




Presentación Cobertura nubosa

**Presentado por Luis Juárez Erick, Barrera
Hérmendez Tomás, Rodríguez Jiménez Brenda,
Sánchez González Oscár Iván y Peña Villegas Diego
Eduardo**

SECCIONES

- Descripción del problema
 - Análisis del problema
 - Selección a la mejor alternativa
 - Diagrama de flujo
 - Pensando a futuro (Futuros Mantenimientos)
 - ¿Por qué nuestra solución es la mejor?
- 

DESCRIPCIÓN DEL PROBLEMA



DESCRIPCIÓN DEL PROBLEMA

Dentro del marco del monitoreo atmosférico, resulta esencial cuantificar y examinar la cobertura de las nubes para efectuar pronósticos climáticos exactos. Nuestra respuesta soluciona este inconveniente mediante la captura de imágenes cada 15 minutos por una cámara de 360 grados dotada de una lente de gran angular. Estas fotografías, que ilustran la bóveda celeste, necesitan ser examinadas para establecer la proporción del cielo que está revestido de nubes.

La meta es crear un sistema que determine el Índice de Cobertura Nubosa (CCI), que se establece como la proporción de píxeles en la imagen que simbolizan nubes en comparación con el total de píxeles en la zona de interés.

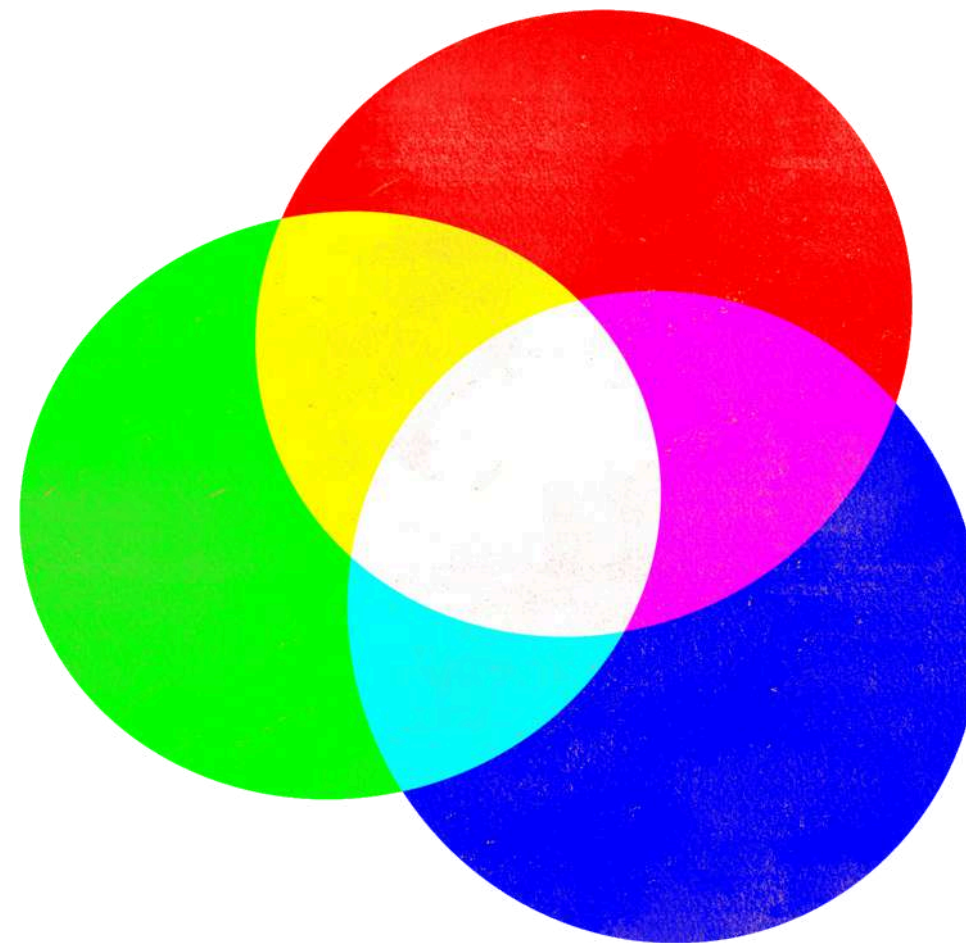
ANALISIS DEL PROBLEMA



ARSENAL

- **Lenguaje de programación:** Java
- **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 código, 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.
- **Imagenes proporcionadas hacia nosotros**



