Herramienta de Procesamiento Digital de Imágenes en ambiente WEB (WEPI)

Documentación Técnica

Setiembre, 2016

INTRODUCCION

El presente documento es una recopilación de toda la documentación relacionada a los aspectos técnicos, en cuanto a infraestructura, plataforma, diseño y configuraciones del Sistema Wepi desarrollado en el marco del "Programa Paraguayo para el Desarrollo de la Ciencia y Tecnología" del Consejo Nacional de Ciencia y Tecnología, por la Facultad Politécnica de la Universidad Nacional de Asunción.

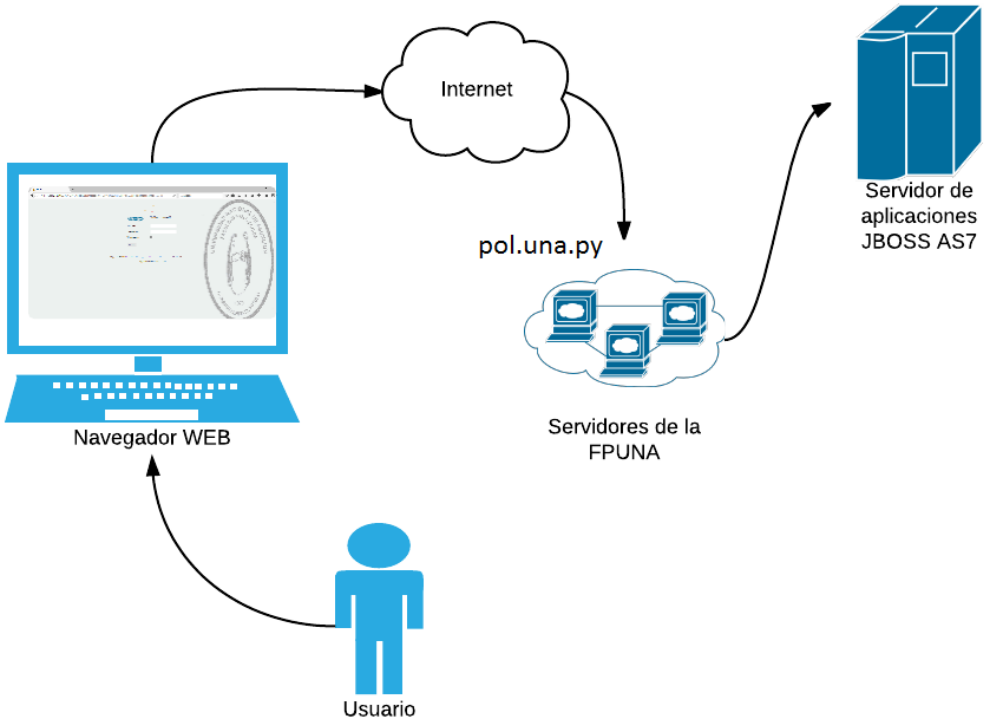
En el primer apartado se describe el esquema general así como la arquitectura del sistema considerando la plataforma tecnológica escogida para el desarrollo. Posteriormente, se presentan las características de la aplicación en base a un modelo de desarrollo.

Contenido

1. ARQUITECTURA

1.1. Esquema general de la solución

En el siguiente diagrama se esquematiza el contexto general de la solución desarrollada.



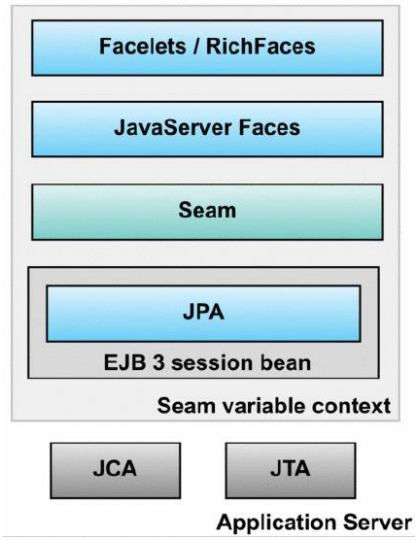
El esquema representa un ambiente WEB en donde cualquier usuario, mediante un navegador WEB e internet, podrá conectarse a la aplicación y realizar los diferentes algoritmos de procesamiento digital de imágenes desarrollados.

