**Curso** Gestão e Programação de Sistemas Informáticos | **Turma** | **Ano** | **Data**

**Disciplina** Programação de Sistemas de Informação | **Módulo** 18 – Ferramentas de Tratamento de Imagem

**Nome do aluno** | **Número**

**Professor** André Rolo | **Avaliação**

# Editor Web de Imagens com Flask e Pillow

## Objetivo:

Desenvolver um editor de imagens web interativo que permita ao utilizador criar, editar e manipular imagens diretamente no navegador. O backend será construído com **Flask**, enquanto a manipulação das imagens será feita utilizando a biblioteca **Pillow** em Python. A aplicação permitirá funções básicas de edição, como redimensionamento, rotação, ajustes de brilho e contraste, e a criação de imagens do zero.

## Requisitos:

1. **Funcionalidades Principais:**
    - **Carregar Imagens:** O utilizador poderá carregar imagens de seu dispositivo local através do frontend.
    - **Edição de Imagens:** O editor permitirá as seguintes funcionalidades de edição:
        - Redimensionamento de imagem.
        - Rotação da imagem.
        - Ajuste de brilho e contraste.
        - Aplicação de filtros básicos (como preto e branco, sepia, etc.).
        - Adição de texto e formas geométricas simples (como linhas e círculos).
    - **Criação de Imagens:** O utilizador poderá criar imagens a partir do zero, utilizando ferramentas de desenho (ex.: desenhar formas e escrever textos).
    - **Visualização de Resultados:** O sistema deve permitir a visualização em tempo real das alterações feitas na imagem antes de salvar.

- **Salvar Imagens:** O utilizador poderá baixar as imagens editadas ou criadas diretamente no seu dispositivo.

## Tecnologias:

- **Backend:** Python com o framework **Flask** para criar a aplicação web.
- **Manipulação de Imagens: Pillow**, uma biblioteca Python para processamento e manipulação de imagens.
- **Frontend:** HTML, CSS, JavaScript para a interface de utilizador. O Flask irá fornecer rotas para servir a página web e interagir com o backend.
- **Armazenamento Temporário:** O Flask irá gerenciar o armazenamento temporário das imagens para permitir a edição e salvar no dispositivo do utilizador.

## Documentação:

- **Manual do Utilizador:** Documento explicativo sobre como usar a aplicação, com instruções detalhadas sobre as funcionalidades e ferramentas disponíveis no editor de imagens.
- **Manual Técnico:** Descrição do funcionamento do sistema, arquitetura da aplicação, estrutura do código e como as tecnologias são integradas.
- **Relatório do Projeto:** Relatório final detalhado, abordando as etapas do desenvolvimento do projeto, principais desafios e soluções adotadas.

## Código e Versionamento:

- **GitHub:** O código deve ser hospedado no GitHub, com commits claros e frequentes. O repositório deve incluir a documentação necessária.
- **JavaDoc:** O código Python deverá ser bem documentado, explicando a lógica por trás de funções e módulos. Utilizar comentários e docstrings para garantir clareza no código.

## Avaliação:

- **Funcionalidade do Código:** O código deve implementar corretamente todas as funcionalidades propostas no projeto, sem falhas, e deve ser bem estruturado.
- **Organização das Pastas:** A estrutura do projeto deve ser organizada de forma lógica, separando claramente os arquivos do backend (Flask), templates HTML, arquivos estáticos (CSS, JS), e imagens.
- **Documentação:** A qualidade da documentação (manual do utilizadores, manual técnico e relatório do projeto) será avaliada.

- **Commits no GitHub:** O projeto deve ser mantido no GitHub, com commits frequentes e bem documentados, registando o progresso do desenvolvimento.
- **Modelos de Baixa e Média Fidelidade:** Serão exigidos modelos de interface de usuário (UI) de baixa e média fidelidade, mostrando a evolução do design da aplicação, tanto em papel quanto em ferramentas de prototipagem (como Figma ou Adobe XD).

## Critérios de Avaliação:

- **JavaDoc (Docstrings no Python):** A clareza e a qualidade da documentação do código Python, explicando o funcionamento de cada função e módulo.
- **Commits no GitHub:** A frequência dos commits e a clareza das mensagens de commit, organizando o código e o progresso do projeto de maneira eficiente.
- **Funcionalidade do Código:** O código implementa as funcionalidades de maneira eficiente e sem falhas. Deve ser fácil de entender e testar.
- **Organização das Pastas:** A estrutura das pastas deve ser lógica e modularizada, separando claramente o código do backend, frontend e arquivos estáticos.
- **Manuais e Relatório:** A clareza e profundidade da documentação fornecida, que deve ser compreensível tanto para o utilizadores quanto para outros desenvolvedores.
- **Modelos de UI:** A criação de protótipos de interface de utilizadores que demonstrem a evolução do design da aplicação.

## Links e Entregáveis:

- **Repositório no GitHub:** Link para o repositório com o código-fonte do projeto.
- **Manual do Utilizador:** Documento que explica como usar a aplicação, com detalhes sobre cada ferramenta de edição.
- **Manual Técnico:** Documento técnico com a descrição da arquitetura, tecnologias utilizadas e estrutura do código.
- **Relatório do Projeto:** Relatório final com uma visão geral do desenvolvimento, desafios encontrados e soluções implementadas.

Bom trabalho!

Prof. André Rolo