Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Proyecto de grado
Ingeniería en computación

Microscopio Mágico

Guillermo Canabal, Guzmán Oholeguy y Camila Rosso
gcanabal.89@gmail.com, guzman.oho@gmail.com,
caamila110@gmail.com

Diciembre de 2019

Tutores del proyecto:

Héctor Cancela, Universidad de la República.
Federico Lecumberry, Universidad de la República.

Microscopio Mágico

Canabal, Guillermo, Oholeguy, Guzmán, Rosso, Camila

Proyecto de grado

Instituto de Computación - Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, Diciembre de 2019

RESUMEN

Este proyecto tiene como principal objetivo la creación de una herramienta que simule el comportamiento de un microscopio, motivado por el impulso de la utilización de herramientas tecnológicas en la educación y por la búsqueda de acercar cada vez más a los niños a áreas científicas a una edad más temprana. Los principales objetivos de este simulador son permitir el acercamiento de los niños al microscopio y poder darles una herramienta de soporte educativo para las clases de ciencia en la escuela. Este proyecto de grado se encuentra dentro del marco del proyecto Microscopio Mágico que tiene como uno de sus objetivos la creación de esta herramienta.

A partir de este objetivo se diseñó e implementó un sistema que incluye una base de imágenes tomadas con microscopios, una base de datos con los metadatos de las imágenes, un administrador para ingresar los datos y una aplicación web que simula el comportamiento del microscopio. Este sistema se diseñó teniendo en cuenta la posibilidad de escalar para ser utilizada por los alumnos de escuelas de todo el país.

AGRADECIMIENTOS

Agradecemos a Gabriela Speroni, Anabel Fernández, María Vittoria Di Tomaso, Silvia Olivera, Gonzalo Rosso, Juan Carlos Rosillo, Alejandro Bielli, José M. Verdes, Gabriela Casanova, Flavio Zoelsi, Patricia Casina, Rossana Sapiro, Gustavo Brum, Gregory Randall, Juan Claudio Benech por todos los aportes a este proyecto.

A Gustavo Armagno y Gonzalo Vilar por los aportes en diseño y experiencia de usuario que permitieron mejorar la aplicación del proyecto.

A Alejandra Kun responsable del proyecto Microscopio Mágico por su dedicación y motivación.

A nuestros tutores y también participantes del proyecto Microscopio Mágico Héctor Cancela y Federico Lecumberry por sus consejos, ayuda y dedicación durante el proyecto.

Tabla de contenidos

Capítulo 1

Introducción

Cada vez es más común el uso de software y recursos informáticos en el área de la educación, permitiendo mejorar su calidad e impulsar procesos de innovación social, inclusión y crecimiento personal. En el contexto de Uruguay desde diciembre del 2007 se ha implantado en las distintas escuelas, por medio del Plan Ceibal, el uso de computadoras portátiles por parte de los alumnos como herramienta de aprendizaje. Aprovechando este impulso, y entendiendo la dificultad de que la mayoría de los alumnos tenga acceso a microscopios y preparados en óptimas condiciones, surge el interés y motivación de brindar la posibilidad de contar con un microscopio virtual al cual puedan acceder desde su portátil.

En este marco surge el proyecto Microscopio Mágico que fue presentado en 2017 a la fundación Agencia Nacional de Investigación e Innovación (ANII) por biólogos, biofísicos, físicos e ingenieros pertenecientes a las Facultades de Ciencias, Veterinaria, Ingeniería y Medicina de la UdelaR y al Instituto de Investigaciones Biológicas Clemente Estable. Este proyecto tiene como uno de sus principales objetivos la creación de un microscopio virtual que simule la interacción con uno real. Este simulador obtendrá las imágenes a partir de un "banco" de fotografías generadas por el proyecto Microscopio Mágico mediante el uso de microscopios.

El proyecto que se describe en este informe esta incluido dentro del proyecto Microscopio Mágico, el cual posee un alcance mayor. Siendo así que en este proyecto no se incluye el tratamiento de los preparados y la obtención de las imágenes, también se debe tener en cuenta que el software implementado será una primera versión de la solución final y quedará por fuera del alcance la

puesta en producción de la solución en los servidores de Ceibal.

El proyecto de grado tiene dos objetivos principales, el primero de ellos es la implementación de una aplicación web que simule el uso de un microscopio y sea compatible con las computadoras provistas por Ceibal. Debe permitir a los usuarios finales, maestros y alumnos, acceder y explorar mediante el uso de una interfaz amigable e intuitiva la base de imágenes obtenidas a partir de microscopios. El segundo objetivo del proyecto es el diseño y desarrollo de un sistema que almacene las imágenes con su respectiva metadata que podrá ser accedida mediante una API REST pública. Una vez finalizado el proyecto se contará con la documentación del diseño y la arquitectura de los sistemas antes descriptos así como también el código implementado.

El resto del informe se organiza de la siguiente manera. En el capítulo 2 se describe la investigación y análisis de proyectos similares. En el capítulo 3 se listan y detallan los requerimientos del sistema. En el capítulo 4 se presenta el diseño de la solución del sistema. En el capítulo 5 se presentara la implementación del sistema. En el capítulo 6 se detalla el despliegue de la solución. En el capítulo 7 se muestran las pruebas realizadas y sus resultados. En el capítulo 8 se detalla la gestión del proyecto. Finalmente en el capítulo 9 se presentan los resultados, conclusiones y trabajo a futuro.

Capítulo 2

Contexto y antecedentes

En este capítulo se detalla el contexto y antecedentes del proyecto presentado en este informe. En la sección 2.1 se describe el proyecto Microscopio Mágico que es el marco de este proyecto, en la sección 2.2 se detallan los conceptos relacionados necesarios para mejorar la comprensión del proyecto, y en la sección 2.3 se detalla el relevamiento de productos y aplicaciones similares.

2.1. Microscopio Mágico

El proyecto Microscopio Mágico tiene como objetivo central la creación de una herramienta informática que acerque a los niños al área científica y al uso del microscopio. El fin de la herramienta es complementar y promover el uso del microscopio a partir de la simulación.

Este proyecto fue presentado en 2017 a la fundación Agencia Nacional de Investigación e Innovación (ANII)[1] a través de llamados que se desarrollan con la finalidad de financiar diferentes proyectos de innovación. En particular este proyecto se encuentra dentro del llamado Fondo Sectorial de educación: Inclusión digital[2] que está dirigido a grupos de investigación radicados en instituciones tanto nacionales como internacionales, públicas o privadas.

Dado que la creación del simulador involucra varios campos, como lo es la biología, la pedagogía y la ingeniería, en el equipo del proyecto están incluidos varios subgrupos de distintas áreas. Para poder crear el simulador es necesario contar con una gran base de imágenes tomadas con microscopios, estas imáge-

[1] https://www.anii.org.uy/inicio/

[2] https://www.anii.org.uy/apoyos/investigacion/73/fondo-sectorial-de-educacion-inclusion-digital/

nes de preparados son producidas por un equipo de biólogos, biofísicos, físicos e ingenieros pertenecientes a las Facultades de Ciencias, Veterinaria, Ingeniería y Medicina de la UdelaR y al Instituto de Investigaciones Biológicas Clemente Estable. Por otro lado, es necesario diseñar e implementar la aplicación en si misma, la cual incluye como principal funcionalidad la simulación de un microscopio, en esta parte es donde se cuenta con una mayor participación de los ingenieros y alumnos de la Facultad de Ingeniería.

Dentro de los objetivos también se incluyen las visitas a las escuelas para acercar esta aplicación a los alumnos pudiendo así explicarles de qué se trata el proyecto, mostrarles el funcionamiento del microscopio llevando microscopios ópticos de la Facultad de Ciencias y brindarles soporte sobre los microscopios que se encuentran en la escuelas. En estas actividades realizadas en las escuelas se busca por un lado enseñar los conceptos básicos de la microscopía y el funcionamiento del microscopio apoyándose en los microscopios, tarea que llevan a cabo los biólogos. Por otro lado, mostrar el funcionamiento del prototipo del simulador para que los niños puedan interactuar con ésta, tarea llevada a cabo por el equipo de la Facultad de Ingeniería y los diseñadores. Además se busca analizar los aspectos pedagógicos tanto de las actividades como de la aplicación, de forma de poder mejorar este aspecto para la utilización en la enseñanza. Por último, se busca recabar material audiovisual de las actividades y de entrevistas con los alumnos y maestros.

Los objetivos planteados en el llamado a la ANII fueron los siguientes:

"1.- Preparación de materiales. Los materiales biológicos serán preparados de acuerdo a protocolos habituales desarrollados por los laboratorios que integran el Proyecto. Brevemente, los materiales serán fijados con diferentes técnicas, de acuerdo a sus características propias y a los tiempos de mantenimiento que se prevean, tratando de conservar la fidelidad estructural y conformación máximas. Posteriormente las muestras serán incluidas en los medios de soporte elegidos de acuerdo a las secuencias convencionales habituales (deshidratación, infiltración, imbibición, polimerización). Cuando el procedimiento así lo requiera, las piezas serán seccionadas de acuerdo a su medio de inclusión, mediante micrótomo, criostato, vibrátomo o ultramicrótomo. Las secciones obtenidas serán teñidas con diversas técnicas y montadas en medios de montaje apropiados para su conservación.

2.- Obtención de imágenes. La observación al microscopio se acompañará del registro de imágenes digitales, a través de los sistemas de cada equipo en las

Instituciones en donde se obtendrán imágenes. Se trabajará con adquisición en diferentes planos focales que permitan variar los objetos enfocados, empleando diferentes niveles de magnificación. Las imágenes digitalizadas serán procesadas por diferentes programas de uso habitual (básicamente Adobe Phothoshop e Image J).

3.- Software. Se utilizarán técnicas estándar de Ingeniería de Software para las etapas de especificación de la herramienta, diseño, desarrollo y testing. Se buscará que el testing esté integrado con el desarrollo desde etapas tempranas.

4.- Videos. El objetivo en cuanto a la realización audiovisual es lograr una producción atractiva, que tenga un lenguaje contemporáneo y de aspecto profesional, que logre concitar la atención de docentes y alumnos mediante un uso didáctico y atractivo de la gramática de la imagen."

En el documento compartido[1] se puede encontrar la propuesta completa presentada a la ANII y en la página de la ANII en la sección de beneficiarios puede encontrarse el llamado del proyecto[2].

2.1.1. Otros materiales

Parte del equipo de biólogos del IIBCE realizó una entrevista en la radio El Espectador que se puede encontrar en su página[3].

Para el IV Congreso Internacional de Enseñanza de las Ciencias Básicas el 10 de Octubre de 2019 en Paysandú[4] se realizó un poster[5] y una presentación oral dentro del eje "Herramientas o medios para despertar vocaciones en carreras universitarias que incluyan alta carga de Ciencias Básica" realizada por Gustavo Armagno.

2.2. Conceptos relacionados

Dado que la microscopía tiene un papel fundamental dentro de este proyecto, a continuación se presentan conceptos relacionados que ayudan a com-

[1] https://docs.google.com/document/d/1y1arzTr3H41tVL73SrO0fpI4j739RGcpCXJwcd2O4wk/edit?usp=sharing

[2] https://www.anii.org.uy/proyectos/FSED_2_2017_1_138850/el-microscopio-magico-explorando-micromundos/

[3] https://espectador.com/mastemprano/entrevista/microscopio-magico-la-iniciativa-que-busca-acercar-la-ciencia-a-los-escolares

[4] http://cieciba.multisitio.interior.edu.uy/presentacion-de-trabajos/

[5] https://drive.google.com/file/d/1k26uMck7KdqdxFnvDDxuCDmfpK7IGdzF/view?usp=sharing

prender la realidad planteada.

Microscopio[14][15]

El microscopio (del griego micrós, 'pequeño', y scopéo, 'mirar') es una herramienta que permite observar objetos que son demasiado pequeños para ser observados a simple vista. El tipo más común y el primero que fue inventado es el microscopio óptico. Se trata de un instrumento que contiene dos lentes que permiten obtener una imagen aumentada del objeto y que funciona por refracción. La ciencia que investiga los objetos pequeños utilizando este instrumento se llama microscopía. Es a través de este dispositivo que se toman las imágenes del banco de imágenes.

En la imagen 2.1 se puede ver un microscopio óptico donde sus partes principales están numeradas, a continuación se nombra cada una de las partes.

Figura 2.1: Microscopio óptico

- 1: Lentes oculares
- 2: Revólver
- 3: Objetivos
- 4: Macrométrico
- 5: Micrométrico
- 6 y 9: Platina
- 7: Fuente de luz
- 8: Diafragma y condensador

Preparados

Los preparados son las muestras en este caso biológicas que son observadas mediante el uso de microscopios. Estas muestras pueden ser sometidas a distintos tratamientos los cuales pueden estar determinados por el tipo de microscopio o técnica a utilizar para su estudio.

Magnificación

En un microscopio se tienen varios objetivos (lente situado en el revólver) que permiten ver el preparado con diferentes magnificaciones. Estas magnificaciones indican el aumento con el que se está viendo el preparado, las cuales dependen del microscopio.

Foco[13]

En óptica geométrica un foco es el punto donde convergen los rayos de luz originados desde un punto en el objeto observado. Aunque el foco es conceptualmente un punto, físicamente el foco tiene una extensión espacial, llamada círculo borroso. Una imagen, o punto de imagen, se dice que está en foco si la luz de los puntos del objeto converge lo más posible en la imagen, y fuera de foco si la luz no converge adecuadamente.

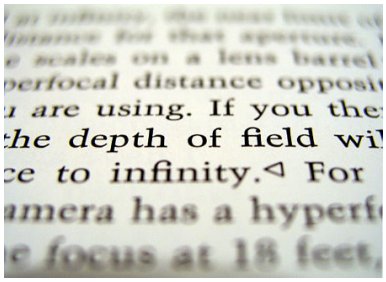


Figura 2.2: Imagen parcialmente enfocada, pero mayormente fuera de foco en grados variables.

Macrométrico y micrométrico[15]

Son tornillos de enfoque, mueven la platina o el tubo hacia arriba y hacia abajo. El macrométrico, permite desplazamientos amplios para un enfoque inicial y el micrométrico, desplazamientos muy cortos, para el enfoque más preciso. Pueden llevar incorporado un mando de bloqueo que fija la platina o el tubo a una determinada altura.

Condensador[15]

Lente que concentra los rayos luminosos sobre la preparación.

Diafragma[15]

Regula la cantidad de luz que llega al condensador.

2.3. Relevamiento de aplicaciones similares

En esta sección se presenta la investigación y análisis de aplicaciones similares a la propuesta en el proyecto. En el análisis se busca contemplar si existen aplicaciones que cumplan con los requerimientos de la aplicación propuesta en el proyecto.

Las aplicaciones analizadas fueron obtenidas a partir de integrantes del proyecto que las encontraron o conocían previamente y búsquedas realizadas en la Web.

2.3.1. Productos analizados

Para el análisis de las aplicaciones encontradas se definieron funcionalidades que se espera que posea la aplicación final y luego se validaron y compararon contra las funcionalidades provistas por las diferentes aplicaciones.

Las aplicaciones relevadas son las siguientes:

- Histology Guide[1] (HG)
- Simulador Universidad de Delaware[2][3] (SUD)
- Virtual Labs: Using the Microscope[4] (VL)
- Online Scanning Electron Microscope (SEM) Simulator[5]
- MyScope: Virtual Light Microscopy[6] (MS)
- Virtual Microscope[7] (VM)
- WebScope[8] (WS)
- Atlas de Histología Vegetal y Animal[9] (AHVA)

En la tabla 2.1 se presenta el análisis de las distintas funcionalidades para cada una de las aplicaciones del listado anterior.

Cabe destacar que ninguna de las aplicaciones encontradas es de código abierto, lo cual impide su modificación. No se relevaron aplicaciones pagas, únicamente aplicaciones de uso gratuito.

[1] http://www.histologyguide.com
[2] http://www1.udel.edu/present/profiles/ketcham/index.html
[3] https://www1.udel.edu/biology/ketcham/microscope/scope.html
[4] https://virtuallab.nmsu.edu/micro.php
[5] http://myscope-explore.org/index.html
[6] http://www.ammrf.org.au/myscope/confocal/virtual/
[7] https://www.virtualmicroscope.org/
[8] https://histology.medicine.umich.edu/full-slide-list
[9] https://mmegias.webs.uvigo.es/7-micro-virtual/flash/inicio-flash-todas.html

	HG	SUD	VL	SEM	MS	VM	WS	AHVA
Variedad de preparados (superior a veinte)	Si	No	No	Si	No	No	Si	Si
Galería de preparados	Índice alfabético	No	No	No	No	Si	No	No
Exploración de preparado	Si	Si	No	No	Si	Si	Si	Si
Minimapa	Si	No	No	No	No	No	Si	Si
Magnificaciones (objetivos)	Barra	Objetivos	No	Barra	Objetivos	Barra	Objetivos y Barra	Barra
Micrométrico y macrométrico	No	Si	No	Barra	No	No	No	No
Intensidad de luz	No	Si	No	Si	No	No	No	No
Contenidos teóricos	Si	No	Si	Si	Si	Si	Si	Si
Contenidos lúdico	Preguntas	Actividades con microscopio	Tutoriales interactivos	Desafíos	Preguntas	No	No	No
Tipo de microscopio	Óptico y Electrónico	Óptico	Óptico	Electrónico	Óptico	No se nombra	Óptico y Electrónico	No se nombra
Enseñar a usar el microscopio físico	No	Tutoriales	Tutorial interactivo	Tutoriales	Tutorial interactivo	No	Tutoriales	No
Facilidad de uso	Si	No	Si	Si	No	Si	Si	Si
Guías	Si	Si	Si	Si	Si	Si	Si	No
Conocimientos previos necesarios	Si	No	No	Si	Si	Si	Si	No
Idioma	Inglés	Inglés	Inglés	Inglés	Inglés	Inglés	Inglés	Español
Online	Si	Si	Si	Si	Si	Si	Si	Si
Necesidad de instalar extensiones	No	Adobe Flash Player	Adobe Flash Player	No	Adobe Flash Player	No	Adobe Flash Player	Versión con Adobe Flash Player y versión con HTML5
Año de finalización	2008	2003	2014	–	–	2008	–	2007

Tabla 2.1: Aplicaciones similares.

El caso de la aplicación web Virtual Labs: Using the Microscope no es exactamente un microscopio virtual porque no permite explorar preparados, pero tiene muchos tutoriales interactivos de distintas actividades con el microscopio, desde su utilización a la limpieza y la preparación de preparados.

De estas aplicaciones, por sus características, se eligieron las siguientes para hacer un análisis más en detalle: Histology Guide, Simulador Universidad de Delaware y Virtual Microscope. A continuación se detalla el análisis realizado para estas tres aplicaciones.

2.3.1.1. Histology Guide

Histology Guide[1] tiene como objetivo enseñar el reconocimiento visual de estructuras de células y tejidos y el entendimiento de como estas estructuras están determinadas por su función. No reproducen la información que se puede encontrar en un libro de histología, sino que se le muestra a los usuarios como aplicar ese conocimiento para interpretar las células y tejidos cuando las ven a través de un microscopio. Histology Guide recrea el uso del microscopio con una interfaz web intuitiva y está destinado a ser utilizado en conjunto con libro de histología. La autoría de esta aplicación es de T. Clark Brelje, Ph.D. and Robert L. Sorenson.

Dentro de la página se puede encontrar varias pestañas con diferente contenido, dentro de las cuales se encuentra la sección de diapositivas donde tienen contenido teórico dividido en capítulos. Además tienen una pestaña de microscopia electrónica donde también tienen contenido teórico dividido en capítulos, a partir de los cuales se puede ingresar a los preparados para visualizarlos como si se estuviera utilizando un microscopio. Los preparados fueron obtenidos con microscopios ópticos y microscopios electrónicos con imágenes de alta resolución. Una vez dentro del preparado permite explorar el preparado y cambiar la magnificación. En esta aplicación no se incluye el concepto de foco al explorar un preparado.

Histology Guide tiene una sección de evaluaciones donde para cada capítulo de las otras secciones tienen preguntas de lo aprendido. También tienen un índice con todos los preparados y un buscador, los cuales se encuentran en pestañas separadas. Por ultimo tiene una sección de ayuda donde explican distintas características de la página.

[1] http://www.histologyguide.com

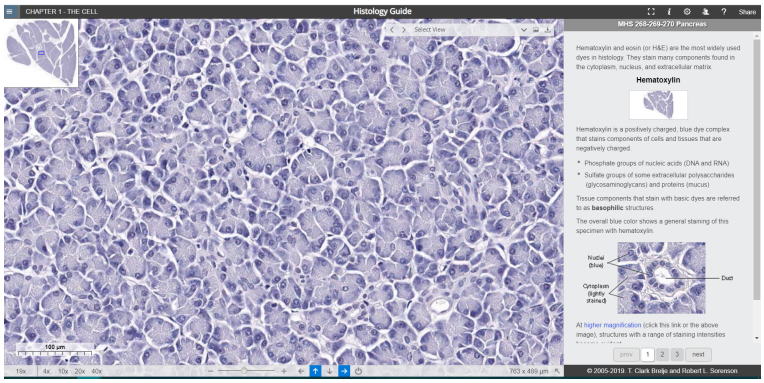


Figura 2.3: Histology Guide

2.3.1.2. Simulador Universidad de Delaware

El simulador de la universidad de Delaware[1] no contiene contenido teórico y la aplicación consta de una única pantalla, pero tiene implementados todas las funcionalidades del microscopio óptico: colocación del portaobjetos en el microscopio, movimiento de la platina horizontalmente y verticalmente, magnificación con objetivos, micrométrico y macrométrico, intensidad de la luz, diafragma, encendido de la luz y ajuste de oculares. Sumado a esto, todos los botones que implementan las funcionalidades están diseñados en semejanza con los botones físicos del microscopio, también simulando el movimiento de los botones físicos como girar las perillas o mover el revolver con los objetivos.

Se puede cambiar entre dos vistas, una vista donde se manipula el microscopio como el real, y otra vista que se centra en la visualización del preparado pero donde los botones con las funcionalidades continúan siendo los mismos pero redistribuidos en la pantalla. Esto facilita la asociación de los botones de la aplicación con los del microscopio.

El simulador tiene un tutorial que explica el funcionamiento de todos los botones del microscopio y como utilizarlos para mejorar la visualización del preparado ingresado.

Como contenido lúdico tiene una parte con actividades para utilizar el microscopio, por ejemplo mejorar la visualización de un preparado modificando la posición del preparado, la magnificación o la luz.

[1] https://www1.udel.edu/biology/ketcham/microscope/scope.html

Figura 2.4: Simulador Universidad de Delaware

2.3.1.3. Virtual Microscope

La aplicación Virtual Microscope[1] cuya autoría es de The Open University del Reino Unido, permite explorar y examinar preparados de minerales y rocas principalmente del Reino Unido aunque también brinda la posibilidad de acceder a algunos preparados de minerales y rocas espaciales y algunos otros preparados de índole variada.

Cada preparado se encuentra categorizado y se provee información detallada. La aplicación permite, para el caso de los preparados del Reino Unido, a los usuarios acceder a los minerales desde un mapa que muestra la ubicación de estos en el territorio del Reino Unido, pudiendo visualizar su distribución territorial.

