UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

VitiScan: Detección de gotas de producto antifúngico en hojas de vid
Documentación Técnica

Presentado por David Merinero Porres
en Universidad de Burgos — 9 de julio de 2024
Tutor: Carlos Cambra Baseca

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se va a analizar la planificación temporal y se va a estudiar la viabilidad económica y legal del proyecto.

A.2. Planificación temporal

Para la planficación temporal se han utilizado los Sprints de la metodología Scrum. Antes de empezar cada Sprint es necesario realizar una reunión para decidir la duración del mismo, el objetivo del mismo y las tareas que se van a llevar a cabo para cumplirlo. Una vez transcurrido ese tiempo se vuelve a celebrar una reunión y se actualizan los objetivos y tareas en función de lo realizado en el Sprint anterior. Y así sucesivamente hasta finalizar el desarrollo del proyecto software.

Sprints

Este apartado contiene los diferentes Sprints con las tareas que se han ido realizando en cada uno.

Sprint Inicial

14/03/24 - 17/04/24

- Subir plantilla de la documentación a GitHub.

- Crear el fichero Python.
- Registro en Overleaf.
- Registro en Zube.
- Buscar información sobre Streamlit.
- Crear un prototipo de la aplicación.
- Añadir más funcionalidades al prototipo.
- Buscar información sobre el tratamiento de imágenes con OpenCV.
- Implementar las transformaciones a las imágenes.

Sprint 2

18/04/24 - 10/05/24

- Añadir contenido inicial a la memoria.
- Cambiar canales RGB de las imágenes.
- Eliminar fondo de las imágenes.
- Determinar el porcentaje de la superficie de la hoja.
- Detectar el borde de las gotas.
- Determinar el porcentaje de la superficie de las gotas respecto a la hoja.

Sprint 3

11/05/24 - 20/05/24

- Descargar imágenes hiperespectrales.
- Buscar información sobre las imágenes hiperespectrales y cómo trabajar con ellas.
- Crear método para cargar imágenes hiperespectrales.
- Crear método para visualizar las diferentes capas de la imagen hiperespectrales.
- Añadir documentación sobre las imágenes hiperespectrales.

Sprint 4

21/05/24 - 28/05/24

- Crear pestaña de trinarización.
- Crear pestaña de visualización..

Sprint 5

29/05/24 - 06/06/24

- Buscar librería de python para visualizar imágenes y poder editar parámetros.
- Poner la opción de elegir bandas en la pestaña Trinarizar.
- Mostrar porcentajes y guardarlos en un fichero .csv en la pestaña Trinarizar.
- Cambiar disposición de elementos en ambas pestañas.
- Cambiar el aspecto de los widgets de subir archivos y convertirlos en un botón.
- Desactivar la animación de carga de Hydralit.
- Crear el botón .Añadir Capas.en una ventana emergente.

Sprint 6

07/06/24 - 28/06/24

- Añadir aspectos teóricos y herramientas utilizadas a la memoria.

Sprint 7

29/06/24 - 03/07/24

- Implementar en la pestaña Visualizar el editor de imágenes con las capas.
- Carga de datos por defecto por fichero en la pestaña Trinarizar.
- Añadir información sobre el proyecto y su instalación a los

- Crear la pestaña de Trinarizar por capas. anexos.
- Añadir transformaciones a la imagen trinarizada para eliminar ruido.
- Creación del fichero estilos.css.

Sprint Final

04/07/24 - 09/07/24

- Terminar el resto apartados de la documentación.
- Desplegar la página web.
- Documentación interna del código.

A.3. Estudio de viabilidad

En este apartado se va realizar un análisis de la viabilidad del proyecto, teniendo en cuenta tanto el apartado económico, como el legal.

Viabilidad económica

Antes de comenzar el desarrollo un proyecto software, es necesario hacer un cálculo de los gastos y de los posibles beneficios que nos vaya a aportar. Una vez calculados estos datos se podrá discernir si el proyecto es viable o no (desde un punto de vista puramente económico). En los siguientes apartados se van a detallar los costes que supondrá el desarrollo de este proyecto.

Coste de Hardware

El hardware utilizado para la realización de este proyecto ha sido un ordenador portátil y un ratón inalámbrico.

- Ordenador portátil MSI: 1200€
- Ratón inalámbrico Logitech: 50€

Se va a suponer una amortización a 5 años, por lo que el coste anual para el portátil sería de 1200€/5= 240€ y el del ratón sería de 50€/5=10€.

Coste de Software

El único software de pago que se ha utilizado en este proyecto es el sistema operativo del ordenador portátil, Windows 10 Pro.

- Windows 10 Pro: 200€

Se va a suponer una amortización a 5 años, por lo que el coste anual para el software, Windows 10 Pro en este caso sería de 200€/5=40€.

Coste de Personal

Aquí vamos a considerar que estoy contratado por una empresa para la realización de este proyecto. Partiendo de esta base, se va a realizar una aproximación de los costes que les supondría contratar a un programador junior.

- Sueldo anual de programador junior: 22000€

se va a suponer que se reparte el sueldo en 14 pagas y que se ha trabajado a media jornada, por lo que sería un sueldo mensual de 22000€/14 pagas = 1571,43€, pero como es a media jornada sería la mitad. Por lo que el sueldo mensual percibido corresponde a un total de 787,71€. El coste anual sería de 787,71€ X 12 meses = 9428,57€.

Otros Costes

En este apartado se incluyen gastos extras que han aparecido a lo largo de la realización del proyecto.

- 3 Pendrives entrega: 30€

Estos gastos se amortizan en un año, por lo que valdrá 30€.

Coste Total de un año de desarrollo

Finalmente vamos a sumar todos los costes para ver cuánto dinero supondría la realización de este proyecto durante un año completo.

Tras realizar los cálculos de gastos, no se ve viable el proyecto en términos económicos, ya que como no se consiga desarrollar un software muy potente y se pueda comercializar a los dueños de viñedos, para que reduzcan el uso de fertilizantes en sus campos, no se ve otra forma de conseguir ya no beneficio económico, sino simplemente cubrir los costes.

Recurso	Coste Anual
Windows 10 Pro	40€
Portátil	240€
Ratón	10€
Sueldo	9428,57€
Pendrives	30€
Total:	9748,57€

Tabla A.1: Coste anual por recursos.

Viabilidad legal

En este apartado se van a revisar las licencias que poseen las diferentes librerías que se han utilizado en el proyecto, en qué consiste cada una, y también se va a explicar qué licencia encaja mejor con las características y finalidad de este proyecto.

Licencias Software

A continuación, se hace una breve descripción de las licencias que usan las librerías y se adjunta una tabla con las diferentes librerías, su versión y su licencia.

- **Licencia MIT**: La licencia MIT es una de las licencias de código abierto más permisivas. Permite el uso, modificación y distribución del software sin restricciones, siempre que se mantenga el aviso de derechos de autor original. No proporciona garantías ni responsabilidad a los autores por el uso del software.

- **Apache License 2.0**: La Apache License 2.0 es otra licencia de código abierto muy utilizada. Al igual que la licencia MIT, permite el uso, modificación y distribución del software de forma gratuita. Sin embargo, requiere mantener los avisos de derechos de autor y licencia tanto en el código original como en las versiones modificadas. También exime a los autores de cualquier responsabilidad o garantía sobre el software.

- **BSD-3-Clause License**: La BSD-3-Clause License es similar a las anteriores, permitiendo el uso, modificación y distribución del software de forma libre. A diferencia de la Licencia MIT, requiere que se indique

si se han realizado cambios al trabajo original. Al igual que las otras, no ofrece garantías y exime a los autores de responsabilidad. Una característica destacada de esta licencia es su compatibilidad con otras licencias de código abierto, como la GPL.

Librería	**Versión**	**Licencia**
hydralit	1.0.14	Apache License 2.0
numpy	1.26.4	Licencia BSD modificada
opencv-python-headless	4.10.0.84	Apache License 2.0
pandas	2.2.1	BSD 3-Clause License
scikit-image	0.24.0	BSD 3-Clause License
spectral	0.23.1	Licencia MIT
streamlit	1.36.0	Apache License 2.0
streamlit_image_zoom	0.0.4	Licencia MIT

Tabla A.2: Librerías utilizadas, su versión y la licencia que usan.

Tras analizar las diferentes licencias que utilizan las librerías y conocer sus características, se ha llegado a la conclusión de que la licencia MIT es la más adecuada para el proyecto. Esta elección se basa en que la aplicación se ha desarrollado para apoyar una investigación, y permitir que cualquier persona haga lo que desee con el código, siempre y cuando se cite al autor, facilita la colaboración y las mejoras del proyecto.

Apéndice B

Especificación de Requisitos

B.1. Introducción

En esta sección se va a realizar una descripción del sistema software a desarrollar. Para ello se han creado tres apartados:

- El primero describe los objetivos generales del proyecto.
- El segundo muestra un listado de los requisitos funcionales y no funcionales.
- El tercero trata los diferentes casos de uso del sistema software.

