**INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA**

**Relatório do 2º Projecto Multimédia**

Produção de Conteúdos Multimédia

**Professor:** João Beleza

2015/2016

Bruno Tavares - 40644

André Fonseca - 39758

# INDÍCE

[INDÍCE 1](#_Toc443602003)

[1 INTRODUÇÃO 2](#_Toc443602004)

[2 ANÁLISE 3](#_Toc443602005)

[2.1 Análise de utilizadores 3](#_Toc443602006)

[2.2 Análise de tarefas 3](#_Toc443602007)

[3 DESIGN 4](#_Toc443602008)

[4 PROTOTIPAGEM 4](#_Toc443602009)

[4.1 Processamento da base de dados 4](#_Toc443602010)

[4.1.1 Cor dominante 5](#_Toc443602011)

[4.1.2 Imagem semelhante 5](#_Toc443602012)

[4.2 Pesquisas 7](#_Toc443602013)

[4.2.1 Pesquisa por palavra chave 7](#_Toc443602014)

[4.2.2 Pesquisa por cor dominante 7](#_Toc443602015)

[4.2.3 Pesquisa por imagem semelhante 8](#_Toc443602016)

[5 ANÁLISE DOS RESULTADOS 9](#_Toc443602017)

# 1 INTRODUÇÃO

O objectivo deste trabalho foi desenvolver uma aplicação multimédia em HTML5 e *JavaScript* para pesquisa e visualização de uma colecção de fotografias.

O desenvolvimento da aplicação foi feito centrado no utilizador de modo a proporcionar uma boa experiência de utilização ao utilizador.

2 ANÁLISE

A fase de análise consiste em identificar os requisitos e tarefas necessários da aplicação assim como o público-alvo, de modo a desenvolver um produto que seja adequado.

## 2.1 Análise de utilizadores

Uma vez que o desenvolvimento da aplicação deve ser centrado no utilizador, é necessário estudar o tipo de utilizador que a vai usar.

Neste decidimos que a aplicação deve ser abrangente a todo o tipo de utilizadores, logo deve ser acessível e fácil de usar por toda a gente.

## 2.2 Análise de tarefas

Após a leitura do enunciado identificamos os seguintes requisitos da aplicação:

* Pesquisa de imagens por palavras-chave
* Pesquisa de imagens por cor dominante
* Pesquisa de imagens por imagem semelhante

A aplicação que dada como modelo foi a Google Images. Realizamos as tarefas identificadas em cima e observamos o seu funcionamento.

# 3 DESIGN

Uma vez o público alvo da aplicação é muito vasto, optámos por um design simples e leve sem descurar a componente funcional.

Apenas és disponibilizada ao utilizador a informação e controlos necessários ao funcionamento da aplicação.

Separámos cada um dos tipos de pesquisa em páginas diferentes, assim, para cada tipo de pesquisa o utilizador sabe perfeitamente o que está a fazer e como fazer, uma vez que todos os inputs do utilizador se encontram devidamente identificados.

O menu de navegação permite alternar entre os vários tipos de pesquisa disponíveis.

# 4 **PROTOTIPAGEM**

## 4.1 Processamento da base de dados

A base de dados fornecida pelo docente em XML contempla várias informações relativamente a cada uma das imagens.

Para implementar as funcionalidades de pesquisa por cor dominante e imagem semelhante foi necessário efectuar um processamento prévio de todas as imagens da base de dados.

O resultado desse processamento foi guardado na *LocalStorage*.

Uma vez que o processamento de todas as imagens da base de dados provocava excepções do tipo *OUT\_OF\_MEMORY* no browser, adaptamos a base de dados fornecida de modo a utilizar apenas 30 imagens de cada categoria.

O processamento pode ser efectuado através da página “processDB.html” e o código necessário encontra-se nos ficheiros “processDB.js”, “processHistograma.js” e “processMomentos.js”.

### 4.1.1 Cor dominante

Para implementar a funcionalidade de pesquisa por cor dominante recorreu-se ao uso do histograma de 12 cores.

O histograma de 12 cores consiste num array de 12 elementos em que cada posição contém um inteiro que corresponde ao número de ocorrências dessa cor na imagem.

Para produzir o histograma de 12 cores é necessário percorrer todos os pixéis da imagem e verificar se ocorre alguma das 12 cores escolhidas. Em caso afirmativo, incrementa-se o valor na posição correspondente a essa cor.