El servidor de aplicación utilizado es JBOSS AS7, siendo alojado en uno de los clusters de la Facultad Politécnica, por ello el acceso a la herramienta está bajo el dominio de la facultad (http://wepi.cc.pol.una.py).

1.2. Arquitectura de la aplicación

La arquitectura implementada por la aplicación es fundamentalmente cliente-servidor. Como framework de desarrollo se utilizó SeamFramework 2.2, el cual está desarrollado en Java, de código abierto, modular y extensible.

El siguiente diagrama nos muestra la arquitectura de la aplicación:



A continuación se describen los diferentes componentes de la arquitectura:

- **Facelets/RichFaces:** representan el componente visual de interacción con el usuario.
- **JavaServer Faces:** es una tecnología y framework para aplicaciones Java basadas en web que simplifica el desarrollo de interfaces de usuario en aplicaciones Java EE.
- **Seam:** permite combinar dos frameworks Enterprise JavaBeans (EJB3) y JavaServerFaces (JSF). A través del concepto de un contexto es posible acceder a cualquier componente EJB desde la capa de presentación refiriéndose a él mediante su nombre de componente seam.
- **Java Persistence API (JPA**): es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE). El sistema WEPI no hace uso de esta parte del componente, ya que no realiza ningún tipo de persitencia a una base de datos.

2. DISEÑO

2.1. Modelo de desarrollo

El diseño de la aplicación fue hecha mediante el modelo de desarrollo denominado MVC (Modelo-Vista-Controlador). El modelo separa los datos y la lógica de negocio de una aplicación, desde la interfaz de usuario, el módulo encargado de gestionar los eventos y las comunicaciones.

2.2. Seguridad

La aplicación cuenta con esquema de seguridad brindado por el framework Seam. El esquema presenta un modo de acceso a través de la autenticación del usuario por credenciales (Usuario/Password) permitiendo de esa manera el uso de la herramienta. El sistema WEPI no posee un sistema de gestión de credenciales, por lo que posee una credencial fija utilizada por todos los usuarios.

2.3. Modelo

El modelo en MVC es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información. El WEPI no realiza operaciones que requieran persistencia, pero si necesita de entidades que representen a las imágenes. Por ello, en un diagrama de clases podríamos representar los distintos tipos de imágenes de la siguiente forma:

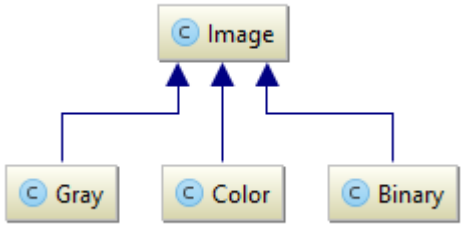


Figura 1. Clases que extienden de Image.

Las clases de Color (Color), Gris (Gray) y Binario (Binary) heredan de la clase Image ciertas propiedades comunes como también extienden métodos propios de una imagen cualquiera (**Figura 2**).

