
**ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL**

**Facultad de Ingeniería en Electricidad y Computación**

# "DESARROLLO DE UNA HERRAMIENTA WEB PARA ETIQUETADO DE IMÁGENES HYPERESPECTRALES"

**INFORME DE MATERIA INTEGRADORA**

Previo a la obtención del Título de:

**INGENIERO EN COMPUTACIÓN**

OSWALDO ALEJANDRO BAYONA ANDRADE

GUAYAQUIL – ECUADOR

AÑO: 2017

# AGRADECIMIENTO

Agradezco a todas las personas que me hayan ayudado durante mi formación, porque hicieron posible que finalizara mi carrera. De verdad que mi gratitud es inconmensurable.

Gracias.

Oswaldo Bayona

# DEDICATORIA

Este proyecto lo dedico a mi amada familia.

# TRIBUNAL DE EVALUACIÓN

**PhD. Luis Mendoza**

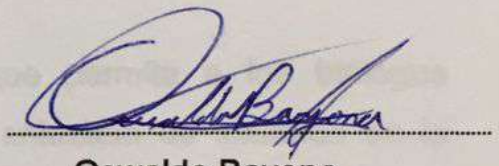
PROFESOR EVALUADOR

**PhD. Daniel Ochoa**

PROFESOR EVALUADOR

# DECLARACIÓN EXPRESA

"La responsabilidad y la autoría del contenido de este Trabajo de Titulación, me corresponde exclusivamente; y doy mi consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la producción intelectual"



Oswaldo Bayona

# RESUMEN

La Sigatoka Negra es una enfermedad que afecta a las hojas de las plantas de banano. Esta enfermedad es devastadora porque puede afectar a toda una plantación. Por ello, el Centro de Visión y Robótica (CVR) en colaboración con el Centro de Investigaciones Biotecnológicas del Ecuador (CIBE) proponen estudiar la Sigatoka Negra usando imágenes hyperespectrales de plantas inoculadas.

El conjunto de datos generados por el CVR debe ser etiquetado por un biólogo, para generar datos de referencia con que comparar el rendimiento de algoritmos. El etiquetado consiste en seleccionar zonas enfermas de la hoja. El problema es que no hay una forma práctica de realizar este proceso.

Como solución se desarrolló una aplicación web que permite a los biólogos visualizar las imágenes para realizar el etiquetado y enlazado de estadios de la enfermedad, de una forma fácil.

# ÍNDICE GENERAL

AGRADECIMIENTO ..... ii
DEDICATORIA ..... iii
TRIBUNAL DE EVALUACIÓN ..... iv
DECLARACIÓN EXPRESA ..... v
RESUMEN ..... vi
ÍNDICE GENERAL ..... vii
CAPÍTULO 1 ..... 1
1. INTRODUCCIÓN ..... 1
1.1 Sigatoka negra ..... 1
1.2 Imágenes Hyperespectrales ..... 2
1.3 Experimento para estudiar Sigatoka negra ..... 3
CAPÍTULO 2 ..... 7
2. ANÁLISIS Y SOLUCIÓN TECNOLÓGICA EMPLEADA ..... 7
2.1 Dataset ..... 7
2.2 Arquitectura de la aplicación ..... 8
2.3 Base de datos ..... 9
2.3.1 Implementación ..... 12
2.4 Servidor Web ..... 12
2.5 Cliente Web ..... 15
2.5.1 Implementación ..... 16
CAPÍTULO 3 ..... 26
3. RESULTADOS ..... 26
3.1 Descripción de la aplicación ..... 26
3.2 Evaluación de la aplicación ..... 35
CONCLUSIONES Y RECOMENDACIONES ..... 36
BIBLIOGRAFÍA ..... 37

# CAPÍTULO 1

## 1. INTRODUCCIÓN

Las bananas son susceptibles a muchas enfermedades; entre ellas, la Sigatoka negra, la cual es muy virulenta y tiene el potencial de devastar una plantación. Por ello, el Centro de Visión y Robótica (CVR), en colaboración con el Centro de Investigaciones Biotecnológicas del Ecuador (CIBE) proponen investigar la Sigatoka negra usando la técnica de imágenes hyperespectrales. Con la finalidad de dar soporte a la investigación llevada a cabo por el CVR, se ha propuesto el desarrollo de una herramienta web para el etiquetado de imágenes hyperespectrales.

En la sección 1.1 se describe brevemente la Sigatoka negra, luego en la sección 1.2, se define que son imágenes hyperespectrales y sus aplicaciones, finalmente, en la última sección se explica el experimento diseñado para estudiar la Sigatoka negra y cómo este proyecto apoya el proceso.

### 1.1 Sigatoka negra

La Sigatoka negra es una enfermedad que afecta a las hojas de banano. Es causada por un hongo llamado *Mycosphaerella fijiensis*, y se caracteriza por que produce manchas negras en las hojas. Fouré [1] clasificó los síntomas visibles de la enfermedad en 6 estadios o etapas. En el primer estadio los síntomas son pequeñas manchas cloróticas (con insuficiente clorofila y coloración verde pálido) y visibles solamente en el envés de la hoja. A medida que evoluciona la enfermedad, las manchas se vuelven marrones-rojizas, luego se unen unas con otras formando grandes manchas negras [2] y, finalmente, se produce la muerte de la hoja. La Figura 1.1 muestra lesiones en diferentes estadios.

La detección de la Sigatoka negra es difícil. Cuando los síntomas más notorios aparecen puede ser demasiado tarde y la enfermedad se pudo haber esparcido a una gran parte del plantío. Por esta razón, el CVR plantea realizar la detección temprana de Sigatoka negra usando imágenes hyperespectrales.

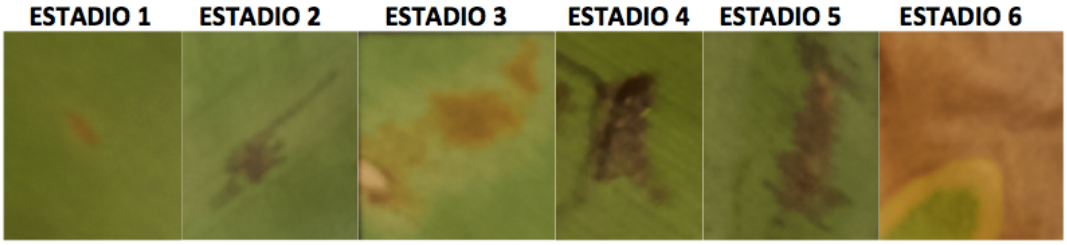



**Figura 1.1: Lesiones de Sigatoka negra en diferentes estadios.**

## 1.2 Imágenes Hyperespectrales

Las imágenes hyperespectrales son un conjunto de imágenes de dos dimensiones $(x, y)$, en la cual cada imagen corresponde a una banda o longitud de onda del espectro electromagnético. Este conjunto de imágenes se conoce como cubo hyperespectral y es un arreglo de tres dimensiones $(x, y, \lambda)$ donde $\lambda$ es una longitud de onda. La Figura 1.2 esquematiza un cubo hyperespectral y se observa imágenes de una hoja de banano en varias longitudes de onda.

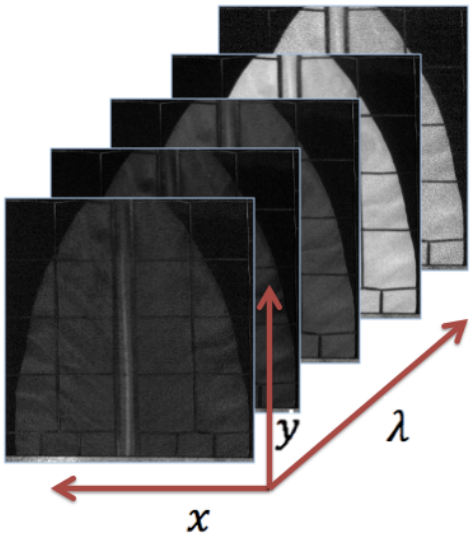



**Figura 1.2: Esquema de un cubo hyperespectral.**

Esta técnica permite obtener información del espectro de reflectancia de un objeto, el cual puede ser usado para caracterizar su composición bioquímica [3]. Las imágenes hyperespectrales han sido aplicadas con éxito en tareas como: el control de calidad de alimentos, en agricultura, mineralogía, física y medicina, entre otros.

## 1.3 Experimento para estudiar Sigatoka negra

El CVR cuenta con un sistema de adquisión de imágenes con el que se obtienen los cubos hyperespectrales. El sistema (ver Figura 1.3) consiste de una estructura metálica, en cuyo interior hay una cámara hyperespectral montada sobre un deslizador. El deslizador traslada la cámara hyperespectral a lo largo de un eje, lo que permite escanear un objeto. El sistema cuenta con una plataforma para colocar una planta y un soporte para colocar una hoja frente a la cámara. Además, el sistema tiene una cámara RGB para capturar imágenes a color de alta resolución. Una detallada descripción del sistema de adquisión de imágenes puede ser encontrada en [4].

**Figura 1.3: Sistema de captura de imágenes hyperespectrales.**

La preparación del experimento consistió en inocular 10 plantas en cada una de sus hojas con el hongo. Las plantas fueron almacenadas en un invernadero, donde las condiciones de temperatura y humedad son controladas para que la enfermedad se desarrolle. Se identificó cada planta y hoja con unas etiquetas (ver Figura 1.4).

**Figura 1.4: Etiquetas usadas para identificar plantas y hojas.**

Una vez realizados los pasos anteriores se empezó el experimento. En primer lugar, se procedió a capturar imágenes de cada hoja identificada, usando el sistema de captura, con una frecuencia de tres veces por semana. Cada hoja escaneada produce una carpeta que contiene el cubo hyperespectral, una imagen con modelo de color Red Green Blue (RGB) de alta resolución y un archivo con meta-datados como los identificadores de la planta y hoja, la fecha y la hora.

