UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Diseño, desarrollo e implementación de un servicio web para la anotación de imágenes de alta resolución: Integración de modelos de Deep Learning para la segmentación automática de regiones.

Trabajo Fin de Máster

Máster Universitario en Gestión de la Información

AUTOR/A: Pulgarín Ospina, Cristian Camilo

Tutor/a: Valderas Aranda, Pedro José

Director/a Experimental: COLOMER GRANERO, ADRIAN

CURSO ACADÉMICO: 2021/2022

Agradecimientos

A mi madre Consuelo por apoyarme incondicionalmente y creer siempre en mi, a mi padre Luis por la cultura del esfuerzo, a mi pareja Lizz, a mi familia, a la familia de CVBlab y a mi tutor de TFM.

Resumen

La sociedad actual está muy digitalizada, gracias a esto se consigue generar inmensas cantidades de datos que si son analizados aportan mucha información que puede ser usada para realizar importantes avances en múltiples campos, uno de estos campos es él de la salud, prueba de ello es la revolución digital en métodos clínicos como el estudio histológico de imágenes tumores, el paradigma en el *Workflow* de la patología tradicional ha hecho que se estandarice el uso de imágenes digitales *Whole Slide Images* en detrimento de las muestras microscópicas, esto se debe en gran parte a la cantidad de beneficios que plantes, una de ellas es el uso de software que permiten la visualización de las imágenes a nivel de píxel con un nivel de calidad óptimo. El grupo CVBlab ofrece una herramientas para la visualización de las imágenes WSI que además tiene funcionalidades para anotar imágenes, esta aplicación tiene el objetivo de facilitar la anotación a los patólogos para generar bases de datos robustas con las que entrenar modelos de *Deep Learning*, el problema es que este sistema tiene dificultades para dar un servicio óptimo al volumen de casos que se quieren evaluar y además no están implementadas nuevas funcionalidades que son necesarias para que ajustarse a la exigencia de determinados proyectos, una de ellas es la implementación de modelos de *Deep Learning* en la propia aplicación para evaluarlos con metodologías *Crowdsourcing*, en este contexto se ha realizado un análisis minucioso de la infraestructura y el código para buscar mejoras y sintetizar una estrategia de mojara de la misma, además se han diseñado e implementado novedosas extensiones que permiten integrar modelos de *Deep Larning* y realizar tareas de *Crowdsourcing*.

Palabras clave: Patología Digital,Procesamiento de imagen, Servicios Web, Crowdsourcing, Explotación de datos, Aprendizaje profundo, Big data.

Resum

La societat actual està molt digitalitzada, gràcies a això s'aconsegueix generar immenses quantitats de dades que si són analitzats aporten molta informació que pot ser usada per a realitzar importants avanços en múltiples camps, un d'aquestos camps es el de la salut, prova d'això és la revolució digital en mètodes clínics com l'estudi histològic d'imatges tumors, el paradigma en el *Workflow* de la patologia tradicional ha fet que s'estandarditze l'ús d'imatges digitals *Whole Slide Images* en detriment de les mostres microscòpiques, això es deu en gran part a la quantitat de beneficis que aquest metode presenta, una d'ells e l'ús de programes que permeten la visualització de imatges a nivell de píxel amb un nivell de qualitat òptim. El grup CVBlab ofereix una eina per a la visualització de les imatges WSI que a més té funcionalitats per a anotar imatges, aquesta aplicació té l'objectiu de facilitar l'anotació als patòlegs pera generen bases de dades robustes amb les quals entrenar models de *Deep Learning*, el problema és que aquest sistema té dificultats per a donar un servei òptim al volum de casos que es volen avaluar i a més no estan implementades noves funcionalitats que són necessàries per ajustar-se a l'exigència de determinats projectes, una d'elles és la implementació de models de *Deep Learning* en la pròpia aplicació per a avaluar-los amb metodologies *Crowdsourcing*,en aquest context es realitzat una anàlisi minuciosa de la infraestructura i el codi per a buscar millores i sintetitzar una estratègia de banyara d'aquesta, a més s'han dissenyat i implementat noves extensions que permeten integrar models de *Deep Learning* i fer tasques de *Crowdsourcing*.

Paraules clau: Patologia Digital,Processament d'imatge Serveis Web ,Crowdsourcig Explotació de dades, Aprenentatge profund,Big data.

Abstract

Today's society is highly digitalized, thanks to this, it is possible to generate immense amounts of data which, if analyzed, provide a great deal of information that can be used to make significant advances in many fields, one of these fields is health, proof of which is the digital revolution in clinical methods such as the histological study tumor images, the paradigm in the workflow of traditional pathology has led to the standardization of the use of digital images to the detriment of microscopic samples, this is mainly due to the number of benefits that they offer, one of which is the use of software that allows the visualization of images at pixel level with an optimum level of quality. The CVBlab group provides tools for the visualization of WSI images that also has functionalities for anotatinging images. This application aims to facilitate annotation for pathologists in order to generate robust databases with which to train Deep Learning models, The problem is that this system has difficulties to provide an optimal service to the volume of cases to be evaluated and also new functionalities that are necessary to meet the requirements of certain projects are not implemented, one of them is the implementation of Deep Learningmodelss in the application itself to evaluate them with Crowdsourcing methodologies. In this context, a thorough analysis of the infrastructure and code has been carried out in order to search for improvements and synthesize a strategy for its wetting, in addition, novel extensions have been designed and implemented to integrate models of crowdsourcing and crowdsourcing tasks.

Key words: Digital Pathology, Images Processing, Web Services, Crowdsourcig, Data Explotation, Deep Learning, Big Data.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

La sociedad está viviendo una gran revolución en el campo de los datos y la información. Este hecho se debe al desarrollo de instrumentación y sensores de alta sensibilidad que permiten realizar medidas qué dan como resultados volúmenes inmensos de datos, esto unido a la digitalización y a los avances en telecomunicaciones propician la adquisición y transmisión masiva de datos.

Hasta hace unos años todos los datos generados eran desechados, esto se debía en gran parte a que no se disponía de infraestructuras y métodos capaces de almacenar cantidades masivas de datos. Con la aparición de clústeres de computación (redes formadas por nodos que funcionan de forma distribuida) y *frameworks* de código abierto como *Hadoop*[20] o *Sparks*[21] se comenzó a desarrollar aplicaciones para realizar tareas de almacenaje y manejo de datos masivos o *Big Data* (en adelante BD).

La motivación detrás de almacenar grandes cantidades de datos nace de conocer de una forma más precisa: el comportamiento de una especie animal en su hábitat, los procesos físicos desencadenados por una supernova, datos de sensores meteorológicos o aspectos menos científicos como el caracterizar a un cliente o sacar información detallada de cómo ha funcionado un producto en el mercado. Lo que se busca después de almacenar los datos es generar información que sea de interés.

El proceso de destilación de información a partir de los datos es un proceso con un costo de computación que aumenta con la cantidad de datos a manejar. El análisis de grandes cantidades datos conlleva un costo de recursos importantísimo, actualmente se puede hacer frente a este costo gracias al avance en hardware, el avance en este campo permite tener clústeres capaces de procesar TB (Terabyte) de datos en paralelo. Poder procesar a la vez bases de datos tan masivas facilita la tarea de aplicar distintos tipos de análisis estadísticos que permitan caracterizar la naturaleza de cada variable y la forma en la que se relacionan con las otras.

Conociendo la relación que poseen las variables entre ellas se pueden realizar proyecciones en el futuro de una variable o un conjunto de ellas, por ejemplo, si se plantea un contexto en el que una variable decrece y sabemos que es directamente proporcional a otra podemos afirmar con cierto grado de certeza que esa variable aumentará y viceversa. El uso de modelos matemáticos que simule el comportamiento de variables en distintos

escenarios es el objetivo principal de logaritmos de *Machine Learning* (ML) como: Árboles de decisión, Naïve Bayes[1], Regresiones Lineales, Regresiones Logísticas, *Suport Vector Machine*(SVM)[2], Análisis de Componentes Principales (PCA)[3], entre otros muchos.

La capacidad de tener modelos de predicción supone un avance muy importante en todos los ámbitos donde son aplicables, poder predecir el comportamiento de una variable simulando diferentes escenarios es una herramienta diferencial que supone tomar decisiones o estrategias a tiempo que garanticen la consecución de un objetivo o meta.

La gran cantidad de datos adquiridos no aportan ningún valor si no son: almacenados, preprocesados, analizados y representados. Como resultado de todo este *pipline* se consigue aportar valor a los datos en forma de conocimiento cuya finalidad es la de ayudar a tomar decisiones, por ejemplo, alcanzar unas ganancias económicas por la venta de un producto, determinar que tipo de campaña de marketing seguir para llegar a un número mayor de consumidores, predecir la población de una especie afectada por el cambio climático o pronosticar eventos meteorológicos.

Obtener conocimiento del dato es hacia donde se encamina la economía, ya que siguiendo el *pipline* explicado se obtiene conocimiento que puede situar a diferentes empresas en una posición favorable respecto a competidores directos, pero la metodología es aplicable a muchos otros campos.

Uno de los campos en los que se aplica métodos de BD es el médico-científico para tareas como: secuenciación gnómica, caracterización de pacientes, detección de factores de riego de distintas enfermedades, además, se está trabajando en la optimización del diagnóstico de enfermedades como el cáncer, ya que un diagnóstico rápido y acertado es clave para el tratamiento de este tipo de patología y el uso de técnicas de BD puede ayudar mucho en este tipo de tareas.

En el campo de la histopatología se está investigando técnicas y métodos que agilicen el proceso de diagnóstico, un ejemplo es la introducción de la tecnología *Whole Slide Image* (en adelante WSI). Hasta la aparición de las WSI el *Workflow* de los patólogos involucraba: la adquisición de las muestras de tejido, el procesado de la muestra, la elaboración del cristal con la *slide* y el análisis con microscopio que en el mejor de los casos es analizado en el mismo hospital por un experto y en el peor de los casos debe de ser enviada por correo para que un patólogo externo diagnostique la muestra, este *Workflow* hace que el diagnóstico sea lento y deficiente en muchos casos para el número de muestras a evaluar.

La tecnología WSI ha cambiado la metodología de diagnóstico patológico. La digitalización de las imágenes consigue que los tiempos de envío de muestras quede reducido al tiempo que se tarda en escanear la muestra, subirla a un servidor y ponerla disponible a través de la red al patólogo experto encargado del diagnóstico, este método logia de trabajo usando las bondades de las telecomunicaciones actuales agiliza mucho los tiempos, cambiando la forma en la que se diagnostica. Este sistema ofrece flexibilidad, seguridad y eficiencia en el proceso de diagnóstico [5], el beneficio que ofrece es tan notable que el diagnóstico digital se ha ido convirtiendo en un estándar, siendo aceptado cada vez más entre los expertos[4].

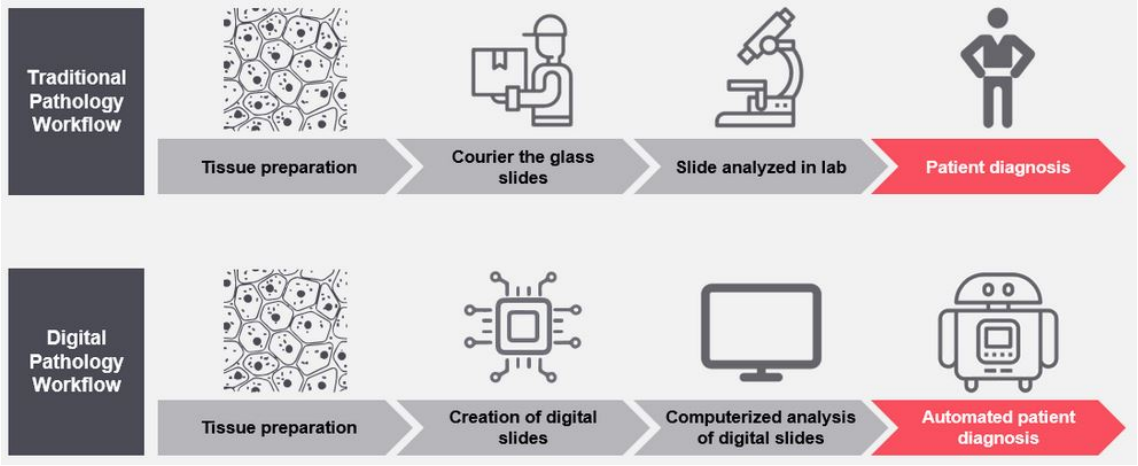


Figura 1.1: Comparación del *Workflow* tradicional y digital [24].

Siendo cierto que la patología digital ha agilizado los tiempos de diagnóstico, sigue siendo un proceso que consume mucho tiempo, el cuello de botella tras la digitalización de las imágenes está en el análisis minucioso que se debe realizar sobre toda la WSI. Las WSI son imágenes gigapíxel microscópicas que representan con gran calidad todos los elementos del tejido escaneado, al cual se le aplica una tinción para que resalten mejor las células y se pueda ver mejor la anatomía de la muestra. En algunos tipos de cáncer la presencia de una única mitosis cambia totalmente el diagnóstico patológico, encontrar células en este estado entre miles dentro la es una tarea difícil, además conlleva un coste importante de tiempo de análisis, para diagnosticar otros tipos de cáncer se tiene que dar un conjunto de fenómenos en los tejidos que en algunos casos son comunes entre tipos de cáncer, pero no todos, por lo que la varianza en un alguna de las características puede cambiar el tipo de cáncer o el grado.

Las tareas de diagnóstico patológico se ha ido automatizando, esto ha dado lugar a herramientas que, con ayuda de ML, ayudan a evaluar determinados biomarcadores en las pruebas Inmunohistoquímica (IHC) de los tejidos, detectando formas y patrones muy reconocibles que se le muestran resaltados a los patólogos, para que estos no tengan que invertir tiempo en detectarlos si no evaluándolos. Otras metodologías aplicadas son las relacionadas con el tratado de imagen. Una técnica muy sencilla es el *Threshold*, que permite segmentar tejido según el nivel de color con respecto al fondo, pero estás técnica son muy básicas comparadas con el *Deep Learning* (en adelante DL) que permiten hacer predicciones de regiones de interés, estructuras anotadas sobre la WSI (en adelante ROI), e incluso emitir diagnósticos en WSI.

Según el informe del 2022 emitido por la Sociedad Española de Oncología Médica [22] la incidencia de tumores en la población mundial es de 18,1 millones de personas con unas cifras de mortalidad de 9,96 M, se estima que para 2040 estas cifras aumenten 49,2 % en el número de casos y un 63,7 % en la mortalidad, a este hecho se une la previsión de falta de expertos en anatomía patológica para el año 2035, según indican datos del informe oferta y necesidad de médicos especialistas en España [23] que indica un creci-

miento muy bajo con respecto a la gran demanda que habrá en el futuro, en los próximos trece años solo aumentará un 0,5 el ratio de especialistas por cada cien mil habitantes.

Estos datos indican que en un futuro habrá más volumen de pacientes con cáncer pero menos expertos en diagnóstico. El diagnóstico patológico es un proceso laborioso y que conlleva ser muy minucioso a la hora de analizar cada uno de los tejidos de una biopsia. La tecnología actual ha conseguido un nivel muy alto de detalle en las imágenes histopatológicas, esto favorece un diagnóstico más acertado, pero incrementa el tiempo de análisis por parte de los expertos, si a esto añadimos los datos que indican un aumento significativo en el volumen de pacientes y una tasa baja de expertos por habitantes, da como resultado un futuro en el que no se podrá hacer frente a la creciente demanda de diagnósticos histopatológicos de una forma eficiente. En un contexto en el que el diagnóstico temprano y acertado del cáncer es tan importante, la dilatación en los tiempos de diagnóstico es un problema gravísimo al que hay que buscar solución.

Los modelos de DL más avanzados, son modelos entrenados a partir de *datasets* de imágenes anotadas por patólogos expertos, es decir, mediante el conocimiento volcado por los patólogos en las anotaciones realizadas en las imágenes gigapixel se logra obtener bases de datos masivas que con sofisticados algoritmos de tratado de imagen y DL logra destilar conocimiento que es aplicable para otros casos de patología digital.

1.1 Motivación

En la práctica clínica, el diagnóstico definitivo se obtiene mediante el análisis de una muestra de tejido del órgano de interés, conocida como biopsia. Este análisis es realizado por expertos patólogos utilizando un microscopio de gran resolución. Este proceso diagnóstico consume grandes cantidades de tiempo debido al gran tamaño de las biopsias, y sufre de variabilidad inter-patólogo debido al gran nivel de especialización requerido para realizarlo. En los últimos años, gracias al desarrollo de dispositivos de digitalización a estas magnificaciones, el uso de técnicas de visión por computador, ha demostrado resultados prometedores a la hora de automatizar este proceso y servir de apoyo al patólogo. En concreto, las técnicas de DL se han convertido en la principal opción metodológica para analizar e interpretar las imágenes de histología. En la bibliografía científica, distintos trabajos han demostrado que estos sistemas son capaces de obtener resultados al nivel del diagnóstico realizado por expertos patólogos en distintos tipos de cáncer. En concreto, sistemas de ayuda al diagnóstico con resultados prometedores basados en DL y redes neuronales convolucionales han sido propuestos para el diagnóstico de cáncer de próstata [6] [7] [8], mama [9], pulmón [10] o colon [11], entre otros. Sin embargo, las técnicas empleadas en estos trabajos incluyen técnicas computacionalmente costosas, tales como normalización de color, extracción de regiones de interés, y extracción de características de las imágenes mediante redes neuronales. Por ello, la translación de estos métodos a la práctica clínica, con las limitaciones computacionales y de tiempo real, supone un desafío en el uso de técnicas de visión por computador en el campo de la anatomía patológica.