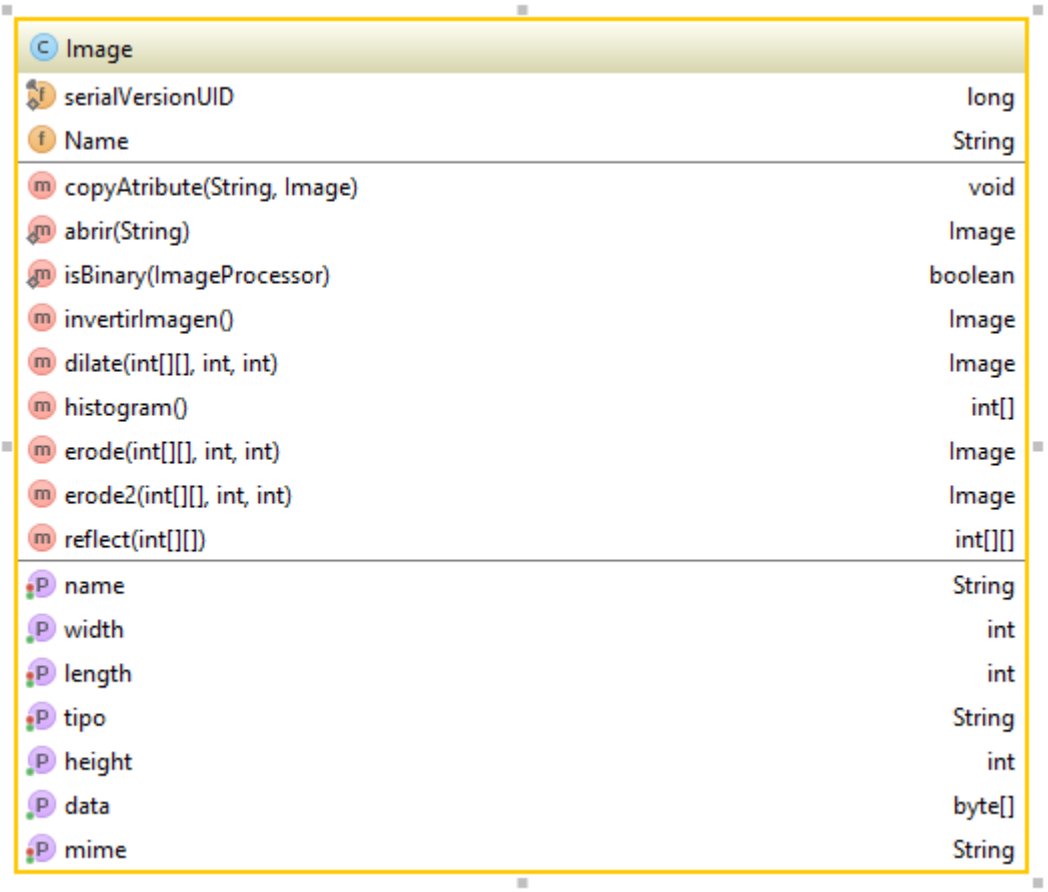


Figura 2. Clase Image.

2.3.1. Imágenes Binarias.

Las imágenes binarias extienden de la clase padre **Image**, heredando de ese modo atributos y comportamientos comunes para cualquier tipo de imagen. En la **Figura 3** podemos observar los métodos y atributos propios de la clase **Binary**.

