

Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

Ano Letivo 2022/2023

**Produção de Conteúdos Multimédia**

**Relatório Google Imagens**

**Turma 32D**

Tomás de Matos – A49711

Guilherme Gouveia– A49740

Docente: Pedro Fazenda

Janeiro 2023

Índice

[**Introdução** 3](#_Toc124432366)

[**Objetivos** 3](#_Toc124432367)

[**Processo de desenvolvimento da aplicação** 3](#_Toc124432368)

[**ISearchEngine** 3](#_Toc124432369)

[**Image\_Processing** 12](#_Toc124432370)

[**ISearchEngine\_Manager** 15](#_Toc124432371)

# **Introdução**

Este projeto tem como objetivo o desenvolvimento de uma aplicação multimédia em HTML5, incluindo API’s em JavaScript, para pesquisa e visualização de fotos digitais de uma coleção de fotografias. A aplicação deverá incluir vários tipos de informação (conteúdos) multimédia e deverá também permitir a visualização (disposição no ecrã) dos resultados da pesquisa de diversas formas. O desenvolvimento da aplicação deverá ser centrado no utilizador desde o início do projeto até ao protótipo final.

# **Objetivos**

Os alunos têm de desenvolver uma versão simplificada do Google Imagens composta por uma interface utilizador em HTML5 para permitir que o utilizador faça as pesquisas e visualize os resultados das pesquisas.

2. Processo de desenvolvimento da aplicação

Começámos o trabalho introduzindo o código fornecido no moodle, com métodos em javascript, e com um XML “database” com as imagens que temos de utilizar no trabalho.

O arquivo XML "Image\_database.xml" é usado para guardar os “paths” das imagens no localStorage. No localStorage, todas as imagens estão classificadas por categorias e cores.

2.1. Presentation Layer

2.1.1. Back-end

Neste trabalho o utilizador consegue fazer dois tipos de pesquisa, uma pesquisa onde o utilizador pode escolher uma das categorias apresentadas e iram aparecer 30 imagens correspondentes a essa categoria. Na outra pesquisa, o utilizador pode escolher a cor e serão apresentadas as 30 imagens que têm a cor escolhida em maior quantidade da categoria correspondente.

# **Processo de desenvolvimento da aplicação**

## **ISearchEngine**

Na implementação começámos por realizar a continuação do método databaseProcessing, este método é responsável por processar as imagens que estão na base de dados do sistema. Começamos por criar dois objetos, um chamado "ColorHistogram" e outro chamado "ColorMoments", que são utilizados para calcular as características das imagens. Em seguida, a função carrega um arquivo XML que contém informações sobre as imagens presentes na base de dados.

Em seguida, a função usa um ciclo "for" para percorrer as categorias de imagens existentes. Dentro desse ciclo, outro ciclo "for" é usado para percorrer as imagens de cada categoria. Para cada imagem, é criado um caminho (path) para acessá-la (usando a categoria e o número da imagem) e é criado um objeto "Picture" para representar a imagem.

A função também cria um evento chamado "eventname", que é usado para indicar quando a imagem foi processada. Um listener é adicionado para esse evento, que chama a função "imageProcessed" quando o evento é disparado. Finalmente, a função chama o método "computation" no objeto "Picture", passando os objetos "cnv", "h12color" e "colorMoments" como parâmetros, além do evento "eventP". Esse método é responsável por processar a imagem e disparar o evento "eventname" quando terminar.



De seguida implementámos o método createXMLColordatabaseLS, que cria um XML database de imagens classificadas por cor e, em seguida, guarda-o no localStorage.

O método declara primeiro três variáveis: cor, que é atribuído o valor de this.colors, categoria, que é atribuído o valor de this.categories, e xmlString, que é uma variável que será usada para construir a string XML.

Em seguida, criámos quatro ciclos for. O primeiro for é executado durante 14 (número de categorias) vezes e o segundo for é executado durante 100 vezes (número de imagens por categoria) vezes. Este for verifica se a categoria atual é igual à categoria da imagem atual e, se for, adiciona a imagem a um array chamado arrayPaths.

O terceiro for é executado durante 12 (número de cores) vezes, é usado para classificar o array de imagens arrayPaths por cor e, em seguida, atribuir uma classe a estas imagens na string XML.

O último for é executado 25 (número de imagens) vezes, e é usado para adicionar o caminho de cada imagem na string XML.