El siguiente paso es la identificación o etiquetado de las lesiones. Un biólogo

debe abrir la imagen RGB de una hoja, y marcar regiones en la imagen que correspondan a lesiones e identificarlas; es decir, agregarle un atributo que indique el estadio, o si la lesión se debe a daños mecánicos, deshidratación u otros (ver Figura 1.5). El etiquetado se lo realizó en las imágenes RGB porque las imágenes hyperespectrales tienen una resolución muy baja que dificulta visualizar las regiones. El resultado de este paso es un conjunto de regiones, definidas por coordenadas dentro de la imagen (RGB) y que tienen un atributo estadio.

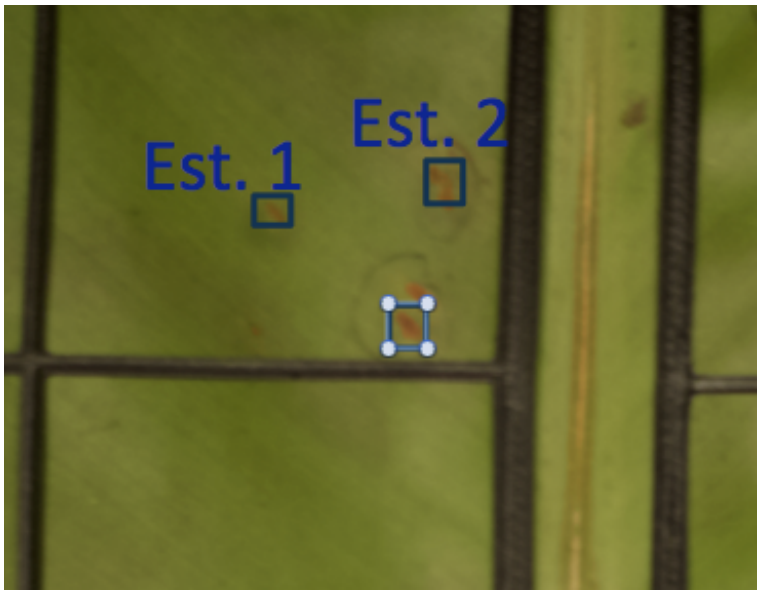



**Figura 1.5: Procedimiento de cómo se debería etiquetar las lesiones. Se observa que se crean regiones (rectángulos azules) que tienen un atributo estadio.**

Del paso anterior se obtienen regiones por cada hoja en cada sesión de escaneo. Debido a que se quiere estudiar la evolución de las lesiones (que están encerradas en las regiones), se deben enlazar las regiones correspondientes a través del tiempo. La Figura 1.6 esquematiza este proceso, se observa tres sesiones de escaneo de un mismo segmento de hoja,

realizadas en diferentes días. Se muestra también las regiones etiquetadas y unos enlaces que relacionan las regiones correspondientes.

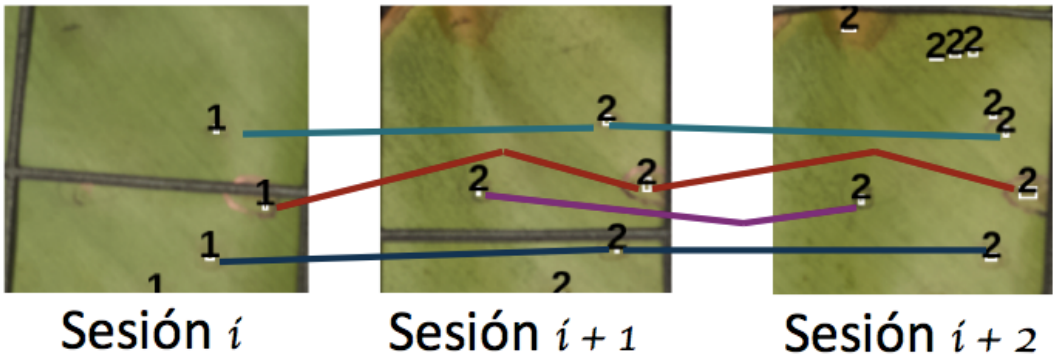



**Figura 1.6: Enlazado de regiones correspondientes a través del tiempo.**

Luego hay que transformar las coordenadas de las regiones obtenidas, de tal forma que las coordenadas resultantes correspondan al sistema de coordenadas de las imágenes hyperespectrales. Esto se lo realiza alineando la imagen RGB y una de las hyperespectrales.

Este procedimiento experimental se lo realiza hasta que las hojas lleguen al último estadio. Por lo tanto, al final se obtienen datos de la evolución de la enfermedad que son posteriormente analizados.

Con base en todo lo descrito anteriormente, el proyecto de materia integradora está enfocado en desarrollar una herramienta web que permita acceder fácilmente a los datos (las carpetas obtenidas al escanear una hoja), visualizar las imágenes hyperespectrales y la RGB, etiquetar las lesiones de las hojas, y enlazar las regiones.

# CAPÍTULO 2

## 2. ANÁLISIS Y SOLUCIÓN TECNOLÓGICA EMPLEADA

En este capítulo se analiza el problema a solucionar, y se detalla la solución propuesta. Se empieza revisando los datasets que la aplicación usa. Luego se explica en detalle la arquitectura de la aplicación, que consta de tres capas: base de datos, servidor web y cliente web. Además, se justifica las decisiones de diseño.

### 2.1 Dataset

Al escanear una hoja de banano se genera una carpeta cuyo nombre sigue el patrón: “Experimento-año-mes-día hora-minuto-segundo”; es decir, se compone de un prefijo “Experimento” y de la fecha y hora en que la hoja fue escaneada. Todas las carpetas son almacenadas en un directorio raíz. Las carpetas no están organizadas en subcarpetas, ni por planta ni por hoja, simplemente están todas al mismo nivel.

El contenido de una carpeta experimento se compone de, una carpeta cubo, una carpeta fotoThor, una carpeta imagenesCalibracion, una imagen rgb.tiff y un archivo Data_Experimento.txt. La Tabla 1 resume lo que cada archivo es.

| Nombre | Descripción |
|---|---|
| **cubo/** | En esta carpeta se almacena el conjunto de imágenes que forman el cubo hyperespectral. |
| **fotoThor/** | Contiene las imágenes en crudo, las cuales son procesadas para obtener el cubo hyperespectral. |
| **imagenesCalibracion/** | Carpeta con imágenes que son usadas para calibrar las imágenes de fotoThor. |
| **rgb.tiff** | Imagen RGB de alta resolución (3264 x 4352 pixeles) de la hoja escaneada. |

| | |
|---|---|
| **Data_Experimento.txt** | Archivo con metadatos del experimento. Contiene el identificador de la planta y de la hoja, la hora y la fecha. |

**Tabla 1: Descripción del contenido de una carpeta experimento**

## 2.2 Arquitectura de la Aplicación

La aplicación está compuesta por tres capas, la capa de datos, el servidor web y el cliente web. La Figura 2.1 muestra el diagrama de modelo de la arquitectura general de la aplicación. La capa de datos consiste de una base de datos y del dataset. El servidor web se encarga de conectarse a la base de datos, cargar los archivos del dataset y servir los datos a los clientes. El cliente web comprende la interfaz gráfica de usuario, la lógica de negocio y la comunicación con el servidor.

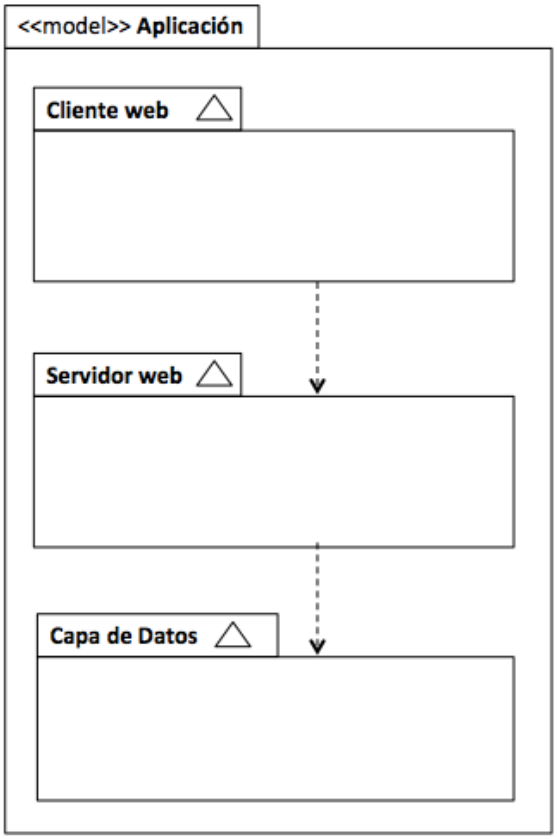



**Figura 2.1 Diagrama de modelos de la aplicación.**

## 2.3 Base de datos

Para la aplicación se decidió implementar una base de datos con el objetivo de indexar el dataset. Este índice permite realizar operaciones deseables como, agrupar los experimentos por hoja o por planta, ordenar por fechas y agrupar por estadio, entre otras que, sin la ayuda del índice, sería complicado de implementar y propenso a errores. El indexado se lo realiza insertando los metadatos del archivo Data_Experimento.txt y el nombre de la carpeta experimento en un registro de una tabla.

Además, se puede usar la base de datos para otras tareas tales como: implementar la autenticación de los usuarios al sistema, guardar las regiones y enlaces y agregar otros metadatos que ayuden al control del proceso; por ejemplo, una bandera que indique si a un experimento ya se le generó el cubo hyperespectral.

La Figura 2.2 muestra el diagrama de la base de datos. Se observa que las entidades son Planta, Hoja, Experimento, Region, transformedRegion, Enlace, Usuario y SesionInfo. Una planta tiene una o muchas hojas por lo que esta relación se refleja en el esquema de la base de datos. También se modela la relación: una hoja tiene uno o muchos experimentos. Se pudo haber fusionado las entidades Planta y Experimento porque Planta no tiene muchos atributos, pero se decidió conservarlos separados porque en el futuro una hoja podría tener más atributos (por ejemplo, método de inoculación usado en la hoja).

