

# PLATAFORMA DE GESTÃO E NAVEGAÇÃO EM CONTEÚDOS VISUAIS ENRIQUECIDOS COM METADADOS

Diogo Gonalo Lima de Freitas



Departamento de Engenharia Eletrot cnica

Instituto Superior de Engenharia do Porto

2021



Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Projeto/Estágio, do 3º ano, da Licenciatura em Engenharia Eletrotécnica e de Computadores

Candidato: Diogo Gonçalo Lima de Freitas, Nº 1180919, 1180919@isep.ipp.pt

Orientador: Pedro Miguel Soares, pms@isep.ipp.pt



Departamento de Engenharia Eletrotécnica

Instituto Superior de Engenharia do Porto

6 de setembro de 2021



## *Agradecimentos*

Quero agradecer ao Engenheiro Pedro Carvalho pelo acompanhamento realizado ao longo do semestre e do projeto, auxiliando em todos os parâmetros necessários ao projeto.

Quero agradecer também a todos os meus colegas de curso e fora do curso pela ajuda prestada ao longo do projeto.



## *Resumo*

Os ficheiros presentes na internet guardam cada vez mais informações sobre as suas características, como o criador, hora e data de criação, entre outros dados relevantes. Estes dados são chamados de metadados, e estão presentes em todos os tipos de documentos, guardando sempre informações sobre o documento. Um exemplo de documento onde estes metadados estão presentes são as imagens tiradas e publicados pelas pessoas, que guardam informações de quem, onde, quando, com quê, que estas foram criadas. Para este projeto, é utilizado este tipo de documento para demonstrar um exemplo de utilização de metadados.

Este projeto tem como objetivo desenvolver uma plataforma de gestão e navegação de conteúdos visuais, utilizando os metadados para interligar cada imagem adicionada no sistema. A plataforma permite ao utilizador que submeta as suas imagens e partilhar com os outros utilizadores e que este possa visualizar e interagir com as imagens partilhados por outros utilizadores. De modo a obter os dados das imagens, todas as imagens submetidas são analisadas, sendo os dados extraídos armazenados e organizados numa base de dados, para que seja mais fácil a procura e comparação dos dados entre as várias imagens.

Esta solução apresentou-se válida para o problema abordado, como é demonstrado pelos resultados dos testes realizados. De uma forma futura, poderia haver algumas correções/ adições a serem feitas, tais como, adicionar mais dados requisitados e a respetiva remodelação no código e na base de dados, aprimorar as funcionalidades já presentes na interface e/ou adicionar mais funcionalidades, de forma a facilitar a navegação do utilizador pela plataforma.

## *Palavras-Chave*

Metadados, Imagem, Exif, Base de dados, Python, Interface Gráfica, *Flask*.





## *Abstract*

The files present on the internet save more and more information about their characteristics, such as the creator, time and date of creation, among other relevant data. These data are called metadata, and are present in all types of documents, always keeping information about the document. An example of a document where this metadata is present are the images taken and published by people, which store information of who, where, when, with what, that these were created. For this project, this type of document is used to demonstrate an example of metadata usage.

This project aims to develop a platform for managing and browsing visual content, using metadata to link each image added to the system. The platform allows the user to submit his images and share with other users and allows the user to view and interact with the images shared by other users. In order to obtain the data from the images, all the images submitted are analysed, and the extracted data is stored and organised in a database, so that it is easier to search and compare the data between the various images.

This solution proved to be valid for the problem addressed, as demonstrated by the results of the tests carried out. In a future way, there could be some corrections/additions to be made, such as, adding more requested data and the respective remodelling in the code and in the database, improving the functionalities already present in the interface and/or adding more functionalities, in order to facilitate the user's navigation through the platform.

## ***Keywords***

Metadata, Image, Exif, Data Base, Python, Graphical interface, *Flask*



# Índice

<b>AGRADECIMENTOS .....</b>	<b>I</b>
<b>RESUMO .....</b>	<b>III</b>
<b>ABSTRACT .....</b>	<b>V</b>
<b>ÍNDICE .....</b>	<b>VII</b>
<b>ÍNDICE DE FIGURAS .....</b>	<b>IX</b>
<b>ÍNDICE DE TABELAS .....</b>	<b>XI</b>
<b>ACRÓNIMOS .....</b>	<b>XIII</b>
<b>1. INTRODUÇÃO .....</b>	<b>1</b>
1.1. CONTEXTUALIZAÇÃO .....	1
1.2. OBJETIVOS .....	3
1.3. ESTRUTURA DO RELATÓRIO .....	4
<b>2. IDENTIFICAÇÃO E ANÁLISE DOS REQUISITOS .....</b>	<b>5</b>
2.1. IDENTIFICAÇÃO DOS REQUISITOS .....	5
2.2. ANÁLISE DE REQUISITOS .....	5
<b>3. REVISÃO CIENTÍFICA E TECNOLÓGICA .....</b>	<b>7</b>
3.1. ANÁLISE TECNOLÓGICA .....	7
3.2. REVISÃO CIENTÍFICA .....	9
<b>4. ESPECIFICAÇÃO .....</b>	<b>11</b>
4.1. METADADOS – EXIF .....	11
4.2. BASE DE DADOS .....	15
4.3. USER INTERFACE .....	17
<b>5. IMPLEMENTAÇÃO .....</b>	<b>20</b>
5.1. INTERFACE GRÁFICA E AS SUAS FUNCIONALIDADES ASSOCIADAS .....	20
5.2. BASE DE DADOS .....	24
5.3. SUPORTE DO BACK-END .....	25
5.4. EXTRAÇÃO E UTILIZAÇÃO DOS METADADOS .....	31
5.5. GRAFOS .....	34
<b>6. TESTES E RESULTADOS .....</b>	<b>41</b>
<b>7. CONCLUSÕES .....</b>	<b>47</b>
<b>REFERÊNCIAS DOCUMENTAIS .....</b>	<b>49</b>