Finalmente, o método chama o método saveLS\_XML do objeto LS\_db, passando como argumentos a categoria atual e a string XML. Isso guarda a string XML no localStorage como um arquivo XML para cada categoria.

Uma imagem com texto

Descrição gerada automaticamente

De seguida implementámos o método searchcolor(), que é usado para procurar imagens de uma determinada categoria e cor no XML database e guardado no localStorage.

O método recebe dois parâmetros, category e color. Este começa por ler o arquivo XML guardado no localStorage usando o método readLS\_XML do objeto LS\_db, passando a categoria como argumento. Usamos a propriedade getElementsByClassName para acessar os elementos XML com a cor especificada.

Em seguida, o método esvazia o conjunto de imagens atuais usando o método empty\_Pool() e muda a cor de fundo usando a função changebackgroundColor(color).

É criado um ciclo “for” que percorre as imagens com a cor que queremos, e é usado para criar novas instâncias da classe Picture, passando os valores necessários como parâmetros e adicioná-las ao conjunto de imagens atuais.

O método changebackgroundColor muda a cor de fundo consoante a cor escolhida para que assim o utilizador possa ter uma noção melhor de como a pesquisa está a ser feita.

Uma imagem com texto

Descrição gerada automaticamente

De seguida criámos o método searchKeywords que é usado para procurar imagens de uma determinada categoria no XML database guardado no localStorage, usando a tag "path" como base.

O método recebe um parametro, category. Ele começa por ler o arquivo XML guardado no localStorage usando o método readLS\_XML do objeto LS\_db, passando a categoria como argumento. Usamos a propriedade getElementsByTagName para acessar os elementos XML com o nome da tag "path" especificada.

Em seguida, o método limpa o conjunto de imagens atuais usando o método empty\_Pool().

É criado um ciclo "for" que roda pathKeywordsElem.length vezes, é usado para verificar se a imagem já está presente no conjunto de imagens atuais, caso contrário, é criado uma nova instância da classe Picture, passando os valores necessários como parâmetros e adicionando-a ao conjunto de imagens atuais.

Esse método é semelhante ao método anterior searchColor mas é usado para buscar imagens através das informações contidas nas tags e não pela classe.

Uma imagem com texto

Descrição gerada automaticamente

De seguida realizámos o método gridView que é usado para desenhar imagens em um estilo de grid no canvas. Ele recebe como parâmetro o canvas onde as imagens serão desenhadas. Ele inicializa duas variáveis, image e counter e então passa por dois ciclos "for" para gerar a coordenadas X e Y para cada célula da grid. Dentro do laço, é verificado se o contador é menor que this.numshownpic (número de imagens que devem ser exibidas). Se sim, a próxima imagem no conjunto de imagens atuais é atribuída à variável image e desenhada no canvas na posição específica da grid. O contador é então incrementado para garantir que a próxima imagem do conjunto de imagens atuais seja desenhada na próxima posição da grid.

Uma imagem com texto

Descrição gerada automaticamente

o da aplicação

## **Image\_Processing**

O método count\_Pixels é usado para contar o número de pixels de uma imagem que correspondem a cada uma das 12 cores pré-definidas. Ele recebe como parâmetro pixels, que é um objeto que contém os dados de cor de cada pixel da imagem.

O método começa criando um array chamado histogram com tamanho 12 e preenchendo-o com zeros. Ele define duas variáveis de limite, lim1 e lim2. Ele também define três variáveis, cJr, cJg e cJb, que contêm as cores vermelhas, verdes e azuis, respectivamente, das 12 cores pré-definidas.

Em seguida, ele entra em ciclo "for" para percorrer cada pixel da imagem. Dentro do ciclo "for", ele extrai os valores da cor vermelho, verde e azul do pixel atual e compara esses valores com as cores pré-definidas. Se os valores estiverem dentro dos limites especificados, o contador correspondente para a cor pré-definida é incrementado.

Finalmente, o método retorna uma cópia do array histograma.

