Manual de Procedimientos

El propósito de este manual de procedimientos es ofrecer una orientación exhaustiva para la creación y conservación del proyecto de análisis de Cobertura Nubosa, enfocándose en el cálculo del Índice de Cobertura Nubosa (CCI) basándose en imágenes. Este sistema es crucial para el seguimiento del clima, aportando a la interpretación y pronóstico de patrones climáticos.

Propósito

El objetivo del proyecto de Cobertura Nubosa es crear un sistema de software sólido y eficaz que determine el Índice de Cobertura Nubosa (CCI) a partir de imágenes meteorológicas, favoreciendo así áreas como la aviación, la agricultura y la administración de recursos naturales. El objetivo es establecer procesos normalizados que garanticen el funcionamiento adecuado del software, asegurando la observancia de las regulaciones y directrices pertinentes en cuanto a protección de datos y derechos de autor. Además, se busca simplificar el cuidado y la renovación del sistema a lo largo del tiempo, proporcionando herramientas visuales como imágenes en color y negro que destaquen las nubes para simplificar la comprensión de los datos.

Leyes y lineamientos

Este segmento define las bases legales y regulaciones que sustentan el progreso del proyecto de cobertura de nubosidad. Este proyecto comprende varias fases de análisis de imágenes, que incluyen la extracción y procesamiento de píxeles, la modificación de colores y la determinación de índices de nubosidad, todo ello basado en un conjunto de imágenes suministradas específicamente para estos propósitos. No solo se garantiza que el proyecto se adhiera a las normativas locales e internacionales relevantes, sino que también establece los fundamentos para un desarrollo ético y responsable, honrando la privacidad, la propiedad de la información y los principios de transparencia. A continuación, se describen los elementos fundamentales del marco legal, acuerdos internacionales, leyes, códigos y normativas vinculados al proyecto.

Marco Juridico

Este proyecto de cobertura de nubosidad se lleva a cabo bajo el marco legal mexicano, respetando la **Ley Federal del Derecho de Autor** a través del empleo legítimo de imágenes suministradas específicamente para este propósito, previniendo cualquier violación de derechos de autor y reproducciones no autorizadas.

Tratados Internacionales

El plan de cobertura de nubosidad concuerda con acuerdos internacionales fundamentales en la salvaguarda de los derechos de imagen y datos. El **Convenio de Berna para la Protección de Obras Literarias y Artísticas** garantiza el respeto a los derechos de autor en obras visuales, incluyendo imágenes en formato digital, asegurando que cualquier recurso externo que se pudiera emplear en el futuro se adhiera a estas directrices. Igualmente, el **Tratado de la OMPI sobre Derecho de Autor (WCT)**, gestionado por la Organización Mundial de la Propiedad Intelectual, expande la salvaguarda de derechos en medios digitales y dicta métodos sugeridos para la utilización de obras visuales en proyectos tecnológicos. Respecto a la protección de datos, se adopta el **Reglamento General de Protección de Datos de la Unión Europea (GDPR)** como guía para asegurar la privacidad y el uso responsable de cualquier información visual o de localización, garantizando un enfoque ético en la gestión de datos digitales.

Leyes

A escala nacional, las normativas vinculadas al manejo y tratamiento de datos digitales e imágenes en proyectos de investigación o análisis son relevantes. En nuestra situación, al incorporar un lector de píxeles y otras capacidades de procesamiento, cumplimos con la **Ley Federal de Protección de Datos Personales en Posesión de los Particulares** en México, si las imágenes contenían información que pudiera estar relacionada con individuos o lugares de interés. A pesar de que en este proyecto no se trata directamente la información personal, la legislación también define normas para el manejo seguro de la información. Además, se respeta la **Ley de Ciencia y Tecnología**, que fomenta el progreso en ciencia y tecnología, dirigiendo este proyecto hacía progresos en el estudio de las imágenes.