Una vez que se accede a la exploración del preparado se puede cambiar la magnificación, además en algunos casos se resaltan puntos de puntos de interés donde se permite acceder a información extra. También brinda al usuario la posibilidad de compartir el preparado que se está visualizando a través de un enlace generado por la aplicación.

[1] https://www.virtualmicroscope.org/

Figura 2.5: Virtual Microscope

2.3.2. Conclusiones del relevamiento

Dadas las características deseadas en la aplicación de este proyecto en comparación con las características relevadas en los productos, se concluye que los productos relevados no cumplen con todas ellas. La mayoría de las aplicaciones relevadas parecen ser destinadas a usuarios con conocimientos científicos y no a niños que es el principal destinatario de la aplicación de este proyecto. Además las aplicaciones de las que pudimos obtener fecha de publicación, fueron desarrolladas hace ya varios años y ninguna es de código abierto como para poder ser adaptada a los requerimientos del proyecto.

Capítulo 3

Requerimientos

Este proyecto tiene como objetivo crear una aplicación que sirva como soporte para la enseñanza de primaria tanto para el aprendizaje del funcionamiento de un microscopio, como para el apoyo al programa del curso. En este marco se plantean como funcionalidades principales del producto final las funciones que cumple un microscopio real, permitiendo así a través de imágenes explorar el micromundo. Las funcionalidades principales son:

- Listado de preparados
- Búsqueda de preparados
- Visualizar contenido teórico
- Exploración de un preparado
- Cambios de magnificación de un preparado
- Cambio de foco de un preparado

En la sección 3.1 se detallan los requerimientos funcionales, en la sección 3.2 los requerimientos no funcionales y en la sección 3.3 se define el alcance para este proyecto.

3.1. Requerimientos funcionales

3.1.1. Biblioteca de preparados

Se debe contar con un biblioteca de preparados que contenga las fotografías de varios preparados obtenidas por medio de los distintos microscopios. Para cada preparado se debe contar con un conjunto de imágenes con diferentes

niveles de magnificación. Se permitirá almacenar imágenes de gran tamaño. Los formatos de imágenes aceptados serán JPEG.

3.1.2. Listado de preparados

El sistema debe contar con una galería de preparados que liste los preparados obtenidos a partir de la biblioteca para poder acceder luego a un preparado en particular.

3.1.3. Búsqueda de preparados

Debe permitir buscar preparados por palabras dentro del nombre, la descripción y/o categoría.

3.1.4. Visualización y exploración de preparado

Los usuarios tendrán la posibilidad de experimentar el uso de un microscopio mediante una simulación. Esta simulación implica que los usuarios puedan visualizar y explorar los preparados de la biblioteca, comprendiéndose por explorar la navegación en el plano del preparado.

Dentro de la visualización del preparado debe haber una pantalla informativa que tenga texto descriptivo sobre el preparado que se está explorando.

3.1.5. Cambio de objetivo

El sistema debe contar con una funcionalidad que simule el cambio de objetivo del microscopio, lo que permite al usuario visualizar los preparados con distintos niveles de detalles.

3.1.6. Cambio de foco

El sistema debe simular el acceso a la posición en foco (en el eje óptico Z) del microscopio. Se debe contar con imágenes fotografiadas desde distintos focos, teniendo imágenes en foco y fuera de foco para cada preparado.

3.1.7. Cambio en intensidad de la luz

El sistema debe contar con una funcionalidad que simule el cambio en la intensidad de luz del microscopio. Debe realizarse modificando la imagen dinámicamente.

3.1.8. Contenido teórico

Se debe contar con distintos contenidos referentes a la teoría de la microscopía.

3.1.9. Anotaciones

Se permitirá al usuario agregar sus propias anotaciones al preparado que esté visualizando.

3.1.10. Ingresar preparados (administrador)

El sistema debe contar con un administrador que permita ingresar los preparados con todos sus datos.

3.1.11. Post-procesamiento de imágenes (administrador)

Al ingresar el preparado se debe poder subir las imágenes del preparado y que se realice el post-procesamiento necesario: *stitching* y *tiling* de las imágenes.

3.1.12. Modificar preparados (administrador)

El sistema debe contar con un administrador que permita modificar los preparados ingresados en el sistema.

3.1.13. Eliminar preparados (administrador)

El sistema debe contar con un administrador que permita eliminar los preparados ingresados al sistema.

3.1.14. Ingresar categorías (administrador)

El sistema debe contar con un administrador que permita ingresar nuevas categorías para categorizar a los preparados.

3.1.15. Eliminar categorías (administrador)

El sistema debe contar con un administrador que permita eliminar categorías ingresadas en el sistema.

3.2. Requerimientos no funcionales

3.2.1. Funcionamiento en equipos Ceibal

El sistema debe poder ser utilizado desde los equipos Ceibal provistos a las escuelas.

Restricciones de hardware Debe tenerse en cuenta el uso de CPU, memoria y almacenamiento en disco del hardware sobre el que va a operar.

Restricciones de ancho de banda Se debe poder interactuar con el sistema utilizando las redes del instituto educativo.

3.2.2. Usabilidad y accesibilidad

El manejo del software tendrá una aproximación lúdica, instrumentada mediante interacciones que permitan el ensayo y el error. Brindando descripción de las componentes, ventanas de ayuda y permitiendo una fácil utilización por parte de los niños.

3.2.3. Concurrencia y escalabilidad

La aplicación debe soportar el uso concurrente de varios usuarios con la capacidad de escalar, de forma tal que pueda ajustarse a un aumento de la demanda.

3.3. Alcance

Dentro del alcance del proyecto se incluyen los requerimientos listados anteriormente exceptuando los siguientes:

- 3.1.9: Dentro del alcance queda parte de este requerimiento, se permite agregar anotaciones pero no se guardan.
- 3.1.11: El post-procesamiento de las imágenes debe realizarse por un usuario y subirse al servidor por FTP. Queda fuera del alcance que esto sea realizado por el sistema.
- 3.1.12 y 3.1.13: Queda fuera del alcance la modificación y eliminación de preparados.
- 3.1.15: Queda fuera del alcance la eliminación de categorías.
- 3.2.2: Queda fuera del alcance la aproximación lúdica de este requerimiento.

Capítulo 4

Diseño de la solución

En este capítulo se detalla la arquitectura con los casos de uso obtenidos a partir de los requerimientos y el diseño de la arquitectura. Luego se explica detalladamente el modelo de datos utilizado en la solución. Seguido de esto se listan las tecnologías utilizadas para las componentes y funcionalidades de la aplicación. Para continuar se explican los detalles de la infraestructura y las estrategias de escalabilidad, finalmente se detalla el diseño de la interfaz de usuario.

En la sección 4.1 se detalla el diseño de la arquitectura y el análisis de la escalabilidad. En la sección 4.2 se detalla el modelo de datos utilizado. Luego en la sección 4.3 se profundiza sobre las tecnologías utilizadas. Por último, en la sección 4.4 se detalla el diseño de la interfaz de usuario.

4.1. Arquitectura

4.1.1. Introducción

En este documento se presentan los diseños y detalles con respecto a la arquitectura del sistema Microscopio Mágico. Esto incluye los casos de uso identificados, la vista lógica del sistema, estilo arquitectónico, diseños de las distintas capas y la distribución final del sistema.

4.1.2. Vista de Casos de Uso

En esta sección se identifican y describen las secuencias de interacciones entre los actores externos y el sistema.

4.1.2.1. Casos de uso identificados

A partir de los requerimientos del sistema se definen para cada componente los casos de uso, identificando cuales de estos son críticos.

Se debe contar con una componente BackOffice, la cual va a ser utilizada únicamente por los usuarios administradores del sistema, esta componente deberá cumplir los siguientes casos de uso:

- Autenticación de usuarios: El administrador de BackOffice podrá autenticarse a través de un usuario y contraseña de administrador.
- Ingresar nuevo preparado: El administrador podrá ingresar nuevos preparados al sistema.
- Eliminar preparado: El administrador podrá eliminar preparados del sistema.
- Obtener listados de preparados: El administrador podrá obtener un listado de todas los preparados ingresados.
- Configuración de preparados: El administrador podrá configurar los preparados ingresados.

Se contará con una componente FrontOffice, a la cual tendrán acceso todos los usuarios del sistema, esta componente deberá cumplir los siguientes casos de uso:

- Autenticación de usuarios: Los alumnos, maestros y administradores del sistema podrán autenticarse a través de un usuario y contraseña.
- **Obtener listados de preparados (crítico):** El usuario podrá obtener un listado de todas los preparados ingresados.
- Buscar preparado: El usuario podrá realizar una búsqueda sobre la lista de preparados del sistema.
- **Explorar preparado (crítico):** Los usuarios podrán explorar los preparados ingresados, simulando el desplazamiento de la platina.
- Cambiar magnificación: Los usuarios contarán con la posibilidad de seleccionar las distintas magnificaciones disponibles para cada preparado, simulando el uso de los objetivos del microscopio.
- Cambiar profundidad: Los usuarios contarán con la posibilidad de seleccionar las distintas profundidades disponibles para cada preparado, simulando el uso del micrométrico y micrométrico del microscopio.

- Modificar entrada de luz: Los usuarios contarán con la posibilidad de aumentar y disminuir el brillo del preparado, simulando el uso del regulador de la intensidad de luz del microscopio.
- Visualizar información del preparado: El usuario podrá visualizar información teórica correspondiente al preparado que se encuentra explorando.
- Visualizar minimapa: Al momento de explorar el usuario contará con un minimapa que lo ayudará a ubicarse en el preparado.
- Agregar puntos de interés: Al momento de explorar el administrador contará con la posibilidad de agregar puntos de interés y una breve descripción de los mismos.
- Visualizar puntos de interés: Al momento de explorar el usuario podrá visualizar los puntos de interés añadidos por el administrador.
- Visualizar contenido teórico: El usuario podrá visualizar los contenidos teóricos existentes.

4.1.2.2. Diagrama de Casos de Uso Críticos

Se toma como críticos los casos de uso explorar preparado y obtener listados de preparados ya que se considera que son los más utilizados por los actores y los que tienen un mayor impacto sobre el sistema. En la figura 4.1 se muestra el diagrama de casos de uso críticos.

4.1.2.3. Actores

A continuación se listan distintos actores (agentes externos) que utilizan el sistema:

- Administrador BackOffice: Usuario que accede al sistema habiendo iniciado sesión en el BackOffice.
- Administrador FrontOffice: Usuario que accede al sistema habiendo iniciado sesión en el FrontOffice.
- Alumnos y maestros: Usuario que accede al sistema habiendo iniciado sesión en el FrontOffice.

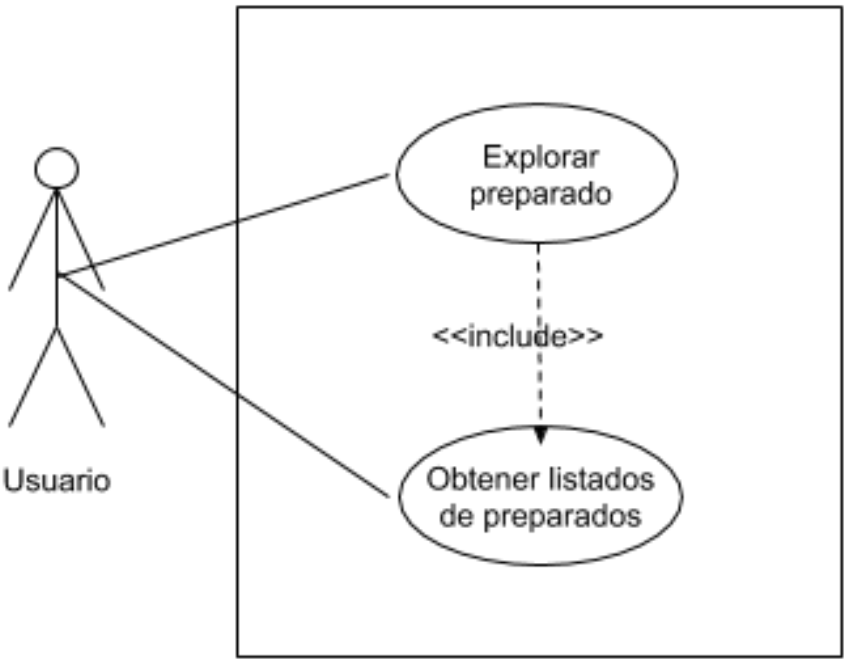


Figura 4.1: Diagrama de caso de uso críticos

4.1.3. Especificación de Casos de Uso Críticos

El primer caso de uso crítico definido es que un usuario obtenga el listado de los preparados ingresados en el sistema. En la tabla 4.1 se detalla el flujo de eventos necesarios para el éxito del caso de uso.

Pre-condiciones	El usuario está autenticado.
Acción de Actor	**Respuesta del Sistema**
1 - El usuario selecciona la página galería.	
	2 -El sistema redirige al usuario a la página de galería.
	3 -El sistema lista los preparados existentes.
Post-condiciones	Se despliega el listado de preparados en pantalla.

Tabla 4.1: Eventos del caso de uso obtener listado de preparados

En el segundo caso de uso crítico el usuario explora el preparado seleccionado, pudiendo desplazarse a las distintas áreas. En la tabla 4.2 se detalla el flujo de eventos necesarios para el éxito del caso de uso.

Pre-condiciones	El usuario está autenticado y se encuentra en la vista de galería.
Acción de Actor	**Respuesta del Sistema**
1 -El usuario selecciona el preparado de su interés.	
	2 -El sistema carga el preparado seleccionado en la vista de exploración.
3 -El usuario desplaza por los distintos sectores del preparado.	
	4 -El sistema despliega los sectores según el usuario se vaya desplazando a los mismos.
Post-condiciones	Se despliega el simulador del preparado mostrando las secciones elegidas por el usuario.

Tabla 4.2: Eventos del caso de uso explorar preparado

4.1.4. Vista Lógica

A continuación se describe la estructura del sistema, mostrando el estilo arquitectónico y los subsistemas que lo componen.

4.1.4.1. Estilo Arquitectónico

Para la resolución del proyecto se decidió utilizar una arquitectura en capas estricta (ver figura 4.2) basada en microservicios, donde las capas inferiores brindan servicios consumidos por las capas superiores. Es importante mantener independencia entre las distintas componentes, permitiendo que en un futuro otras aplicaciones independientes puedan utilizar los servicios brindados por la capa de servicios para acceder a los datos almacenados.

4.1.4.2. Subsistemas

En la capa de presentación que se puede observar en la figura 4.3, se implementan los componentes encargadas de generar las interfaces de usuario a ser utilizadas por los administradores, alumnos, y maestros.

- BackOffice: encargado de la interfaz gráfica utilizada por los administradores. Está dividida en dos secciones donde una se encarga de la interfaz y la otra de la comunicación con la capa de servicios.

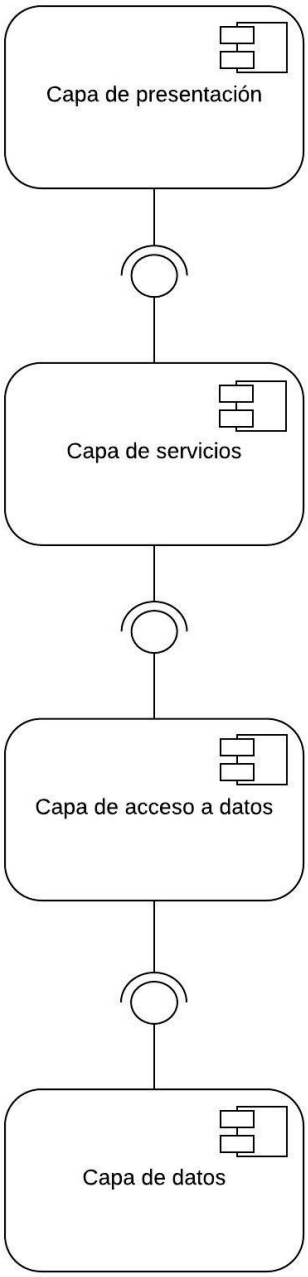


Figura 4.2: Diagrama de capas

- FrontOffice: encargado de la interfaz gráfica utilizada por los administradores, alumnos, y maestros. Está dividida en dos secciones donde una se encarga de la interfaz y la otra de la comunicación con la capa de servicios.

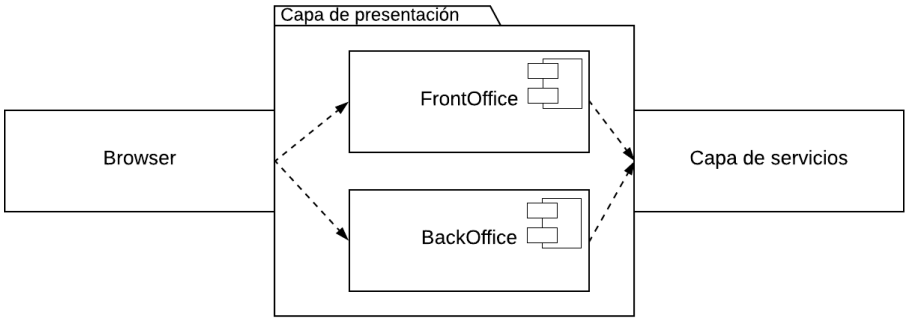


Figura 4.3: Diagrama de componentes de la capa de presentación

La capa de servicios que se puede observar en la figura 4.4, se encarga de publicar e implementar la lógica y requerimientos de cada caso de uso. Los componentes identificados son:

- PreparadoService: encargado de dar de alta y gestionar los preparados.
- TeoricoService: encargado de gestionar los contenidos teóricos.
- UsuarioService: encargado de manejar los registros de los usuarios y autorizar el acceso a los diferentes componentes de la capa de presentación.

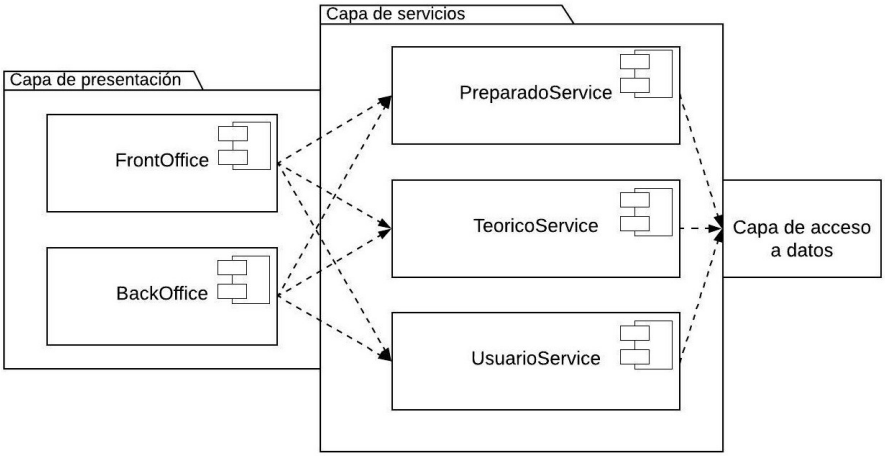


Figura 4.4: Diagrama de componentes de la capa de servicios

La capa de acceso a datos que se puede observar en la figura 4.5, se encarga de centralizar las operaciones hacia la capa de datos. Los componentes identificados son:

- PreparadoDataAccess: encargado de las operaciones de alta, baja, modificación y consulta de preparados.
- TeoricoDataAccess: encargado de las operaciones de alta, baja, modificación y consulta de los contenidos teóricos.
- UsuarioDataAccess: encargado de las operaciones de alta, baja, modificación y consulta de los usuarios.

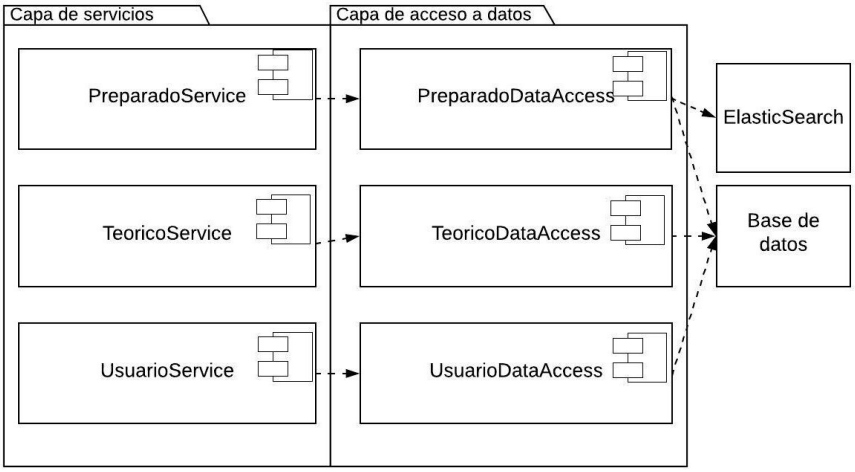


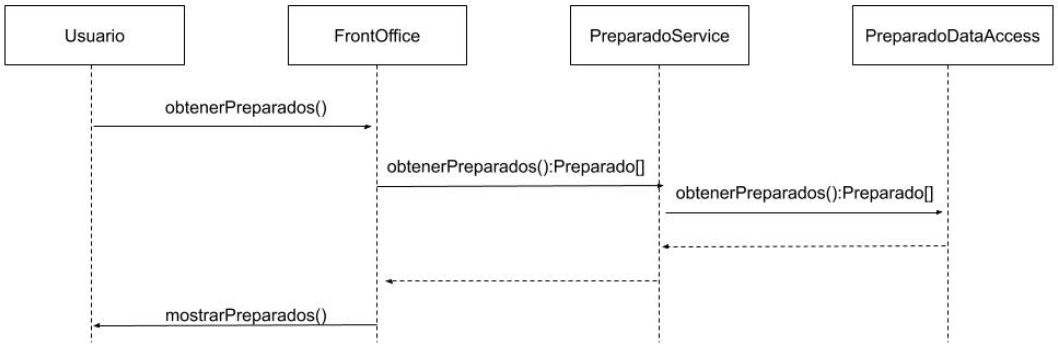
Figura 4.5: Diagrama de componentes de la capa de acceso a datos

La capa de datos, que se explica en detalle en la sección 4.2, incluye la base de datos y Elasticsearch. En la base de datos se almacena toda la información mientras que Elasticsearch se utiliza para realizar consultas de forma más eficiente con la información suficiente para realizar las consultas de búsqueda.

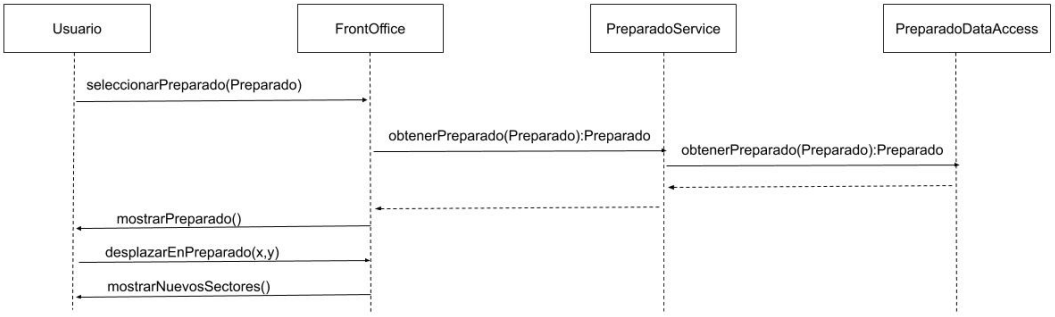
4.1.5. Diagramas de Interacción

A continuación se presentan los diagramas de interacción de los casos de uso críticos para comprender las principales interacciones entre los subsistemas.