El grupo "*Computer Vision and Behaviour Analysis Lab*"(en adelante CVBLAb) a colaborado en proyectos como SICAP cuyo objetivo es desarrollar un sistema de ayuda al diagnóstico para el cáncer de próstata clasificando las imágenes histopatológicas procedentes de biopsias en diferentes grados según la escala Gleason, y coordina el proyecto CLARIFY que propone crear una infraestructura de investigación basada en inteligencia artificial y algoritmos de datos orientados a la nube. La interpretación y el diagnóstico en base a grandes imágenes histológicas WSI's en todas partes a través de herramientas novedosas. Estas herramientas pueden suponer un cambio de paradigma en el campo de la patología con el objetivo de maximizar los beneficios de la patología digital y ayudar a los patólogos en su trabajo diario.

Se han desarrollado novedosos métodos de detección y diagnóstico [8], [12], [13] como fruto de los proyectos de CLARIFY y SICAP, los cuales no han podido incorporarse a la práctica clínica debido a la falta de los recursos y tecnologías que se precisan para desplegar arquitecturas tan complejas. Convierte la tarea de integrar estos avances en el proceso de diagnóstico y detección en un gran desafío.

Para el desarrollo de modelos de DL se necesita grandes conjuntos de datos etiquetados lo más heterogéneos posible, para que los modelos aprendan de ellos y puedan etiquetar nuevas muestras de la forma más precisa posible, en este caso se necesitan muchos casos de WSI con ROI etiquetadas, para la creación de estas bases de datos el grupo CVBLAB cuenta con una aplicación web para que los patólogos anoten dichas imágenes, esta aplicación fue diseñada para un volumen de usuarios, anotaciones y funcionalidades que se han quedado cortas para las tareas, proyectos y adquisición de datos que se necesitan en la actualidad, además, los novedosos modelos de DL fruto del conocimiento del grupo no se han aplicado aún a la práctica.

1.2 Objetivos

1.2.1. Objetivo Principal

El objetivo principal del presente trabajo de fin de máster es el de la evaluación y mejora de una herramienta web con la cual los patólogos pueden visualizar imágenes giga píxel, dotándoles de herramientas de anotación y segmentación automática de regiones de interés a partir de la salida de potentes modelos de Inteligencia Artificial e incluir un protocolo para métodos de *Crowdsorcing*, en el que se evalúe un modelo de segmentación de regiones tumorales entrenado a partir de anotaciones de expertos, con anotadores inexpertos.

1.2.2. Objetivos Secundarios

- Analizar la plataforma de anotación.
  - Detallar las funcionalidades de anotación.
  - Estudiar las tecnologías usadas en el Backend y el Frontend.

    - Observar la base de datos para conocer su estructura.
    - Analizar el manejo y representación de las WSI en la aplicación web.
- Analizar y mejorar la estrategia de despliegue de la aplicación.
- Integrar funciones de *Crowdsourcing*.
- Estudiar la integración de los modelos de DL creados.
    - Conocer los modelos DL a integrar.
    - Diseñar de estrategia de integración de los datos de salida.

Para explicar la metodología seguida para realizar el análisis y la implantación de nuevas funcionalidades que dan como resultado la correcta consecución de los objetivos planteados se propone la siguiente estructura: en primer lugar se realiza un estado del arte que tiene por objetivo la búsqueda de metodologías y tecnologías similares a las que se quieren implantar, en tercer lugar se realizará un análisis profundo de la herramienta Microdraw, para conocer en que se puede mejorar sus prestaciones para que ofrezcan un servicio, por último con los resultados del análisis se va a diseñar una estrategia de mejora y se van a implementar métodos para garantizar la funcionalidad de metodologías de *Crowdsourcing* y la integración de modelos de *Deep Learning*.

CAPÍTULO 2

Estado del Arte

Existen herramientas para visualizar, segmentar y etiquetar imágenes a nivel de píxel, como ImageJ[26], QUAPATH [15], ICY[16]. Estos programas permiten trabajar con imágenes en distintos formatos, como los de imágenes médicas.

ImageJ es una aplicación desarrollada en Java, se basa en NIH Image [25] de Macintosh, se puede descargar para distintos sistemas operativos como: Mac OS, Mac OS X, Linux y Windows, en un principio solo se podía descargar y usar en local, ahora ofrece la opción de acceder vía web [26] como se ve en la figura ImageJUi. Para usar la aplicación compilada en Java usa CheerpJ [27] que permite compilar y correr cualquier aplicación desarrollada con Java SE, librería o Java Applet en web, además, la nueva versión web de ImageJ permite el uso de plugins con modelos básicos de ML y DL, esta funcionalidad implementa usando ImJoy [28].

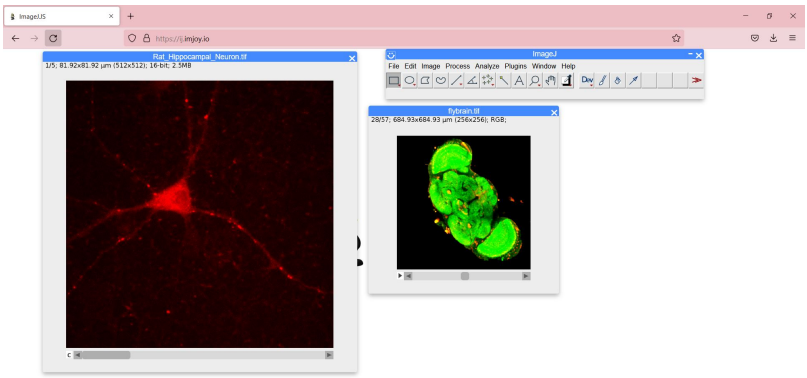


Figura 2.1: UI de ImageJ.

ImJoy es una plataforma *open-source* [29] para integrar de una forma sencilla aplicaciones de análisis de datos, especialmente aquellas que implican DL, esta plataforma es una web progresiva (*WPA*), por lo que es accesible desde cualquier tipo de dispositivo, además, es un servicio *Serverless*, esto significa que no precisa de un servidor fijo que atienda las peticiones de los clientes, el método es distinto, las aplicaciones *Serverless* se despliegan cuando se recibe una petición, ejecutan una tarea y se cierran, de esta forma no se aprovisionan recursos fijos sino de una forma dinámica. ImJoy es un integrador

de aplicación, así que puede comunicarse mediante *plugins* con plataformas o servicios codificados con diferentes lenguajes como por ejemplo Javascript o Python [14]. Como resultado del uso de esta tecnología se logra manejar las conexiones entre aplicaciones de distintas naturalezas, como por ejemplo ImageJ y modelos de DL implementados en Python, realizando el proceso de interconexión de una forma transparente al usuario.

ImageJ fue una de los primeros programas para el análisis de imágenes a nivel de píxel, sin ser una plataforma para un tipo de tarea en específico, su uso se extendió en la comunidad científica para realizar tareas de segmentación y visualización de imágenes. La necesidad de tener herramientas para cada tipo de tares hizo que surgieran otras plataformas basadas en ImageJ como: AstroImageJ, Bio7, Fiji Project, ImageJ2, µManager, etc. Entre las distintas herramientas, basadas en ImageJ surgieron ICY y QuPath que son plataformas para el análisis de imágenes biológicas e imágenes biomédicas respectivamente.

Icy [16] nació en 2012 como respuesta al problema de la reproducibilidad de experimentos con imágenes de biología, este problema resulta de la dificultad de reproducir resultados experimentales que incluían en las metodologías: el preprocesado de una imágenes, la técnica de tratado de esta y el análisis de imágenes sobre ella, este proceso se realiza usando unas técnicas y unas tecnologías diferentes según el *paper*, por lo que reproducir los mismos resultados y usarlo en un *paper* nuevo era de gran dificultad si no se usaban las mismas herramientas. Icy proponía una aplicación de análisis de imagen en la que mediante programación visual de bloques un investigador pudiera crear todo el *pipline* necesario para reproducir los resultados de su investigación, pero Icy iba más allá ya al basarse en ImageJ permitía el análisis de imagen, segmentación, detección de ROI's con métodos de ML y el uso de *plugins* para ampliar sus funcionalidades.

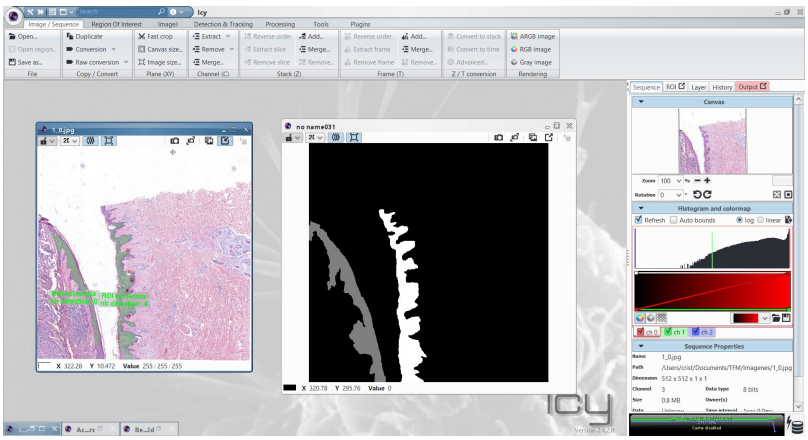


Figura 2.2: UI de Icy: Carga de una imagen y creación de un máscara.

Otra herramienta creada en 2012 para dar solución al problema de la reproducibilidad de experimentos es Fiji [17] es una aplicación creada en el mismo año para dar solución al mismo problema que Icy, en este caso Fiji propone una aplicación que permite la transformación de algoritmos en plugins para ImageJ, los creadores ven en su aplicación una herramienta que aumente la sinergia entre el campo de la ciencia de computadores y los investigadores en biología.

Tanto Icy como Fiji son aplicaciones que acercan el mundo de la computación, software y análisis de imagen a campos con menos conocimientos técnicos como la biología, pero para poder usar estas herramientas se necesita conocer las técnicas que ofrecen estos programas y, un mínimo de habilidades de computación, teniendo en cuanta estas limitaciones, estos dos servicios son apropiados para aquellos investigadores que conocen de ambos campos, como puede ser un investigador que desarrollo modelos de ML para la detección de forma automática de un tipo de tejido en una biopsia, pero no serían herramientas indicadas para la detección automática de ellas en uso clínico.

Hasta el 2017 las aplicaciones de visualización, manejo y análisis de imagen se centraban en imágenes biológicas con un sentido muy científico de investigación, hasta este año nace Qupath [15], este sofware, es una de las primeras aproximaciones que se realizan en el campo de la patología digital. QuPath es una plataforma de análisis de imágenes médicas, nace como un software de código abierto fácil de usar y extensible al análisis de WSI's. Esta herramienta ofrece herramientas para la identificación de tumores y biomarcadores, mediante el uso de árboles de decisión, modelos de ML, además, incluye la novedad de segmentar ROI's definiéndolas como objetos únicos, con lo que se consigue clasificar por tipos de ROI asignándoles una clase, forma o color. Igualmente QuPath innovo en el tipo de datos que se maneja, ya que esta plataforma permite el uso y manejo de imágenes WSI. Para ello se programó, en Java 8 con JavaFX para la interfaz de usuario y OpenSlide8 para el manejo de imágenes gigapíxel, ImageJ para el manejo de imágenes estándar.

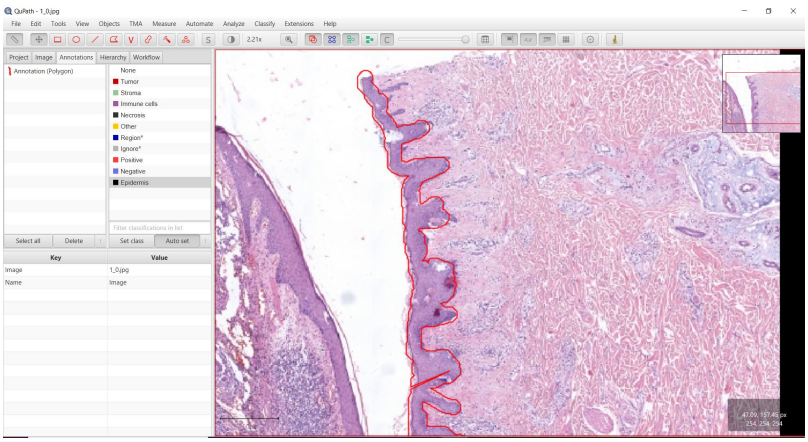


Figura 2.3: UI de QuPath.

Con los avances tecnológicos en hardware, especialmente en las *Graphics Processing Unit* en adelante GPU, se ha conseguido un nivel de clúster de procesamiento que permiten crear complejos modelos de DL, modelos que son capaces de realizar tareas de detección de objetos o segmentación más sofisticadas, el avance que ha vivido el este campo también ha afectado a la histopatología, prueba de ello son los múltiples proyectos nacionales y europeos que tienen como objetivo el desarrollo de modelos eficientes y sofisticado, que puedan ser usados clínicamente. Como consecuencia han nacido múltiples empresas financiadas con fondos públicos cuyo objetivo es el de crear este tipo de herramientas como: ContextVision [33], Aiforia [32] o Aignostic [31].

A raíz del interés de usar modelos DL en aplicaciones y servicios de histopatología a aumentado, las empresas privas han ido desarrollando sus propias soluciones para llegar a un mercado en pleno crecimiento. Empresas como ROCHE desarrollar su solución, Upath [30], una aplicación web que permite el uso de modelos de detección y medición semi-cuantitativa de la proteína PD-L1 "PD-L1 (SP263)" y "CPNM (CE-IVD)", modelos de determinación del estado del gen HER2 "ER2 Dual IS" y modelos de detección semi-cuantitativa de la proteína HER2 "HER2 (4B5)".Upath ofrece funcionalidades para cargar y manejar WSI's y segmentar ROI's. Además, esta aplicación cuenta con la integración de los formatos propios de los escáneres de biopsias que vende como producto ROCHE por lo que facilita el análisis en aquellas clínicas que posean este tipo de escáneres.

En la Tabla 2.1 se muestra un resumen de las tecnologías que se han detallado con anterioridad.

	ImageJ[26]	**Icy**[16]	**Fiji**[17]	**QuPath**[15]	**Upath**[30]
Año	1997	2012	2012	2017	2019
Tecnología	Java, CheeperJ e Imjoy.	Java	Java	Java8, JvaFx, OpenSlide8.	Plataforma cerrada.
Innovación	Análisis de imagen multiformato.	Programación gráfica por bloques del *pipline* de análisis de imagen.	Transformación de metodología de análisis de imagen en plugins.	Carga y manejo de imágenes en formato WSI.	Uso de algoritmos de ML HER2 (4B5), Dual ISH, PD-L1 (SP263) y CPNM (CE-IVD).
Plataformas	Ejecutable y web.	Software	Software	Software	Web
Objetivos	Herramienta de análisis de imágenes.	Permite análisis de imagen y la programación por bloques de la metodología para la reproducibilidad de experimentos.	Análisis de imagen y transformación de metodología en *plugins* para la reproducibilidad de experimentos.	Análisis y segmentación de Roi's en imágenes WSI.	Servicio para segmentación de imágenes histopatologicas WSI y detección automática de estado y semi-cuantitava de gen HER2.

Tabla 2.1: Tabla resumen de herramientas de visualización y análisis de imágenes científicas e histopatológicas.

CAPÍTULO 3

Microdraw

El Cvblab nació en 2015 como un grupo de investigación centrado en el análisis de imagen médica, principalmente análisis de imagen histológica aplicando métodos de *Deep Learning*. Para adquirir bases de datos con las que entrenar los modelos de DL, se comenzó a usar en 2016 una aplicación web de código abierto llamada Microdraw[19]. Esta aplicación permite visualizar *Batch* de imágenes WSI desde cualquier navegador mediante el uso de una URL que apunta a un recurso web en específico, además, cuenta con un conjunto de herramientas que permiten segmentar regiones a nivel de píxel.

El *Stack* del servicio web es LAMP (Figura 3.1): Linux [36] como sistema operativo del servidor que aloja la aplicación, Apache [34] para el servidor web que procesa las solicitudes HTTP, MySQL [35] para el motor de bases de datos y PHP lenguaje que hace de nexo para comunicarse con todas las tecnologías y ofrecer páginas web dinámicas. En las aplicaciones web se diferencia entre *Frontend* y *Backend*, el *Frontend* es la parte visual y todo lo que ayuda que esta se vea bien, esto se realiza con *HyperText Markup* o HTML [39] y con *Cascading Style Sheets* o CSS [40], el front es interpretado por los motores web: Edge, Chrome, Firefox, Safari, Opera, etc. Por otro lado, el *Backend* es el encargado de atender y procesar las funcionalidades que tiene el servicio, por ejemplo: traducir texto, listar elementos de una BBDD en una tabla o representarlos en una gráfica, realizar la predicción y etiquetado de una imagen con modelos de DL, etc.