Códigos

En el campo de la ética y la profesión, se destaca el **Código de Conducta del Consejo Nacional de Ciencia y Tecnología**. Este proyecto respeta los derechos de autor de las imágenes, empleando solo las que se han proporcionado de forma legal y autorizada. Además, se considera la integridad del análisis, como en el procedimiento de recortar áreas negras en las fotografías y modificar los colores de nubes a blanco y cielo a negro, para prevenir manipulaciones fraudulentas y preservar la imparcialidad en el procesamiento.

Reglamentos

Además, cumplimos con el **Reglamento de la Ley General del Equilibrio Ecológico y la Protección al Ambiente (LGEEPA)**. Este reglamento dicta normas acerca de la conservación y recuperación del balance ecológico en México, y su contexto puede orientar la creación de instrumentos tecnológicos y científicos vinculados al seguimiento ambiental, como el índice de cobertura de nubes (CCI)..

Generalidades

¿Cómo está formado el sistema?

El proyecto de cobertura de nubosidad está concebido para llevar a cabo un estudio minucioso de las imágenes a través de varias funcionalidades particulares. Primero, se pondrá en marcha un lector de píxeles, esencial para obtener los datos de cada píxel y guardarlos en un arreglo, lo cual es vital para el procesamiento subsiguiente y la determinación de índices de nubosidad. Para optimizar el análisis, se implementará una función de recorte de imágenes que suprimirá las áreas de color negro a través de la identificación de los píxeles correspondientes, garantizando que el análisis se enfoque únicamente en las zonas pertinentes. Además, se desarrollará una función de cambio de colores que convertirá los píxeles de las nubes en blanco y los del cielo en negro, simplificando de esta manera la observación y el estudio de la cobertura de nubesidad. Finalmente, se pondrá en marcha un calculador del Índice de Nubosidad (INN), el cual ofrecerá una medición numérica de la nubosidad en las imágenes, siendo este cálculo simple pero esencial para la comprensión de los resultados.

¿A quién está dirigido?

El proyecto está enfocado en investigadores, meteorólogos, alumnos de ciencias ambientales y cualquier individuo que tenga interés en el estudio de datos visuales vinculados a la nubosidad. Además, resulta beneficioso para expertos en tecnología que desean incorporar herramientas de procesamiento de imágenes en sus trabajos.

Roles

- **Usuarios:** Toda persona que interactúe con el sistema, realizando análisis y consultas sobre las imágenes procesadas.
- **Administrador de contenido:** Responsables de administrar la información y los recursos del proyecto, garantizando que la información se mantenga al día y esté al alcance de los usuarios. Asignados a: Luis Juárez Erick, Barrera Hérnandez Tomás, Rodríguez Jiménez Brenda, Sánchez Gónzalez Oscár Iván y Peña Villegas Diego Eduardo

- **Web Master:** Responsable de supervisar el progreso técnico y la aplicación de los instrumentos del proyecto. Este papel implica garantizar que el sistema opere eficazmente y alcance los objetivos definidos. Asignado a: Peña Villegas Diego Eduardo