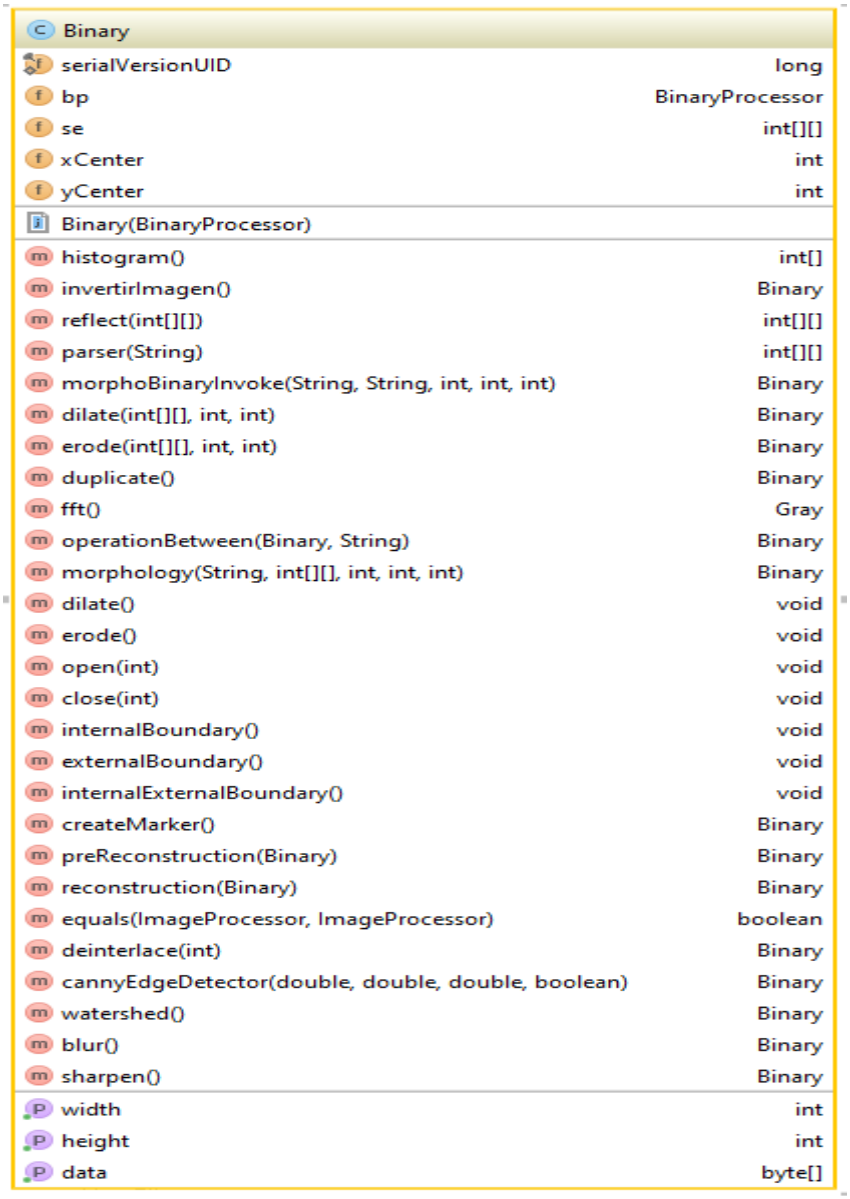


Figura 3. Clase Binaria.

En el modelo de la entidad Binary, adicionalmente fueron agregados procesos que representan a algoritmos de artículos científicos como también de una librería de Java para procesamiento digital de imágenes (ImageJ). La **Figura 4** nos muestra un diagrama de clases con los procedimientos.

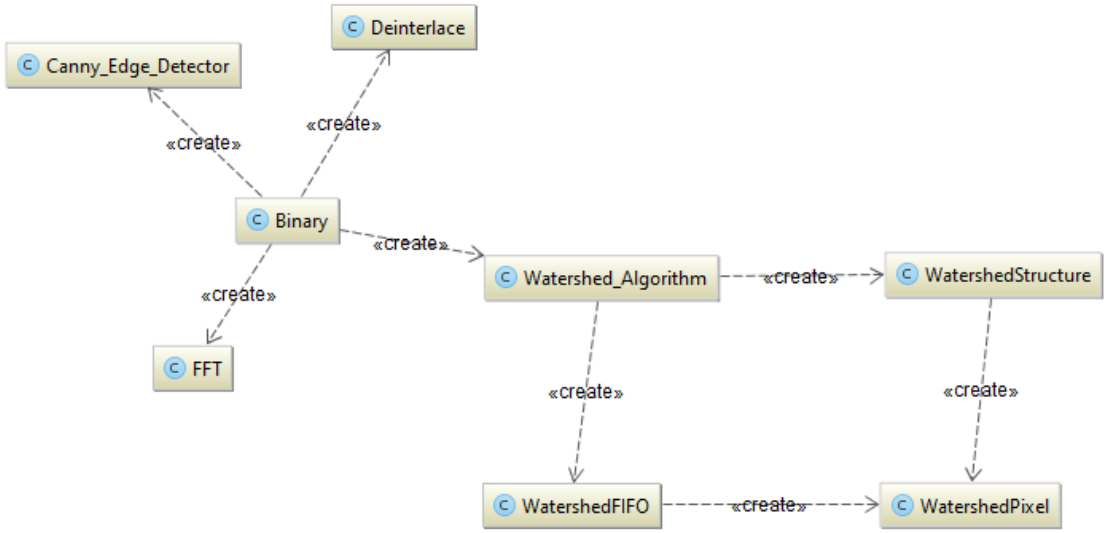


Figura 4. Diagrama de clases de los procedimientos para la clase Binary.