## Índice de figuras

Figura 1: Metadados por detrás de imagens .....	1
Figura 2: "Knowledge Graph" da Google [Fonte: (Singhal, 2012)].....	2
Figura 3: Diagrama de blocos da solução a implementar.....	6
Figura 4: Google Fotos Layout [Fonte: (Farias, 2021)] .....	8
Figura 5: Elementos de dados numa imagem [Fonte: (Photometadata, 2021)].....	10
Figura 6: Estudo realizado que demonstra a quantidade de imagens que contêm dados “Exif” [Fonte: (Dietmar Wueller, 2007)] .....	11
Figura 7: Estrutura dos dados "EXIF" [Fonte: (Tesic, 2005)].....	12
Figura 8: Esquema de tabelas da base de dados .....	15
Figura 9: Navegação entre páginas.....	17
Figura 10: Layout da página com o formulário necessário para a criação da conta.....	17
Figura 11: Página de visualização das imagens apresentadas ao utilizador .....	18
Figura 12: Página de upload das imagens para a interface .....	19
Figura 13: Excerto de código disponibilizado pelo Bootstrap 5 .....	20
Figura 14: Utilização de <i>Jinja</i> no código <i>html</i> .....	21
Figura 15: Excerto de código: Utilização de <i>localStorage</i> na página “/UserHome” .....	21
Figura 16: Layout da página com o formulário para o login na conta .....	22
Figura 17: Ranking DB-Engines [Fonte: (Db-Engines, 2021)].....	24
Figura 18: Exemplo de pedido HTTP [Fonte: (MDN Web Docs, 2021)] .....	27
Figura 19: Exemplo de uma resposta HTTP [Fonte: (MDN Web Docs, 2021)] .....	28
Figura 20: Request <i>POST</i> ao servidor .....	29
Figura 21: Interação com a base de dados: função "validar_user" .....	30
Figura 22: Esquema de extração e armazenamento dos metadados .....	31
Figura 23: Nomes do executável "exiftool" e suas funcionalidades [Fonte: (Harvey, 2021)] .....	32
Figura 24: Excerto do código de análise do ficheiro dos dados da imagem .....	32
Figura 25: Dicionário criado para o armazenamento dos dados extraídos .....	33
Figura 26: Interação com a base de dados .....	34
Figura 27: Conexão do servidor á base .....	34

Figura 28: Estrutura gráfica de um grafo [Fonte: (Denise Cristiane Santos, 2016), editada]	35
Figura 29: Grafo de exemplo das relações entre as imagens	35
Figura 30: Estrutura do ficheiro JSON	36
Figura 31: Excerto da utilização "d3.json()"	38
Figura 32: Extração dos dados do ficheiro JSON	39
Figura 33: Processo de upload de uma imagem e a sua cronometragem	41
Figura 34: Imagem presente no nó central do grafo e os seus metadados	42
Figura 35: Imagem de teste: Grupo "Utilizador"	43
Figura 36: Imagem de teste: Grupo "Hora"	43
Figura 37: Imagem de teste: Grupo "Marca do dispositivo"	44
Figura 38: Interação de <i>drag</i> , alterando o nó de posição	45
Figura 39: Grafo com o zoom inicial	45
Figura 40: Grafo ampliado	45
Figura 41: Interação do <i>mouseover</i> e a respetiva relação do nó selecionado	46
Figura 42: Separador aberto através da interação <i>mouseclick</i>	46

## *Índice de Tabelas*

Tabela 1: Módulos utilizados .....	25
Tabela 2: Ficheiro JSON: Elemento "Nodes" .....	37
Tabela 3: Ficheiro JSON: Elemento "Links" .....	37





## *Acrónimos*

DC	-	Dublin Core
DCMES	–	Dublin Core Metadata Element Set
DSC	-	Digital Still-Camera
EXIF	–	Exchangeable image file format
GPS	–	Global Positioning System
IPTC	–	International Press Telecommunications Council
IPTC- IIM	–	International Press Telecommunications Council-Information Interchange Model
JPEG	–	Joint Photographic Experts Group
OEMM	–	Oracle Enterprise Metadata Management
PNG	–	Portable Network Graphics
RDF	–	Resource Description Framework
TIFF	–	Tagged Image File Format
URL	–	Uniform Resource Locator
VSC	–	Visual Studio Code
XMP	–	Extensible Metadata Platform
WSGI	–	Web Server Gateway Interface



# 1. INTRODUÇÃO

## 1.1. CONTEXTUALIZAÇÃO

Com o desenvolvimento das tecnologias digitais e multimédia, a quantidade de informação incluída em cada documento, quer seja imagem, vídeo, texto, áudio, é cada vez maior, guardando inúmeras características e detalhes. A informação adicional sobre os dados incluídos nos ficheiros é chamada de metadata e pode ser apresentada em diferentes tipos de dados, desde hora e data da criação até à resolução ou tamanho do ficheiro, sendo utilizada para diversos fins em torno da logística da pesquisa e apresentação de resultados [Figura 1].



Figura 1: Metadados por detrás de imágenes

Metadados são dados que se encontram associados a quase todos os tipos de ficheiros. É um tipo de dados descritivos que ajuda uma pessoa ou um computador a identificar as características de um ficheiro. Por exemplo, os metadados para um documento Microsoft Word incluem coisas como: tamanho do ficheiro, autor e data de criação, mas existem inúmeros tipos mais diferentes de metadados visíveis e escondidos para ajudar a identificar as características do ficheiro específico.

“Os metadados têm uma presença crescente nos sistemas de informação, e vêm em muitas formas. As características principais de a maioria dos pacotes de software que utilizamos todos os dias são movidos a metadata. As pessoas ouvem música através de Spotify; publicar fotos no Instagram; localizar vídeo no YouTube; gerir as finanças através do Quicken; ligar com outros através de e-mail, texto e redes sociais; e armazenar longas listas de contactos nos seus dispositivos móveis. Todo este conteúdo vem com metadata - informação sobre a criação do artigo, nome, tópico, características e afins. Os metadados são a chave muitas funcionalidades dos sistemas que detêm o conteúdo, permitindo aos utilizadores encontrar itens de interesse, registar informações essenciais sobre os mesmos, e partilhar essa informação com outros.” (Riley, 2017)

Um exemplo de uma ferramenta baseada em recolha de metadados é o “Knowledge Graph” da Google, lançado em 2012, com o objetivo de poder apresentar ao utilizador, uma resposta mais completa a cada pesquisa efetuada no seu browser. O “Knowledge Graph” permite-lhe procurar coisas, pessoas ou lugares que o Google reconhece/identifica - marcos, celebridades, cidades, equipas desportivas, edifícios, características geográficas, filmes, objetos celestes, obras de arte e mais - e obter instantaneamente informações relevantes para a sua consulta. (Singhal, 2012)[Figura 2]

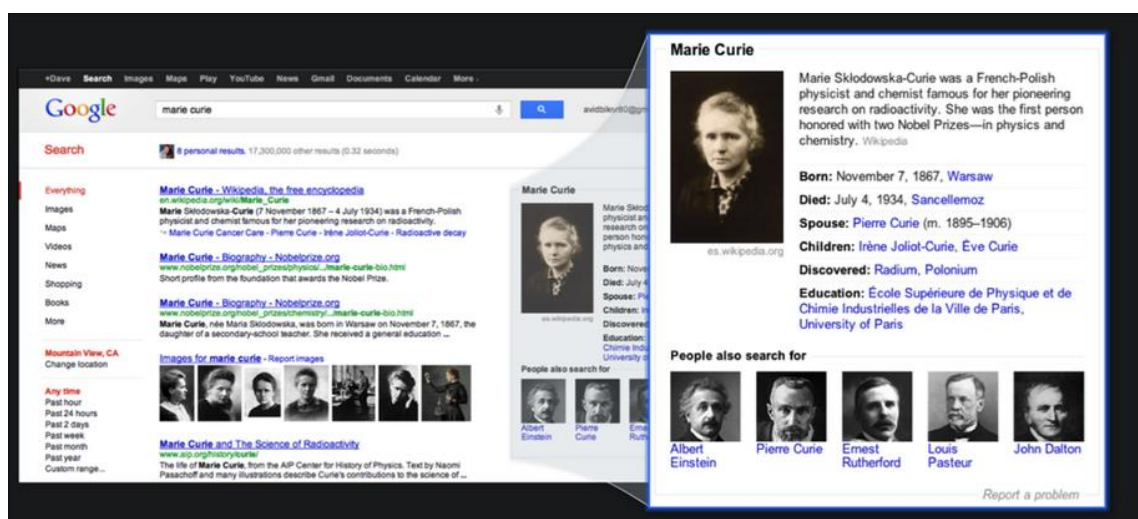


Figura 2: "Knowledge Graph" da Google [Fonte: (Singhal, 2012)]

