){
      $("#antiag_drugs_susp").show();
      mostrarSusp = true;
    }else if(mostrarSusp){
      $("#antiag_drugs_susp").hide();
      mostrarSusp = false;
    }
  })
});
```

Fig. 17 Ejemplo fichero JavaScript

7.2.4 Directorio Resources

1. Directorio resources\views: en este directorio se encuentran las diferentes vistas que se han creado. Estas se pueden observar en la Figura 18. En general se han creado tres vistas para el paciente y otras tres para cada momento. Estas tres vistas corresponden con la vista del formulario que se rellena por primera vez, la vista que muestra los datos almacenados y la vista que permite modificar los datos almacenados.

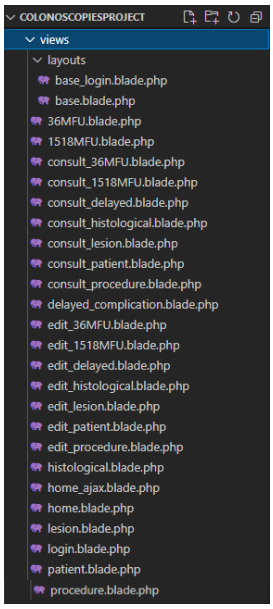


Fig. 18 Contenido archivo Views

a. Directorio resources\views\layouts: en este directorio se encuentran las vistas base que serán utilizadas por el resto de las vistas para evitar el uso repetitivo de código.

En el subapartado 7.3.3 se explica más detalladamente la forma en la que se han creado, en qué se diferencian y algunas capturas de estas.

7.2.5 Directorio Routes

Dentro de este directorio, el documento más importante es web.php, que define todas las rutas de la aplicación web. Enlaza una URL del navegador con un método del controlador. Se pueden ver todas las rutas de la aplicación a través de la siguiente figura 19:

web.php

routes > web.php

```php
Route::any('/','Auth\LoginController@showLogin')->name('/');
Route::any('/receivelogin','Auth\LoginController@login')->name('login');
Route::any('/logout','Auth\LoginController@logout')->name('logout');

Route::any('/index', 'HomeController@anyIndex')->name('index');


/* 3-6 months follow up assessment */
Route::any('/3_6_mfollow','Monthsfu36Controller@any36MFU')->name('3_6_mfollow');
Route::any('/send36MFUInfo', 'Monthsfu36Controller@Receive36MFUForm')->name('send36MFUInfo');
Route::any('/edit36MFUInfo','Monthsfu36Controller@edit36MFU')->name('edit36MFUInfo');
Route::any('/save36MFUInfo','Monthsfu36Controller@Modify36MFU')->name('save36MFUInfo');

/* 15-18 months follow up assessment */
Route::any('/15_18_mfollow','Monthsfu1518Controller@any1518MFU')->name('15_18_mfollow');
Route::any('/send1518MFUInfo', 'Monthsfu1518Controller@Receive1518MFUForm')->name('send1518MFUInfo');
Route::any('/edit1518MFUInfo','Monthsfu1518Controller@edit1518MFU')->name('edit1518MFUInfo');
Route::any('/save1518MFUInfo','Monthsfu1518Controller@Modify1518MFU')->name('save1518MFUInfo');

/* lesion assessment */
Route::any('/lesion','LesionController@anyLesion')->name('lesion');
Route::any('/sendLesionInfo', 'LesionController@ReceiveLesionForm')->name('sendLesionInfo');
Route::any('/editLesionInfo','LesionController@editLesion')->name('editLesionInfo');
Route::any('/saveLesionInfo', 'LesionController@ModifyLesion')->name('saveLesionInfo');

/* scar assessment : histological */
Route::any('/scar_assessment_histological', 'HistologicalController@anyHistological')->name('scar_assessment_histologica
Route::any('/sendScarHistologicalInfo', 'HistologicalController@ReceiveScarHistologicalForm')->name('sendScarHistologica
Route::any('/editHistologicalInfo','HistologicalController@editHistological')->name('editHistologicalInfo');
Route::any('/saveHistologicalInfo', 'HistologicalController@ModifyHistological')->name('saveHistologicalInfo');


Route::any('/searchPatient','PatientController@searchPatient')->name('searchPatient');
Route::any('/registerPatient','PatientController@registerPatient')->name('registerPatient');
Route::any('/sendPatientInfo','PatientController@ReceivePatientForm')->name('sendPatientInfo');
Route::any('/editPatientInfo','PatientController@editPatient')->name('editPatientInfo');
Route::any('/savePatientInfo','PatientController@ModifyPatientForm')->name('savePatientInfo');

/* scar assessment : procedure */
Route::any('/scar_assessment_procedure','ProcedureController@anyProcedure')->name('scar_assessment_procedure');
Route::any('/sendScarProcedureInfo', 'ProcedureController@ReceiveScarProcedureForm')->name('sendScarProcedureInfo');
Route::any('/editProcedureInfo','ProcedureController@editProcedure')->name('editProcedureInfo');
Route::any('/saveProcedureInfo','ProcedureController@ModifyProcedure')->name('saveProcedureInfo');


Route::any('/delayed_complication','DelayedComplicationController@anyDelayedComplication')->name('delayed_complication')
Route::any('/sendDelayedInfo', 'DelayedComplicationController@ReceiveDelayedComplicationForm')->name('sendDelayedInfo');
Route::any('/editDelayedInfo','DelayedComplicationController@editDelayed')->name('editDelayedInfo');
Route::any('/saveDelayedInfo','DelayedComplicationController@ModifyDelayed')->name('saveDelayedInfo');
```

Fig. 19 Contenido archivo web.php

7.3 Modelo, Vista, Controlador

7.3.1 Controladores

Para la creación de un nuevo controlador se hizo uso del comando “make controller” de Artisan, presentado en la Figura 20.

```
PS C:\xampp2\htdocs\ColonoscopiesProject> php artisan make:controller PatientController
```

Fig. 20 Comando para crear un controlador

Los controladores, tanto del paciente como de cualquier momento asociado, tienen algunos métodos comunes: un método que controla que sólo tenga acceso a la información las personas que hayan sido autenticadas correctamente, y otro método empleado para cambiar los colores de los botones de los momentos según su estado.

En el controlador del paciente, se han implementado métodos para: registrar un nuevo paciente, buscar un paciente según su número de la seguridad social y modificar la información almacenada y guardarla. Se pueden observar la Figura 21.

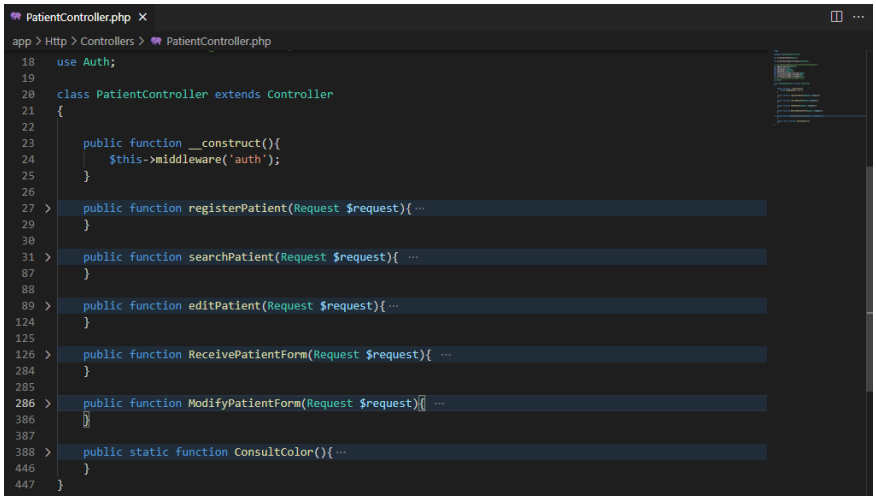


Fig. 21 Contenido archivo PatientController

En los controladores de los momentos se han implementado métodos para mostrar la información almacenada en ese momento y modificarla. En caso de que no tuviese información almacenada, mostrar el formulario correspondiente. Se pueden ver en la siguiente Figura 22.

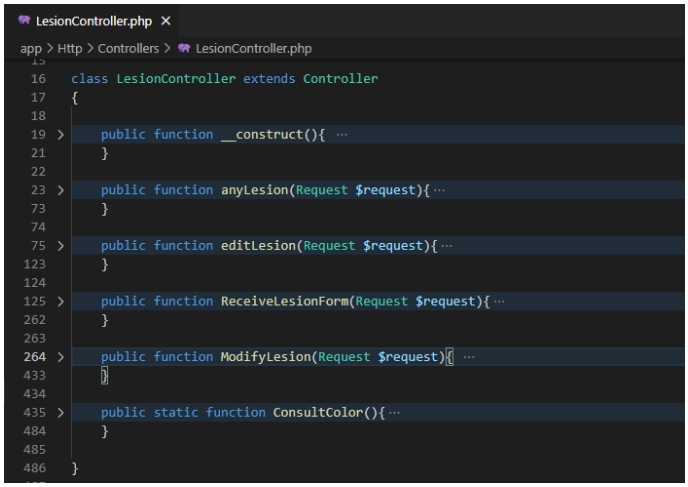


Fig. 22 Contenido archivo LesionController

7.3.2 Modelos

Para la creación de un nuevo modelo se hizo uso del comando “make model” de Artisan, presentado en la Figura 23.