2.3.2. Imágenes en escala de Grises.

En el modelo de la entidad **Gray** fueron implementados procesos que representan a algoritmos de artículos científicos como también de una librería de Java para el procesamiento digital de imágenes (ImageJ). La **Figura 5** nos muestra un diagrama de clases con los procedimientos.

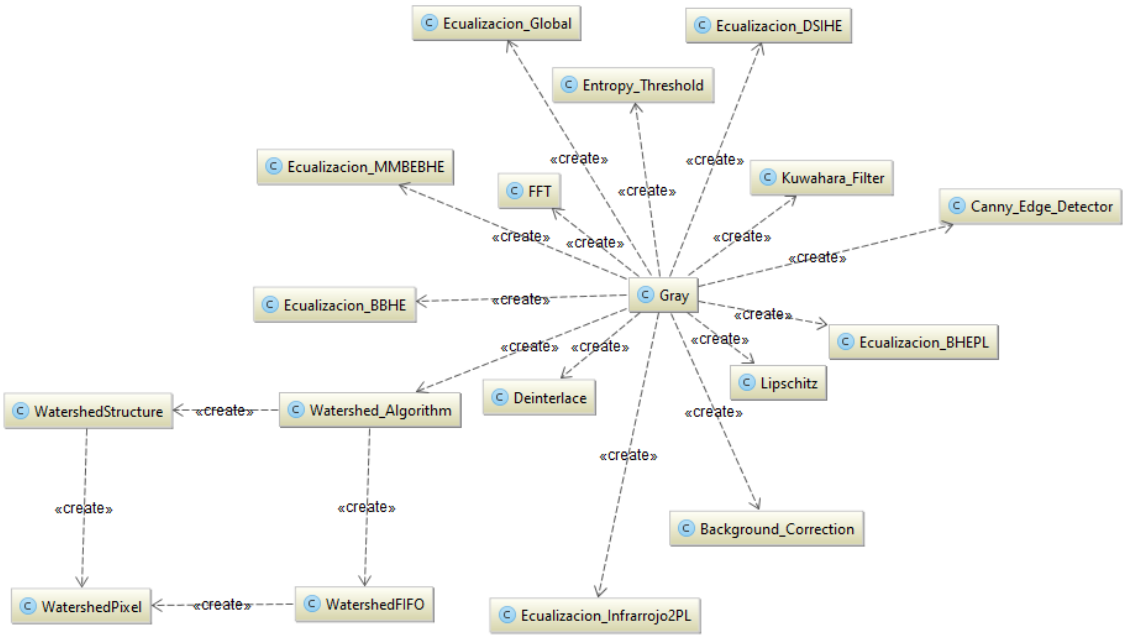


Figura 5. Diagrama de clases de los procedimientos para la clase Gray.

Las imágenes en escala de grises extienden de la clase **Image**, heredando de ese modo propiedades y métodos comunes para cualquier tipo de imagen. En la **Figura 6** podemos los métodos de la clase **Gray**.