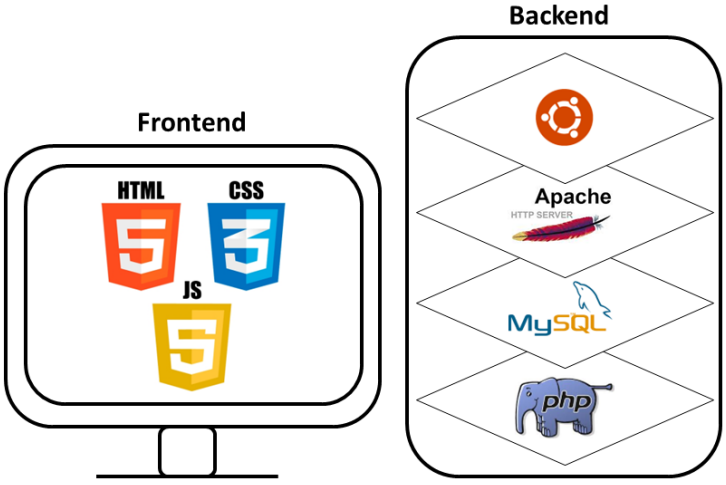


Figura 3.1: Stack del servicio web.

3.1 Frontend

La web ha evolucionado pasando de web estática, declarada en HTML, a sofisticados servicios capaces de sintetizar imágenes totalmente nuevas a partir de la descripción de la misma. Todos estos servicios se han creado y se han pensado siguiendo las leyes del protocolo web, donde es importante no correr elementos pesados en el lado del cliente, entiendo por el lado del cliente todo lo que es corrido e interpretado por el motor del navegador.

Los principales lenguajes interpretables por un navegador son HTML y CSS, el primero, es un lenguaje de etiquetas que permite definir la estructura de una página web, dividía principalmente en: *Header*, *Main Content*, *Navegation Bar*, *Sidebar* y *Footer*, estos objetos definen el esqueleto de la web, pero no son los únicos, también se definen un gran conjunto de etiquetas para declarar gran multitud de objetos estáticos.

Cuando se visita un recurso web se renderiza un archivo HTML y como resultado vemos contenido en el navegador, si a este no se le aplica ningún estilo, se renderizaría los objetos definidos en el HTML sin ningún tipo de color y o forma diferente al de por defecto, es por ello que de la mano de HTML va CSS, esta tecnología se encarga de definir características como color, tamaño o posición de objetos definidos en HTML, la forma que se tiene de insertar CSS en HTML es mediante la etiqueta <*stile*>. HTML permite usar hojas de CSS evitando insertar código directamente en el HTML, esto se hace referencia al script CSS con la etiqueta <*href*> y la ruta donde se aloja el archivo, dentro de la cabecera <*header*>, esto permite tener una organización adecuada de los recursos web, pudiendo tener distintas hojas de estilo en una carpeta llamada css y referencia a cada una de ellas como indica el siguiente código HTML:

```
<header>
   <link rel="stylesheet" href="css/microdraw.css" type="text/css"
<header>
```

En el caso de la aplicación, en la carpeta "*./css/*" se encuentra el archivo "*microdraw.css*" donde se aloja todo el estilo de los objetos descritos en el HTML.

La página web que aloja el servicio tiene dos archivos HTML. El primer documento define dos formularios, uno de inicio de sesión y otro de registro de usuario (Figura 3.2).

(a) Formulario de Acceso. **(b)** Formulario de Registro.

Figura 3.2: Página de inicio de la aplicación web.

Una vez el proceso de *login* se ha ejecutado con éxito, se accede a la segunda vista donde se puede visualizar y anotar las imágenes WSI.

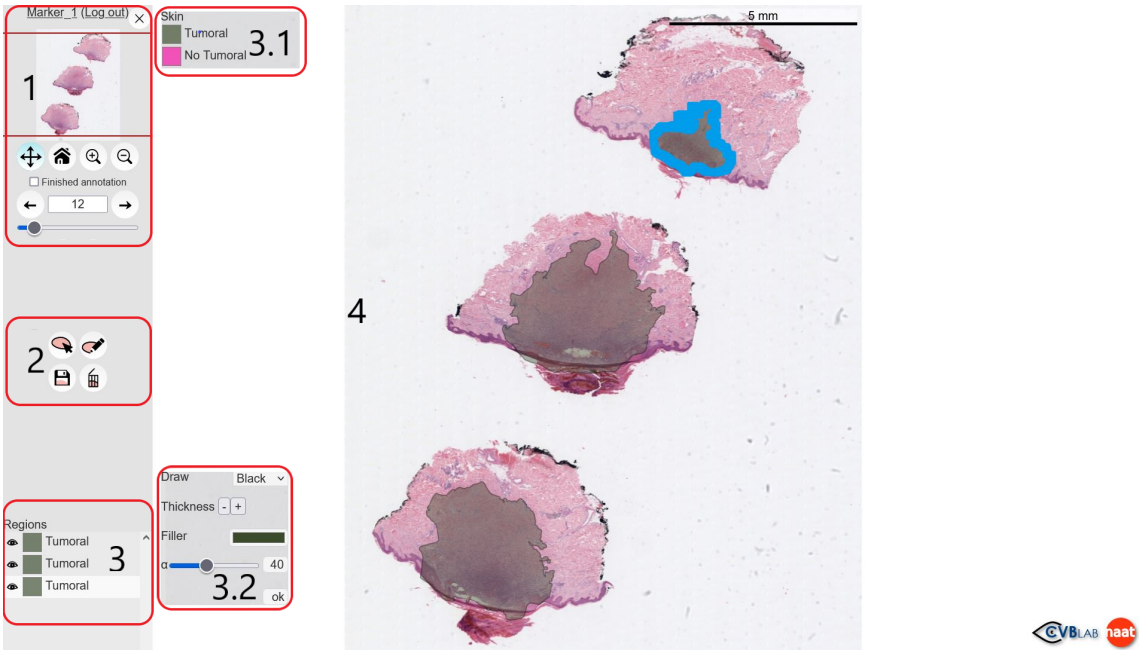


Figura 3.3: Página de anotación de Microdraw.

Como se puede ver en la Figura 3.3, la página de anotación se divide en cinco grandes partes:

- 1. Visualización y navegación:
  - En primer lugar, tenemos un recuadro de navegación que indica en que parte de la imagen se sitúa la vista principal (4), de esta forma el usuario siempre puede situar que parte de la WSI está visualizando en ese momento.
  - En segundo lugar, se encuentran un conjunto de herramientas de navegación que permiten: moverse por la zona de visualización de la imagen (4), volver al nivel de Zoom 0, hacer Zoom-in y hacer Zoom-out, respectivamente.
  - Por último, se encuentran herramientas para moverse entre las distintas imágenes que componen un *Batch*, las flechas permiten retroceder a la imagen anterior o continuar a la siguiente, adicionalmente se muestra el número de la imagen que se está visualizando, para moverse de una forma rápida se ha implementado un objeto *Slider* que permite desplazarse de forma dinámica por todo el *Batch*.

  Los objetos que se definen con iconos son objetos de tipo botón al cual se le asigna como *src* una imagen alojada dentro de los recursos web de la página, por ejemplo, el botón de navegación se define como:

```
<img class="button selected" id="Zoom" title="Navegator"
src="img/navigate.svg">
```

- 2. Herramientas de anotación:
  - Selección de una anotación, permite seleccionar cualquier región anotada haciendo clic encima de ella.
  - Dibujar región, una vez se activa posibilita la opción de anotar sobre la imagen visualizada en 4.
  - Guardar.
  - Eliminar anotación.
- 3. Lista de anotaciones, en esta zona se indexan las regiones dibujadas, cada entrada en la lista la componen tres elementos: un botón con un icono en forma de ojo que se carga desde los recursos, un cuadrado de color que define el color de la región dibujada, un texto con el nombre y la etiqueta. Cada nueva entrada en la lista lo hace mediante una división declarada con la etiqueta <*div*>(véase la Figura 3.4).

```
<div class= "cell">
  <br> Region <br>
    <div id="regionList" style="height: 119.533px;">
      <div class="region-tag deselected" id="1" style="padding:2px">
        <img class="eye" title="Region visible" id="eye_1" src="img/eyeOpened.svg">
          <div class="region-color" style="background-color:rgba( 58 , 76 , 44 ,0.67 )">
          </div>
        <span class="region-name">Tumoral</span>
      </div>
      <div class="region-tag deselected" id="2" style="padding:2px">
        <img class="eye" title="Region visible" id="eye_2" src="img/eyeOpened.svg">
          <div class="region-color" style="background-color:rgba( 58 , 76 , 44 ,0.67 )">
          </div>
        <span class="region-name">Tumoral</span>
      </div>
    </div>
</div>
```

Figura 3.4: Listado de regiones.

Interactuando con los elementos de la lista de regiones se despliegan dos ventanas:

- 3.1. Es un panel de etiquetado, sirve para elegir la etiqueta de la anotación seleccionada, se define como una lista de objetos en los que indexan con color y nombre las etiquetas asociadas al *Batch*.
- 3.2. Es una ventana para configurar aspectos visuales de la anotación, los objetos con los cuales interactúan, se definen como input de distinto tipo: *dropdown* (color de la línea de la región), botones (selector de *thinkness*), color (selector de color) y rango (slider de nivel de transparencia).

- 4. Visualizador de las WSI y anotaciones, permite navegar por la imagen, visualizando las distintas partes de la imagen sobre las que se hace Zoom, además permite

el manejo de las anotaciones (dibujar, mover y modificar) para lo cual se define un objeto *Canvas* sobre el que se anotan las regiones.

3.1.1. Javascript

Como se ha descrito con anterioridad, HTML y CSS definen el contenido estático y visual de la web, pero no permiten implementar funcionalidades complejas, ya que las primeras generaciones de webs únicamente mostraban información, pero con la introducción de la web 3.0 y la actual, la web 4.0, se prestan más servicios webs. Ante la necesidad de incorporar tareas más complejas en una red que no gozaba de velocidades como las actuales, se creó un lenguaje de programación que puede ser ejecutado en el lado del cliente, con esto se consigue ejecutar taras más complejas como incorporar gráficos dinámicos con los que el usuario pueda interactuar, con el uso de Javascript mejora notablemente la experiencia de usuario, ya que al ejecutar muchas de las funcionalidades desde el navegador no es necesario un envío constante de datos al servidor para que ejecute determinadas funciones, esto hace que el servicio web sea menos demandante de ancho de red, este hecho en la actualidad es muy importante, puesto que es uno de los factores claves para que una página web esté bien posicionada en motores de búsqueda como Google.

Javascript es ejecutable en el navegador, esto quiere decir que los motores de navegación comprenden y entienden su lenguaje al igual que lo hacen con HTML y CSS, es por esto que se puede usar Javascripst incrustándolo directamente en un archivo HTML dentro de las etiquetas "*<script>code </script>*", otra opción es la de referenciar un documento Javascript con el lugar donde está ese archivo alojado de la siguiente forma "*<script src="folder/file.js»</script>*". Este lenguaje de programación permite coger los id de los objetos definidos en HTML y asociarles eventos y o funciones, por ejemplo, cada vez que se hace clic sobre un botón de la barra lateral de la vista de anotación (Figura 3.3) salta un evento *clic* que realiza una función dependiendo del tipo de botón, por ejemplo, el botón *next* de las herramientas de visualización y navegación tiene asociado un evento clic que cuando salta realiza una función encargada de leer la siguiente imagen, esto ocurre con el resto de elementos que componen el menú vertical, todos ellos tienen un evento y una función asociada.

La aplicación objetivo de este TFM permite la visualización de imágenes gigapíxel en web y además permite realizar anotaciones, estas dos tareas sin Javascript serían difíciles de realizar, para implementar esas funciones se usa dos librerías:

- Paper.js [45] es una librería de código abierto que se ejecuta sobre un objeto Canvas de HTML y ofrece la capacidad de realizar sobre él: formas, curvas y objetos que son definidos como gráficos vectoriales. Crea los objetos visuales que son mostrados sobre la WSI a modo de anotaciones, define su posición, puntos que la definen, forma y color.

- Openseadragon [46], permite visualizar imágenes de alta resolución en la web, maneja los recursos de las WSI que permite al usuario hacer Zoom sobre una imagen sin que pierda calidad y fluidez en la aplicación.

3.2 Backend

Como se indica en la Figura 3.1 el *Backend* lo componen: Linux, Apache, Mysql y PHP.

El uso de Linux como SO es conveniente, ya que es el principal SO usado en servidores, esto se debe a que al ser un sistema operativo tipo Unix compuesto por software libre y de código abierto permite gran nivel de configuración, además tiene una comunidad amplia que publica multitud de paquetes para diferentes tareas, en este caso el servidor se ha levantado con la distribución de Linux, Ubuntu Server 20.04.

3.2.1. Servidor web

Para el despliegue de la aplicación se necesita un servidor web y una base de datos, para desplegar ambos servicios y gestionarlos fácilmente se usa el paquete de software libre Xampp [41], este paquete contiene Apache, Mysql, Filezilla (servidor FTP), Mercury (servidor de correo) y Tomcat (contenedor de *servlets*). Desde la interfaz de usuario permite el despliegue de todos los servicios, también se puede realizar el despliegue desde la consola de comandos. Al instalar el paquete Xampp, la hace en un directorio específico que contiene un conjunto de carpetas en las que hay una llamada "htdocs", en esta carpeta se debe de poner los recursos web que se van a hacer públicos y a los cuales se va a dar acceso.

Figura 3.5: Interfaz de usuario de la consola Xampp.

Desde la consola de comandos lanzada desde la carpeta de instalación de Xammp ejecutando "start" arranca apache y MySql. Por defecto, los puertos en los que se arranca son: 80 (8080 para *debugear*) para las conexiones http web, 443 para las conexiones https con el protocolo de seguridad *Transport Layer Security* o TLS (protocolo de criptografía que dota la comunicación http de integridad y privacidad, para lo cual se usa sistema de cifrado asimétrico con intercambio de clave pública).

Al iniciar el servicio web, el servidor escucha peticiones http y https por el puerto 80, si la conexión se realiza con éxito el servidor devuelve al navegador del cliente los recursos necesarios para renderizar la web, estos archivos deben de estar en la carpeta

"htdocs", para acceder al servicio que aquí se detalla se realiza una conexión mediante una URL compuesta por los siguientes campos:

- Protocolo de conexión http o https según el nivel de seguridad con la que se quiera dotar la trasmisión de datos: *http://* o *http://*.
- Nombre del dominio que hace referencia a la máquina donde se aloja el servicio o la IP: *https://ServerIP*
- Directorio donde se encuentra el archivo HTML que se renderiza en el navegador del cliente: *https://ServerIP/folder/index.html*

  Accediendo a esta ruta se visualizaría en el navegador la Figura 3.3 sin la imagen WSI, ya que se está accediendo a la web sin indicarle a qué *Batch* se quiere acceder, por tanto, no se visualizaría ninguna imagen, para realizar esto hay que añadir a la URL la ubicación del *Batch*.
- Para obtener los recursos asociados a la visualización de la imagen WSI, es necesario indexar la ruta donde se alojan dichos recursos para que se pueda realizar un GET sobre ellos y puedan ser representados en el navegador del cliente, la URL quedaría definida de la siguiente forma.

```
https://ServerIP/folder/index.html?source=images/batch_images/dzi_images.json
```

3.2.2. Base de datos

Entre los distintos servicios ofrecidos por Xampp, está un servidor de base datos MySql, en concreto se trata un servidor con MariaBD que aloja bases de datos rlacionales, el servidor de base datos es accesible desde el botón "*Admin*" de la intefaz de usuario de la consola (Figura 3.5) este acto abre un servicio web PHP alojado en el nuestro servidor llamado "phpMyAdmin" [42] que permite gestionar las bases de datos MySql alojadas en un servidor, pero esta no es la única forma, también se puede hacer estableciendo una conexión con el servidor por medio de la consola de comandos [43], además existen programas de instalación local que permiten gestionar varias conexiones a la vez, estos poseen una interfaz amigable con múltiples funcionalidades automatizada, para este TFM se ha elegido MySql Workbench [44]. Para realizar la conexión con cualquiera de los métodos expuestos se necesita: la IP del servidor de la base de datos, el puerto, que por defecto es el 3306, pero por conveniencia y seguridad es mejor modificarlo, un usuario y pass con acceso a la base de datos.

La base de datos de la aplicación está compuesta por dos tablas con los siguientes campos:

- Users, donde se guarda los datos de los usuarios registrados, además es la tabla que se consulta a la hora de hacer el login.
  - UserID con formato int(25) auto incremental.

- Username formato varchar(65).
- Password formato varchar(65).
- EmailAdress formato varchar(65).

- KeyValue, es la tabla donde se guarda la información relativa a las anotaciones.
  - UniqueID, formato int(11).
  - myTimestamp formato timestamp.
  - myOrigin formato text.
  - myKey formato text.
  - myValue formato longtext.
  - mySlice formato int(6).
  - mySliceName formato varchar(65).
  - mySource formato varchar(65).
  - myUser formato varchar(65).
  - finished formato tinyint(1).