Com o evoluir dos anos, as empresas começaram a recorrer á obtenção de dados fornecidos pelos clientes, quer seja pelas pesquisas efetuadas ou pelas interações e comentários feitos, de forma a saberem melhor as ideias, desejos e até rotinas dos clientes para perceberem o que têm de fazer para melhorar a produtividade da empresa. Os dados

recolhidos são guardados em bases de dados, de forma a organizar e disponibilizar esta informação para a empresa toda. Através desta organização, a empresa reduz a duplicação de esforços e recursos, facilita o rastreio de quem é responsável pela qualidade e atualidade desses dados, identifica conteúdos potencialmente comercializáveis, e identifica buracos na sua estratégia de dados que podem ser preenchidos com investimento em fontes de dados de qualidade. (Cagle, 2019)

Neste projeto, os metadados abordados são de um tipo específico de ficheiro: metadados de imagem. Os metadados de imagem referem-se à informação adicional sobre imagens reais, que são armazenados nos ficheiros de imagem juntamente com as próprias imagens. Estes metadados são utilizados em várias áreas no contexto online pelas empresas, forçando estas a investir nesta vertente de aquisição de dados e, por consequência, no armazenamento dos mesmos, havendo a necessidade da criação de base de dados.

## **1.2. OBJETIVOS**

Este projeto consiste no desenvolvimento de uma aplicação de acesso e visualização de conteúdo visual com a ferramenta de upload de imagens pelos utilizadores, em que cada imagem submetida é guardada e analisada, sendo guardada todos os metadados extraídos das imagens. A informação irá ser utilizada na pesquisa assistida pelos metadados. Depois, os resultados encontrados irão ser apresentados num formato que será implementado na plataforma *Web*. Para o efeito, devem ser atingidos os seguintes objetivos principais:

- Disponibilizar uma interface web acessível, de forma que o utilizador possa interagir com as funcionalidades disponibilizadas, tanto de upload como de visualização das imagens.
- Analisar e extrair corretamente os metadados pretendidos das imagens submetidas pelo utilizador.
- Criar e implementar um servidor para a gestão e armazenamento das imagens e dos respetivos metadados.

### **1.3. ESTRUTURA DO RELATÓRIO**

A estruturação deste documento é a seguinte:

No capítulo 1, é apresentado e explicado o contexto em que o projeto se insere, o seu impacto tecnológico e os vários desafios enfrentados durante o mesmo;

No capítulo 2, é apresentado o estado de arte sobre diferentes tecnologias já existentes no mercado, de obtenção de dados de imagens e formas de manipulação de bases de dados;

No capítulo 3, são apresentados os vários requisitos necessários para a realização do projeto, e de que maneira estes se interligam durante o projeto;

E nos capítulos seguintes, são apresentadas e explicadas todas as etapas do projeto, desde a especificação dos temas e tecnologias utilizadas até à implementação das tecnologias para o devido funcionamento do projeto.

De seguida, são apresentados alguns testes propostos e os seus respetivos resultados.

Por último, são apresentadas reflexões sobre o trabalho realizado e possíveis melhorias

## 2. IDENTIFICAÇÃO E ANÁLISE DOS REQUISITOS

### 2.1. IDENTIFICAÇÃO DOS REQUISITOS

Para a realização deste projeto, foi identificado um conjunto de requisitos que devem ser satisfeitos pelo sistema a desenvolver. Esses requisitos são:

- Existência de uma interface web para interação do utilizador, em que seja possível a interação com as imagens dos partilhadas pelos utilizadores
- Capacidade de analisar e guardar as imagens submetidas, incluindo suporte para os seguintes formatos: JPG, JPEG, HEIC
- Capacidade de armazenamento, de forma estruturada, das imagens e dos respetivos dados extraídos
- Capacidade de comparar os dados das várias imagens submetidas e de apresentar as mesmas na interface web, apresentado as imagens e as suas relações de uma forma acessível e interativa
- Criação de um servidor programado em Python

### 2.2. ANÁLISE DE REQUISITOS

O projeto consistirá numa interface web que dará ao utilizador a opção de upload e visualização e interação com imagens submetidas. As imagens submetidas na interface gráfica serão redirecionadas pelo servidor para um repositório local, onde estas serão guardadas e organizadas. Antes de serem redirecionadas, as imagens serão analisadas, guardando os dados extraídos numa base de dados, que se encontra conectada ao servidor. Este processo anteriormente descrito encontra-se ilustrado na Figura 3.

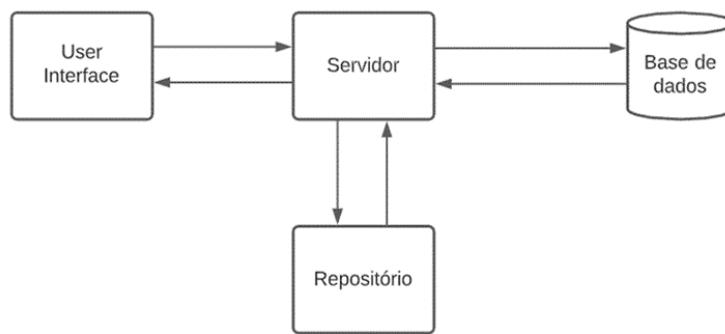


Figura 3: Diagrama de blocos da solução a implementar

A interface gráfica tem como objetivo a interação do utilizador com as funcionalidades disponibilizadas, de modo a que haja troca de informação entre o utilizador e o sistema, permitindo a obtenção de informação e de imagens e, posteriormente, a apresentação dessas mesmas, de acordo com a necessidade e o interesse do utilizador.

A base de dados irá servir para armazenamento e organização dos dados extraídos das imagens. Da mesma forma, o repositório das imagens irá servir para guardar todas as imagens submetidas, de modo que possam ser acedidas novamente pelo servidor.

