**Estrutura e Planejamento do Módulo**

**Versão:** 1.0  
**Data:** Outubro 2025  
**Autor:** Sistema de Processamento de Imagens  
**Tecnologia:** Python 3.x + Streamlit

**📋 Índice**

1. [Visão Geral do Sistema](#vis%C3%A3o-geral-do-sistema)
2. [Arquitetura do Módulo](#arquitetura-do-m%C3%B3dulo)
3. [Estrutura de Dados](#estrutura-de-dados)
4. [Funções Centrais](#fun%C3%A7%C3%B5es-centrais)
5. [Fluxo de Execução](#fluxo-de-execu%C3%A7%C3%A3o)
6. [Tecnologias e Dependências](#tecnologias-e-depend%C3%AAncias)
7. [Diagrama de Classes](#diagrama-de-classes)
8. [Casos de Uso](#casos-de-uso)

**1. Visão Geral do Sistema**

**1.1 Propósito**

O Sistema de Processamento de Imagens é uma aplicação web desenvolvida para realizar análise, processamento e realce de imagens digitais com validação quantitativa de qualidade através de métricas consagradas na literatura.

**1.2 Objetivos**

* Fornecer interface intuitiva para processamento de imagens
* Aplicar técnicas avançadas de pré-processamento, realce de nitidez e contraste
* Calcular métricas quantitativas de qualidade (PSNR, SSIM, LC, Edge Sharpness)
* Gerar relatórios técnicos em PDF
* Manter histórico completo de operações

**1.3 Público-Alvo**

* Operadores técnicos
* Administradores de sistema
* Pesquisadores em processamento de imagens
* Profissionais de análise de qualidade de imagem

**2. Arquitetura do Módulo**

**2.1 Padrão Arquitetural**

O sistema utiliza arquitetura **Model-View-Controller (MVC)** adaptada para Streamlit:

┌─────────────────────────────────────────────────┐

│ INTERFACE │

│ (Streamlit - View Layer) │

│ ┌──────────┬──────────┬──────────┬──────────┐ │

│ │ Upload │ Process │ Analysis │ Report │ │

│ └──────────┴──────────┴──────────┴──────────┘ │

└─────────────────────────────────────────────────┘

│

▼

┌─────────────────────────────────────────────────┐

│ CONTROLADOR PRINCIPAL │

│ (ImageProcessingSystem Class) │

│ ┌──────────────────────────────────────────┐ │

│ │ • Gerenciamento de Estado (Session) │ │

│ │ • Orquestração de Processamento │ │

│ │ • Controle de Fluxo de Dados │ │

│ └──────────────────────────────────────────┘ │

└─────────────────────────────────────────────────┘

│

▼

┌─────────────────────────────────────────────────┐

│ CAMADA DE PROCESSAMENTO │

│ ┌──────────────┬──────────────┬─────────────┐ │

│ │ Preprocessing │ Sharpening │ Contrast │ │

│ │ Filters │ Enhancing │ Enhancing │ │

│ └──────────────┴──────────────┴─────────────┘ │

└─────────────────────────────────────────────────┘

│

▼

┌─────────────────────────────────────────────────┐

│ CAMADA DE ANÁLISE │

│ ┌──────────────┬──────────────┬─────────────┐ │

│ │ Metrics │ Visual │ Report │ │

│ │ Calculation │ Analysis │ Generation │ │

│ └──────────────┴──────────────┴─────────────┘ │

└─────────────────────────────────────────────────┘

│

▼

┌─────────────────────────────────────────────────┐

│ PERSISTÊNCIA │

│ ┌──────────────────────────────────────────┐ │

│ │ • Session State (st.session\_state) │ │

│ │ • Histórico de Operações │ │

│ │ • Exportação PDF/PNG │ │

│ └──────────────────────────────────────────┘ │

└─────────────────────────────────────────────────┘

**2.2 Componentes Principais**

**2.2.1 Gerenciador de Interface (main)**

* **Responsabilidade:** Renderização da UI e gerenciamento de eventos
* **Tecnologia:** Streamlit
* **Componentes:** 5 abas (Upload, Processamento, Análise, Métricas, Relatório)