4.1.5.1. Caso de uso obtener listados de preparados



4.1.5.2. Explorar preparados



4.1.6. Vista de Distribución

A continuación se describe cómo es el despliegue de la aplicación a partir de los componentes descritos anteriormente, donde en la figura 4.6 podemos visualizar cómo se conectan entre ellos. También se mencionan algunas de las tecnologías utilizadas para la implementación de cada nodo las cuales se profundizan en la sección 4.3. Se comienza describiendo cada uno de los nodos, luego se explica como se relacionan entre si:

- El nodo Backend contiene las componentes de la capa de servicios y de acceso a datos y se despliega en un servidor web con Node.js. Este nodo expone sus servicios mediante una API REST.
- En el cliente (Navegador) se despliegan el BackOffice y FrontOffice utilizando ReactJS.
- En el nodo FrontOffice se encuentran los html, css y javascript necesarios para realizar el *render* del FrontOffice de la aplicación en el cliente.

Además se encuentran las imágenes procesadas de los preparados para mostrar en la aplicación.

- En el nodo BackOffice se encuentran los html, css y javascript necesarios para realizar el *render* del BackOffice de la aplicación en el cliente. Además se encuentran las imágenes originales de los preparados.
- En el servidor de base de datos se despliega MongoDB.
- ElasticSearch se despliega en un servidor web el cual expone una API REST para ejecutar consultas sobre el conjunto de datos.

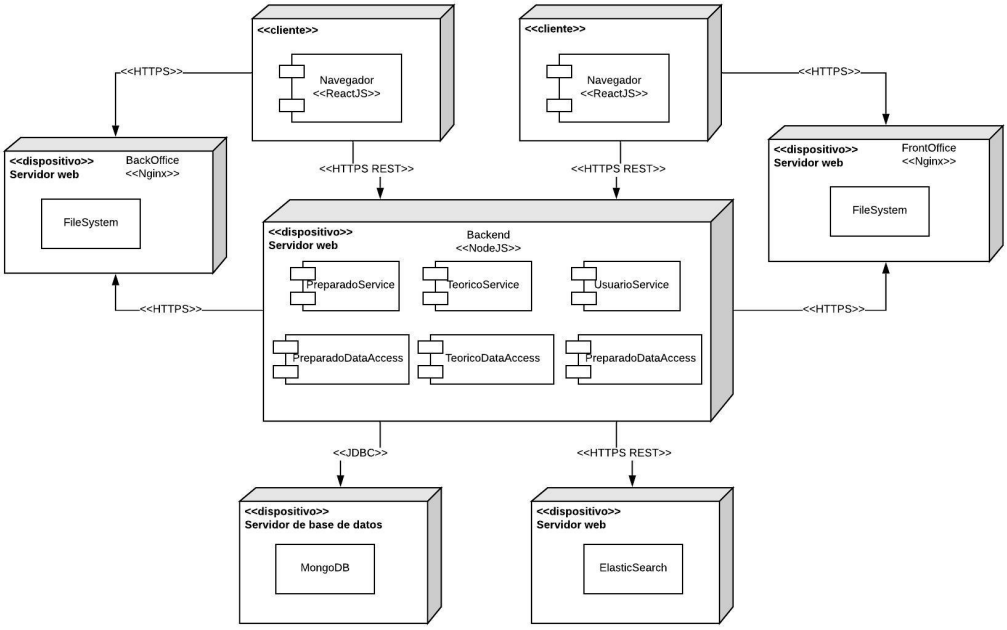


Figura 4.6: Diagrama de despliegue de las componentes

Los clientes se comunican con el nodo Backend mediante el protocolo HTTPS utilizando REST. Además tanto los clientes como el Backend se comunican por HTTPS con los nodos FrontOffice y BackOffice. Mientras que el nodo Backend se comunica con la base de datos a través de JDBC y con ElasticSearch mediante HTTPS utilizando REST. La sincronización de ElasticSearch y MongoDB está a cargo del BackEnd.

4.1.7. Escalabilidad

Una de las principales características a tener en cuenta en el desarrollo de esta aplicación es la capacidad de ser escalable. Esto se da por las carac-

terísticas de la realidad que se plantea resolver, debiendo soportar el acceso concurrente de un gran número de usuarios. Hay que tener en cuenta que habrá rangos horarios en los que puede haber picos de acceso a la aplicación de forma aleatoria, esto impacta en la definición de su escalabilidad.

Teniendo en cuenta lo anterior la aplicación debe poder escalar horizontalmente donde cada nodo escale de forma independiente. A continuación se detalla la escalabilidad para cada nodo.

4.1.7.1. FrontOffice y BackOffice

El FrontOffice y BackOffice son recursos html, css y javascript que corren exclusivamente del lado del cliente. Esto se explica por el hecho de que se elegió *Client Side Rendering* (CSR) como método de *rendering*, se profundiza sobre esto en la sección 5.1.6.

El FrontOffice y el BackOffice son independientes entre sí por lo cual pueden escalar de forma diferenciada. Esto es importante debido a que se espera un alto tráfico en el FrontOffice y no así en el BackOffice que será accedido únicamente por usuarios administradores.

En caso de ser necesario se utiliza la estrategia de escalado horizontal, lo que implica utilizar un balanceador de carga. La estrategia que se opta por utilizar es Round Robin.

4.1.7.2. Backend

Se utiliza una estrategia de escalado horizontal. Esto implica utilizar un balanceador de carga para distribuir el tráfico entre las distintas instancias. La estrategia que se opta por utilizar es Round Robin, se elige este tipo de estrategia dado que todos los *requests* tendrán un tiempo similar.

4.1.7.3. MongoDB

MongoDB provee dos formas de escalado horizontal, replicación [7] y fragmentación [9].

Replicación

La replicación consiste en replicar la base en varios servidores, donde un servidor es el primario y los restantes son secundarios. Las operaciones de escritura se realizan sobre el servidor primario y se replican sobre los secundarios. En cuanto a las operaciones de lectura, es configurable la forma en la que se

manejan. Además MongoDB provee de un método de recuperación en caso de que el servidor primario falle.

Fragmentación

La fragmentación consiste en una arquitectura que fragmenta los datos con una clave y los distribuye en varias instancias. En MongoDB la fragmentación se realiza a través de *sharded clusters*, que tiene varios servidores llamados *shard* que tienen un subconjunto de los datos fragmentados y cada uno de éstos puede ser una replicación que se comporta como se describió anteriormente.

Para la fragmentación se debe definir la clave con la que MongoDB particiona los datos [8]. La clave que se elija afecta tanto el rendimiento como la eficiencia y la escalabilidad del *sharded cluster*, por lo tanto esta elección podría afectar negativamente en su funcionamiento. Hay dos estrategias de fraccionamiento: realizando el *hash* de la clave o por rango.

Una base de datos puede tener colecciones fragmentadas y colecciones no fragmentadas, esto implica que las colecciones no fragmentadas estarán en un único *shard* que es el primario. Por lo tanto, el *shard* primario tendrá las colecciones no fragmentadas y un subconjunto de las colecciones fragmentadas.

Para esta realidad, donde no es factible que haya una cantidad tan grande de preparados que amerite utilizar la estrategia de fragmentación, replicación es la estrategia más adecuada. En el caso de llegar al punto de ser necesario escalar la base de datos, otra opción sería de incluir un cache en memoria implicando cambios en la implementación del Backend. En principio, puede que no sea necesario que la base de datos necesite escalar, pero depende del comportamiento de los usuarios.

4.1.7.4. ElasticSearch

En caso de ser necesario se utiliza la estrategia de escalado horizontal. El escalado horizontal en ElasticSearch [4] funciona de forma similiar que en MongoDB, con *shards* y *clusters*. ElasticSearch es distribuido de forma nativa, permite agregar nodos (nuevos servidores) al *cluster* y ElasticSearch se encarga de manejar estos nodos y realizar el balanceo de carga. Al distribuir los documentos de un índice en diferentes *shards* y estos *shards* en diferentes nodos ElasticSearch puede asegurar redundancia, que protege la información de fallas en el hardware y aumenta la capacidad de las consultas agregando los nodos a un *cluster*. Cuando la cantidad de nodos en el *cluster* es aumentada o

disminuida ElasticSearch automaticamente migra los shards para rebalancear el *cluster*.

4.2. Modelo de datos

Al comienzo del prototipado se utilizó un modelo de datos relacional en PostgreSQL, pero avanzado el proyecto, a partir de la asistencia de algunos integrantes del equipo de desarrollo al curso de Base de datos no relacionales dictado por Lorena Etcheverry en la Faculta de Ingeniería [1], se decidió realizar un análisis más profundo del modelo de datos a utilizar. A raíz de la necesidad de utilizar un modelo de datos que se adapte a los requerimientos de este proyecto, teniendo en cuenta la alta demanda y la necesidad de escalabilidad, se planteó realizar la comparación entre dos modelos de datos, uno relacional y otro no relacional. Hay varios modelos de datos no relacionales pero dada la naturaleza de este proyecto y basándose en el artículo [2] y documentación de MongoDB[2], se decidió utilizar un modelo de datos documental. La elección de utilizar un modelo documental se realizó debido a los siguientes puntos.

- Los datos de los preparados son muchos y variables. Actualmente las imágenes de los preparados se obtienen con un único tipo de microscopio, y al agregar imágenes con otros microscopios podrían agregarse nuevos datos a los preparados que pueden tener sentido únicamente para los preparados obtenidos con un microscopio en particular. Esto implica que los datos para cada preparado pueden ser diferentes según el microscopio con el que se obtengan las imágenes. Un modelo documental permite que los datos que no necesiten definirse puedan no estar presentes dentro del documento. Esto último implica que el desarrollador tenga en cuenta que no todos los datos están obligatoriamente en el documento del preparado.
- Las operaciones prioritarias son de consulta, ya que cuando un usuario utiliza la aplicación las consultas realizadas son de consulta sobre los preparados y sus datos. Mientras que las altas son realizadas por un administrador que no requiere que las operaciones de inserción tengan un límite de tiempo.
- La escalabilidad y el desempeño de la base de datos son importantes,

[1] https://www.fing.edu.uy/node/34576
[2] https://docs.mongodb.com/manual/core/data-models/

de modo de no generar un cuello de botella al momento de obtener los datos.

Como objetivo principal de este análisis se plantea comparar el desempeño de los dos modelos de base de datos utilizando un conjunto de consultas seleccionado, sobre un conjunto de datos de prueba generados. Sumado a esto se plantea comparar los modelos a nivel de escalabilidad. A partir de los resultados experimentales obtenidos se definió cuál de los dos modelos es el más adecuado para la realidad planteada.

A continuación se describe el diseño del modelo relacional y el documental. Seguido de esto se describe la generación de los datos de prueba, la implementación de las consultas seleccionadas para las pruebas, la configuración de los ambientes y la ejecución y resultados de las pruebas realizadas. Por último, se presenta el análisis de los resultados y las conclusiones que llevaron a tomar la decisión de utilizar el modelo de datos documental.

4.2.1. Diseño e implementación

A partir de los requerimientos de la base de datos se diseñó la base de datos relacional y la base de datos documental. El diseño de la base de datos relacional se realizó utilizando un Modelo Entidad Relación (MER), mientras que el diseño de la base de datos documental se realizó teniendo en cuenta las consultas prioritarias y no intentando hacer un equivalente de la base relacional.

4.2.1.1. Modelado de la base de datos relacional

Para el modelo de la base de datos relacional se realizó el MER (de la Figura 4.7) a partir del cual se obtuvo el modelo relacional y sus dependencias.

A continuación se listan las entidades que se obtuvieron:

- Preparado(idp, nombre, path, descripción, fecha-adq, microscopio, tejido, tinción, observaciones)
- Categoría(idc, nombre, descripción)
- Magnificación(valor, nombre, descripción)
- Profundidad(valor, descripción)
- C_C(idcp, idch)
- P_C(idp, idc)

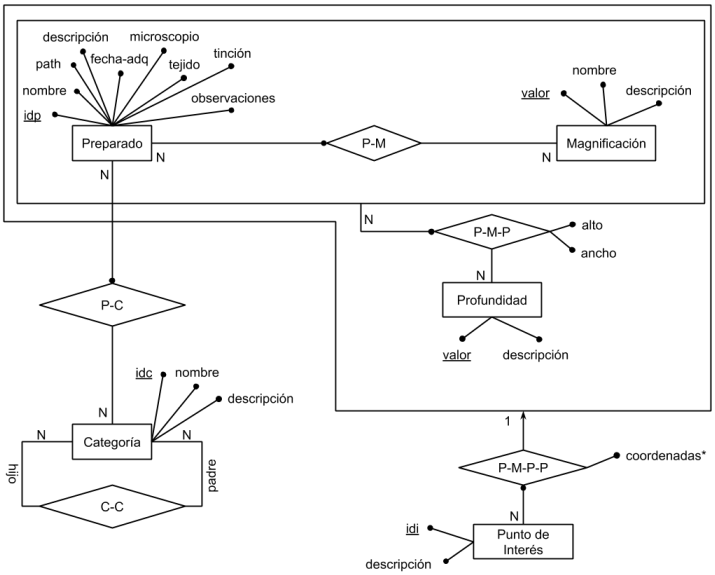


Figura 4.7: MER base relacional.

- P_M(idp, idm)
- P_M_P(idp, idm, idprof, alto, ancho)
- PuntoDeInteres(idp, idm, idprof, idi, descripción)
- Coordenadas(idp, idm, idprof, idi, x, y)

Por otro lado se obtuvieron las siguientes dependencias:

$$\Pi idcp(C\_C) \subseteq \Pi idc(Categoria)$$

$$\Pi idch(C\_C) \subseteq \Pi idc(Categoria)$$

$$\Pi idp(P\_C) \subseteq \Pi idp(Preparado)$$

$$\Pi idc(P\_C) \subseteq \Pi idc(Categoria)$$

$$\Pi idp(P\_M) \subseteq \Pi idp(Preparado)$$

$$\Pi idm(P\_M) \subseteq \Pi valor(Magnificacion)$$

$$\Pi idp, idm(P\_M\_P) \subseteq \Pi idp, idm(P\_M)$$

$$\Pi idprof(P\_M\_P) \subseteq \Pi valor(Profundidad)$$

$$\Pi idp, idm, idprof(PuntoDeInteres) \subseteq \Pi idp, idm, idprof(P\_M\_P)$$

$$\Pi idp, idm, idprof, idi(Coordenadas) \subseteq$$
$$\Pi idp, idm, idprof, idi(PuntoDeInteres)$$

4.2.1.2. Modelado de la base de datos documental

Para el modelado de la base de datos documental se agruparon los conceptos en dos grupos, por un lado el preparado con sus datos:

- Preparado
- Magnificacion
- Profundidad
- P_M
- P_M_P
- PuntoDeInteres
- Coordenadas

Por otro lado las categorías con sus hijos, padres y preparados:

- Preparado
- Categoria
- C_C
- P_C

La agregación se puede observar en la Figura 4.8, la cual queda representada con la Figura 4.9.

Esta agregación permite mejorar el desempeño de las consultas prioritarias. La consulta para obtener un preparado con todos sus datos equivale a obtener un documento de una colección, mientras que en el modelo relacional hay que realizar cruces entre varias tablas.

Las magnificaciones y profundidades están dentro del preparado para que se devuelvan en conjunto con todos los datos del preparado.

Para mejorar la consulta de categorías, las categorías se tienen en una colección separada, donde cada documento de la colección tiene todos los preparados que forman parte de esa categoría. Se debe tener en cuenta que introducir en cada categoría sus padres y sus hijos puede llevar a inconsistencias al insertar/modificar una categoría.

A partir de esas agregaciones, se definen dos colecciones, una de preparados y otra de categorías. Estas colecciones se definen con los documentos JSON

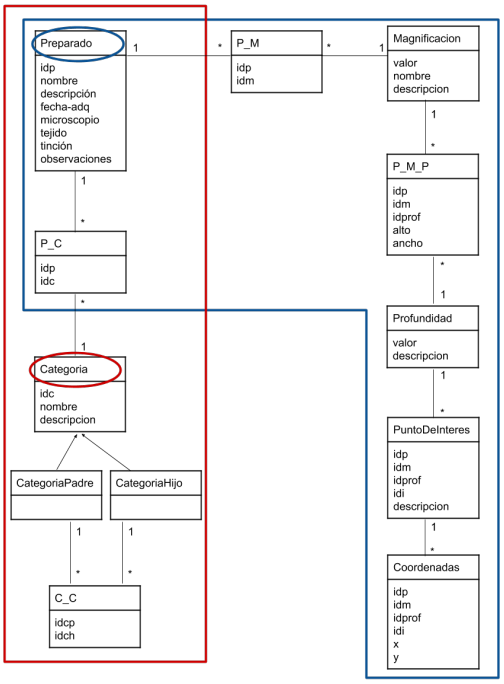


Figura 4.8: Agregaciones.

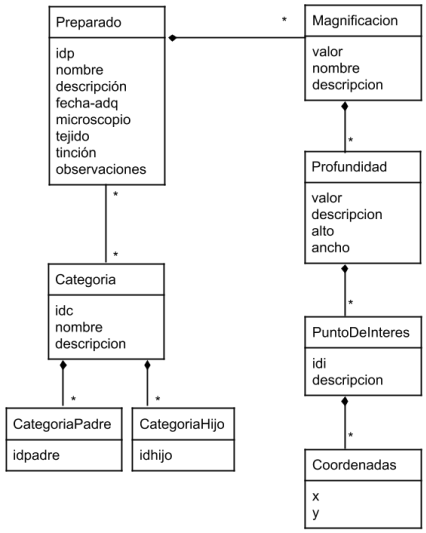


Figura 4.9: Agregaciones Final.

presentados a continuación en los Códigos 4.1 y 4.2. Para cada una de las colecciones se definieron índices, las pruebas se realizaron primero sin los índices y luego con éstos para verificar si mejoraban o no el desempeño. Estos índices se pueden encontrar luego de la definición de cada una de las colecciones.

Código 4.1 Colección de preparados

```
{
    _idp:,
    nombre:'',
    path:'',
    descripcion:'',
    fecha_adq:'',
    microscopio:'',
    tejido:'',
    tincion:'',
    observaciones:'',
    categorias:[{_idc:}],
    magnificaciones:[{
        _valor:,
        nombre:'',
        descripcion:'',
        profundidades:[{
            _valor:,
            descripcion:'',
            alto:,
            ancho:,
            puntosDeInteres:[{
                _idi:,
                descripcion:'',
                coordenadas:[{x:,y:}]
            }]
        }]
    }]
}
```

Índices:

- Índice por idp:
  db.preparados.ensureIndex({'_idp':1}, {'unique':true})
- Índice por nombre y descripción:
  db.preparados.ensureIndex({'nombre':1,'descripcion':1})

Código 4.2 Colección de categorías

```
{
    _idc:,
    nombre:'',
    descripcion:'',
    padres:[{idpadre:}],
```

```
6     hijos:[{idhijo:}],
7     preparados:[{
8         idp:,
9         nombre:,
10        descripcion:''
11    }]
12 }
```

Índices:

- Índice por idc:
  db.categorias.ensureIndex({'_idc':1}, {'unique':true}).
- Índice por nombre y descripción:
  db.categorias.ensureIndex({'nombre':1,'descripcion':1}).

4.2.2. Experimentación

Para las pruebas experimentales se utilizaron como apoyo dos artículos que analizan el desempeño de modelos de datos relacionales y no relacionales, comparándolos en base a un conjunto de datos dado.

El artículo [6] utiliza un conjunto de datos de twitters para comparar el desempeño de cuatro sistemas manejadores de bases de datos (DBMS): Microsoft SQL Server, PostgreSQL, MongoDB y Redis.

El artículo [5] utiliza un conjunto de datos con información judicial para realizar una comparación entre sistemas SQL y NoSQL al utilizar una gran cantidad de datos.

Para las pruebas se seleccionó un conjunto de consultas a realizar en las bases de datos, estas consultas se implementaron para ambos modelos sobre un conjunto de datos generado con un procedimiento implementado en Python[1] versión 3.7.4, y luego a través de JMeter[2] se realizaron las pruebas de carga. JMeter permite obtener una serie de métricas sobre las consultas realizadas, estas métricas son las que permiten la comparación entre ambos modelos. Para que la comparación sea posible se implementó cada modelo en un servidor con las mismas especificaciones.

[1] https://www.python.org/
[2] https://jmeter.apache.org/

4.2.2.1. Datos de prueba

El procedimiento implementado genera aleatoriamente los valores de los campos y guarda las consultas SQL en un archivo (para el caso de la base relacional) y los documentos JSON en otro archivo (en el caso de la base documental). Las categorías, magnificaciones y profundidades son fijas.

Para las pruebas se utilizó un conjunto de cien preparados, donde cada preparado tiene una cantidad aleatoria de categorías, magnificaciones, profundidades y puntos de interés. Acotando la generación de datos de la siguiente forma:

- Categorías por preparado: un máximo de tres
- Magnificaciones por preparado: un máximo de cuatro
- Profundidades por magnificación: un máximo de cinco
- Puntos de interés por profundidad: un máximo de veinte

4.2.2.2. Consultas de prueba

A continuación se listan las consultas utilizadas en las pruebas, con su correspondiente implementación.