Sobre la imagen RGB de un experimento, el biólogo debe dibujar las regiones y etiquetarlo con su respectivo estadio. A las regiones se las puede mover, borrar, cambiar el tamaño y cambiar el estadio. Entonces, a las regiones también se decidió guardarlas en una tabla de la base de datos porque cambian constantemente y guardarlas en archivos requeriría implementar algo que manipule archivos, lo cual es complicado y propenso a errores.

Una región debe poder enlazarse con otra región que es la correspondiente a la misma zona de la hoja pero en otro instante de tiempo. Varias regiones pequeñas pueden, en el futuro, unirse y formar una región grande; por lo tanto, todas ellas se deben enlazar entre sí (ver Figura 2.3). Claramente, la manera en

que se enlazan las regiones forma algo similar a un grafo, en donde los nodos son las regiones y los arcos son los enlaces. En consecuencia, un enlace debe tener referencias a dos regiones, y eso es lo que se observa en la Figura 2.2, que se modela la relación: un enlace tiene muchas regiones, pero una región puede pertenecer a cero o muchos enlaces.

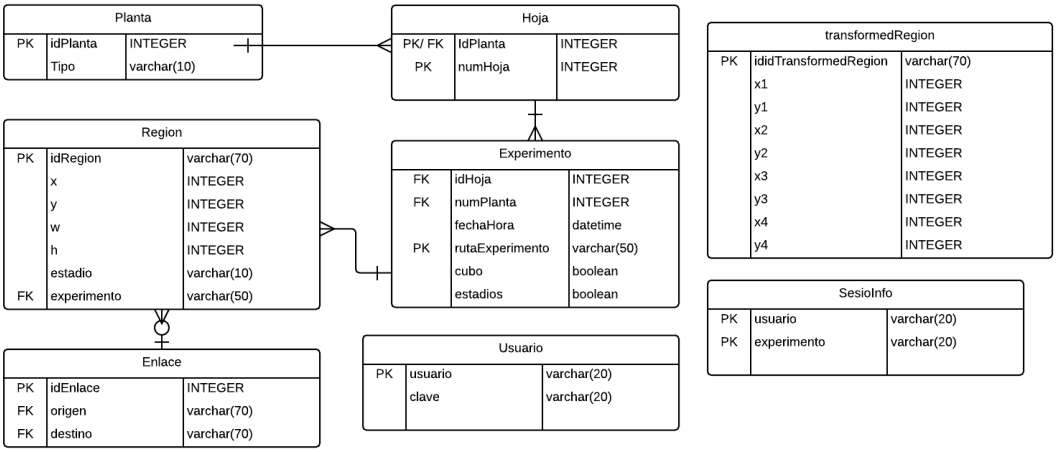



**Figura 2.2 Diagrama de la base de datos.**

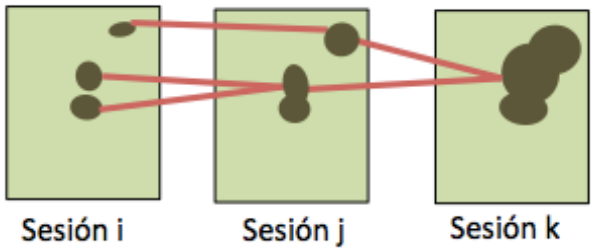



**Figura 2.3 Enlazado de regiones que crecen y se unen. Se aprecia que se forma algo similar a un grafo.**

La tabla Planta tiene el atributo idPlanta que es el identificador que la planta posee (ver Figura 1.4), también posee el atributo tipo que sirve para indicar si la planta fue inoculada o si es una planta de control (sin inocular). La tabla Hoja tiene dos atributos idPlanta y numHoja, que forman su clave primaria

compuesta. El atributo numHoja corresponde a la etiqueta que la hoja posee (ver Figura 1.4)

La tabla Experimento tiene como claves foráneas a los atributos idPlanta y numHoja, para referenciar la hoja a la cual pertenece. Como clave primaria usa la columna rutaExperimento, que es el nombre de la carpeta experimento que está siendo indexada. El atributo fechaHora indica la fecha y la hora del escaneo, el cual es útil para ordenar los experimentos. La bandera cubo indica si a ese experimento se le generó el cubo hyperespectral.

La tabla Region tiene como clave primaria una cadena de caracteres que es generada por el cliente web. Un registro de Region representa un rectángulo dentro de la imagen RGB; entonces, los atributos x, y son la coordenada de la esquina superior izquierda; los atributos w, h son el ancho y la altura en pixeles del rectángulo. El atributo estadio es una cadena de caracteres que guarda el estadio de la lesión. Y la columna experimento es una referencia al experimento al cual pertenece la región.

En la tabla transformedRegion se almacenan los resultados de transformar las coordenadas de las regiones. Este proceso es realizado alineando la imagen RGB y una imagen hyperespectral, lo que produce una matriz de transformación que es aplicada a los cuatro puntos de las regiones, dando como resultado otros cuatro puntos que definen una región en el sistema de coordenadas de las imágenes hyperespectrales.

La tabla Enlace tiene como clave primaria una cadena de caracteres que es generada por el cliente web. Los atributos origen y destino son referencias a dos regiones distintas, esto es lo que crea el enlace.

La tabla Usuario tiene como clave primaria el nombre del usuario. Y también tiene el atributo clave, que es la contraseña.

Finalmente, la tabla SesionInfo es usada para guardar información de la sesión de un usuario. En un registro de SesionInfo se almacena el identificador de un usuario y el identificador de un experimento que el usuario está utilizando.

### 2.3.1 Implementación

Para la implementación se escogió MySQL, el cual es un sistema gestor de bases de datos relacional que es desarrollado, distribuido y soportado por Oracle Corporation. Se decidió usar MySQL porque es Open Source, confiable, escalable, fácil de usar, portable [5] y tiene muchas interfaces de programación; es decir, se puede acceder a la base de datos usando lenguajes como C, C++, Python, Java, Javascript (Node.js), Pascal, etcétera.

Se crearon dos tipos de usuarios. Uno es el administrador, que tiene todos los privilegios tales como creación, eliminación y actualización de tablas y registros. Y el otro usuario sólo puede realizar consultas de los datos.

La base de datos cuenta también con storage procedures, que definen una interfaz para las operaciones que realiza la aplicación, evitando así tener que modificar el código de las aplicaciones clientes cuando el esquema de la base se modifique [6]. Se tienen procedimientos para, crear un nuevo experimento, hoja, planta, región, enlace; para autenticar un usuario; para eliminar enlaces y regiones; editar regiones y obtener los datos.

## 2.4 Servidor Web

Esta capa es la más simple de todas, debido a que la aplicación web está concebida como un Single-Page Application (SPA), lo cual implica que la mayor parte del trabajo es realizado en el cliente. Un SPA es un sitio web que está contenido en un simple archivo HTML y que se va actualizando dinámicamente mientras el usuario interactúa con ella [7], sin necesidad de recargar la página constantemente. Esto mejora la interacción con el usuario, dando la apariencia de estar usando una aplicación nativa.

Para implementar un SPA, el cliente web debe usar tecnologías como Asynchronous Javascript And XML (AJAX), web-sockets, frameworks de Model View Controller (MVC) para Javascript, entre otros. Entonces, el cliente web es

el que se encarga de renderizar la interfaz gráfica de usuario usando Javascript, mas no el Servidor. Se decidió que el Servidor siga la arquitectura de servidor delgado, lo que significa que actúa como un API de datos.

Las tareas del servidor son: conectarse a la base de datos; servir datos a los clientes; insertar, editar y eliminar registros de la base de datos y administrar la sesión de los clientes.

### 2.4.1 Implementación

Para implementar el servidor web se decidió usar Node.js. Node.js es un entorno de ejecución para Javascript construido sobre el motor V8 de Chrome. Node.js usa un modelo de entrada y salida conducido por eventos, sin bloqueo, lo que lo hace ligero y eficiente, por lo que es muy adecuado para aplicaciones de tiempo real [8].

Una aplicación en Node.js está compuesta por módulos (o paquetes), que son librerías reutilizables que se incluyen en el proyecto [9]. Existen módulos que vienen instalados junto a Node.js. Mientras que otros módulos pueden ser descargados desde NPM, el cual es un manejador de paquetes de código abierto [10].

Se decidió usar el Framework Express.js, el cual proporciona un conjunto de métodos de utilidad HTTP, HTTPS y middlewares para organizar la estructura de la aplicación web [11].

La Figura 2.4 muestra el diagrama de paquetes del servidor. El diagrama lista los módulos de Node.js y las dependencias entre módulos. La función de cada módulo es explicada a continuación.

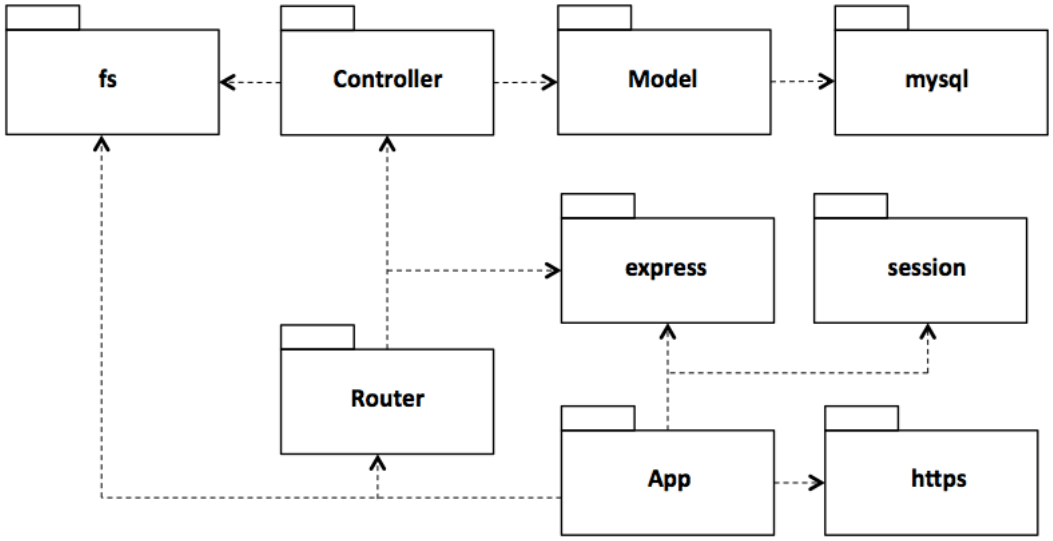



**Figura 2.4 Diagrama de paquetes del servidor web**

El módulo App es el principal: configura la aplicación e inicializa los otros módulos. Express es el framework usado para estructurar la aplicación. HTTPS se encarga de escuchar las peticiones HTTPS de los clientes web. Cuando una petición llega al servidor, ésta pasa por todos los middlewares de la aplicación. Session es un paquete que sirve para el manejo de coockies, los cuales son pequeños archivos que se usan para guardar variables de sesión. El módulo session es usado como un middleware, de tal forma que los siguientes middlewares tienen acceso a los datos de la sesión.