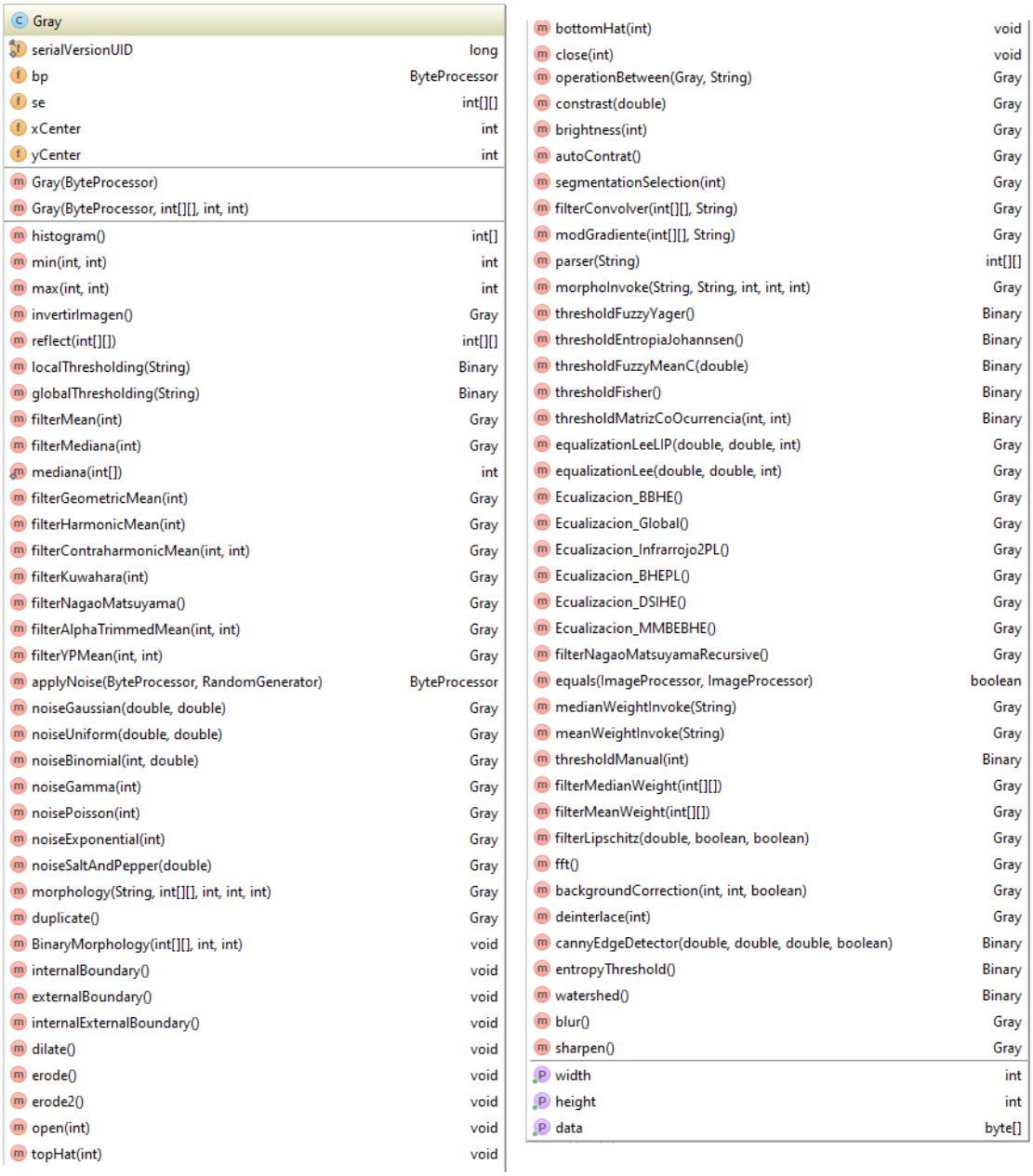


Figura 6. Clase Gray.

2.3.3. Imágenes en Color

Las imágenes en Color extienden de la clase **Image**, heredando de ese modo atributos y procedimientos comunes para cualquier tipo de imagen. En la **Figura 7** podemos observar los comportamientos y atributos propios de la clase **Color**.