---

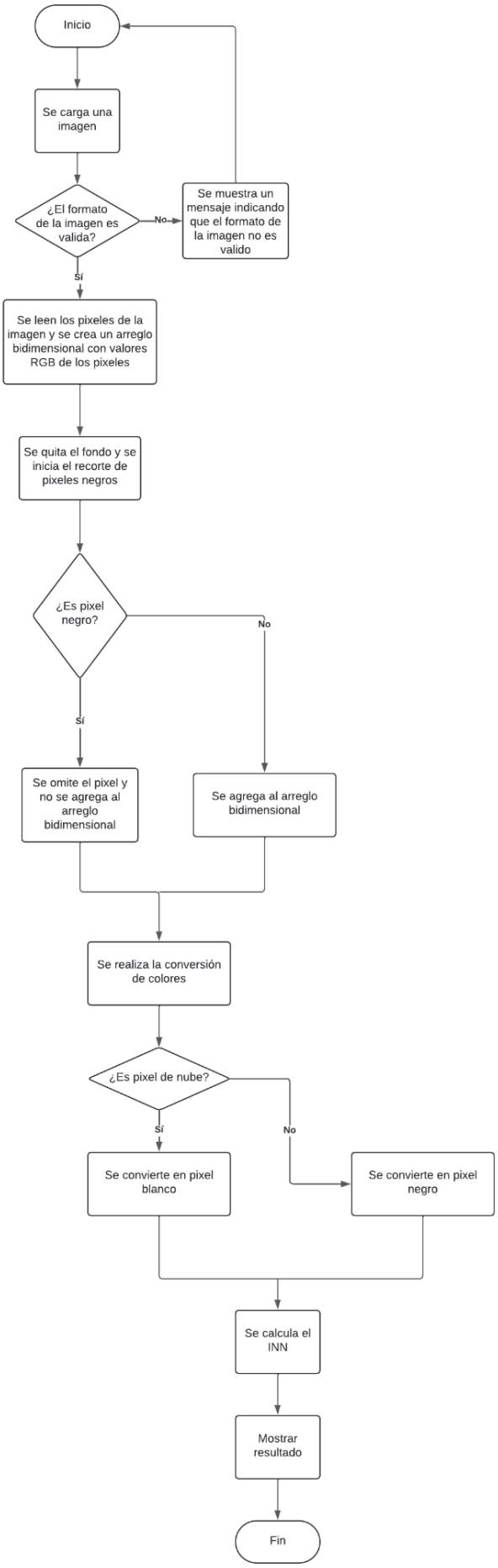
Información del software

Software:
1. Linux Fedora 40 (64 bits)
2. Ubuntu (64 bits)

Requisitos para ejecución y pruebas unitarias:
Tener instalado Java

Enfoque:
El programa está enfocado en el cálculo de cobertura de nubes de una imagen del cielo.

Diagrama de flujo:


Inicio
Se carga una imagen
¿El formato de la imagen es valida?
No
Se muestra un mensaje indicando que el formato de la imagen no es valido
Sí
Se leen los pixeles de la imagen y se crea un arreglo bidimensional con valores RGB de los pixeles
Se quita el fondo y se inicia el recorte de pixeles negros
¿Es pixel negro?
No
Sí
Se omite el pixel y no se agrega al arreglo bidimensional
Se agrega al arreglo bidimensional
Se realiza la conversión de colores
¿Es pixel de nube?
Sí
No
Se convierte en pixel blanco
Se convierte en pixel negro
Se calcula el INN
Mostrar resultado
Fin

Imágenes usadas:

Librerías usadas:

- **java.awt.Color:** Este import trae la clase Color de la biblioteca AWT. Donde Color permite representar colores en formato RGB (rojo, verde, azul), y se usa para manipular los colores de cada píxel en la imagen.
- **java.awt.image.BufferedImage:** Aquí estoy importando BufferedImage, una clase que permite trabajar directamente con imágenes. Con esta clase, la imagen se carga en la memoria, acceder a cada píxel, y modificarla fácilmente.
- **java.io.File:** File es una clase que utilizamos para manejar archivos y directorios. En el codigo, File me sirve para representar la imagen que voy a cargar y para guardar el archivo de imagen final.
- **javax.imageio.ImageIO:** Este import nos permite usar ImageIO, una clase útil para leer y escribir imágenes en varios formatos. ImageIO.read() carga una imagen desde un archivo, mientras que ImageIO.write() me permite guardarla una vez que la he procesado.
- **java.nio.file.Path:** Path nos permite manejar la ruta de archivos de una manera más flexible y segura. Lo uso para definir las rutas del archivo de entrada (imagen) y del archivo de salida.

- **java.nio.file.Paths:** Paths nos ayuda a crear objetos Path desde una cadena de texto que contiene una ruta de archivo. Utilizamos Paths.get() para construir las rutas de los archivos de entrada y salida en mi programa.
- **import java.util.Scanner:** Este import nos permite usar Scanner, que facilita la lectura de datos desde la consola. En mi código, Scanner lee el nombre del archivo de imagen y cualquier otra información que el usuario quiera proporcionar al ejecutar el programa.