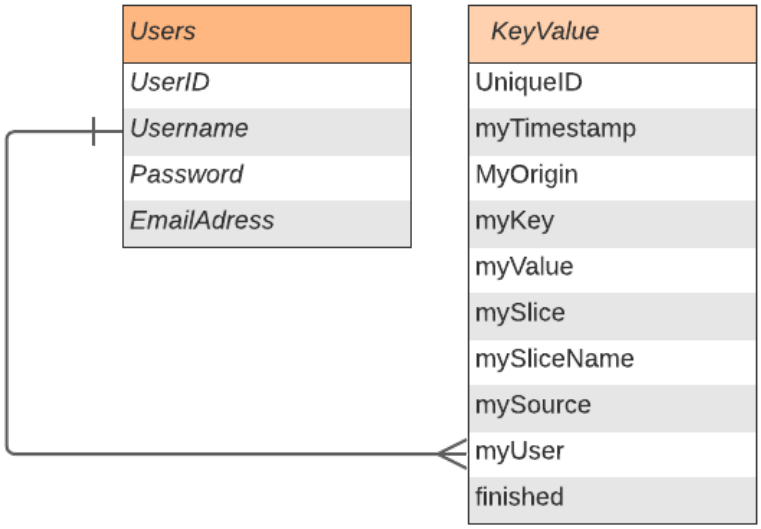


Figura 3.6: Tablas y campos de la base de datos.

Para guardar o consultar datos en la base de datos es necesario crear una conexión al servidor, y una vez establecida realizar la petición correspondiente, este proceso no se realiza en el *Frontend*, lo hace el *Backend*, tanto la conexión como las peticiones son ejecutadas en PHP, este lenguaje de programación es el más usado para la programación web, culpa de ello es la estabilidad que tiene y la gran comunidad que posee, gestores de contenidos como Wordpress lo usan, además de por estas características, por la cantidad de *plugins* codificados en este lenguaje.

PHP no es la única tecnología involucrada en la gestión de datos con la BBDD, también lo está Javascritp, ya que es la encargada de tomar los datosde los objetos HTML

y pasarlos al código de PHP para que este realice la SQL Query de guardado o consulta. Para realizar esta tarea Javascript cuenta con una tecnología llamada *"Asynchronous Javascript and XML"* o AJAX, esta permite ejecutar tareas de forma asíncrona, en otras palabras permite la ejecución y funcionamiento normal de la aplicación mientras se ejecuta una tarea en segundo plano, en este caso esa tarea es una *POST* o *GET* a la base de datos.

En los archivos de PHP se configura la conexión a la base de datos MySql con los datos de: IP o dominio, puerto, usuario (con privilegios de escritura y lectura) y contraseña, adicionalmente implementa los métodos para insertar o leer datos en las tablas "User" y "KeyValues", para enviar datos o cargarlos desde Javascript se debe definir el Ajax, con la acción el tipo de petición y los datos que necesita el método PHP para crear la consulta SQL, por ejemplo, en el caso de guardar datos, el botón de guardar(elemento 2, Figura 3.3) tiene definido un evento *clic* al cual se le asocia una función en Javascript que ejecuta un Ajax con los datos necesarios para insertarlos en la base de datos.

- El Ajax que se configura se define con un Type "*Post*" por que se va a enviar datos.
- data, define los datos que se van a pasar al script de PHP para que este se encargue de gestionar la tarea.
  - Action, determina a que método de los que se han definido en PHP se va a llamar.
  - "origin", "key", "value" y "finished" son lo datos que se deben de introducir en la tabla "KeyValue" (Figura 3.6) para generar una nueva entrada correspondiente al guardado de nuevas anotaciones.
  - Succes y Error definen funcionen que se ejecutan en caso de que el proceso sea exitoso o no.

```
$.ajax({
    url:dbroot,
    type:"POST",
    data:{
        "action":"save",
        "origin":JSON.stringify({
            appName:myOrigin.appName,
            slice: sl,
            source:myOrigin.source,
            user:myOrigin.user
            }),
        "key":key,
        "value":JSON.stringify(value),
        "finished":slice.finished
        },
success: function(data){}
error: function(jqXHR,textStatus){}
})
```

El campo "*Value*" del Ajax contiene todas las anotaciones de un slide en formato JSON, las anotaciones cuando se crean son objetos definidos con "paper.js" que tienen: un conjunto de segmentos que definen el contorno de la anotación sobre el Canvas, un color de relleno, un nombre, el folder donde se encuantra la WSI sobra la que se sitúa dicha anotación, un código *Hash* resumen de las coordenadas de las anotaciones para comprobar si ha cambiado la anotación, y una etiqueta (Figura 3.7).

Una vez los datos son enviados al script PHP, este se encarga de crear la conexión con la base de datos, crear la SQL Query "*INSERT INTO*", seleccionando como tabla

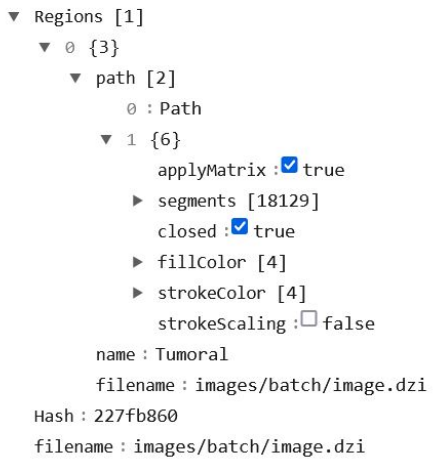


Figura 3.7: Anotaciones en formato JSON.

"*KeyVaue*" y campos todos los necesarios (Figura 3.6), pasando como "*VALUES*" todos los que son pasados desde el Front. Una vez hecha la consulta, PHP se encarga de recibir la respuesta de sí el proceso se ha ejecutado con éxito o no y envía el código de estado al *Frontend* para que los eventos Success o Error del Ajax realicen una tarea de alerta al usuario.

3.3 Carga y lectura de WSIs

La principal función de Microdraw es el de visualizar y anotar imágenes WSI, estas son imágenes gigapíxel, por lo que son archivos muy pesados, y la web penaliza mucho este tipo de archivos, es por ello que la imagen no se puede leer directamente en el navegador, para cargarla en la web es necear aplicar algún método con el que se consiga archivos ligeros que si puedan ser mostrados sin perder calidad.

El método que se usa para resolver el problema de cargar imágenes de alta densidad, es el de cortar dichas imágenes en trozos más pequeños que en conjunto representen la imagen original con la misma calidad, este proceso se puede realizar con una librería de procesado de imagen llamada Libvips [49], con ella se puede dividir la WSI en diferentes parches por niveles de Zoom, las divisiones que se realizan se guardan en carpetas diferentes según el nivel de Zoom.

Libvips es usa mediante comandos en la cmd, para realizar la tarea de división de una WSI ha de ejecutarse un comando que tiene como argumentos: el archivo a dividir, el folder de destino, el solape ,las dimensiones de recorte de la imagen y la calidad.

```
os.system("vips dzsave " +fileIn + " " +fileOut+'  --overlap 0
--tile-size 512  --suffix .jpg[Q=90]')
```

Como resultado de aplicar el código anterior se tiene un sistema de carpetas en la que cada una contiene las divisiones que en conjunto forman la imagen original, por ejemplo, si pasamos como argumento una WSI con los argumentos arriba descritos tenemos que se genera un sistema de carpetas del 0 al 9 en el que la carpeta 0 contiene la misma imagen de entrada con un taño de 512x512, la carpeta 1 contiene dos divisiones de la imagen original cada una con un tamaño de 512x512 y en conjunto representan la imagen origen, pero con algo más de Zoom, este proceso de recorte se aplica de forma incremental hasta la carpeta 9, esta última contiene todas las divisiones producto de la división de la imagen origen en 366 columnas y 155 filas y contiene las imágenes que se visualizarán con el nivel más alto de Zoom, el conjunto de todas ella representa la WSI con el nivel máximo de Zoom (Figura 3.8). Como resultado de aplicar esta estrategia tenemos imágenes individuales que pueden ser cargas en la web porque tienen un peso del orden de Kilobyte.

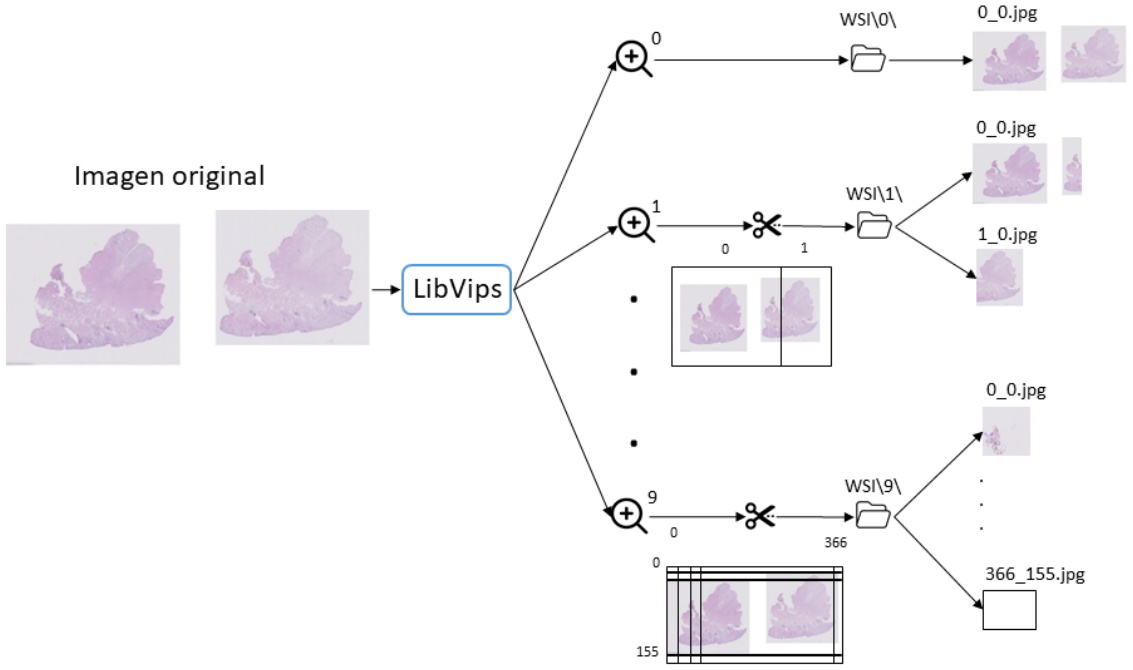


Figura 3.8: Sistema de archivos generados con Libvips.

Los *Batch*es de Microdraw están formados por más de una slide (WSI), por lo que es necesario conocer la posición de cada de las slides y además gestionar los niveles de Zoom recorriendo el sistema de archivos que genera la librería *libvips*, estas tareas son llevadas a cabo por la librería de Javascript Openseadragon [46], pero para ello necesita datos de cada una de las imágenes del *Batch* al que está accediendo el usuario. En primer lugar, se debe de conocer el lugar donde se aloja cada uno de los sistemas de carpetas que representan a cada slide, para conocer esta información el Front realiza una petición GET al servidor indicándole que recurso quiere, esto tal y como se ha explicado en la Subsección 3.2.1 se hace en el URL de acceso, ahí se indica al servidor que devuelva el archivo *"dzi_images.json"*, este archivo contiene la ruta a los archivos *"dzi"*, a continuación se muestra un ejemplo.

```
{
  "pixelsPerMeter": 4000000,
```

```
  "tileSources": [
    "images/batch_images/slide_1.dzi",
    "images/batch_images/slide_2.dzi",
    "images/batch_images/slide_3.dzi",
    ...
    ]
}
```

El archivo anterior devuelve la posición de los DZI e indirectamente la posición de las carpetas generadas por libvips para cada slide, ya que cada carpeta se llama igual que el DZI, por ejemplo, la carpeta que contiene todas las imágenes para representar en web la slide está en la ruta *"./images/batch_images/slide_1/"*, con esto, Openseadragon conoce la posición de los archivos y el formato (jpg, png,etc) de las divisiones, *overlap*, el tamaño de las subdivisiones y las dimensiones originales de la WSI. El motivo de querer conocer el *".dzi"* es que toda información descrita con anterioridad está declarada en este archivo con formato XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<Image xmlns="http://schemas.microsoft.com/deepzoom/2008"
    Format="jpg"
    Overlap="0"
    TileSize="375"
    >
    <Size
    Height="95456"
    Width="76368"
    />
</Image>
```

Teniendo todos los datos en el *Frontend*, Openseadragon sobre que recursos realizar peticiones GET, para cargar en el navegador la imagen correspondiente al nivel de Zoom y zona que el usuario quiera visualizar en la pantalla de visualización y anotación (Figura 3.3 zona 4).

CAPÍTULO 4

Extensión de Microdraw: Integración de Modelos de IA y Tareas de Crowdsourcing

Como se ha explicado en el capítulo Capítulo 3, Microdraw es una aplicación web con un *stack* LAMP, por lo que precisa de unos requisitos para ser desplegada, además, la herramienta debe garantizar una disponibilidad y experiencia de usuario óptima. El análisis de los esos factores, su mejora y la implantación de nuevas funcionalidades que garanticen la consecución de los objetivos expuestos en la Sección 1.2 será el propósito de este capítulo.

4.1 Mejora en la estrategia de alojamiento del servicio con Kubernetes

Uno de los principales problemas que puede tener una aplicación, es el de la disponibilidad, ya sea por falta de ancho de banda o por caída eventual del servidor que aloja el servicio, este es uno de los primeros aspectos a analizar.

El grupo de investigación CVBLab posee múltiples servidores de computación para el entrenamiento de sus modelos de DL, sin embargo, aloja la aplicación en un servidor NAS o "*Network Attached Storage*", este tipo de servidores son comúnmente usados para almacenar grandes volúmenes de datos (del orden de Terabytes) y hacerlos accesibles a través de la red, de tal forma que los usuarios lo usen como una unidad de red que desde el punto de vista del ordenador del cliente se vea como un disco duro más, aunque ese es su objetivo principal no es la única funcionalidad que tiene, al poseer como sistema operativo LINUX (no en todos los casos) es fácil desplegar otros servicios como un *stack* LAMP, a pesar de que no es lo más recomendable.

Tener alojado un servicio web en un servidor NAS es algo que puede funcionar si la aplicación se dimensiona para un uso de red pequeño, por tanto, unas funcionalidades básicas o para un número muy limitado de usuarios recurrentes, esto último era lo que

sucedía con Microdraw, en un principio esta aplicación fue pensada para que únicamente un patólogo anotara, por lo que no se dimensionó para que tuviera un tráfico muy alto, con el aumento de proyectos de colaboración, aumentó el nivel de usuarios de la aplicación lo que se traduce en más tráfico, al que hay que incluir el del uso propio de la NAS, ya que se trabaja de forma habitual con ella incluso como sistema para almacenar datos de entrenamiento de modelo. Todos estos motivos hacen que sea necesario migrar la aplicación a otro servidor.

Actualmente, la gran mayoría de servicios y aplicaciones que se desarrollan se despliegan en la nube, puesto que ofrece múltiples ventajas: escalabilidad, disponibilidad, flexibilidad, pago por recursos consumidas, evitando así grandes dispendios sobredimensionando servidores locales. El *cloud* ofrece todo lo que necesitamos para alojar nuestro servicio si esta no fuera una aplicación que trate con datos médicos sensibles, la Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales, establece una normativa que hace complicado el alojamiento de estos datos en la nube, ya que cada base de datos tiene un origen distinto al que habría que analizar particularmente y comprobar si permite o no el alojamiento en el *cloud*, a pesar de que los proveedores *cloud* como Amazon tienen certificados de *Compliance*[47] que garantizan el cumplimiento legislativo de ciertos países. Para evitar posibles complicaciones y puesto que el CVBLab no dispone de un experto en esta ley, pero si tiene firmados contratos específicos para el manejo y uso de las bases de datos, es más conveniente alojar el servicio en un servidor local.

Cuando se dimensiona un servidor local para alojar una aplicación, es mejor sobredimensionarlo, puesto que la escalabilidad de este siempre será limitada respecto al *cloud*. Se adquirió un servidor con las siguientes características:

- Procesador Intel Core i7 11700K 3.6Ghz.
- Placa base MSI Placa Base Z590-A PRO.
- 32 GB DDR4 de RAM.
- M.2 NVMe de 1 TB para el sistema operativo y datos a los que se necesite acceso rápido.
- Disco duro 8 Tb HDD.

A pesar de que el servidor fue adquirido para alojar Microdraw, no es el único servicio que se quiere alojar, ya que el grupo de investigación está envuelto en múltiples proyectos y alguno de ellos exige la creación de servicios web, por lo que se pretende que este nuevo servidor pueda albergarlos. Esto plantea un problema en cuanto a incompatibilidades entre librerías, este se puede resolver creando varias máquinas virtuales en el mismo servidor de tal forma que se aprovisione parte de los recursos hardware compartidos, otra forma es el uso de Docker, la principal diferencia entre uno y otro es que las máquinas virtuales están pensadas para alojar distintos sistemas operativos compartiendo recursos hardware y Docker es tecnología de contenedores que permite aislar aplicaciones que

son totalmente independientes una de las otras pudiendo estar en el mismo sistema operativo, esto es lo que se busca con este servidor, albergar distintas aplicaciones bajo un sistema operativo Linux, por ende se instala Docker en el servidor.

Dockerizar servicios en un mismo servidor es una práctica habitual y fácil de gestionar cuando los servicios alojados no son muchos o dependen de otros, cuando se comienza a configurar un servidor de microservicios, es decir, un servidor que va a alojar muchos servicios que dan soporte a otros servicios, por ejemplo bases de datos es necesario tener un gestor de Dockers que automatice su despliegue, monitorice su desempeño, maneje su acceso y controle carga de trabajos, todas estas funcionalidades son implementables con la plataforma de código libre Kubernetes.

Kubernetes [48] es una herramienta creada por Google y liberada en 2014 que facilita la automatización y la configuración declarativa, permitiendo la orquestación de Dockers, con kubernetes se puede automatizar *deployments* para ello usa archivos con lenguaje declarativo llamados YAML, donde se define las características que debe de tener el Docker que se va a levantar para alojar un servicio: librerías, versiones, configuración de puertos, espacio, etc.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

(a) Definición de un Pod.

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 4
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

(b) Definición del *deployment* de un Pod.

Figura 4.1: Declaración y despliegue de un proxy con YAML.

La Figura 4.1 muestra cómo se define la estructura de un Docker o POD y la estrategia de despliegue de este, con estos dos archivos kubernetes es capaz de levantar un servicio. Para desplegar Microdraw se necesita desplegar dos pods uno con el servidor web y otro con la base de datos MySQL, al definir el YAML del *Pods* y del *deployment* Kubernetes despliega automáticamente el servicio cuando se arranca el servidor, esto junto a una configuración de la BIOS del servidor mejoran la disponibilidad de la aplicación, ya que cuando se produce algún corte de luz el servidor se inicia solo y automáticamen-

te kubernetes se encarga del despliegue del *stack* LAMP dividido en dos microservicios: base de datos y servidor web. Como se ha indicado con anterioridad, el servidor va a alojar distintos servicios alojados en distintos Pods por lo que es necesario un proxy para redirigir las conexiones al servicio apropiado. NGINX Ingress Controller es un servidor proxy que permite exponer los microservicios de un clúster a internet, NGINX escucha el puerto 80 y 443 de la máquina donde está desplegado el *cluster* de kubernetes y en función del dominio al que se hace la petición redirige la conexión a un pod o a otro, con esta estrategia se puede desplegar muchos servicios bajo una misma IP y puerto.

Para migrar la base de datos a un *cluster* de kubernetes es importante tener en cuanta que el espacio de un Pod que aloja una base de datos no puede borrarse si se mata el Pod es decir, en kubernetes cuando se declara un Pod y se levanta, si no se indica lo contrario, se le asigna un espacio de almacenamiento que en el momento de que el Pod se caiga o sea matado por cualquier motivo ,todos los datos que se hayan almacenado en el volumen declarado se borran liberando dicho espacio, por tanto, el volumen, por esto es importante saber que un servidor de base de datos es un servicio con estado o *Statefulset* es decir es una aplicación que necesita un volumen permanente para almacenar los datos sin perdidas ni discrepancias, todo lo contrario que ocurro con los servicios de servidores web como Apache que son *stateless* y se pueden desplegar y matar sin ningún tipo de problema, en consecuencia, a la hora de definir el Pod de la base de datos es importante configurar los volúmenes como almacenamiento persistente, de esta forma cuando muera el Pod desplegado con la base de datos no se perderán los datos y seguirán estando disponibles para el nuevo Pod que se despliegue.

Otro punto de análisis es la gestión de espacio en el servidor, ya que el conjunto de archivos generados para poder visualizar imágenes gigapíxel en la web de forma individual son ligeros, pero de forma conjunta ocupan mucho espacio, estas carpetas al ser migrada la aplicación es necesario migrarlos con ella. Microdraw se alojaba en un servidor NAS y esto significaba tener la ventaja de poder gestionar la carpeta de los batches de la aplicación de forma remota montándola como una unidad de red, al cambiar a un nuevo servidor se debe garantizar el acceso de la misma forma, para que de esta forma los investigadores encargados de generar y gestionar la subida de un *batch* o añadir slides a un *batch* no tengan que cambiar en exceso su forma de trabajo.

El servidor adquirido tiene un disco duro HDD de 8 TB, es ahí donde se traslada todos los batches que tiene Microdraw, ya que esta carpeta ocupa 1,5 TB y dicho disco duro tiene espacio suficiente incluso para poder añadir nuevos batches sin ningún problema. El siguiente paso es garantizar el acceso red a esa carpeta, para ello se implementa SAMBA, este protocolo de implementación libre permite compartir archivos entre máquinas con distintos sistemas operativos, de tal forma que admite compartir la carpeta con los batches de Microdraw con máquinas Windows.

La instalación de samba se puede hacer usando el gestor de paquetes "Apt" de Linux, una vez instalado se debe configurar el archivo *smb.conf* para definir que rutas del servidor van a ser accesible y que usuarios con qué permisos pueden acceder a dichos

recursos, la forma de configurarlo es añadir la siguiente plantilla con los datos correspondientes por cada unidad que se quiera montar.

