
This is the **published version** of the bachelor thesis:

García Martín, Marc. *Diseño e implementación de una herramienta web para el análisis de imágenes hiperspectrales*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334447>

under the terms of the  license.

Diseño e implementación de una herramienta web para el análisis de imágenes hiperespectrales

Marc Garcia Martín

Resumen—Este trabajo de Fin de Grado presenta el diseño e implementación de una herramienta web abierta y accesible orientada a la docencia, investigación y experimentación en el análisis de imágenes hiperespectrales. La plataforma mitiga las barreras económicas y de instalación de las soluciones comerciales y propietarias actuales. Siguiendo una arquitectura cliente-servidor, se desarrolla un backend en Python y Flask equipado con un pipeline quimiométrico de preprocesamiento (normalización, SNV y derivadas) y modelos de aprendizaje automático (Random Forest, SVM, k-NN y PLS-DA). El frontend, estructurado como una Single Page Application (SPA) en Flutter Web, interactúa mediante una API REST para cargar cubos de datos, examinar firmas espectrales en tiempo real y renderizar mapas de clasificación. Los resultados con datasets de referencia (Indian Pines, Salinas y Pavia University) validan la precisión del sistema.

Palabras clave—Imágenes hiperespectrales, aplicación web, accesibilidad técnica, aprendizaje automático, clasificación espectral, quimiometría

Abstract—This Bachelor's Thesis presents the design and implementation of an open and accessible web tool tailored for teaching, research, and experimentation in hyperspectral image analysis. The platform bridges the economic and technical installation barriers imposed by current commercial and proprietary software. Adopting a client-server architecture, a Python and Flask backend is developed, featuring a chemometric preprocessing pipeline (normalization, SNV, and derivatives) and machine learning classification algorithms (Random Forest, SVM, k-NN, and PLS-DA). The frontend, built as an interactive Single Page Application (SPA) using Flutter Web, communicates via a REST API to load datasets, inspect spectral signatures in real-time, and render classification maps. Experimental evaluation with reference datasets (Indian Pines, Salinas, and Pavia University) validates the platform's accuracy.

Index Terms—Hyperspectral imaging, web application, technical accessibility, machine learning, spectral classification, chemometrics

1 INTRODUCCIÓN

La combinación de la luz con tonalidades e intensidades distintas da lugar a todo el espectro de colores que vemos en el día a día. El ejemplo más puro de este fenómeno es el arcoíris, donde la luz solar se separa y muestra de forma natural la paleta de colores de la luz visible al reflejarse sobre ciertas superficies y descomponerse en ese espectáculo de color.

Para comprender el origen de la tecnología hiperespectral, es necesario entender cómo los seres humanos percibimos dicha luz en nuestro entorno. El ojo humano es capaz de captar la luz a través de tres canales de color principales: rojo, verde y azul (RGB), como podemos apreciar en la Figura 1.

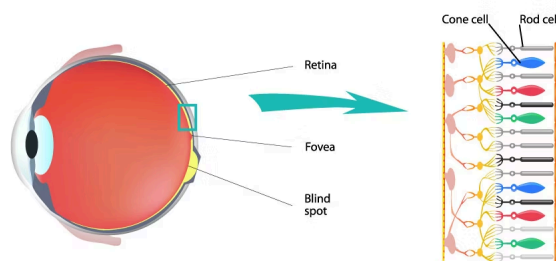


Figura 1: Los conos receptores del ojo humano

Sin embargo, la realidad física va mucho más allá de nuestras limitaciones biológicas. En el siglo XIX, experimentos como los del astrónomo William Herschel, el cual utilizó un prisma de Newton para descomponer la luz en un pequeño arcoíris y medir la temperatura de cada franja de color, demostraron que existía "luz

- E-mail de contacto: marcgarciamartin37@gmail.com
- Mención realizada: Ingeniería del Software
- Trabajo tutorizado por: Coen Antens
- Curso 2025/26

invisible" más allá del color rojo (el infrarrojo), descubriendo que el espectro electromagnético es inmenso y que los humanos solo vemos una pequeña fracción de él [1].

Es precisamente gracias a este descubrimiento y la evolución que provoca que hoy en día podemos hablar de imágenes hiperespectrales. A diferencia de una fotografía convencional RGB (como la de un teléfono móvil, que imita al ojo humano y solo captura tres canales de color), una cámara hiperespectral es capaz de capturar información en cientos de bandas del espectro electromagnético, incluyendo regiones invisibles para nosotros, gracias a un prisma similar al mencionado anteriormente. De este modo, cada píxel de una imagen hiperespectral no es solo un color, sino un vector de datos continuo, llamado firma hiperespectral (Figura 3), que describe las propiedades físicas y químicas exactas del objeto observado. En esta gráfica podemos apreciar los distintos tipos de plásticos que se pueden encontrar en una planta de residuos, prácticamente indistinguibles al ojo humano, clasificados gracias a la tecnología hiperespectral.

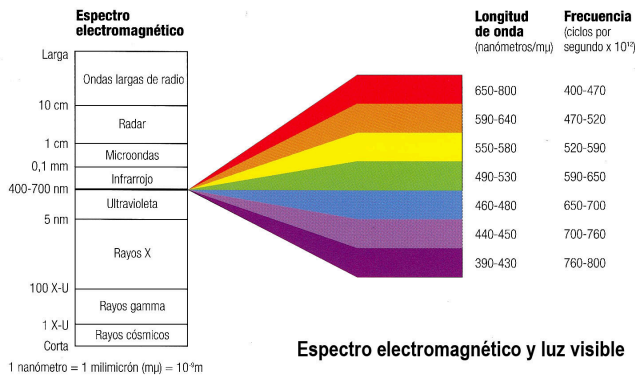


Figura 2: El espectro electromagnético

El ejemplo de uso más cotidiano de esta tecnología puede observarse en productos alimentarios adquiridos en un supermercado, como Mercadona. Por ejemplo, una manzana puede presentar un aspecto visual aparentemente correcto en el momento de la compra, pero deteriorarse rápidamente a los pocos días debido a daños internos producidos durante su transporte o manipulación. Estos defectos, que a simple vista resultan imperceptibles, pueden deberse a pequeños golpes que alteran la estructura interna del fruto. Mediante el uso de cámaras hiperespectrales capaces de analizar la información espectral del producto, sería posible detectar este tipo de daños internos antes de que el alimento llegue al consumidor, mejorando así los procesos de control de calidad en la cadena de distribución.

Otro ejemplo encontrado durante el desarrollo de este proyecto es el caso del control de calidad de muchos productos agroalimentarios, desde pienso para animales hasta productos de consumo propio como fuet. La mayoría de las empresas que comercian con dichos productos no tienen las capacidades ni el entendimiento

tecnológicos para poder utilizar y analizar cámaras y software dedicados a imágenes hiperespectrales, y por culpa de ello deben gastar más dinero del necesario en análisis profesionales sobre las muestras de su producto.