Router es un middleware que recibe el URL de las peticiones HTTPS y le indica al módulo Controller que operaciones realizar. Controller se encarga de procesar las peticiones de los clientes y enviarles la respuesta. Si una petición requiere realizar una operación en la base de datos, entonces usa el módulo Model. Model se encarga de conectarse a la base de datos, realizar la llamada de los storage procedures, y procesar los resultados. Model utiliza el paquete mysql el cual es un cliente de MySQL.

El paquete fs, sirve para realizar operaciones sobre archivos. App lo utiliza para abrir el archivo de configuración de la aplicación (El archivo de configuración contiene el puerto, el directorio raíz de las carpetas experimento, entre otros). Controller utiliza fs para cargar las imágenes RGB e hyperespectrales.

### 2.5 Cliente Web

El cliente web es el frontend de la aplicación. Comprende la interfaz gráfica de usuario, la lógica de negocio y la comunicación con el servidor. El cliente está compuesto por 4 páginas (archivos HTML) que son: la página de inicio de sesión, la página principal, la página con información de contacto y una página de error.

La Figura 2.5 muestra el prototipo low-fi de la página principal. La barra de título (etiquetado como A) muestra el nombre del proyecto de investigación y contiene las opciones: ayuda, contacto y cerrar sesión. El administrador de archivos (etiquetado como B) permite encontrar los experimentos de una hoja de una planta. La página contiene dos paneles (etiquetados como C) los cuales permiten visualizar la imagen rgb.tiff del experimento y realizar operaciones sobre las regiones. La barra de herramientas (etiquetada como D) contiene las opciones: acercar, alejar, seleccionar regiones, dibujar región, borrar región, deshacer, rehacer y enlazar regiones. A las regiones hay que asignarle un estadio y, un atributo que indique si la región se encuentra sobre una superficie plana; esto se lo consigue usando las opciones etiquetadas como E. Es posible cambiar el modo de visualización, usando la opción etiquetada como F, por defecto se muestra la imagen a color, pero también se puede mostrar el cubo hyperespectral. El elemento etiquetado como G, muestra si las acciones realizadas están guardadas en el servidor, o si se están guardando. La barra etiquetada como H muestra los links de las instituciones que colaboraron con el proyecto de investigación.

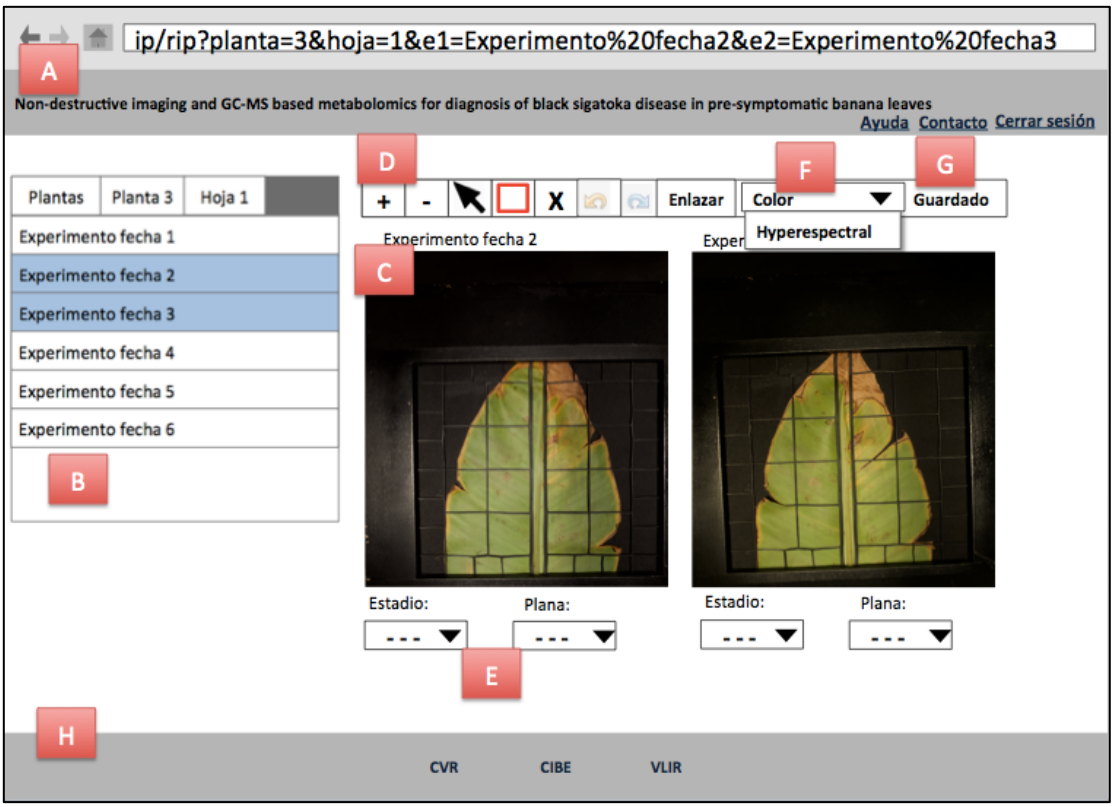



**Figura 2.5 Prototipo low-fi de la página principal**

### 2.5.1 Implementación

El cliente web está implementado en HTML5, el cual es la quinta versión del lenguaje HTML. Entre las nuevas características de HTML5 destacan: los elementos semánticos, elementos gráficos (<canvas>,<svg>) y elementos multimedia (<audio>, <video>). El uso de HTML5 implica que la aplicación es soportada por todos los navegadores modernos [12].

El documento HTML de la página principal importa archivos con código Javascript, los cuales definen el comportamiento de la página. En la Figura 2.6 se muestra un diagrama de clases simplificado de la página principal.

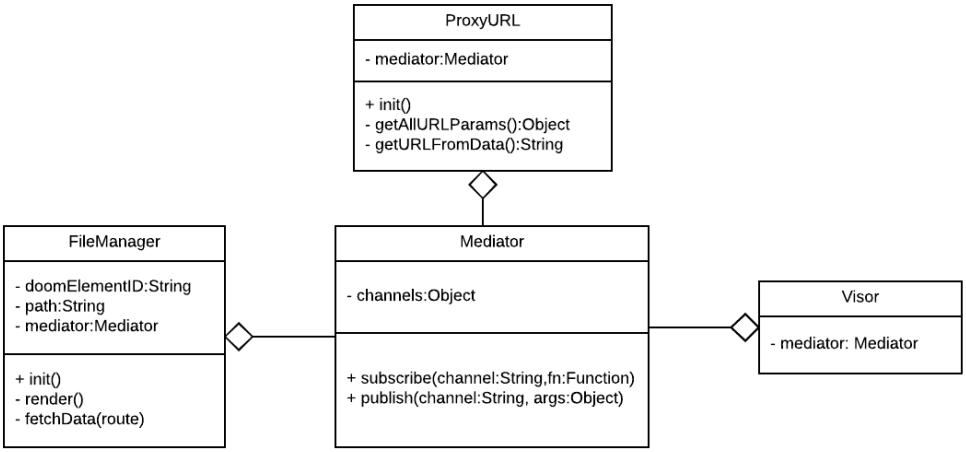



**Figura 2.6 Diagrama de clases de la página principal**

Como se observa en la Figura 2.6, existen 4 clases principales que son: FileManager, ProxyUrl, Mediator y Visor. El diagrama no muestra todos los atributos y métodos de las clases. De hecho, la clase Visor está compuesta por otras clases y subclases que no se detallan para mantener el diagrama sencillo.

La clase Mediator sirve para comunicar objetos usando un modelo publicador/subscriptor. Un objeto puede subscribirse a un tópico de interés y será notificado cada vez que un objeto publique en ese tópico. La idea de usar el Mediator es tener objetos desacoplados, de tal forma que es posible agregar o remover clases sin afectar a las otras.

La clase FileManager se encarga de controlar el administrador de archivos. El administrador de archivos simula una jerarquía en los datos mediante una ruta /plantas/planta i/hoja j/ que sirve para que el usuario localice un experimento. FileManager tiene un atributo doomElementID que contiene el identificador del elemento del DOOM en donde la vista debe ser renderizada. El atributo path es la ruta actual del administrador de archivos y sirve para pedir al servidor los elementos que pertenecen a

esa ruta. También tiene una referencia al Mediator. El método init() inicializa el objeto; el método render() renderiza la vista y, el método fetchData() obtiene los datos del servidor.

La clase ProxyUrl realiza operaciones sobre el URL de la página. Cuando la página se carga, ProxyUrl obtiene todos los parámetros del URL que son: identificador de la planta, identificador de la hoja, y los identificadores de dos experimentos. Estos parámetros obtenidos son publicados usando el Mediator, y los demás objetos se encargan de actualizarse. Cuando el FileManager cambia su ruta o cuando se cargan experimentos en los paneles, ProxyUrl actualiza el url de la página, agregando o removiendo parámetros y creando una entrada en el historial de navegación.