```
[Nombre de unidad en red]
   comment = R/W Nombre de unidad en red
   path = ruta que se va a exponer
   valid users = usuarios que van a tener acceso a esa unidad
   public = no
   writable = yes
```

La configuración correspondiente para este caso será la de crear una unidad de red por *batch*, es decir una unidad de red por ruta *images/batches/bacth_x*, ya que cada *batch* está gestionado por una persona diferente, por lo que montar una única unidad con acceso a todos los batches no es conveniente.

4.2 Funcionalidades para crowdsorcing

Uno de los objetivos del presente TFM es el de integrar funcionalidades de Crowdsourcing o colaboración colectiva, en el campo de la patología digital, la necesidad de aplicar técnicas de colaboración colectiva provienen del escaso número de patólogos expertos en un tipo de cáncer en concreto. En general los patólogos son capaces de diagnosticar distintos tipos de cáncer, pero se especializan en un tipo en concreto, esto unido a la falta de patólogos a medio plazo hace que sea necesario un sistema en el que los patólogos pueden diagnosticar de tal forma que un tercero experto pueda evaluar esas anotaciones y comprobar si el diagnóstico emitido es el correcto, por esta razón en Microdraw se va a integrar estas funcionalidades que van a ser probadas por un conjunto diez residentes en patología, estos realizarán la tarea de expertos supervisando las anotaciones generadas por un modelo de Deep Learning que de forma automática segmentará y etiquetará una slide, las predicciones serán guardadas en la base de datos sin la etiqueta global de tal forma que los residentes visualicen únicamente las regiones anotadas y digan si son o no tumorales a la vez que anoten otras zonas que no hayan sido segmentadas por el modelo y de esta forma con la base de datos creada a partir de las rea notaciones de los residentes se reentrene el modelo y se mejore, lo mismo que ocurriría con patólogos expertos y no expertos, que al ser corregidas sus anotaciones con comentarios, estos adquirirían más experiencia en ese tipo de cáncer obteniendo un conocimiento más amplio que les ayudaría a realizar mejores diagnósticos en múltiples tipos de cáncer haciendo que la carga de diagnósticos se pueda repartir de una forma uniforme. Como primera aproximación de esto se definen dos funcionalidades que deben ser garantizadas por la aplicación:

- En primer lugar, ha de habilitarse un conjunto de herramientas dentro del menú que permitan escribir observaciones y etiquetar de forma global una slide.
- En segundo lugar, se debe crear un sistema para etiquetar muestras que no son claras y ser guardadas con el fin de que estas puedan ser evaluadas por un tercero.

Para la primera tarea se debe de crear dos objetos HTML en el menú lateral de la pantalla de anotación y visualización (Figura 3.3), en primer lugar ha de crearse una caja de texto donde se pueda escribir el tipo de cáncer con el que el patólogo etiqueta a nivel global una slide, y en segundo lugar se debe definir una caja de texto que permita escribir textos más largos con observaciones que el patólogo vea conveniente resaltar.

Desde el punto de vista de HTML y CSS es bastante sencillo definir los dos objetos que necesitamos, en primer lugar para etiquetar la imagen se usa un input de texto al cual se le aplica unas dimensiones menores que la del menú, y para el segundo se define un objeto ”textarea” que permita a los usuarios escribir texto largos.