En este contexto, este Trabajo de Fin de Grado propone el desarrollo de una aplicación web abierta, accesible y orientada tanto a la experimentación como a la docencia, que permita visualizar, explorar, editar y analizar imágenes hiperespectrales mediante una herramienta versátil y aplicando técnicas de Machine Learning y Deep Learning.

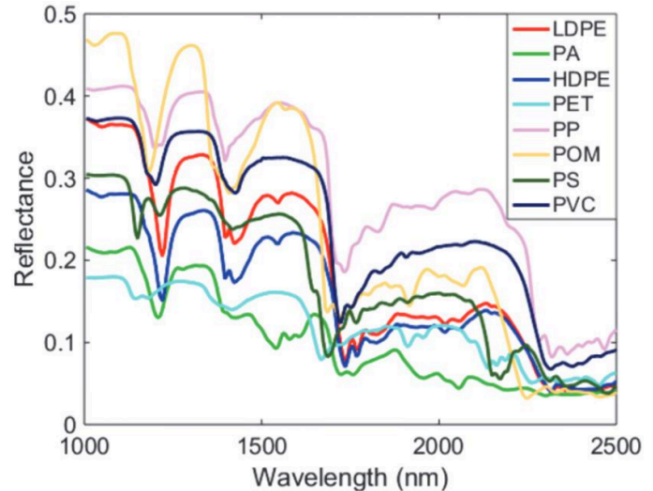


Figura 3: firma hiperespectral de plásticos

2 OBJETIVOS

En este apartado se definen los objetivos del proyecto, estableciendo el propósito principal que se pretende alcanzar con el desarrollo de la herramienta. A partir de este objetivo general se derivan una serie de objetivos específicos que permiten estructurar el trabajo y guiar el desarrollo del sistema durante las distintas fases del proyecto.

2.1 Objetivo General

Desarrollar una aplicación web interactiva que permita la visualización, exploración y clasificación de imágenes hiperespectrales mediante algoritmos de aprendizaje automático, proporcionando una herramienta accesible, modular y extensible para el análisis científico y educativo.

2.2 Objetivos Específicos

- Realizar una revisión del estado del arte en procesamiento y clasificación de imágenes hiperespectrales para poder recoger los requisitos necesarios para el desarrollo.
- Diseñar una arquitectura software escalable basada en backend con Python con API REST, frontend interactivo desarrollado en Flutter Web, y un sistema modular para integración de modelos de Machine Learning.

- Implementar funcionalidades de visualización que permitan la carga de imágenes hiperespectrales en distintos formatos.
- Implementar modelos de clasificación y evaluar su rendimiento mediante métricas objetivas.

3 METODOLOGÍA

La metodología utilizada para el desarrollo del proyecto es de tipo Agile con un modelo Scrum adaptado a un entorno de desarrollo individual. Este enfoque permite organizar el trabajo mediante iteraciones cortas y realizar un seguimiento continuo del progreso del proyecto. Durante el desarrollo se realiza un seguimiento continuo de las tareas, revisando periódicamente el estado del proyecto y ajustando la planificación en función de los avances obtenidos.

3.1 Herramienta de seguimiento

Para realizar el seguimiento del desarrollo del proyecto se utiliza la herramienta Jira [2], que permite gestionar las tareas, organizar los sprints y mantener un control detallado del progreso del proyecto. Las tareas se agrupan dentro de sprints, que tienen una duración aproximada de cuatro a cinco semanas. Cada sprint incluye un conjunto de tareas concretas cuyo objetivo es implementar nuevas funcionalidades o mejorar las existentes. Este enfoque permite realizar entregas progresivas del proyecto y facilita la identificación temprana de posibles problemas durante el desarrollo.

3.2 Herramientas de desarrollo

Para la implementación de la herramienta se utilizan diferentes tecnologías orientadas tanto al procesamiento de datos como al desarrollo de aplicaciones web. El backend de la aplicación se desarrolla en Python, utilizando bibliotecas de aprendizaje automático como scikit-learn [3], que permiten implementar algoritmos de clasificación y análisis de datos.

En cuanto a la interfaz de usuario, se emplea Flutter [4], una herramienta de desarrollo multiplataforma que permite crear aplicaciones web modernas con una interfaz interactiva y accesible desde cualquier navegador.

El uso conjunto de estas tecnologías permite desarrollar una herramienta flexible y modular, donde el procesamiento de las imágenes hiperespectrales se realiza en el backend mientras que la visualización y la interacción con el usuario se gestionan desde la interfaz web.

4 PLANIFICACIÓN

La planificación del proyecto se realiza con el objetivo de organizar las diferentes tareas necesarias para el desarrollo de la herramienta y establecer una distribución temporal adecuada de las mismas. Para ello se identifican las principales fases del proyecto, desde el análisis inicial

y el diseño del sistema hasta la implementación, las pruebas y la documentación final.

Siguiendo la metodología de desarrollo descrita en el apartado anterior, el trabajo se estructura en cuatro sprints de varias semanas de duración. En cada uno de estos sprints se agrupan diferentes tareas relacionadas con una fase concreta del desarrollo, permitiendo avanzar de forma progresiva en la implementación del sistema y realizar un seguimiento continuo del progreso del proyecto.

Para facilitar la organización y el seguimiento de las tareas se utiliza Jira [2], donde se definen las diferentes actividades del proyecto y su duración estimada, divididas en cinco fases: Diseño, Programación, Modelo, Pruebas y Documentación. A partir de esta información se genera un diagrama de Gantt que permite visualizar de forma clara la planificación temporal del proyecto y la relación entre las distintas tareas, el cual se encuentra visible en el anexo A2.

5 ESTADO DEL ARTE

El desarrollo de herramientas para el análisis de imágenes hiperespectrales ha crecido significativamente en los últimos años debido al aumento de aplicaciones en ámbitos científicos e industriales. Actualmente existen soluciones comerciales especializadas, como las desarrolladas por Clyde HSI [5] y PerClass [6].

Sin embargo, la mayoría de estas soluciones están orientadas a entornos industriales y suelen ser herramientas propietarias, con acceso restringido o licencias comerciales. Por este motivo, su utilización en contextos académicos o educativos puede resultar limitada.

A pesar del creciente número de aplicaciones, actualmente existen pocas herramientas accesibles que permitan visualizar y experimentar con datos hiperespectrales de forma sencilla desde una interfaz web. En muchos casos, el análisis requiere el uso de software especializado o conocimientos avanzados en programación científica.

En este contexto, el presente proyecto propone el desarrollo de una herramienta web que permita cargar, visualizar y analizar imágenes hiperespectrales de forma interactiva, integrando algoritmos de aprendizaje automático en un entorno accesible y orientado tanto a la investigación como al aprendizaje.