```
PS C:\xampp2\htdocs\ColonoscopiesProject> php artisan make:model Models\Lesion
```

Fig. 23 Comando para crear un modelo

Dentro de cada modelo, definimos la clave primaria de la tabla asociada. Por ejemplo, el modelo “Lesion” creado se presenta en la Figura 24.

Lesion.php

app > Models > Lesion.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Lesion extends Model
{
    protected $table = 'lesion_characteristics';
    protected $primaryKey = 'id_patient';
}

```

Fig. 24 Contenido modelo Lesion.php

De esta manera se han ido creando los diferentes modelos que usa Eloquent. Estos han sido utilizados para recuperar datos de la base de datos o para insertar o actualizarlos.

Como el sitio correcto para realizar estas acciones es en el Controlador, incluimos el modelo en el espacio de nombres del controlador que utilizará. En las siguientes Figuras [25,26] se puede observar el ejemplo de utilización del modelo “Lesion” para insertar nuevos datos dentro de “LesionController”.

LesionController.php

app > Http > Controllers > LesionController.php

```php
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Lesion;
use App\Models\Images;

```

Fig. 25 Incluir el modelo Lesion

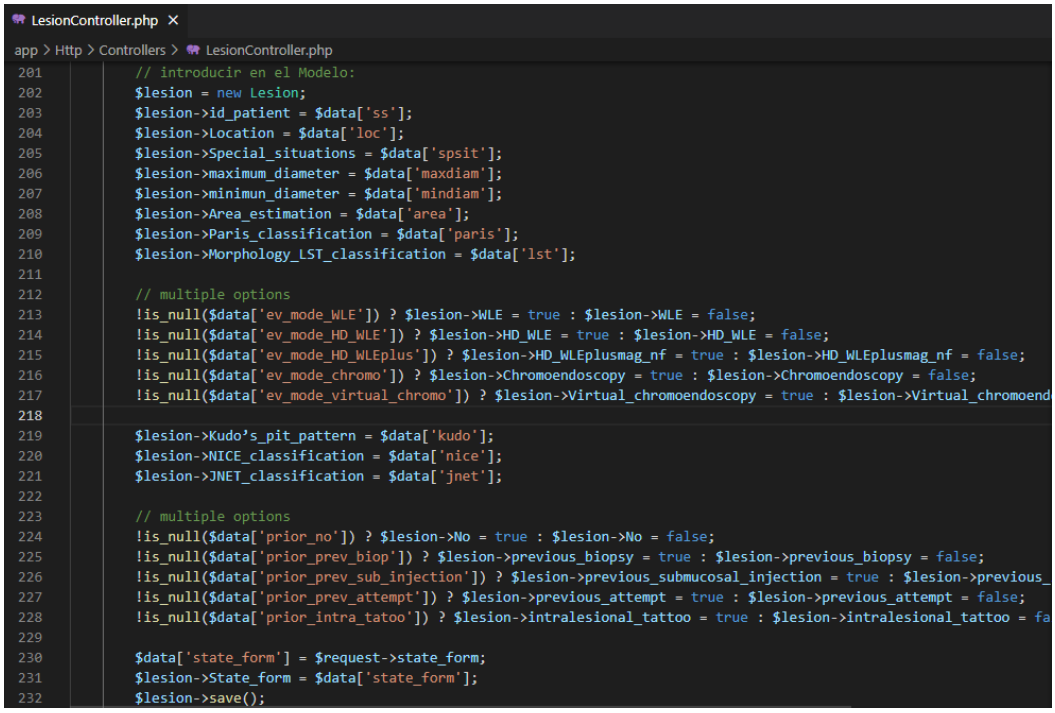


Fig. 26 Utilización del modelo Lesion

Hemos empleado el Query builder que ofrece Laravel para facilitar la construcción de consultas, creando una notación más legible y que previene de ataques por inyección de código SQL.

Un ejemplo de su utilización queda reflejado en la Figura 27. Se realiza la búsqueda de un paciente a través de su número de la seguridad social empleando Query builder.

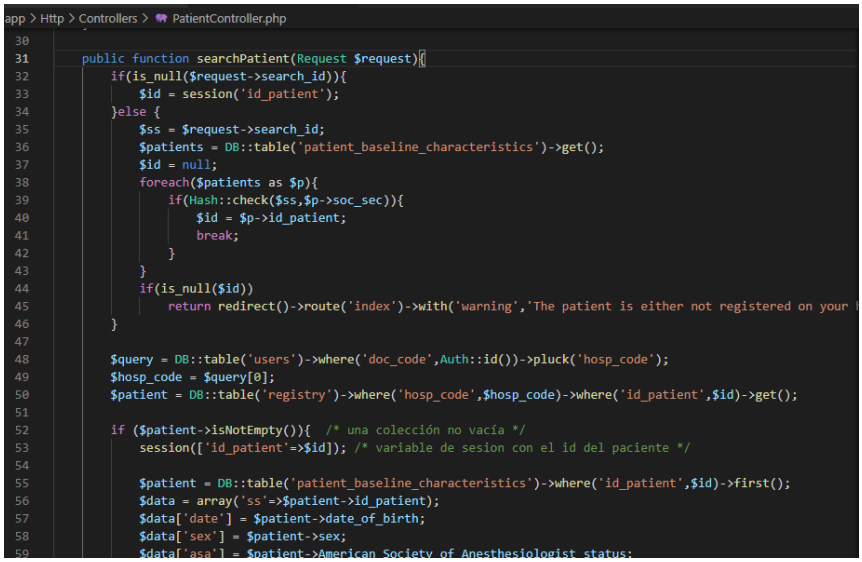


Fig. 27 Ejemplo de utilización de Query Builder

7.3.3 Vistas

La creación de las diferentes vistas se hizo de forma manual en VisualStudio, utilizando: Blade, HTML y JavaScript.

La siguiente Figura 28 corresponde al formulario del paciente que se rellena cuando se registra un nuevo paciente:

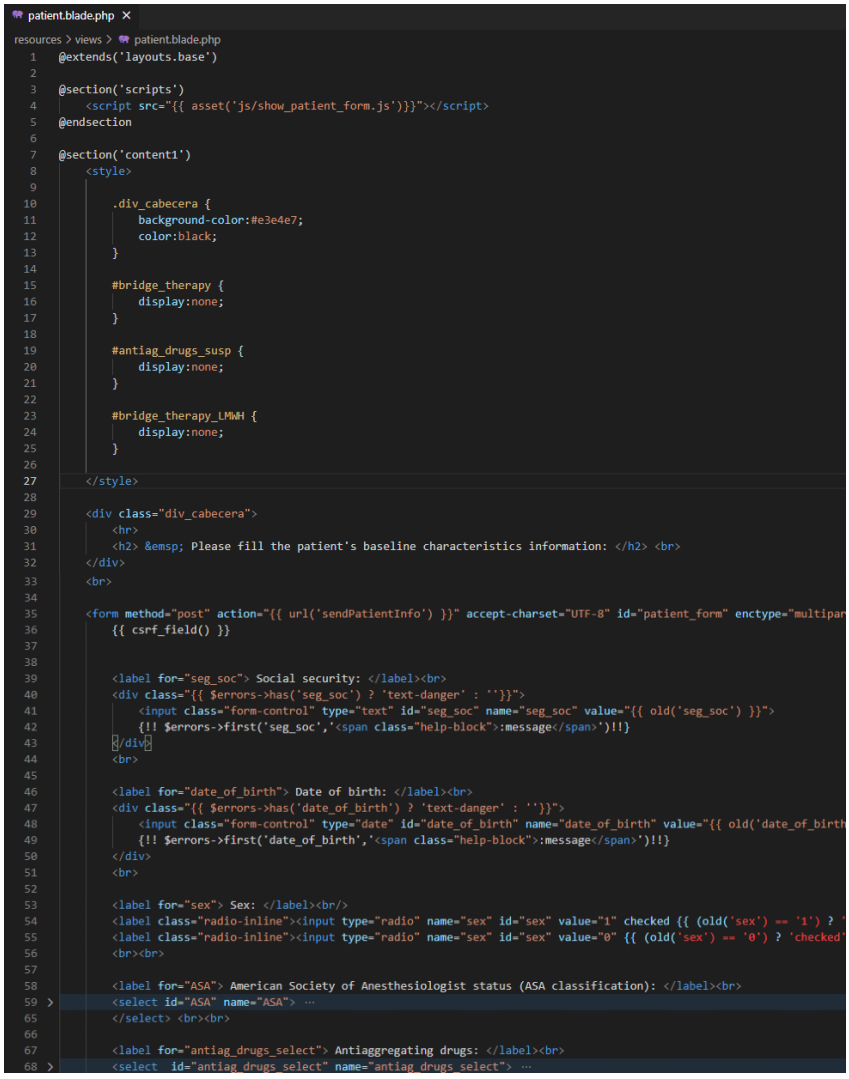


Fig. 28 Vista formulario características basales de un paciente

La Figura 29 corresponde a la vista que muestra la información almacenada del paciente en la base de datos, sin permitir su modificación.

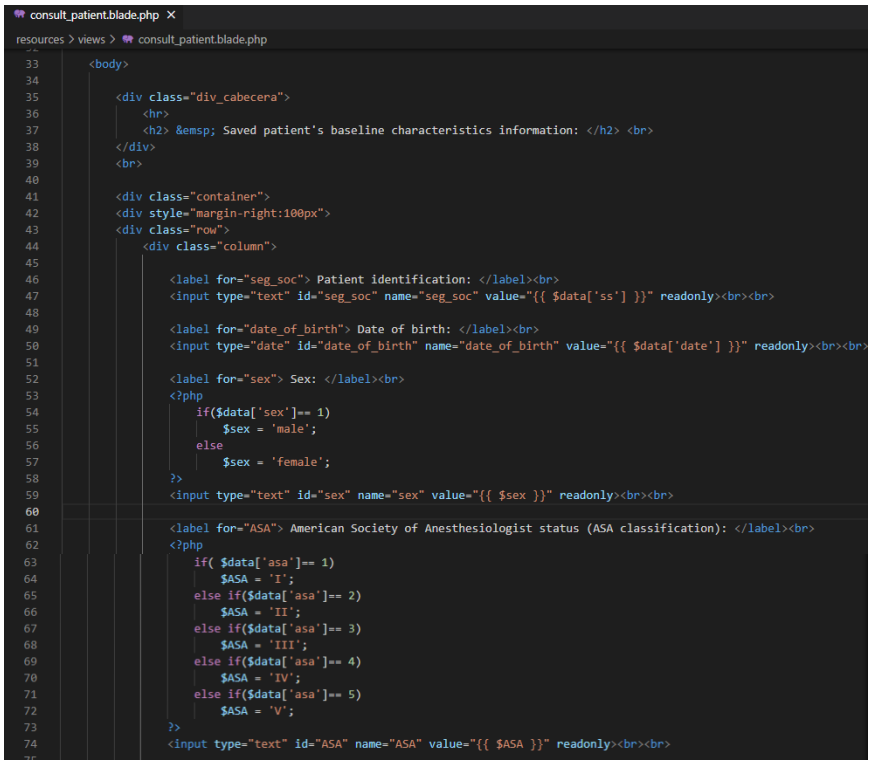


Fig. 29 Vista consulta información paciente

La Figura 30 corresponde a la vista de modificación de la información almacenada del paciente, permitiendo cambiarla.

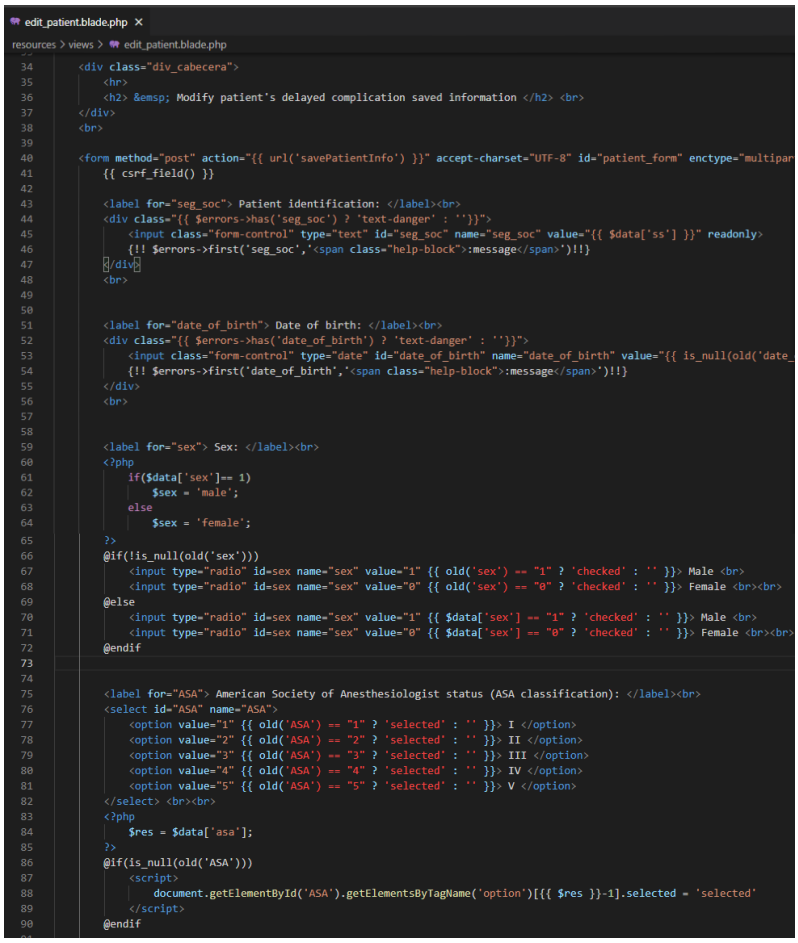


Fig. 30 Vista modificación de las características basales de un paciente

8. Aplicación final

En este apartado se presenta a través de numerosas capturas, el resultado final de la aplicación y la funcionalidad que ofrece. Se podrá observar cómo cumple con los casos de usos mencionados en la Figura 2.

8.1 Sistema de Login

La Figura 31 muestra el sistema de Login implementado.

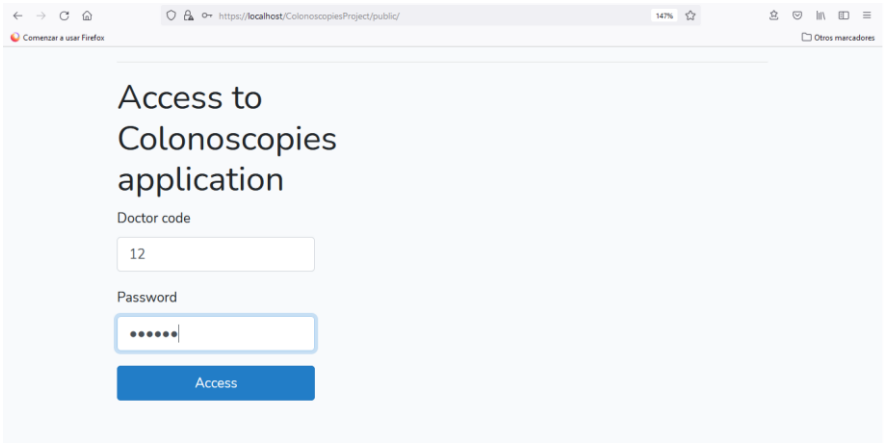


Fig. 31 Sistema de Login

Los campos “Doctor code” y “Password” son obligatorios para poder acceder a la aplicación. En caso de que alguno de los dos campos se quede sin rellenar, aparecerían los mensajes de error que se presentan en las Figuras [32,33].