B.2. Objetivos Generales

1. Poder procesar las imágenes hiperespectrales de las hojas de vid con la aplicación en gotas de diferentes productos antifúngicos con base de cobre y azufre, teniendo cada gota una concentración diferente del producto.
2. Poder extraer información de estas imágenes hiperespectrales. Principalmente conocer píxel por píxel si se trata de hoja(01), fondo(00) o gota(10) con producto antifúngico. A esta asignación a cada píxel de uno de los tres valores se ha denominado Trinarización. Después esta información se guarda en un fichero .csv. Lo siguiente es la creación de una imagen pintando cada valor de la trinarización de un color diferente, generándose la imagen trinarizada. Finalmente, se crea otro

archivo .csv, este contiene el porcentaje de hoja que hay en la imagen y el porcentaje de gotas con producto que hay en la hoja de vid.

B.3. Catálogo de requisitos

Una vez sabemos los objetivos del proyecto vamos a pasar a definir los requisitos:

Requisitos Funcionales

Los requisitos funcionales son aquellos que definen que acciones debería poder llevar a cabo la aplicación.

- **RF-1 Carga de datos:**
  - **RF-1.1 Carga de imágenes hiperespectrales:** Permitir al usuario subir archivos de imágenes hiperespectrales en formato .bil y .bil.hdr.
  - **RF-1.2 Carga de imágenes estándar**: Permitir al usuario subir archivos de imágenes estándar en formatos .jpg, .jpeg, .png, .tiff y .tif.
  - **RF-1.3 Carga de imágenes trinarizadas**: Permitir al usuario subir archivos de imágenes que han sido trinarizadas previamente.
  - **RF-1.4 Carga de imágenes de un directorio origen**: Permitir al usuario seleccionar un directorio de origen para subir múltiples imágenes hiperespectrales de una sola vez.
  - **RF-1.5 Carga de bandas predeterminadas desde un fichero XML**: Permitir al programa cargar valores predeterminados para seleccionar bandas de una imagen hiperespectral desde un fichero XML.
- **RF-2 Configuración de parámetros de procesamiento:**
  - **RF-2.1 Introducción de nombre para los ficheros resultantes de la trinarización**: Permitir al usuario ingresar el nombre de los archivos resultantes del proceso de trinarización que se vayan a guardar.
  - **RF-2.2 Introducción del número de banda de la imagen hiperespectral**: Permitir al usuario ingresar el número de banda específico de la imagen hiperespectral que desea procesar.

  - **RF-2.3 Introducción del rango del umbral para la binarización de las gotas** : Permitir al usuario definir el rango del umbral que se utilizará para binarizar las gotas de producto respecto de la hoja de vid.

- **RF-3 Procesamiento de imágenes:**

  - **RF-3.1 Binarizar hoja:** El programa debe ser capaz de binarizar la hoja respecto al fondo de la imagen.
  - **RF-3.2 Binarizar gotas de la hoja:** El programa debe ser capaz de binarizar las gotas de producto fúngico presentes en la hoja de vid.
  - **RF-3.3 Eliminar ruido de las gotas detectadas:** El programa debe ser capaz de eliminar el ruido de las gotas de producto detectadas en la imagen.
  - **RF-3.4 Crear imagen trinarizada:** El programa debe ser capaz de crear una imagen trinarizada que combine la información de las hojas y las gotas.

- **RF-4 Visualización de imágenes:**

  - **RF-4.1 Añadir capas al visualizador de imágenes:** Permitir al usuario añadir diferentes capas de imágenes al visualizador.
  - **RF-4.2 Eliminar capas del visualizador de imágenes:** Permitir al usuario eliminar capas del visualizador de imágenes.
  - **RF-4.3 Alternar selección de capas en el visualizador de imágenes:** Permitir al usuario seleccionar y deseleccionar las capas en el visualizador.
  - **RF-4.4 Ajustar la transparencia de las capas:** Permitir al usuario ajustar la transparencia de las capas mediante un slider.
  - **RF-4.5 Hacer zoom en la imagen resultante:** Permitir al usuario hacer zoom sobre la imagen visualizada utilizando la rueda del ratón.

- **RF-5 Exportación de resultados:**

  - **RF-5.1 Guardar la imagen en el visualizador**: Permitir al usuario guardar la imagen que ha editado en el visualizador.
  - **RF-5.2 Guardar imagen y archivos .csv en la pestaña trinarizar**: Permitir al usuario guardar la imagen y exportar

archivos .csv con los datos generados durante el proceso de trinarización.

- **RF-5.3 Guardar imagenes trinarizadas en un directorio destino**: Permitir al usuario guardar imágenes trinarizadas por lotes en un directorio especificado.
- **RF-5.4 Cargar imagen trinarizada en una lista**: Permitir al usuario cargar la imagen trinarizada generada en una lista para su posterior visualización.

- **RF-6 Interfaz de usuario:**

  - **RF-6.1 Ventana emergente para elegir las nuevas capas:** Proporcionar una ventana emergente para la selección de nuevas capas de imágenes.
  - **RF-6.2 Navegación entre diferentes pestañas:** Permitir al usuario navegar entre las diferentes pestañas del programa.
  - **RF-6.3 Botones para realizar diferentes acciones:** Incluir botones que permitan al usuario realizar distintas acciones de manera dinámica.
  - **RF-6.4 Slider para elegir un rango:** Incluir un slider que permita al usuario seleccionar un rango de valores.
  - **RF-6.5 Barra de carga para mostrar el progreso:** Mostrar una barra de carga que indique el progreso de las operaciones en curso.

Requisitos No Funcionales

Los requisitos funcionales son aquellos que definen que acciones debería poder llevar a cabo la aplicación.

- **RNF-1 Rendimiento:** El programa tiene que responder rápido a las peticiones del usuario y no tardar mucho en realizar las operaciones.
- **RNF-2 Seguridad:** El programa tiene que ser seguro y que su uso no suponga un problema para los usuarios.
- **RNF-3 Privacidad:** El programa debe manejar correctamente los datos de los usuarios y proteger aquellos que sean privados.

- **RNF-4 Usabilidad:** El programa debe ser intuitivo y fácil de utilizar para que el usuario final sea capaz de aprovechar todas las funcionalidades que ofrece.
- **RNF-5 Compatiblidad:** El programa debe ser compatible con el mayor número de navegadores posibles, por no decir que con todos.
- **RNF-6 Mantenimiento:** El programa debe ser tener un código bien documentado y modular para facilitar su mantenimiento o de cara a añadir nuevas funcionalidades.

B.4. Especificación de requisitos

En este apartado se han definido los diferentes casos de usos del usuario en la aplicación, creando una tabla para cada uno. Estos son los casos de uso:

CU-1	**Añadir capas al visualizador**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-1, RF-1.1, RF-1.2, RF-1.3, R-2.2, R-4.1, RF-6.1, RF-6.3
Descripción	Permite al usuario añadir capas al visualizador usando diferentes formatos de imagen
Precondición	-
Acciones	1. El usuario pulsa el botón de AÑADIR CAPAS. 2. El usuario elige en el desplegable el formato de la imagen que va a subir. 3. Si es trinarizada elige una de las imágenes disponibles en un desplegable. 4. El usuario pulsa en el widget para subir la imagen. 5. El usuario selecciona la imagen que va a subir. 6. Si la imagen es hiperespectral se elige la banda. 7. La imagen se añade a la lista como una nueva capa.
Postcondición	-
Excepciones	-
Importancia	Alta

Tabla B.1: CU-1 Añadir capas al visualizador.

CU-2	**Editar la imagen en el visualizador**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-4, RF-4.1, RF-4.2, RF-4.3, RF-4.4, RF-4.5
Descripción	El usuario puede seleccionar las capas que quiere ver, cambiarlas la transparencia, hacer zoom y añadir o eliminar capas.
Precondición	-
Acciones	1. Pasos del CU 2. Pasos del CU (añadir tantos como sean necesarios)
Postcondición	-
Excepciones	-
Importancia	Alta

Tabla B.2: CU-2 Editar la imagen en el visualizador.

CU-3	**Eliminar todas las capas**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-4.2
Descripción	El usuario elimina todas las capas pulsando el botón ELIMINAR TODAS.
Precondición	Existencia de capas.
Acciones	1. El usuario pulsa el botón de ELIMINAR TODAS. 2. Se eliminan todas las capas disponibles.
Postcondición	-
Excepciones	-
Importancia	Baja

Tabla B.3: CU-3 Eliminar todas las capas.

CU-4	**Hacer zoom en la imagen resultante**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-4.5
Descripción	La descripción del CU
Precondición	Existencia de al menos una capa seleccionada.
Acciones	1. El usuario pone el cursor encima de la imagen. 2. El usuario mueve la rueda del ratón para aumentar o disminuir el zoom de la imagen.
Postcondición	-
Excepciones	-
Importancia	Media

Tabla B.4: CU-4 Hacer zoom en la imagen resultante.