**2.2.2 Controlador Central (ImageProcessingSystem)**

* **Responsabilidade:** Orquestração de todas as operações
* **Padrão:** Singleton (via session\_state)
* **Funções:** Coordenar processamento e manter estado

**2.2.3 Módulos de Processamento**

* **Pré-processamento:** Filtros de suavização
* **Nitidez:** Realce de bordas e detalhes
* **Contraste:** Equalização e CLAHE

**2.2.4 Módulo de Análise**

* **Métricas:** Cálculos quantitativos
* **Visualização:** Gráficos e comparações
* **Relatórios:** Geração de documentos

**3. Estrutura de Dados**

**3.1 Session State (Estado da Aplicação)**

python

st.session\_state = {

*# Controle de Inicialização*

'initialized': bool,

*# Dados de Imagem*

'original\_image': np.ndarray, *# Imagem original carregada*

'normalized\_image': np.ndarray, *# Imagem normalizada 512x512*

'processed\_image': np.ndarray, *# Imagem processada atual*

*# Histórico e Logs*

'history': List[str], *# Lista de operações realizadas*

*# Métricas Calculadas*

'metrics': {

'PSNR': float, *# Peak Signal-to-Noise Ratio*

'SSIM': float, *# Structural Similarity Index*

'LC': float, *# Local Contrast*

'Edge\_Sharpness': float *# Nitidez de Bordas*

},

*# Informações do Usuário*

'user': str *# Nome do usuário atual*

}

**3.2 Estrutura de Imagens**

python

*# Formato de Imagem (NumPy Array)*

image: np.ndarray

shape: (512, 512, 3) *# Altura x Largura x Canais RGB*

dtype: np.uint8 *# Valores de 0 a 255*

color\_space: RGB *# Espaço de cor padrão*

**3.3 Estrutura de Histórico**

python

history\_entry: str

format: "[YYYY-MM-DD HH:MM:SS] Usuario: Ação realizada"

exemplo: "[2025-10-29 14:30:45] Operador: Imagem carregada e normalizada"

**4. Funções Centrais**

**4.1 Gerenciamento de Imagens**

**4.1.1 \_\_init\_\_(self)**

python

def \_\_init\_\_(self):

"""

Inicializa o sistema de processamento de imagens.

Responsabilidades:

- Inicializar session\_state se necessário

- Definir valores padrão para todas as variáveis

- Preparar sistema para primeira execução

Parâmetros: Nenhum

Retorno: None

Estado Modificado:

- st.session\_state.initialized: True

- st.session\_state.original\_image: None

- st.session\_state.processed\_image: None

- st.session\_state.normalized\_image: None

- st.session\_state.history: []

- st.session\_state.metrics: {}

- st.session\_state.user: "Operador"

"""

**4.1.2 log\_action(action: str)**

python

@staticmethod

def log\_action(action: str):

"""

Registra ação no histórico do sistema.

Responsabilidades:

- Criar timestamp da operação

- Adicionar usuário responsável

- Inserir no início do histórico

Parâmetros:

- action (str): Descrição da ação realizada

Retorno: None

Efeitos Colaterais:

- Adiciona entrada em st.session\_state.history

Exemplo:

>>> log\_action("Imagem carregada")

# Adiciona: "[2025-10-29 14:30:45] Operador: Imagem carregada"

"""

**4.1.3 load\_image(uploaded\_file)**

python

@staticmethod

def load\_image(uploaded\_file):

"""

Carrega e normaliza imagem para processamento.