La clase Visor controla el visor de imágenes, el cual está compuesto de varios elementos, que son los etiquetados como C, D, E, F y G en la Figura 2.5. La Figura 2.7 muestra un diagrama de clases más completo del visor. Como se puede observar, la clase Visor está compuesta de varias clases. Tiene una instancia del objeto Mediator que sirve para la comunicación interna entre sus clases; el diagrama muestra que las clases tienen una referencia al Mediator.

Se observa en el diagrama que Visor tiene dos instancias de la clase GraphicViewer. GraphicViewer se encarga de responder a los eventos que se realizan sobre el panel (etiquetado como C en la figura 2.5). GraphicViewer tiene una instancia de la clase Panel y de la clase RegionManager. Panel se encarga de dibujar la imagen y las regiones, contiene métodos que permiten acercar o alejar la imagen, y controla unas barras de desplazamiento que desplazan la imagen. RegionManager contiene una colección de regiones y métodos que sirven para crear, borrar y editar regiones. Panel contiene una referencia a RegionManager para poder dibujar las regiones, y RegionManager contiene una referencia a Panel para indicarle que debe actualizar la vista cuando ocurre un cambio en las regiones.

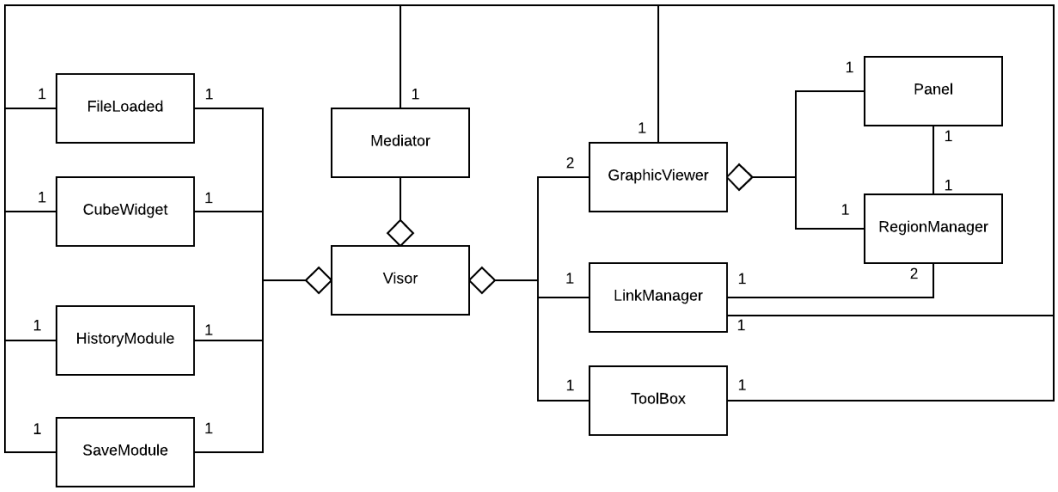



**Figura 2.7 Diagrama de clases de Visor**

GraphicViewer también contiene dos clases internas que sirven para controlar las opciones etiquetadas como E en la Figura 2.5, cuya función es asignar un estadio y el atributo plana a una región.

La clase ToolBox controla la barra de herramientas. Cuando se escoge una herramienta, ToolBox la publica con el Mediator, lo cual permite que las clases subscritas a ese tópico se comporten adecuadamente.

La clase LinkManager, se encarga de administrar los enlaces entre regiones, por lo que tiene dos referencias a los RegionManager que le permiten acceder a las regiones y crear los enlaces.

FileLoaded contiene métodos que sirven para descargar las imágenes TIFF desde el servidor. Una imagen se descarga como un arreglo de bytes, que son parseados para obtener un mapa de pixeles.

CubeWidget controla un componente que permite cambiar el modo de visualización. El modo por defecto es mostrar la imagen RGB, pero se puede cambiar y ver el cubo hyperespectral.

La clase HistoryModule mantiene dos pilas, que sirven para implementar las opciones de deshacer y rehacer. Cuando ocurren cambios en los

datos, como crear un enlace o editar regiones, HistoryModule agrega un elemento a su pila.

Por utlimo, SaveModule es una clase que implementa el auto-guardado de los datos en el servidor. Cuando ocurre un cambio en los datos, este cambio se guarda en una pila de requerimientos. Luego de un intervalo de tiempo SaveModule revisa si hay elementos en su pila y los guarda en el servidor.

De todas estas clases, la más compleja es Panel, por lo que se detalla su implementación. La clase Panel contiene una referencia a un objeto canvas, que es propio del API de Javascript, que sirve para dibujar los elementos visuales; un objeto llamado background de tipo canvas que contiene los pixeles de la imagen RGB; variables que definen el nivel de zoom y traslación de la imagen que son zoomLevel, translationX y translationY; una variable llamada factor que es un factor de escala entre las dimensiones de la imagen original y las dimensiones del visor; un objeto ScrollManager encargado de controlar las barras de desplazamiento; y una referencia a RegionManager.

El funcionamiento de Panel depende de un método llamado drawImage del objeto context que pertenece al API de Javascript y que sirve para dibujar una imagen en un canvas [13]. La Tabla 2 muestra los parámetros del método. El parámetro img define la imagen a ser dibujada en el canvas. Los parámetros sx, sy, swidth, sheight definen un área de interés de la imagen. Los parámetros x, y, width, height, definen donde colocar la imagen en el canvas. El sistema de coordenadas usado posiciona el origen en la esquina superior izquierda, el eje x es el horizontal y aumenta de izquierda a derecha y el eje y es el vertical y aumenta de arriba a abajo. La Figura 2.8 esquematiza el significado de los parámetros.

Para implementar el acercamiento (zoom) en la imagen se deben definir los argumentos de drawImage de tal forma que, el área de la imagen mostrada en el visor sea el área de interés de la imagen de origen.

Además, los parámetros x y y deben ser igual a cero y width y height deben ser igual al ancho y al alto del canvas de destino. La Figura 2.9 ejemplifica este concepto.

| Parámetro | Descripción |
|---|---|
| **img** | Imagen, canvas o video a ser dibujado. |
| **sx** | Coordenada x del recorte |
| **sy** | Coordenada y del recorte |
| **swidth** | Ancho del recorte |
| **sheight** | Alto del recorte |
| **x** | Coordenada x en donde se debe colocar la imagen |
| **y** | Coordenada y en donde se debe colocar la imagen |
| **width** | Ancho de la imagen resultante (Puede alargar o encoger) |
| **height** | Alto de la imagen resultante (Puede alargar o encoger) |

**Tabla 2: Parámetros del método drawImage**

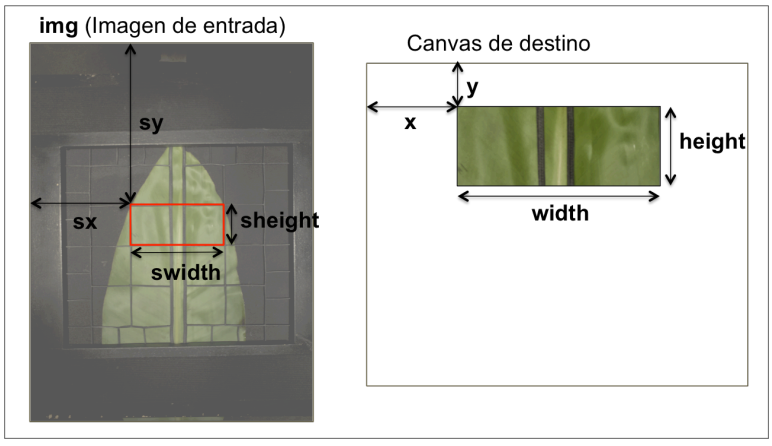



**Figura 2.8 Esquema de los parámetros de drawImage**

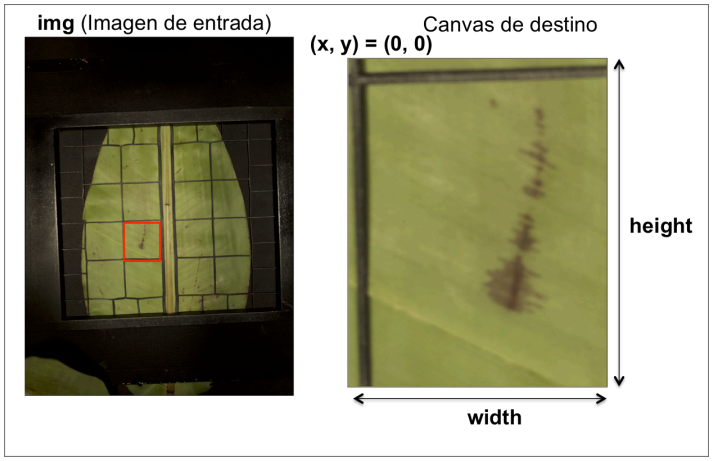



**Figura 2.9 Ejemplificación del zoom**

La variable background se asigna a Panel al llamar a su método setBackground(). En ese momento se calcula la variable factor, el cual es un factor de escala que agranda o disminuye las dimensiones de la imagen original para que se adapte al tamaño fijo del visor.

El uso de la variable factor implica que, el tamaño de una imagen mostrada en el visor no corresponde al tamaño real de la imagen, entonces fue necesario definir tres dominios o tres sistemas de coordenadas para lidiar con esto. El primer dominio es Image Domain (ID) el cual corresponde a la imagen original. El siguiente dominio es Panel Domain (PD) el cual representa la región rectangular que se muestra en el visor. En la Figura 2.9, PD sería el rectángulo rojo. Y el último dominio es Canvas Domain (CD) el cual corresponde a las coordenadas del objeto canvas del API de Javascript; este dominio es necesario para poder dibujar elementos en posiciones relativas al <canvas>.

PD está definido por las variables factor, translationX, translationY y zoomLevel. La variable factor es calculada usando la ecuación 2.1. Como se puede ver, es sólo un cociente entre el lado menor (ancho o alto) de la imagen original y el lado menor del canvas. Se toma el lado menor para que éste encaje en el visor, mientras que el lado mayor no encaja y es necesario usar las barras de desplazamiento para poder visualizar el resto.

$$factor = \frac{\min(\text{ancho ID},\ \text{alto ID})}{\min(ancho\ CD,\ alto\ CD)} \qquad (2.1)$$

Las variables translationX y translationY definen la traslación de la región de interés, la Figura 2.10 muestra el concepto. El rango de translationX es [0, ancho ID – ancho PD] y el rango de translationY es [0, alto ID – alto PD].