CU-5	**Guardar imagen resultante del visualizador**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-5.1, RF-6.3
Descripción	El usuario pulsa el botón Guardar imagen para almacenar la imagen que ha editado.
Precondición	Existencia de al menos una capa seleccionada.
Acciones	1. El usuario pulsa el botón Guardar imagen 2. La imagen se descarga en la memoria del usuario.
Postcondición	-
Excepciones	-
Importancia	Alta

Tabla B.5: CU-5 Guardar imagen resultante del visualizador.

CU-6	**Cargar imagen hiperespectral en trinarizar**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-1.1, RF-6.3
Descripción	Permite al usuario cargar una imagen hiperespectral al programa, pudiendo elegir los ficheros de forma interactiva desde el explorador de archivos.
Precondición	-
Acciones	1. El usuario pulsa en el widget de subida de archivos. 2. El usuario elige los ficheros de la imagen hiperespectral. 3. La imagen hiperespectal queda subida.
Postcondición	Se permite elegir los parámetros de trinarización.
Excepciones	Salta un error si no se suben dos archivos con el mismo nombre y con extensión .bil y .bil.hdr.
Importancia	Alta

Tabla B.6: CU-6 Cargar imagen hiperespectral en trinarizar.

CU-7	**Elegir nombre de los ficheros en trinarizar**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-2.1
Descripción	El usuario introduce el nombre que quiere que tengan los ficheros resultantes en un cuadro de texto.
Precondición	-
Acciones	1. El usuario escribe el nombre deseado en el cuadro de texto. 2. Al exportar los archivos resultantes tendrán este nombre.
Postcondición	-
Excepciones	-
Importancia	Baja

Tabla B.7: CU-7 Elegir nombre de los ficheros en trinarizar.

CU-8	**Elegir parámetros para la trinarización y procesarlos**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-2.2, RF-2.3, RF-3.1, RF-3.2, RF-3.3, RF-3.4 RF-6.3, RF-6.4
Descripción	Permite al usuario elegir los parámetros para la trinarización y después de pulsar el boton "Procesar imagen", se crea la imagen trinarizada y se muestran los porcentajes.
Precondición	Archivos de la imagen hiperespectral subidos previamente.
Acciones	1. El usuario elige la banda para aplicar la primera binarización. 2. El usuario elige la banda para aplicar la segunda binarización. 3. El usuario elige el umbral para aplicar la segunda binarización. 4. El usuario da al botón "Procesar imagen". 5. Se crea la imagen trinarizada y se muestran los porcentajes.
Postcondición	Se permite descargar la imagen trinarizada, el fichero .csv con los valores de los píxeles y el fichero .csv con los porcentajes. También se permite cargar la imagen trinarizada en la lista.
Excepciones	-
Importancia	Alta

Tabla B.8: CU-8 Elegir parámetros para la trinarización y procesarlos.

CU-9	**Guardar imagen y datos de la trinarización**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-5.2, RF-6.3
Descripción	El usuario pulsa el botón "Descargar imagen trinarizada y csvz se descargan los ficheros.
Precondición	Existencia de una imagen trinarizada y sus porcentajes.
Acciones	1. El usuario pulsa el botón "Descargar imagen trinarizada y csv". 2. Se descargan los ficheros.
Postcondición	Se dispone de los ficheros en la carpeta descargas del usuario.
Excepciones	-
Importancia	Alta

Tabla B.9: CU-9 Guardar imagen y datos de la trinarización.

CU-10	**Cargar imagen trinarizada a la lista**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-5.4, RF-6.3
Descripción	El usuario pulsa el botón Çargar trinarizadaz la imagen queda cargada en la lista.
Precondición	Existencia de una imagen trinarizada
Acciones	1. El usuario pulsa el botón Çargar trinarizada". 2. Se carga la imagen trinarizada en la lista.
Postcondición	Se dispone de la imagen trinarizada en la lista.
Excepciones	-
Importancia	Media

Tabla B.10: CU-10 Cargar imagen trinarizada a la lista.

CU-11	**Trinarización por lotes**
Versión	1.0
Autor	David Merinero Porres
Requisitos asociados	RF-1.4, RF-1.5, RF-2.2, RF-2.3, RF-3.1, RF-3.2, RF-3.3, RF-3.4, RF-5.3, RF-6.3, RF-6.4
Descripción	El usuario sube varios ficheros de imáganes hiperespectrales, elige los parámetros de procesamientos y pulsa al botón "Trinarizar por lotesz se crean las imágenes trinarizadas y se guardan de forma local.
Precondición	-
Acciones	1. El usuario pulsa en el widget de subida de archivos. 2. El usuario elige los ficheros de la imagen hiperespectral. 3. El usuario elige la banda para aplicar la primera binarización. 4. El usuario elige la banda para aplicar la segunda binarización. 5. El usuario elige el umbral para aplicar la segunda binarización. 6. El usuario da al botón "Trinarizar por lotes". 7. Se crean las imágenes trinarizadas y se guardan de forma local.
Postcondición	-
Excepciones	-
Importancia	Media

Tabla B.11: CU-11 Trinarización por lotes.

Apéndice C

Especificación de diseño

C.1. Introducción

En este apartado se explicará qué datos se usan en la aplicación, su diseño arquitectónico y diferentes diagramas sobre el funcionamiento de la aplicación.

C.2. Diseño de datos

En esta aplicación no se han usado bases de datos, los datos con los que se trabaja los carga el usuario cada vez que la ejecuta. Lo que sí se ha utilizado son los llamados estados de sesión (`st.session_state`). Estos estados de sesión permiten almacenar datos mientras se está ejecutando la aplicación, lo cuál es necesario para que funcione correctamente al cambiar de pestaña. Estos son estados de sesión que utiliza la aplicación:

- **lista_capas**: Lista que almacena las diferentes capas que se podrán mostrar en la pestaña *Visualizar*.
- **trinarizada**: Variable que almacena la imagen trinarizada que se ha creado en la pestaña *Trinarizar*.
- **trinarizadas_cargadas**: Diccionario que almacena las trinarizadas que se han cargado para poderlas elegir como capa en la pestaña *Visualizar*.

C.3. Diseño procedimental

Se van a mostrar algunos diagramas que representan acciones que se siguen en la aplicación. El primer caso a tratar el cuando se añade una capa al visualizador.

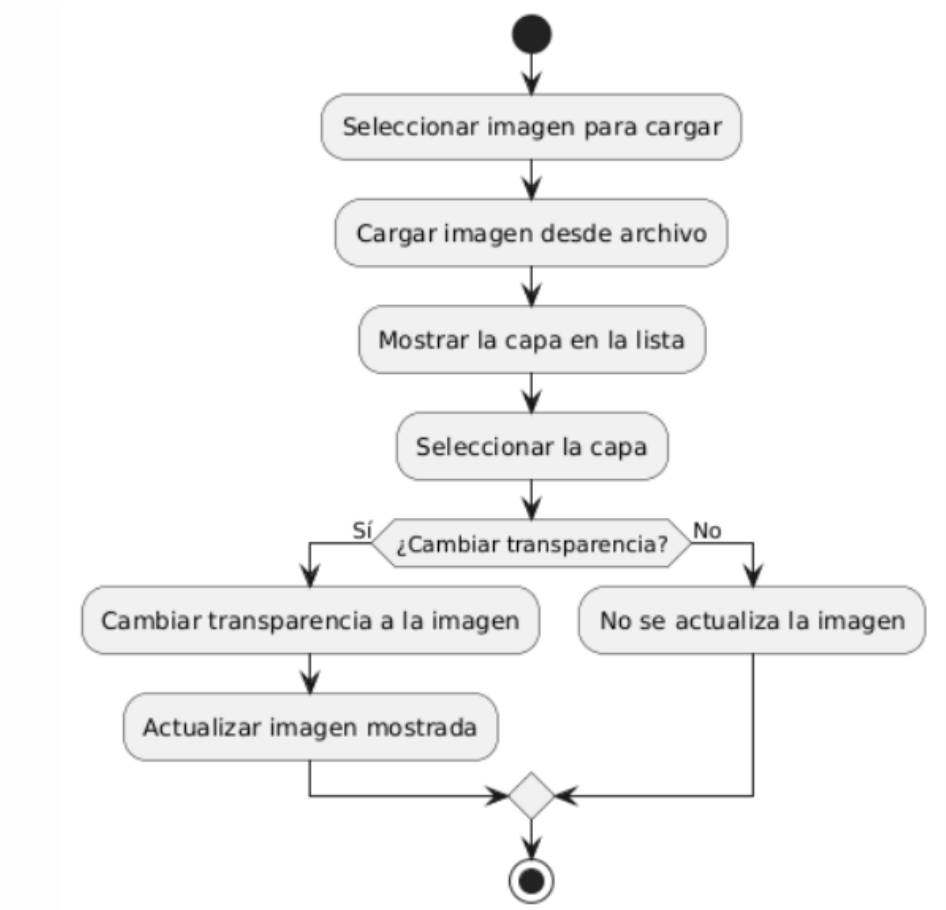

Figura C.1: Diagrama de añadir capa al Visualizador.