Funciones y clases:

Clase Pixel: Representa un píxel de una imagen con tres componentes de color: rojo, verde y azul (RGB).

- **rojo, verde, azul**: Variables de instancia que almacenan los valores de color del píxel en formato RGB.
- **Constructor (public Pixel(int rojo, int verde, int azul)):** El constructor inicializa un píxel con los valores de color proporcionados (rojo, verde y azul). Este constructor es utilizado en otras clases para representar un píxel en formato RGB. Este mismo, recibe tres parámetros (rojo, verde y azul) y los asigna a las variables correspondientes de la clase.

Clase Programa: La clase contiene las funciones principales para el funcionamiento del programa las cuales están relacionadas con con el procesamiento de imágenes y el cálculo de indicadores de cobertura nubosa. La clase también utiliza las importaciones relacionadas con la manipulación de imágenes y colores de la librería **java.awt.Color**, **java.awt.image.BufferedImage**, **java.io.File** y **javax.imageio.ImageIO:**

- **lectorPixeles (public static Pixel[][] lectorPixeles(String nombreArchivo) throws Exception):** La función lectorPixeles puede leer una imagen desde un archivo y transforma sus píxeles en un arreglo bidimensional de objetos Pixel. Primero, utiliza la clase File para generar un objeto que simboliza la ruta del archivo de imagen, basándose en el nombre del archivo que se obtuvo como argumento. Este objeto File pasa al metodo ImageIO.read(), que extrae la imagen del archivo y la guarda en un objeto BufferedImage. Después, se determinan el ancho y la altura de la imagen utilizando los métodos de getWidth() y getHeight(). A continuación, se genera un arreglo de Pixel en dos dimensiones acorde a las dimensiones de la imagen. Se realizan dos bucles anidados por cada píxel de la imagen (uno para las filas y otro para las columnas), y se extrae el color para cada píxel con getRGB(), el cual se divide en los valores de los elementos rojo, verde y azul mediante la clase Color. Estos valores se guardan en un objeto nuevo Pixel , que se ubica en la ubicación adecuada en el arreglo. Al final, la función devuelve el arreglo de píxeles.
- **calcularRadioCirculo (public static int calcularRadioCirculo(Pixel[][] pixeles)):** Determina la radio de un círculo que rodea la imagen, empleando el centro de la

matriz de píxeles. Primero, establece el núcleo de la imagen basándose en sus dimensiones, y posteriormente determina el radio en las cuatro direcciones (izquierda, derecha, arriba y abajo). En cada sentido, rastrea los píxeles desde el medio hacia el extremo hasta localizar el primer píxel con valores bajos en los tres elementos de color (rojo, verde y azul), lo que señala un margen o zona vacía. El radio de cada dirección se determina a partir de la separación entre el centro y este píxel vacío, y finalmente, la función proporciona el radio más bajo entre las cuatro direcciones, garantizando que el círculo que alberga la imagen sea el más pequeño posible