- Consulta de las categorías con sus preparados con datos básicos: id, nombre y descripción.
  Para PostgreSQL esta consulta se dividió en tres:

Consulta 4.1 Las categorías con sus padres - PostgreSQL

```
(select * from Categoria) c
left join (select * from CategoriasRelaciones) r
on c.idc = r.idch;
```

Consulta 4.2 Las categorías con sus hijos - PostgreSQL

```
(select * from Categoria) c
left join (select * from CategoriasRelaciones) r
on c.idc = r.idcp;
```

Consulta 4.3 Las categorías con sus preparados - PostgreSQL

```
select Categoria.idc, Categoria.nombre, Categoria.descripcion,
PreparadoCategorias.idp, Preparado.nombre, Preparado.path, Preparado.descripcion,
Preparado.fecha_adq, Preparado.microscopio, Preparado.tejido, Preparado.tincion,
Preparado.observaciones from Categoria
left join PreparadoCategorias on Categoria.idc = PreparadoCategorias.idc
left join Preparado on Preparado.idp = PreparadoCategorias.idp;
```

Para MongoDB la consulta es:

Consulta 4.4 Las categorías con sus datos - MongoDB

```
db.getCollection('Categoria').find({})
```

- Consulta de preparados con todos sus datos incluidos las magnificaciones, profundidades y puntos de interés.
  Para PostgreSQL esta consulta es:

Consulta 4.5 Preparados con sus datos - PostgreSQL

```
select Preparado.idp, Preparado.nombre, Preparado.path, Preparado.descripcion,
Preparado.fecha_adq, Preparado.microscopio, Preparado.tejido, Preparado.tincion,
Preparado.observaciones,
PreparadoMagnificaciones.idm, Magnificacion.nombre, Magnificacion.descripcion,
PreparadoMagnificacionProfundidades.idprof, PreparadoMagnificacionProfundidades.alto,
PreparadoMagnificacionProfundidades.ancho, Profundidad.descripcion,
PuntosDeInteres.idi, PuntosDeInteres.descripcion,
Coordenadas.x, Coordenadas.y
from Preparado
left join PreparadoMagnificaciones on Preparado.idp = PreparadoMagnificaciones.idp
left join Magnificacion on Magnificacion.valor = PreparadoMagnificaciones.idm
left join PreparadoMagnificacionProfundidades
on Preparado.idp = PreparadoMagnificacionProfundidades.idp
and PreparadoMagnificaciones.idm = PreparadoMagnificacionProfundidades.idm
left join Profundidad on Profundidad.valor = PreparadoMagnificacionProfundidades.idprof
left join PuntosDeInteres on PuntosDeInteres.idp = PreparadoMagnificacionProfundidades.idp
and PuntosDeInteres.idm = PreparadoMagnificacionProfundidades.idm
and PuntosDeInteres.idprof = PreparadoMagnificacionProfundidades.idprof
left join Coordenadas on Coordenadas.idp = PuntosDeInteres.idp
and Coordenadas.idm = PuntosDeInteres.idm
and Coordenadas.idprof = PuntosDeInteres.idprof
and Coordenadas.idi = PuntosDeInteres.idi;
```

Para MongoDB esta consulta es:

Consulta 4.6 Preparados con sus datos - MongoDB

```
db.getCollection('Preparado').find({})
```

- Consulta de un preparado a través del identificador con todos sus datos incluidos las magnificaciones, profundidades y puntos de interés.
  Para PostgreSQL esta consulta es:

Consulta 4.7 Preparado 1 con sus datos - PostgreSQL

```
select Preparado.idp, Preparado.nombre, Preparado.path, Preparado.descripcion,
Preparado.fecha_adq, Preparado.microscopio, Preparado.tejido, Preparado.tincion,
Preparado.observaciones,
PreparadoMagnificaciones.idm, Magnificacion.nombre, Magnificacion.descripcion,
PreparadoMagnificacionProfundidades.idprof, PreparadoMagnificacionProfundidades.alto,
```

```
PreparadoMagnificacionProfundidades.ancho, Profundidad.descripcion,
PuntosDeInteres.idi, PuntosDeInteres.descripcion,
Coordenadas.x, Coordenadas.y
from Preparado
left join PreparadoMagnificaciones on Preparado.idp = PreparadoMagnificaciones.idp
left join Magnificacion on Magnificacion.valor = PreparadoMagnificaciones.idm
left join PreparadoMagnificacionProfundidades
on Preparado.idp = PreparadoMagnificacionProfundidades.idp
and PreparadoMagnificaciones.idm = PreparadoMagnificacionProfundidades.idm
left join Profundidad on Profundidad.valor = PreparadoMagnificacionProfundidades.idprof
left join PuntosDeInteres on PuntosDeInteres.idp = PreparadoMagnificacionProfundidades.idp
and PuntosDeInteres.idm = PreparadoMagnificacionProfundidades.idm
and PuntosDeInteres.idprof = PreparadoMagnificacionProfundidades.idprof
left join Coordenadas on Coordenadas.idp = PuntosDeInteres.idp
and Coordenadas.idm = PuntosDeInteres.idm
and Coordenadas.idprof = PuntosDeInteres.idprof
and Coordenadas.idi = PuntosDeInteres.idi
where Preparado.idp = 1;
```

Para MongoDB esta consulta es:

Consulta 4.8 Preparado 1 con sus datos - MongoDB

```
db.getCollection('Preparado').find({_idp:1})
```

- Consulta de un preparado a través del nombre y descripción con todos sus datos incluidos las magnificaciones, profundidades y puntos de interés. Para PostgreSQL esta consulta es:

Consulta 4.9 Preparado donde el nombre contiene 'Preparado 1' o la descripcion contiene 'Ipsum' - PostgreSQL

```
select Preparado.idp, Preparado.nombre, Preparado.path, Preparado.descripcion,
Preparado.fecha_adq, Preparado.microscopio, Preparado.tejido, Preparado.tincion,
Preparado.observaciones,
PreparadoMagnificaciones.idm, Magnificacion.nombre, Magnificacion.descripcion,
PreparadoMagnificacionProfundidades.idprof, PreparadoMagnificacionProfundidades.alto,
PreparadoMagnificacionProfundidades.ancho, Profundidad.descripcion,
PuntosDeInteres.idi, PuntosDeInteres.descripcion,
Coordenadas.x, Coordenadas.y
from Preparado
left join PreparadoMagnificaciones on Preparado.idp = PreparadoMagnificaciones.idp
left join Magnificacion on Magnificacion.valor = PreparadoMagnificaciones.idm
left join PreparadoMagnificacionProfundidades
on Preparado.idp = PreparadoMagnificacionProfundidades.idp
and PreparadoMagnificaciones.idm = PreparadoMagnificacionProfundidades.idm
left join Profundidad on Profundidad.valor = PreparadoMagnificacionProfundidades.idprof
left join PuntosDeInteres on PuntosDeInteres.idp = PreparadoMagnificacionProfundidades.idp
and PuntosDeInteres.idm = PreparadoMagnificacionProfundidades.idm
and PuntosDeInteres.idprof = PreparadoMagnificacionProfundidades.idprof
left join Coordenadas on Coordenadas.idp = PuntosDeInteres.idp
and Coordenadas.idm = PuntosDeInteres.idm
and Coordenadas.idprof = PuntosDeInteres.idprof
```

```
and Coordenadas.idi = PuntosDeInteres.idi
where Preparado.nombre like '%Preparado 1%' or Preparado.descripcion like '%Ipsum%';
```

Para MongoDB esta consulta es:

Consulta 4.10 Preparado donde el nombre contiene 'Preparado 1' o la descripcion contiene 'Ipsum' - MongoDB

```
db.getCollection('Preparado').find({$or:[
{'nombre': /.*Preparado 1.*/},
{'descripcion': /.*ipsum.*/}]})
```

4.2.2.3. Configuración del ambiente de pruebas

Las pruebas se realizaron en un servidor donde se levantaron dos máquinas virtuales con las mismas especificaciones.

Servidor	Máquinas virtuales
▪ Sistema operativo: Windows 10 64 bits. ▪ Procesador: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz. ▪ Disco duro: WDC WD10JPVX 1TB.	▪ Sistema operativo: Windows 10 64 bits. ▪ Tamaño disco: 127 GB. ▪ Memoria: 4 GB. ▪ Cantidad de procesadores: 2.

Tabla 4.3: Especificaciones.

En una máquina virtual se configuró el ambiente para el modelo relacional:

- Postgresql versión 11 para la base de datos.
- SQLWorkbench versión 125 para las consultas a la base de datos.
- JMeter versión 5.1.1 para las pruebas de carga.

En la otra máquina virtual se configuró el ambiente para el modelo documental:

- MongoDB versión 4 para la base de datos.
- Robo3T versión 1.3 para las consultas a la base de datos.
- JMeter versión 5.1.1 para las pruebas de carga.

Ambas máquinas virtuales son iguales para que la comparación fuera posible. Y los ambientes se configuraron en dos máquinas virtuales diferentes para que uno no afecte al otro.

4.2.2.4. Pruebas con JMeter y resultados obtenidos

Para realizar las pruebas de desempeño se utilizó JMeter, que fue instalado en ambas máquinas virtuales. Teniendo como objetivo principal simular múltiples peticiones concurrentes a cada base de datos para obtener un tiempo promedio de respuesta, se creó un plan de prueba en JMeter para cada ambiente con la siguiente configuración:

- Noventa usuarios concurrentes.
- Una única petición por usuario.
- Las peticiones se realizaran en un periodo de dos segundos.

Cada una de estas pruebas se ejecutó 5 veces.

De los datos obtenidos a partir de la ejecución de las pruebas el análisis se centró en el campo promedio. El campo promedio representa el tiempo promedio de ejecución en milisegundos (ms) de cada prueba a las noventa peticiones concurrentes de los usuarios.

En el Cuadro 4.4 se presentan los resultados obtenidos en JMeter a partir de la ejecución de las pruebas de desempeño sobre PostgreSQL y MongoDB sin índices.

Consulta	Ejec 1	Ejec 2	Ejec 3	Ejec 4	Ejec 5
	Prom	Prom	Prom	Prom	Prom
4.1 (SQL)	2960	2792	3095	3274	3053
4.2 (SQL)	2997	3409	3363	3125	3295
4.3 (SQL)	4592	3208	3076	3010	3299
4.4 (NoSQL)	1966	1228	747	638	93
4.5 (SQL)	33655	40868	30387	35858	42540
4.6 (NoSQL)	3849	3583	3583	3735	3534
4.7 (SQL)	4333	3559	3410	3116	3657
4.8 (NoSQL)	1018	497	22	26	9
4.9 (SQL)	14879	16817	14120	14113	12109
4.10 (NoSQL)	3231	3039	3110	2764	3445

Tabla 4.4: Promedios PostgreSQL vs MongoDB sin índices (tiempo en ms).

Consulta	Ejec 1	Ejec 2	Ejec 3	Ejec 4	Ejec 5
	Thro	Thro	Thro	Thro	Thro
4.1 (SQL)	16.11748	17.05191	17.00359	16.20454	16.54412
4.2 (SQL)	17.34104	15.76734	16.17251	16.51982	15.41096
4.3 (SQL)	8.90648	16.3369	16.41737	16.99717	15.89825
4.4 (NoSQL)	23.22581	27.95031	35.58719	34.92433	46.4876
4.5 (SQL)	1.57063	1.38141	1.77683	1.51375	1.34086
4.6 (NoSQL)	9.80179	9.90317	10.21566	9.86409	10.2916
4.7 (SQL)	10.83815	14.30388	15.84786	15.91793	14.67232
4.8 (NoSQL)	34.41683	40.08909	47.51848	46.24872	46.15385
4.9 (SQL)	3.09704	3.1474	3.49515	3.94875	4.02757
4.10 (NoSQL)	11.46497	11.6159	11.99201	12.46365	11.42712

Tabla 4.5: *Throughput* PostgreSQL vs MongoDB sin índices (tiempo en ms).

En los Cuadros 4.6 y 4.7 se presentan los resultados obtenidos a partir de la ejecución de las pruebas de desempeño sobre MongoDB sin índices y MongoDB con índices.

Consulta	Ejec 1	Ejec 2	Ejec 3	Ejec 4	Ejec 5
	Prom	Prom	Prom	Prom	Prom
4.4 (sin)	1966	1228	747	638	93
4.4 (con)	1193	198	83	13	13
4.6 (sin)	3849	3583	3583	3735	3534
4.6 (con)	2460	2534	2165	2017	2257
4.8 (sin)	1018	497	22	26	9
4.8 (con)	1203	84	194	17	12
4.10 (sin)	3231	3039	3110	2764	3445
4.10 (con)	1539	1704	1992	808	939

Tabla 4.6: Promedios MongoDB sin índices vs MongoDB con índices (tiempo en ms).

En los cuadros, Prom es el promedio y Thro es el *throughput* (cantidad de ejecuciones sobre tiempo total de ejecución).

En el Cuadro 4.8 se muestra el promedio para las consultas en SQL, NoSQL sin y con índices.

Para verificar los tiempos de inserción en ambos modelos se implementó una inserción para la base de datos en PostgreSQL y una inserción para la base en MongoDB. La inserción en PostgreSQL implicó realizar catorce inserciones en diferentes tablas de la base de datos. Mientras que en MongoDB la inserción implicó agregar un documento a la colección de preparados y luego para las

Consulta	Ejec 1	Ejec 2	Ejec 3	Ejec 4	Ejec 5
	Thro	Thro	Thro	Thro	Thro
4.4 (sin)	23.22581	27.95031	35.58719	34.92433	46.4876
4.4 (con)	32.75109	42.67425	43.22767	43.94531	46.82622
4.6 (sin)	9.80179	9.90317	10.21566	9.86409	10.2916
4.6 (con)	13.7216	14.34263	14.46713	15.26459	14.60091
4.8 (sin)	34.41683	40.08909	47.51848	46.24872	46.15385
4.8 (con)	31.89227	45.75496	46.41568	46.17753	46.39175
4.10 (sin)	11.46497	11.6159	11.99201	12.46365	11.42712
4.10 (con)	18.34488	18.38987	14.75652	28.23087	26.57219

Tabla 4.7: *Throughput* MongoDB sin índices vs MongoDB con índices (tiempo en ms).

Consulta	SQL	NoSQL sin índices	NoSQL con índices
Categorías con padres	3034,8	934,4	300
Categorías con hijos	3237,8	934,4	300
Categorías con preparados	3437	934,4	300
Preparados	36661,6	3656,8	2286,6
Preparado por id	3615	314,4	302
Preparado por nombre y descripción	14407,6	3117,8	1394,4

Tabla 4.8: Promedios con cincuenta preparados (tiempo en ms).

categorías del preparado realizar una actualización del documento para agregar los datos del preparado al arreglo de preparados.

Las pruebas de las inserciones también se realizaron a través de JMeter, donde la inserción se realizó cinco veces para cada base de datos. A partir de estas pruebas se obtuvieron los resultados que se muestran en el Cuadro 4.9.

Base de datos	Ejec 1	Ejec 2	Ejec 3	Ejec 4	Ejec 5	Promedio total
PostgreSQL	127	157	150	140	126	140
MongoDB	146	89	81	162	70	109,6

Tabla 4.9: Inserciones para PostgreSQL y MongoDB (tiempo en ms).

4.2.2.5. Conclusiones del modelo de datos

A partir de los resultados de las pruebas se observa que en cuanto a las operaciones de lectura, el desempeño de la implementación de la base de datos en MongoDB supera ampliamente el desempeño de la implementación en PostgreSQL. Por lo tanto, a nivel de desempeño el modelo de datos documental es el más apropiado para la realidad planteada.

En cuanto a las inserciones no hay una gran diferencia, aunque PostgreSQL debería tener un mejor desempeño seguramente esto no sucedió por la cantidad de índices que debe crear con las inserciones (por las claves foráneas).

En cuanto a los índices creados para la base de datos en MongoDB, se puede observar que influyeron positivamente en algunas de las consultas.

4.3. Tecnologías

A continuación se presentan las tecnologías utilizadas en la solución diseñada.

4.3.1. ReactJS

ReactJS es una biblioteca de Javascript para construir interfaces de usuario. En la selección de ésta tecnología se hizo una comparativa entre ReactJS, Angular y Vue para la implementación del Frontend. No se encontró una razón

técnica para elegir entre una u otra dado que las tres son una buena alternativa. Por eso, para la decisión se tuvo en cuenta la curva de aprendizaje, la popularidad de la tecnología, extensiones de desarrollo, el peso de la biblioteca y la experiencia previa de los desarrolladores. Para el análisis de ReactJS se utilizó el articulo [1].

4.3.2. Node.js

Node.js es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome. Una de las principales características de esta tecnología es el hecho de que su arquitectura está basada en eventos utilizando un único hilo para la ejecución del código de la aplicación. Esto además aporta la ventaja de utilizar los recursos muy eficientemente dado que este hilo es el único que atiende todos los *requests* al servidor. En contraposición un servidor donde por *request* utilice un nuevo hilo, éste necesitará reservar recursos lo que en este tipo de aplicación lo haría menos eficiente, además de que se agregaría complejidad a la implementación.

Basados en lo recién mencionado, se entiende que la aplicación utilizando Node.js es más económica desde el punto de vista de infraestructura dado que dos servidores con igualdad de recursos uno utilizando Node.js y el otro una tecnología multihilos, el servidor Node.js es capaz de atender una mayor cantidad de *requests*. Esta aplicación, donde cada *request* solo se encarga de operaciones de entrada y salida sin tener que ejecutar procesamientos costos, es un buen candidato para ser implementada con esta tecnología. Para el análisis de Node.js se utilizó el articulo [3].

4.3.3. Nginx

Nginx es considerado por noveno año consecutivo como el "Web Server of the Year"por W3Techs[1]. Además de este reconocimiento, dentro de las principales características se pueden destacar la velocidad, balanceador de carga integrado, capacidad de escalado, actualizaciones sin afectar disponibilidad, alto desempeño para servir contenidos estáticos y su facilidad de uso y mantenimiento. Para el análisis de Nginx se utilizó el libro [12].

[1] https://w3techs.com/blog/entry/web_technologies_of_the_year_2018

4.3.4. Leaflet

Leaflet[1] es una biblioteca javascript para mapas interactivos de código abierto. Esta biblioteca se utiliza para navegar las imágenes del preparado y permite implementar las funcionalidades de cambio de lente y cambio de foco. Sumado a esto, al ser una biblioteca javascript la integración con React es fácilmente realizable, sumado a esto parte del equipo tiene experiencia previa en la herramienta con mapas. Además esta herramienta permite la implementación de extensiones de forma sencilla con controles provistos por Leaflet[2]. Estos controles permiten implementar los diferentes botones para las funcionalidades del simulador.

4.3.5. MongoDB

Como se detalló en la sección 4.2 MongoDB se utiliza para la base de datos, la elección de esa tecnología se realizó en base a las características del proyecto y a la comparación con un modelo relacional. El modelo documental se eligió sobre otros modelos no relacionales por la estructura de los datos a guardar y las consultas prioritarias de la aplicación.

4.3.6. ElasticSearch

Elasticsearch[3] es un motor de búsqueda y análisis RESTful distribuido. Se utilizó esta tecnología para realizar consultas de cierta complejidad de manera eficiente. ElasticSearch cuenta con extensiones que permiten la sincronización con un amplio número de motores de bases de datos.

Cabe destacar que todas las tecnologías descritas anteriormente cuentan con una extensa documentación, gran comunidad y versiones gratuitas sin comprometer funcionalidades críticas.

4.4. Interfaz

El diseño de la interfaz de usuario y del flujo de la aplicación se realizó inicialmente por el equipo de desarrollo y luego en conjunto con un diseñador

[1] https://leafletjs.com/
[2] https://leafletjs.com/examples/extending/extending-3-controls.html
[3] https://www.elastic.co/es/products/elasticsearch

contratado dentro del proyecto Microscopio Mágico. El flujo de la interfaz comienza en la galería donde al usuario se le muestran los preparados donde puede realizar filtros y búsquedas. Dentro de esta pantalla puede ir a la interfaz del simulador seleccionando un preparado de la galería o a un contenido teórico a partir de las pestañas.

La interfaz de la galería tuvo dos versiones, mientras que la interfaz del simulador fue la más trabajada con el equipo de experiencia de usuario (UX) formado por Gonzalo Vilar y Gustavo Armagno del proyecto Microscopio Mágico. Del simulador se tuvieron tres versiones principales, donde la primera fue diseñada por los desarrolladores únicamente y las dos siguientes con cambios planteados por el equipo de UX, siendo la tercera un prototipo del diseño realizado por el equipo de UX.

Para llegar a la versión actual de la interfaz del simulador se utilizó como base principal el requerimiento de que la interfaz debía recordar al usuario el microscopio físico, no solo por las funcionalidades de cada botón sino por la forma de los botones y los diferentes efectos. En la ultima versión uno de los efectos más complejos es el que simula el cambio de objetivo, donde la imagen sale del circulo y vuelve a entrar con la imagen en la nueva magnificación.

Además de los requerimientos se utilizó como *input* para el diseño las actividades realizadas con los alumnos en las escuelas. Estas actividades se explican con detalle en la sección 7.3.1.

4.4.1. Interfaz de la galería

La primer versión de la galería que se puede ver en la figura 4.10 solo mostraba una lista de los preparados con la imagen a partir de la cual se accedía al simulador.

En el segundo diseño (figura 4.11) se incorporaron el buscador, los filtros por categoría y el paginado. En la lista de preparados que se muestra cada preparado tiene la imagen, el nombre y la descripción.

4.4.2. Interfaz del simulador

La primer interfaz del simulador utilizaba el tamaño por defecto del mapa de Leaflet que es rectangular, se optó por la pantalla completa para tener mejor visualización del preparado. Los tres botones de la izquierda simulaban el

Figura 4.10: Galería primer diseño

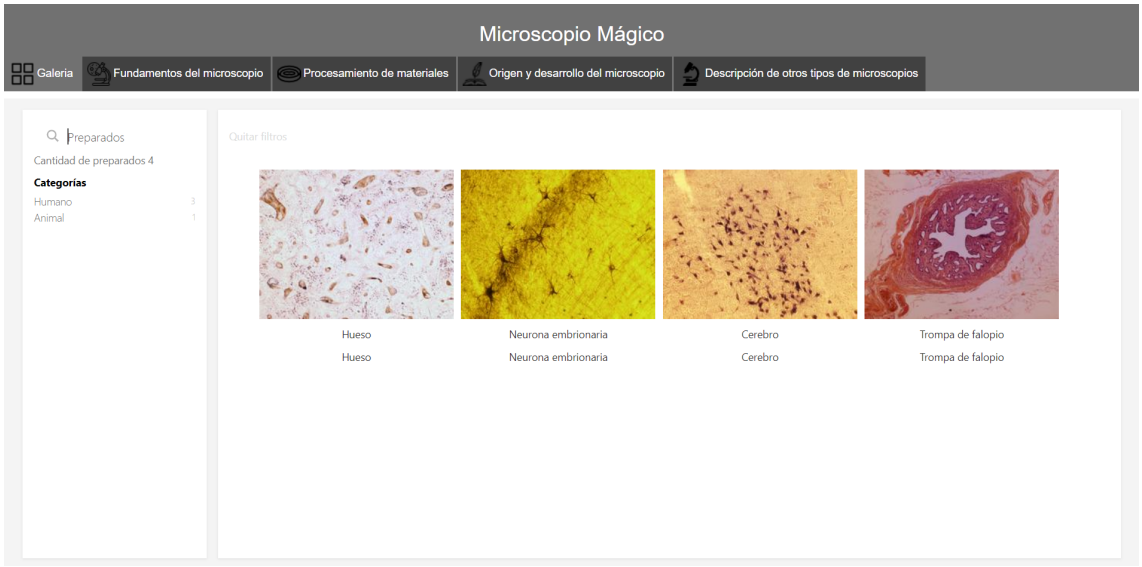


Figura 4.11: Galería segundo diseño

cambio de magnificación y el botón de la derecha el cambio en el micrométrico y macrométrico.

