



Escola Profissional
BENTO DE JESUS CARAÇA
DELEGAÇÃO DO BARREIRO

Hetal Verma, Lúri Novas, Pedro Moretti

Curso Técnico de Gestão e Programação de Sistemas Informáticos

Escola Profissional Bento de Jesus Caraça, Delegação do Barreiro

P.S.I.: Programação e Sistemas de Informação

Professor André Rolo

19 de março de 2025

Índice

Objetivos do Projeto.....	6
Descrição do Projeto.....	7
Estrutura do projeto	7
→ Diretórios e Ficheiros Principais	7
→ Dependências e Tecnologias Utilizadas	7
Backend - Flask e Funções Principais	8
→ Inicialização da Aplicação	8
→ Caminho Principal(/).....	8
→ Upload de Imagem (/upload)	8
→ Aplicar Filtro à Imagem (/apply_filter).....	9
Frontend - JS e HTML	10
→ Criar e Exibir Imagem (<i>createImage.js</i>)	10
→ Upload e Processamento de Imagens (<i>handleUpload.js</i>).....	10
Funcionalidades.....	11
Conclusão.....	12

Índice de Ilustrações

Figura 1	8
Figura 2	8
Figura 3	8
Figura 4	9
Figura 5	10
Figura 6	10

Índice Acrónimo

- **HTML** – HyperText Markup Language
- **CSS** – Cascading Style Sheets
- **JS** – JavaScript
- **Flask** – Microframework Python para aplicações web
- **PIL** (Pillow) – Biblioteca Python para manipulação de imagens
- **API** – Application Programming Interface

Introdução

Neste projeto, foi desenvolvido um editor de imagens interativo para a *Web*, utilizando a estrutura Flask para o *backend* e a biblioteca Pillow para a manipulação de imagens. A aplicação permite aos utilizadores carregar, editar e criar imagens diretamente no navegador, oferecendo funcionalidades como redimensionar, rodar, ajustar o brilho e o contraste, aplicar filtros básicos e adicionar texto. Também permite criar imagens de raiz utilizando ferramentas de desenho.

A estrutura do projeto foi organizada de forma modular, separando os componentes de *backend*, *frontend* e armazenamento temporário de imagens. O *backend*, desenvolvido em *Python* com Flask, gere as operações de edição e a comunicação com o *frontend*, que foi construído usando HTML, CSS e JS para fornecer uma interface intuitiva. O código foi colocado no *GitHub*, garantindo um registo estruturado do desenvolvimento e facilitando a colaboração.

Objetivos do Projeto

O objetivo do projeto era desenvolver uma aplicação *web* que pudesse tornar a edição de imagens mais acessível e intuitiva para qualquer utilizador. A ideia era criar uma ferramenta leve e funcional que corresse diretamente no *browser*, permitindo edições rápidas sem a necessidade de *software* específico.

Outro ponto importante foi a experiência prática com o Flask e o Pillow, explorando como essas tecnologias podem ser usadas para manipular imagens de forma eficiente. Além disso, o projeto também serviu para reforçar conceitos de organização de código, integração entre *frontend* e *backend*, e boas práticas de versionamento no *GitHub*.

Descrição do Projeto

A aplicação desenvolvida pretende oferecer uma plataforma web interativa onde os utilizadores podem ter acesso a várias funcionalidades através de uma interface fácil de utilizar. O sistema foi projetado para ser eficiente, intuitivo e fácil de manter.

Estrutura do projeto

A estrutura do projeto está organizada da seguinte forma:

→ Diretórios e Ficheiros Principais

- ***app.py***: Ficheiro principal do *backend*.
 - ***templates***: Contém o ficheiro *index.html*, que define a interface.
 - ***static/css/styles.css***: Ficheiro CSS para a estilização.
 - ***static/js/createImage.js* e *static/js/handleUpload.js***: Ficheiros JS para carregar e editar imagens.
 - ***static/uploads/***: Pasta onde são guardadas as imagens enviadas pelos utilizadores.
-