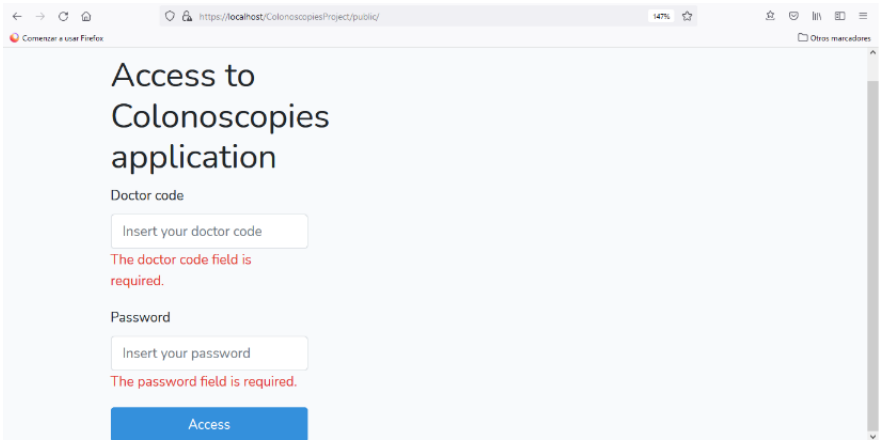


Fig. 32 Error faltan los dos campos del Login

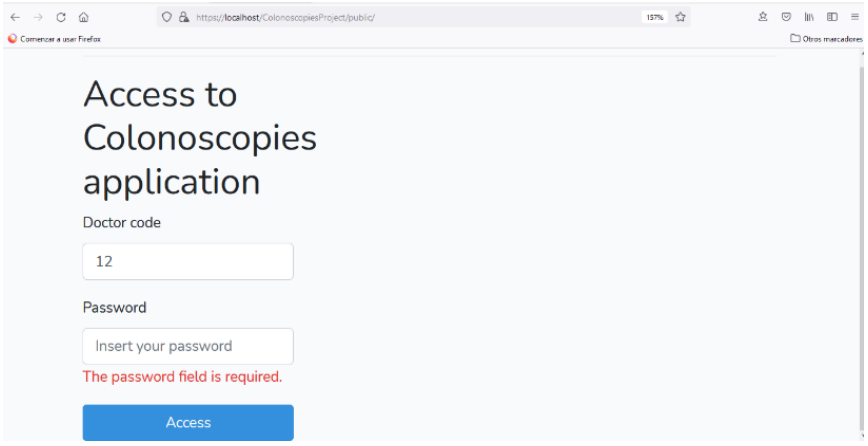


Fig. 33 Error contraseña incompleta en Login

En el caso de que un médico intente acceder a la aplicación a través de una contraseña que no sea correcta, aparecería el mensaje de error que se presenta en la Figura 34.

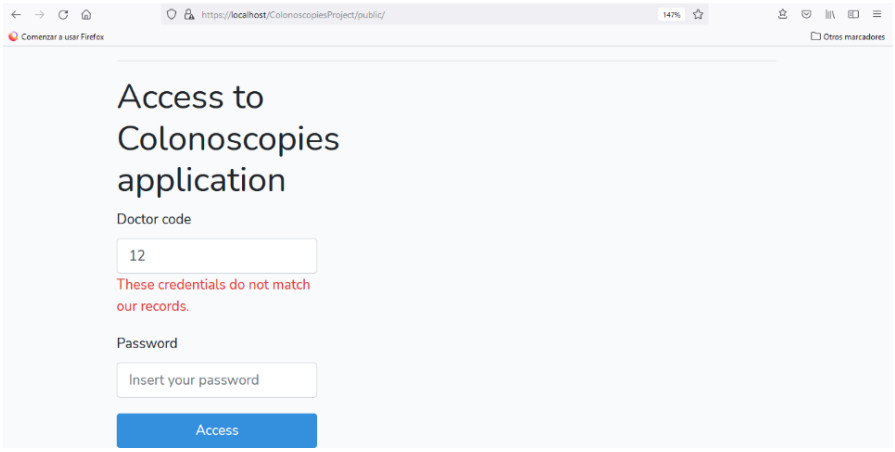


Fig. 34 Error en las credenciales del Login

8.2 Vistas acceso a la aplicación

8.2.1 Buscador de pacientes

Se puede observar la vista a la que se accede una vez se tiene acceso a la aplicación en la Figura 35.

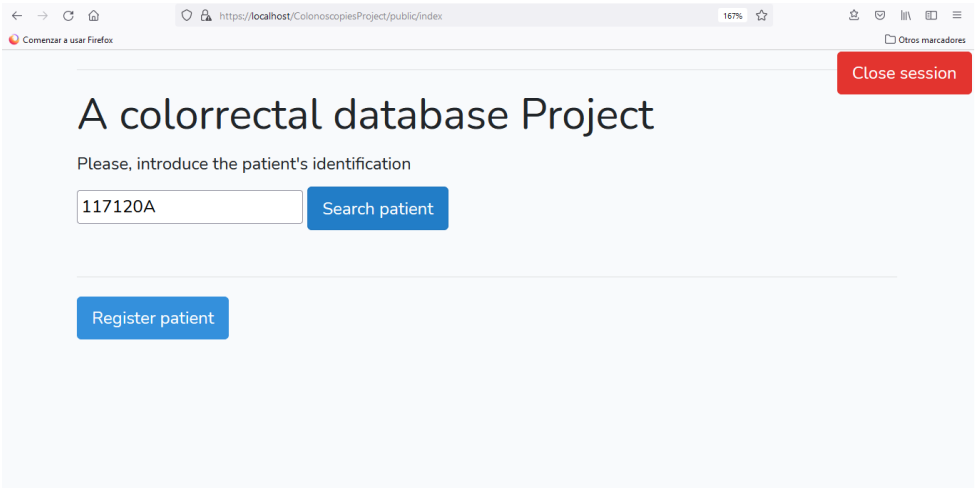


Fig. 35 Buscador de pacientes

En caso de que el paciente buscado no se hubiese dado de alta hasta ese momento en el hospital al que pertenece el médico especialista o ya se hubiera dado de alta en otro centro hospitalario, aparecería el mensaje de error representado en la Figura 36.

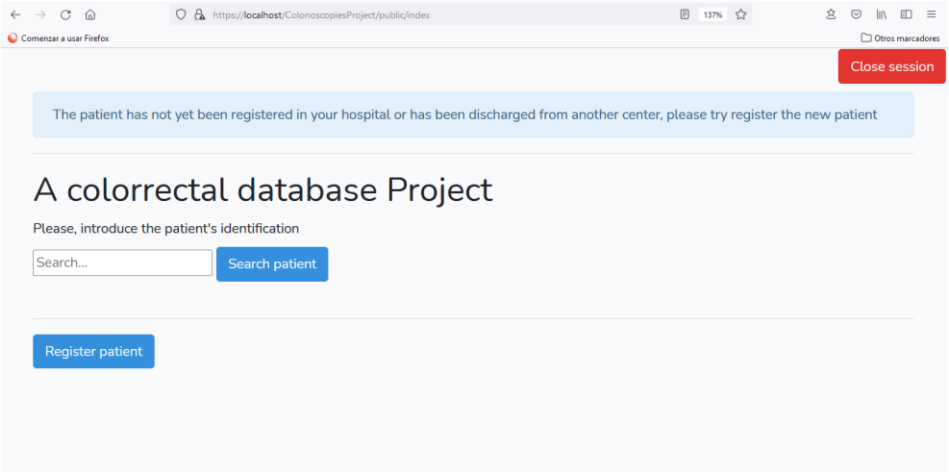


Fig. 36 Error en búsqueda de paciente

8.2.2 Vistas relacionadas con un paciente

El formulario que se presenta en la Figura 37 es el que se muestra una vez se clique en el botón "Register patient" o se clique en el botón "Search patient" teniendo acceso al paciente buscado.