Uma imagem com texto

Descrição gerada automaticamente

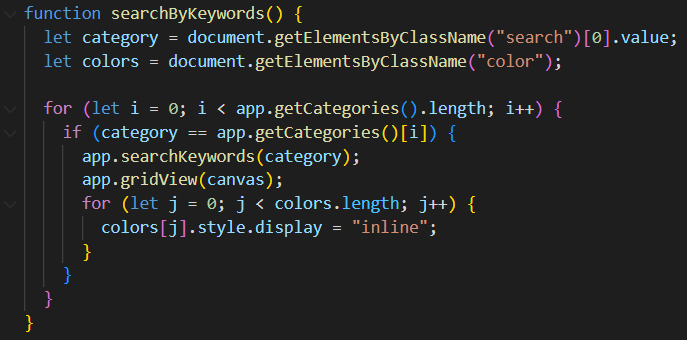
## **ISearchEngine\_Manager**

A função searchByKeywords é usada para pesquisar imagens em uma categoria específica com base em palavras-chave.

A função começa por pegar o valor do elemento de entrada com a classe "search", que representa a categoria das imagens a serem pesquisadas.

Em seguida, obtemos todos os elementos com a classe "color".

Então percorremos todas as categorias que o objeto app tem, se a categoria inserida é a mesma que a da lista chamamos o método searchKeywords enviando a categoria como argumento, e o metodo gridView.



A função searchByColor é usada para pesquisar imagens por cor. A função pega o ID do botão clicado, que representa a cor desejada, e a categoria indicada. Em seguida, ela verifica se a categoria fornecida existe na lista de categorias disponíveis. Se existir, a função chama o método "searchColor" passando a categoria e a cor desejada.



A função searchByKeywordsButton procura imagens por palavras-chave. Ela pega o valor da categoria selecionada no elemento de procura na página HTML, e verifica se a categoria é uma das categorias disponíveis. Se for, ele chama o método searchKeywords()para procurar imagens nessa categoria com base nas palavras-chave e, em seguida, chama o método gridView() para exibir as imagens encontradas.

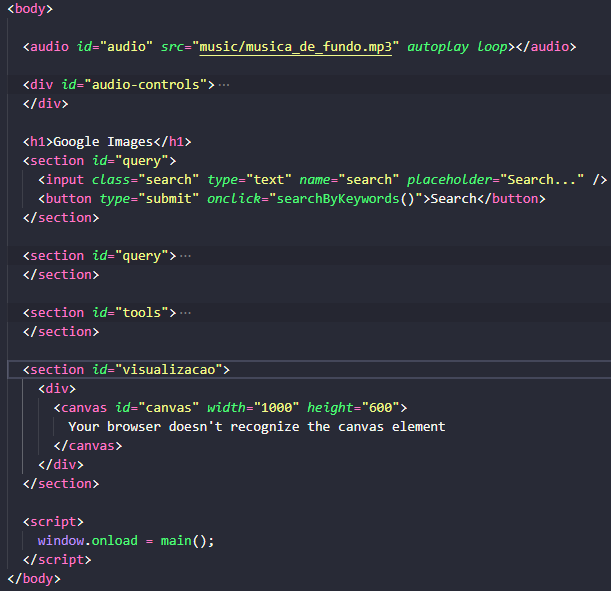
Uma imagem com texto

Descrição gerada automaticamente

## **HTML**

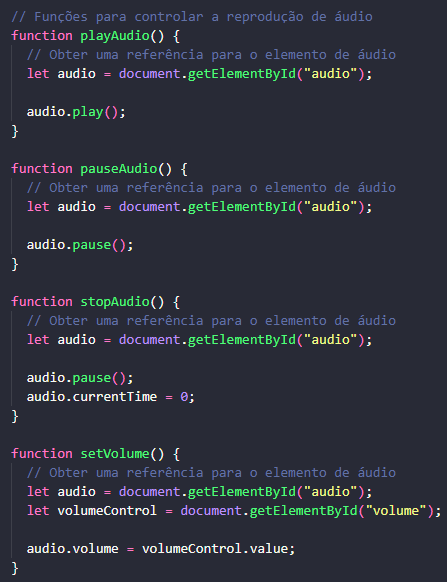
No html para além do canvas, colocámos duas sections, uma onde estão os botões para a pesquisa por categorias, que ao ser clicado chama a função searchByKeywordsButton mandando o botão como argumento para fazer a pesquisa de acordo com a categoria do botão. A outra section serve para a pesquisa por cores, que ao ser clicado chama a função searchByColor mandando o botão como argumento para fazer a pesquisa de acordo com a cor do botão.

No topo metemos uma barra de pesquisas para procurar imagens por categorias. Tem ainda três botões e um slider, para controlar a música de fundo.