Responsabilidades:

- Validar formato do arquivo (PNG/JPEG)

- Decodificar arquivo de imagem

- Converter de BGR para RGB

- Redimensionar para 512x512 pixels

- Atualizar session\_state

- Registrar operação no histórico

Parâmetros:

- uploaded\_file: Arquivo enviado via st.file\_uploader

Retorno:

- bool: True se sucesso, False se erro

Validações:

- Verifica integridade do arquivo

- Valida formato de imagem

Algoritmo:

1. Ler bytes do arquivo

2. Decodificar com cv2.imdecode()

3. Converter BGR → RGB

4. Redimensionar com INTER\_LANCZOS4

5. Salvar em session\_state

Exceções:

- ValueError: Formato inválido

- Exception: Erro genérico de carregamento

"""

**4.2 Pré-processamento**

**4.2.1 apply\_preprocessing(filter\_type, kernel\_radius, sigma)**

python

@staticmethod

def apply\_preprocessing(filter\_type: str, kernel\_radius: int, sigma: float):

"""

Aplica filtros de pré-processamento para suavização.

Responsabilidades:

- Validar parâmetros de entrada

- Aplicar filtro selecionado

- Atualizar imagem processada

- Registrar operação

Parâmetros:

- filter\_type (str): 'Gaussiano' ou 'Mediana'

- kernel\_radius (int): Tamanho do kernel (ímpar, 1-9)

- sigma (float): Desvio padrão gaussiano (0.5-2.0)

Retorno:

- bool: True se sucesso, False se erro

Validações:

- sigma deve estar entre 0.5 e 2.0

- kernel\_radius deve ser ímpar (ajusta automaticamente)

- Imagem deve estar carregada

Algoritmos:

Filtro Gaussiano:

- Aplica convolução gaussiana em cada canal RGB

- Utiliza skimage.filters.gaussian()

- Preserva faixa dinâmica [0, 255]

Filtro Mediana:

- Aplica filtro de mediana não-linear

- Utiliza cv2.medianBlur()

- Eficaz contra ruído salt-and-pepper

Complexidade: O(N\*M\*K²) onde N,M = dimensões, K = kernel\_radius

"""

**4.3 Realce de Nitidez**

**4.3.1 apply\_sharpening(method, weight, threshold, intensity)**

python

@staticmethod

def apply\_sharpening(method: str, weight: float, threshold: int, intensity: float):

"""

Aplica técnicas de realce de nitidez e detecção de bordas.

Responsabilidades:

- Detectar bordas/detalhes na imagem

- Realçar características desejadas

- Combinar com imagem original

- Registrar operação

Parâmetros:

- method (str): 'Laplaciano', 'Sobel' ou 'Alta Frequência'

- weight (float): Peso do filtro (0.1-3.0)

- threshold (int): Limiar para Sobel (10-200)

- intensity (float): Intensidade para alta freq (1.0-1.5)

Retorno:

- bool: True se sucesso, False se erro

Algoritmos:

1. Laplaciano:

- Operador de segunda derivada

- Detecta mudanças rápidas de intensidade

- Fórmula: ∇²f = ∂²f/∂x² + ∂²f/∂y²

- Máscara 3x3:

[ 0 1 0]

[ 1 -4 1]

[ 0 1 0]

- Combinação: I\_out = I\_in + weight \* Laplacian(I\_in)

2. Sobel:

- Operador de primeira derivada

- Calcula gradiente em X e Y

- Magnitude: G = √(Gx² + Gy²)

- Aplica threshold binário

- Máscaras 3x3:

Gx: [-1 0 1] Gy: [-1 -2 -1]

[-2 0 2] [ 0 0 0]

[-1 0 1] [ 1 2 1]

3. Alta Frequência:

- Unsharp masking

- I\_sharp = intensity\*I - (intensity-1)\*Blur(I)

- Realça componentes de alta frequência

Complexidade: O(N\*M) para cada método

"""

**4.4 Realce de Contraste**

**4.4.1 apply\_contrast\_enhancement(method, clip\_limit, tile\_size)**

python

@staticmethod

def apply\_contrast\_enhancement(method: str, clip\_limit: float, tile\_size: int):

"""

Aplica técnicas de realce de contraste global ou local.