5.1: Hardware: Cámaras Hiperespectrales



Figura 4: Cámara hiperespectral de Specim

El análisis de imágenes hiperespectrales requiere dispositivos de captura capaces de registrar información espectral en múltiples longitudes de onda. Actualmente existen diversas cámaras hiperespectrales en el mercado que permiten capturar este tipo de datos con alta resolución espectral.

Entre los fabricantes más conocidos se encuentran empresas como Specim [7], que desarrolla cámaras hiperespectrales utilizadas en aplicaciones industriales, agrícolas y medioambientales. Otros fabricantes relevantes incluyen a Headwall Photonics [8], cuyos sistemas se emplean en investigación científica y monitorización ambiental, Cubert GmbH [9], que produce cámaras compactas capaces de capturar información espectral en una única adquisición, e InnoSpec [10], una empresa centrada en el análisis químico de especialización.

5.2: Software para el análisis de las imágenes

El procesamiento y análisis de imágenes hiperespectrales requiere herramientas software capaces de gestionar grandes volúmenes de datos y aplicar técnicas de análisis avanzado.

En el ámbito científico es habitual utilizar bibliotecas de programación como scikit-learn [3], una librería de aprendizaje automático para Python que proporciona algoritmos de clasificación, regresión y clustering ampliamente utilizados en análisis de datos. Estas técnicas permiten clasificar píxeles o regiones de una imagen hiperespectral según sus características espectrales.

Otra herramienta ampliamente utilizada en el procesamiento de imágenes científicas es ImageJ [11], un software de código abierto desarrollado originalmente por el National Institutes of Health (NIH) que permite visualizar, procesar y analizar imágenes multidimensionales mediante diferentes plugins y extensiones.

5.3 Tecnologías web para el desarrollo de la app

El desarrollo de aplicaciones web modernas permite crear herramientas accesibles desde cualquier navegador sin

necesidad de instalar software especializado. Para ello es habitual utilizar frameworks que faciliten la creación de aplicaciones web y la integración con sistemas de procesamiento de datos.

En el ecosistema de Python destacan frameworks como Flask [12] y Django [13], que permiten desarrollar aplicaciones web y APIs REST capaces de comunicarse con módulos de procesamiento de datos y aprendizaje automático.

5.4 Datasets y repositorios de imágenes hiperespectrales

El desarrollo de algoritmos de análisis y clasificación requiere el uso de conjuntos de datos adecuados. En el ámbito hiperespectral existen diversos datasets públicos utilizados en investigación. Algunos de los más conocidos son:

- Indian Pines [14]: dataset ampliamente utilizado en clasificación de cultivos.
- Pavia University [15]: imagen urbana utilizada en tareas de clasificación supervisada.
- Salinas [16]: dataset agrícola con alta resolución espacial.

Estos conjuntos de datos actúan como repositorios de referencia para la evaluación de algoritmos, aunque no constituyen un data warehouse en sentido estricto.

Cabe mencionar la importancia del proyecto europeo Copernicus [17], un ecosistema abierto creado por la ESA [18] que permite el acceso libre a imágenes satelitales de la Tierra, incluyendo la visualización de los distintos canales de dichas imágenes, para el posterior análisis.

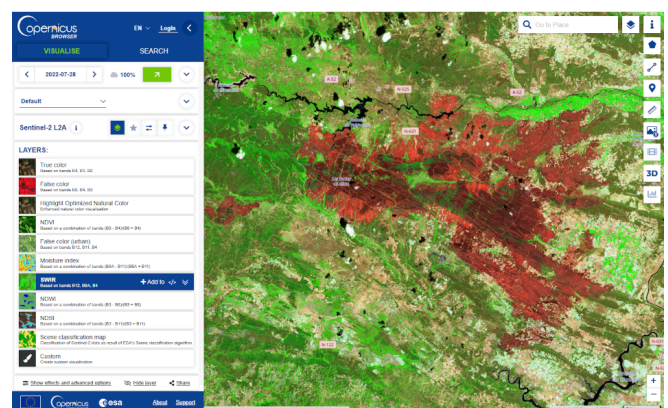


Figura 5: Imagen extraída del proyecto Copernicus

Actualmente no existe un estándar universal de data warehouse para imágenes hiperespectrales, ya que los datos suelen estar distribuidos en repositorios de investigación o instituciones específicas. Sin embargo, iniciativas como la NASA [19] o la propia ESA también proporcionan acceso a grandes volúmenes de datos espectrales a través de sus plataformas de observación

terrestre.

6 DESARROLLO

En este apartado se describe el proceso de desarrollo de la aplicación web propuesta, incluyendo tanto la implementación del backend como del frontend, así como la integración de los distintos componentes del sistema. El objetivo principal es construir una herramienta capaz de procesar, visualizar y analizar imágenes hiperespectrales mediante técnicas de aprendizaje automático, ofreciendo una interfaz accesible desde el navegador.

El desarrollo se lleva a cabo siguiendo una arquitectura cliente-servidor, en la que el procesamiento de datos se realiza en el backend, mientras que la interacción con el usuario se gestiona desde el frontend.

6.1 Arquitectura del sistema

La aplicación se diseña siguiendo una arquitectura basada en servicios, separando claramente las responsabilidades entre frontend y backend, como se puede ver en la Figura 5.

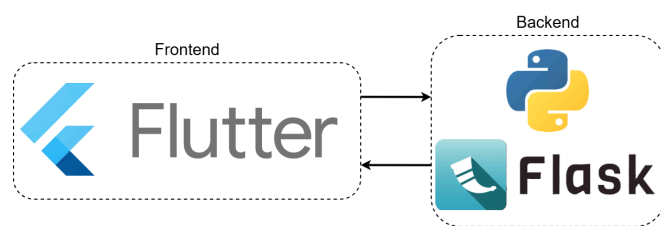


Figura 6: Arquitectura del sistema

Por un lado, el backend, desarrollado en Python, se encarga del procesamiento de las imágenes hiperespectrales, la ejecución de los algoritmos de aprendizaje automático y la gestión de los datos. Por otro lado, el frontend, desarrollado con Flutter, proporciona una interfaz interactiva que permite al usuario cargar imágenes, visualizar resultados y configurar los parámetros del análisis.

La comunicación entre ambos componentes se realiza mediante una API REST, lo que permite desacoplar el sistema y facilitar su escalabilidad y mantenimiento.

6.2 Desarrollo del backend

El backend constituye el núcleo funcional de la aplicación, siendo responsable de la orquestación de datos y la ejecución de modelos. Se ha diseñado como una API REST robusta utilizando Flask, optimizada para el manejo de estructuras de datos hiperespectrales de alta dimensionalidad.