Otro caso que se va representar es el proceso de subir una imagen hiperespectral a la pestaña trinarizar.

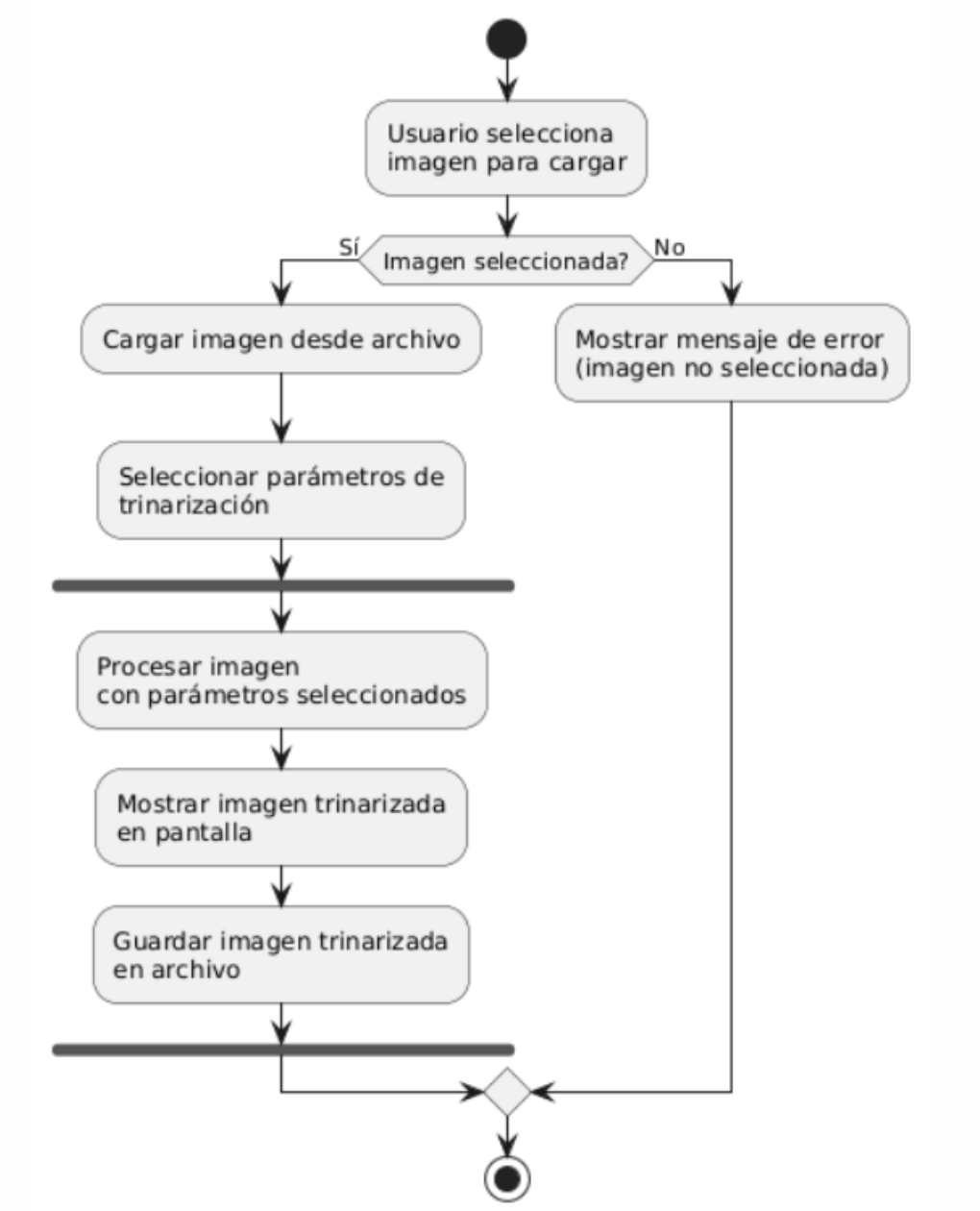

Figura C.2: Diagrama de trinarizar una imagen.

Y así se podrían hacer diagramas de todos los casos de uso y acciones que se llevan a cabo en el programa.

C.4. Diseño arquitectónico

En este apartado se van a mostrar las estructuras que usa la aplicación web desarrollada en este proyecto.

Modelo-Vista-Controlador

El proyecto software utiliza el patrón de diseño Modelo-Vista-Controlador [2], aunque no de manera perfecta. Pero aún así podemos diferenciar en el código elementos que se desarrollan esas funciones diferentes.

Modelo

En este caso, el Modelo estaría representado por estas funciones ya que modifican los datos, ya sea procesando las imágenes hiperespectrales o almacenándolos en una lista. Las funciones son:

- procesar_hyperespectral()
- procesar_imagen()
- procesar_trinarizada()
- guardar_capa()

Vista

La Vista estaría representada por las secciones de código que se encargan de la interfaz de usuario y la presentación de la información, como:

- La tabla de porcentajes que se crea en la pestaña Trinarizar.
- Las imágenes que se muestran en la pestaña Visualizar o Trinarizar.

Controlador

El Controlador sería la parte que enlaza el Modelo y la Vista, manejando los eventos y las interacciones del usuario. En este caso, el controlador estaría representado por:

- Los campos de texto y sliders que permiten la introducción de los valores que se utilizan para procesar las imágenes.

- Los botones que desencadenan alguna acción, como descargar una imagen.

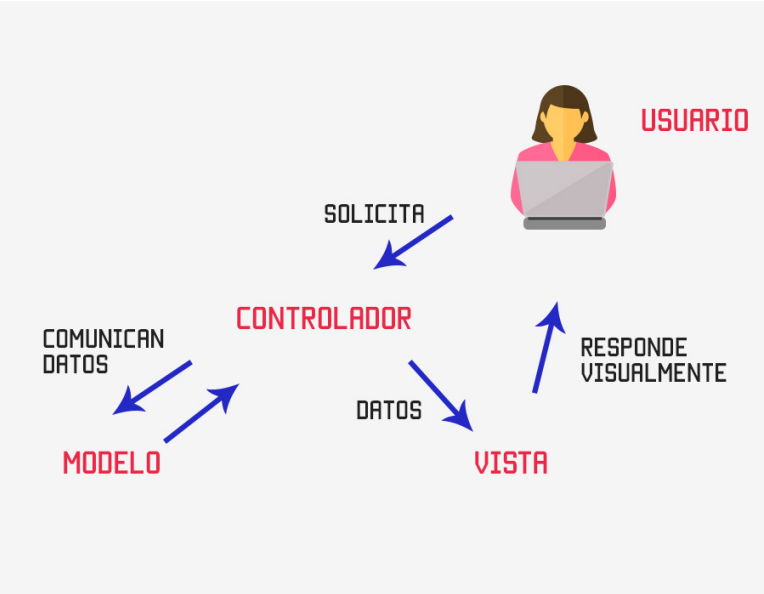

Figura C.3: Esquema del modelo MVC.

Frontend-Backend

El código claramente separa las responsabilidades entre el frontend y el backend. El frontend se enfoca en la interacción con el usuario, mientras que el backend se encarga de la lógica de procesamiento de las imágenes hiperespectrales.

Frontend

La aplicación Streamlit actúa como el frontend, proporcionando una interfaz interactiva para que los usuarios carguen y procesen imágenes. El frontend se encarga de la interacción con el usuario, la presentación de los datos y la gestión del estado de la aplicación.

Backend

Las funciones de procesamiento de imágenes, como procesar_hyperespectral(), procesar_imagen() y procesar_trinarizada(), representan el backend. El backend se encarga de la lógica de procesamiento de imágenes, utilizando

bibliotecas como OpenCV y Spectral para realizar operaciones como la extracción de bandas espectrales, la normalización o las sucesivas binarizaciones.

Esta separación entre frontend y backend permite una mejor organización del código, facilitando la modularidad, el mantenimiento y el posterior desarrollo de nuevas funcionalidades.

C.5. Guía de estilo

En este apartado se van a detallar los elementos que componen la guía de estilo de la aplicación.

Colores

Estos son los diferentes colores que se han escogido para el estlio de la aplicación. Como se puede ver, se usa un verde de color primario, un fondo de color gris oscuro, un gris claro para el segundo color del fondo y el texto va en color blanco.Estos datos van almacenados en el fichero config.toml, que usa Streamlit para cargar las diferentes configuraciones.