Por fim, o funcionamento geral do sistema consiste na troca de informação entre os diferentes elementos e o servidor central, fazendo com esta comunicação seja bidirecional, ou seja, pode ser o servidor a solicitar informação aos restantes elementos, como imagens ou informação presente na base de dados, ou algum dos elementos a enviar informações para o servidor, como por exemplo, o upload de imagens a partir da interface gráfica.



# 3. REVISÃO CIENTÍFICA E TECNOLÓGICA

## 3.1. ANÁLISE TECNOLÓGICA

### 3.1.1. EXTRAÇÃO DOS METADADOS

Existe várias ferramentas disponíveis para a obtenção dos dados presentes nas imagens, desde ferramentas online até programas de edição e extração dos mesmos dados. Estas ferramentas, como já foi dito, têm como objetivo extrair os dados associados aos ficheiros de imagem, sendo, posteriormente, guardados numa base de dados. Desta forma irão ser apresentadas alguns tipos de ferramentas de extração de dados:

**Metadata2Go:** ferramenta online gratuita que permite ter acesso aos metadados guardados no ficheiro submetido. Esta plataforma permite a análise de qualquer tipo de formato de ficheiro, dando o acesso às informações que cada ficheiro guarda. Após a análise e apresentação dos dados, esta plataforma oferece a possibilidade de alterar alguns dos dados extraídos do documento, e posteriormente, fazer o download da imagem com os dados novos

**MediaInfo:** software gratuito e disponível para todas as plataformas que permite ter acesso às informações dos ficheiros escolhidos. Este software é mais orientado para ficheiros de vídeo e áudio e dá acesso a dados mais específicos em relação a este tipo de ficheiros. Apesar disso, também analisa imagens, apresentando apenas as informações principais e gerais, em relação ao documento

### 3.1.2. UTILIZAÇÃO DOS METADADOS

O Google Fotos é um exemplo da utilização dos metadados presentes nas fotos que são armazenadas na cloud. Trata-se de um serviço de compartilhamento e armazenamento de fotos desenvolvido pelo Google, onde os utilizadores podem guardar todas as suas fotos de uma forma segura e acessível. No entanto, para além destas funcionalidades principais, o Google Fotos dispõe de algumas funcionalidades extras, facilitando, ao utilizador, a procura e a apresentação das fotos. Consoante a pesquisa do utilizador, o Google Fotos utiliza alguns dos dados presentes nas fotos, para facilitar a procura do utilizador, agrupando as fotos tendo em conta, por exemplo, a localização, o horário ou até as pessoas e paisagens presentes nas fotos. [Figura 4]

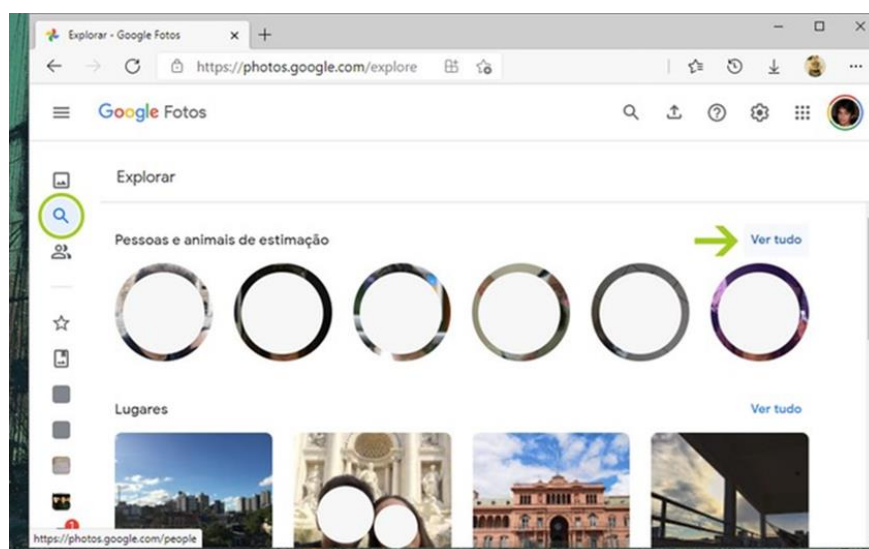


Figura 4: Google Fotos Layout [Fonte: (Farias, 2021)]

Os metadados também são muito utilizados pelas redes sociais. Os utilizadores do Facebook criam metadados quando gerem listas de amigos, publicam status ou adicionam descrições nas publicações, interagem com as publicações dos amigos e partilham conteúdos já publicados. Ao acompanhar estas atividades, o Facebook analisa tópicos de tendência e promove publicações patrocinadas que geram rendimento. Os utilizadores do Instagram fornecem legendas às imagens que publicam e partilham, e seguem as contas de outros utilizadores e empresas. O Instagram utiliza estes dados de interação para melhorar a sua publicidade. Os utilizadores do Twitter organizam as pessoas que seguem em listas, postam texto e imagens, usam *hashtags* para comentar tweets e conectá-los a outros, retweetar conteúdo de outros com ou sem comentários, e tweets "favoritos", impulsionando funcionalidades como a lista de tópicos de tendência do Twitter. (Riley, 2017)

### 3.2. REVISÃO CIENTÍFICA

No mundo da multimédia, existem várias normas de metadata de imagem. Cada um é mais especializado para um tipo de objetivos, havendo algumas normas que são mais utilizados, no geral, atualmente. A lista das normas mais relevantes para o projeto é a seguinte: [Figura 5]

- EXIF

Exchangeable Image File Format (EXIF) é o padrão de armazenamento de ficheiros de imagem para câmaras fotográficas digitais. Foi desenvolvido pela Associação para o Desenvolvimento da Indústria Eletrónica do Japão (JEIDA) como uma forma padrão de armazenamento de imagens criadas por câmaras digitais, bem como metadados sobre as imagens. Os metadados de imagens EXIF podem ser armazenados em imagens em formato TIFF e JPEG. (Pelski, 2005)

- XMP

Extensible Metadata Platform (XMP) é um formato padrão de metadados, desenvolvido pela Adobe, para a criação, processamento, e intercâmbio de metadados numa variedade de aplicações. O XMP utiliza a tecnologia Resource Description Framework (RDF) para a modelação de dados. XMP também define como o modelo de dados é serializado (convertido num fluxo de bytes), e incorporado num ficheiro de imagem. (Pelski, 2005)

- IPTC

International Press Telecommunications Council-Information Interchange Model (IPTC-IIM) é uma norma desenvolvida conjuntamente pelo International Press Telecommunications Council e pela Newspaper Association of America. Esta norma de metadados foi concebida para capturar informação que é importante para as atividades de recolha de notícias, reportagens e publicação. Estes registos de informação são geralmente referidos como etiquetas IPTC. A utilização de etiquetas IPTC incorporadas em formatos de ficheiros de imagem tornou-se generalizada com a utilização da ferramenta Adobe Photoshop para a edição de imagens. Os metadados IPTC podem ser armazenados em imagens em formato TIFF e JPEG. (Pelski, 2005)