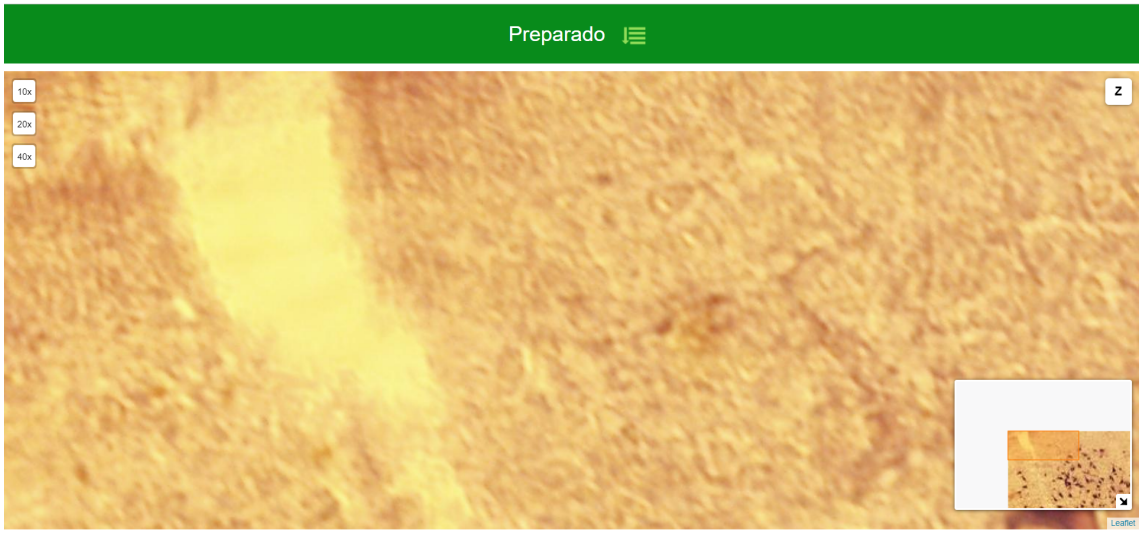


Figura 4.12: Simulador primer diseño

El principal cambio de la segunda versión es la modificación en la forma y tamaño del mapa para que simulara cómo se ve el del microscopio (circular). Además se agregaron botones de información y opciones.

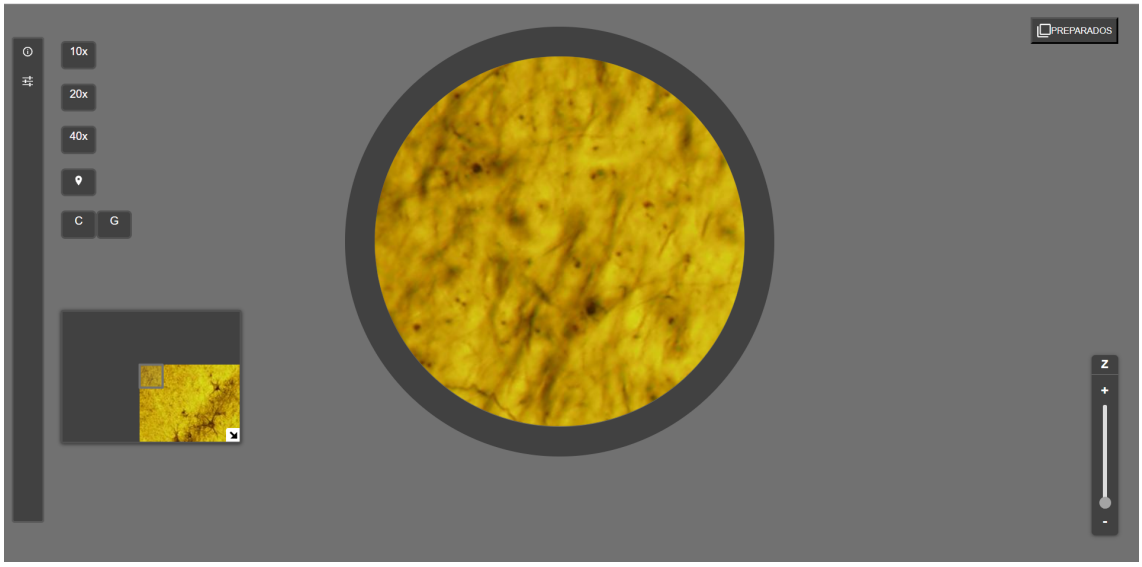


Figura 4.13: Simulador segundo diseño

En la tercer versión se modificó el diseño de todos los botones, dejando los botones de cambio de magnificación en el círculo alrededor del mapa y los botones de micrométrico y macrométrico como perillas. Además se agregó un

botón para simular la intensidad de la luz y se modificó la visualización de la información del preparado.

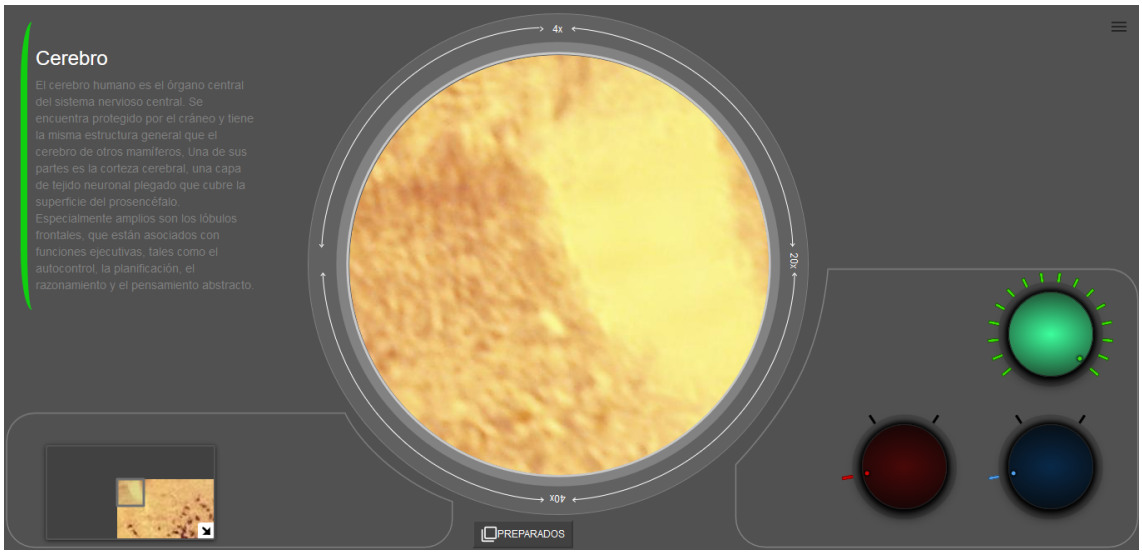


Figura 4.14: Simulador tercer diseño

Esta última versión es la utilizada para el proyecto de grado, mientras que en el proyecto Microscopio Mágico se continúa iterando sobre la interfaz para mejorar tanto la usabilidad como la experiencia de usuario.

Capítulo 5

Implementación de la solución

A partir del diseño de la solución propuesto en el capítulo anterior, se realiza la implementación teniendo en cuenta los requerimientos dentro del alcance. Para la implementación además de tener en cuenta las capas de la arquitectura, dentro de cada nodo se modularizó el código para que éste sea reutilizable, mantenible, más simple y entendible. El FrontOffice y BackOffice se dividen en dos componentes principales, el componente encargado de mostrar lo que se ve en la UI y la capa de acceso a servicios. Por otra parte, el BackEnd se dividió entre la capa de servicio y la capa de acceso a datos.

En la sección 5.1 se detalla la implementación de los nodos FrontOffice y BackOffice. En la sección 5.2 se detalla la implementación del nodo BackEnd. En la sección 5.3 se detalla la implementación de los nodos de la capa de datos con MongoDB y ElasticSearch. Por ultimo, en la sección 5.4 se describe el proceso de la adquisición de imágenes y su procesamiento.

5.1. Implementación del nodo FrontOffice y el nodo BackOffice

Como se comentó en el diseño de la solución, el FrontOffice[1] y el BackOffice[2] están implementados con ReactJS. Ambas aplicaciones se dividen en varias componentes permitiendo separar la interfaz de usuario en piezas independientes, reutilizables y pensar en cada pieza de forma aislada. Para la obtención de los datos se comunican con el BackEnd a través de la API REST.

[1] https://gitlab.fing.edu.uy/camila.rosso/micromagico_frontoffice
[2] https://gitlab.fing.edu.uy/camila.rosso/micromagico_admin

A continuación se detallan las componentes que implementan la presentación de ambas aplicaciones y distintos aspectos de su implementación como lo son la comunicación con el BackEnd y el método de *rendering*.

5.1.1. Componentes del nodo FrontOffice

Se tienen tres componentes principales: la galería, el simulador y los contenidos teóricos. Cada uno de estos a su vez utiliza componentes globales y componentes específicas. Las componentes globales son el cabezal, el pie de página y las pestañas. Mientras que las componentes específicas dependen de la componente principal, las cuales son detalladas a continuación.

5.1.1.1. Implementación de la galería

La galería es la componente donde se muestra el listado de preparados a partir del cual se puede seleccionar uno para acceder al simulador. También se permite realizar búsquedas y filtrar los preparados por sus categorías. Además de las componentes globales, la galería utiliza componentes que conforman el buscador, los filtros por categoría y la visualización de preparados.

Para la obtención y búsqueda de los preparados se utiliza ElasticSearch, realizando las consultas sobre los datos guardados dentro del motor de búsqueda. En la sección 5.3.2 se detallan los datos guardados en ElasticSearch y cómo se utilizan para la galería.

Para la galería se utilizó SearchKit[1] que es un paquete de componentes React que permiten armar un buscador con ElasticSearch. Con las componentes permite construir el árbol de categorías tomando las categorías de ElasticSearch, permite agregar un buscador pudiendo configurar qué campos debe utilizar ElasticSearch para la búsqueda y de qué forma hacerlo. También tiene componentes para distintos filtros como checkbox o por rangos, y componentes que muestran los filtros elegidos. Además la forma en que se muestra cada resultado es una componente modificable que permite personalizar completamente la visualización. Tiene una amplia documentación[2] que explica como utilizar cada una de las componentes y como incluirlas en la visualización final. En particular, para la galería se utilizaron las siguientes componentes:

[1] http://www.searchkit.co
[2] http://docs.searchkit.co/stable/

- El componente del buscador para filtrar por un texto ingresado por el usuario.
- Una componente para mostrar el árbol de categorías que permite filtrar por categoría.
- Una componente que muestra los filtros elegidos y permite quitarlos.
- Paginado, que permite obtener los resultados en cantidades controladas. ElasticSearch permite obtener los resultados por páginas, lo cual ayuda a la utilización de páginas y de obtención de resultados en pocas cantidades.
- Loading, componente personalizable que se muestra mientras se cargan los resultados.
- Error, componente personalizable que se muestra en caso de error, en ciertos casos con sugerencias de búsquedas.

5.1.1.2. Implementación del simulador

El simulador es la componente que simula el comportamiento del microscopio. Para simular la visualización y exploración del preparado se utiliza Leaflet. Esta biblioteca permite tratar a la imagen del preparado como un mapa, de esta manera se puede explorar el preparado y cambiar las magnificaciones y profundidades. Leaflet maneja el preparado como un conjunto de pequeñas imágenes que permiten cargar únicamente la parte del preparado que el usuario está visualizando a partir de las pequeñas imágenes que lo componen, en vez de cargar toda la imagen completa. Esto permite ahorrar los recursos del dispositivo y el ancho de banda.

Al utilizar Leaflet, para implementar las funcionalidades sobre la exploración del preparado se utilizaron algunas de las extensiones de Leaflet[1] y se implementaron otras. Dentro de las extensiones de Leaflet utilizadas se encuentra la usada para el minimapa[2]. Dentro de las extensiones implementadas se encuentra la de los botones que implementan las funcionalidades de cambio de profundidad y cambio de intensidad de luz. Éstas extensiones se implementan como controles[3] que son elementos HTML que pueden interactuar con el mapa.

[1] https://leafletjs.com/plugins.html
[2] https://github.com/Norkart/Leaflet-MiniMap
[3] https://leafletjs.com/examples/extending/extending-3-controls.html

5.1.1.3. Implementación de contenidos teórico

Para los contenidos teóricos se implementó la componente pestañas que muestra en pestañas los diferentes contenidos teóricos habilitados en la aplicación. Además se implementó una componente simple donde se cargan los datos del contenido teórico para mostrarlo. En caso de existir contenidos teóricos que tengan una estructura diferente debería definirse una nueva componente con esa estructura para mostrar el contenido.

5.1.2. Componentes nodo BackOffice

Dentro del BackOffice hay cuatro componentes principales: inicio de sesión, visualización de preparados, agregar preparado y agregar categoría. Las últimas tres utilizan la componente menú que permite navegar entre estas y cerrar sesión.

5.1.2.1. Inicio de sesión

Permite al usuario iniciar sesión con un usuario y contraseña que le permite acceder al resto de las pantallas. Al iniciar sesión, el BackEnd devuelve un token del usuario que permite luego consumir los servicios que permiten agregar preparados y categorías, esto se profundiza en la sección 5.2.

5.1.2.2. Visualización de preparados

Para la visualización de los preparados se utiliza la misma galería que se utiliza en el FrontOffice, de esta forma el usuario administrador que utiliza el BackOffice puede visualizar cómo quedará la galería con los preparados y categorías que ingrese.

5.1.2.3. Agregar preparado

Esta componente permite agregar un preparado con todos sus datos. Para los datos de magnificaciones y profundidades se crearon componentes separadas que se agregan a la pantalla dinámicamente según eventos que genere el usuario. Como el preparado puede tener varias categorías de diferentes ramas, para agregar las categorías del preparado se implementó un flujo que permite al usuario recorrer el árbol de categorías para agregar las categorías al preparado. El usuario selecciona de un *select* la categoría de la rama, en ese momento

se le muestra la categoría elegida y en el *select* se cargan sus categorías hijas, de esta manera el usuario va recorriendo el árbol hasta llegar a la categoría correcta.

5.1.2.4. Agregar categoría

Esta componente permite agregar una nueva categoría, donde para la selección de las categorías padre se utiliza el mismo flujo explicado anteriormente para los preparados.

5.1.2.5. Menú

Esta componente es utilizada por las tres anteriores para poder navegar entre estas. Es un menú lateral que muestra únicamente los iconos de las opciones, desplegando el texto de éstas si el usuario posiciona el cursor sobre el menú. El menú también tiene la opción de cerrar sesión que cierra la sesión localmente y en el BackEnd consumiendo un servicio.

5.1.3. Páginas

Tanto el BackOffice como el FrontOffice se implementaron siguiendo el método de Single Page Application (SPA), esto quiere decir que son aplicaciones web que utilizan una única pagina por lo que el HTML, JavaScript, CSS se carga una única vez. Esto permite navegar entre los diferentes apartados de la página, llamémoslos vistas, de forma mas fluida ya que se cuenta con el contenido precargado de antemano.

Para el manejo de las vistas que se mostraran en la navegación del usuario se utiliza la librería react-router-dom [1] la cual permite mediante el uso de rutas establecer cual es la vista que se debe mostrar al usuario.

Esto también trae como beneficio que al tener la aplicación dividida en vistas no es tan complejo el manejo de estados para lo que se debe mostrar al usuario, en contra parte si se tuviera una única vista entonces se debería manejar un estado que en el caso de esta aplicación indique si el usuario está en la vista de la galería, en la del preparado o en un contenido teórico. Otro beneficio es que mejora la navegabilidad en la aplicación, permitiendo al usuario acceder a las diferentes secciones de la aplicación más fácilmente a través

[1] https://www.npmjs.com/package/react-router-dom

de las páginas.

5.1.4. Manejo de estados

Para el manejo de estados dentro de la aplicación se utiliza Redux, que permite tener centralizado el estado actual de la aplicación. Este contenedor se utiliza para el manejo de la comunicación con los servicios utilizados para la obtención de los datos. Esto permite definir en las componentes qué información o mensaje debe mostrarse dependiendo del resultado de la comunicación.

5.1.5. Manejo de sesiones

Para el manejo de las sesiones se utiliza localStorage del navegador que permite guardar información para poder obtenerla luego. En este caso se guarda el usuario y el token, que junto con la IP con la cual se inicio sesión permite consumir luego los servicios del BackEnd. Se utiliza localStorage para almacenar información de la sesión porque Redux pierde el estado que tiene guardado cuando se recarga la pagina. De esta manera, con localStorage se puede mantener la información de la sesión hasta que el usuario cierre la sesión.

5.1.6. Client Side Rendering

El método de *rendering* Client Side Rendering (CSR) implica que el *render* de la aplicación se realiza en el navegador cliente a través de Javascript, en vez de obtenerse el HTML completo desde el servidor como con el método Server Side Rendering (SSR). Una de las principales diferencias entre ambos métodos es que en CSR se realiza más de un *request* al servidor para poder realizar el *render* de la página al inicio y debe descargar todos los Javascripts para poder generar la página, mientras que en SSR se pide una única vez todo el HTML de la página. Al obtener todo el HTML completo de la página, ésta se carga de nuevo completa, en vez de volver a cargar únicamente las partes modificadas como lo hace CSR.

En esta aplicación se debe tener en cuenta que los dispositivos donde se usa tienen recursos limitados, por lo cual se podría utilizar SSR para que la carga inicial de la página sea más rápida y requiera de menos recursos en el navegador. Por otro lado, se debe tener en cuenta que Leaflet, al ser una biblioteca javascript, funciona realizando el *render* en el cliente y no en el

servidor, esto implica que si se hace SSR se debe realizar en toda la aplicación, exceptuando la pantalla del simulador. Teniendo en cuenta que la pantalla del simulador es la principal de la aplicación y que en las pruebas experimentales con los equipos de Ceibal no se notó una desmejora en el desempeño de la aplicación, se decidió utilizar CSR.

5.1.7. Comunicación con el BackEnd

La comunicación con el BackEnd se encuentra en una componente separada, lo cual permite realizar modificaciones sobre la utilización de los servicios sin modificar la implementación de la presentación. La implementación realizada permite el manejo de errores de comunicación y permite tener en toda la aplicación el conocimiento de qué servicios están siendo consumidos y cuándo estos finalizaron tanto exitosamente como con error. En la sección 5.1.4 se detalla el manejo de estados, dentro del cual se maneja el estado de los servicios consumidos.

5.2. Implementación del nodo BackEnd

En el nodo BackEnd[1] se implementan las funcionalidades de la capa de servicios y de la capa de acceso a datos. Este nodo expone los servicios a través de una API Rest implementada en Node.js. En esta sección se listan y detallan los servicios expuestos por la API y cómo esta se comunica con la capa de datos.

5.2.1. Servicios

Los servicios se asignaron a distintas componentes según su funcionalidad, a continuación se listan las componentes y los servicios que se exponen.

- PreparadoService: servicios encargados de la gestión de los preparados. Los servicios se encuentran expuestos en /preparados. Esta componente contiene los siguientes servicios:
  - Obtener preparados: Este servicio retorna todos los preparados con sus datos del sistema. Se encuentra expuesto en /getpreparados.

[1] https://gitlab.fing.edu.uy/guzman.oholeguy/micromagico-api

- Obtener preparados habilitados: Este servicio retorna todos los preparados, con sus datos, que se encuentran habilitados en el sistema. Se encuentra expuesto en /getenabledpreparados.
- Obtener preparado: Dado un id de un preparado este servicio retorna el preparado que tiene ese id con sus datos. Se encuentra expuesto en /getpreparado/:id.
- Obtener categorías: Este servicio retorna todas las categorías del sistema con sus datos. Se encuentra expuesto en /getcategorias.
- Agregar nuevo preparado: Este servicio agrega un nuevo preparado con todos sus datos al sistema. Se encuentra expuesto en /agregar-preparado.
- Agregar nueva categoría: Este servicio agrega una nueva categoría con todos sus datos al sistema. Se encuentra expuesto en /agregar-categoria.

- TeoricoService: servicios encargados de gestionar los contenidos teóricos. Los servicios se encuentran expuestos en /teoricos. Esta componente contiene los siguientes servicios:

  - Obtener teóricos: Este servicio retorna todos los teóricos del sistema con sus datos. Se encuentra expuesto en /getteoricos.
  - Obtener teóricos habilitados: Este servicio retorna todos los teóricos, con sus datos, que se encuentran habilitados en el sistema. Se encuentra expuesto en /getenabledteoricos.

- UsuarioService: servicios encargados de la gestión de los usuarios. Los servicios se encuentran expuestos en /usuarios. Está componente contiene los siguientes servicios:

  - Iniciar sesión: Este servicio autentica al usuario permitiéndole acceder a las funcionalidades brindadas por el backoffice. Se encuentra expuesto en /login.
  - Cerrar sesión: Este servicio permite cerrar la sesión ya iniciada por el usuario. Se encuentra expuesto en /logout.
  - Chequeo de sesión: Este servicio permite, mediante el uso del token y la IP obtenidos en el inicio de sesión, validar si el usuario posee una sesión iniciada en el sistema. Se encuentra expuesto en /checksesion.

5.2.2. Acceso a datos

Cada servicio accede a los datos a través de la capa de acceso a datos. Para la implementación de la capa de acceso a datos se dispone de componentes que obtienen los datos a través de un cliente MongoDB. A continuación se listan y detallan estas componentes.

- PreparadoDataAccess: componente utilizada por PreparadoService para obtener los datos de los preparados.
- TeoricoDataAccess: componente utilizada por TeoricoService para obtener los datos de los contenidos teóricos.
- UsuarioDataAccess: componente utilizada por UsuarioService para obtener los datos de los usuarios.

5.3. Implementación de los nodos MongoDB y ElasticSearch

En esta sección se detalla la implementación del modelo de datos en MongoDB y en ElasticSearch y cómo se sincroniza el motor de base de datos y el motor de búsqueda.

5.3.1. Implementación de la base de datos en MongoDB

Para la implementación de la base de datos en MongoDB se utilizaron las colecciones *preparados* y *categorías* diseñadas en la sección 4.2 y la colección *teoricos* que contiene los datos de los contenidos teóricos a mostrar en las pestañas.

5.3.2. Implementación del buscador con ElasticSearch

Para la implementación del buscador se guardan en ElasticSearch solo los datos del preparado que son necesarios para las búsquedas, estos son el nombre, la descripción y las categorías. Además se guarda el path a la imágen principal que se muestra en la galería para no tener que obtener los datos del preparado con la API.

Para guardar los datos de los preparados en ElasticSearch se debe crear un índice con la siguiente configuración:

Código 5.3 Configuración del índice

```
1     "mappings": {
2         "_doc": {
3             "properties": {
4                 "categoriaN1":{
5                     "properties": {
6                         "name": { "type": "keyword", "index":"true" },
7                         "desc": { "type": "keyword", "index":"true" }
8                     }
9                 },
10                "categoriaN2":{
11                    "properties": {
12                        "name": { "type": "keyword", "index":"true" },
13                        "desc": { "type": "keyword", "index":"true" }
14                    }
15                },
16                "categoriaN3":{
17                    "properties": {
18                        "name": { "type": "keyword", "index":"true" },
19                        "desc": { "type": "keyword", "index":"true" }
20                    }
21                }
22            }
23        }
24    }
25 }
```

Esta configuración permite guardar las categorías de forma tal que pueda armarse luego el árbol de categorías para los filtros. A partir de esta configuración, los documentos de los preparados que se guardan en ElasticSearch se definen con el siguiente JSON:

Código 5.4 JSON de preparados

```
1     id:,
2     name:'',
3     desc:'',
4     image:'',
5     categoriaN1: {
6         name: '',
7         desc: ''
8     },
9     categoriaN2: {
10        name: '',
11        desc: ''
12    },
13    categoriaN3: {
14        name: '',
15        desc: ''
16    }
17 }
```

Dado que ElasticSearch es desplegado en un servidor a partir del instalador, éste debe ser configurado para ser accedido desde otros servidores.

5.3.3. Sincronización entre MongoDB y ElasticSearch

Para sincronizar los datos de MongoDB con ElasticSearch se optó por realizar la inserción en ambos al ser utilizado el servicio de agregar preparado. Esto es por la gran diferencia que hay entre el documento que se guarda en MongoDB y el que se guarda en ElasticSearch.