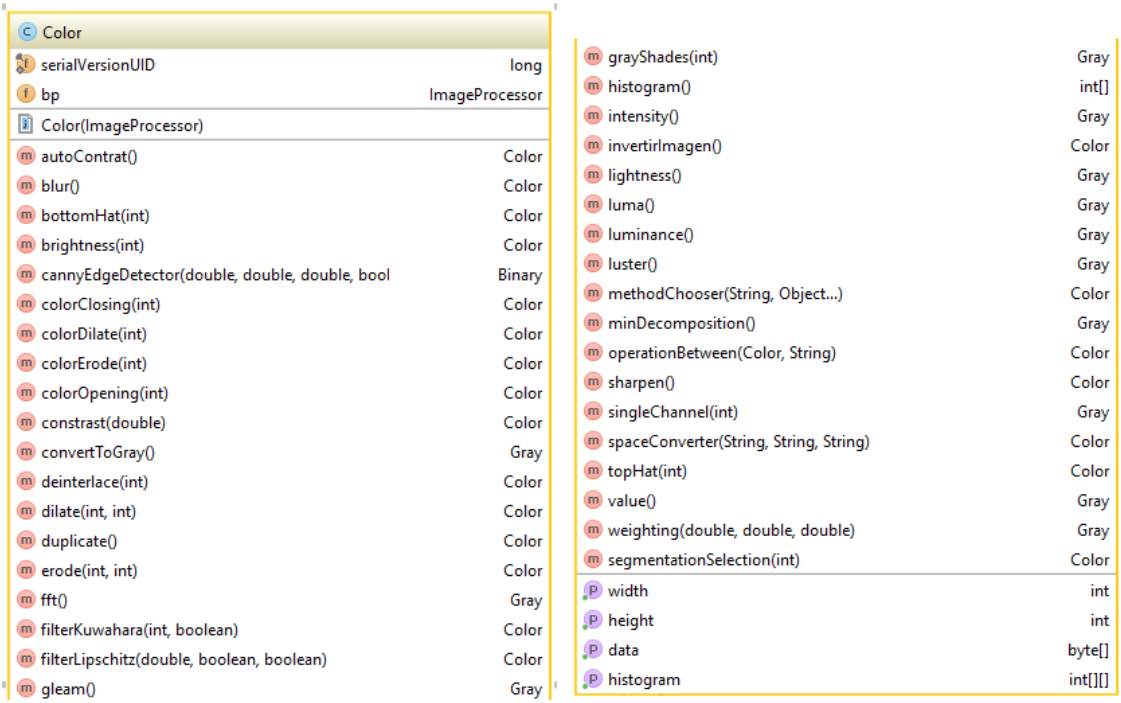


Figura 5. Clase Color.

Al modelo de la entidad Color, adicionalmente fueron agregados procesos que representan a algoritmos de artículos científicos como también de una librería de Java para el procesamiento digital de imágenes (ImageJ). La **Figura 8** nos muestra un diagrama de clases con los procedimientos.