- Dublin Core

O Dublin Core Metadata Element Set (DCMES) surgiu de uma reunião de 1995 em Dublin, Ohio, que se focou em metadados para informação eletrônica em rede. Os participantes foram incumbidos de identificar um conjunto de funcionalidades comuns à maioria dos tipos de informação digital. Neste primeiro encontro foram definidos 13 elementos fundamentais, que rapidamente cresceram para os 15 elementos conhecidos como DCMES hoje. Estes são: contribuinte, cobertura, criador, data, descrição, formato, identificador, idioma, editor, relação, direitos, fonte, sujeito, título e tipo. Este conjunto, também conhecido como "Simple Dublin Core", ou Dublin Core (DC), é padronizado como ISO 15836 e ANSI/NISO Z39.85, ambos chamados “The Dublin Core metadata element set”. (Riley, 2017)

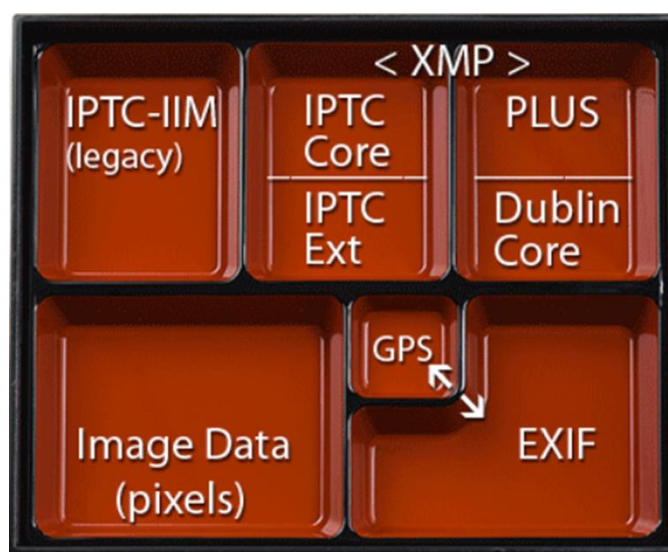


Figura 5: Elementos de dados numa imagem [Fonte: (Photometadata, 2021)]

# 4. ESPECIFICAÇÃO

## 4.1. METADADOS – EXIF

O Exchangeable Image File Format (EXIF) é um padrão que define informações específicas relacionadas com uma imagem ou outros meios capturados por uma câmera digital. É capaz de armazenar dados tão importantes como a exposição à câmera, data/hora em que a imagem foi capturada e até mesmo a localização do GPS (Global Positioning System). Atualmente, todas as câmeras digitais modernas têm a capacidade de gravar esta informação, juntamente com muitas outras configurações de câmeras e outros dados relevantes, diretamente em fotografias. Estas configurações podem então ser usadas mais tarde para organizar fotografias, realizar pesquisas e fornecer informações vitais aos fotógrafos sobre a forma como uma determinada fotografia foi capturada. (MANSUROV, 2020)

### 4.1.1. ENQUADRAMENTO

A escolha do standard EXIF baseia-se no propósito deste projeto, contendo as informações e dados necessários para a realização do mesmo. Para além disso, como está demonstrado na Figura 6, a maior parte das imagens existentes contêm dados Exif, sendo mais um motivo para escolha da norma Exif para o projeto presente neste relatório.

#### 2.1 How many images have embedded Exif data?

How many images contain Exif information	Images	Percent
Evaluated images	4.788.645	
Images containing Exif information (ISO tag or camera manufacturer tag)	3.447.579	72,0
Images without Exif information (no ISO tag and no camera manufacturer tag)	1.341.066	28,0

Figura 6: Estudo realizado que demonstra a quantidade de imagens que contêm dados “Exif” [Fonte: (Dietmar Wueller, 2007)]

#### 4.1.2. ESQUEMA DE DADOS DO EXIF

A imagem digital contém uma estrutura específica, como demonstra a Figura 7, estando os dados “Exif” guardados numa secção específica, apresentando todo o tipo de dados presentes na imagem analisada.

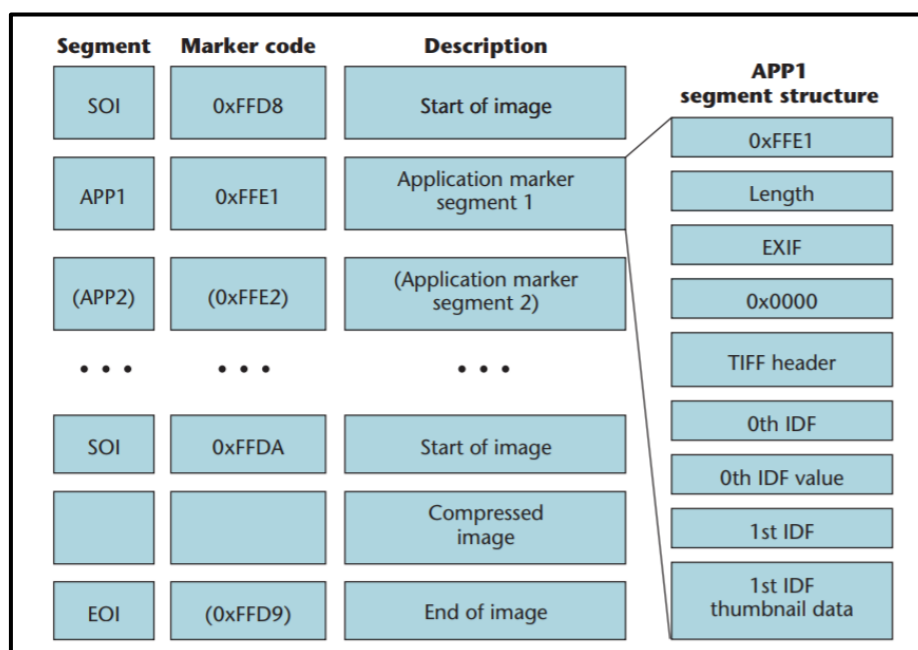


Figura 7: Estrutura dos dados "EXIF" [Fonte: (Tescic, 2005)]

*“Uma imagem digital comprimida é gravada como um ficheiro JPEG, com segmentos de marcador de aplicação (APP1 e APP2). Os metadados são compostos por segmentos de aplicação JPEG e são determinados por marcadores de aplicação, variando em valor binário de 0xFFE0 a 0xFFEF. Estes segmentos de aplicação são armazenados antes do Start of Stream (SOS) (0xFFED) que contém dados de imagem comprimidos. A Figura 7 mostra a estrutura de um ficheiro de imagem digital EXIF comprimido que está em conformidade com a norma JPEG.*

*Cada ficheiro JPEG começa com um Start of Image (SOI) (0xFFD8) com o valor binário 0xFFD8 e termina com um End of Image (EOI) com o valor binário xFFD9. A norma EXIF utiliza marcadores de segmento APP1 (0xFFE1) e APP2 (0xFFE2) para segmentos JPEG para armazenar metadados de imagem. Se os metadados EXIF estiverem disponíveis, são armazenados no segmento APP1. Este segmento segue imediatamente após o marcador SOI.*