5.4. Adquisición de imágenes

La adquisición de imágenes es realizada por distintos integrantes del proyecto Microscopio Mágico que forman parte de un equipo de biólogos, biofísicos, físicos e ingenieros pertenecientes a las Facultades de Ciencias, Veterinaria, Ingeniería y Medicina de la UdelaR y al Instituto de Investigaciones Biológicas Clemente Estable.

Estas imágenes se toman de forma tal que una misma magnificación está formada por varias imágenes, por lo cual para formar la imagen final se debe realizar la unión (*stitching*) de las imágenes. Luego, para que esta imagen sea utilizada por Leaflet se debe realizar una partición de la imagen en imágenes pequeñas, este proceso es *tiling* donde cada partición se llama *tile*.

Para profundizar en el procesamiento de imágenes se realizó el curso Tutorial: Procesamiento de Imágenes y Visión Artificial con Federico Lecumberry[1].

5.4.1. Stitching

Para la implementación del *stitching* inicialmente se utilizó una implementación realizada por Adrian Rosebrock [11] que utiliza OpenCV[2] y Python para unir varias imágenes generando una nueva. Luego se utilizó una segunda implementación del mismo autor pero más reciente[10] que también utiliza OpenCV y Python para la implementación.

OpenCV (Open Source Computer Vision Library) es una biblioteca libre de visión artificial y aprendizaje automático. Esta biblioteca tiene más de 2500

[1] https://site.ieee.org/uruguay/tutorial-procesamiento-de-imagenes-y-vision-artificial/

[2] https://opencv.org/

algoritmos optimizados que incluyen algoritmos de visión artificial y aprendizaje automático. Estos algoritmos puede ser utilizados para reconocer caras, identificar objetos, realizar *stitching* de imágenes, encontrar imágenes similares dentro de una base de imágenes entre otras cosas.

A través de esta biblioteca es posible implementar un algoritmo que toma una carpeta de imágenes y las une generando una nueva. Esto lo hace comparando las imágenes y buscando puntos en común para encontrar cuáles imágenes deben unirse y de qué manera. Para poder encontrar puntos en común las imágenes deben tener un margen de superposición para que los objetos se repitan y así encontrar los puntos en común.

5.4.2. Tiling

Para realizar el *tiling* de las imágenes ya unidas se utiliza la librería de procesamiento de imágenes libvips[1][2], que permite tanto ejecutar comandos por la linea de comandos como utilizar la librería desde diferentes lenguajes. Para este proyecto se utilizaron los comandos desde la linea de comandos para partir la imagen inicial en muchas imágenes pequeñas para que sean utilizadas por Leaflet.

Esta librería también se utilizó para obtener una versión de la imagen principal de menor tamaño para utilizarla en el minimapa.

En el anexo C se encuentra el detalle de las configuraciones e implementaciones realizadas tanto para el proceso de *stitching* como el de *tiling*.

[1] https://jcupitt.github.io/libvips/API/current/
[2] https://github.com/libvips/libvips

Capítulo 6

Despliegue de la solución

En el desarrollo del proyecto el sistema se desplegó en diferentes plataformas. Inicialmente se trabajó en máquinas locales pero una vez que se necesitó utilizar la aplicación en las visitas a las escuelas esta se desplegó en Amazon Web Services (AWS)[1]. La decisión de realizar el despliegue en la nube se tomó principalmente por falta de infraestructura propia del proyecto, y se eligió en particular AWS por conocimiento previo del equipo y la posibilidad de uso limitado gratuito de la herramienta. Esto permitió realizar las primera pruebas sobre la aplicación sin costos de infraestructura.

Luego a partir de la posibilidad de un convenio de Antel con la Facultad de Ingeniería, se obtuvieron códigos promocionales que entregó Antel a la Facultad para utilizar la plataforma Mi nube Antel[2] en el curso Taller de Sistemas Empresariales (TSE), cuya responsable Laura Gózales fue quién proporcionó los códigos promocionales para este proyecto. Esto permitió utilizar los servidores con más libertad debido a los límites de la versión gratuita de AWS.

A continuación se describe el despliegue del sistema en AWS y en los servidores de Antel, comparando las ventajas y desventajas de utilizar cada uno de ellos.

6.1. Despliegue en AWS

Para el despliegue en AWS se debieron utilizar los siguientes servicios:

[1] https://aws.amazon.com/es/
[2] https://store.minubeantel.uy/

- RDS[1] para la base de datos en PostgreSQL que se utilizó inicialmente. Si fuera en MongoDB debería utilizarse DynamoDB[2].
- S3[3] para la base de imágenes.
- EC2[4] para el BackEnd en NodeJS.
- S3 para el FrontEnd en ReactJS.

También se pueden utilizar infraestructura como servicio (IaaS) pero esto no incluye todas las ventajas que proporciona AWS con plataforma como servicio (PaaS).

6.2. Despliegue en Antel

En Mi nube Antel con los códigos de promoción se contó con el plan Linux - Estándar que es un servidor Privado Virtual con

- 2 CPU de 1GHz
- 8GB de Memoria RAM
- 100GB de Disco Duro
- 1 IPv4,10 IPv6
- 200GB Tráfico Saliente
- 10GB de Almacenamiento de Respaldo
- Ancho de Banda de 10Mbps

Dado que es un servidor limpio, se deben instalar todas las herramientas necesarias:

- MongoDB
- ElasticSearch
- NodeJS
- Nginx

Luego de estas instalaciones pueden realizarse los despliegues de los distintos nodos.

[1] https://aws.amazon.com/es/rds/?nc2=h_m1
[2] https://aws.amazon.com/es/dynamodb/?nc2=h_m1
[3] https://aws.amazon.com/es/s3/?nc2=h_m1
[4] https://aws.amazon.com/es/ec2/?nc2=h_m1

Para los nodos de capa de datos deben desplegarse MongoDB y ElasticSearch, donde en el primero se crea la base de datos y las colecciones con los documentos y en el segundo se agregan los índices con los documentos.

Para el servidor de imágenes se debe desplegar Nginx y transferir las imágenes por FTP al directorio correspondiente.

Los nodos BackEnd, FrontOffice y BackOffice se pueden descargar desde gitlab o transferir por FTP. Estos tres nodos se instalan utilizando el comando de NodeJS *nmp install*. El BackEnd se despliega con el comando *node app.js* , mientras que el FrontOffice y el BackOffice se despliegan utilizando la biblioteca forever con el comando *forever start -c "npm start"./*

La versión de la aplicación entregada en este proyecto se encuentra en `http://179.27.98.29:3000/galeria/`, mientras que la ultima versión en la cual se continua trabajando se puede encontrar en `http://179.27.98.29:3001/galeria/`. Estas páginas son válidas hasta el 15/01/2020.

6.3. Comparación

6.3.1. Costo y escalabilidad

Una de las ventajas de AWS sobre el servidor de Antel es que las aplicaciones desplegadas en AWS escalan a demanda, lo cual también implica que los costos dependerán del uso de la aplicación y no es un costo fijo. Esto puede ser una ventaja si hay épocas del año en las que no se utilice la aplicación, pero una desventaja si hay picos de uso muy seguido.

En contraparte, en el servidor de Antel el escalado debe hacerse de forma manual, pero el costo es fijo, así que siempre se tiene conocimiento del monto que se debe pagar por el uso de la aplicación. De todos modos, la escalabilidad de la aplicación puede implicar utilizar más servidores de Antel, lo cual aumentaría el costo. Sumado a esto, al ser la escalabilidad manual debe definirse las situaciones en las cuales se debe aumentar la cantidad de servidores o tener siempre habilitados todos los servidores disponibles, lo cual implica que se está pagando y se están ocupando recursos que no se están utilizando.

6.3.2. Seguridad

Otra ventaja de AWS es la seguridad dado que ya tienen la seguridad implementada y en la configuración de los sitios se puede agregar restricciones por IP, etc. Mientras que en los servidores de Antel la seguridad debe implementarse manualmente configurando el Firewall y agregando otras herramientas para mejorar la seguridad.

6.3.3. Disponibilidad

AWS permite una alta disponibilidad de las aplicaciones dada su infraestructura, mientras que en Antel se tiene un servidor privado donde no existe un protocolo en caso de fallos en el servidor, por lo que no se puede asegurar la alta disponibilidad del sistema.

6.3.4. Soberanía de los datos

Uno de los temas sensibles a analizar en esta comparación es la soberanía de los datos, donde en AWS se indica que éste no utiliza los datos del cliente sin consentimiento del usuario[1], mientras que en Antel no se especifica claramente las políticas en este caso, aunque sería posible realizar algún tipo de contrato específico por este proyecto.

6.3.5. Cercanía de los servidores

Antel al ser Uruguaya tiene los Datacenters en Uruguay, mientras que AWS los más cercanos están en Brasil. Esto implica que se mejora la latencia en la descarga de las imágenes.

6.3.6. Conclusiones

En comparación AWS provee muchos más servicios y beneficios que Antel, mientras que en principio Antel la ventaja principal es en cuanto a los costos, cercanía y a la posibilidad de contratos personalizados para este proyecto.

[1] https://aws.amazon.com/es/compliance/data-privacy-faq/

Capítulo 7

Calidad del software

Este capítulo se centra en las distintas pruebas realizadas durante el desarrollo de la herramienta. En la sección 7.1 se detallan las pruebas unitarias y las pruebas end-to-end realizadas en la aplicación para validar el funcionamiento de las diferentes funcionalidades. Luego en la sección 7.2 se detallan las pruebas de rendimiento realizadas para verificar el funcionamiento de la aplicación en diferentes escenarios. Por ultimo, en la sección 7.3 se describen las pruebas de usabilidad y accesibilidad realizadas.

7.1. Pruebas unitarias y *End-to-End*

7.1.1. Pruebas unitarias

Las pruebas unitarias son necesarias para asegurar la calidad en todo desarrollo de software. Estas se utilizan para un testeo granular de la aplicación asegurando que los métodos y funciones implementados tengan el comportamiento deseado. Esto significa que dado un conjunto de parámetros de entrada se debe de esperar siempre el mismo resultado.

La utilización de este tipo de pruebas son una buena práctica en el modelo de desarrollo utilizando Integración Continua. Estas pruebas permiten detectar rápidamente fallos en los nuevos cambios que se introducen, impidiendo integrar estos cambios hasta haberse corregido.

En este proyecto se utilizó Jest y Enzyme para realizar las pruebas unitarias del Front-End. Jest permite realizar pruebas que validen las salidas de las funciones dado un conjunto de entradas. Por otra parte, Enzyme se utilizó para las pruebas de Snapshots, Enzyme permite simular el HTML a ser desplegado

por los componentes React y es una forma de validar que los cambios que se hagan no tengan cambios indeseados en lo que se desplegará al usuario.

7.1.2. End-to-End

Para poder probar los diferentes flujos del FrontOffice se realizaron pruebas *End-to-End* con Nightwatch.js[1]. Nightwatch.js es una herramienta implementada en Node.js que permite realizar pruebas *End-to-End* en aplicaciones web de forma tal que se pueden realizar pruebas replicando el flujo que realizaría un usuario, por ejemplo presionando un botón o una imagen. Además permite que una vez terminado el flujo se verifique que se obtuvo la respuesta esperada, por ejemplo que cierto elemento fuera cargado en la pantalla. Para las pruebas se tuvieron en cuenta los flujos críticos de la aplicación:

- Ingresar a la galería y seleccionar un preparado.
- Ingresar a la pantalla principal y seleccionar un contenido teórico.

Las pruebas se ejecutaron tanto en Chrome como en Firefox, donde agregar un nuevo navegador implica únicamente realizar una configuración en la herramienta.

Las pruebas se ejecutan por linea de comandos donde al ejecutar cada prueba indica el resultado en cada uno de los navegadores configurados y el tiempo de ejecución de la prueba.

7.1.2.1. Seleccionar preparado

En esta prueba se ingresa a la página de la galería y a partir del listado de preparados se selecciona uno y se ingresa a la página de la visualización del preparado. Una vez se ingresa en la página de la galería se verifica que se haya cargado el listado de preparados y una vez en la página de visualización se verifica que haya cargado el mapa.

Código 7.5 Prueba selección de preparado

```
1 firefox ============================================
2 firefox Results for: 1: navegar a la pagina de microscopio magico
3 firefox  Element <body> was visible after 70 milliseconds.
4 firefox  [Unittests\seleccionar Preparado] 1: navegar a la pagina de microscopio magico
      (1.564s)
5 firefox Results for: 2: acceder a un preparado
```

[1] https://nightwatchjs.org/

```
6  firefox  Testing if element <div[id="map"]> is present - 8 ms.
7  firefox  [Unittests\seleccionar Preparado] 2: acceder a un preparado (4.538s)
8  chrome [Unittests\seleccionar Preparado] Test Suite
9  chrome ========================================
10 chrome Results for: 1: navegar a la pagina de microscopio magico
11 chrome  Element <body> was visible after 51 milliseconds.
12 chrome  [Unittests\seleccionar Preparado] 1: navegar a la pagina de microscopio magico
       (1.76s)
13 chrome Results for: 2: acceder a un preparado
14 chrome  Testing if element <div[id="map"]> is present - 25 ms.
15 chrome  [Unittests\seleccionar Preparado] 2: acceder a un preparado (3.404s)
16
17 OK. 4 total assertions passed. (11.668s)
```

7.1.2.2. Seleccionar contenido teórico

En esta prueba se ingresa a la página de la galería y a partir de las pestañas de contenidos teóricos se selecciona uno y se ingresa a la página del contenido. Una vez se ingresa en la página de la galería se verifica que hayan cargado correctamente las pestañas y al ingresar a la página del contenido se verifica que se haya cargado el contenido.

Código 7.6 Prueba selección de contenido teórico

```
1  firefox ========================================
2  firefox Results for: 1: navegar a la pagina de microscopio magico
3  firefox  Element <body> was visible after 78 milliseconds.
4  firefox  [Unittests\seleccionar Teorico] 1: navegar a la pagina de microscopio magico
       (1.892s)
5  firefox Results for: 2: acceder a un contenido teorico
6  firefox  Testing if element <h1> is present - 11 ms.
7  firefox  [Unittests\seleccionar Teorico] 2: acceder a un contenido teorico (6.948s)
8  chrome [Unittests\seleccionar Teorico] Test Suite
9  chrome ========================================
10 chrome Results for: 1: navegar a la pagina de microscopio magico
11 chrome  Element <body> was visible after 37 milliseconds.
12 chrome  [Unittests\seleccionar Teorico] 1: navegar a la pagina de microscopio magico (1.73
       s)
13 chrome Results for: 2: acceder a un contenido teorico
14 chrome  Testing if element <h1> is present - 18 ms.
15 chrome  [Unittests\seleccionar Teorico] 2: acceder a un contenido teorico (5.565s)
16
17 OK. 4 total assertions passed. (14.259s)
```

7.2. Pruebas de rendimiento

Para las pruebas de rendimiento se realizaron pruebas de carga y estrés para las cuales se utilizó la herramienta JMeter[1]. Se realizaron los dos tipos de pruebas tanto sobre la API como sobre el servidor de imágenes con Nginx. Además se realizaron pruebas de rendimiento del el FrontOffice, para obtener una métrica del tiempo de carga del FrontOffice.

Las pruebas se realizaron contra un servidor en Mi nube Antel con el plan Linux - Estándar que es un servidor Privado Virtual con

- 2 CPU de 1GHz
- 8GB de Memoria RAM
- 100GB de Disco Duro
- 1 IPv4,10 IPv6
- 200GB Tráfico Saliente
- 10GB de Almacenamiento de Respaldo
- Ancho de Banda de 10Mbps

7.2.1. FrontOffice

Dado que el *render* del FrontOffice se realiza del lado del cliente, únicamente se realizaron pruebas de rendimiento visualizando en el navegador el tiempo de carga de la página. Las pruebas se realizaron en diferentes dispositivos y en diferentes navegadores. Para estas pruebas se tuvieron en cuenta dos casos de prueba, el acceso a la página de la galería y el acceso a la página del simulador.

7.2.2. API

Para las pruebas sobre la API se realizaron pruebas sobre cada uno de los servicios provistos por esta. Las pruebas se realizaron a través de JMeter utilizando diferente cantidad de usuarios según el tipo de prueba. A continuación se detallan las configuraciones y resultados para los dos tipos de prueba: carga y estrés.

[1] https://jmeter.apache.org/

Caso	Dispositivo	Navegador	Tiempo carga
Galería	Sistema operativo: Windows 10 64 bits, Procesador: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, Disco duro: WDC WD10JPVX 1TB.	Chrome V76	DOMContentLoaded: 1,94 s Load: 2,66 s
Galería	Sistema operativo: Windows 10 64 bits, Procesador: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, Disco duro: WDC WD10JPVX 1TB.	Firefox V67	DOMContentLoaded: 573 ms Load: 600 ms
Simulador	Sistema operativo: Windows 10 64 bits, Procesador: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, Disco duro: WDC WD10JPVX 1TB.	Chrome V76	DOMContentLoaded: 1,76 s Load: 1,78 s
Simulador	Sistema operativo: Windows 10 64 bits, Procesador: Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz, Disco duro: WDC WD10JPVX 1TB.	Firefox V67	DOMContentLoaded: 563 ms Load: 569 ms

Tabla 7.1: Pruebas rendimiento FrontOffice.

7.2.2.1. Pruebas de carga

Este tipo de pruebas de rendimiento se utiliza para observar y analizar el comportamiento de la aplicación ante diferentes conjuntos de peticiones esperadas. En este caso, cómo no se conoce como se comportará a futuro el uso de la aplicación, se tomaron en cuenta casos bastante variados. Primero se tiene en cuenta el caso de que una clase completa utilice la aplicación, este es el caso básico. La siguiente prueba se realizó para seis clases, ya que la aplicación está destinada principalmente para cuarto, quinto y sexto de escuela (asumiendo que una escuela tiene dos grupos por año). Por último, se probó con un conjunto más amplio de clases con un total de veinte clases.

Asumiendo que una clase tiene aproximadamente treinta alumnos las pruebas se realizaron con treinta, 180 y seiscientos usuarios concurrentes donde cada usuario realiza una única petición y donde las peticiones se realizan en un período de cinco segundos. Cada prueba sobre cada servicio se realizó un total de cinco veces para poder obtener un promedio confiable.

Los resultados se pueden observar en las tablas 7.2, 7.3 y 7.4, donde de los datos obtenidos de la ejecución de las pruebas en JMeter el análisis se centró

en el campo promedio, el cual representa el tiempo promedio de ejecución en milisegundos de las solicitudes.

Servicio	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Promedio
getpreparados	238	34	74	55	41	88,4
getenabledpreparados	55	40	48	36	38	43,4
getpreparado	60	49	38	37	35	43,8
getcategorias	29	31	32	40	79	42,2
getteoricos	64	51	43	60	50	53,6
getenabledteoricos	48	38	43	42	40	42,2

Tabla 7.2: Pruebas de carga con treinta usuarios (una clase)

Servicio	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Promedio
getpreparados	41	31	51	32	37	38,2
getenabledpreparados	40	38	39	42	53	42,4
getpreparado	93	189	119	147	159	141,4
getcategorias	162	171	177	168	122	160
getteoricos	175	265	113	142	389	216,8
getenabledteoricos	621	144	258	193	160	275,2

Tabla 7.3: Pruebas de carga con 180 usuarios (seis clases)

Servicio	Prueba 1	Prueba 2	Prueba 3	Prueba 4	Prueba 5	Promedio
getpreparados	677	1096	1302	2877	766	1343,6
getenabledpreparados	1056	68	700	155	237	443,2
getpreparado	41	41	133	41	104	72
getcategorias	43	72	112	170	171	113,6
getteoricos	319	721	440	674	493	529,4
getenabledteoricos	600	2362	2328	1060	428	1355,6

Tabla 7.4: Pruebas de carga con seiscientos usuarios (veinte clases)

7.2.2.2. Pruebas de estrés

Las pruebas de estrés sirven para conocer el comportamiento de la aplicación en casos extremos donde la carga sobre la aplicación sea mayor a la esperada. De esta forma se puede tener un estimativo de la cantidad de usuarios máxima que la aplicación puede atender con los recursos utilizados. A partir de esto pueden tomarse medidas de escalabilidad en caso de ser necesario.

Este tipo de pruebas se deben hacer igual que las pruebas de carga pero aumentando la cantidad de usuarios concurrentes hasta que la aplicación

comience a funcionar incorrectamente o falle. En este caso las pruebas se realizaron únicamente sobre los servicios getpreparados y getpreparado que son los más críticos.

A continuación en la tabla 7.5 pueden verse los resultados de estas pruebas, donde para las distintas cantidades de usuarios concurrentes se puede ver el porcentaje de errores que se obtuvo.

Servicio	1200	1800	2400	3000	3600	4200
getpreparados	0	0	0	10,43	–	–
getpreparado	0	0	0	0	0	19,12

Tabla 7.5: Pruebas de estrés

A partir de estos resultados se puede ver que hasta 2400 usuarios la obtención de preparados funciona correctamente, pero luego de los 3000 comienza a fallar por la sobrecarga. En cuento a la obtención de un preparado, este servicio funciona con una mayor cantidad de usuarios concurrentes comenzando a fallar a partir de 4200 usuarios concurrentes. A partir de estos datos pueden definirse medidas de escalabilidad utilizando los métodos descriptos en la sección 4.1.

7.2.3. Nginx

Sobre Nginx se realizaron pruebas con diferentes cantidades de usuarios concurrentes donde cada usuario realiza cuatro *requests*, esto debido a que el mapa que se muestra se compone de cuatro imágenes.

La selección de la cantidad de usuarios para las pruebas se realizó de forma similar a las pruebas sobre la API, primero se realizaron pruebas sobre conjuntos esperados de usuarios (treinta usuarios, 180 usuarios y seiscientos) y luego se continuó aumentando la cantidad de usuarios hasta que comenzó a fallar.

A continuación en la tabla 7.6 pueden verse los resultados de estas pruebas, donde para las distintas cantidades de usuarios concurrentes se puede ver el promedio de tiempo en milisegundos y el porcentaje de errores que se obtuvo.

Usuarios concurrentes	Cantidad de requests	Promedio	% Error
30	120	33	0
180	720	26	0
600	2400	415	0
1200	4800	1710	0
2400	9600	3762	3,08
3600	14400	5993	9,56
4800	19200	8809	29,05

Tabla 7.6: Pruebas sobre Nginx (tiempo en ms)

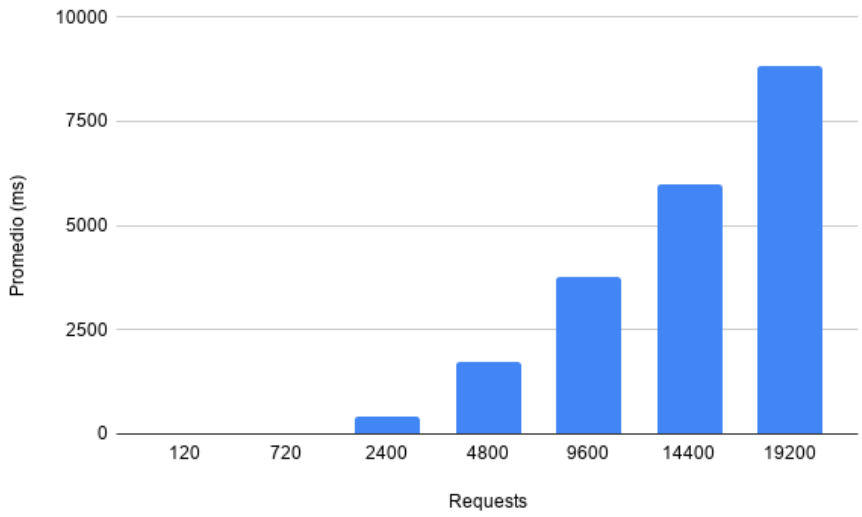


Figura 7.1: Gráfico de resultados de pruebas sobre Nginx (requests en 5 s)

En los resultados se puede visualizar que al aumentar la cantidad de usuarios aumenta rápidamente el tiempo que demora en responder cada *request*. A partir de estos datos se puede definir una estrategia de escalado en base a la cantidad de *requests* estableciendo un tiempo máximo de respuesta, de forma tal que la experiencia del usuario no se vea comprometida.