Para garantizar la escalabilidad y facilitar la integración de nuevos datasets, se ha implementado un sistema de carga modular. Este módulo centraliza la configuración de

cada dataset, con los datos correspondientes como rutas de archivos, claves de diccionario o nombres de etiquetas en un diccionario de configuración centralizado, permitiendo añadir nuevas escenas simplemente registrando sus metadatos sin alterar el código base.

Además, para optimizar el rendimiento y evitar latencias innecesarias, se ha implementado un mecanismo de caché en memoria dentro del controlador principal. Antes de realizar cualquier operación de lectura costosa desde el disco, el sistema verifica si el hipercubo solicitado ya reside en la memoria caché del servidor. Si es así, se reutiliza la instancia existente, reduciendo drásticamente los tiempos de respuesta en las peticiones sucesivas.

El backend integra un pipeline especializado en el tratamiento de firmas espectrales, y su flujo de procesamiento se divide en dos fases críticas:

- **Transformación Dimensional:** Dado que las herramientas de scikit-learn operan sobre matrices bidimensionales, se ha implementado una función que permite transformar la imagen hiperespectral (de dimensiones $H \times W \times B$) en una matriz de entrada $N \times B$ (siendo N el número total de píxeles y B el número de bandas), preservando la estructura original para la reconstrucción espacial posterior mediante el almacenamiento de la forma original.
- **Normalización Espectral:** Se han implementado técnicas de preprocesamiento para mitigar los efectos de dispersión de luz y variaciones de iluminación. Las técnicas han sido tres: la normalización de curvas entre mínimos y máximos, la cual escala los valores de reflectancia entre los valores 0 y 1, la Variante Normal Estándar (SNV por sus siglas en inglés), que corrige los efectos de dispersión centrando los datos respecto a su media y normalizándolos por su desviación estándar, y la primera derivada espectral, donde se calculan las tasas de cambio entre bandas contiguas, lo que permite enfatizar características sutiles del espectro que no son evidentes en los datos brutos.

La capa de servicio expuesta mediante Flask permite una interacción fluida con el frontend. Se ha configurado una política de Cross-Origin Resource Sharing que permite al cliente web realizar peticiones asíncronas sin bloqueos de seguridad.

Un componente clave del backend es el cargador dinámico de modelos. En lugar de tener modelos estáticos, el sistema identifica el modelo preentrenado correspondiente basándose en el identificador del dataset, el método de preprocesamiento aplicado y el algoritmo seleccionado.

Finalmente, tras la inferencia, los resultados son reconstruidos a su formato espacial original y serializados a formato JSON para su transmisión al cliente. Este paso

es crítico, ya que convierte tensores de alta dimensionalidad de NumPy en estructuras de listas de listas compatibles con el ecosistema de serialización web, garantizando que el frontend reciba un mapa de clasificación íntegro y listo para ser renderizado.

6.3 Procesamiento y clasificación de imágenes hiperespectrales

Para validar el funcionamiento del sistema, se ha utilizado el dataset Indian Pines [14], del cual podemos observar su *groundtruth* en la Figura 7, ampliamente reconocido en el ámbito de la teledetección.



Figura 7: "Groundtruth" del dataset Indian Pines

Cada píxel de la imagen se representa mediante un vector que contiene información de múltiples bandas espectrales. A partir de estos datos, se entrena un modelo de clasificación basado en cuatro algoritmos: Random Forest, k-Nearest Neighbor, PLS-DA y SVM (RBF), capaces de identificar patrones en las firmas espectrales y asignar una clase a cada píxel. Una vez aplicados los cuatro modelos, se hace un estudio para comprobar qué algoritmo es el más efectivo a la hora de clasificar y, por lo tanto, cuál es el recomendado para usar en cada dataset. Dicho estudio de los resultados se observa en el apartado 7 de este documento.

Una vez entrenado el modelo, este se aplica sobre la totalidad de la imagen, generando un mapa de clasificación que representa la distribución espacial de las distintas clases presentes en la escena.

Este proceso demuestra la capacidad del sistema para analizar imágenes hiperespectrales y extraer información relevante de forma automática.

6.4 Visualización de resultados

La visualización de los resultados se realiza mediante la generación de mapas de clasificación, en los que cada píxel se representa mediante un color asociado a la clase asignada por el modelo.

Antes de tener desarrollada la parte gráfica de la página web, se ha utilizado la librería matplotlib, que permite representar de forma gráfica la distribución espacial de las clases. Esta representación facilita la interpretación de los resultados y permite identificar patrones visuales en la

imagen, como se observa en la Figura 8.

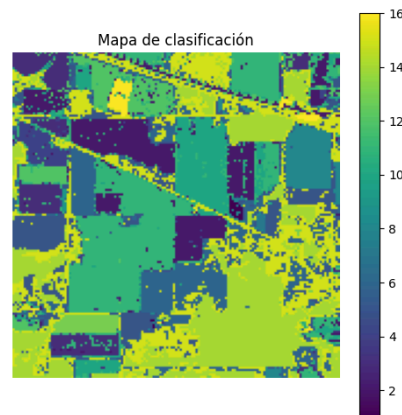


Figura 8: Resultado de clasificación del dataset Indian Pines

Una vez implementado todo el código, se ha podido comprobar que dicha clasificación coincide con lo mostrado en la propia app, observable en el mapa de clasificación de la Figura 9.

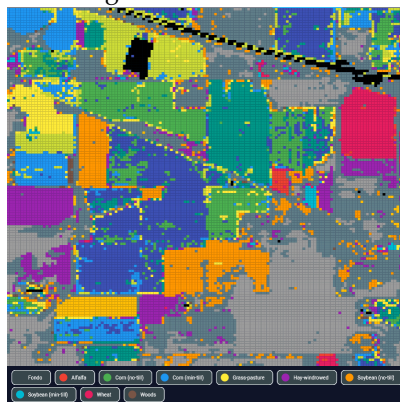


Figura 9: Resultado de clasificación en el modelo gráfico del dataset Indian Pines

6.5 Desarrollo del Frontend

Se ha desarrollado una interfaz web de una sola página (SPA - Single Page Application) utilizando el framework Flutter Web. Esta elección tecnológica asegura un despliegue inmediato en entornos educativos sin fricciones de instalación.

La arquitectura visual del frontend se estructura en tres zonas de control bien delimitadas, como se puede apreciar en la Figura 10, emulando la distribución de las estaciones de trabajo clásicas de teledetección como ImageJ:

- Panel de Control Izquierdo:**
 Permite al usuario examinar las propiedades físicas y dimensiones del dataset activo. Contiene un módulo de selección de Lookup Tables (LUT) para el mapeado cromático de los píxeles (Escala de grises, Térmico, Jet y Frío), selectores excluyentes para preprocesamiento quimiométrico (Normalización de curvas, Standard Normal Variate y Primera Derivada) y reguladores numéricos para Brillo, Contraste y Umbralización (Threshold).