→ Dependências e Tecnologias Utilizadas

- **Flask**: Framework para gerir o *backend*.
 - **PIL (Pillow)**: Biblioteca *Python* para editar imagens.
 - **HTML, CSS e JS**: Linguagens usadas para a interface.
-

Backend - Flask e Funções Principais

O *app.py* é o ficheiro que gere o *backend* e tem as seguintes funcionalidades:

→ Inicialização da Aplicação

```
app = Flask(__name__)  
UPLOAD_FOLDER = 'static/uploads'  
os.makedirs(UPLOAD_FOLDER, exist_ok=True)
```

Figura 1

Define o Flask como base da aplicação e cria a pasta para guardar imagens.

→ Caminho Principal(/)

```
@app.route('/')  
def index():  
    return render_template('index.html')
```

Figura 2

Define a página inicial.

→ Upload de Imagem (/upload)

```
@app.route('/upload', methods=['POST'])  
def upload_image():  
    if 'file' not in request.files:  
        return jsonify({'error': 'Nenhum ficheiro enviado.'  
    file = request.files['file']  
    if file.filename == '':  
        return jsonify({'error': 'Nenhum ficheiro selecionado.'  
    filepath = os.path.join(UPLOAD_FOLDER, file.filename)  
    file.save(filepath)  
    return jsonify({'filepath': filepath})
```

Figura 3

- Permite carregar imagens.
- Guarda as imagens na pasta *static/uploads*.

→ **Aplicar Filtro à Imagem (/apply_filter)**

```
@app.route('/apply_filter', methods=['POST'])
def apply_filter():
    data = request.json
    filepath = data.get('filepath')
    filter_type = data.get('filter')
    if not filepath or not os.path.exists(filepath):
        return jsonify({'error': 'Ficheiro não encontrado.'})
```

Figura 4

Recebe um pedido *POST* com a imagem e o filtro a aplicar.

Frontend - JS e HTML

O *frontend* interage com o *backend* através de JS e tem as seguintes funcionalidades:

→ Criar e Exibir Imagem (*createImage.js*)

```
document.getElementById("create-image-button").addEventListener(
  const canvas = document.getElementById("canvas");
  const ctx = canvas.getContext("2d");
  canvas.width = 800;
  canvas.height = 600;
  ctx.fillStyle = "white";
  ctx.fillRect(0, 0, 800, 600);
  document.getElementById("editor-menu").classList.remove(
  });
```

Figura 5

Cria um *canvas* onde o utilizador pode editar a imagem.

→ Upload e Processamento de Imagens (*handleUpload.js*)

```
document.addEventListener('DOMContentLoaded', function ()
  document.getElementById('file-input').addEventListener(
    const file = event.target.files[0];
    if (file) {
      const reader = new FileReader();
      reader.onload = function(e) {
        const image = new Image();
        image.onload = function() {
          ctx.drawImage(image, 0, 0, 400, image.
        };
        image.src = e.target.result;
      };
      reader.readAsDataURL(file);
    }
  });
});
```

Figura 6

Carrega e exibe a imagem do utilizador.

Funcionalidades

- Implementação de novos filtros.
 - Criação de uma base de dados para guardar imagens editadas.
 - Melhor otimização da interface para ser mais intuitiva.
-

Conclusão

Este projeto desenvolveu uma aplicação *web* que permite carregar, editar e aplicar filtros a imagens de forma simples e acessível. A estrutura do código baseia-se em Flask para o *backend* e JS para o *frontend*, garantindo um funcionamento dinâmico e eficiente.

Através da organização em diretórios e da utilização de tecnologias modernas, a aplicação permite ao utilizador carregar uma imagem, visualizar as alterações em tempo real e descarregar a versão final editada. Além disso, a implementação de diferentes filtros proporciona uma experiência mais interativa e útil.

Este projeto pode ser melhorado no futuro com novas funcionalidades, como integração com bases de dados para armazenar imagens, mais opções de edição e otimização da interface para facilitar ainda mais a utilização. A aplicação demonstra a importância da combinação entre *backend* e *frontend* para criar soluções eficientes e práticas para o utilizador final.