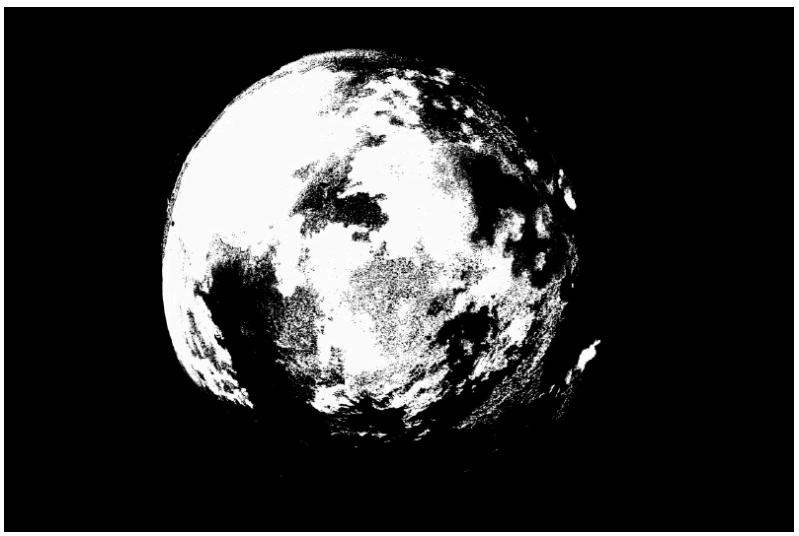
- **esNube (public static boolean esNube(int r, int g, int b):** Evalúa si un píxel simboliza una nube basándose en sus elementos de color RGB. Para llevarlo a cabo, inicialmente se determina el brillo del píxel, que es la proporción entre el valor máximo del color y 255. Después, determina la saturación, que evalúa la pureza del color (un valor reducido de saturación señala un color gris o similar al blanco). Además, determina el tono, que es la inclinación en el círculo cromático que simboliza el color del píxel. Si el tono se encuentra dentro del espectro de colores específicos azules, verdes o rojos, se adapta. Finalmente, determina que el píxel simboliza una nube si presenta un alto brillo, baja saturación y su tono no se encuentra dentro del espectro específico de colores (que serían azules o morados, característicos del cielo despejado), retornando true si satisface estas condiciones y false si no lo hace.
- **convertirBlancoYBlanco (public static Pixel[][] convertirBlancoYNegro(Pixel[][] pixeles, int radio)):** Transforma una imagen a blanco y negro, procesando únicamente los píxeles dentro de un círculo con una radio específica. Por lo tanto, determina la distancia cuadrática de cada píxel en relación al centro de la imagen empleando las coordenadas de la columna y la fila. Si la distancia se encuentra dentro del radio determinado, verifica si el píxel se asemeja a una nube utilizando la función esNube, otorgándole un valor blanco (255, 255, 255) si se trata de una nube, o negro (0, 0, 0) si no es así. Los píxeles que se encuentran fuera del círculo son directamente atribuibles al color negro. Finalmente, proporciona una nueva matriz de píxeles en color y negro que ilustra la imagen que ha sido procesada.
- **guardarImagenRecortada (public static void guardarImagenRecortada(Pixel[][] pixeles, String salida)):** Guarda una imagen en formato PNG en el archivo "Imagenes", a partir de una matriz de píxeles en color blanco y negro. Inicialmente, se genera un objeto BufferedImage con las dimensiones de la matriz de píxeles y la clase TYPE_INT_RGB, que facilita y permite el almacenamiento de colores. Después, explora la matriz de píxeles, transforma cada píxel en un objeto de color empleando los valores RGB del mismo, y los configura en el BufferedImage utilizando el método setRGB. Luego, elabora la dirección de salida fusionando el nombre del archivo suministrado con la dirección de la carpeta "Imagenes". Emplea la clase File para elaborar el archivo de salida y confirmar la existencia del directorio, generándolo si se requiere con mkdirs. Si ya existe el archivo, se comunica al usuario que será sobrescrito. Finalmente, usa ImageIO.write, para almacenar la imagen en el archivo específico. Si se produce algún fallo durante el procedimiento, se registra una excepción y se publica el aviso de error con e.printStackTrace().