En la Figura 37 también se puede observar que en caso de que se clique el botón "Submit patient's baseline characteristics form" y falte alguna información, aparecerá un mensaje de error debajo de aquellos campos que faltan por rellenar.

Esta vista es la que se muestra a partir del código que aparece en la Figura 28, de manera que esta sería el aspecto final.

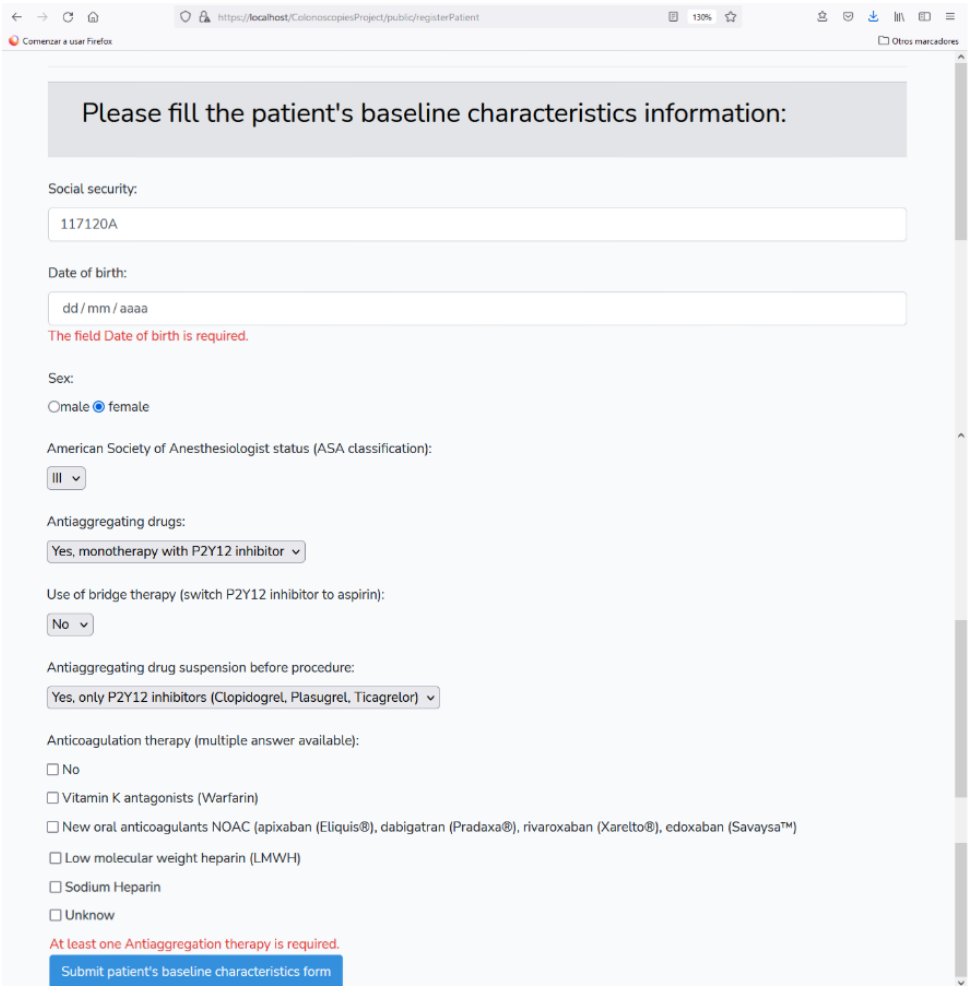


Fig. 37 Formulario registrar paciente

Una vez se hayan introducido todos los campos obligatorios y se pulse el botón de "Submit patient's baseline characteristics form", aparecerá la vista de la Figura 38 con las características basales guardadas del paciente.

Además, aparecerán los botones para poder acceder a los momentos de seguimiento del paciente pudiendo ver su estado dependiendo de su color.

Esta vista es la que se muestra a partir del código que aparece en la Figura 29, de manera que esta sería el aspecto final.

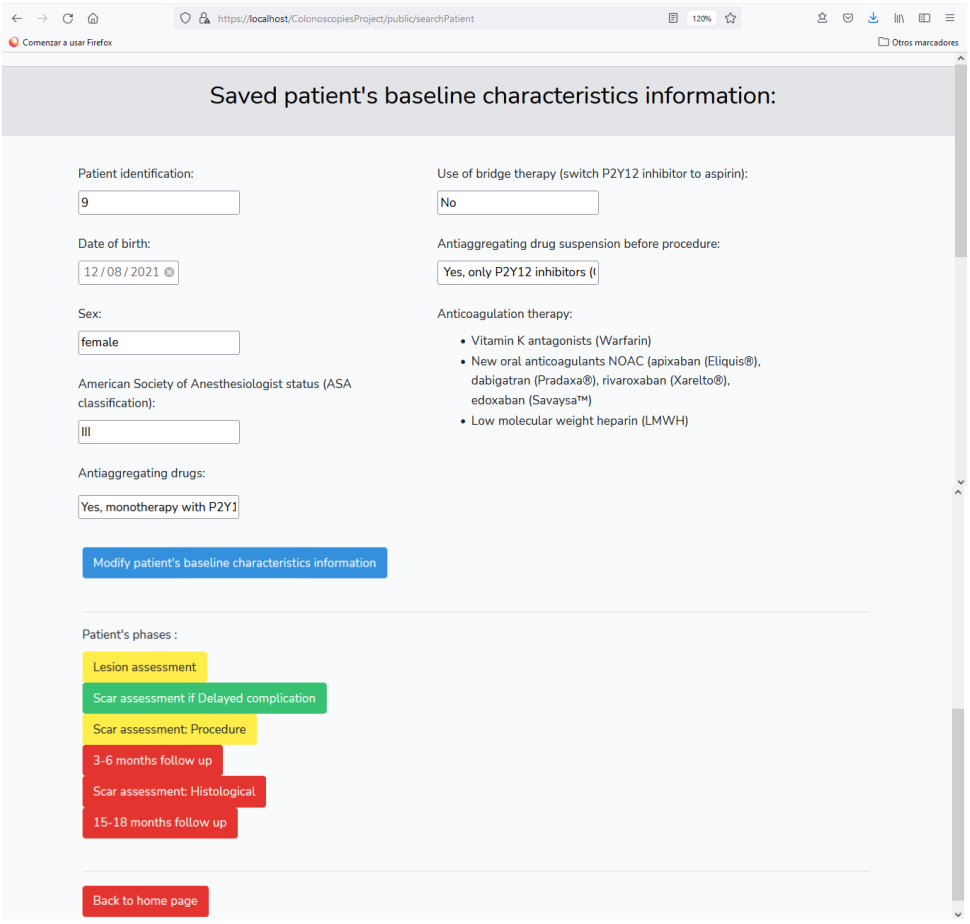


Fig. 38 Consulta de las características basales del paciente

Estas características basales del paciente se podrán modificar si se clica en el botón "Modify patient's baseline characteristics information".

8.2.3 Vista momentos paciente

Si se clica en algún momento del paciente en el cual no se haya introducido ninguna información todavía, aparecerá siempre un formulario como el de la Figura 39 a rellenar.

Se podrá seleccionar el estado del formulario a incompleto si no se dispone de toda la información en ese momento (incluyendo una imagen como mínimo). En caso contrario se deberá seleccionar a completo.