*A Figura 7 também mostra a estrutura de um segmento APP1. O código de identificação EXIF 4 byte (ascii caracteres EXIF) indica que a interoperabilidade do segmento APP1 está no formato EXIF. É seguido por 0x00 gravados em 2 bytes, e depois os dados exif seguem. O EXIF utiliza o formato TIFF para armazenar dados.*

*A informação do atributo é gravada em dois diretórios de ficheiros de imagem (IFDs) em campos de metadados. O IFD inicial num ficheiro é o 0º IFD, e contém a informação geral de metadados de uma imagem. O próximo IFD, o 1º IFD, contém uma informação de atributos de uma miniatura.” (Tescic, 2005)*

A estrutura de dados do EXIF contem vários tipos de dados, cada um identificado por uma tag, disponibilizando todo o tipo de informações sobre as imagens analisadas. De acordo com o projeto, foi escolhido um subconjunto de tags que mais se adequaram ao problema proposto. As tags escolhidas são as seguintes:

- **Make:** Permite guardar a identificação do fabricante do equipamento de gravação, por exemplo, DSC (digital still-camera), scanner, digitalizador de vídeo ou outros equipamentos que geraram a imagem. Quando o campo é deixado em branco, é tratado como desconhecido.
- **Model:** Permite guardar o nome ou o número do modelo do equipamento, por exemplo, DSC, scanner, digitalizador de vídeo ou outros equipamentos que geraram a imagem. Quando o campo é deixado em branco, é tratado como desconhecido.
- **Date/Time Original:** Permite guardar a data e a hora em que os dados originais da imagem foram gerados. O formato é "YYYY:MM:DD HH:MM:SS", com a hora apresentada no formato de 24 horas, e a data e a hora separadas por um caracter em branco.

A extração das coordenadas GPS da imagem está dividida em 2 tags diferentes:

- **GPSLatitude:** Indica a latitude. A latitude é expressa como três valores racionais dando os graus, minutos e segundos, respetivamente. Se a latitude for expressa em graus, minutos e segundos, um formato típico seria dd/1,mm/1,ss/1. Quando se utilizam graus e minutos e, por exemplo, frações de minutos são dadas até duas casas decimais, o formato seria dd/1,mmmm/100,0/1.

- **GPSTLongitude:** Indica a longitude. A longitude é expressa como três valores racionais dando os graus, minutos e segundos, respetivamente. Se a longitude for expressa em graus, minutos e segundos, um formato típico seria ddd/1,mm/1,ss/1. Quando graus e minutos são usados e, por exemplo, frações de minutos são dadas até duas casas decimais, o formato seria ddd/1,mmmm/100,0/1.
- **UserComment:** Uma tag para os utilizadores escreverem palavras-chave ou comentários sobre a imagem, além das que estão ou podem estar na ImageDescription, e sem as limitações do código de caracteres da tag ImageDescription.

A extração da resolução da imagem está dividida em 2 tags diferentes:

- **ImageWidth:** Permite guardar o número de colunas de dados de imagem.
- **ImageLength:** Permite guardar o número de filas de dados de imagem.

Por último, a extração do formato da imagem baseia-se numa tag:

- **File Type Extension:** Permite guardar o formato da imagem analisada.



## 4.2. BASE DE DADOS

De forma a guardar os valores obtidos de cada imagem carregada para o sistema, define-se uma base de dados relacional, com o objetivo de todos os dados serem guardados e organizados corretamente, estando apta para ser analisada. A base de dados irá conter várias tabelas, para que a informação armazenada esteja melhor organizada. Estas tabelas irão guardar os valores que foram extraídos das imagens submetidas, organizando-os em termos da relação que têm com a imagem e os campos dos metadados utilizados. [Figura 8]

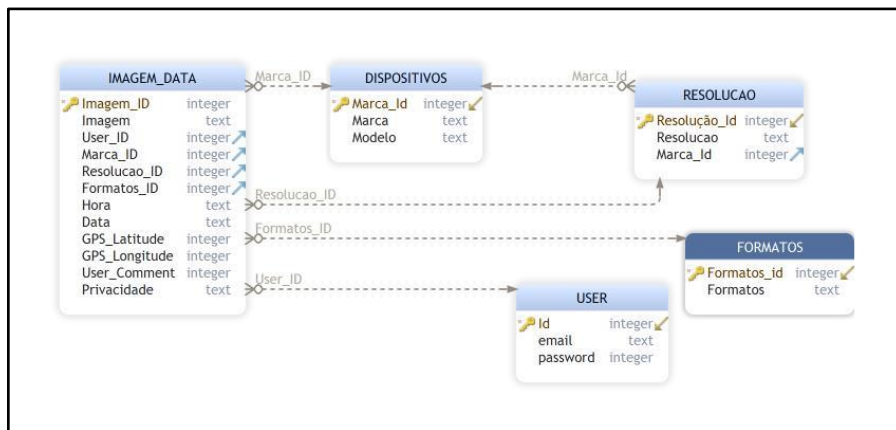


Figura 8: Esquema de tabelas da base de dados

A tabela “IMAGEM\_DATA” é a tabela principal da base de dados, onde vão estar organizados todos os dados de cada imagem submetida pelo utilizador. Aqui são organizados dois tipos de dados em relação às imagens: os dados comuns e os dados não comuns.

Alguns dos dados que são extraídos das imagens diferem de imagem para imagem, sendo praticamente impossíveis de prever, como a hora, dia e até localização onde a imagem foi tirada. Estes dados serão chamados de dados não comuns e vão ser armazenados diretamente na tabela principal. Os dados não comuns vão ser apresentados diretamente nas colunas:

- Imagem
- Hora
- Data
- GPS\_Latitude
- GPS\_Longitude
- User\_Comment
- Privacidade (onde dirá se a foto pode ser apresentada publicamente ou não)

Os dados comuns vão ser apresentados na tabela principal, através de *ids*, que correspondem á linha onde estes se encontram na respetiva tabela independente. Estas tabelas independentes foram criadas para armazenar os dados que são, de alguma forma, previsíveis ou repetitivos em todas as imagens analisadas. Esta organização dos dados comuns em várias tabelas separadas, torna a base de dados mais eficiente, evitando armazenamento desnecessário com a repetição de informação adquirida. Cada tabela vai estar relacionada com um tópico específico, tendo sempre uma coluna que guardará os *ids* únicos de cada linha dessa tabela, que será a “Primary Key” da respetiva tabela. Desta forma, vão existir 4 tabelas específicas:

- User

A tabela “User” vai guardar as credenciais de cada utilizador que fizer registo no sistema através da página Web. Cada utilizador vai ter um ID associado na coluna “Id”, “Primary Key” da tabela, sendo utilizado na tabela “IMAGEM\_DATA” para associar a imagem submetida ao utilizador a que pertence.