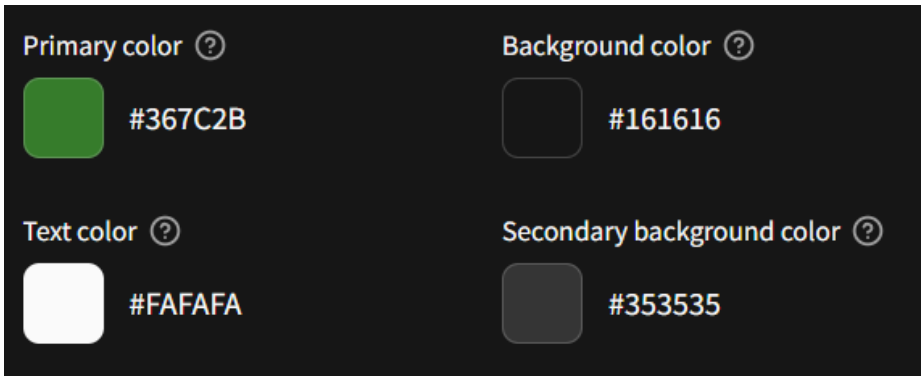

Figura C.4: Colores utilizados en la aplicación.

Tipografía

En este proyecto se ha decidido usar una fuente de tipo Sans Serif, ya que se usa en muchos programas y aplicaciones y es una fuente bastante legible.

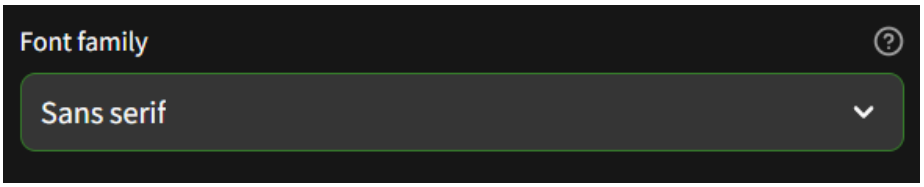

Figura C.5: Tipo de letra escogido para la aplicación.

Iconografía

Se han utilizado emojis para identificar de forma visual la acción que efectúa cada botón. En algunos casos, el emoji va acompañado de texto para explicar en más detalle la acción del botón.

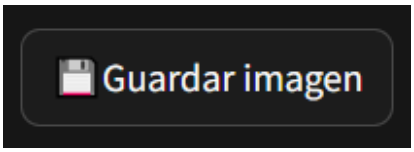

Figura C.6: Ejemplo de botón con texto más un emoji.

Nombre

Para la elección del nombre se pensó utilizar palabras como Vitis (viñedo en latín), Trinarizar, Visualizar, Scan (escanear en inglés), Lab (abreviatura de laboratorio) o Drop (gota en inglés).

Los candidatos propuestos fueron:

- TriVitis
- VitiScan
- VitisLab
- VitisDropScan

Finalmente, se eligió VitiScan como nombre de la aplicación. Los motivos de esta decisión fue porque sonaba bien, era corto, con la S se fusionaban las palabras y sirve como descripción de la funcionalidad del proyecto.

Logotipo

Para el logotipo se ha elegido un cubo con una hoja de viña, respresentando así dos conceptos claves en este proyecto, las imágenes hiperespectrales o hipercúbicas y las hojas de vid con las que se trabaja.

Figura C.7: Logotipo escogido para la aplicación.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En esta sección se explicará la estructura de directorios y archivos que componen el proyecto. También se enseñará a crear el entorno de desarrollo que necesita la aplicación en el manual del programador. Además, se detallará paso a paso cómo compilar, instalar y ejecutar la aplicación en diferentes sistemas operativos y utilizando diferentes métodos.

D.2. Estructura de directorios

El código fuente del proyecto se encuentra dentro de la carpeta `/src`.

src/

En la carpeta src se encuentran una serie de ficheros:

- `bandas.xml`: Este archivo .xml contiene los valores predeterminados de bandas que toma el programa.
- `estilos.css`: Este archivo .css contiene una serie de estilos utilizados en la aplicación.
- `vitiscan.py`: En este archivo se encuentra el código fuente del proyecto.

- `requirements.txt`: Este archivo contiene las dependencias necesarias a instalar para poder compilar y ejecutar la aplicación.
- `Dockerfile`: Este archivo es necesario para poder desplegar la aplicación en un contenedor Docker.

src/.streamlit

Esta carpeta contiene el fichero de configuración `config.toml`

D.3. Manual del programador

En esta sección se explica cómo crear el entorno de desarrollo que necesita esta aplicación.

Instalación de Python

GNU/Linux (Ubuntu)

Para la instalación del entorno de desarrollo en GNU/Linux se ha decidido utilizar la distribución de Ubuntu, ya que es la distribución de Linux más popular. [3]

Lo primero que debemos hacer es instalar Python 3.11, para ello introduciremos el siguiente comando en la consola:

```
sudo apt install python3.11
```

Para verificar la instalación se puede usar este comando:

```
python3 --version
```

Nos debería mostrar la versión de Python que hay instalada.

Windows

En el caso de Windows, debemos ir a la página oficial de Python y descargar el ejecutable de la versión 3.11.5 (enlace). Para verificar la instalación se puede usar este comando:

```
python --version
```

Nos debería mostrar la versión de Python que hay instalada.

Creación y activación del entorno virtual

Se recomienda utilizar un entorno virtual a la hora de crear un entorno de desarrollo en Python, ya que nos permite instalar las dependencias de forma independiente para cada proyecto.

GNU/Linux (Ubuntu)

Para crear el entorno virtual en Linux introduciremos el siguiente comando en la consola:

```
python3 -m venv (nombre del entorno virtual)
```

Posteriormente pasaremos a activar el entorno virtual introduciendo el siguiente comando en la consola:

```
source (nombre del entorno virtual)/bin/activate
```

Y con esto ya tendríamos todo listo para instalar las dependencias del proyecto.

Windows

Para crear el entorno virtual en Windows introduciremos el siguiente comando en la consola:

```
python -m venv (nombre del entorno virtual)
```

Posteriormente pasaremos a activar el entorno virtual introduciendo el siguiente comando en la consola:

```
(nombre del entorno virtual)/Scripts/activate
```

Y con esto ya tendríamos todo listo para instalar las dependencias del proyecto.

Instalación de dependencias

Esta sección es igual para los dos sistemas operativos por lo que procedemos a unificarlo. Para instalar las dependencias del proyecto hay que utilizar el siguiente comando en la consola:

```
pip install -r requirements.txt
```

Una vez hecho todo esto, ya tendríamos el entorno de desarrollo en funcionamiento. Ya se podría tanto modificar la aplicación y como ejecutar el proyecto en modo local, lo cuál se detalla en el siguiente apartado.

D.4. Compilación, instalación y ejecución del proyecto

El proyecto se puede ejecutar de varias maneras, que se procederán a explicar a continuación. El código fuente se puede descargar del repositorio de GitHub del proyecto.

De todas formas se recomienda usar siguiente el comando, pero únicamente si se tiene el cliente de git instalado en el sistema:

```
git clone https://github.com/dmp1002/TFG.git
```

Este comando se deberá ejecutar en la ruta del entorno virtual creado previamente.

Ejecución en local

Suponiendo que se han realizado todos los pasos previamente explicados en el manual del programador, la ejecución en local no tiene ninguna dificultad.

Se debe ir a la ruta /src/ del proyecto y una vez allí ejecutar este comando en la consola:

```
streamlit run vitiscan.py
```

Ejecución con Docker

En este apartado se explica como desplegar la aplicación y cómo ejecutarla utilizando el software de creación de contenedores Docker.

Instalación de Docker

La instalación de Docker explicará únicamente para el sistema operativo Windows.

Lo primero de todo es instalar el Windows Subsystem for Linux, una funcionalidad de Windows que permite ejecutar distribuciones de Linux sin tener que usar una máquina virtual. Para instalarlo sólo hay que poner este comando en la consola Powershell.

```
powershell wsl --install
```

Una vez instalado te pedirá reiniciar. Al volver a encender el ordenador se abrirá una consola que te pedirá crear un usuario y contraseña para la version de Ubuntu que se ha instalado.

Antes de instalar Docker, debemos asegurarnos que en las características de Windows estén activadas dos opciones:

1. El subsistema de Windows para Linux.
2. La plataforma de máquina virtual.

Una vez cumplidos estos requisitos, se pasa a descargar el instalador de Docker desde el siguiente enlace:

https://docs.docker.com/desktop/install/windows-install/

Ya sólo falta ejecutarlo y una vez finalice la instalación reinciar el sistema, entonces ya estará listo para usarse.

Construcción del contenedor Docker

Para construir un contenedor Docker sólo necesitamos ir a la ruta donde tenemos instalado el proyecto y allí ejecutar este comando en la consola:

```
docker build -t nombrecontenedor .
```

Entonces se nos habrá creado el contenedor Docker donde podremos ejecutar el proyecto. En nuestro caso lo llamaremos vitiscan para evitar confusiones.