```
<input id="annotation-label" type="text" name="label" style="width:
    173px; height: 22px;">
<textarea id="annotation-observation" rows="10" cols="22"
    style="width: 173px; height: 82px; resize: none;"></textarea>
```

Una vez tenemos definidos los objetos y antes de asociarle un evento y una función hay que crear una estrategia para guardar estos datos.

Para guardar y leer los datos hay que crear una tabla nueva en la base de datos, los datos que se deben guardar en ella es el contenido de *”annotation-label”* y de *”annotation-observation”* y además se ha de crear un campo que sea clave foránea para relacionar estos datos con la tabla ”users” y otros campos que permitan aplicar filtros y ser relacionada con la tabla ”keyValue”, por tanto, la estructura de la tabla sería la siguiente:

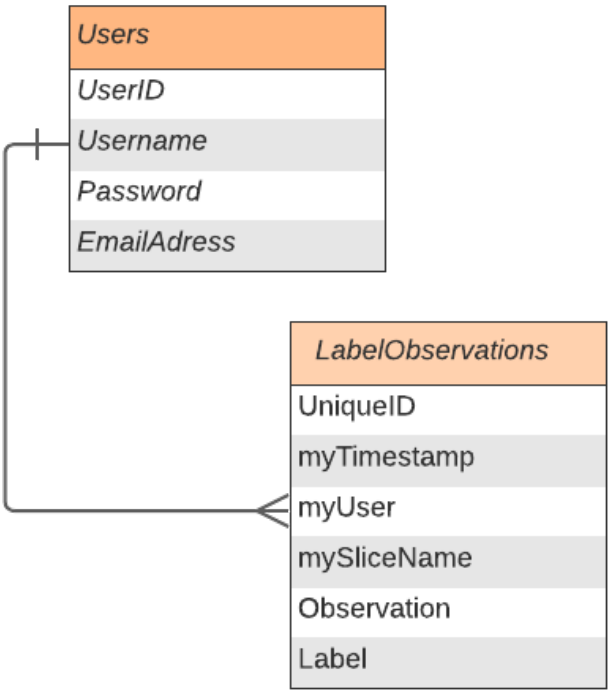


Figura 4.2: Inserción de base tabla ”LabelObservation” en la base de datos.

Donde los campos "UniqueID" *int(11)* definido como *Primary Key*, "MyUser", "mySliceName" y "Label" son de tipo *varchar(255)*, y "Observation" es *varchar(4000)* para que pueda albergar textos largos. Una vez definida las base de datos hay que definir el evento que salte y realice la función de guardado en la base de datos.

En primer lugar, hay que definir en PHP qué función va a realizar la inserción de un nuevo registro en la base datos, y la función que se encargue de leer los datos cuando se tenga que inicializar la información en los objetos, para ello se definen las funciones que deben ser llamada desde el Javascript y dentro de ellas se declara la conexión y el tipo de SQL según la necesidad una SQL de inserción para guardar y una de Selección para leer. La función de inserción realiza una SQL de "Select" filtrando por usuario, slide para asegurarse de que no hay una nueva función, en función de si se devuelve o no se realizaría una Query de inserción o de modificación.

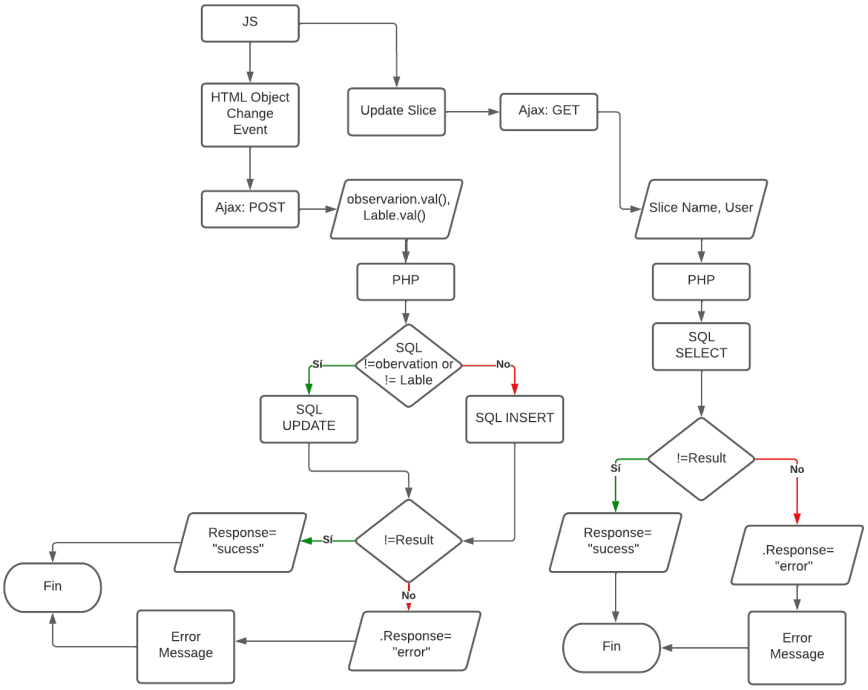


Figura 4.3: Diagrama de flujo para el guardad de etiquetas y observaciones.

Para enviar los datos desde el *Frontend* se realiza mediante Ajax por lo que se ha de definir el evento con el que salte, en este caso se usará el evento "*Change*", este evento hace que se ejecute una rutina cuando el objeto HTML al que se asocia, es decir, en el caso de los campos de escritura, cuando se haya acabado de escribir sobre ello el evento saltará y ejecutará una función, que en este caso será el Ajax que comunica con el PHP que realiza la función de inserción o carga de datos, tal y como se indica en el diagrama de flujo de la Figura 4.3.

El siguiente paso es cumplir la tarea de crear un sistema para etiquetar muestras que no son claras y ser guardadas, este sistema está pensado para que un patólogo que está anotando y realizando un diagnóstico pueda etiquetar la imagen a nivel global de tres formas diferentes, esto está pensado para usar la herramienta en un proyecto donde se

pretende comprobar las segmentaciones de biopsias de piel generadas automáticamente por un modelo, con el fin de evaluar dicho modelo, la idea es que un usuario tenga acceso a un *batch* que ha sido anotado por un modelo de IA, e indique según las anotaciones y los tejidos segmentados que tipo de neoplasia tiene esa biopsia.

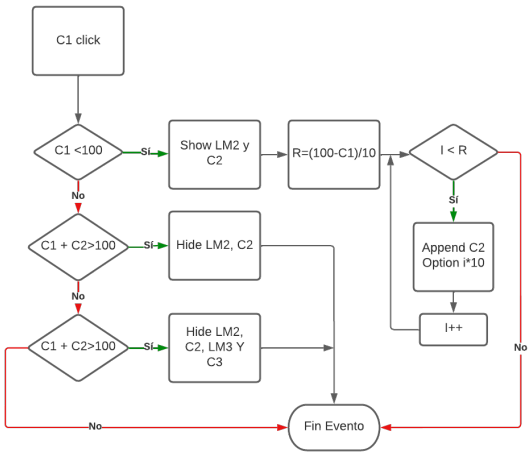
La cantidad de neoplasias que con la que puede ser etiquetada la imagen son siete: Leiomyoma, Leiomyosarcoma, Dermatofibroma, Dermatofibrosarcoma, Atypical Fibroxanthoma, Squamous Cell Carcinoma y Spindle Cell Melanoma, el usuario debe poder elegir entre una de ellas tres veces poniendo en cada una un nivel de confianza. Para realizar esto se define en el menú de la pantalla de visualización de la aplicación tres objetos "*Select*" para listar las neoplasias y tres "*Select*" para listar el nivel de confianza, esto últimos de listar un nivel de porcentaje de 10 % a 100 %.

El funcionamiento de los botones para listar las neoplasias debe ser el siguiente: en primer lugar al acceder, solo se debe de mostrar dos "*Select*" alineados horizontalmente, el primero de ellos permite elegir entre los siete tipos de neoplasia definidos con anterioridad, el segundo lista porcentajes de nivel de confianza (Nc1) que el usuario debe elegir, en caso de que el usuario elija uno menor de 100 % se indexa debajo dos nuevos "*Select*" uno para elegir otro tipo de neoplasia distinta a la que se ha elegido en primer lugar y el otro para listar el porcentaje de confianza (Nc2) dejando elegir entre de 10 % y 100-Nc1 %, en caso de que la suma de Nc1+Nc2 sea menor que 100 se repite el proceso, la idea es que el porcentaje de confianza de las etiquetas sume 100 siendo tres en número máximo de neoplasia.

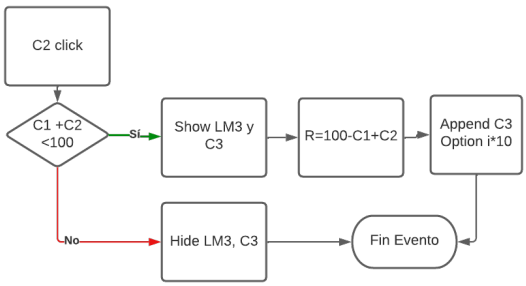
En primer lugar, se ha de definir de forma dinámica los objetos para que aparezcan según si se cumple las condiciones anteriormente expuestas, por tanto, en el HTML se define los dos primeras parejas de "*Select*" definiendo como "*values*" en uno, las siete posibles neoplasias y en el segundo los porcentajes, las siguientes parejas de "*Select*" definen como "*hide*" para que al iniciar la página web no se muestren, los "*values*" de las desplegables con las neoplasias, pero se dejan los de los porcentajes sin inicializar, el diagrama de flujo de la Figura 4.4, explica el comportamiento y codificación en Javascript.

Una vez definida la forma en la que se muestra e indexan lo valores, hay que diseñar la estrategia de guardado y lectura de estos datos, para ello hay que desplegar una nueva tabla donde guardar dichos datos, para ello se crea una tabla que llamada *conviction*, donde se va a guardar las anotaciones y su nivel de confianza en formato texto con las anotaciones estructuradas como un JSON, lo ideal es que se guardara directamente como formato JSON, pero la versión de la base de datos no permite campos definidos con ese tipo y migrar la base de datos daba errores de incompatibilidad con el código. La tabla se define con los siguientes campos y tipo de datos.

- UniqueID formato *Int(11)*.
- MyTimeStamp formato *timestamp*.
- My user formato *text* y clave foránea.
- MySliceName formato *text*.



(a) Diagrama de Flujo etiqueta y confianza 1.



(b) Diagrama de Flujo etiqueta y confianza 2.

Figura 4.4: Diagrama de flujo del evento asociado a los botones de confianza, LMx es el menú desplegable con las siete posibles etiquetas, Cx son los botones con el porcentaje de confianza Pod.

- Conviction formato *text*.

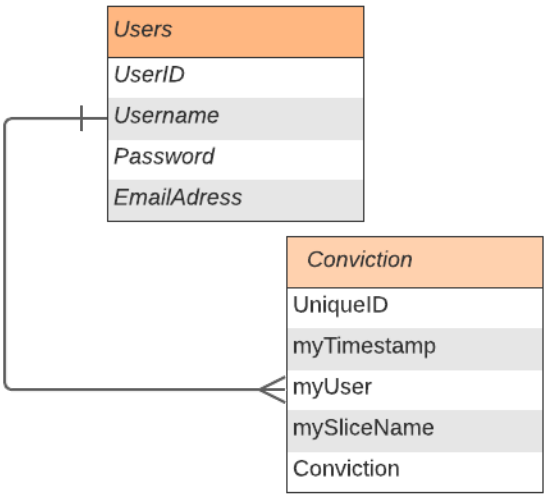


Figura 4.5: Diagrama de flujo para el guardar y leer las etiquetas a nivel global y el valor de confidencia.

Definida la estructura de la base de datos y el tipo de datos a guardar con el formato se diseña la forma de conectar los datos del *Frontend* con el *Backend* para que se envíen a la base de datos para que guarden y lean de forma correcta, como todos los procesos de comunicación de forma asíncrona entre Javascript y PHP se usa un Ajax que se ejecuta cuando salta un evento, en este caso cuando se hace clic en guardar se guardan los dato y para la lectura se define que se realice el proceso de GET cuando inicializa la web o cuando se cambia de slide, este proceso se diseñó y codificó siguiendo el diagrama de flujo de la Figura 4.6.

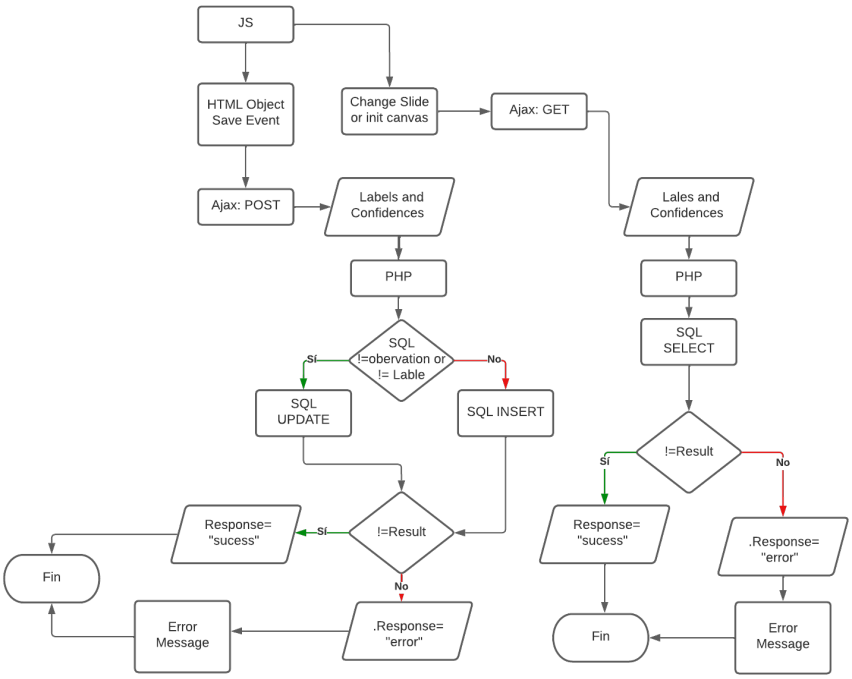


Figura 4.6: Diagrama de flujo para el guardar y leer las etiquetas a nivel global y el valor de confidencia.

4.3 Integración de Modelos de IA

Los modelos de *Deep Learning* a grandes rasgos funcionan como un cerebro, los humanos aprendemos a identificar objetos y formas por la experiencia, por ejemplo, desde pequeño nos acostumbramos a ver árboles, la primera vez que vemos uno lo etiquetamos como árbol y aprendemos que un árbol debe de tener las características del que hemos visto para etiquetarlo igual, el segundo árbol que vemos puede tener diferentes características, pero nos enseñan que ese tipo de forma aunque no es exactamente igual que el primero también debe etiquetarse como un árbol, entonces nuestro cerebro extrae las características comunes entre el primer y el segundo árbol, de esta forma cuando venga un tercero no va a pensar que para etiquetar lo que ve como árbol tiene que tener la misma forma que uno de los anteriores, se fijará que tenga ramas, un tronco, y demás características comunes entre los objetos de esta clase.

Para crear potentes modelos de *Deep Learning* sucede como el ejemplo anterior, es importante contar con amplias y heterogéneas bases de datos, para que de este modo los modelos visualicen clases de distintos tipos para definir que características comunes comparten los objetos de una clase, con el fin de que el modelo tenga un buen desempeño, de lo contrario los modelos pueden caer en lo que se denomina "*Overfitting*", es decir, se ajustan mucho al *dataset* con el qué se ha entrenado, y no evalúan bien nuevos casos, es decir solo etiquetan árboles con una misma forma.

Por tanto, los modelos de aprendizaje profundo se pueden ver como conexiones entre neuronas de un cerebro que realizan la función de identificación, este paradigma definen

la unidad básica de los modelos de DL, las neuronas, el conjunto de estas neuronas generas redes neuronales, caracterizadas por[51]:

- Tener "neuronasçomo unidad básica encargada de procesar información y compartirla con otras neuronas.
- Con entrenamiento supervisado (dataset con clases etiquetadas) pueden aprender y ser capaces de reconocer características comunes en datos, texto, imágenes, etc.
- Con técnicas de "*Active Learning*" (reajuste del modelo con nuevos datos) pueden mejorar el desempeño que tienen.

Las neuronas se agrupan para formar modelos de DL, una de las estructuras básicas son las redes neuronales y se componen de las siguientes partes (Figura 4.7):

- Capa de entrada de datos.
- *Hidden layers* o capas ocultas, son las capas que extraen información de los datos definiendo que características componen cada clase.
- La capa de salida clasifica el dato de entrada con una etiqueta teniendo en cuanta las características extraídas en las capas ocultas.

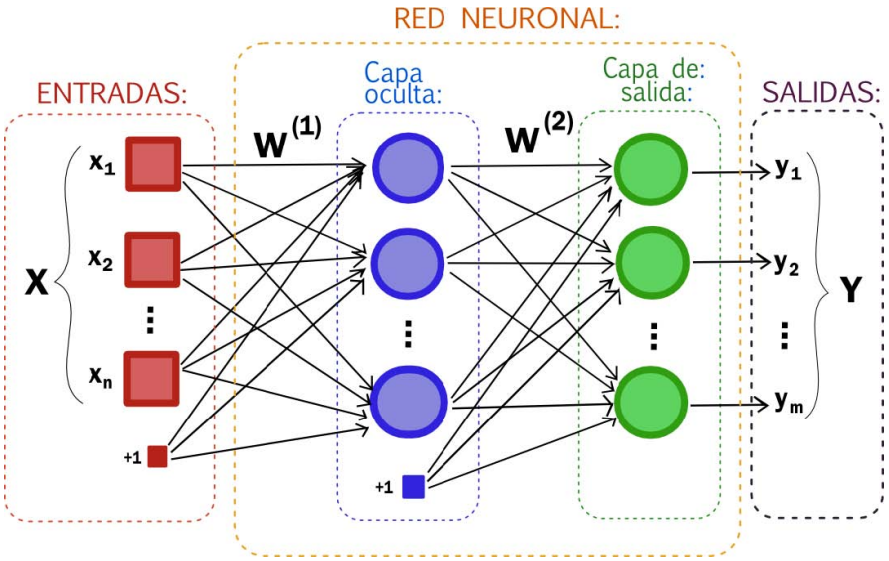


Figura 4.7: Estructura de una red Neuronal [50]

Este tipo de redes tiene múltiples ventajas como:

- La capacidad de abstracción, lo que facilita la tarea de extraer las características o *features* más importantes de los datos.
- Capacidad de aprendizaje adaptativo, ya que aprende a partir de las experiencias.
- Tolerancia a fallos: en caso de daños, pueden recuperarse ciertas características.

- Se pueden embeber en sistema electrónico como chips, controladores...
- Gracias al avance en el campo de las "*Graphic Procesing Unit*" o GPU Los cálculos necesarios para su entrenamiento se pueden realizar en paralelo, agilizando el proceso.

El funcionamiento a más bajo nivel de las redes neuronales, es algo que no es objeto de análisis para el propósito de este TFM, pero se puede decir a grandes rasgos que el funcionamiento de estas se basa en que cada una de las neuronas que compone la red realice una función de extracción de características de una dato mediante el uso de fórmulas matemáticas, con esto se consigue detectar por ejemplo, bordes, texturas, siluetas propias de una clase, estas características se van filtrando por la red hasta alcanzar las más genuinas para cada clase, con lo que la red construye un criterio de etiquetado.

El grupo CVBLab es puntero en el uso de técnicas de *Deep Learning* aplicando modelos sofisticados de DL basados en las estructuras expuestas con anterioridad, en el caso de este grupo de investigación, posee mucha experiencia en la creación de modelos específicos para la detección automática de regiones tumorales en imágenes patológicas. Microdraw se implantó para que los patólogos pudieran anotar y generar bases de datos con las que entrenar los modelos, ahora se quiere evaluar su desempeño a la hora de realizar tareas de autosegmentación, pero esto no se puede realizar de una forma sencilla por varias razones:

- Las pocas habilidades en tecnologías por parte de los patólogos hace complicado que estos sepan cómo gestiona el despliegue y mantenimiento de la aplicación, por esto es que se usa una aplicación web fácilmente accesible a través de un navegador.
- El uso de modelo de DL implica el aprovisionamiento de un servidor con recursos de computación para correr las predicciones.
- El funcionamiento de los modelos es complicado para que los patólogos puedan usarlos por si solos.

Por los motivos anteriormente expuestos, se propone la extensión de funcionalidades para poder hacer uso de los modelos de DL desarrollados en la aplicación web, para ello se quiere realizar una primera aproximación en la que se diseñe la metodología para representar predicciones automáticas en el servicio ya desplegado, permitiendo evaluar de una forma real el desempeño de un modelo, esta tarea se ha definido como un objetivo en la Subsección 1.2.2, donde se indica la labor de integrar modelos de Deep Learning en la Microdraw.

Hasta ahora las salidas de los modelos (las máscaras anotadas) se usaban para ser analizadas y discutidas en la producción científica del grupo donde el buen desempeño de los modelos, ahora se plantea la implementación de estos modelos para agilizar el proceso de diagnóstico, es decir darle un uso real más allá del académico, esto por asuntos de regulación y certificación no es algo que se puede usar en clínica hasta que no pasa una serie de procesos que den como resultado un marcado CE que permita su uso clínico.