REQUISITOS FUNCIONALES

El usuario debe de interactuar con el programa de la siguiente forma:

- Abrir la terminal
- Situarse en la carpeta de Código fuente
- Debe compilar el archivo con `fpc Programa.pas`
- Ejecutar el programa mediante `./Programa -s`
- Una vez ejecutado el programa, se muestra en la terminal el porcentaje de el índice de cobertura nubosa de la imagen y en la carpeta imágenes de la carpeta Proyecto2-MyP se mostrara de igual manera la imagen ya procesada, apareciendo en blanco y negro.
- En donde la información e imágenes de salida son únicamente el porcentaje de índice de cobertura nubosa y la imagen procesada.

REQUISITOS NO FUNCIONALES

El programa tiene un buen rendimiento y cumple con su funcionalidad, ya que entrega los resultados que se solicitan en cuestión de poco tiempo. Además de ser portable, pues puede ser ejecutado en distintos sistemas operativos, como lo serían Linux o Windows.

El diseño es claro, con una cohesión alta y un acoplamiento bajo, por lo que el código puede ser modificado si se requiere, permitiendo futuras actualizaciones de ser necesario; además de que no permite errores graves o difíciles.

El programa no solo es intuitivo y amigable con el usuario, pues también es fácil de usar.



SELECCION A LA MEJOR ALTERNATIVA

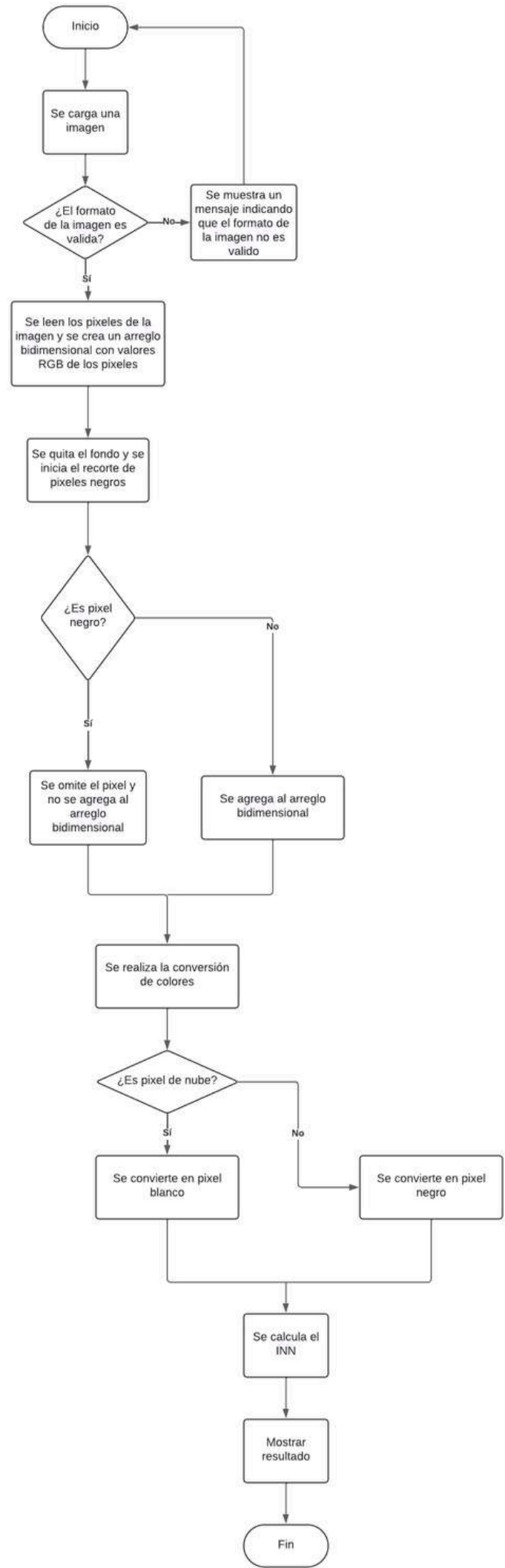
SELECCIÓN A LA MEJOR ALTERNATIVA

Nuestro programa además de ser la mejor alternativa por el fácil uso, no es un programa que sea saturado, siendo claro y conciso, así como la muestra de datos de cobertura es bastante acertada con una aproximación bastante cercana, con un margen de error pequeño.