Ejecución del contenedor Docker

Para ejecutar este contenedor se puede hacer por consola o directamente desde la interfaz del programa. Se recomienda hacerlo por comando, ya que es más sencillo. Sólo hay que ejecutar el siguiente comando en la misma ruta donde se haya creado el contenedor:

```
docker run -p 8501:8501 nombrecontenedor
```

Nosotros pondremos vitiscan para evitar confusiones. Finalmente, se deberá abrir un navegador y acceder al siguiente enlace: http://localhost:8501/

En este proyecto se usa el puerto por defecto 8501, ya que así se ha configurado previamente en el fichero Dockerfile del proyecto.

Exportación del contenedor Docker

Con este comando se puede exportar el contenedor creado a una imagen Docker comprimida con extensión.tar con el siguiente comando:

```
docker save -o nombrecontenedor.tar nombrecontenedor
```

Así exportamos el contenedor actual a una imagen Docker comprimida .tar que se podrá cargar posteriormente en otro equipo y así poder ser ejecutada. Nosotros llamaremos vitiscan tan a la imagen con al contenedor para evitar confusiones.

D.5. Pruebas del sistema

Aunque no se han llevado a cabo pruebas automatizadas para verificar el programa, se han realizado diferentes pruebas para asegurar que todas las funcionalidades implementadas se ejecutan correctamente.

Apéndice E

Documentación de usuario

E.1. Introducción

En esta sección se explicarán los requisitos necesarios para ejecutar el proyecto, los detalles de cómo instalarlo, y también explicará el funcionamiento del programa.

E.2. Requisitos de usuarios

Al tratarse de una aplicación web, no se requieren de requisitos muy complejos. A continuación, se adjunta un listado con los requisitos que se deben cumplir:

- Un **ordenador** que pueda ejecutar un navegador moderno.
- Un **navegador moderno**, a ser posible con la última versión disponible instalada, ya que debe soportar elementos como HTML5, CSS3, JavaScript, WebSockets o tener las cookies habilitadas. Todos los navegadores más conocidos cumplen estos requisitos, ya sea Chrome, Firefox, Safari, Edge u Opera.
- Sería recomendable el **acceso a internet** por si se necesitase instalar algún programa o descargar el proyecto del repositorio.

E.3. Instalación

Ejecución en local

Si se quisiera ejecutar de forma local, sería necesario seguir todos los pasos que se han explicado en el siguiente apartado: Compilación, instalación y ejecución del proyecto de la documentación técnica de programación.

Ejecución con Docker

Suponiendo que tenemos Docker ya instalado en el sistema, y que disponemos una imagen Docker comprimida con extensión .tar con el proyecto en su interior. Para ejecutar el proyecto se deben ejecutar los siguientes comandos en la ruta donde se encuentre la imagen Docker:

```
docker load -i nombre.tar
```

En nuestro caso la imagen se llamará vitiscan. Con esto se carga la imagen en Docker, ahora sólo nos falta introducir el siguiente comando en la consola para que se ejecute la aplicación:

```
docker run -p 8501:8501 nombrecontenedor
```

En nuestro caso el contenedor se llamará vitiscan. La imagen comprimida con extensión .tar se llamará igual que el nombre del contenedor para que no hay confusiones con los nombres.

Finalmente, se deberá abrir un navegador y acceder al siguiente enlace: http://localhost:8501/

E.4. Manual del usuario

Se va a proceder a explicar el funcionamiento de la aplicación, para ello se irá pestaña por pestaña explicando todas las funcionalidades de las que dispone. Las explicaciones se acompañarán de capturas para facilitar su comprensión.

Imágenes hiperespectrales

Antes de nada se va a explicar los datos que se adjuntan para poder utilizar el programa.

El dataset utilizado se compone de imágenes hiperespectrales de las hojas de viñedo en diferentes formatos y con diferentes tipos de aplicaciones de gotas con antifúngicos. Las que se han utilizado en este proyecto principalmente son unas imágenes hiperespectrales de unas hojas de viñedo de tamaño medio y pequeño. Respecto a tipo de aplicación de gotas se han utilizado aquellas con unas dispersión de gotas utilizando una plantilla, es decir, en todas las hojas de vid se ha utilizado la misma plantilla y las gotas están en posiciones parecidas.

No se adjunta el dataset completo porque ocupa más de 50 GB, por lo que se han seleccionado unas pocas de cada tipo. La hojas también puede ser wet o dry, es decir, con la aplicación de gotas recién echada o cuando se secó y se quedó el compuesto adherido a la superficie. En este proyecto se han utilizado las de tipo wet, ya es donde se pueden apreciar las gotas. Cada imagen hiperespectral se compone de un fichero .bil y de otro fichero bil.hdr, que comparten el mismo nombre. Adicionalmente, hay una foto de la imagen a color en formato .tiff.

Sabiendo esto ya se puede pasar explicar el funcionamiento del programa.

Pestaña Visualizar

Este es el aspecto de la pestaña visualizar nada más ejecutar el programa:

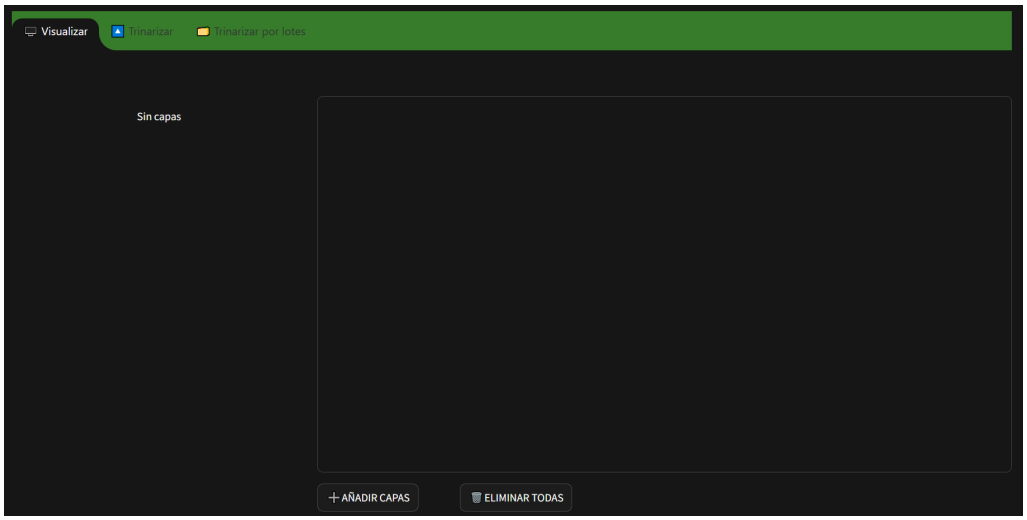

Figura E.1: Pestaña visualizar nada más arrancar la aplicación.

Tiene un botón para cargar capas y otro para eliminar todas.

Si se pulsa el botón de añadir capas se abrirá un ventana emergente y te permitirá subir capas en tres formatos diferentes: imágenes hiperespectrales, imágenes estándar y imágenes trinarizadas. Las trinarizadas no están disponibles de momento, porque no se ha creado ninguna todavía.

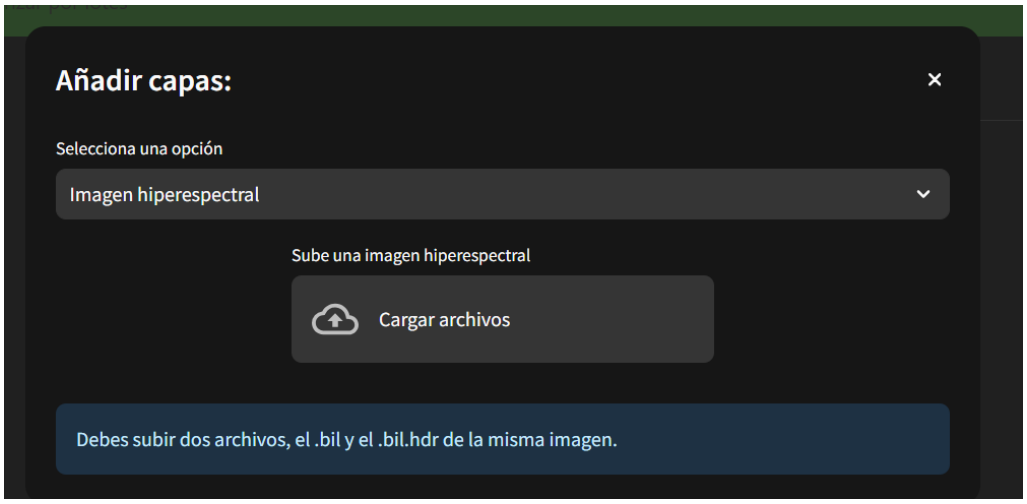

Figura E.2: Ventana de añadir capas.

Se va a proceder a subir una capa de la imagen hiperespectral, eligiendo la banda 40 en este caso.