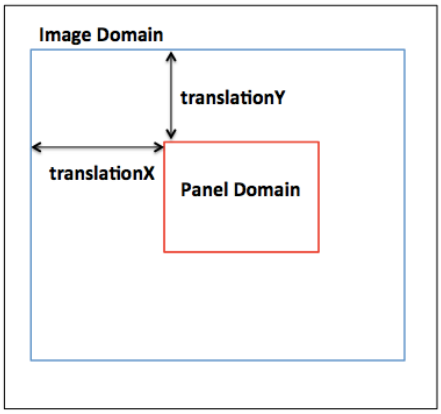



**Figura 2.10 Variables de traslación**

La variable zoomLevel es un número que define el nivel de acercamiento, ésta influye en el tamaño de los lados PD, ya que mientras más acercamiento más pequeña es la región PD. Si se tiene la ecuación 2.2 la cual transforma un punto de ID en un punto en PD. Las regiones están definidas naturalmente en ID, pero para poder dibujarlas correctamente independientemente del nivel de zoom y del desplazamiento actual, las regiones deben ser transformadas a PD.

$$punto\ PD = (punto\ ID) * factor * zoomLevel\ - translation \qquad (2.2)$$

Para crear una nueva región, el usuario debe dibujarla sobre el elemento <canvas>. El proceso de dibujado consiste en dar click en un punto del <canvas>, luego sin soltar el click ir arrastrando el mouse, y finalmente soltar el click. Durante este proceso se va obteniendo la coordenada del mouse relativa a <canvas>, y se va dibujando un rectángulo, pero éste también debe ser dibujado relativo al <canvas> para dar el efecto de que el tamaño del rectángulo está variando mientras sigue al mouse. Por eso

fue necesario definir el dominio CD, el cual simplemente no toma en cuenta las variables de PD para dibujar. La dimensión de CD es fija al igual que ID, a diferencia de PD cuya dimensión depende de factor y zoomLevel.

El acercamiento se lo realiza dando click en un punto del <canvas>, como resultado se tiene el acercamiento, con la región de interés (PD) centrado en ese punto. Para esto primero se debe transformar el punto que está en PD a un punto en ID usando la ecuación 2.3. Seguido de eso, se duplica la variable zoomLevel, para aumentar el nivel de zoom. Luego el punto en ID es transformado al nuevo PD multiplicando por el nuevo zoomLevel. A continuación, se calculan las variables de traslación de tal forma que el punto quede centrado. Si un punto está en un extremo, éste no queda centrado pero igual queda dentro del nuevo PD. Finalmente, se redibuja la escena para reflejar el cambio.

$$punto\ ID = (punto\ PD + translation)/(factor * zoomLevel) \quad (2.3)$$

El alejamiento se lo hace de forma similar al acercamiento, pero en este caso zoomLevel es dividido para dos. El rango de valores de zoomLevel es entonces $[2^0, 2^4]$. El alejamiento también queda centrado al punto donde el usuario dio click. Cuando zoomLevel es igual a 1, evidentemente, la imagen no queda centrada en ningún punto.

Para controlar las barras de desplazamiento la clase Panel tiene un objeto interno llamado ScrollManager. Esta clase usa la librería jqWidgets [14], la cual ofrece varios elementos de interfaz gráfica para web, incluyendo barras de desplazamiento. ScrollManager tiene dos instancias de jqxScrollBar una para la barra horizontal y otra para la vertical.

Cuando una barra es desplazada, ésta dispara el evento valueChanged. ScrollManager escucha éste evento, y al ser disparado actualiza su variable de traslación correspondiente. Además, se redibuja toda la escena para reflejar el desplazamiento.

ScrollManager también tiene el método updateScrollBars(), el cual debe ser llamado al realizar un acercamiento o alejamiento en la imagen, porque el rango de desplazamiento cambia y las barras deben ser actualizadas para reflejar ese cambio. Además, cuando un lado encaja con el lado del visor, la barra respectiva se oculta.

Panel también tiene una referencia al regionManager y cuenta con métodos que permiten dibujar regiones en diferentes circunstancias. Cuando una región es seleccionada, se muestran las dimensiones de la región, el estadio y unos vértices denominados anchors que permiten cambiarle el tamaño a la región.

# CAPÍTULO 3

## 3. RESULTADOS

En este capítulo se describe la aplicación desarrollada. Se muestran todas sus funcionalidades y se hace énfasis en cómo la aplicación ayuda a resolver el problema expuesto en el capítulo 1.

### 3.1 Descripción de la aplicación

Al ingresar la dirección de la página en un navegador. El servidor analiza que página debe enviarle al cliente. En el caso de que un usuario no tenga una sesión iniciada, se envía la página de inicio de sesión, la cual se puede observar en la Figura 3.1


.

**Figura 3.1 Página de inicio de sesión**

En la página de inicio de sesión, el usuario ingresa sus credenciales. Si la autenticación es correcta, el navegador redirige a la página principal (ver Figura 3.2). Si la autenticación falla, el navegador muestra la página de inicio de sesión pero con un mensaje de error (ver Figura 3.3).

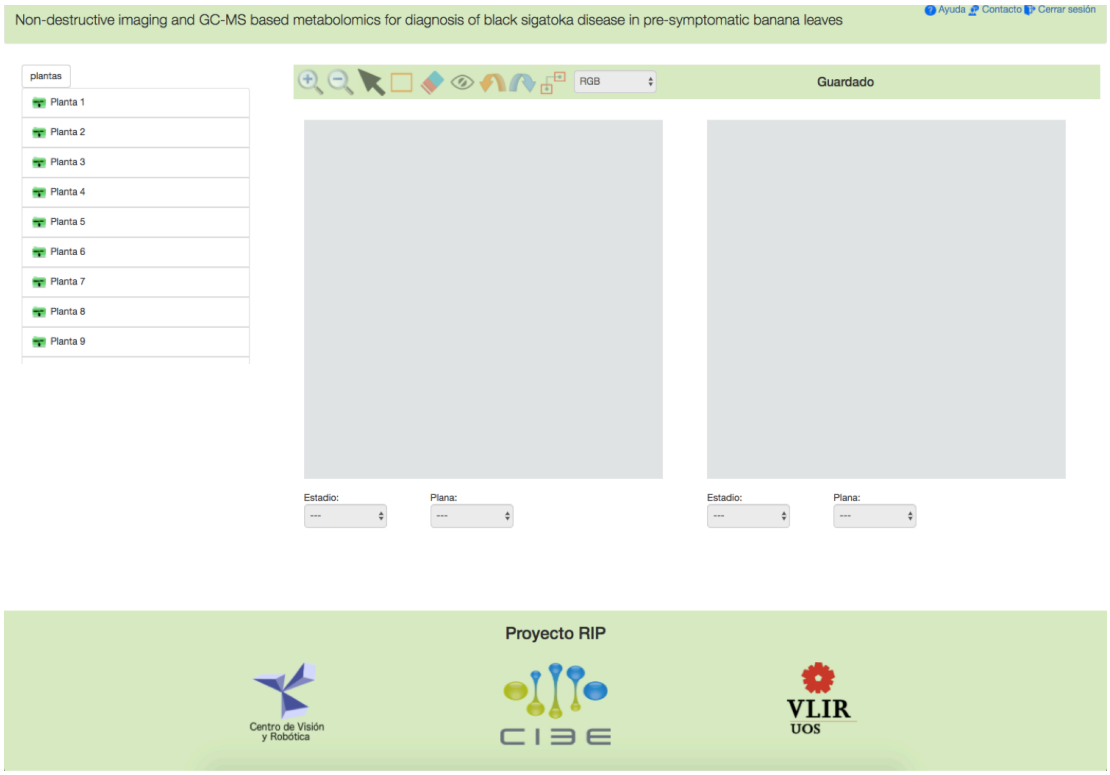



**Figura 3.2 Página principal**

La página principal es muy similar al prototipo low-fi (ver Figura 2.5). Tiene todos los componentes que fueron descritos en el capítulo 2. La Figura 3.2 muestra el estado inicial de la página, con el administrador de archivos en el nivel raíz, con la barra de herramientas desactivada, con los paneles en blanco, y con las opciones de estadio y atributo plana desactivados.