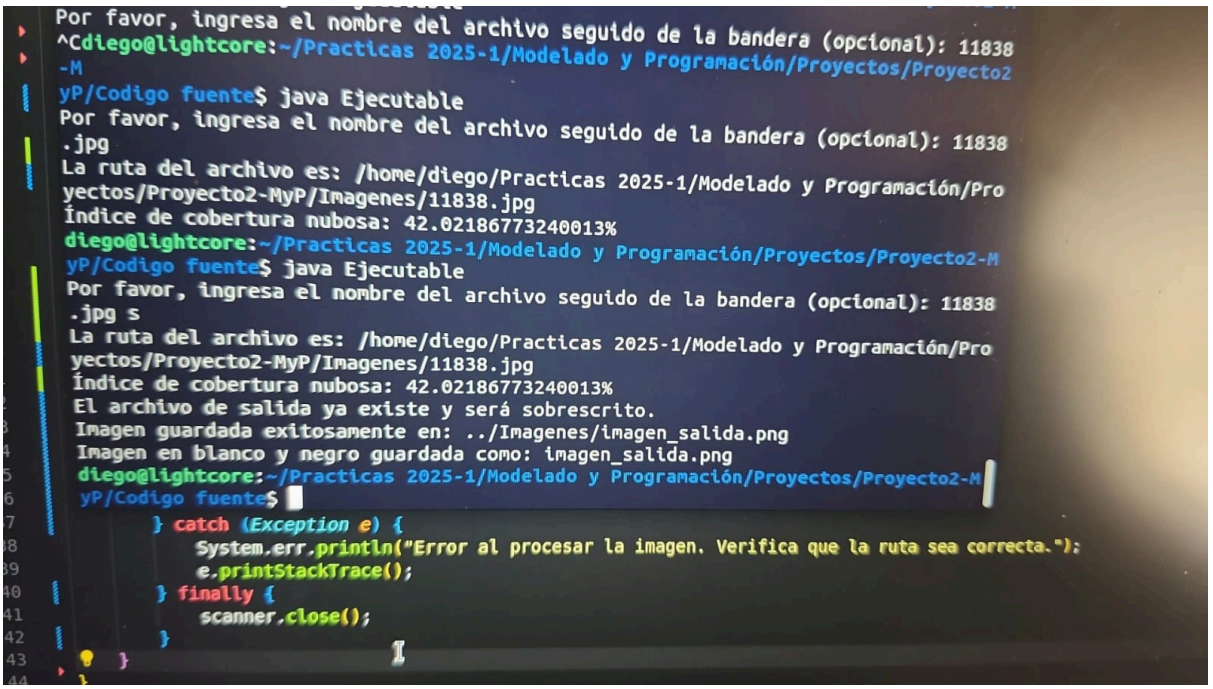
- **calcularIndice (public static double calcularIndice(Pixel[][] pixeles, int radio)):** Calcula el indice de cobertura nubiosa dentro de un círculo determinado por un radio en una matriz de píxeles en color y negro. Primero, determina el núcleo de la matriz de píxeles y después examina cada píxel de la imagen, determinando si está dentro del círculo mediante la fórmula de la distancia al cuadrado. Si el píxel se encuentra dentro del círculo, se considera un píxel válido. Después, comprueba si el píxel es blanco (que simboliza una nube) al contrastar sus valores RGB con 255 en los tres elementos (rojo, verde y azul). Si el píxel se encuentra en blanco, aumenta el contador de píxeles en la memoria. Finalmente, calcula el porcentaje de píxeles de nube en el círculo mediante la división del número de píxeles de nube entre el total de píxeles válidos y la multiplicación por 100 para determinar el porcentaje de cobertura de la nube.

Clase ejecutable: Sirve como punto de entrada para el programa, donde se ejecuta la lógica principal que gestiona el flujo general del proceso.utilizando main:

- **main(String[] args):** El método le solicita al usuario que brinde el nombre de un archivo de imagen y, de manera opcional, una bandera ("S") para almacenar la imagen recortada. Con la clase Scanner de java.util, el software procesa la información del usuario, mientras que con las clases Paths y Path de java.nio.file, gestiona el camino del archivo de forma segura. Después, el principal llama a los métodos en la clase Programa para interpretar los píxeles de la imagen, determinar el radio del círculo de interés, transformar la imagen a color y negro, y determinar el índice de cobertura de la nube. Si la bandera es "S", el software almacena la imagen que ha sido procesada en un archivo en un lugar determinado.