Responsabilidades:

- Equalizar distribuição de intensidades

- Melhorar contraste local ou global

- Preservar informação de cor

- Registrar operação

Parâmetros:

- method (str): 'CLAHE (Local)' ou 'Equalização Global'

- clip\_limit (float): Limite de clipagem CLAHE (2.0-3.0)

- tile\_size (int): Tamanho do bloco CLAHE (4, 8, 16)

Retorno:

- bool: True se sucesso, False se erro

Algoritmos:

1. CLAHE (Contrast Limited Adaptive Histogram Equalization):

- Equalização adaptativa local

- Divide imagem em tiles (blocos)

- Equaliza histograma de cada tile

- Limita contraste com clip\_limit

- Interpola tiles para evitar artefatos

Processo:

a) Converter RGB → LAB

b) Extrair canal L (luminância)

c) Aplicar CLAHE no canal L

d) Recombinar com canais A e B

e) Converter LAB → RGB

Vantagens:

- Preserva detalhes locais

- Evita amplificação excessiva de ruído

- Melhor para imagens com iluminação não-uniforme

2. Equalização Global:

- Equalização de histograma tradicional

- Redistribui intensidades uniformemente

- Maximiza entropia da imagem

Processo:

a) Converter RGB → YCrCb

b) Extrair canal Y (luminância)

c) Calcular histograma cumulativo

d) Mapear intensidades: h(I) = (L-1) \* CDF(I)

e) Recombinar com Cr e Cb

f) Converter YCrCb → RGB

Função de transformação:

T(r) = (L-1) ∫₀ʳ p(w)dw

onde p(w) é PDF do histograma

Complexidade:

- CLAHE: O(N\*M + tiles\*256)

- Global: O(N\*M + 256)

"""

**4.5 Cálculo de Métricas**

**4.5.1 calculate\_metrics()**

python

@staticmethod

def calculate\_metrics():

"""

Calcula métricas quantitativas de qualidade de imagem.

Responsabilidades:

- Calcular PSNR entre original e processada

- Calcular SSIM estrutural

- Calcular contraste local (LC)

- Calcular nitidez de bordas

- Armazenar resultados

- Registrar operação

Parâmetros: Nenhum

Retorno:

- bool: True se sucesso, False se erro

Métricas Calculadas:

1. PSNR (Peak Signal-to-Noise Ratio):

- Mede relação sinal/ruído

- Fórmula: PSNR = 20 \* log₁₀(MAX / √MSE)

- MSE = (1/N\*M) Σ(I₁ - I₂)²

- Unidade: decibéis (dB)

- Alvo: ≥ 30 dB

- Interpretação:

\* > 40 dB: Excelente

\* 30-40 dB: Bom

\* 20-30 dB: Aceitável

\* < 20 dB: Pobre

2. SSIM (Structural Similarity Index):

- Mede similaridade estrutural

- Componentes: Luminância, Contraste, Estrutura

- Fórmula: SSIM(x,y) = l(x,y) \* c(x,y) \* s(x,y)

- onde:

\* l = (2μₓμᵧ + C₁)/(μₓ² + μᵧ² + C₁)

\* c = (2σₓσᵧ + C₂)/(σₓ² + σᵧ² + C₂)

\* s = (σₓᵧ + C₃)/(σₓσᵧ + C₃)

- Range: [-1, 1], ideal = 1

- Alvo: ≥ 0.85

- Melhor correlação com percepção humana que PSNR

3. LC (Local Contrast):

- Mede variação local de intensidade

- Fórmula: LC = σ / μ

- onde σ = desvio padrão, μ = média

- Valores mais altos = maior contraste local

4. Edge Sharpness:

- Mede definição de bordas

- Utiliza detector Canny

- Fórmula: ES = N\_edge\_pixels / N\_total\_pixels

- Range: [0, 1]

- Valores mais altos = bordas mais definidas

Complexidade: O(N\*M) para cada métrica

Dependências:

- skimage.metrics.structural\_similarity

- cv2.Canny

- numpy.std, numpy.mean

"""

**4.6 Geração de Relatórios**

**4.6.1 generate\_pdf\_report()**

python

@staticmethod

def generate\_pdf\_report():

"""

Gera relatório técnico em formato PDF.