- **Visor de Stack Central:**

Permite al usuario explorar visualmente la imagen espectral. Integra dos modos principales de renderizado, el modo banda única, el cual proyecta la intensidad de reflectancia de un único *Z-slice* espectral, y el modo combinación RGB, que permite realizar una proyección aditiva mapeando tres rebanadas independientes sobre los canales Rojo, Verde y Azul del monitor para componer imágenes de falso color o color verdadero.

- **Panel de Resultados Derecho:**

Muestra de forma dinámica la firma de reflectancia espectral del píxel seleccionado mediante un gráfico vectorial continuo trazado por coordenadas físicas en nanómetros. Asimismo, integra una tabla estadística en tiempo real, calculando de forma instantánea el área seleccionada, el valor medio de reflectancia de la región, la desviación estándar, y los valores mínimos y máximos de la firma.

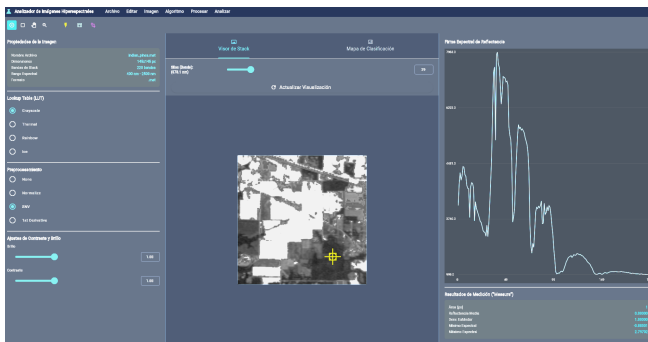


Figura 10: Diseño SPA de la página web

Para satisfacer la necesidad de un control milimétrico durante el análisis científico, todos los controles de la interfaz (deslizadores de contraste, brillo, umbral, bandas y canales RGB) se han programado con un mecanismo de enlace de datos bidireccional, el cual permite tener una sincronización a tiempo real con la imagen.

Cada deslizador opera en paralelo con un campo de texto numérico editable. Esto permite que el operario del laboratorio realice ajustes rápidos mediante gestos táctiles o de ratón, o bien introduzca directamente con el teclado el canal espectral preciso o el valor decimal exacto de un filtro. El sistema valida automáticamente las entradas de texto para evitar desbordamientos de rango matemático y redibuja la imagen inmediatamente tras presionar la tecla de confirmación.

Debido a que cada sensor hiperespectral opera en un rango físico propio de la radiación electromagnética, se ha desarrollado un motor de conversión dinámico en el cliente. Al cargar un dataset, la aplicación identifica el hardware de adquisición para calibrar las etiquetas de visualización del espectro, seleccionando así los valores máximos y mínimos permitidos para el número de bandas y su longitud de onda permitidas.

7 ANÁLISIS DE RESULTADOS

En esta sección se exponen los resultados de la evaluación experimental realizada. El objetivo principal es identificar la configuración óptima que maximiza la capacidad discriminativa del sistema frente a los datasets Indian Pines, Pavia University y Salinas.

Para ello, se ha diseñado un pipeline que somete a cada dataset a cuatro métodos de normalización distintos y entrena cuatro modelos de aprendizaje automático, permitiendo una comparativa exhaustiva del rendimiento obtenido en condiciones controladas.

7.1 Métricas de evaluación

Para cuantificar con rigor la capacidad predictiva de cada modelo para discriminar materiales, se han calculado tres métricas clásicas a partir de la matriz de confusión multiclase generada sobre el conjunto de test independiente:

- **Precisión:** Mide la exactitud de las predicciones positivas del modelo, indicando el porcentaje de píxeles asignados a una clase que realmente pertenecían a ella.

$$Precision = \frac{TP}{TP+FP}$$

- **Sensibilidad:** Mide la capacidad del modelo para localizar todos los píxeles reales de una determinada categoría física.

$$Recall = \frac{TP}{TP+FN}$$

- **F1 Score:** Representa la media armónica entre la Precisión y la Sensibilidad, proporcionando una métrica equilibrada robusta ante clases desbalanceadas.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Donde *TP*, *FP* y *FN* representan los Verdaderos Positivos, Falsos Positivos y Falsos Negativos calculados de manera agregada a nivel de píxel.

7.2 Tablas de resultados

El análisis de los datos revela una jerarquía clara en cuanto a la capacidad predictiva:

- **Random Forest (RF):** Se posiciona como el algoritmo más robusto y preciso, alcanzando un F1-Score máximo del 95.17% en el dataset Salinas, visible en la Figura 12. Su capacidad para manejar la alta dimensionalidad de los datos hiperespectrales y su resistencia al sobreajuste (overfitting) mediante la creación de múltiples árboles de decisión lo convierten en la opción óptima para esta herramienta.
- **Support Vector Machines (SVM):** Presenta un rendimiento muy sólido, situándose muy cerca de RF. El kernel RBF ha demostrado ser altamente efectivo al encontrar hiperplanos de separación en espacios de alta dimensionalidad. Sin embargo, se observa una ligera mayor sensibilidad a la escala de los datos en comparación con RF.

- **k-Nearest Neighbors (k-NN):** Ofrece resultados aceptables pero inconsistentes ante cambios en el preprocesamiento. Su dependencia de la métrica de distancia euclidiana lo hace vulnerable a variaciones de intensidad, lo que explica por qué su rendimiento fluctúa más que el de los métodos basados en árboles o hiperplanos.
- **PLS-DA:** Es el método que presenta el menor rendimiento. Esto era esperable, dado que PLS-DA es un método esencialmente lineal. La complejidad de las firmas espectrales en los datasets seleccionados es intrínsecamente no lineal, lo que limita la capacidad de un modelo de regresión lineal por mínimos cuadrados para capturar todas las variables discriminatorias.

Pese a que la mayoría de clasificadores de imágenes hiperespectrales utiliza variaciones del algoritmo PLS, este estudio demuestra que, en datasets pequeños con muchas variaciones, RF funciona con mucha más precisión que el PLS, siendo el algoritmo predilecto de este proyecto.