Además de que el programa es eficiente, intuitivo, fácil de usar, no está saturado de información; por lo que es un programa sencillo, pero que cumple con los requisitos necesarios que fueron solicitados, sin información de más.



DIAGRAMA DE FLUJO



DESCRIPCIÓN DIAGRAMA DE FLUJO

Este diagrama de flujo describe el proceso para analizar una imagen y calcular el Índice de Nubes Negras (INN). Comienza cargando una imagen y verificando si su formato es válido; en caso de no serlo, muestra un mensaje de error y termina. Si el formato es correcto, lee los píxeles y crea un arreglo bidimensional con sus valores RGB. Luego, recorta el fondo negro al omitir los píxeles que sean negros y agregar solo los demás al arreglo. Posteriormente, convierte los colores de la imagen para diferenciar áreas de nubes y cielo: los píxeles de nube se vuelven blancos y los de cielo, negros. Finalmente, calcula el INN basado en los píxeles procesados y muestra el resultado.

PENSANDO A FUTURO (FUTUROS MANTENIMIENTOS



PENSANDO A FUTURO (FUTUROS MANTENIMEINTOS)

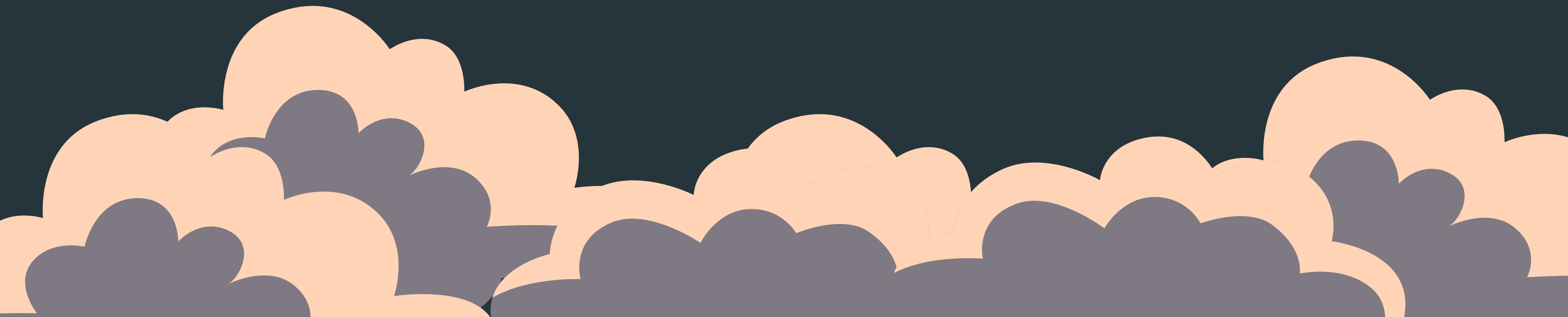
Para asegurar la escalabilidad y eficacia a largo plazo de nuestro sistema de análisis de cobertura nubosa, es crucial tener en cuenta algunos factores esenciales para el mantenimiento futuro del proyecto:

- **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 simplificaría dicha integración.

- **Seguridad y Monitoreo Continuo:** Con el crecimiento del proyecto, será crucial implementar estrategias de seguridad para salvaguardar tanto la información como el sistema en sí mismo. Además, la puesta en marcha de un seguimiento constante del sistema facilitará la identificación de problemas de desempeño o errores en tiempo real, garantizando un mantenimiento proactivo.



**POR QUE NUESTRA SOLUCIÓN ES LA
MEJOR**



POR QUE NUESTRA SOLUCIÓN ES LA MEJOR

Nuestra solución es la mejor porque cubre las necesidad de saber el indice de cobertura nubosa dependiendo de la imagen que sea escaneada, proporcionándonos los datos del indice de cobertura nubosa.

Así mismo, nuestro programa es fácil de usar, amigable con el usuario y sencillo, cumpliendo su propósito de los datos requeridos.

Bibliografía

- 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>



iMuchas gracias!