Responsabilidades:

- Criar documento PDF estruturado

- Incluir imagens original e processada

- Adicionar métricas calculadas

- Incluir histórico de operações

- Adicionar metadados (data, usuário)

- Retornar bytes do PDF

Parâmetros: Nenhum

Retorno:

- bytes: Conteúdo do PDF ou None em caso de erro

Estrutura do Relatório:

1. Cabeçalho:

- Título do documento

- Data e hora de geração

- Nome do usuário

2. Seção de Imagens:

- Imagem original (200x200 px)

- Imagem processada (200x200 px)

- Legendas descritivas

3. Seção de Métricas:

- PSNR com indicação de alvo

- SSIM com indicação de alvo

- LC (Contraste Local)

- Edge Sharpness

- Status de validação

4. Seção de Histórico:

- Últimas 15 operações realizadas

- Timestamp de cada operação

- Usuário responsável

5. Rodapé:

- Número de página

- Identificação do sistema

Tecnologia: ReportLab

Formato: PDF/A4

Fluxo de Execução:

1. Criar buffer de memória

2. Inicializar canvas PDF

3. Salvar imagens temporariamente

4. Adicionar elementos ao PDF

5. Finalizar documento

6. Limpar arquivos temporários

7. Retornar bytes

Complexidade: O(N) onde N = elementos no relatório

"""