Dataset	Preprocesado	Modelo	Accuracy	Precision	Recall	F1-Score
indian_pines	snv	RF	89.07	89.15	89.07	88.93
indian_pines	none	RF	86.83	86.9	86.83	86.54
indian_pines	minmax	RF	86.21	86.17	86.21	85.99
indian_pines	derivative	RF	78.8	80.47	78.8	77.92
indian_pines	snv	k-NN	77.98	78.04	77.98	77.33
indian_pines	minmax	k-NN	77.11	77.09	77.11	76.62
indian_pines	none	k-NN	76.75	76.65	76.75	76.05
indian_pines	none	SVM	72.23	73.42	72.23	70.29
indian_pines	minmax	SVM	71.64	72.21	71.64	69.52
indian_pines	derivative	SVM	70.11	72.4	70.11	68.24
indian_pines	derivative	k-NN	66.15	65.84	66.15	64.84
indian_pines	snv	SVM	66.05	67.94	66.05	62.93
indian_pines	snv	PLS-DA	64.52	64.82	64.52	59.59
indian_pines	none	PLS-DA	62.5	65.23	62.5	56.04
indian_pines	derivative	PLS-DA	60.65	63.49	60.65	55.29
indian_pines	minmax	PLS-DA	60.2	61.09	60.2	54.13

Figura 11: Tabla de resultados del dataset Indian Pines

Dataset	Preprocesado	Modelo	Accuracy	Precision	Recall	F1-Score
salinas	snv	RF	95.17	95.16	95.17	95.12
salinas	none	RF	94.95	94.94	94.95	94.9
salinas	minmax	RF	94.9	94.92	94.9	94.83
salinas	derivative	RF	93.71	93.81	93.71	93.58
salinas	snv	k-NN	92.87	92.8	92.87	92.81
salinas	minmax	k-NN	92.84	92.78	92.84	92.79
salinas	none	k-NN	92.28	92.19	92.28	92.19
salinas	derivative	SVM	91.34	91.58	91.34	90.96
salinas	derivative	k-NN	90.49	90.35	90.49	90.37
salinas	none	SVM	90.72	91.08	90.72	90.24
salinas	minmax	SVM	89.98	90.18	89.98	89.53
salinas	snv	SVM	89.78	90.05	89.78	89.29
salinas	snv	PLS-DA	80.3	80.74	80.3	76.03
salinas	minmax	PLS-DA	76.67	76.9	76.67	72.12
salinas	derivative	PLS-DA	74.23	76.48	74.23	67.49
salinas	none	PLS-DA	72.98	74.89	72.98	65.26

Figura 12: Tabla de resultados del dataset Salinas

Dataset	Preprocesado	Modelo	Accuracy	Precision	Recall	F1-Score
pavia_university	none	SVM	94.16	94.17	94.16	94.11
pavia_university	derivative	SVM	92.99	93.02	92.99	92.95
pavia_university	none	RF	92.85	92.85	92.85	92.75
pavia_university	snv	RF	91.05	91.1	91.05	90.86
pavia_university	none	k-NN	90.19	90.26	90.19	89.99
pavia_university	minmax	RF	88.94	89.09	88.94	88.57
pavia_university	minmax	SVM	88.96	89.07	88.96	88.49
pavia_university	snv	SVM	88.89	89.05	88.89	88.36
pavia_university	derivative	RF	87.95	88.41	87.95	87.21
pavia_university	minmax	k-NN	84.87	84.83	84.87	84.32
pavia_university	snv	k-NN	82.4	82.67	82.4	81.85
pavia_university	derivative	k-NN	81.52	81.37	81.52	80.9
pavia_university	none	PLS-DA	76.08	73.36	76.08	71.57
pavia_university	derivative	PLS-DA	75.28	72.31	75.28	70.51
pavia_university	minmax	PLS-DA	72.85	72.56	72.85	67.05
pavia_university	snv	PLS-DA	71.3	63.92	71.3	65.39

Figura 13: Tabla de resultados del dataset Pavia University

8 CONCLUSIÓN

En este último apartado se finaliza el trabajo analizando los éxitos y las limitaciones del proyecto, además de mostrar la posible evolución del proyecto con líneas de trabajo futuras.

8.1 Síntesis del trabajo realizado

En este proyecto se ha diseñado e implementado con éxito una solución de software interactiva y multiplataforma orientada al análisis espectral científico. La consecución de los objetivos generales y específicos planteados al inicio del proyecto ha validado la viabilidad de utilizar frameworks modernos de desarrollo móvil y web (como Flutter) en combinación con entornos de computación científica basados en Python.

La herramienta permite superar la brecha tecnológica y económica que presentaban las soluciones comerciales de análisis hiperespectral. Al proporcionar acceso directo desde cualquier navegador web moderno, se elimina la barrera de entrada para estudiantes, docentes e investigadores que no disponen de recursos para adquirir licencias propietarias o hardware especializado de procesamiento.

8.2 Limitaciones y objetivos no alcanzados

A pesar del éxito general del sistema, la naturaleza limitada de un Trabajo de Fin de Grado impone ciertas restricciones que deben ser señaladas, como por ejemplo un entrenamiento de modelo en el propio navegador, ya que el modelo de Machine Learning se entrena y ejecuta de forma estricta en el servidor backend (Python).

Habría sido interesante poder entrenar modelos ligeros directamente en el cliente Web para lograr una independencia total del servidor, aunque esta arquitectura habría sobrecargado el navegador del cliente en datasets muy densos.

Como última mención, se decidió implementar un sistema de cuentas en la fase de diseño del programa, pero al haber convertido la página web en una SPA, se ha creído conveniente eliminar dichos requisitos de la

aplicación, dejando únicamente la importación y exportación de sesiones en formato JSON, permitiendo retomar análisis pasados sin necesidad de inicios de sesión, agilizando el proceso.

8.3 Líneas de trabajo futuras

Como continuación y evolución natural del software desarrollado, se proponen como extensiones del proyecto la integración de modelos de aprendizaje profundo (Deep Learning), permitiendo aumentar la precisión utilizando clasificadores basados en Redes Neuronales Convolucionales Tridimensionales.

También se propone validar la herramienta integrando de manera modular conjuntos de datos procedentes de múltiples disciplinas de vanguardia. Esta incorporación permitiría expandir y diversificar la capacidad de clasificación del analizador web a sectores con un alto impacto socioeconómico, como el ámbito médico o de conservación medioambiental, con datasets procedentes de páginas y repositorios como el publicado por la Universidad Tokyo Denki, Awesome Hyperspectral Datasets [20].

USO DE LA IA GENERATIVA

Cabe mencionar que, durante el desarrollo de este proyecto, se ha utilizado la IA Generativa en casos puntuales como soporte. Se ha aplicado mayoritariamente a la estructuración de este documento, en casos puntuales para reformular oraciones y siempre bajo vigilancia y criterio propios. También se ha aplicado a la búsqueda de conocimiento inicial sobre los temas en los que menos información se tenía, ya que, por ejemplo, tanto las imágenes hiperespectrales como todo el proceso de machine learning eran temas nuevos y relativamente desconocidos al inicio de todo este proyecto. Algunos ejemplos son:

- Corrección del formato de citas
- Búsqueda de artículos informativos sobre imágenes hiperespectrales
- Breve introducción y explicación sobre los métodos más eficientes de implementar machine learning y búsqueda de tutoriales adecuados
- Correcciones ortográficas puntuales

AGRADECIMIENTOS

Me gustaría agradecer a mis amigos por su apoyo constante a lo largo de este curso y este proyecto. También agradecer enormemente a Coen Antens un seguimiento tan eficiente y constante para sacar adelante esta idea, sin su guía este trabajo no habría sido lo mismo. Finalmente, quiero darle las gracias a Marta Gutiérrez Vila, quien ha estado brindando la ayuda y el apoyo que he necesitado en todo momento con su infinita paciencia, por los cuales siempre estaré agradecido.