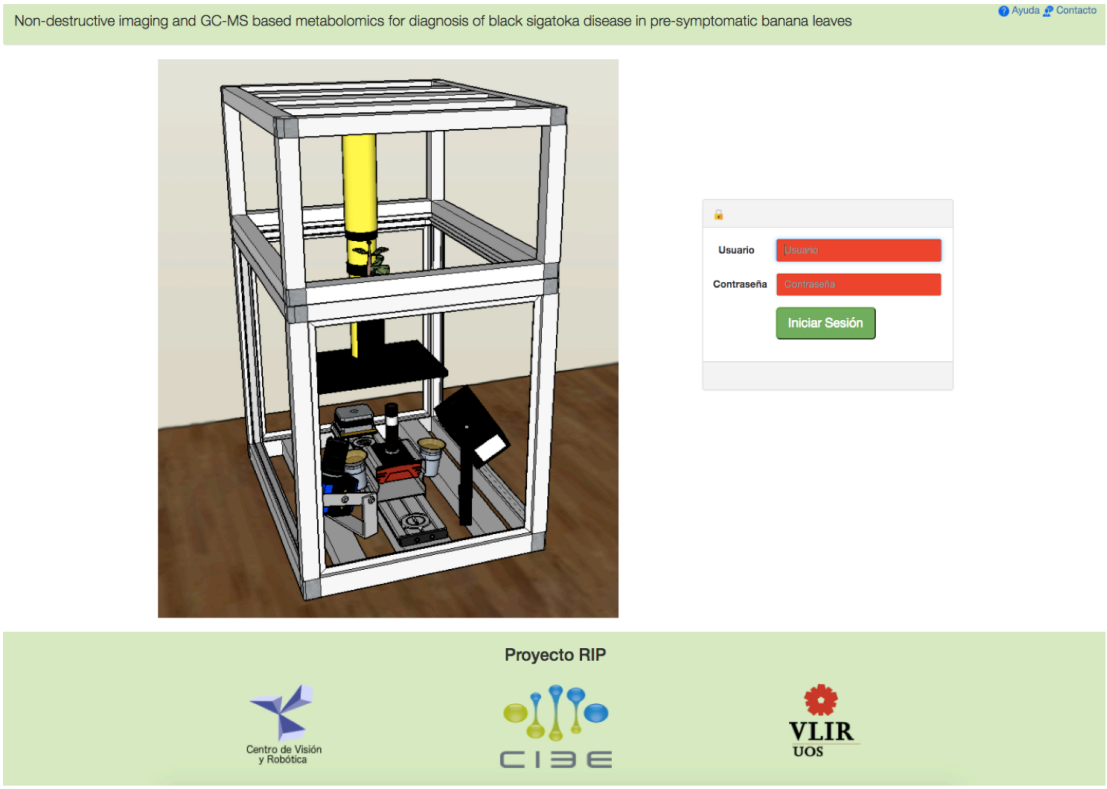



**Figura 3.3 Página de inicio de sesión con error**

El administrador de archivos tiene tres niveles. En el nivel raíz se muestra la lista de plantas, en donde cada elemento tiene el ícono de una carpeta con una planta y el identificador de la planta. Al seleccionar una planta dándole click, se muestra la lista de hojas que pertenecen a esa planta, en donde cada elemento tiene el ícono de una carpeta con una hoja y el nombre de la hoja. Así mismo, al seleccionar una hoja, se muestra la lista de experimentos de esa hoja, en donde cada elemento tiene el ícono de un archivo TIFF y el nombre del experimento.

En cada nivel del administrador de archivos, la barra de ruta se actualiza. La barra de ruta está compuesta por lengüetas que al darle click se redirige a ese nivel, lo cual permite regresar a los niveles anteriores.

En el nivel de los experimentos, los elementos de la lista son arrastrables. Para cargar un experimento en un panel, hay que darle click y sin soltar el click se lo debe arrastrar hasta el panel que se desea utilizar, y en ese instante soltar el

click. Luego de eso el panel muestra un mensaje que indica que está cargando los datos y al terminar de cargar muestra la imagen RGB con sus regiones (ver Figura 3.4).

Cuando se carga un experimento en un panel, se activa la barra de herramientas. Si se tiene cargado un experimento y en el administrador de archivos se regresa a un nivel anterior, el experimento se descarga y se regresa al estado inicial.

Como se observa en la Figura 3.4 las regiones son dibujadas de color blanco, pero al cargar dos experimentos, se trae los datos de los enlaces que conectan esas regiones. Y las regiones que estén enlazadas son mostradas del mismo color. La Figura 3.5 muestra dos experimentos consecutivos cargados y se observa como las regiones están coloreadas.

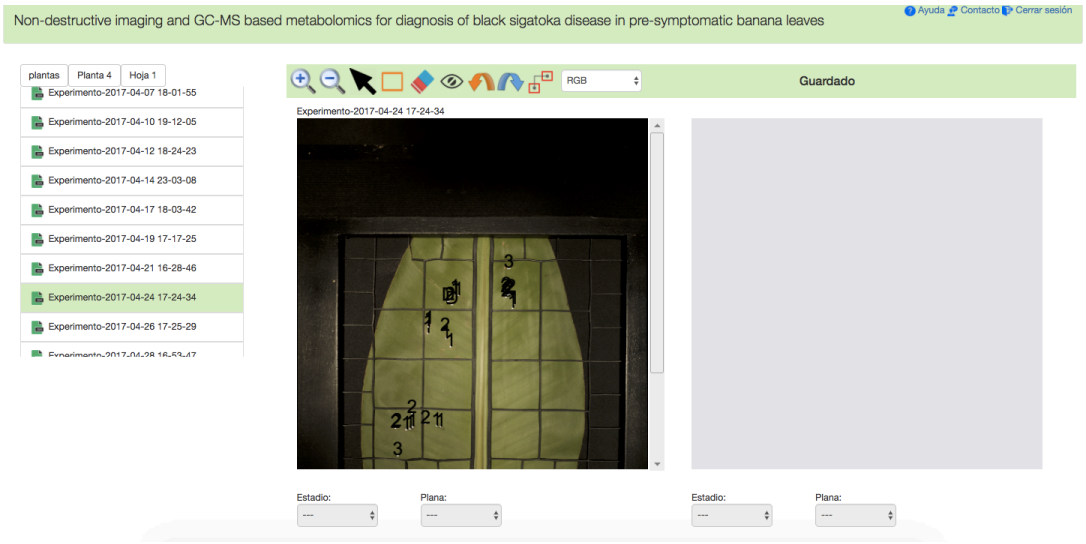



**Figura 3.4 Panel mostrando un experimento.**

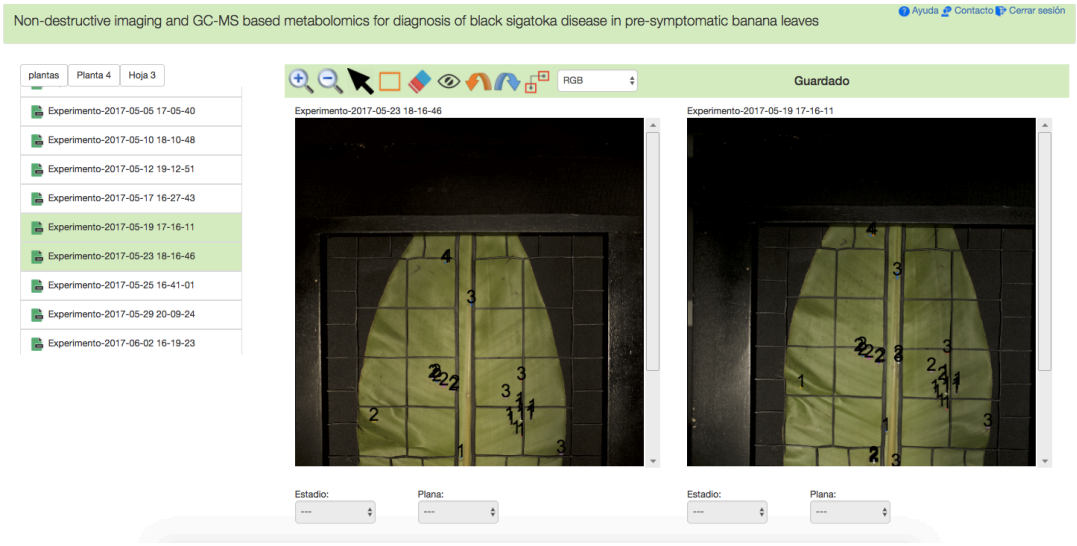



**Figura 3.5 Panel mostrando dos experimentos.**

Las opciones de la barra de herramienta son: acercar, alejar, seleccionar regiones, dibujar región, borrar regiones, ocultar/mostrar estadio, deshacer, rehacer y enlazar regiones.

Al hacer click sobre el botón acercar, éste se queda marcado y el cursor del mouse se muestra como una lupa con signo más. Si se da click sobre un punto del panel, éste mostrará la imagen acercada y centrada en ese punto. La Figura 3.6 muestra esta funcionalidad.

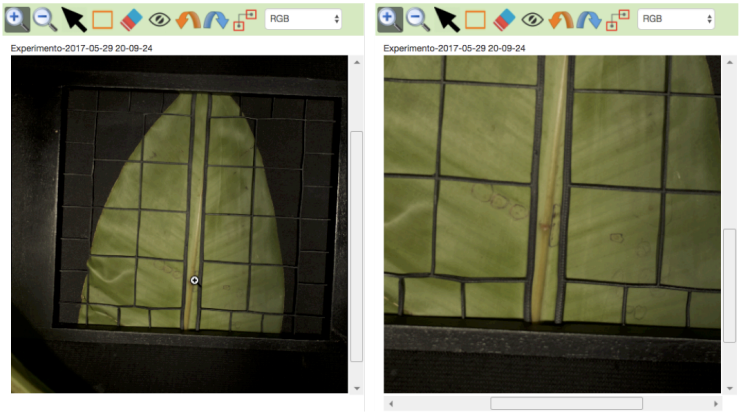



**Figura 3.6 Herramienta acercar**

La herramienta alejar funciona de manera similar a acercar, pero en este caso, se queda seleccionado el botón alejar y el cursor se muestra como una lupa con un signo menos. Se permiten los niveles de acercamiento 1, 2, 4, 8 y 16.

La opción seleccionar regiones permite realizar varias acciones sobre las regiones. Al dar click sobre el botón éste se queda marcado y el cursor del mouse se muestra como una flecha. La primera acción que se puede realizar es seleccionar una región, para esto se debe dar click sobre una región y ésta quedará seleccionada. También se puede seleccionar varias regiones dibujando un área que contenga todas las regiones. La Figura 3.7 muestra cómo se realiza la selección múltiple, se observa también que las regiones seleccionadas tienen unos vértices y un cuadro que muestra su tamaño en pixeles.


**Figura 3.7 Selección múltiple de regiones**

Una vez que las regiones estén seleccionadas, la herramienta permite cambiarlas de posición, para esto se deben arrastrar las regiones. Además se puede cambiar el tamaño de las regiones arrastrando los vértices, pero no se permite obtener regiones de ancho o alto cero.

Dibujar región permite dibujar un rectángulo sobre la imagen. Al escoger esta opción el cursor muestra una cruz. Mientras se va dibujando la región se muestra su tamaño en pixeles. La Figura 3.8 muestra el proceso de dibujo. Al crear una nueva región ésta tiene los bordes blancos porque aún no está enlazada.