**5. Fluxo de Execução**

**5.1 Fluxo Principal**

┌─────────────────────┐

│ Iniciar Aplicação │

└──────────┬──────────┘

│

▼

┌─────────────────────┐

│ Inicializar Sistema │

│ (Session State) │

└──────────┬──────────┘

│

▼

┌─────────────────────┐

│ Renderizar UI │

│ (Streamlit Tabs) │

└──────────┬──────────┘

│

▼

┌─────────────────────────────────────┐

│ TAB 1: UPLOAD │

│ 1. Selecionar arquivo │

│ 2. Validar formato │

│ 3. Carregar imagem │

│ 4. Normalizar para 512x512 │

│ 5. Exibir resultado │

└──────────┬───────────────────────────┘

│

▼

┌─────────────────────────────────────┐

│ TAB 2: PROCESSAMENTO │

│ ┌────────────────────────────────┐ │

│ │ ETAPA 1: Pré-processamento │ │

│ │ - Selecionar filtro │ │

│ │ - Ajustar parâmetros │ │

│ │ - Aplicar processamento │ │

│ └────────────┬───────────────────┘ │

│ ▼ │

│ ┌────────────────────────────────┐ │

│ │ ETAPA 2: Realce de Nitidez │ │

│ │ - Selecionar método │ │

│ │ - Ajustar peso/intensidade │ │

│ │ - Aplicar processamento │ │

│ └────────────┬───────────────────┘ │

│ ▼ │

│ ┌────────────────────────────────┐ │

│ │ ETAPA 3: Realce de Contraste │ │

│ │ - Selecionar método │ │

│ │ - Ajustar clip limit │ │

│ │ - Aplicar processamento │ │

│ └────────────┬───────────────────┘ │

│ ▼ │

│ ┌────────────────────────────────┐ │

│ │ Calcular Métricas │ │

│ └────────────────────────────────┘ │

└──────────┬───────────────────────────┘

│

▼

┌─────────────────────────────────────┐

│ TAB 3: ANÁLISE VISUAL │

│ 1. Comparação lado a lado │

│ 2. Mapa de diferença │

│ 3. Detecção de bordas │

│ 4. Histogramas RGB │

└──────────┬───────────────────────────┘

│

▼

┌─────────────────────────────────────┐

│ TAB 4: MÉTRICAS │

│ 1. Exibir PSNR │

│ 2. Exibir SSIM │

│ 3. Exibir LC │

│ 4. Exibir Edge Sharpness │

│ 5. Validar contra alvos │

│ 6. Gerar gráficos │

└──────────┬───────────────────────────┘

│

▼

┌─────────────────────────────────────┐

│ TAB 5: RELATÓRIO │

│ 1. Exibir histórico │

│ 2. Gerar PDF │

│ 3. Download imagem processada │

│ 4. Opções de reset │

└──────────────────────────────────────┘

**5.2 Fluxo de Processamento de Imagem**

Imagem Original

│

▼

┌──────────────┐

│ Normalização │ ← 512x512 pixels

│ Bilinear │

└──────┬───────┘

│

▼

┌──────────────┐

│Pré-processo │ ← Gaussian/Median Filter

│ Suavização │

└──────┬───────┘

│

▼

┌──────────────┐

│ Realce │ ← Laplacian/Sobel/HighPass

│ Nitidez │

└──────┬───────┘

│

▼

┌──────────────┐

│ Realce │ ← CLAHE/Histogram Eq.

│ Contraste │

└──────┬───────┘

│

▼

Imagem Processada

│

▼

┌──────────────┐

│ Cálculo de │

│ Métricas │

└──────────────┘

**6. Tecnologias e Dependências**

**6.1 Bibliotecas Principais**

| **Biblioteca** | **Versão Mínima** | **Propósito** |
| --- | --- | --- |
| **streamlit** | 1.28.0 | Framework web e interface |
| **opencv-python** | 4.8.0 | Processamento de imagens |
| **numpy** | 1.24.0 | Operações matriciais |
| **scikit-image** | 0.21.0 | Algoritmos de processamento |
| **matplotlib** | 3.7.0 | Visualização e gráficos |
| **Pillow** | 10.0.0 | Manipulação de imagens |
| **reportlab** | 4.0.0 | Geração de PDF |
| **scipy** | 1.11.0 | Operações científicas |

**6.2 Dependências por Módulo**

**Processamento de Imagens**

python

import cv2 *# OpenCV para operações de imagem*

from skimage import filters *# Filtros avançados*

from skimage.filters import gaussian, median

from scipy import ndimage *# Operações multidimensionais*

**Métricas e Análise**

python

from skimage import metrics *# SSIM e outras métricas*

import numpy as np *# Cálculos matriciais*

**Visualização**

python

import matplotlib.pyplot as plt *# Gráficos*

from PIL import Image *# Manipulação de imagens*

**Interface**

python

import streamlit as st *# Framework web*

**Relatórios**

python

from reportlab.lib.pagesizes import A4

from reportlab.pdfgen import canvas

from reportlab.lib.units import inch

**6.3 Requisitos de Sistema**

* **Python:** 3.8 ou superior
* **RAM:** Mínimo 4GB (recomendado 8GB)
* **Processador:** Dual-core ou superior
* **Espaço em disco:** 500MB para dependências
* **Navegador:** Chrome, Firefox, Safari ou Edge (versões recentes)

**7. Diagrama de Classes**

┌─────────────────────────────────────────────────────────┐

│ ImageProcessingSystem │

├─────────────────────────────────────────────────────────┤

│ - original\_image: np.ndarray │

│ - processed\_image: np.ndarray │

│ - normalized\_image: np.ndarray │

│ - history: List[str] │

│ - metrics: Dict[str, float] │

│ - user: str │

├─────────────────────────────────────────────────────────┤

│ + \_\_init\_\_() │

│ + log\_action(action: str): void │

│ + load\_image(uploaded\_file): bool │

│ + apply\_preprocessing(filter\_type, radius, sigma): bool│

│ + apply\_sharpening(method, weight, thresh, intens): bool│

│ + apply\_contrast\_enhancement(method, clip, tile): bool │

│ + calculate\_metrics(): bool │

│ + generate\_pdf\_report(): bytes │

└─────────────────────────────────────────────────────────┘

**7.1 Relacionamentos**

┌────────────────┐

│ Streamlit │ (Interface)

│ UI │

└───────┬────────┘

│ usa

▼

┌────────────────────────┐

│ ImageProcessingSystem │ (Controlador)