BIBLIOGRAFÍA

[1] CSIC, "Experimento de Herschel: el descubrimiento del infrarrojo." [En Línea]. Disponible: http://sac.csic.es/astrosecundaria/es/proyectos_con_unesco/dia_i

[internacional de la luz 2018/HerschelCas.pdf](#) [Accedido: 22-jun-2026].

[2] Atlassian, "Jira — Issue Tracking and Project Management Tool." [En línea]. Disponible: <https://www.atlassian.com/software/jira>. [Accedido: 23-feb-2026].

[3] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python." [En línea]. Disponible: <https://scikit-learn.org>. [Accedido: 23-feb-2026].

[4] Google, "Flutter — UI Toolkit for Building Applications." [En línea]. Disponible: <https://flutter.dev>. [Accedido: 23-feb-2026].

[5] ClydeHSI, "Hyperspectral Imaging Solutions." [En línea]. Disponible: <https://www.clydehsi.com>. [Accedido: 22-feb-2026].

[6] PerClass BV, "Hyperspectral Image Classification Software." [En línea]. Disponible: <https://www.perclass.com>. [Accedido: 22-feb-2026].

[7] Specim, Spectral Imaging Ltd., "Hyperspectral Cameras and Systems." [En línea]. Disponible: <https://www.specim.com>. [Accedido: 22-feb-2026].

[8] Headwall Photonics Inc., "Hyperspectral Imaging Systems." [En línea]. Disponible: <https://www.headwallphotonics.com>. [Accedido: 22-feb-2026].

[9] Cubert GmbH, "Snapshot Hyperspectral Cameras." [En línea]. Disponible: <https://www.cubert-gmbh.com>. [Accedido: 22-feb-2026].

[10] Innospec Inc., "Innospec." [En línea]. Disponible: <https://innospec.com>. [Accedido: 20-jun-2026].

[11] National Institutes of Health, "ImageJ — Image Processing Program." [En línea]. Disponible: <https://imagej.nih.gov/ij/>. [Accedido: 23-feb-2026].

[12] A. Ronacher, "Flask Web Framework." [En línea]. Disponible: <https://flask.palletsprojects.com>. [Accedido: 23-feb-2026].

[13] Django Software Foundation, "Django Web Framework." [En línea]. Disponible: <https://www.djangoproject.com>. [Accedido: 23-feb-2026].

[14] Purdue University, "Indian Pines Dataset." [En línea]. Disponible: https://www.ehu.es/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes. [Accedido: 29-mar-2026].

[15] University of Pavia, "Pavia University Dataset." [En línea]. Disponible: https://www.ehu.es/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes. [Accedido: 29-mar-2026].

[16] University of California, "Salinas Dataset." [En línea]. Disponible: https://www.ehu.es/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes. [Accedido: 29-mar-2026].

[17] Copernicus Data Space Ecosystem, "Project Copernicus." [En Línea]. Disponible: <https://dataspace.copernicus.eu> [Accedido: 20-jun-2026].

[18] European Space Agency, "Copernicus Programme." [En línea]. Disponible: <https://www.esa.int>. [Accedido: 29-mar-2026].

[19] NASA, "Earth Data." [En línea]. Disponible: <https://earthdata.nasa.gov>. [Accedido: 29-mar-2026].

[20] Tokyo Denki University, "Awesome Hyperspectral Datasets."
 [En Línea]. Disponible:
<https://033labcodes.github.io/awesome-hyperspectral-datasets/>
 [Accedido: 22-may-2026].

APÉNDICE

A1. REQUISITOS

ID	Requisito	Descripción	Prioridad
RF1	Carga de imágenes hiperespectrales	La aplicación permite al usuario cargar imágenes hiperespectrales desde su dispositivo.	Alta
RF2	Validación de formato	El sistema verifica que el formato del archivo cargado es compatible.	Alta
RF3	Lectura de metadatos	El sistema extrae y muestra información básica de la imagen (dimensiones, número de bandas, etc.).	Media
RF4	Visualización inicial de la imagen	El sistema muestra una representación inicial de la imagen cargada.	Alta
RF5	Selección de banda espectral	El usuario puede seleccionar una banda concreta para su visualización.	Alta
RF6	Visualización de múltiples bandas	El sistema permite combinar varias bandas para generar imágenes compuestas.	Media
RF7	Navegación por la imagen	El usuario puede hacer zoom y desplazarse por la imagen.	Alta
RF8	Selección de píxeles	El usuario puede seleccionar un píxel específico de la imagen.	Alta
RF9	Selección de regiones (ROI)	El usuario puede seleccionar una región de interés dentro de la imagen.	Media
RF10	Visualización de firma espectral	El sistema muestra el vector espectral asociado a un píxel o región seleccionada.	Alta
RF11	Exportación de datos espectrales	El usuario puede exportar los datos espectrales seleccionados.	Alta
RF12	Preprocesamiento de datos	El sistema permite aplicar operaciones básicas (normalización, filtrado, etc.).	Media
RF13	Aplicación del modelo	El sistema aplica el modelo de clasificación entrenado a la imagen completa.	Alta
RF14	Visualización de resultados de clasificación	El sistema muestra el resultado de la clasificación como mapa de clases.	Alta
RF15	Interacción con clases	El usuario puede seleccionar una clase y visualizar sus píxeles asociados.	Alta
RF16	Guardado de resultados	El sistema permite guardar los resultados obtenidos.	Media

ID	Requisito	Descripción	Prioridad
RF17	Carga de resultados previos	El usuario puede cargar resultados previamente guardados.	Media
RF18	Comunicación frontend-backend	El sistema permite la comunicación mediante API REST.	Alta
RF19	Gestión de errores	El sistema informa al usuario en caso de errores en carga o procesamiento.	Alta
RF20	Reinicio de análisis	El usuario puede reiniciar el proceso de análisis con una nueva imagen.	Alta
RNF1	Usabilidad	La aplicación debe ser intuitiva y fácil de usar, con un enfoque accesible para usuarios sin experiencia avanzada.	Media
RNF2	Rendimiento	El sistema debe procesar y visualizar imágenes en tiempos razonables, incluso con datos de gran tamaño.	Baja
RNF3	Portabilidad	La aplicación debe ser accesible desde un navegador web sin necesidad de instalación adicional.	Alta
RNF4	Mantenibilidad	El código debe estar estructurado de forma modular para facilitar su mantenimiento y evolución.	Media