Imagen generada y fotografías:





Procedimientos

Mantenimiento: Para asegurar la escalabilidad y eficacia a largo plazo de nuestro sistema de análisis de cobertura nubosa, es crucial tener en cuenta algunos factores esenciales para el mantenimiento futuro del proyecto:

- **Optimización de Procesamiento de Imágenes:** Con el incremento del número de imágenes a procesar, será imprescindible perfeccionar los algoritmos para optimizar los tiempos de procesamiento y minimizar el consumo de recursos. Más adelante, podrían aplicarse métodos de paralelización o la utilización de procesamiento en GPU para agilizar el análisis.
- **Adaptación a Nuevas Tecnologías:** En el futuro, la cámara o los sensores empleados en la toma de imágenes podrían ser actualizados. Así pues, será vital que el sistema tenga la suficiente flexibilidad para ajustarse a imágenes de alta resolución o variados formatos sin la necesidad de reestructuraciones considerables.
- **Mantenimiento de la Infraestructura:** Conforme el sistema progresa, la gestión y conservación de datos se tornarán más complejas. La implementación de una solución escalable en la nube para el almacenamiento y procesamiento de imágenes contribuirá a garantizar que el sistema pueda gestionar grandes cantidades de información sin sacrificar eficacia.
- **Soporte para Integración con Otros Sistemas:** A futuro, podría ser imprescindible fusionar este sistema con otras plataformas de vigilancia climática, aplicaciones para móviles o APIs públicas para ampliar su capacidad de funcionamiento. Un diseño modular y centrado en los servicios simplificará dicha integración.

Actualizaciones: En las próximas versiones del sistema para el análisis de cobertura nubosa, se tomarán en cuenta las siguientes mejoras y actualizaciones:

- **Gráfica de Interfaz de Usuario (GUI)**: Crear una interfaz gráfica intuitiva que facilite a los usuarios la interacción con el sistema. Esta interfaz gráfica proporciona alternativas para insertar imágenes, mostrar resultados en tiempo real y modificar parámetros del análisis.
- **Conexión con Formatos de Imagen de Vanguardia**: Incrementar la compatibilidad con diversos formatos de imagen, como TIFF o RAW, facilitará a los usuarios el manejo de un mayor número de archivos de imagen y potenciará la calidad de la información de entrada.
- **Implementación de Algoritmos de Aprendizaje Automático**: Incorporar algoritmos de inteligencia artificial para optimizar la categorización de nubes y cielos. Esto facilitará un reconocimiento más exacto de los atributos de las imágenes, mejorando la transformación de colores y el cálculo del INN.
- **Exportación de Resultados en Diferentes Formato**s: Facilitar la exportación de los hallazgos del análisis en diversos formatos, tales como CSV, JSON o XML, para simplificar su utilización en otras aplicaciones o su análisis futuro.
- **Generación de Informes Automáticos**: Elaborar una función que produzca reportes automáticos que describen los resultados del análisis, incorporando gráficos, estadísticas y el INN, lo cual simplificará la exposición de los datos a los usuarios o a otras partes interesadas.
- **Integración con Sensores Meteorológicos**:: Futuras actualizaciones pueden incluir la integración del sistema con datos de sensores meteorológicos en tiempo real, lo que proporcionará un contexto adicional y enriquecerá el análisis de nubosidad.
- **Capacidad de Procesamiento en Lote**: Implementar la capacidad de procesar múltiples imágenes a la vez, lo que aumentaría la eficiencia del sistema y permitiría realizar análisis masivos en un solo proceso.

Aportaciones al manual: Las aportaciones al manual contendrán documentación exhaustiva acerca de diversos elementos fundamentales del sistema, iniciando con una explicación del funcionamiento de cada componente, en la que se detalla el objetivo y el funcionamiento del lector de píxeles, la función de corte, la transformación de colores y el calculador del INN. Además, se ofrecerán directrices de mantenimiento que proporcionarán directrices precisas sobre la manera de realizar el mantenimiento del sistema, incluyendo la optimización y las actualizaciones requeridas. Adicionalmente, se incluirán ejemplos prácticos y casos de uso que demuestren el uso de cada función del sistema y la interpretación de los resultados logrados, favoreciendo de esta manera la comprensión y uso del sistema por los usuarios.