El proceso de conseguir un marcado CE es complejo y largo, por lo que se pretende realizar un primer enfoque donde los modelos sean evaluados en entornos controlados, esto quiere decir que los modelos no van a ser usado para el diagnóstico real, la idea es a través de metodología Crowdsourcing, evaluar modelos que han sido testados con bases de datos anotadas por un mismo patólogo, es decir la base de datos se divide en tres partes, una para entrenamiento otra para validación y una tercera para test, con la parte de test se puede comprobar la *performance* del modelo y eso es lo que se discute en los *papers*, pero esta metodología tiene el problema, en las clases más complicadas de analizar el más mínimo detalle no percibido por un patólogo puede cambiar el diagnóstico, a lo que hay que añadir factores interpretables que pueden afectar, es por esto que la experiencia del patólogo es crucial para un buen diagnóstico, esto no siempre se puede asegurar por lo que se propone que estos modelos entrenados con anotaciones de un único patólogo sean evaluados por un conjunto de patólogos para de esta forma comprar de una forma más real el desempeño del modelo.

Como se ha indicado con anterioridad Microdraw es usada para generar bases de datos, con datos de regiones anotadas en WSI y etiquetadas con un tipo de patologías, estos datos se guardan en la tabla "KeyValues" (Figura 3.6) (Figura 4.8.A) y son leídos con un software propietario del grupo que genera que realiza consultas a la base de datos y genera máscaras con las regiones anotadas en las imágenes de un *batch* (Figura 4.8.B), además las máscaras tienen en cuenta las etiquetas de cada región, por lo que tenemos máscaras de las WSI que se han subido a la plataforma como se indica en la Figura 3.8 y la imagen completa (WSI), con estas dos entradas se entrena los modelos (Figura 4.8.C), como resultado del entrenamiento se obtiene modelos de DL que a partir de una WSI es capaz de detectar estructuras de interés y segmentarlas devolviendo una imagen con la máscara de las anotaciones(Figura 4.8.D).

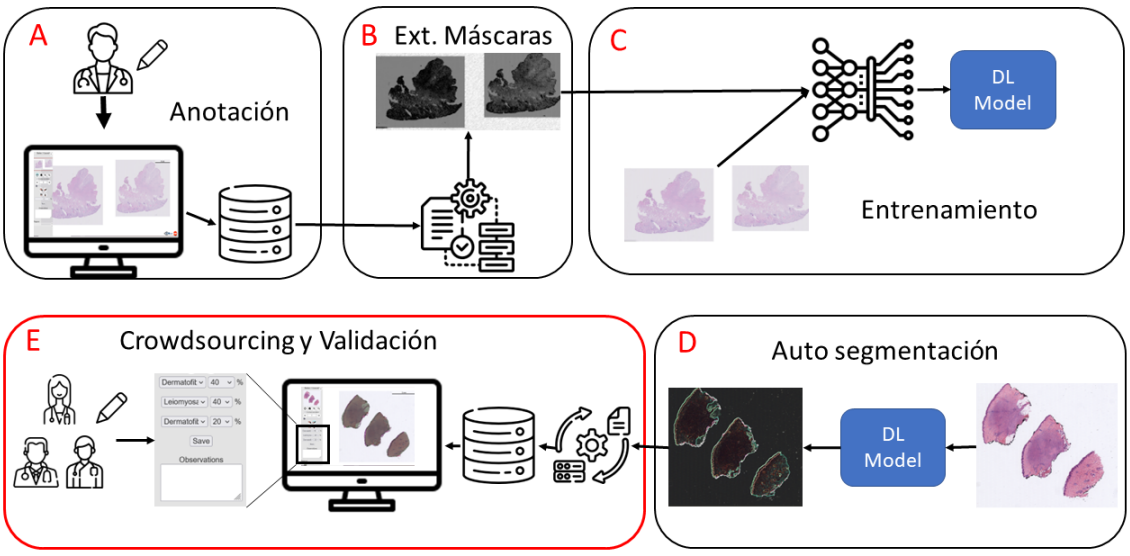


Figura 4.8: *Framework* de la integración de modelos.

La primera red de DL que se va a implementar dentro de la aplicación es un modelo que predice siete tipos de neoplasias en piel, el modelo fue entrenado con las bases de datos generadas a partir de anotaciones de un patólogo experto. La estructura, tipo de red, entrenamiento y desempeño del modelo son datos que será publicado en un *paper* que se está escribiendo y, por lo tanto, no se pueden detallar aquí de momento. Una vez se tiene la red, entrenada, validad y testeada obteniendo uno bueno niveles de *Acurracy* en test, se usa nuevas WSI de piel para que la red las etiquete de forma automática (Figura 4.8.D), el resultado que se obtiene son máscaras con regiones de tejido tumoral anotadas a nivel de píxel y la clasificación de toda las imágenes con una de las siete etiquetas de neoplasia.

Las máscaras no se pueden subir directamente como se ven en la imagen (Figura 4.8.D), ya que la idea es que se incluyan dentro de Microdraw como una WSI con las regiones anotada, visualizándose como se ve en la Figura 3.3, por tanto, se realiza el paso inverso a la extracción de las máscaras mediante una ETL (*Extract, Load, Transform*) que extrae la información de posición de las anotaciones en la WSI, las transforma a coordenadas y las carga en un archivo de salida. El proceso da como resultado un conjunto de archivos de extensión "txt" que llevan por nombre el de las WSI y la etiqueta global, cada archivo alberga los datos de las regiones anotadas en formato JSON, tal como se indica en Figura 3.7, para poder subir los datos y que se visualicen hay que crear registros nuevos en la tabla "KeyValue" de la base de datos, esto implica la introducción de valores que son propios del *batch*: "MySlice" (la posición que ocupa la slide en el *batch*) o "MySource" (la ruta del *batch* donde está alojada la slice), por lo que no puedes ser introducidos hasta no tener construido el dicho *batch*.

Para crear el *batch* se aplica procedimiento explicado Sección 3.3 representado en Figura 3.8, con esto conseguimos recortar las imágenes con el fin de conseguir los sistemas de carpetas necesarios para poder representar en la web los distintos niveles de la WSI sin perder calidad. Una vez subidas las WSI y generados los batches, el o la usurario/a puede acceder a la aplicación, pero aún no podría ver la segmentación realizada de forma automática por el modelo, para ello hace falta implementar un método que realice esto de forma automática (Figura 4.8E).

Para relacionar las imágenes del *batch* con las anotaciones correspondientes, se crea un script (Figura 4.9) que de forma automática lea un documento JSON donde se lista la ruta a cada una de las slides que compone el *batch*, estas rutas son tomadas y cortadas tomando la última parte de la ruta que corresponde al nombre de la slide, con esto se busca un documento "txt" con el mismo nombre en la carpeta donde se alojan todas las anotaciones automáticas extraídas con la ETL, cuando se encuentra este documento, se lee y se cargan los datos con formato un formato JSON estos datos son insertados en la base de datos junto con la información respectiva a la posición de la slide en el *batch*, la ruta al *batch* y demás campos especificados en una SQL Query que contiene todos los datos necesarios para crear un nuevo registro en la tabla "KeyValue" (Figura 3.6).

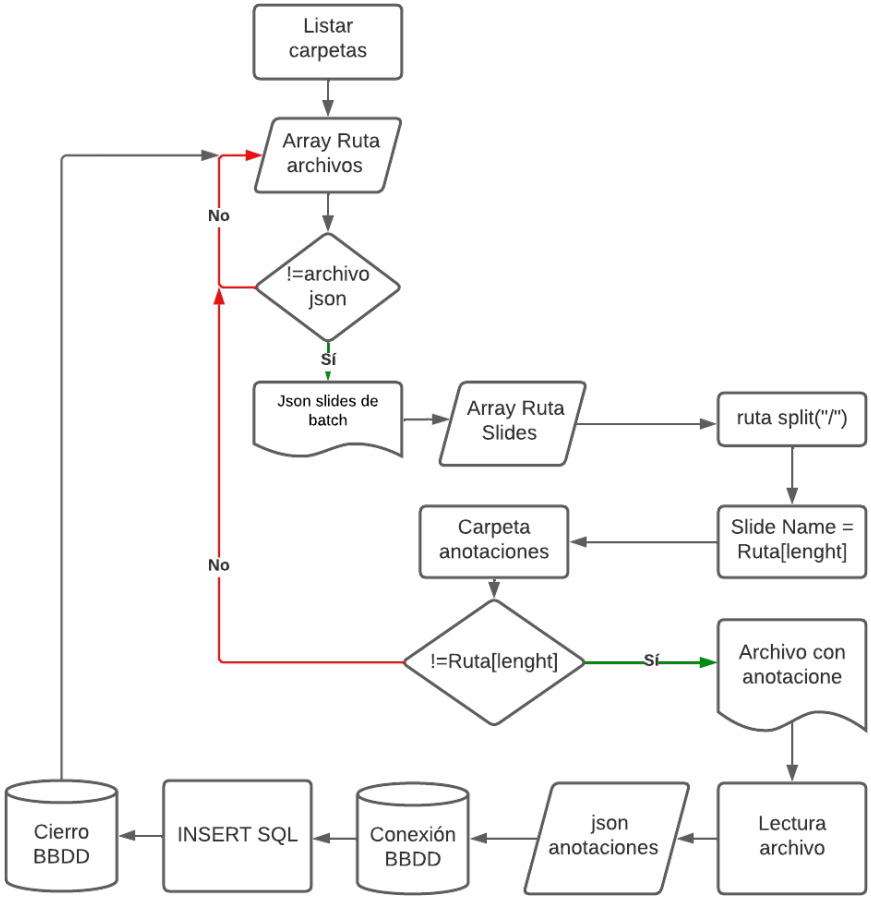


Figura 4.9: Diagrama de flujo para la subida de anotaciones autosegmentadas por los modelos de *Deep Learning*.

La ejecución de este proceso junto con el de predecir una slide se puede incluir en una API Figura A.1 que sea llamada desde el programa con el fin de ejecutar una predicción sobre la slide, esto por tiempo no se ha podido realizar, pero el diseño y funcionamiento es sencillo una vez se tiene el modelo y el código para generar y cargar las anotaciones a partir de la predicción, la idea que se deja planteada es que en un servidor que cuente con GPU se despliegue un servicio API/REST con HTTPS (para proteger la conexión y los datos) que escuche por el puerto 80, este servicio es fácilmente integrable con Kubernetes, el servicio se definiría con un método POST que reciba datos de la ruta donde se encuentra la WSI a la que pertenece la slide que se visualiza en la aplicación, con estos datos el servicio que tendría previamente cargado el modelo, ejecutaría la predicción y se encargaría de forma de realizar la ETL de las máscaras y guardaría en la tabla "KeyValue" (Figura 3.6) un nuevo registro para con los datos necesarios para que se visualicen las imágenes en la aplicación.

CAPÍTULO 5
Resultados

El resultado principal de este TFM es la mejora y extensión de una herramienta de anotación de imágenes gigapíxel que maneja grandes cantidades de datos de una forma ágil e inteligente para poder ser usada en la web. Gracias a un análisis profundo de la herramienta se ha alcanzado el conocimiento de las fortalezas, necesidades y limitaciones de la aplicación, lo que ha ayudado a diseñar una estrategia para mejorar sus prestaciones, y alcanzar los objetivos establecidos en este TFM.

- Mejoras en la infraestructura de la aplicación.
    - Se ha diseñado, evaluado e implementado una nueva estrategia de despliegue de la aplicación, la metodología que se ha seguido ha sido escogida teniendo en cuenta aspectos: tecnológicos, regulatorios y de funcionamiento de la herramienta, este análisis ha dado como resultado el despliegue de la herramienta usando el paradigma de microservicios con orquestación de contenedores Docker mediante el uso de Kubernetes.
    - El uso de Kubernetes junto con una configuración óptima del servidor consigue que la administración de la aplicación sea semiautomática, dando lugar a una mayor disponibilidad de la herramienta y a una gestión inteligente de los recursos que necesita.
    - Se ha desplegado un sistema de gestión de archivos en red eficiente y seguro para la subida y gestión de bathches (conjunto de WSI) a la aplicación web.
    - Las mejoras en la infraestructura dan como resultado una aplicación con mejor desempeño y con capacidad para dar servicio a más conexiones simultáneas, lo que se traduce en mayor capacidad de anotadores simultáneos sin perder calidad o de prestaciones en la experiencia de usuario de la aplicación.
- Extensión de funcionalidades.
    - Se ha implementado la funcionalidad de etiquetar a nivel global una imagen y además añadir observaciones para explicar el razonamiento que ha seguido el patólogo para ello.
    - Herramientas de etiquetado siguiendo el modelo Crowdsourcing, se ha habilitado un conjunto de menús que permite etiquetar una slide con un nivel de

certeza, el propósito de esto es poder evaluar desde distintos puntos de vista una misma imagen.

Estas dos funcionalidades tienen han tiene la premisa de compartir conocimiento entre patólogos, ya que en casos complicados puede llevar a un diagnóstico más acertado, de tal forma que repercutirá directamente en el tratamiento del paciente y en su posterior recuperación.

- Integración de modelos de predicción automática de regiones y etiquetado a nivel de slide. Se ha definido un *workflow* con el que se puede integrar las predicciones de modelos de DL creados en el grupo CVBLab en la aplicación web, esta integración permite que se evalúen los modelos con patólogos, para ello se les muestra las anotaciones en Microdraw dándoles la posibilidad junto con las herramientas de Crowdsourcing que puedan modificar la etiqueta global y pueden reanotar las regiones, lo que genera una nueva base de datos que ofrece la posibilidad de volver a entrenar el modelo ajustándolo, para que ofrezca con cada corrección una mejor *performance*.

Con la herramienta aquí explicada y con los resultados de este TFM, se ha alcanzado una herramienta que ofrece un fácil acceso, alta disponibilidad, manejo intuitivo y sencillo de las funcionalidades de la aplicación, además ofrece la posibilidad de compartir conocimiento entre patólogos. En resumen, se ha mejorado la herramienta Microdraw, potenciando las bondades que ofrece un servicio web e implementando nuevas funcionalidades que permiten que la patología digital siga creciendo y ofreciendo interesantes posibilidades en el apoyo al diagnóstico histopatológico.

Para comprobar el correcto funcionamiento de la aplicación se ha habilitado temporalmente un acceso para probar Microdraw (hacer clic aquí), al acceder a la aplicación hay que logearse con el server usando *como nombre de usuario cvblab* y *contraseña cvblab* (Figura 5.1a), al iniciar sesión se nos redirige al *login* de la aplicación donde se puede iniciar sesión con un usuario y contraseña.

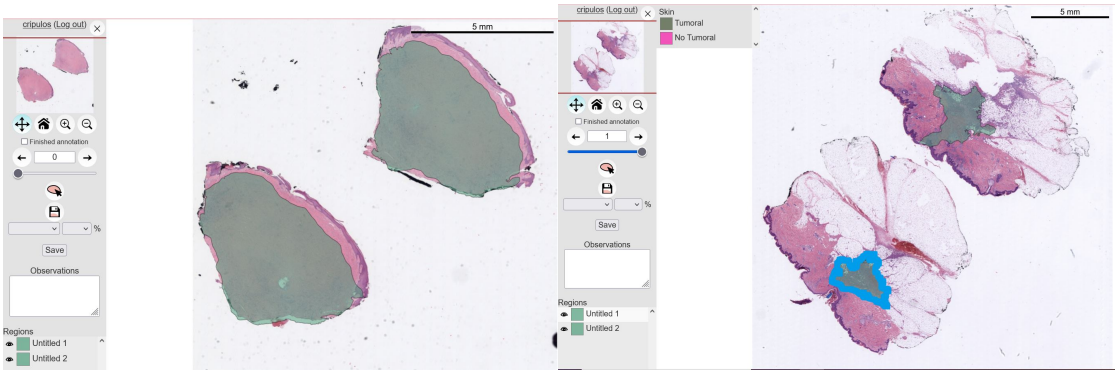
(a) *Login* server. **(b)** *Login* de la aplicación.

Figura 5.1: Página de inicio de la aplicación web.

Para acceder a la aplicación (Figura 5.1b) se han creado cinco cuentas, todas ellas tienen acceso al mismo *batch* por tiempo limitado hasta la defensa del presente TFM.

- Cuenta 1. Usuario **tfm**, pass **tfm**.
- Cuenta 2. Usuario **tfm2**, pass **tfm2**.
- Cuenta 3. Usuario **tfm3**, pass **tfm3**.
- Cuenta 4. Usuario **tfm4**, pass **tfm4**.
- Cuenta 5. Usuario **tfm5**, pass **tfm5**.

El *batch* que se ha habilitado contiene dos imágenes auto segmentadas con el modelo de DL que anota regiones tumorales, al entrar se verán dos anotaciones por slide, dichas anotaciones tiene se etiquetan como "*untitled*, tal y como se muestra en la Figura 5.2



(a) Slide uno. **(b)** Slide dos.

Figura 5.2: Pantalla inicial.

Las anotaciones del apartado "*Regions*" se etiquetan haciendo doble clic sobre el nombre, esto abrirá una ventana al lado de la barra lateral llamada "*Skin*" donde se puede elegir dos etiquetas, "Tumoral" o "No Tumoral" (Figura 5.3).

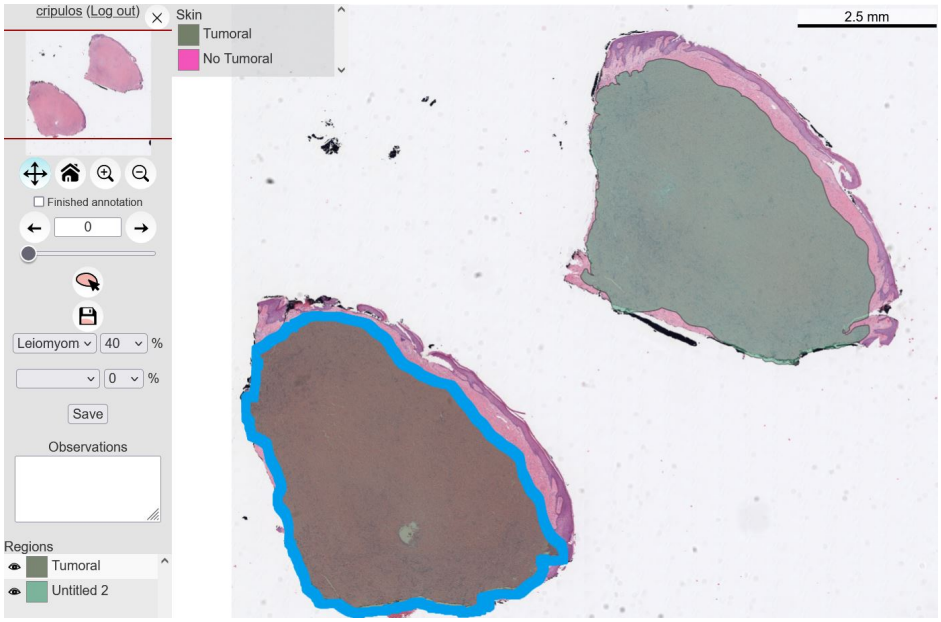
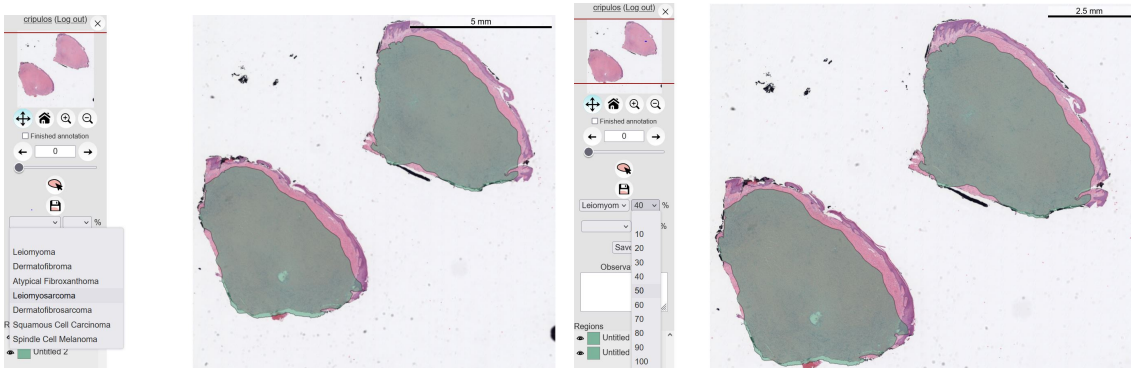


Figura 5.3: Selección de etiqueta de anotación.

Una vez evaluadas las anotaciones y etiquetadas, se tiene que etiquetar la slide a nivel global, para ello hay que elegir una de las siete de las neoplasias en las herramientas habilitadas para *Crowdsorucing* (Figura 5.4a) y el nivel de confianza (Figura 5.4b).



(a) Selección de neoplasia. **(b)** Selección de nivel de neoplasia.

Figura 5.4: Herramientas crowdsourcing.

Una vez se han dado nombre a las anotaciones se habilita el botón de dibujar, de esta forma se pueden crear nuevas regiones (Figura 5.5).

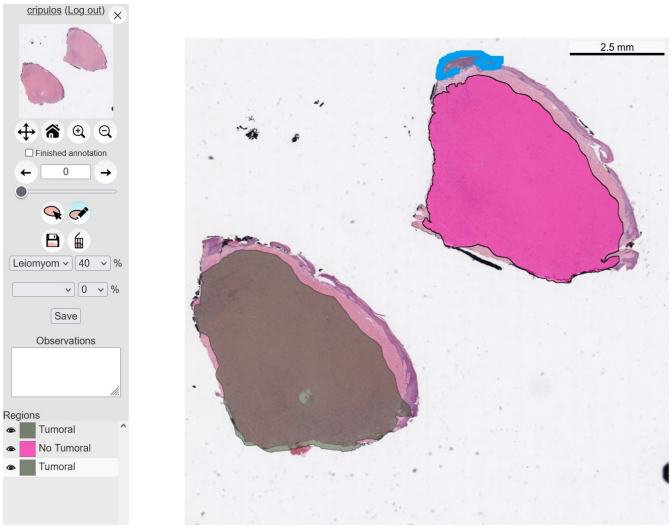


Figura 5.5: Creación de nueva anotación.

CAPÍTULO 6

Conclusiones y Trabajos Futuros

Los resultados expuestos en el apartado anterior demuestran que con las extensiones implementadas Microdraw se convierte en una herramienta muy completa y que ofrece funcionalidades innovadoras que no ofrecen las otras herramientas del estado del arte (Tabla 2.1); integración de innovadores modelos que sean evaluados con técnicas de crowdsourcing, la posibilidad de etiquetar batches por parte de muchos patólogos distintos a la vez, prestaciones que hacen posible la conexión en paralelo de muchos usuarios, interfaz amigable y sencilla que asegura una buena experiencia de usuario y alta disponibilidad de la herramienta.