A2. DIAGRAMAS

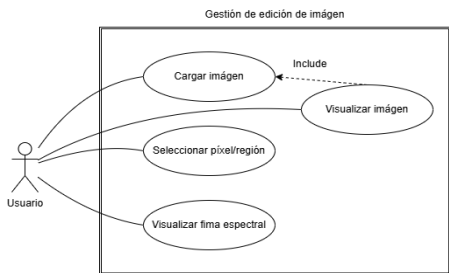


Diagrama 1: Gestión de edición de imagen

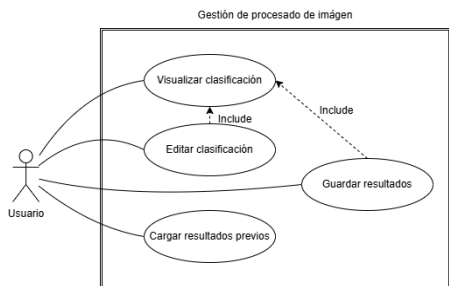


Diagrama 2: Gestión de procesamiento de imagen

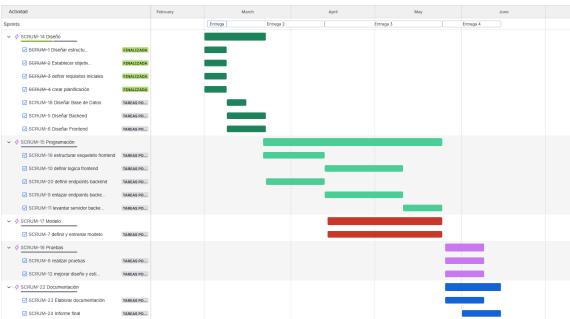


Diagrama 3: Diagrama de Gantt

A3. CAPTURAS DE PANTALLA DE LA APP

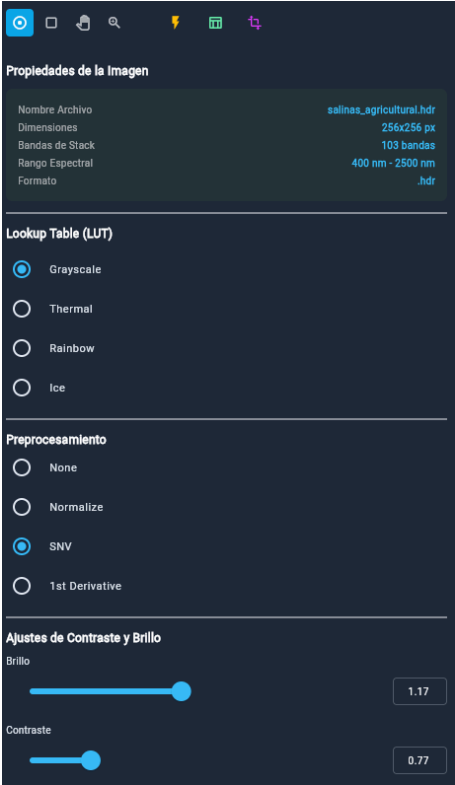


Figura A1: Menú de preselección de imagen

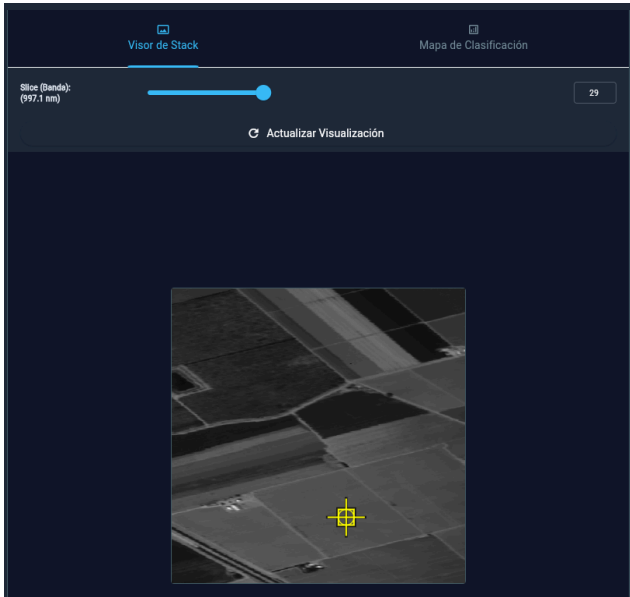


Figura A2: Visor de stack y selector de bandas de imagen

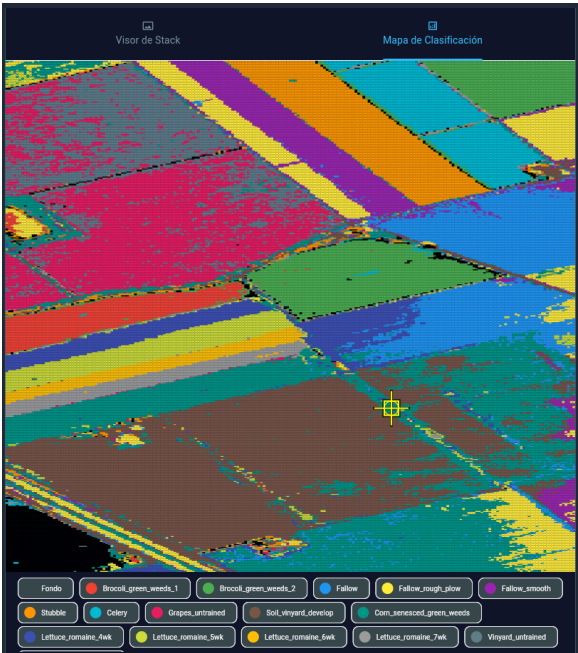


Figura A3: Mapa de clasificación de imagen

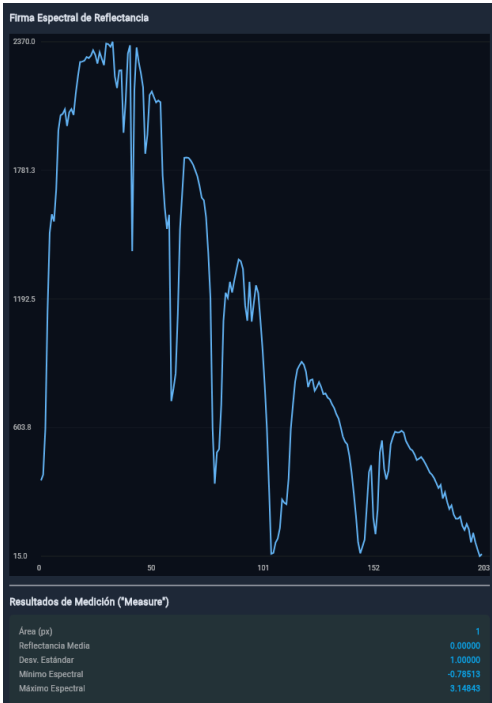


Figura A4: Firma espectral de píxel seleccionado.