- Dispositivos

A tabela “Dispositivos” vai guardar a marca e modelo de cada dispositivo utilizado para tirar a foto submetida pelo utilizador. Cada marca vai ter um ID associado na coluna “Marca\_id”, “Primary Key” da tabela, sendo este utilizado na tabela “IMAGEM\_DATA” para associar a imagem submetida à marca do dispositivo que a tirou.

- Formatos

A tabela “Formatos” vai guardar o tipo de formatos de cada imagem submetida. Cada formato vai ter um ID associado na coluna “Formatos\_id”, “Primary Key” da tabela, sendo este utilizado na tabela “IMAGEM\_DATA” para associar a imagem submetida ao formato dessa imagem.

- Resolução

A tabela “Resolução” vai guardar as resoluções de cada imagem submetida. Cada resolução vai ter um ID associado na coluna “Resolução\_Id”, “Primary Key” da tabela, sendo este utilizado na tabela “IMAGEM\_DATA” para associar a resolução obtida à imagem submetida.

### 4.3. USER INTERFACE

Para a existência da interação do utilizador, deverá ser implementada uma interface web em que o utilizador poderá fazer upload das suas imagens, ver as imagens publicadas pelos outros utilizadores, relacionando-as entre elas tendo em conta os dados de cada uma. Estas funcionalidades estarão distribuídas por diferentes páginas, que estão identificadas por rotas, como está apresentado na Figura 9.

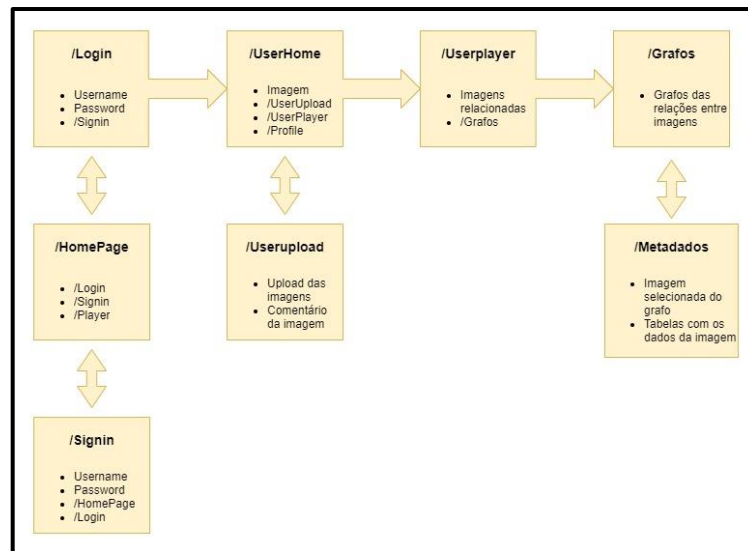


Figura 9: Navegação entre páginas

#### 4.3.1. LAYOUT DA INTERFACE

Uma das primeiras páginas apresentadas ao utilizador será a página de criação de conta. [Figura 10]

A imagem mostra a interface de login de uma aplicação web. No topo, há uma barra de navegação com links para "Home", "Feed" e "Upload", e botões para "Login" e "Sign In" no canto direito. O formulário principal, intitulado "Sign In", está centralizado e contém:

- Um campo de entrada para "Username" com um ícone de lupa à direita.
- Um campo de entrada para "Password" com um ícone de lupa à direita e um link "Forgot?" em azul.
- Um botão "Sign In" de cor escura.
- Um link "Already a member? Login now" em azul na base do formulário.

Figura 10: Layout da página com o formulário necessário para a criação da conta

Nesta página, o utilizador poderá criar uma conta, de forma a poder fazer o upload das suas imagens e, assim, associar essas imagens ao seu perfil. Para criar a conta, o utilizador terá de preencher o formulário acima apresentado, obtendo, assim, acesso às funcionalidades disponibilizadas pela interface. Após a criação da conta, o utilizador poderá aceder á sua conta através da página de login e, assim, ter acesso á página de visualização das imagens disponíveis na interface. [Figura 11]

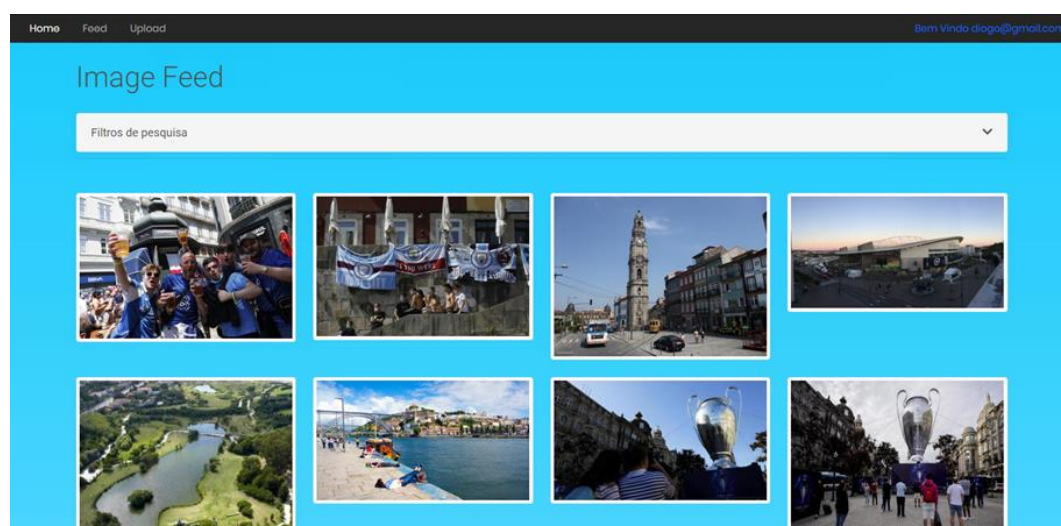


Figura 11: Página de visualização das imagens apresentadas ao utilizador

Nesta página, o utilizador pode interagir com as imagens apresentadas, desde filtrar as imagens que são apresentadas (data e hora), até saber as relações entre as várias imagens apresentadas, podendo assim, encontrar novas imagens dentro das condições que o utilizador deseje. A partir desta página, o utilizador pode aceder às restantes funcionalidades disponibilizadas pela interface, como por exemplo, a página onde irá poder fazer o upload das imagens desejadas. [Figura 12]