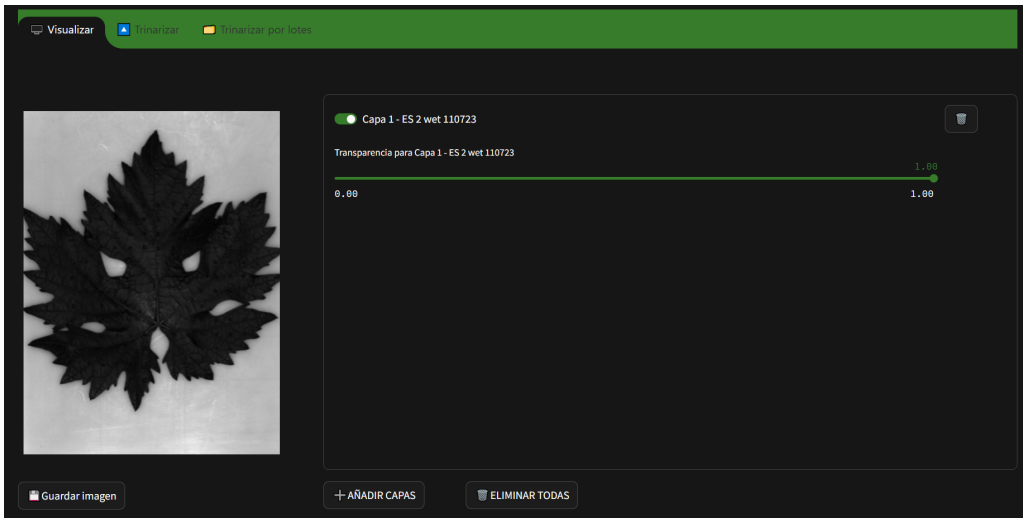

Figura E.3: Pestaña visualizar con una imagen seleccionada

Una vez subida se pueden hacer más acciones, como seleccionar la capa, eliminarla o cambiarle la transparencia. Sin embargo, la transparencia no está habilitada para la primera capa, ya que se intenta imitar el funcionamiento de Photoshop.

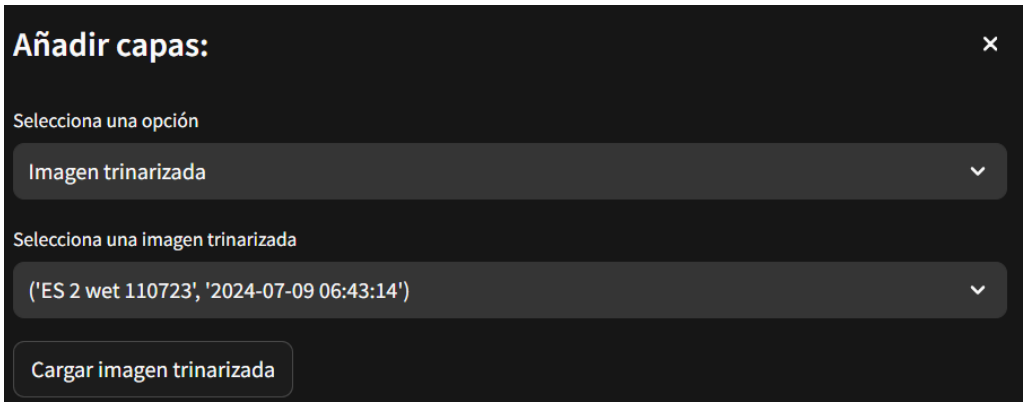

Figura E.4: Ventana de añadir capas con la imagen trinarizada cargada.

Posteriormente, tras crear una imagen trinarizada en la pestaña Trinarizar, se procede a cargar la imagen trinarizada como otra capa. Entonces la seleccionamos y le cambiamos la transparencia al gusto. Si ponemos el cursor encima de la imagen se podrá hacer zoom sobre ella.

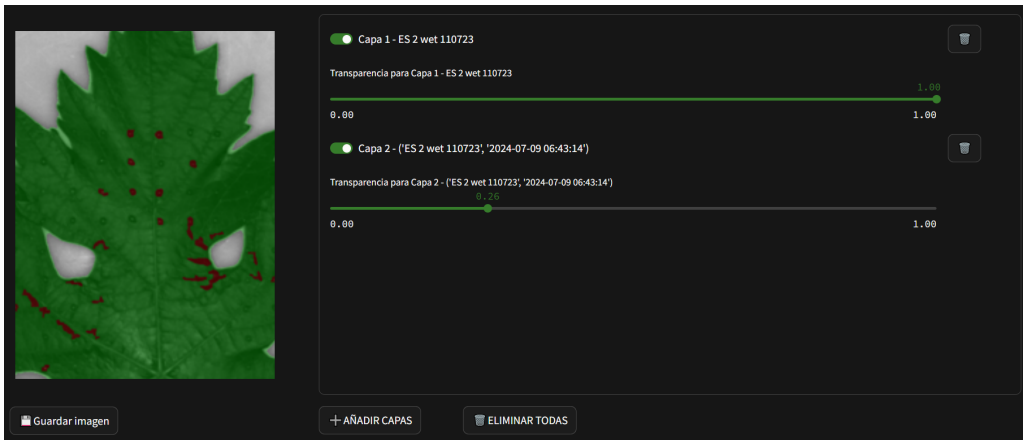

Figura E.5: Pestaña visualizar con el zoom y la imagen resultante

Finalmente guardamos el resultado de la imagen editada por capas pulsando el boton guardar imagen.

Pestaña Trinarizar

En esta pestaña se ve primero un cuadro de texto para introducir el nombre de los ficheros resultantes, un widget para subir una imagen trinarizada y la lista de imágenes trinarizadas que se han cargado en la lista.

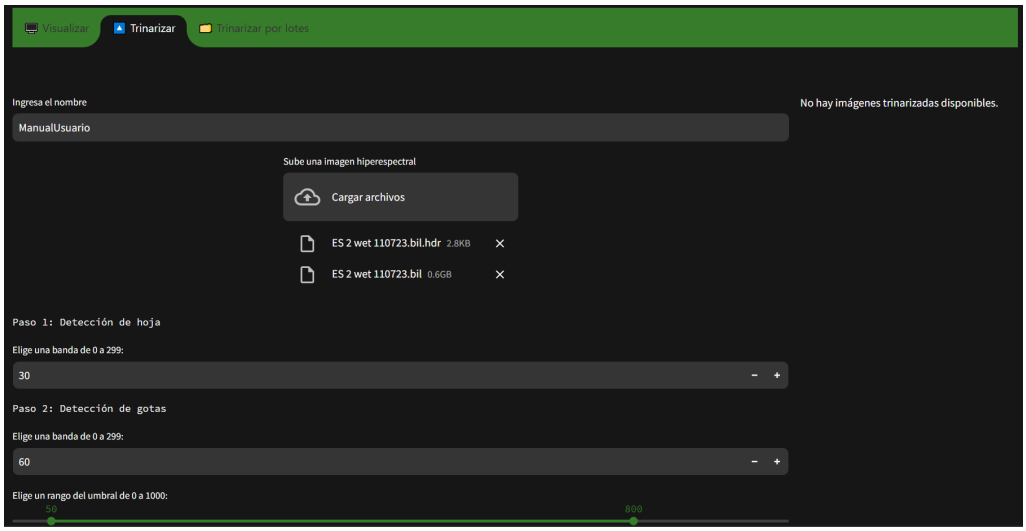

Figura E.6: Pestaña trinarizar después de haber subido los ficheros de una imagen hiperespectral.

Una vez subida la imagen hiperespectral se pide introducir los parámetros para realizar la trinarización. Se piden dos bandas, una para cada binarización y el rango de valores que queremos que se usen para la segunda binarización. Los valores de las bandas se cargan de forma predeterminada desde un fichero XML.

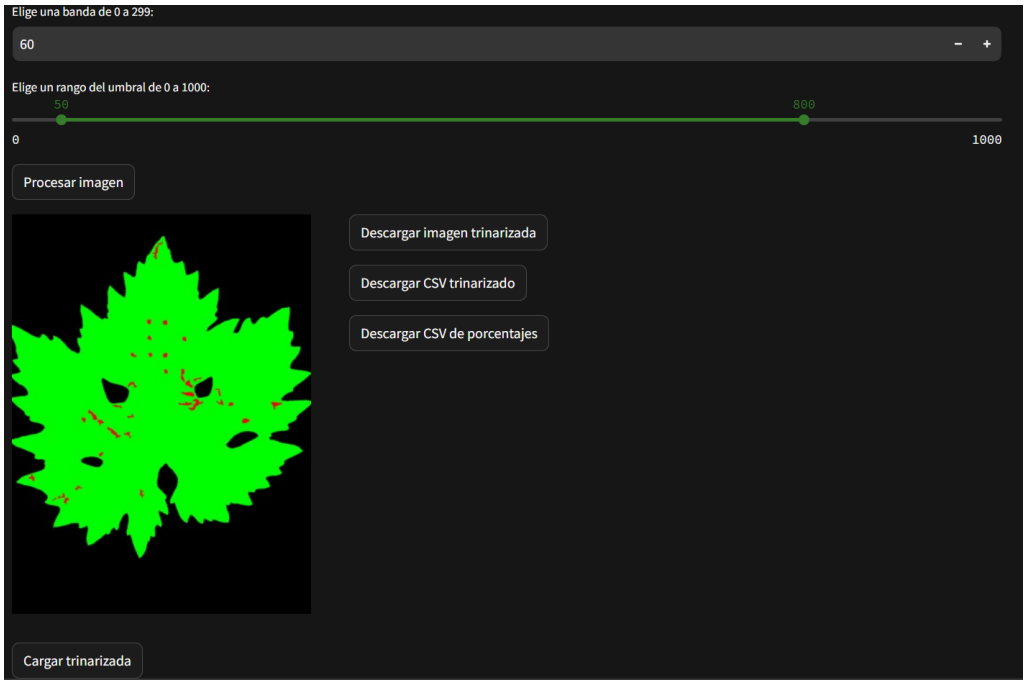

Figura E.7: Resultado de aplicar la trinarización.