7.3. Pruebas de usabilidad y accesibilidad

7.3.1. Usabilidad y experiencia de usuario

Durante la implementación de la aplicación se realizaron diversas visitas a escuelas en diferentes etapas de la implementación. Al comienzo se realizaron

visitas a escuelas dentro de Montevideo y ya avanzado el proyecto se realizaron visitas a escuelas de Tacuarembó en dos instancias diferentes. Algunas de las visitas fueron las siguientes:

- Escuela Nº 262 en Bella Italia, Montevideo el 29/11/2018 con alumnos de sexto año.
- Escuela Nº 50 en Colón, Montevideo el 29/11/2018 con alumnos de quinto año.
- Escuela Nº 48 La Blanqueada, Montevideo Diciembre 2018.
- Escuela rural Nº 107 en Los Furtado, Tacuarembó el 12/04/2019 con aproximadamente diez alumnos de edades diversas desde primero de escuela a sexto.
- Escuela de tiempo completo Nº 13 de la ciudad de Tacuarembó el 10/06/2019 con aproximadamente treinta alumnos de diferentes años entre cuarto, quinto y sexto.
- Escuela de tiempo completo Nº 9 en Curtina, Tacuarembó el 10/06/2019 con alumnos de diferentes años entre cuarto, quinto y sexto.
- Actividad el 05/10/2019 por el día del patrimonio en el Instituto de Investigaciones Biológicas Clemente Estable. Esta actividad fue similar a las visitas a las escuelas pero era abierta por lo cual también participaron alumnos de escuelas privadas y sus padres.

Las actividades realizadas al comienzo implicaron mostrar a los usuarios (alumnos y maestros) la aplicación para que la usaran, se les permitía navegar por la aplicación sin indicarles su funcionamiento previamente. A partir de esta actividad se lograron obtener varios puntos de mejora en la interfaz de usuario y se logró tener una mejor comprensión de la forma de utilizar la aplicación. Para mejorar la actividad y la retroalimentación, se incorporó a esta actividad la creación de una interfaz donde los alumnos podían dibujar cómo se imaginaban que podía verse la aplicación, enfocándose en la pantalla del simulador. Teniendo en cuenta el desconocimiento de varios niños sobre lo que significaba realizar una interfaz de usuario, se mejoró la actividad incluyendo distintos componentes ya impresos para que los alumnos pudieran crear la interfaz realizando un rompecabezas. Esta mejora permitió que se obtuviera la opinión de varios alumnos, a partir de las cuales se pudo realizar un mejor análisis de los requerimientos que debía cumplir la interfaz para mejorar la usabilidad.

En la carpeta compartida[1] se pueden encontrar algunos de los trabajos realizados por los alumnos en las actividades.

7.3.2. Accesibilidad

Para las pruebas de accesibilidad se realizaron evaluaciones automáticas mediante el uso de herramientas disponibles en la web. Tomando como referencia la guía propuesta de AGESIC [2] las pruebas realizadas fueron las siguientes:

- Validar Gramática (X)HTML
- Validar Hojas de Estilo en Cascada (CSS)

7.3.2.1. Validar Gramática (X)HTML

La validación de estándares (X)HTML es recomendada en la Pauta 4.1 de accesibilidad de la WCAG2.0. Es más fácil realizar correcciones y modificaciones a un código bien formado y prolijo, por lo que se recomienda comenzar con esta validación. Validar el código de las páginas garantiza la portabilidad a distintos dispositivos y mejora la indexación en buscadores.

World Wide Web Consortium (W3C) dispone de un servicio en línea [3] para validar la gramática del código (X)HTML de documentos web.

Los resultados obtenidos en una primera validación se pueden observar en la imagen 7.2 donde se muestran cuatro advertencias y un error.

En la segunda iteración 7.3 se corrige el error y queda únicamente pendiente una advertencia debido a que la dependencia se genera dinámicamente por React.

7.3.2.2. Validar Hojas de Estilo en Cascada (CSS)

La validación de hojas de estilo contribuirá a mejorar la mantenibilidad del código además de garantizar la portabilidad, ya que evitará que se use código no estándar propietario de algún navegador que no funcione en otro navegador. W3C dispone del servicio en línea [4] para validar la gramática del código de las Hojas de Estilo en Cascada (CSS).

[1] https://drive.google.com/drive/folders/1P4HeeBEruit2MBqeiE7EwZ-2sj0EIMDb?usp=sharing

[2] https://www.gub.uy/agencia-gobierno-electronico-sociedad-informacion-conocimiento/comunicacion/publicaciones/evaluacion-de-accesibilidad

[3] https://validator.w3.org/

[4] http://jigsaw.w3.org/css-validator/

Figura 7.2: Prueba de accesibilidad (X)HTML primera iteración

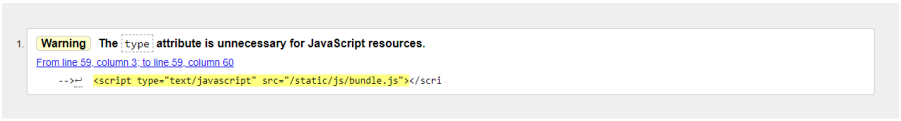


Figura 7.3: Prueba de accesibilidad (X)HTML primera iteración

Los resultados obtenidos en una primera validación se pueden observar en la imagen 7.4 cinco errores.

Figura 7.4: Prueba de accesibilidad CSS

Estos errores son introducidos por el uso de las librerías de Leaflet y Searchkit.

En el caso del error introducido por Leaflet, el cual señala que la propiedad *behavior* no existe, se debe únicamente a que *behavior* es un prefijo propietario de Microsoft por lo que no esta incluido dentro del estándar de la W3C. El uso de prefijos propietarios en el CSS es una practica común en la elaboración de los estilos y no afectará de forma negativa al sitio.

En el caso de la librería SearchKit los estilos que se utilizan actualmente son los proporcionados por la herramienta. Como trabajo a futuro se plantea realizar un diseño propio de la galería, el cual seria implementado por los desarrolladores subsanando los errores introducidos por la librería.

Capítulo 8

Gestión

En este capítulo se describe la gestión del proyecto centrándose en la metodología de trabajo en la sección 8.1, la comunicación entre los diferentes equipos dentro del proyecto en la sección 8.2 y las herramientas utilizadas durante la realización del proyecto en la sección 8.3.

8.1. Metodología de trabajo

Una vez realizado un relevamiento de requerimientos inicial, se observó que estos no estaban del bien definidos por lo que se decidió comenzar realizando un prototipo rápido con las funcionalidades detectadas como críticas. Esto tuvo como principal objetivo entender los requerimientos planteados y trabajar para mejorar la calidad de estos. Este prototipo ayudó a terminar de definir los requerimientos y sirvió como base (prototipo evolutivo) para incorporar nuevas funcionalidades, efectos solicitados y aplicar el diseño una vez que se contó con un diseñador en el equipo.

La incorporación de nuevas funcionalidades y del diseño se realizo de forma iterativa e incremental, donde en cada iteración se incorporan elementos de diseño y se agregan, mejoran y corrigen funcionalidades. Esto permitió contar con versiones utilizables por los usuarios sin necesidad de tener todo el producto desarrollado, siendo esencial para sacar provecho en las visitas a las escuelas y realizar pruebas desde el principio del desarrollo.

8.2. Comunicación

Los medios de comunicación utilizados durante el proyecto fueron diferentes según los grupos involucrados.

8.2.1. Integrantes de todo el proyecto

Con los integrantes de todo el proyecto se utilizó principalmente el correo electrónico para definir reuniones o actividades en las escuelas.

Para el relevamiento de requerimientos y la toma de decisiones se realizaron reuniones semanalmente durante un tiempo prolongado, esto permitió tener una constancia dentro del equipo, pero dada la cantidad de gente que asistía y la variedad de disciplinas y opiniones se dificultaba la toma de decisiones.

8.2.2. Integrantes del grupo de la Facultad de Ingeniería

En cuanto al grupo de Facultad de Ingeniería, se realizaron reuniones principalmente al comienzo del proyecto donde se realizaron las decisiones de arquitectura y metodología de trabajo. Luego al tener las reuniones con el equipo completo, las reuniones especificas fueron menos frecuentes.

8.2.3. Equipo de desarrollo

El equipo de desarrollo realizó reuniones semanales, incluso varias veces en una misma semana si se requería, para toma de decisiones, análisis de requerimientos, diseño de la solución y planeamiento de las tareas para la iteración. En las etapas de implementación las reuniones no se centraron en implementación, sino en planeamiento, diseño y división de tareas, dado que el proyecto incluye varias componentes esto facilitó la división de tareas.

8.2.4. Equipo de desarrollo en conjunto con diseño

Una vez incorporado al proyecto el equipo de diseño se comenzó con el intercambio, que inicialmente fue menos frecuente dado que se debió realizar un análisis del producto para realizar el *branding* y así continuar con el diseño. Más avanzado el diseño, las reuniones de coordinación se realizaron más frecuentemente.

8.3. Herramientas

Para el desarrollo del sistema se utilizaron varias herramientas para diferentes fines y como soporte a las actividades del equipo de desarrollo. Dentro de estas herramientas se encuentran Trello[1], GIT[2] y Google Drive.

8.3.1. Trello

Se utilizó Trello como herramienta para definir tareas, por un lado para el equipo de desarrollo y por otro lado en conjunto con el equipo de diseño, ya que en las últimas etapas del desarrollo se interactuó con mayor fluidez con el equipo de diseño por el enfoque en el diseño y experiencia de usuario del FrontOffice.

8.3.2. GIT

La herramienta GIT se utilizó como repositorio para el código de las distintas aplicaciones dentro del sistema: FrontOffice, BackOffice y BackEnd.

8.3.3. Repositorio de documentos

Como repositorio de documentos se utilizó Google Drive tanto dentro del equipo de desarrollo como dentro del equipo completo del proyecto. Dentro del equipo de desarrollo se utilizó para almacenar la documentación e información sobre el estado de las aplicaciones para complementar la herramienta Trello. Dentro del equipo del proyecto se utilizó como medio para intercambiar las imágenes entre los equipos que las tomaban y el equipo de desarrollo.

[1] https://trello.com
[2] https://gitlab.fing.edu.uy

Capítulo 9

Resultados y trabajo a futuro

9.1. Resultados

A partir de la ejecución del proyecto se fueron obteniendo diversos resultados, donde el resultado principal es la construcción del simulador del microscopio. La aplicación completa se compone por el banco de imágenes de los preparados, la base de datos con los metadatos de las imágenes, la API para acceder a las imágenes y sus metadatos, el administrador para agregar los preparados y el simulador del microscopio.

Dentro de los distintos componentes de la aplicación se obtuvieron los siguientes resultados:

- Un banco de imágenes con preparados.
- Una base de datos con todos los metadatos necesarios para esos preparados que permiten implementar las distintas funcionalidades dentro del simulador.
- Una API con todos los servicios de consulta para el simulador y para la utilización por terceros, así como servicios para el administrador. La implementación de la API se encuentra disponible en GitLab[1].
- Un administrador para ingresar los metadatos de los preparados. Esta versión no incluye la funcionalidad de subir las imágenes y el post-procesamiento de imágenes, esto debe hacerse previamente. La implementación del administrador se encuentra disponible en GitLab[2].

[1] https://gitlab.fing.edu.uy/guzman.oholeguy/micromagico-api
[2] https://gitlab.fing.edu.uy/camila.rosso/micromagico_admin

- Para el post-procesamiento de las imágenes se implementaron procesos en Python que realizan los distintos procesamientos necesarios para la utilización de las imágenes.
- El simulador del microscopio con exploración de preparado, cambio de magnificaciones, cambio de profundidades, cambio en la intensidad de luz y minimapa. La implementación del simulador se encuentra disponible en GitLab[1].

Dentro de la construcción de la aplicación, además de la construcción de los distintos componentes, se obtuvieron varios resultados parciales y finales:

- Prototipos evolutivos
- Diseño de la arquitectura de la aplicación
- Análisis de la escalabilidad de la aplicación
- Diseño de la interfaz del simulador
- Despliegue de la aplicación tanto en AWS como en Antel
- Pruebas de rendimiento de la API y la base de datos
- Pruebas de accesibilidad y usabilidad del simulador
- Documentación de usuario del administrador
- Documentación técnica de la API
- Documentación técnica del procesamiento de las imágenes de los preparados

Además, dentro del proyecto Microscopio Mágico se pueden destacar varios resultados que aportaron a la difusión del proyecto y al involucramiento de los participantes del proyecto:

- Vistas a escuelas de diferentes contextos.
- Participación en el IV Congreso Internacional de Enseñanza de las Ciencias Básicas el 10 de Octubre de 2019 en Paysandú[2].
- Entrevista en la radio El Espectador que se puede encontrar en su página[3].
- Entrevista en Cambiando el aire de Canal 5 el 01/11/2019.

[1] https://gitlab.fing.edu.uy/camila.rosso/micromagico_frontoffice
[2] http://cieciba.multisitio.interior.edu.uy/presentacion-de-trabajos/
[3] https://espectador.com/mastemprano/entrevista/microscopio-magico-la-iniciativa-que-busca-acercar-la-ciencia-a-los-escolares

9.2. Conclusiones

El objetivo principal del proyecto presentado en este informe era la creación de una aplicación que simule un microscopio el cual consideramos que se realizó de forma exitosa. Del desarrollo de este proyecto se pueden realizar varias reflexiones y evaluaciones de sus distintos aspectos, como lo son el diseño de la arquitectura, el proceso de desarrollo, los resultados obtenidos y la ejecución del proyecto.

9.2.1. Aplicación y desarrollo

Se espera que en un futuro esta aplicación forme parte de la plataforma Ceibal y sea utilizada por las escuelas en todo el país. Teniendo esto en cuenta, uno de los principales requerimientos de la aplicación es la escalabilidad, la cual siempre se tuvo presente al momento de realizar el diseño de la arquitectura y la elección de las tecnologías. Una de las ventajas del diseño realizado es la posibilidad de escalar cada nodo de forma diferenciada lo que se ve reflejado en un uso más eficiente de recursos permitiendo ahorrar en costos.

Dado que la aplicación está enfocada en la simulación del funcionamiento del microscopio, el mayor desafío en la implementación se encontró en el FrontEnd y no en el Backend. Sumado a esto, la biblioteca Leaflet tiene muchas extensiones que permiten personalizar el funcionamiento del mapa pero para la personalización de la interfaz, el efecto de los botones y el efecto sobre las imágenes se debieron implementar nuevos controles de Leaflet e implementar varias animaciones y efectos a través de *Cascading Style Sheets* (CSS). A pesar de que en el equipo de desarrollo se tenia experiencia en ReactJS y conocimiento en JavaScript y JQuery, no se contaba con un experto en FrontEnd, aumentando aun más el desafío al momento de implementar el FrontEnd. Esto permitió a los desarrolladores mejorar las habilidades y conocimiento en el desarrollo de FrontEnd.

En cuanto a las tecnologías utilizadas se tuvieron buenos resultados. La dificultad en la integración y comunicación entre las diferentes tecnologías utilizadas fue mitigada gracias a la amplia documentación que existe de cada una de ellas. En cuanto a este aspecto la mayor dificultad residió en el manejo de errores y de las demoras al consumir los servicios del BackEnd desde el FrontEnd y como impactar esto en la interfaz de usuario.

La utilización de Leaflet como biblioteca para mostrar las imágenes de los

preparados facilitó ampliamente el manejo de las imágenes. La integración de Leaflet con ReactJS fue bastante directa debido a que Leaflet es una biblioteca Javascript, como se comentó anteriormente el mayor desafío se encontró en la personalización del comportamiento de la imagen mostrada por Leaflet.

Dada la naturaleza del proceso de este proyecto y la utilización de prototipado evolutivo, el diseño de la base de datos fue incremental, lo cual impidió realizar un diseño completo a partir de todos los requerimientos. El proyecto final del curso de Base de datos no relacionales realizado por algunos de los miembros del equipo de desarrollo permitió realizar un análisis más amplio y detallado sobre el diseño de la base de datos que en otro caso no hubiera sido posible realizar. Realizar este análisis ya avanzado el desarrollo de la aplicación implicó un costo en retrabajo, pero la modularización realizada en el diseño de la arquitectura permitió disminuir este costo.

El cambio a utilizar una base de datos documental con MongoDB en vez de una base relacional con PostgreSQL facilitó la implementación y mejoró el rendimiento de las operaciones de consulta pero dificultó las operaciones de inserción, siendo esto un aspecto que se tuvo en cuenta en el análisis y diseño del modelo de base de datos.

La incorporación de ElasticSearch como herramienta de búsqueda y filtrado permitió mejorar la galería pero incorporó la necesidad de sincronizar la información entre MongoDB y ElasticSearch al momento de insertar un nuevo preparado, es parte del trabajo a futuro mejorar este aspecto implementando un proceso que sincronice ambas herramientas.

La modularización de las componentes de la aplicación tuvo varios beneficios a lo largo del desarrollo, mayormente teniendo en cuenta el retrabajo que implicaron ciertas decisiones tomadas y dificultades enfrentadas durante el proyecto.

9.2.2. Lecciones aprendidas y dificultades encontradas

Dada la naturaleza del proyecto donde se tienen varios involucrados con diferentes expectativas y visiones del comportamiento de la herramienta a crear, se realizaron varias reuniones de relevamiento para entender las distintas visiones y establecer acuerdos del comportamiento esperado y el alcance del proyecto. Una vez realizadas las primeras reuniones de relevamiento de requerimientos se realizó una búsqueda de productos o aplicaciones similares donde

no se pudo encontrar ninguno que cumpliera todos los requerimientos del proyecto, sumado a esto la mayoría de estas aplicaciones parecían ser destinadas a usuarios con conocimientos científicos y no a niños que es a quienes apunta este proyecto. Este análisis fue de utilidad en la definición del comportamiento de algunas de las funcionalidades que sí estaban implementadas en los productos analizados.

Durante el proyecto se realizaron múltiples reuniones, en cierto período del proyecto reuniones semanales, en las cuales se tomaban decisiones de las diferentes ramas del proyecto. Estas reuniones resultaron mucho más efectivas cuando se tenía un plan de reunión con temas concretos a tratar y con los participantes necesarios para tratar estos temas, mientras que en otros casos al ser muchos participantes con distintos puntos de vista y prioridades las reuniones resultaron poco efectivas. Esto provocó que la definición de los comportamientos de las diferentes funcionalidades del simulador tomara más tiempo del esperado. Y debido a las diferentes expectativas que tenían los distintos miembros del equipo fue necesaria la realización de prototipos que permitieran validar las diferentes funcionalidades.

En un comienzo se definió un método de obtención de las imágenes de los preparados para que la simulación del microscopio tenga el comportamiento esperado, priorizando el realismo de la simulación, lo cual generó ciertas expectativas de las imágenes con las que se iba a contar para el simulador. Durante la obtención de imágenes, al analizar los costos de tiempo del método de obtención, se observó que eran mayores a los estimados, lo cual provocó que no sea posible contar con la cantidad de imágenes planteadas en un principio. Esto tuvo un impacto en la definición del comportamiento de la aplicación en etapas avanzadas del proyecto, implicando la definición del comportamiento de la aplicación al no contar con estas imágenes.

Otro aspecto que afectó la definición del comportamiento del simulador fue la incorporación tardía de un diseñador, que sumado al retraso de la incorporación de un diseño de interfaz de usuario, implicó entre otras cosas retrabajar o descartar trabajo realizado previamente.

Un aspecto importante del proyecto Microscopio Mágico son las visitas a las escuelas, que fueron otra motivación para realizar el prototipado rápido y evolutivo para tener una versión a utilizar en las actividades. Las actividades se realizaron con alumnos de diferentes edades y en escuelas de diferentes contextos, rurales y urbanas, que aportaron variadas devoluciones que permitieron

realizar mejoras en los prototipos. A partir de la incorporación del equipo de diseño y experiencia de usuario se realizaron mejoras en las actividades en las escuelas, lo cual a su vez mejoró la comunicación con los alumnos y maestras y así su devolución. Esto generó se enfocara más la atención en el diseño de la aplicación y la experiencia de usuario, permitiendo una amplia mejora en la usabilidad de la aplicación.

Por último se quiere destacar que el proyecto Microscopio Mágico, al ser un proyecto multidisciplinario con objetivos variados, tuvo dificultades, como algunos de los puntos analizados anteriormente, pero fue enriquecedor tanto a nivel personal como profesional.

9.3. Trabajo a futuro

Como trabajo a futuro se plantea la mejora de la interfaz de la aplicación enfocado en la usabilidad y la accesibilidad teniendo en cuenta que el destinatario principal de la aplicación son los alumnos de las escuelas.

Parte del trabajo que quedó fuera del alcance de este proyecto es uno de los siguientes puntos a definir y trabajar debido a que algunos de los requerimientos fueron realizados al inicio del proyecto y con el correr del tiempo y a raíz de las actividades en las escuelas pueden ser modificados. De todos modos, algunos de estos requerimientos son de vital importancia para la correcta implantación de la herramienta en las escuelas como lo es la integración de la aplicación con la plataforma Ceibal, que permitirá a los alumnos de las escuelas ingresar de la misma forma que ingresan a las herramientas que utilizan actualmente.

Durante el proyecto y motivado por las visitas a las escuelas se propusieron nuevas ideas que podrían ampliar el funcionamiento de la herramienta con un funcionamiento interactivo, donde los alumnos pueden ingresar sus investigaciones y sus propios preparados pudiendo así compartirlo con el resto de los alumnos de las diferentes escuelas. Para que esto sea posible es necesario primero cumplir el objetivo de integrar la herramienta con la plataforma Crea de Ceibal.

Teniendo en cuenta que no se tiene certeza de cómo será la utilización de la aplicación es importante como trabajo a futuro incorporar herramientas de monitoreo, por ejemplo con el Elastic Stack (ELK), que permitan controlar el tráfico de la aplicación y sus cuellos de botella, pudiendo así realizar mejoras

en el rendimiento y la disponibilidad de la aplicación utilizando por ejemplo las estrategias de escalabilidad analizadas en el diseño de la arquitectura.

Bibliografía

[1] Sanchit Aggarwal. "Modern Web-Development using ReactJS". En: *International Journal of Recent Research Aspects* 5 (2018), págs. 133-137.