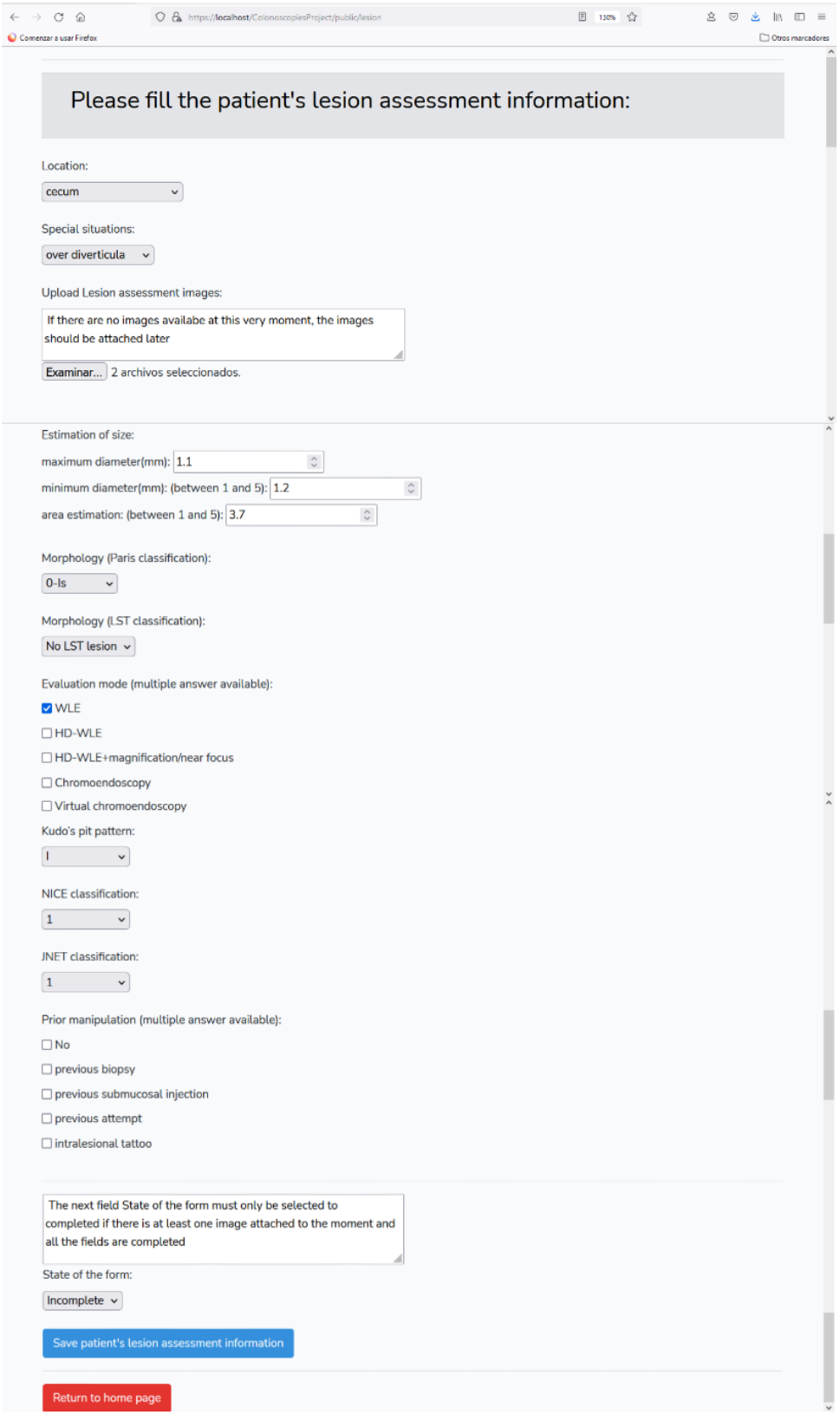


Fig. 39 Formulario lesión de paciente

Una vez el formulario se envíe, aparecerá una vista con sus datos almacenados. En la siguiente Figura 40 se puede observar los datos guardados en “lesion assessment” y se puede acceder a los siguientes momentos del paciente.

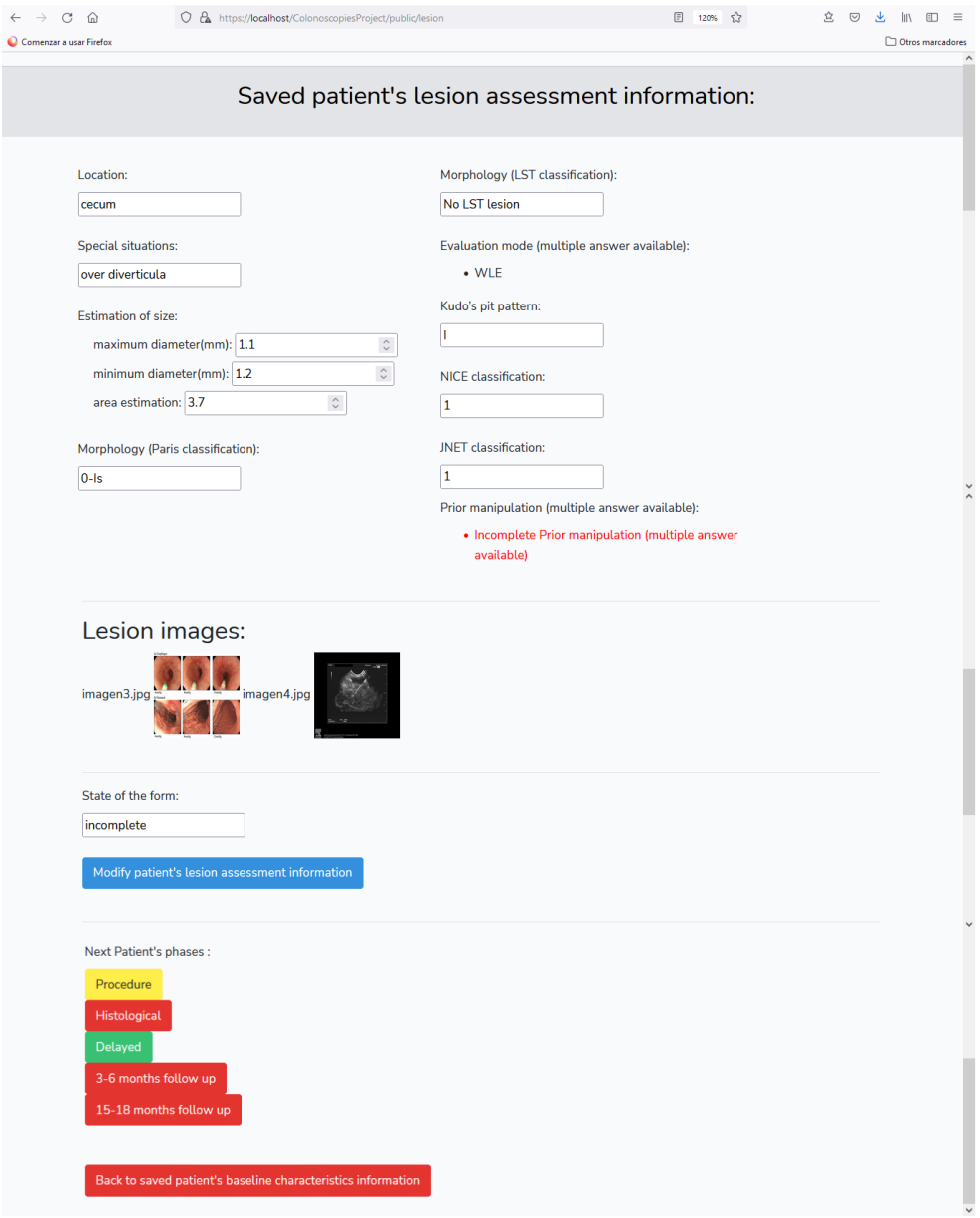


Fig. 40 Vista consulta información de lesión de paciente

Siempre se puede modificar la información almacenada del momento: modificar los campos, añadir y eliminar imágenes endoscópicas del momento del paciente y cambiar el estado del momento.

En este caso clicamos el botón “Modify patient’s lesion assessment information”, y rellenamos todos los campos, seleccionando el estado a “Complete”. Esto queda reflejado en la Figura 41.