Si pulsamos el botón procesar imagen se generará la imagen trinarizada y los porcentajes de hoja y gota que se han detectado. Finalmente se puede decidir si queremos descargar la imagen trinarizada, el csv con los valores de los pixeles u otro csv con los porcentajes. También está la opción de cargar la imagen trinarizada en la lista, y poder así utilizarla en la pestaña Visualizar directamente, sin tener que subirla de forma manual.

Pestaña Trinarizar Por Lotes

La principal diferencia de esta pestaña respecto a la anterior es la posibilidad de procesar múltiples imágenes hiperespectrales a la vez.

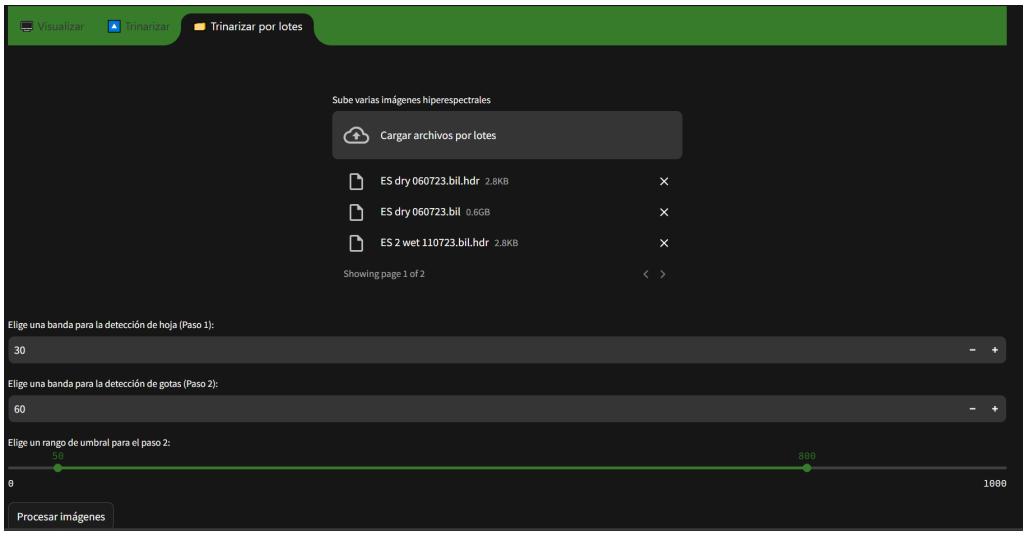

Figura E.8: Pestaña trinarizar por lotes después de cargar varias imágenes hiperespectrales.

Cuando se cargan las imágenes se piden los parámetros como en la pestaña anterior y le damos al botón procesar imagen nos dará la posibilidad de descargar las imagenes trinarizadas que se han creado dentro de un zip.

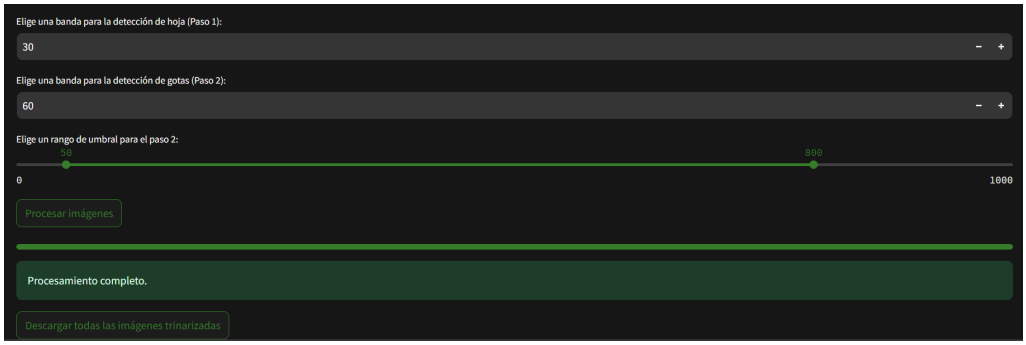

Figura E.9: Pestaña trinarizar por lotes una vez procesadas las imágenes hiperespectrales y generado las imagenes trinarizadas.

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Como estudiante, en este anexo presento mi reflexión personal sobre los aspectos de sostenibilidad que he integrado en mi Trabajo de Fin de Grado. Toda esta reflexión se ha desarrollado siguiendo las instrucciones dadas en el documento sobre la introducción de la sostenibilidad de la CRUE. [1]

Competencias de Sostenibilidad Adquiridas

A lo largo de la realización de mi proyecto, he logrado desarrollar diversas competencias relacionadas con la sostenibilidad:

- **Pensamiento sistémico:** Para la realización de este proyecto se ha tenido una perspectiva holística, analizando los impactos sociales, ambientales y económicos de las soluciones propuestas en mi trabajo. Gracias a esto he sido capaz de reconocer la importancia de la sostenibilidad y del pensamiento sistémico, ya que puede ayudar a que todos nos ayudemos en nuestros proyectos.

- **Anticipación y visión de futuro:** Gracias a pensar a largo plazo he diseñado la aplicación de forma que en un futuro se pueda mejorar y seguir desarrollando. Esto es algo muy importante para ahorrar recursos y el tiempo de otras personas

- **Responsabilidad ética y profesional:** A lo largo de la realización del trabajo se ha ido pensando de forma responsable cómo actuar y qué decisiones tomar a la hora de desarrollar la aplicación.

- **Colaboración interdisciplinar:** En este proyecto me he tenido que sumergir en una temática muy diferente a la que me dedico normalmente, ya que he tenido que aprender sobre temas de agricultura, fertilizantes y el funcionamiento de las imágenes hiperespectrales.

- **Comunicación y sensibilización:** En este proyecto se ha tratado el tema de reducir el uso de fertizantes y la importancia de cuidar el medio ambiente de esta formma.

F.2. Aplicación de la Sostenibilidad al Trabajo de Fin de Grado

En mi Trabajo de Fin de Grado, he logrado integrar los principios de sostenibilidad de la siguiente manera:

- **Análisis del ciclo de vida:** A la hora del desarrollo del proyecto se ha pensado de qué forma se podrá utilizar en un futuro la aplicación y si en algú momento se quedará obsoleta y habrá que crear otra.

- **Diseño circular:** Se han utilizado elementos modulares y que se pueden volver a utiliar como las difrentes librerías de Python que se han añadido al proyecto y que son de uso libre.

- **Inclusión social:** See ha pensado en que la aplicación sea accesible para todo el mundo, por eso se ha elegido una licencia MIT, para que todo el mundo pueda reutilzar el proyecto sin ningún problema.

- **Mitigación y adaptación al cambio climático:** Al estar estudiando la forma de reducir el uso de fertilizantes se está contribuyendo a mitigar el cambio climático. Toda pequeña acción suma.

- **Educación para la sostenibilidad:** Toda persona que utilice este proyecto será capaz de entender la importancia de las imágenes hiperespectrales y como pueden contribuir a la sostenibilidad, como por ejemplo sus usos en la agricultura como en este caso, o en la detección de la frescura de productos vegetales sin destruirlos.

En resumen, a través de este Trabajo de Fin de Grado, he logrado adquirir y aplicar diversas competencias de sostenibilidad que me permitirán desarrollar soluciones más responsables y comprometidas con el bienestar de la sociedad y el medioambiente. El objetivo ha sido integrar los principios de sostenibilidad de manera transversal en todo el proyecto, ya sea en la documentación, en el código o en la realización de los Sprints,

Bibliografía

[1] CRUE. Directrices para la introducción de la sostenibilidad en el curriculum. https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sosteniblidad_Crue2012.pdf.

[2] Mozilla Developers Network. Glosario de MDN Web Docs — MVC. https://developer.mozilla.org/es/docs/Glossary/MVC/. [Internet; accedido el 7-Julio-2024].

[3] TechJury. Linux statistics: All the stats you need in 2024. https://techjury.net/blog/linux-statistics/. [En línea; accedido el 7-Julio-2024].