└───────┬────────────────┘

│ utiliza

▼

┌──────────────────────────────────┐

│ Bibliotecas de Processamento │

│ - OpenCV (cv2) │

│ - scikit-image │

│ - NumPy │

│ - SciPy │

└──────────────────────────────────┘

│ produz

▼

┌──────────────────────────────────┐

│ Saídas do Sistema │

│ - Imagens processadas │

│ - Métricas calculadas │

│ - Relatórios PDF │

│ - Visualizações │

└──────────────────────────────────┘

**8. Casos de Uso**

**8.1 UC01: Carregar e Normalizar Imagem**

**Ator:** Operador/Administrador  
**Pré-condições:** Sistema iniciado, navegador aberto  
**Fluxo Principal:**

1. Usuário acessa aba "Upload"
2. Sistema exibe interface de upload
3. Usuário seleciona arquivo de imagem (PNG/JPEG)
4. Usuário clica em "Carregar Imagem"
5. Sistema valida formato do arquivo
6. Sistema carrega imagem em memória
7. Sistema converte BGR → RGB
8. Sistema redimensiona para 512x512 pixels
9. Sistema salva imagens em session\_state
10. Sistema registra operação no histórico
11. Sistema exibe imagem carregada

**Pós-condições:** Imagem pronta para processamento  
**Fluxos Alternativos:**

* 5a. Formato inválido → Sistema exibe erro e retorna ao passo 3
* 6a. Arquivo corrompido → Sistema exibe erro e retorna ao passo 3

**8.2 UC02: Aplicar Pré-processamento**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagem carregada no sistema  
**Fluxo Principal:**

1. Usuário acessa aba "Processamento"
2. Sistema exibe opções de pré-processamento
3. Usuário seleciona tipo de filtro (Gaussiano/Mediana)
4. Usuário ajusta raio do kernel (slider)
5. Usuário ajusta sigma (slider)
6. Usuário clica em "Aplicar Pré-processamento"
7. Sistema valida parâmetros
8. Sistema aplica filtro selecionado
9. Sistema atualiza imagem processada
10. Sistema registra operação no histórico
11. Sistema exibe comparação visual

**Pós-condições:** Imagem pré-processada armazenada  
**Fluxos Alternativos:**

* 7a. Sigma fora do intervalo [0.5, 2.0] → Sistema exibe erro
* 7b. Kernel radius par → Sistema ajusta para ímpar automaticamente

**Regras de Negócio:**

* RN01: Sigma deve estar entre 0.5 e 2.0
* RN02: Kernel radius deve ser ímpar
* RN03: Operação deve ser registrada no histórico

**8.3 UC03: Aplicar Realce de Nitidez**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagem carregada (preferencialmente pré-processada)  
**Fluxo Principal:**

1. Usuário expande seção "Realce de Nitidez"
2. Sistema exibe controles de nitidez
3. Usuário seleciona método (Laplaciano/Sobel/Alta Frequência)
4. Usuário ajusta peso do filtro
5. Usuário ajusta limiar (se Sobel)
6. Usuário ajusta intensidade (se Alta Frequência)
7. Usuário clica em "Aplicar Nitidez"
8. Sistema aplica método selecionado
9. Sistema combina resultado com imagem original
10. Sistema atualiza imagem processada
11. Sistema registra operação
12. Sistema exibe comparação visual

**Pós-condições:** Imagem com nitidez realçada  
**Fluxos Alternativos:**

* 8a. Erro no processamento → Sistema exibe mensagem de erro

**Métodos Disponíveis:**

* **Laplaciano:** Detecta bordas usando segunda derivada
* **Sobel:** Calcula gradiente direcional com threshold
* **Alta Frequência:** Unsharp masking para realce geral

**8.4 UC04: Aplicar Realce de Contraste**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagem carregada e processada  
**Fluxo Principal:**

1. Usuário expande seção "Realce de Contraste"
2. Sistema exibe controles de contraste
3. Usuário seleciona método (CLAHE/Equalização Global)
4. Usuário ajusta limite de clipagem (se CLAHE)
5. Usuário ajusta tamanho do bloco (se CLAHE)
6. Usuário clica em "Aplicar Contraste"
7. Sistema converte para espaço de cor apropriado
8. Sistema aplica método selecionado no canal de luminância
9. Sistema reconverte para RGB
10. Sistema atualiza imagem processada
11. Sistema registra operação
12. Sistema exibe comparação visual