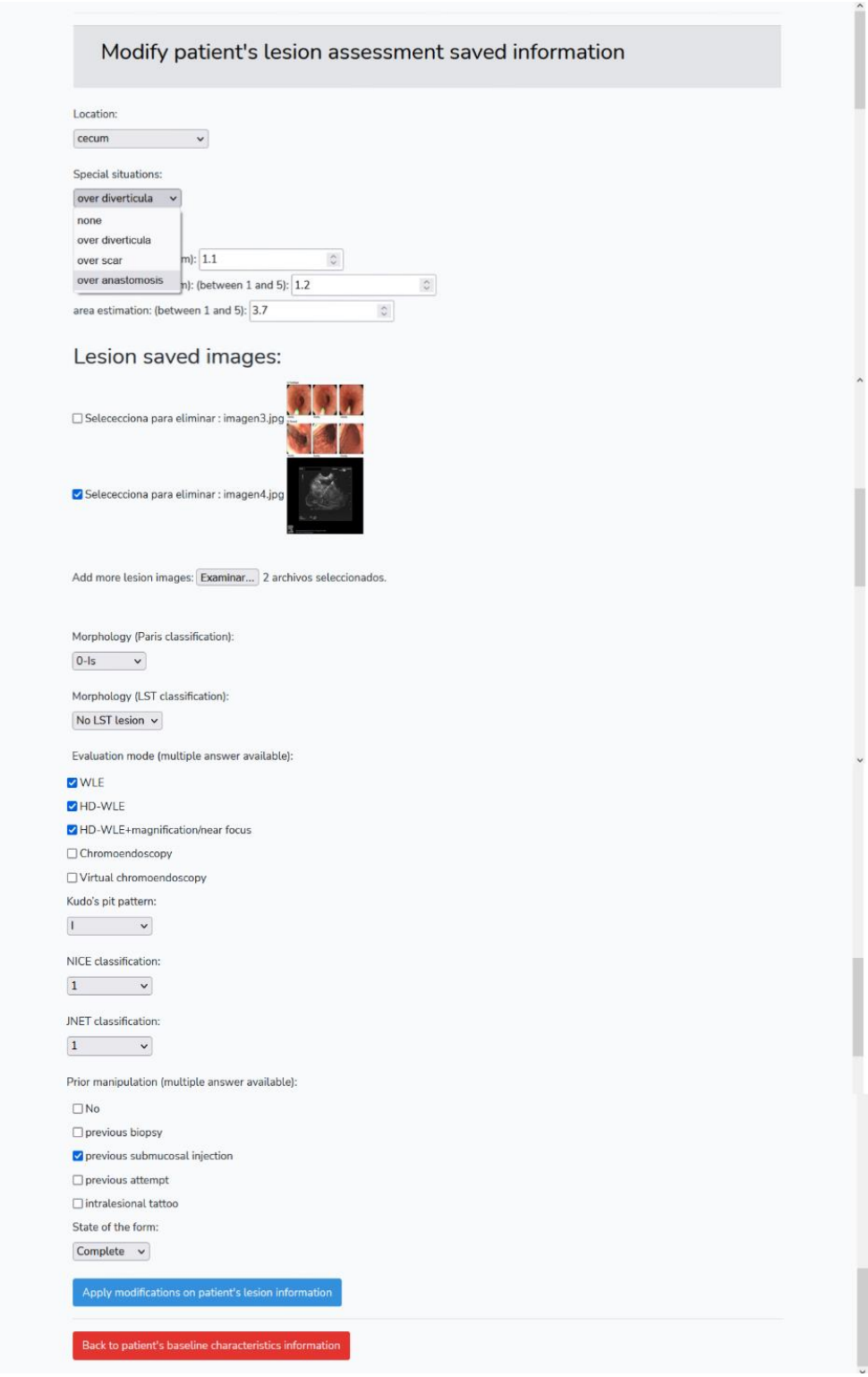


Fig. 41 Vista modificar información de lesión de paciente

Ahora, una vez cliquemos en “Apply modifications on patient’s lesion information”, se accederá a la vista de la Figura 42.

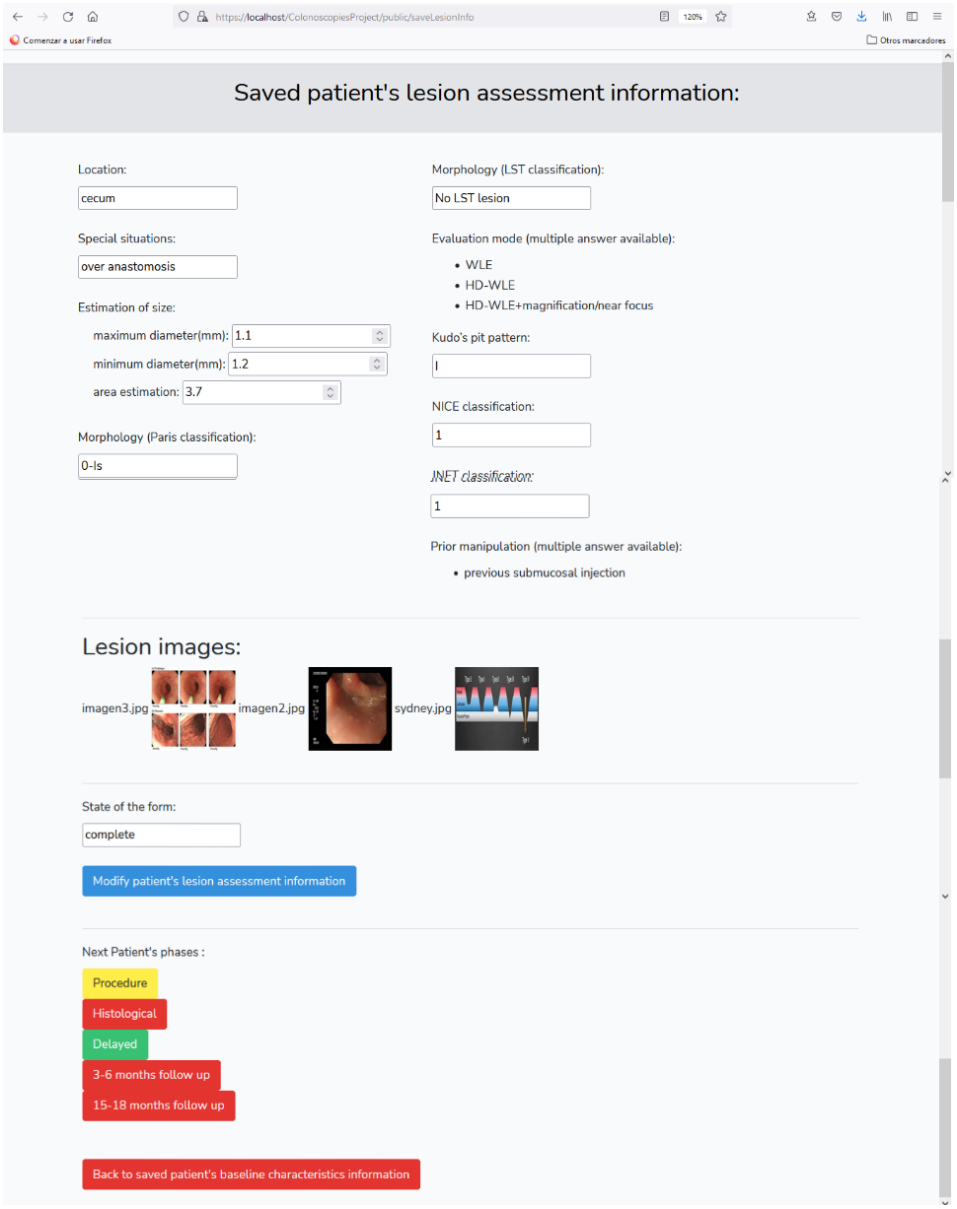


Fig. 42 Vista de la información de lesión modificada

Se han implementado las mismas vistas para cada uno de los momentos de seguimiento del paciente.

9. Conclusiones

9.1 Relación proyecto-estudios cursados

La realización de este proyecto ha sido una forma de aplicar los conocimientos adquiridos en diferentes asignaturas impartidas y además en un problema real.

Las asignaturas de “Gestión de proyectos informáticos I y II” fueron esenciales en la gestión de todo el proyecto. Aportaron el conocimiento básico acerca de los tipos de desarrollo software que existen, así como las fases en las que se dividen. Sabiendo que el desarrollo de todo proyecto software debe comenzar por una etapa de planificación, en esta se eligió el tipo de desarrollo que se debía utilizar dependiendo de las características del proyecto. En esta etapa también se eligieron las tecnologías y herramientas que se iban a utilizar. Permitiendo el desarrollo de una manera ágil y eficaz.

Los conocimientos adquiridos en estas asignaturas también fueron fundamentales para tratar con el cliente de manera adecuada, recoger sus impresiones tras una demostración del prototipo y saber gestionarlas, permitiendo añadir y/o modificar funcionalidades a la aplicación. Además, gracias a la asignatura de “Análisis y Diseño del software” se pudo modelar los requisitos obtenidos a través de los diagramas UML. Permitiendo tener siempre una visión de la funcionalidad que necesitaba cumplir el sistema.