A abordagem utilizada para o processamento e armazenamento da informação dos histogramas de 12 cores foi a seguinte:

* Todas as imagens da base de dados foram carregadas num *array*;
* Para cada uma das imagens desse *array* foi produzido o histograma de 12 cores
* Escolheram-se as 25 imagens com mais pixéis de cada uma das 12 cores escolhidas (*array* ordenado pelo valor da posição correspondente a cada uma das cores do histograma);
* Guardou-se na *LocalStorage* com a chave “hist12XML” uma string XML cuja estrutura se segue:

<images><image class=”cor”><path>…</path>…</image></images>

Assim temos armazenado na LocalStorage o caminho de cada uma das 25 imagens com mais pixéis para cada uma das 12 cores escolhidas.

### 4.1.2 Imagem semelhante

Para implementar a funcionalidade de pesquisa por cor imagem semelhante recorreu-se ao uso do descritor “momentos de cor no espaço HSV”.

Este descritor consiste num vector de 54 elementos (9 blocos \* 3 componentes de cor \* 2 medidas estatísticas.

O vector descritor foi obtido da seguinte maneira:

* Tas imagens da base de dados foram carregadas num *array*;
* Cada imagem do array foi dividida em 9 blocos de pixéis;
* Para cada bloco calculou-se a média e variância de cada componente de cor.

Estrutura do vector descritor: [med\_h,var\_h,med\_s,var\_s,med\_v,var\_v,…]

A abordagem utilizada para organizar as imagens de acordo com a sua semelhança foi a seguinte:

* A cada imagem foi atribuído um *array* com todas as imagens da base de dados (excepto ela própria);
* Ordenou-se o *array* de acordo a distância de Manhattan entre o vector descritor de cada imagem, e o vector descritor das restantes imagens da base de dados;
* Para cada imagem da base de dados, foi guardada na *LocalStorage* uma chave com caminho da imagem e no valor o caminho das 20 imagens mais semelhantes, de acordo com a seguinte estrutura:

<images><image class=”Manhattan”><path>…</path>…</image></images>

## 4.2 Pesquisas

### 4.2.1 Pesquisa por palavra chave

A pesquisa por palavra chave consiste na interpretação de cada uma das *tags* “<image>” da base de dados XML, nomeadamente dos nós:

* *ID;*
* *path;*
* *title;*

Caso a palavra chave seja parte de um destes valores, a imagem é adicionada a um *array* para que posteriormente seja apresentada no *canvas*.

### 4.2.2 Pesquisa por cor dominante

Cada controlo da selecção de cor tem um *id* correspondente à cor que representa.

Uma vez que foi previamente efectuado o processamento da base de dados, no valor da chave “hist12XML” da *LocalStorage* encontramos para cada uma das cores dos controlos, as imagens com mais pixéis dessa mesma cor.

Assim basta procurar na *string* XML o nó “<image>” com classe igual ao *id* do controlo seleccionado, e para todos os nós “<path>” no seu interior, guardar o valor de cada um (corresponde ao caminho da imagem).

A partir do seu caminho, a imagem é adicionada a um *array* para posteriormente ser apresentada no *canvas*.

### 4.2.3 Pesquisa por imagem semelhante

A pesquisa por imagem semelhante funciona de modo interno à base de dados, ou seja, só é possível procurar imagens semelhantes a uma das imagens da base de dados.

A imagem alvo é identificada pelo seu caminho.

Uma vez que foi previamente efectuado o processamento da base de dados, na *LocalStorage* podemos encontrar o caminho das imagens semelhantes à imagem alvo, armazenados no valor da chave com o caminho da imagem alvo.

Caso a chave exista, retira-se o valor de todos os nós do tipo “<path>”, que corresponde ao caminho das imagens semelhantes.

Depois através do caminho, as imagens são adicionadas a um *array* para mais tarde serem apresentadas no *canvas*.

# 5 ANÁLISE DOS RESULTADOS

Após desenvolvido o protótipo da aplicação realizamos alguns testes para verificar se esta cumpria os princípios de usabilidade e as tarefas necessários.

Na pasta “Teste” encontram-se imagens que foram usadas para testar a aplicação.

Também foram realizados alguns questionários de usabilidade, utilizando o questionário realizado anteriormente na disciplina.

Após filtrada a informação considerada pertinente para o desenvolvimento da aplicação obtemos os seguintes resultados (classificados de 1 a 5):

Para além destas, também as perguntas de texto relativamente às dificuldades encontradas ao realizar as tarefas foram as que tivemos em conta para o desenvolvimento.

Um dos utilizadores apontou que durante a utilização da pesquisa por cor dominante, não tinha maneira de saber qual a cor estava a ser pesquisada.

Assim fizemos as seguintes alterações à aplicação:

* Durante o evento *mouseover* nos controlos de cor alterar a borda do elemento de modo a saber onde está o cursor;
* Após seleccionar uma cor para pesquisa, a borda do elemento de controlo mantém-se alterada;