Pensando en convertir esta herramienta como una aplicación de referencia en el campo de la histopatología por el potencial que tiene y su la posición de ventaja al compararse con las otras aplicaciones de código abierto del estado del arte, se han detectado factores que se pueden mejorar y con los cuales se alcanzaría una herramienta más que interesante para el diagnóstico histopatológico por ello se propone una lista de mejoras o implementaciones:

- Se recomienda migrar la aplicación a Javascript completamente, hoy en día con tecnologías como NodeJS permite que el *Backend* de la aplicación sea totalmente Javascript esto facilita mucho la codificación y expansión de las anotaciones.
- Implementar la propuesta de API/Rest expuesta en la Sección 4.3 para integrar la autosegmentación desde la propia aplicación.
- Migrar la base de datos a una no relacional como MongoDB es una buena idea pensando en el volumen de anotaciones que se puede llegar a manejar con el aumento de usuarios, este tipo de base de datos facilita esta tarea, ya que permite el almacenamiento de grandes volúmenes de datos de una forma distribuida, este tipo de base de datos son los usados por aplicaciones como Twitter que usa Cassandra.
- Realizar estudios de estabilidad para mejorar aún más la experiencia de usuario.
- Recomendaría migrar la aplicación a la nube porque ofrece muchas ventajas, la principal para una aplicación como la que aquí se habla es la escalabilidad.

- Revisar en profundidad la Ley de Protección de datos con un experto para encontrar la vía de implementar la aplicación en la nube.
- Realizar estudios y encuestas con patólogos para entender mejor sus necesidades a la hora de anotar, así como su forma de hacerlo, para ello se recomienda encuestas, entrevistas de grupo y entrevistas donde se aplique el método "*Thinkalud*" en el momento que se le presenta la herramienta a nuevos patólogos.
- La idea de esta herramienta es poner conocimiento en común, por lo que se recomienda habilitar algún espacio para compartir conocimientos, casos, dudadas, etc.
- Habilitar en la aplicación un espacio donde se pueda descargar datos para su reutilización, siempre respetando el RGPD, haciendo accesible los datos en formatos XML, JSON, CVS, etc.

Esta aplicación contribuye de una forma importante en la generación de bases de datos para entrenar modelos y en su validación, por lo que tiene una incidencia directa en el campo del diagnóstico patológico. El trabajo futuro es el de aplicar las mejoras que se han indicado y extenderá su uso dentro y fuera del grupo, para lo cual se publicará el código de la aplicación y la API que ha quedado por desarrollar, de este modo se contribuirá al campo y si se consigue extender su uso se generarían muchos modelos de predicción sobre distintos tipos de tejido que contribuirían al diagnóstico de muchos pacientes y mitigaría el cuello de botella derivado del bajo índice de patólogos por paciente al que nos vemos abocados.

Bibliografía

[1] Rish, Irina. An empirical study of the naive Bayes classifier. *IJCAI 2001 workshop on empirical methods in artificial intelligence*, 3:22:41-46, 2001.

[2] Jakkula, Vikramaditya. Tutorial on support vector machine (svm). *School of EECS, Washington State University*, 37:2.5:3, 2006.

[3] Andreas Daffertshofer and Claudine J.C. Lamoth and Onno G. Meijer and Peter J. Beek. PCA in studying coordination and variability: a tutorial. *Clinical Biomechanics*, 19:4:415-428, 2004.

[4] Atallah, Nehal M and Toss, Michael S and Verrill, Clare and Salto-Tellez, Manuel and Snead, David and Rakha, Emad A. Potential quality pitfalls of digitalized whole slide image of breast pathology in routine practice. *Modern Pathology*, 1:8, 2021.

[5] Atallah, Nehal M and Toss, Michael S and Verrill, Clare and Salto-Tellez, Manuel and Snead, David and Rakha, Emad A. Future-proofing pathology: the case for clinical adoption of digital pathology. *Journal of clinical pathology*, 70:12, 2017.

[6] Ström, Peter and Kartasalo, Kimmo and Olsson, Henrik and Solorzano, Leslie and Delahunt, Brett and Berney, Daniel M and Bostwick, David G and Evans, Andrew J and Grignon, David J and Humphrey, Peter A and others. Artificial intelligence for diagnosis and grading of prostate cancer in biopsies: a population-based, diagnostic study. *Journal of clinical pathology*, 70:12, 2017.

[7] Bulten, Wouter and Pinckaers, Hans and van Boven, Hester and Vink, Robert and de Bel, Thomas and van Ginneken, Bram and van der Laak, Jeroen and Hulsbergen-van de Kaa, Christina and Litjens, Geert. Automated deep-learning system for Gleason grading of prostate cancer using biopsies: a diagnostic study. *The Lancet Oncology*, 21:2:233-241 ,2020.

[8] Silva-Rodríguez, Julio and Colomer, Adrián and Sales, María A and Molina, Rafael and Naranjo, Valery. Going deeper through the Gleason scoring scale: An automatic end-to-end system for histology prostate grading and cribriform pattern detection. *Computer Methods and Programs in Biomedicine*, 19, 2020.

[9] Xu, Bolei and Liu, Jingxin and Hou, Xianxu and Liu, Bozhi and Garibaldi, Jon and. Ellis, Ian O and Green, Andy and Shen, Linlin and Qiu, Guoping. Look, investigate,

and classify: a deep hybrid attention method for breast cancer classification. *2019 IEEE 16th international symposium on biomedical imaging (ISBI 2019)*,2019.

[10] Wei, Jason W and Tafe, Laura J and Linnik, Yevgeniy A and Vaickus, Louis J and Tomita, Naofumi and Hassanpour, Saeed. Pathologist-level classification of histologic patterns on resected lung adenocarcinoma slides with deep neural networks. *Scientific reports*, 9:1:1-8, 2019.

[11] Pinckaers, Hans and Litjens, Geert. Neural ordinary differential equations for semantic segmentation of individual colon glands. *arXiv preprint arXiv*, 2019.

[12] Del Amor, Rocío and Morales, Sandra and Colomer, Adrián and Mogensen, Mette and Jensen, Mikkel and Israelsen, Niels M and Bang, Ole and Naranjo, Valery. Automatic segmentation of epidermis and hair follicles in optical coherence tomography images of normal skin by convolutional neural networks. *Frontiers in medicine*, 220, 2020.

[13] García, Gabriel and Colomer, Adrián and Naranjo, Valery. First-Stage Prostate Cancer Identification on Histopathological Images: Hand-Driven versus Automatic Learning. *Entropy*, 21:4:356, 2019.

[14] Ouyang, Wei and Mueller, Florian and Hjelmare, Martin and Lundberg, Emma and Zimmer, Christophe. ImJoy: an open-source computational platform for the deep learning era. *Nature methods*, 16:12:1199-1200, 2019.

[15] Bankhead, Peter and Loughrey, Maurice B and Fernández, José A and Dombrowski, Yvonne and McArt, Darragh G and Dunne, Philip D and McQuaid, Stephen and Gray, Ronan T and Murray, Liam J and Coleman, Helen G and others. QuPath: Open source software for digital pathology image analysis. *Scientific reports*, 7:1:1-7, 2017.

[16] De Chaumont, Fabrice and Dallongeville, Stéphane and Chenouard, Nicolas and Hervé, Nicolas and Pop, Sorin and Provoost, Thomas and Meas-Yedid, Vannary and Pankajakshan, Praveen and Lecomte, Timothée and Le Montagner, Yoann and others. Icy: an open bioimage informatics platform for extended reproducible research. *Nature methods*, 9:7:690-696, 2012.

[17] Schindelin, Johannes and Arganda-Carreras, Ignacio and Frise, Erwin and Kaynig, Verena and Longair, Mark and Pietzsch, Tobias and Preibisch, Stephan and Rueden, Curtis and Saalfeld, Stephan and Schmid, Benjamin and others. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9:7:676-682, 2012.

[18] Georges Ifrah. *Historia universal de las cifras*. Espasa Calpe, S.A., Madrid, sisena edició, 2008.

[19] Código Github Microdraw Consultado el 10 de Abril del 2022. https://github.com/r03ert0/microdraw.

[20] Documentación del framework de código abierto de Apache Hadoop 3.3.2. Consultado el 3 de Mayo del 2022. https://hadoop.apache.org/docs/stable/.

[21] Documentación del framework de código abierto de Apache Sparks 3.2.1. Consultado el 3 de Mayo del 2022. https://spark.apache.org/docs/latest/.

[22] Las cifras del cáncer en España 2022. Consultado el 12 de Mayo del 2022. https://seom.org/images/LAS_CIFRAS_DEL_CANCER_EN_ESPANA_2022.pdf.

[23] Oferta y necesidad de médicos especialistas en España Consultado el 12 de Mayo del 2022. https://www.sanidad.gob.es/profesionales/formacion/necesidadEspecialistas/home.htm.

[24] Traditional Pathology Scenario. Consultado el 25 de Mayo del 2022. https://www.futurebridge.com/industry/perspectives-life-sciences/digital-pathology/.

[25] n NIH Image. Consultado el 09 de Junio del 2022. https://imagej.nih.gov/nih-image/about.html.

[26] ImageJ web version. Consultado el 09 de Junio del 2022. https://ij.imjoy.io/.

[27] CheerpJ documentation. Consultado el 09 de Junio del 2022. https://docs.leaningtech.com/cheerpj/.

[28] imJoy documentation. Consultado el 09 de Junio del 2022. https://imjoy.io/docs/#/.

[29] imJoy repositorio GitHub. Consultado el 09 de Junio del 2022. https://github.com/imjoy-team/ImJoy.

[30] Software Upath de Roche. Consultado el 14 de Junio del 2022. https://diagnostics.roche.com/es/es/products/instruments/upath-enterprise-software.html

[31] Software Aignostic Consultado el 14 de Junio del 2022. https://aignostics.com/

[32] Software Aiforia Consultado el 14 de Junio del 2022. https://www.aiforia.com

[33] Software Aiforia Consultado el 14 de Junio del 2022. https://www.contextvision.com/

[34] Apache HTTP Server. Consultado el 15 de Junio del 2022. https://httpd.apache.org/

[35] MySql, motor de base de datos relacional. Consultado el 15 de Junio del 2022. https://www.mysql.com/

[36] Distro de Linux Ubuntu Server. Consultado el 15 de Junio del 2022. https://ubuntu.com/

[37] PHP, lenguaje de programación para backend. Consultado el 15 de Junio del 2022. https://www.php.net/

[38] JavaScript, información y tutoriales. Consultado el 15 de Junio del 2022. https://www.w3schools.com/js/default.asp

[39] HTML, información y tutoriales. Consultado el 15 de Junio del 2022. https://www.w3schools.com/html/default.asp

[40] CSS, información y tutoriales. Consultado el 15 de Junio del 2022. https://www.w3schools.com/css/default.asp

[41] Distribución de Apache. Consultado el 18 de Junio del 2022. https://www.apachefriends.org/es/index.html

[42] Página oficial de phpMyAdmin. Consultado el 18 de Junio del 2022. https://www.phpmyadmin.net/

[43] Manual de uso de MySQL. Consultado el 18 de Junio del 2022. https://dev.mysql.com/doc/refman/8.0/en/connecting.html

[44] Softaware MySQL Workbench. Consultado el 18 de Junio del 2022. https://www.mysql.com/products/workbench/

[45] Librería Paper.js. Consultado el 18 de Junio del 2022. http://paperjs.org/

[46] Librería OpenSeaDragon. Consultado el 18 de Junio del 2022. https://openseadragon.github.io/

[47] Compliance AWS. Consultado el 20 de Junio del 2022. https://aws.amazon.com/es/compliance/programs/

[48] Kubernetes. Consultado el 20 de Junio del 2022. https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/

[49] Github libvips. Consultado el 24 de Junio del 2022. https://github.com/libvips/libvips

[50] Red Neuronal en Python con Numpy. Consultado el 28 de Junio del 2022. https://artfromcode.wordpress.com/2017/04/18/red-neuronal-en-python-con-numpy-parte-1/

[51] Pulgarín Ospina, Cristian Camilo. Sistema de ayuda en la selección de embriones para la fertilización in vitro mediante la elaboración y el análisis de una robusta base de datos. *Universitat Politècnica de València*, 2019.

APÉNDICE A

Anexos

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**	X			
ODS 4. **Educación de calidad.**	X			
ODS 5. **Igualdad de género.**		X		
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades.**	X			
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

El presente TFM cumple con alguno de los objetivos de los Objetivos de desarrollo sostenible (ODS), a continuación se detallan cuáles y las razones.

- *ODS 3. Salud y bienestar, el nivel de cumplimiento es alto, ya que los resultados detallados en este TFM inciden directamente en la salud, pues usando la aplicación Microdraw con las extensiones que se detallan se puede agilizar los procesos de diagnóstico de cáncer.*
- *ODS 4. Educación de calidad, el nivel de cumplimiento es alto, la principal razón es que la extensión de crowdsourcing para validar modelos puede ser aplicada para evaluar el conocimiento de residentes porque el sistema de poder etiquetar a nivel global, una slide con un nivel de confianza tras ver regiones anotadas puede servir como método práctico para que los residentes adquieran experiencia.*
- *ODS 5. Igualdad de género, nivel de cumplimiento medio, la razón es que las cifras de cáncer de mama doblan al de próstata según el último estudio de la Sociedad Española de Oncología Médica, por lo que los resultados de este TFM consiguen que el volumen de casos asociados a cáncer de mama sean diagnosticados de una forma más ágil.*
- *ODS 8. Trabajo decente y crecimiento económico, nivel de cumplimiento alto. Se prevé que el volumen de casos de cáncer aumente, pero no lo haga el número de patólogos, esto va a producir un cuello de botella donde se sobre cargue de trabajo a los patólogos, con el uso de Microdraw se puede descargar de trabajo a los patólogos de tal forma que se podrá realizar una gestión óptima del volumen de casos.*
- *ODS 9. Industria, innovación e infraestructuras, nivel de cumplimiento bajo, esta aplicación innova en el workflow del diagnóstico patológico.*
- *ODS 10. Reducción de desigualdades, la digitalización de WSI y el diagnóstico digital permite que se puedan realizar el diagnóstico de más casos por patólogo, por lo que todo el mundo podría recibir un diagnóstico en las mismas condiciones.*

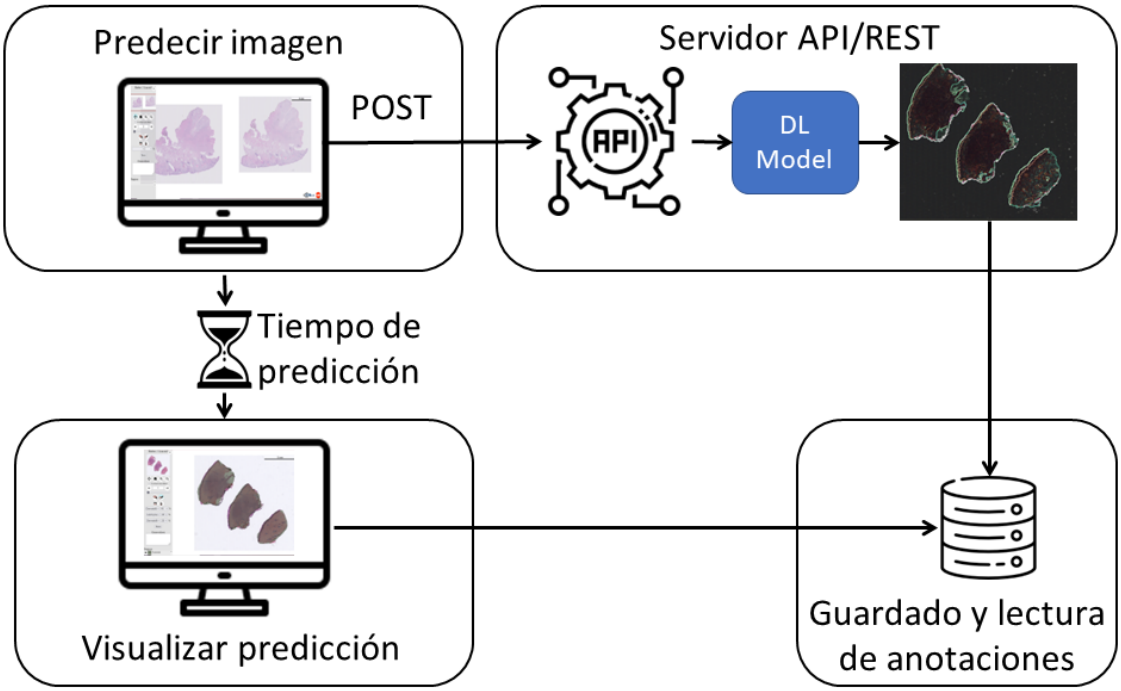


Figura A.1: Modelo API