**Figura 3.8 Proceso de creación de una región**

Para borrar una o varias regiones, primero hay que seleccionarlas y luego dar click en borrar regiones. Está opción no se queda marcada ni cambia el cursor porque no representa un cambio de herramienta.

Al dar click sobre la opción ocultar/mostrar estadio, ésta se selecciona o deselecciona según sea el caso. Mientras la opción está seleccionada, las letras que indican el estadio de una región se ocultan, permitiendo ver mejor la imagen. Si la opción está deseleccionada, se muestran los estadios. En la Figura 3.9 se observa ambos casos.

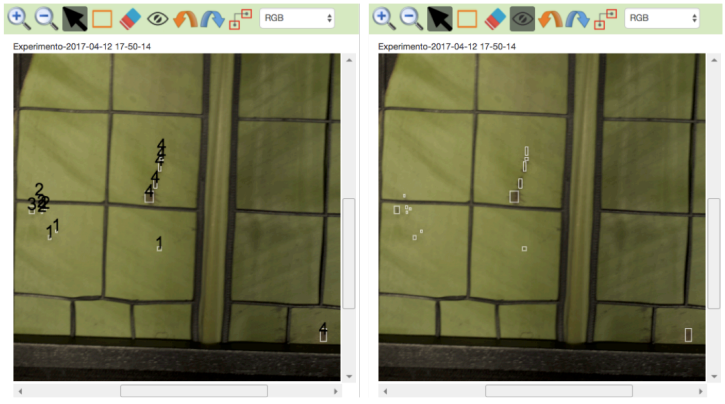



**Figura 3.9 Función de la herramienta ocultar/mostrar estadio**

Las opciones deshacer y rehacer, permiten restaurar cambios realizados anteriormente. Las acciones soportadas son: crear región, borrar regiones, asignar estadios, asignar atributo plana, enlazar regiones y desenlazar regiones. Se puede deshacer o rehacer hasta 10 acciones pasadas. Cuando se carga otro experimento, las pilas de deshacer y rehacer se vacían.

Para enlazar dos regiones, se debe escoger la opción enlazar región, luego dar click a una región para que se seleccione, y finalmente click en la otra región. Al terminar de enlazar las dos regiones se colorean de un mismo color. Para desenlazar se debe dar click en una región enlazada y click en la otra región, como resultado ambas regiones se colorearan de blanco. También se puede enlazar dos o más regiones de un experimento con una región de otro experimento por lo explicado en el Capítulo 2 (ver Figura 2.3).

Aparte de permitir dibujar las regiones, se debe poder asignarle un estadio. Para realizar esto se debe seleccionar regiones, esto activa las opciones para asignar estadio, entonces se debe seleccionar un estadio del menú y las regiones seleccionadas reflejarán ese cambio. La Figura 3.10 muestra el proceso de asignar un estadio.


**Figura 3.10 Asignación de estadio a regiones**

Asignar el atributo plana a una región, es similar a asignar el estadio pero se utiliza la opción plana, que permite escoger entre Si y No. Por defecto, todas las regiones tiene este atributo en Si.

Todas las opciones revisadas permiten realizar todo el proceso de etiquetado y enlazado, y son suficiente para generar el conjunto de datos del proyecto de investigación del CVR. Una vez que se tiene todas las regiones, se les aplica una transformación. Este proceso se lo realiza fuera de línea y no forma parte del proyecto de materia integradora. Pero, sí es posible visualizar las regiones transformadas, para lo cual se debe cambiar el modo de visualización de RGB a Hyperespectral.

En el modo de visualización Hyperespectral, se muestra una imagen del cubo hyperespectral y, aparecen nuevas opciones que permiten seleccionar una longitud de onda para que su imagen correspondiente sea cargada en el panel. También se muestran las regiones transformadas. Y en la barra de herramientas sólo están disponibles las opciones de acercamiento y alejamiento. La Figura 3.11 muestra la aplicación en este modo.

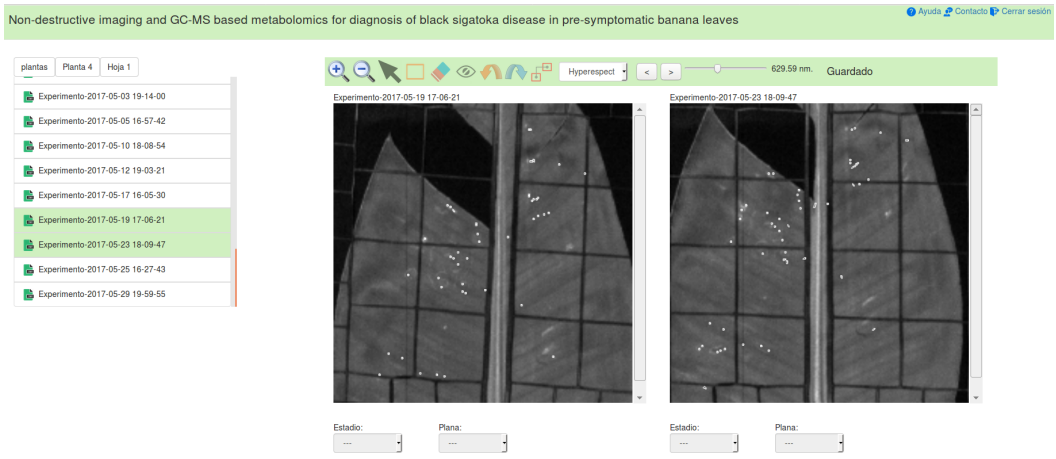



**Figura 3.11 Modo de visualización hyperespectral**

## 3.2 Evaluación de la aplicación

Para realizar las pruebas de software se usó Jasmine, que es un framework de pruebas para Javascript [15]. La Figura 3.12 es una captura de pantalla de uno de los reportes de Jasmine. Por cada clase o módulo a ser probado se escribió una especificación. Una especificación es un archivo en donde se escriben las pruebas mediante aserciones en las cuales se comprueba si una función esta correcta, probando diferentes casos de prueba y comparando los resultados obtenidos con un resultado esperado. Las funcionalidades más críticas de la aplicación fueron probados con Jasmine, si las pruebas fallaban se corregía hasta que todas las pruebas fueran aprobadas.


**Figura 3.12 Reporte de pruebas de Jasmine**

La aplicación ha sido instalada en un computador del CVR. Considerando que las imágenes son de alta resolución y ocupan en disco 56 MB, cargar una imagen desde la red del CVR toma menos de 5 segundos, pero en otras redes el tiempo puede ser de hasta 40 segundos. Aun así, las pruebas con un biólogo demostraron que es capaz de utilizar la aplicación.

El CVR cuenta con un dataset de cuarenta y nueve hojas que fueron escaneadas por tres meses. En total, se tienen seiscientas ochenta y tres carpetas experimento que ocupan 600 GB en disco duro. Hasta el momento de redactar este documento, el biólogo ha etiquetado y enlazado las regiones de 11 hojas.

# CONCLUSIONES Y RECOMENDACIONES

La aplicación permite a un biólogo realizar el proceso de etiquetado y etiquetado en las plantas. Se han tomado en cuenta las sugerencias del biólogo para el diseño final con el objetivo de que la tarea se agilice.

Debido al diseño de software de esta aplicación, es posible realizar con poco esfuerzo otra aplicación similar que permita etiquetar otro tipo de imágenes para otro problema. También se pueden desarrollar sitios con otras variantes; como, por ejemplo, mostrar en los paneles una imagen RGB y una espectral con el fin de seleccionar puntos de control para realizar la alineación de las imágenes.

Se debe realizar esfuerzos para mejorar la escalabilidad. La carga de las imágenes aún es muy lenta. Si se tienen muchos usuarios conectados, el rendimiento se afecta drásticamente. Una solución simple puede ser aumentar las capacidades del hardware del servidor. También se podrían tener varias instancias del servidor web y la base de datos para balancear la carga.

Otra mejora para el sitio es que sea multi-usuario; en las condiciones actuales existe una restricción para que varios usuarios no puedan acceder al mismo experimento simultáneamente. Por lo que implementar un software colaborativo puede ser una solución para que varios usuarios trabajen sobre una misma imagen.

# BIBLIOGRAFÍA

[1] E. Fouré, “Black Leaf Streak Disease of Bananas and Plantains (Mycosphaerella fijiensis Morelet). Study of the symptoms and stages of the disease in Gabon”, París, 1985.

[2] D. Marín et al: “An Increase Threat to banana cultivation” en Plant Disease, 2003. © The American Phytopathological Society.

[3] D. Sun, “Hyperespectral Imaging for Food Quality Analysis and Control”. London: Academic Press, 2010.

[4] D. Ochoa et al. “Hyperspectral imaging system for disease scanning on banana plants” en SPIE, 2016. doi: 10.1117/12.2224242

[5] MySQL™. MySQL 5.7 Reference Manual [Online]. Disponible en: https://dev.mysql.com/doc/refman/5.7/en/

[6] MySQL™. Working with Stored Procedures [Online]. Disponible en: https://dev.mysql.com/doc/connector-net/en/connector-net-tutorials-stored-procedures.html

[7] M. Wasson. (2013, Noviembre) ASP.NET - Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET [Online]. Disponible en: https://msdn.microsoft.com/en-us/magazine/dn463786.aspx

[8] M. Cantelon et al, “Node.js In Action”, Shelter Island, NY: Manning Publications, 2014

[9] W3Schools. Node.js Modules [Online] Disponible en: https://www.w3schools.com/nodejs/nodejs_modules.asp

[10] Node.js®. [online]. Disponible en: https://nodejs.org/es/

[11] Express [Online]. Disponible en: http://expressjs.com/es/

[12] W3Schools. HTML5 Introduction [Online]. Disponible en: https://www.w3schools.com/html/html5_intro.asp

[13] MDN Web Docs. CanvasRenderingContext2D.drawImage() [Online]. Disponible en: https://developer.mozilla.org/es/docs/Web/API/CanvasRenderingContext2D/drawImage

[14] JQWidgets [Online]. Disponible en: http://www.jqwidgets.com

[15] Jasmine [Online]. Disponible en: https://jasmine.github.io/