[2] Francesca Bugiotti y col. "Database design for NoSQL systems". En: *International Conference on Conceptual Modeling*. Springer. 2014, págs. 223-231.

[3] Lakshmi Prasanna Chitra y Ravikanth Satapathy. "Performance comparison and evaluation of Node. js and traditional web server (IIS)". En: *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*. IEEE. 2017, págs. 1-4.

[4] ElasticSearch. *Scalability and resilience: clusters, nodes, and shards*. URL: https://www.elastic.co/guide/en/elasticsearch/reference/current/scalability.html (visitado 14-10-2019).

[5] Ana Flores y col. "Performance Evaluation of NoSQL and SQL Queries in Response Time for the E-government". En: *2018 International Conference on eDemocracy & eGovernment (ICEDEG)*. IEEE. 2018, págs. 257-262.

[6] Franklin Leung y Bing Zhou. "Performance evaluation of Twitter datasets on SQL and NoSQL DBMS". En: *Web Intelligence*. Vol. 14. 4. IOS Press. 2016, págs. 275-286.

[7] MongoDB. *Replication*. URL: https://docs.mongodb.com/manual/replication/ (visitado 14-10-2019).

[8] MongoDB. *Shard Keys*. URL: https://docs.mongodb.com/manual/core/sharding-shard-key/ (visitado 14-10-2019).

[9] MongoDB. *Sharding*. URL: https://docs.mongodb.com/manual/sharding/ (visitado 14-10-2019).

[10] Adrian Rosebrock. *Image Stitching with OpenCV and Python*. 2018. URL: https://www.pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/ (visitado 14-10-2019).

[11] Adrian Rosebrock. *OpenCV panorama stitching*. 2016. URL: https://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/ (visitado 14-10-2019).

[12] Rahul Soni. *Nginx*. Springer, 2016. DOI: 10.1007/978-1-4842-1656-9.

[13] Wikipedia. *Foco (óptica)*. URL: https://es.wikipedia.org/wiki/Foco_(%C3%B3ptica) (visitado 14-10-2019).

[14] Wikipedia. *Microscopio*. URL: https://es.wikipedia.org/wiki/Microscopio (visitado 14-10-2019).

[15] Wikipedia. *Microscopio óptico*. URL: https://es.wikipedia.org/wiki/Microscopio_%C3%B3ptico (visitado 14-10-2019).

ANEXOS

Anexo A

Manual API

Los servicios utilizados para acceder a la base de preparados y sus metadatos son expuestos a través de una API Rest implementada en Node.js. En este manual se detalla cada servicio de la API, brindando una descripción, la forma de acceder a ellos y sus parámetros de entrada y salida.

A.1. Obtener preparados

Este servicio que utiliza el método GET retorna todos los preparados con sus datos del sistema. Se encuentra expuesto en preparados/getpreparados.

Parámetros de entrada: Ninguno.

Parámetros de salida: En el código A.7 se observa la estructura del JSON devuelto.

Código A.7 Salida de obtener preparados

```
1   preparados:[{
2       _id:,
3       idp:,
4       nombre:'',
5       path:'',
6       descripcion:'',
7       fecha_adq:'',
8       microscopio:'',
9       tejido:'',
10      tincion:'',
11      observaciones:'',
12      enable:'',
13      categorias:[{_idc:}],
14      magnificaciones:[{_valor:,nombre:'',descripcion:'',profundidades:[{_valor:,
            descripcion:'',alto:,ancho:,puntosDeInteres:[]}]}]
15  }]
```

En la tabla A.1 se muestra información sobre cada uno de los datos contenidos dentro del JSON.

Dato	Descripción	Tipo de dato
preparados	Arreglo con los preparados del sistema	Arreglo de JSON
_id	Identificador de objeto	Alfanumérico
idp	Identificador de preparado	Entero
nombre	Nombre de preparado	Texto
path	Ubicacion del preparado en el servidor	Texto
descripcion	Descripción del preparado	Texto
fecha_adq	Fecha de adquisición del preparado	Fecha
microscopio	Microscopio usado para obtener las imágenes	Texto
tincion	Método de tinción.	Texto
tejido	Tejido del preparado	Texto
observaciones	Observaciones sobre el preparado	Texto
enable	Preparado habilitado en el sistema.	Booleano
categorias	Arreglo de categorías del preparado	Arreglo de objetos JSON
categorias- _idc	Identificador de la categoría	Entero
maginificaciones	Arreglo de maginificaciones del preparado	Arreglo de objetos JSON
maginificaciones- _valor	Identificador de la maginificación	Entero
maginificaciones-nombre	Nombre de la magnificación	Texto
maginificaciones- descripcion	Descripción de la magnificación	Texto
maginificaciones- profundidades	Arreglo de profundidades de la magnificación	Arreglo de objetos JSON
maginificaciones-profundidades- _valor	Identificador de la profundidad	Entero
maginificaciones-profundidades- descripcion	Descripción de la profundidad	Texto
maginificaciones-profundidades- alto	Alto de las imágenes a esa profundidad	Entero
maginificaciones-profundidades- ancho	Ancho de las imágenes a esa profundidad	Entero
maginificaciones-profundidades- puntosDeInteres	Arreglo que contiene los puntos de interes de esa profundidad	Arreglo de objetos JSON

Tabla A.1: Especificación de datos de obtener preparados.

A.2. Obtener preparados habilitados

Este servicio que utiliza el método GET retorna todos los preparados con sus datos que se encuentran habilitados en el sistema. Se encuentra expuesto en preparados/getenabledpreparados.

Parámetros de entrada: Ninguno.

Parámetros de salida: En el código A.7 se observa la estructura del JSON devuelto.

En la tabla A.1 se muestra información sobre cada uno de los datos contenidos dentro del JSON.

A.3. Obtener preparado

Este servicio que utiliza el método GET dado un id de un preparado retorna el preparado que tiene ese id con sus datos. Se encuentra expuesto en preparados/getpreparado/:id.

Parámetros de entrada: Identificador del preparado.

Parámetros de salida: En el código A.8 se observa la estructura del JSON devuelto.

Código A.8 Salida de obtener preparados

```
1     preparado:{
2         _id:,
3         idp:,
4         nombre:'',
5         path:'',
6         descripcion:'',
7         fecha_adq:'',
8         microscopio:'',
9         tejido:'',
10        tincion:'',
11        observaciones:'',
12        enable:'',
13        categorias:[{_idc:}],
14        magnificaciones:[{_valor:,nombre:'',descripcion:'',profundidades:[{_valor:,
              descripcion:'',alto:,ancho:,puntosDeInteres:[]}]}]
15    }
16 }
```

En la tabla A.1 se muestra información sobre cada uno de los datos contenidos dentro del JSON.

A.4. Obtener categorías

Este servicio que utiliza el método GET retorna todas las categorías con sus datos del sistema. Se encuentra expuesto en preparados/getcategorias.

Parámetros de entrada: Ninguno.

Parámetros de salida: En el código A.9 se observa la estructura del JSON devuelto.

Código A.9 Salida de obtener categorías

```
1     categorias:{
2         _id:,
3         idc:,
4         nombre:'',
5         descripcion:'',
6         padres:[{idpadre:,jerarquia:}],
7         hijos:[{idhijo:}],
8         preparados:[{idp:,nombre:'',descripcion:''}]
9     }
10 }
```

En la tabla A.2 se muestra información sobre cada uno de los datos contenidos dentro del JSON.

Dato	Descripción	Tipo de dato
categorias	Arreglo con las categorías del sistema	Arreglo de JSON
_id	Identificador de objeto	Alfanumerico
idc	Identificador de categoría	Entero
nombre	Nombre de la categoría	Texto
descripción	Descripción de la categoría	Texto
padres	Arreglo con los padres de la categoría	Arreglo de JSON
padres- idpadre	Identificador del padre de la categoría	Entero
padres- jerarquía	Jerarquía de la categoría para ese padre	Entero
hijos	Arreglo con los hijos de la categoría	Arreglo de JSON
hijos- idhijo	Identificador del hijo de la categoría	Entero
preparados	Arreglo con los preparados de la categoría	Arreglo de JSON
preparados- idp	Identificador del preparado perteneciente a la categoría	Entero
preparados- nombre	Nombre del preparado perteneciente a la categoría	Texto
preparados- descripción	Descripción del preparado perteneciente a la categoría	Texto

Tabla A.2: Especificación de datos de obtener categorías.

Anexo B

Manual Administrador

Este manual describe el funcionamiento del BackOffice a ser usado por un usuario administrador para agregar y modificar preparados. Para la utilización de esta aplicación es necesario iniciar sesión con un usuario y contraseña.

El BackOffice tiene como pantalla principal la visualización de los preparados del sistema, que se visualiza de la misma manera que en el FrontOffice, de esta forma el usuario administrador puede visualizar la galería y buscador como lo haría el usuario final del FrontOffice.

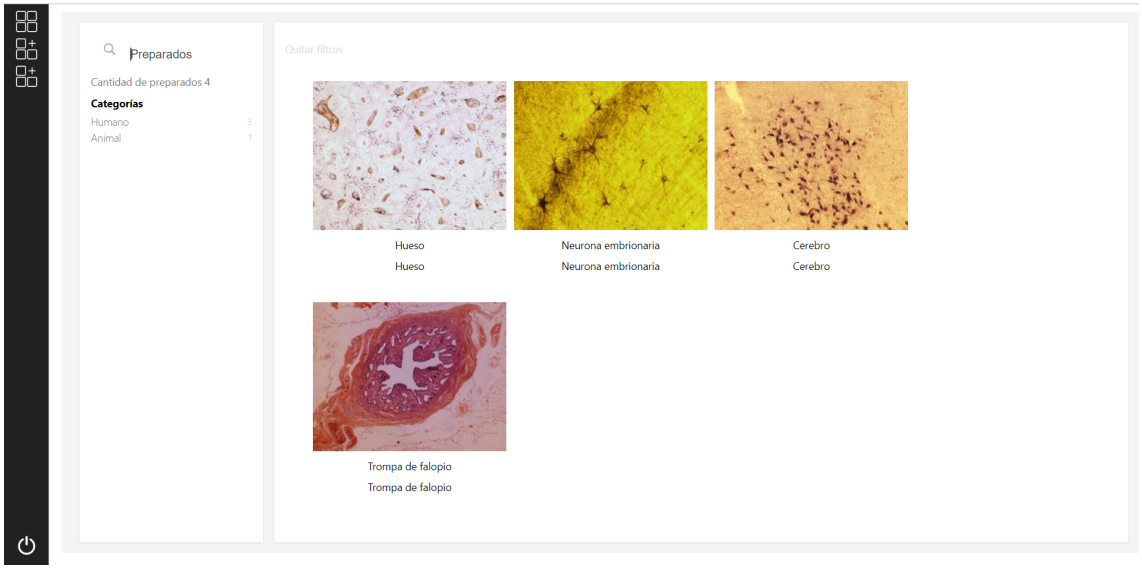


Figura B.1: Visualización de preparados en BackOffice

El menú a la izquierda tiene cuatro opciones:

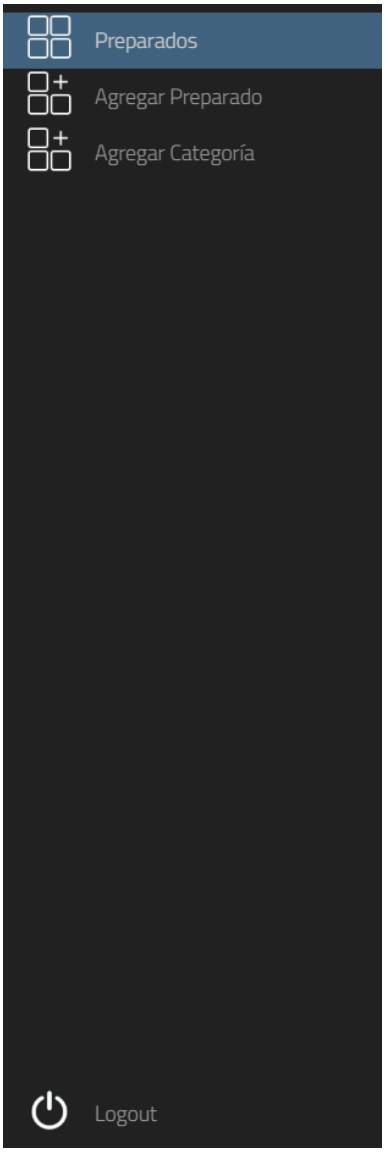


Figura B.2: Menú del BackOffice

- Preparados: redirige al usuario a la pantalla de visualización de preparados ya mostrada.
- Agregar preparado: redirige al usuario a la pantalla para agregar preparados.
- Agregar categoría: redirige al usuario a la pantalla para agregar categorías.
- Logout: Cierra la sesión del usuario.

B.1. Agregar Preparado

La pantalla de agregar preparado permite ingresar todos los datos del preparado para ingresarlo en el sistema.

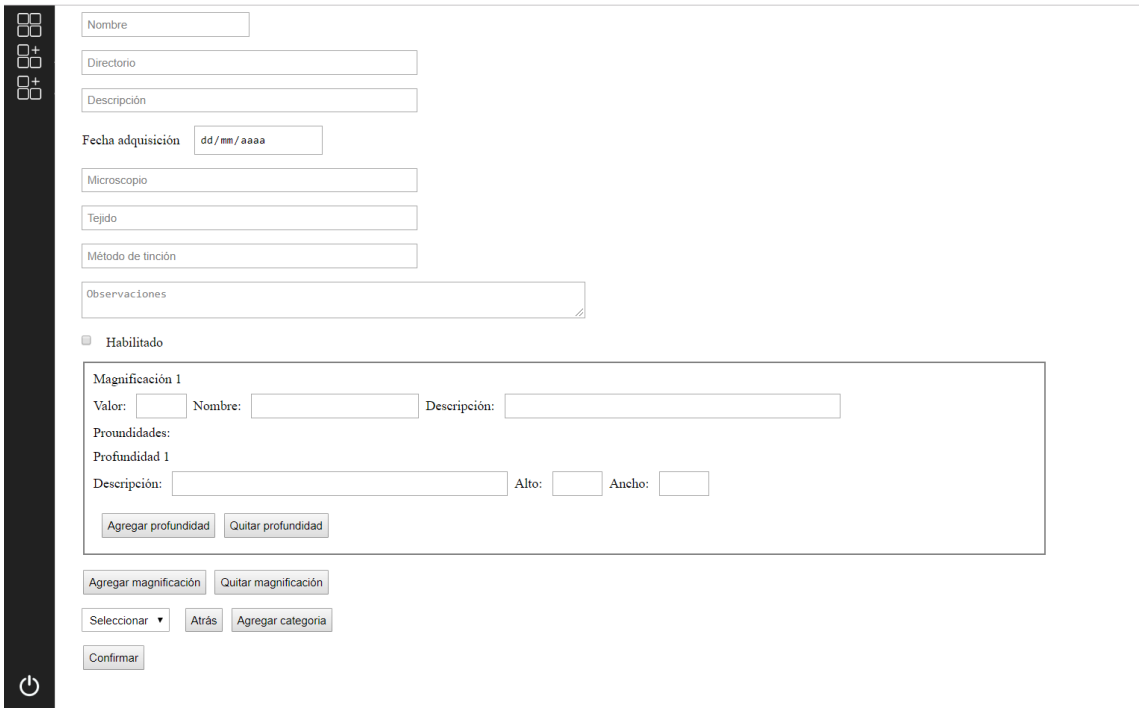


Figura B.3: Agregar preparado

Para cada preparado además de los datos básicos se pueden agregar las magnificaciones, donde para cada una se debe agregar el valor de la magnificación, el nombre y una descripción. Además para cada magnificación se deben agregar las profundidades, donde para cada magnificación se debe ingresar una descripción y el alto y ancho de la imagen. El alto y ancho es importante para la correcta visualización del preparado en la exploración del FrontOffice. Para cada preparado se pueden agregar categorías en las cuales luego aparecerá en el FrontOffice.

B.2. Agregar Categoría

Esta pantalla permite agregar una nueva categoría con sus datos:

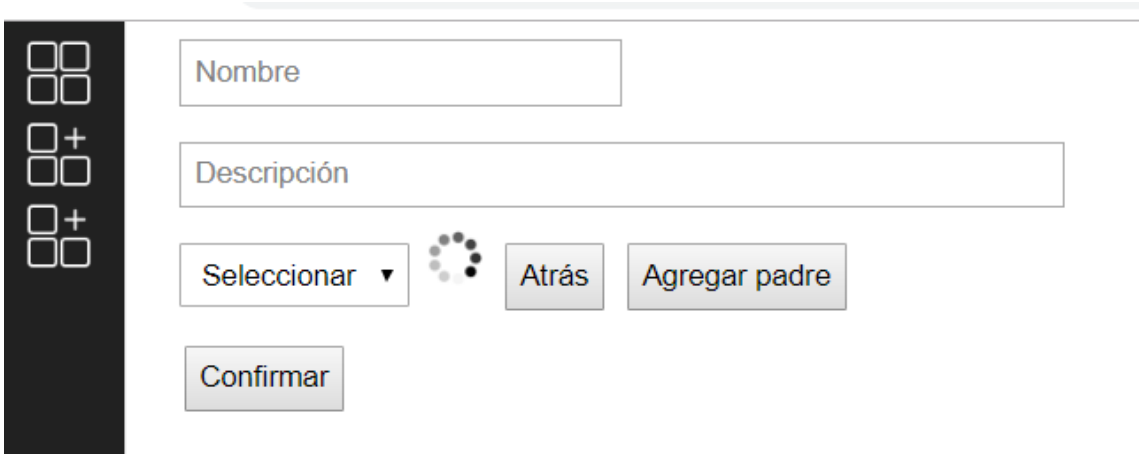


Figura B.4: Agregar categoría

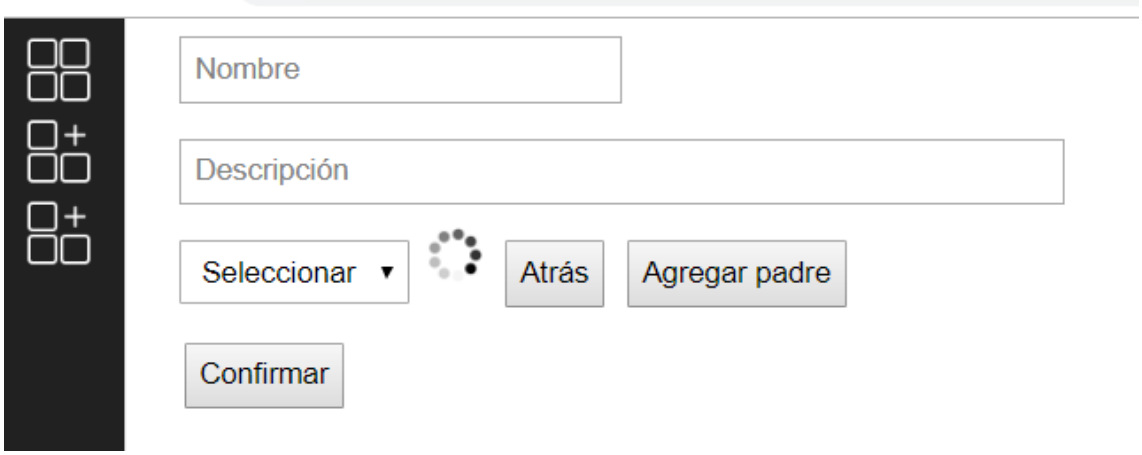


Figura B.5: Agregar categoría

elfig:manualadmin$_v isualizacion$

Anexo C

Procesamiento de imágenes

Luego de que las imágenes son obtenidas a partir de los microscopios se debe realizar un post-procesamiento para poder ser utilizadas en la aplicación. Este post-procesamiento tiene principalmente dos pasos, la unión de las imágenes (*stitching*) detallado en la sección C.1 y la partición de las imágenes (*tiling*) detallado en la sección C.2. Luego de estos dos procesamientos se deben realizar algunos ajustes en la nomenclatura de las imagenes, esto se detalla en la sección C.3.

C.1. Stitching

Para la implementación del *stitching* de las imágenes se utilizó una implementación realizada por Adrian Rosebrock [10] que utiliza OpenCV y Python para la implementación.

Para realizar el *stitching* para generar una imagen se deben dejar todas las imágenes dentro de una misma carpeta y ejecutar el siguiente comando:

Código C.10 Comando *stitching*

```
1 python image_stitching.py --images pathimagenes --output output.png --crop 0
```

Donde pathimagenes es el directorio donde se encuentran todas las imágenes a unir y output.png es la imagen generada. El parámetro *crop* indica si se debe cortar la imagen para que no haya huecos en los bordes.

Si las imágenes son muy grandes se puede generar un problema de memoria al quedarse sin memoria la unidad de procesamiento gráfico (GPU), para esto se debe crear la variable de entorno OPENCV_OPENCL_DEVICE=disabled que deshabilita OPENCL que es lo que utiliza la GPU.

C.2. Tiling

Para realizar el *tiling* de las imágenes se utiliza la librería libvips que tiene el operador dzsave[1] que parte la imagenes *tiles* y las guarda.

Código C.11 Comando *tiling*

```
1  vips dzsave PathImagenfinal.jpg imagenesAUnir.zip --background 0 --centre --layout zoomify
       --depth one
```

Donde PathImagenfinal.jpg es la imagen que se genera al realizar el *stitching* de las imágenes que se encuentran en el archivo comprimido imagenesAUnir.zip. La opciones son:

- background 0: Indica el color del fondo de las partes que no son de la imagen, por ejemplo los bordes agregados.
- centre: centra la imagen en el *tile* agregando bordes si es necesario.
- layout zoomify: Genera la estructura de carpetas con los *tiles* de la imagen.
- depth one: Permite partir una imagen grande en muchos *tiles*.

C.3. Otros procesamientos

Es necesario renombrar las imágenes, porque el comando de *tiling* te deja todas las imágenes con z = 0 pero para que funcionen las magnificaciones y el minimapa correctamente se necesita ponerle el z correspondiente según la magnificación de la imagen:

- Si es la primer magnificación se tiene que renombrar a 4: ren 0-*-*.jpg 4-*-*.jpg
- Si es la segunda magnificación se tiene que renombrar a 5: ren 0-*-*.jpg 5-*-*.jpg
- Si es la tercer magnificación se tiene que renombrar a 6: ren 0-*-*.jpg 6-*-*.jpg

De la misma forma funciona si se tienen más magnificaciones, se debe ir aumentando el número por el cual se sustituye el z. Estos comandos deben ejecutarse en la carpeta donde se encuentra el *tiling* de cada magnificación.

[1] https://libvips.github.io/libvips/API/current/Making-image-pyramids.md.html

Este comando es para cambiar el tamaño de la imagen, esto se debería hacer con la imagen principal para tener la imagen del minimapa:

Para obtener la imagen a utilizar en el minimapa se debe obtener una versión de menor tamaño de la imagen principal, para esto se puede ejecutar el siguiente comando.

Código C.12 Comando minimapa

```
1  vipsthumbnail PathImagenPrincipal.jpg --size 240x176
```

Donde la imagen PathImagenPrincipal.jpg es la imagen que se debe mostrar en el minimapa y –size indica la resolución de la imagen (ancho por alto).