Bibliografía

Ley Federal del Derecho de Autor. (2020, 1 julio). LEY FEDERAL DEL DERECHO DE AUTOR. Recuperado 1 de noviembre de 2024, de https://www.diputados.gob.mx/LeyesBiblio/pdf/LFDA.pdf

Convenio de Berna para la Protección de las Obras Literarias y Artísticas. (s. f.). https://www.wipo.int/treaties/es/ip/berne/

Tratado de la OMPI sobre Derecho de Autor. (s. f.). https://www.wipo.int/treaties/es/ip/wct/

Reglamento - 2016/679 - EN - GDPR - EUR-LEX. (s. f.). https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A32016R0679

LEY FEDERAL DE PROTECCIÓN DE DATOS PERSONALES EN POSESIÓN DE LOS PARTICULARES. (2010, 5 julio). LEY FEDERAL DE PROTECCIÓN DE DATOS PERSONALES EN POSESIÓN DE LOS PARTICULARES. Recuperado 1 de noviembre de 2024, de http://www.diputados.gob.mx/LeyesBiblio/pdf/LFPDPPP.pdf

LEY DE CIENCIA y TECNOLOGÍA. (2020, 6 noviembre). LEY DE CIENCIA y TECNOLOGÍA. Recuperado 1 de noviembre de 2024, de https://sep.gob.mx/work/models/sep1/Resource/15131/2/images/ley_ciencia_tecnologia_01_2020.pdf

CÓDIGO DE CONDUCTA DEL CONSEJO NACIONAL DE CIENCIA y TECNOLOGÍA. (2022, junio). CÓDIGO DE CONDUCTA DEL CONSEJO NACIONAL DE CIENCIA y TECNOLOGÍA. Recuperado 1 de noviembre de 2024, de https://conahcyt.mx/wp-content/uploads/administracion_conacyt/integridad_publica/codigos_de_conducta/Codigo_de_Conducta_Conacyt_2022.pdf

REGLAMENTO DE LA LEY GENERAL DEL EQUILIBRIO ECOLÓGICO y LA PROTECCIÓN AL AMBIENTE EN MATERIA DE EVALUACIÓN DEL IMPACTO AMBIENTAL. (2014, 31 octubre). REGLAMENTO DE LA LEY GENERAL DEL EQUILIBRIO ECOLÓGICO y LA PROTECCIÓN AL AMBIENTE EN MATERIA DE EVALUACIÓN DEL IMPACTO AMBIENTAL. Recuperado 1 de noviembre de 2024, de https://www.diputados.gob.mx/LeyesBiblio/regley/Reg_LGEEPA_MEIA_311014.pdf

- *Color (Java Platform SE 8). (2024, 30 septiembre).* *https://docs.oracle.com/javase/8/docs/api/java/awt/Color.html*

- *BufferedImage (Java Platform SE 8). (2024, 30 septiembre).* *https://docs.oracle.com/javase/8/docs/api/java/awt/image/BufferedImage.html*

- *File (Java Platform SE 8). (2024, 30 septiembre).* *https://docs.oracle.com/javase/8/docs/api/java/io/File.html*

- *ImageIO (Java Platform SE 8). (2024, 30 septiembre).* *https://docs.oracle.com/javase/8/docs/api/javax/imageio/ImageIO.html*

- *Path (Java Platform SE 8). (2024, 30 septiembre).* *https://docs.oracle.com/javase/8/docs/api/java/nio/file/Path.html*

- *Paths (Java Platform SE 8). (2024, 30 septiembre).*

  https://docs.oracle.com/javase/8/docs/api/java/nio/file/Paths.html

- *Scanner (Java Platform SE 8). (2024, 30 septiembre).*

  https://docs.oracle.com/javase/8/docs/api/java/util/Scanner.html