La asignatura “Base de datos” aportó unos conocimientos muy necesarios para la creación de la base de datos de este proyecto. Conocimientos básicos sobre la forma de abstraer un problema real y poder diseñar y crear la arquitectura de los datos. De esta manera se pudo obtener el Modelo Relacional de la Base de datos que se necesitaba implementar.

Los conocimientos básicos a cerca de la programación en los lenguajes clásicos en la creación de una página web, entre ellos PHP, HTML, CSS y JavaScript, fueron adquiridos en la asignatura “Sistemas de información web”. La forma de desarrollar bajo el patrón Modelo, Vista, Controlador también fue aprendida en esta asignatura.

9.2 Trabajo a futuro y líneas de mejora

El proyecto se encuentra en fase de testeo, a la espera de ponerlo en producción. En la fase de testeo se debe asegurar que se cumplen todas las normas vigentes en La Ley Orgánica de Protección de Datos [4]. En la fase de producción se realizará un despliegue de la aplicación en la plataforma “Giara IMage Labelling Infrastructure” [5], del grupo GIARA de la UPNA del departamento de Estadística, Informática y Matemáticas.

Una vez se haya puesto en producción a través la plataforma mencionada, los centros hospitalarios que deseen podrán utilizar esta aplicación web para poder realizar el seguimiento de sus pacientes con lesiones colorrectales. De manera que la base de datos creada podrá actuar como un gran banco de datos donde se recopilará información que será utilizada en el proyecto a gran escala de Inteligencia Artificial.

En el momento en que se considere que se disponga de suficiente información almacenada en la base de datos, dará comienzo el proyecto de Inteligencia Artificial para la predicción de posibles complicaciones de pacientes y mejorar sus tratamientos endoscópicos. Se iniciará con el proceso de creación de datasets. Se implementarán diferentes métodos de predicción a través de minería de datos y/o Deep Learning, así como técnicas de tratamiento

para las imágenes endoscópicas. Para finalmente explotar la información almacenada con estas técnicas.

10. Referencias

[1] R. (2021, 21 junio). «IA y Neoplasias Gastrointestinales» y «Nora» ganan el Concurso Medtech. Navarra Capital. https://navarracapital.es/ia-y-neoplasias-gastrointestinales-y-nora-ganan-el-concurso-medtech/

[2] Eduardo Albéniz, Marco Antonio Álvarez, Jorge C. Espinós, Oscar Nogales, Carlos Guarner, Pedro Alonso, Manuel Rodríguez-Téllez, Alberto Herreros de Tejada, José Santiago, Marco Bustamante-Balén, Joaquín Rodríguez Sánchez, Felipe Ramos-Zabala, Eduardo Valdivielso, Felipe Martínez-Alcalá, María Fraile, Alfonso Elosua, María Fernanda Guerra Veloz, Berta Ibáñez Beroiz, Ferrán Capdevila, Mónica Enguita-Germán,

Clip Closure After Resection of Large Colorectal Lesions With Substantial Risk of Bleeding,

Gastroenterology, Volume 157, Issue 5, 2019, Pages 1213-1221.e4, ISSN 0016-5085, https://doi.org/10.1053/j.gastro.2019.07.037.
(https://www.sciencedirect.com/science/article/pii/S0016508519411347)

[3] Eduardo Albéniz, María Fraile, Berta Ibáñez, Pedro Alonso-Aguirre, David Martínez-Ares, Santiago Soto, Carla Jerusalén Gargallo, Felipe Ramos Zabala, Marco Antonio Álvarez, Joaquín Rodríguez-Sánchez, Fernando Múgica, Óscar Nogales, Alberto Herreros de Tejada, Eduardo Redondo, Noel Pin, Helena León-Brito, Remedios Pardeiro, Leopoldo López-Roses, Manuel Rodríguez-Téllez, Alejandra Jiménez, Felipe Martínez-Alcalá, Orlando García, Joaquín de la Peña, Akiko Ono, Fernando Alberca de las Parras, María Pellisé, Liseth Rivero, Esteban Saperas, Francisco Pérez-Roldán, Antonio Pueyo Royo, Javier Eguaras Ros, Alba Zúñiga Ripa, Mar Concepción-Martín, Patricia Huelin-Álvarez, Juan Colán-Hernández, Joaquín Cubiella, David Remedios, Xavier Bessa i Caserras, Bartolomé López-Viedma, Julyssa Cobian, Mariano González-Haba, José Santiago, Juan Gabriel Martínez-Cara, Eduardo Valdivielso, Carlos Guarner-Argente,

A Scoring System to Determine Risk of Delayed Bleeding After Endoscopic Mucosal Resection of Large Colorectal Lesions, Clinical Gastroenterology and Hepatology, Volume 14, Issue 8, 2016, Pages 1140-1147, ISSN 1542-3565, https://doi.org/10.1016/j.cgh.2016.03.021.

(https://www.sciencedirect.com/science/article/pii/S1542356516002779)

[4] Sáez, Edmundo. (2011). La protección de datos personales en el desarrollo de software. Novática. 50-55.

[5] Login:: Giara IMage Labelling Infrastructure. (2021). Giara IMage Labelling Infrastructure. https://gimli.dev/admin/login

[6] Ju Gang Nam, Sunggyun Park, Eui Jin Hwang, Jong Hyuk Lee, Kwang-Nam Jin, Kun Young Lim, Thienkai Huy Vu, Jae Ho SohnU, Sangheum Hwang, Jin Mo Goo, Chang Min Park. Development and Validation of Deep Learning–based Automatic Detection Algorithm for Malignant Pulmonary Nodules on Chest Radiographs. Journal Article. Radiology 2019 290:1, 218-228
https://pubs.rsna.org/doi/abs/10.1148/radiol.2018180237
https://pubs.rsna.org/doi/full/10.1148/radiol.2018180237

[7] Yun Liu, Timo Kohlberger, Mohammad Norouzi, George E. Dahl, Jenny L. Smith, Arash Mohtashamian, Niels Olson, Lily H. Peng, Jason D. Hipp, Martin C. Stumpe; Artificial Intelligence–Based Breast Cancer Nodal Metastasis Detection: Insights Into the Black Box for Pathologists. Arch Pathol Lab Med 1 July 2019; 143 (7): 859–868. doi: https://doi.org/10.5858/arpa.2018-0147-OA

[8] REDCap. (2021). AI COLORECTAL LESIONS DATABASE: COMPLICATIONS AND INCOMPLETERESECTION/RECURRENCE.
http://redcap.wseed.org/redcap/redcap_v8.5.6/ProjectSetup/index.php?pid=38

[9] HTML: HyperText Markup Language | MDN. (2021, 8 julio). MDN Web Docs/HTML. https://developer.mozilla.org/en-US/docs/Web/HTML

[10] PHP: Hypertext Preprocessor. (2021, 19 agosto). PHP. https://www.php.net/

[11] Cascading Style Sheets. (2021). CSS. https://www.w3.org/Style/CSS/

[12] JavaScript.com. (2021). JavaScript. https://www.javascript.com/

[13] Microsoft. (2016, 14 abril). Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/

[14] XAMPP Installers and Downloads for Apache Friends. (2021). XAMPP. https://www.apachefriends.org/es/index.html

[15] Laravel - The PHP Framework For Web Artisans. (2021). Laravel. https://laravel.com/

[16] MariaDB Foundation. (2019, 13 noviembre). MariaDB.Org. https://mariadb.org/

[17] Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans. (2021). Eloquent. https://laravel.com/docs/8.x/eloquent

[18] "IEEE Recommended Practice for Software Requirements Specifications," in IEEE Std 830-1998 , vol., no., pp.1-40, 20 Oct. 1998, doi: 10.1109/IEEESTD.1998.88286.

[19] "ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering," in ISO/IEC/IEEE 29148:2011(E) , vol., no., pp.1-94, 1 Dec. 2011, doi: 10.1109/IEEESTD.2011.6146379.

[20] What is UML | Unified Modeling Language. (2005). UML. https://www.uml.org/what-is-uml.htm