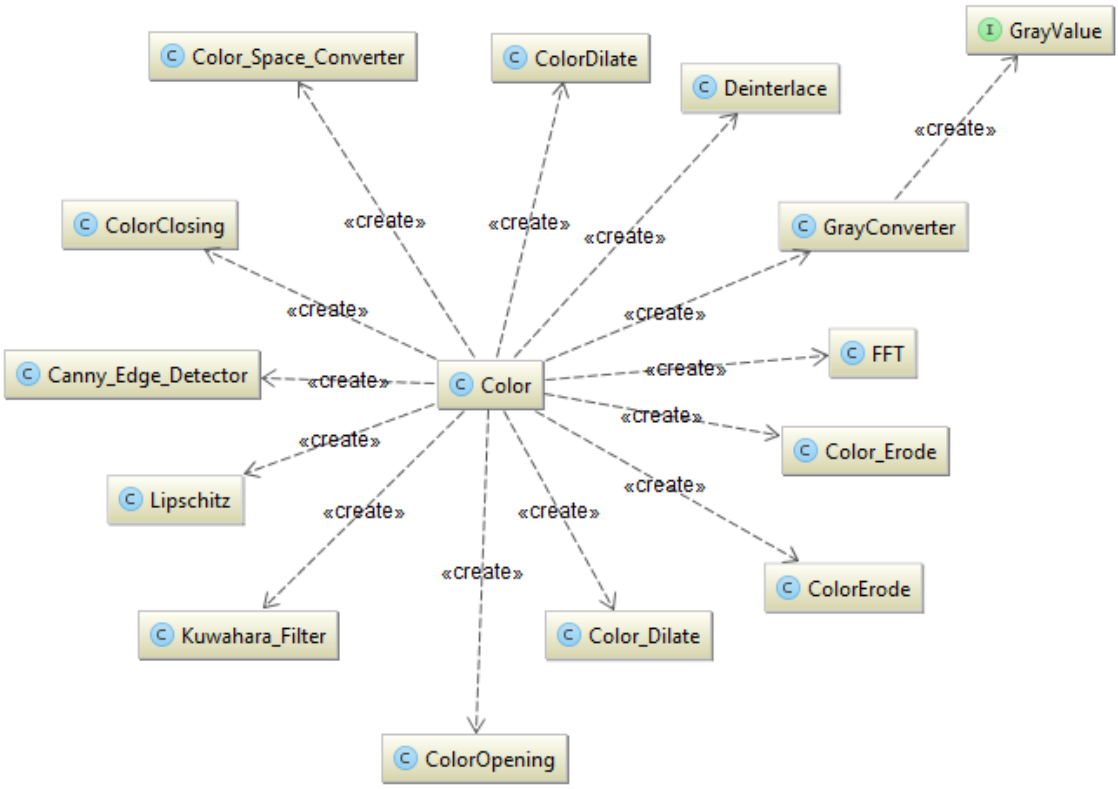


Figura 6. Diagrama de clases de los procedimientos para la clase Color.

2.4. Controlador

Responde a eventos (usualmente acciones del usuario) e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información (por ejemplo, aplicar un filtro mediana sobre la imagen). También puede enviar comandos a su vista asociada si se solicita un cambio en la forma en que se presenta el modelo (por ejemplo, cuando el filtro mediana aplicado retorna una imagen con el resultado), por tanto se podría decir que el controlador hace de intermediario entre la vista y el modelo.

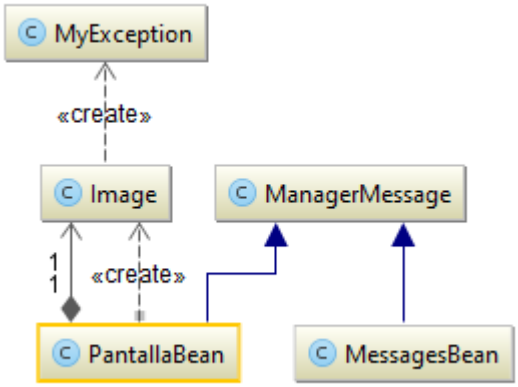


Figura 7. Diagrama de clase para la comunicación entre el modelo y controlador.

En la Figura 9 podemos observar a la clase PantallaBean, esta clase representa al controlador que actúa de intermediario entre la vista y el modelo. A su vez, existe MessagesBean que se encarga de interactuar con las excepciones de manera a exponer los mensajes al usuario.

2.5. Vista (Interfaz)

Presenta el modelo (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario), por tanto, requiere de dicho modelo la información que debe representar como salida. La **Figura 10** nos muestra cómo interactúa el usuario con la aplicación.

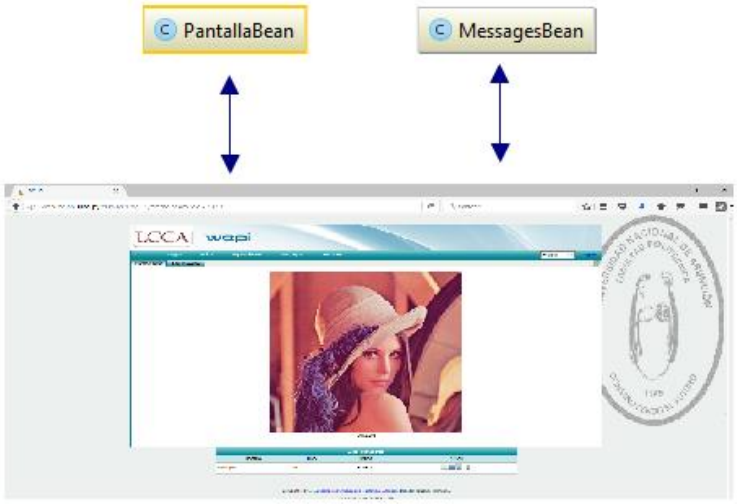


Figura 8. Interacción entre la vista y el controlador.