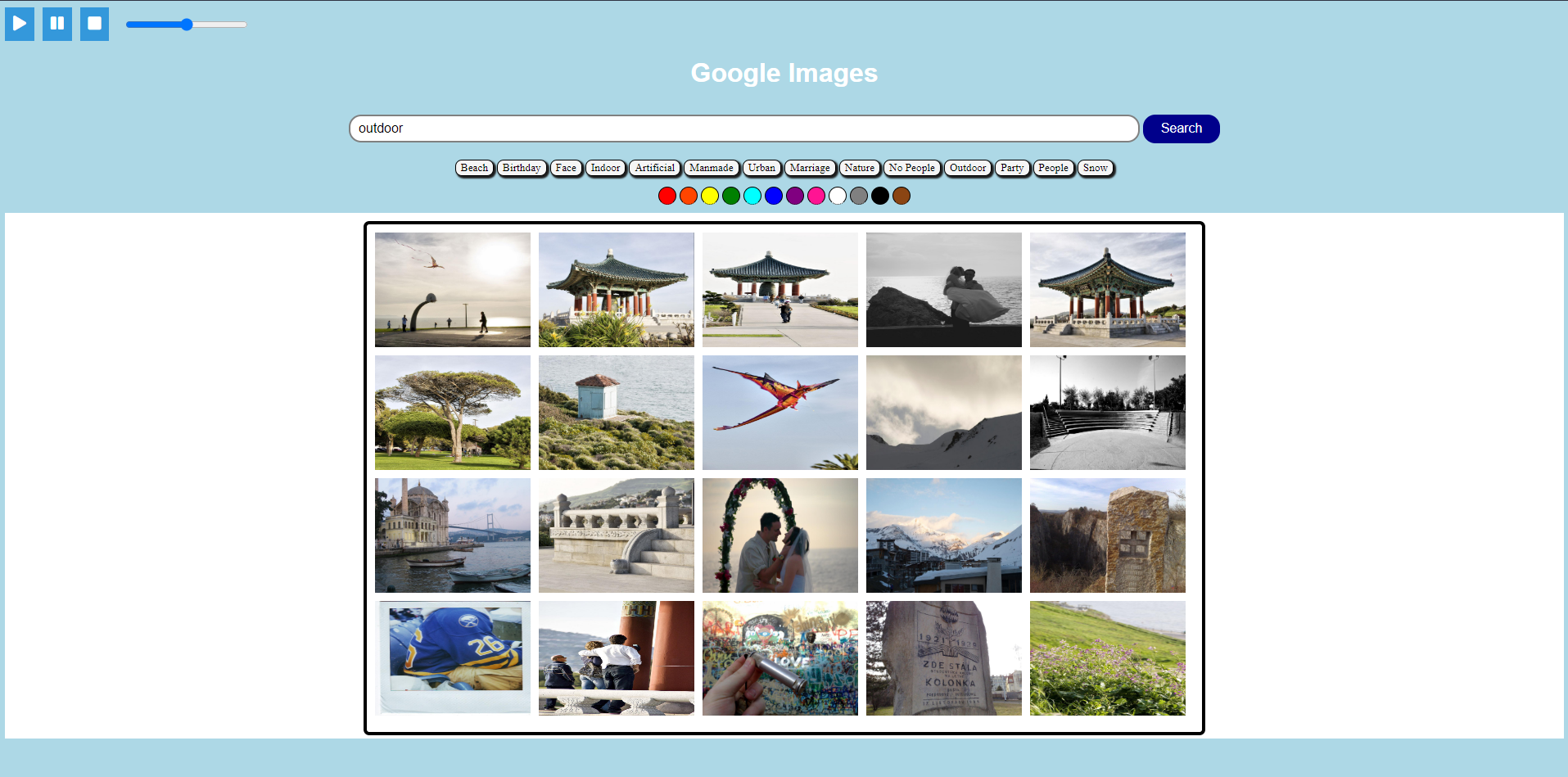


Para a realização do áudio implementámos 4 funções.

A primeira chamada playAudio, que dá play no áudio, a segunda chamda pauseAudio, que pausa a música, a terceira chamada stopAudio, que pausa a música e volta ao inicio e só começa a tocar quando clicarmos no play, e por fim a função setVolume, que nos permite controlar o volume da música.



# **Descrição e interface gráfica da aplicação**



2.1. Presentation Layer

2.1.1. Back-e