**Pós-condições:** Imagem com contraste realçado  
**Fluxos Alternativos:**

* 4a. Clip limit fora de [2.0, 3.0] → Sistema exibe erro

**Regras de Negócio:**

* RN04: Clip limit deve estar entre 2.0 e 3.0
* RN05: Tile size deve ser 4, 8 ou 16
* RN06: Processamento em espaço de cor LAB (CLAHE) ou YCrCb (Global)

**8.5 UC05: Calcular e Visualizar Métricas**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagem original e processada disponíveis  
**Fluxo Principal:**

1. Usuário clica em "Calcular Métricas"
2. Sistema calcula PSNR
3. Sistema calcula SSIM
4. Sistema calcula LC (Local Contrast)
5. Sistema calcula Edge Sharpness
6. Sistema armazena resultados em session\_state
7. Sistema registra operação
8. Sistema exibe métricas em cards
9. Sistema valida contra alvos estabelecidos
10. Sistema exibe status de aprovação/reprovação
11. Usuário acessa aba "Métricas"
12. Sistema exibe visualizações gráficas
13. Sistema exibe histogramas comparativos

**Pós-condições:** Métricas calculadas e disponíveis  
**Métricas Calculadas:**

* **PSNR:** ≥ 30 dB (alvo)
* **SSIM:** ≥ 0.85 (alvo)
* **LC:** Valor informativo
* **Edge Sharpness:** Valor informativo

**8.6 UC06: Gerar Relatório PDF**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagem processada e métricas calculadas  
**Fluxo Principal:**

1. Usuário acessa aba "Relatório"
2. Sistema exibe histórico de operações
3. Usuário clica em "Gerar Relatório PDF"
4. Sistema cria buffer de memória
5. Sistema inicializa canvas PDF
6. Sistema adiciona cabeçalho com metadados
7. Sistema adiciona imagens original e processada
8. Sistema adiciona seção de métricas
9. Sistema adiciona histórico de operações
10. Sistema finaliza documento
11. Sistema limpa arquivos temporários
12. Sistema exibe botão de download
13. Usuário clica em "Download Relatório PDF"
14. Sistema envia arquivo para navegador
15. Sistema registra operação

**Pós-condições:** Relatório PDF gerado e baixado  
**Conteúdo do Relatório:**

* Título e metadados
* Imagens comparativas
* Métricas quantitativas
* Histórico de operações
* Timestamp de geração

**8.7 UC07: Análise Visual Comparativa**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagens original e processada disponíveis  
**Fluxo Principal:**

1. Usuário acessa aba "Análise"
2. Sistema exibe comparação lado a lado
3. Sistema calcula mapa de diferença
4. Sistema aplica detector Canny em ambas imagens
5. Sistema exibe bordas detectadas
6. Sistema calcula histogramas RGB
7. Sistema plota histogramas comparativos
8. Sistema exibe todas as visualizações
9. Usuário analisa resultados visuais

**Pós-condições:** Análise visual completa disponível  
**Visualizações Geradas:**

* Comparação lado a lado
* Mapa de diferença absoluta
* Bordas detectadas (original vs processada)
* Histogramas RGB comparativos

**8.8 UC08: Download de Imagem Processada**

**Ator:** Operador/Administrador  
**Pré-condições:** Imagem processada disponível  
**Fluxo Principal:**

1. Usuário acessa aba "Relatório"
2. Sistema exibe opções de download
3. Usuário clica em "Baixar Imagem Processada"
4. Sistema converte array NumPy para PIL Image
5. Sistema cria buffer de memória
6. Sistema salva imagem em formato PNG
7. Sistema prepara arquivo para download
8. Sistema exibe botão de download
9. Usuário clica em "Download Imagem PNG"
10. Sistema envia arquivo para navegador
11. Sistema registra operação

**Pós-condições:** Imagem processada salva localmente  
**Formato de Saída:** PNG (sem perda de qualidade)