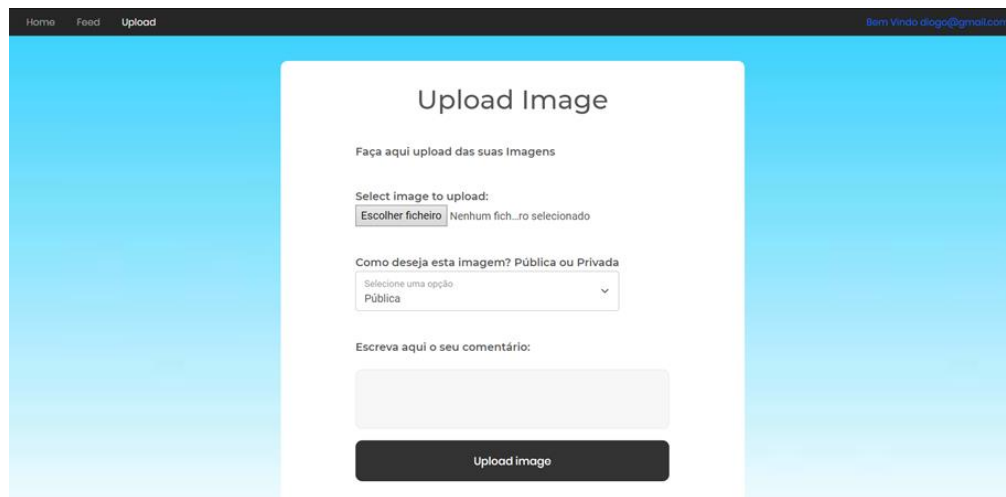


Figura 12: Página de upload das imagens para a interface

Nesta página, o utilizador pode adicionar a imagem que deseja fazer upload, seleccionar a sua privacidade, ou seja, se os outros utilizadores irão poder ver, ou não, a imagem carregada e adicionar um breve comentário sobre a imagem escolhida.

# 5. IMPLEMENTAÇÃO

A implementação deste projeto está dividida em 4 partes: Interface gráfica e as suas respectivas funcionalidades, base de dados, suporte de um servidor com a extração e armazenamento dos metadados e grafos para pesquisa e navegação.

## 5.1. INTERFACE GRÁFICA E AS SUAS FUNCIONALIDADES ASSOCIADAS

### 5.1.1. FERRAMENTAS E FUNCIONALIDADES UTILIZADAS

Na interface gráfica, foi utilizado frameworks para o html como o Bootstrap 5, para a construção da interface para o utilizador, pois disponibiliza vários componentes e ferramentas que agilizam a criação da interface web, bem como templates já estruturados de forma gratuita.

```
54 </head>
55 <body>
56   <div>
57     <nav class="navbar-expand-lg navbar-dark bg-dark">
58       <div class="container-fluid">
59         <div class="collapse navbar-collapse" id="navbarSupportedContent">
60           <ul class="navbar-nav me-auto mb-2 mb-lg-0">
61             <li><a class="nav-link active px-3" href="/HomePage" >Home</a></li>
62             <li><a class="nav-link px-3" href="/Player" >Feed</a></li>
63           </ul>
64           <div class="d-flex align-items-center">
65             <ul class="navbar-nav me-auto mb-2 mb-lg-0">
66               <li><a class="nav-link px-3" href="/Login" >Login</a></li>
67               <li><a class="nav-link px-3" href="/Signin" >Sign in</a></li>
68             </ul>
69           </div>
70         </div>
71       </div>
72     </nav>
73   </div>
```

Figura 13: Excerto de código disponibilizado pelo Bootstrap 5

O excerto de código presente na Figura 13 representa uma barra de navegação presente no topo de cada página para a navegação entre as várias páginas disponibilizadas ao utilizador, onde a cor, tipo de letra e animações estão pré-definidas nos templates disponibilizados.

Para a interface gráfica, foi também utilizado uma funcionalidade do módulo “*Flask*” chamado “*Jinja*”, que permite executar código em Python num documento *HTML*. Desde modo, foi possível enviar algumas variáveis necessárias, presentes no servidor, para a interface, facilitando a construção a interface web. Um exemplo da sua utilização é o *template* da rota “/HomePage”.

```
80
81 <div class="row text-center text-lg-start">
82   {% for image in images %}
83   <div class="col-lg-3 col-md-4 col-6">
84     <a href="/Player" target="_blank" class="d-block mb-4 h-100">
85       
86     </a>
87   </div>
88   {% endfor %}
89 </div>
```

Figura 14: Utilização de *Jinja* no código *html*

Na Figura 14, na linha 82 e 88, a funcionalidade *Jinja* foi utilizado para enviar, para a interface, uma lista com os nomes das fotos pretendidas. Para esta lista ser lida adequadamente, foi criado um loop “*for*”, de forma que a lista ser percorrida do início ao fim, fazendo com que, neste caso, as imagens presentes na lista sejam todas apresentadas. Na linha 85, cada elemento da lista será utilizado para identificar o “*src*” de cada imagem que será apresentada na interface web.

Por último, foi utilizado também uma API da web chamada *localStorage*, que permite o armazenamento de dados do cliente, agilizando o processo de requisição de dados, ao não compartilhar as informações com o servidor, e de transferência de dados entre páginas web, guardando os dados inseridos pelo utilizador entre as páginas.

```
211
212     imagens[i].dataset.index = i;
213     imagens[i].onclick = function() {
214         var index = this.dataset.index;
215         var src = this.src
216         /* console.log(index);
217         console.log(src);*/
218
219         src=src.replace(/thumbnails/g, 'imagens');
220         /* console.log(src);*/
221
222         ImageSource1 = src;
223
224         localStorage.setItem("ImageSource1", ImageSource1);
225         console.log(localStorage.getItem("ImageSource1"));
226     }
```

Figura 15: Excerto de código: Utilização de *localStorage* na página “/UserHome”

Na Figura 15, está apresentado um excerto do código da “/UserHome”, onde é utilizado *localStorage* para guardar o nome de uma imagem que tenha sido selecionada pelo utilizador. Este nome é utilizado na página “/Userplayer”, para que a imagem selecionada pelo utilizador seja aberta noutra separador, e permita a interação com a imagem e/ou outras imagens relacionadas.

### 5.1.2. LAYOUT E FUNCIONALIDADES DOS *TEMPLATES* WEB

Os *templates* presentes em cada página têm várias funcionalidades durante a interação com os elementos presentes, podendo haver vários resultados durante as interações. Ao aceder à página “/Signin”, irá aparecer o formulário apresentado na Figura 10, dando a possibilidade de o utilizador criar a sua conta. Para preencher este formulário devidamente, o utilizador terá de inserir um *username* em formato de email e a sua respetiva *password*, sendo forçado a inserir estes dados, pois são dados requeridos pelo servidor. Ao carregar no botão “Sign in”, os dados inseridos serão verificados no servidor, mais especificamente, o *username* será pesquisado para a possibilidade de já haver um *username* igual. Neste caso, irá aparecer um aviso de “Este user já existe”, e o utilizador terá de o mudar para outra que não tenha sido utilizado. Caso contrário, o registo será completo e submetido, guardando as suas credenciais na base de dados. Após este procedimento, o utilizador será redirecionado para a página de “Login”, para efetuar o seu login na página. [Figura 16]

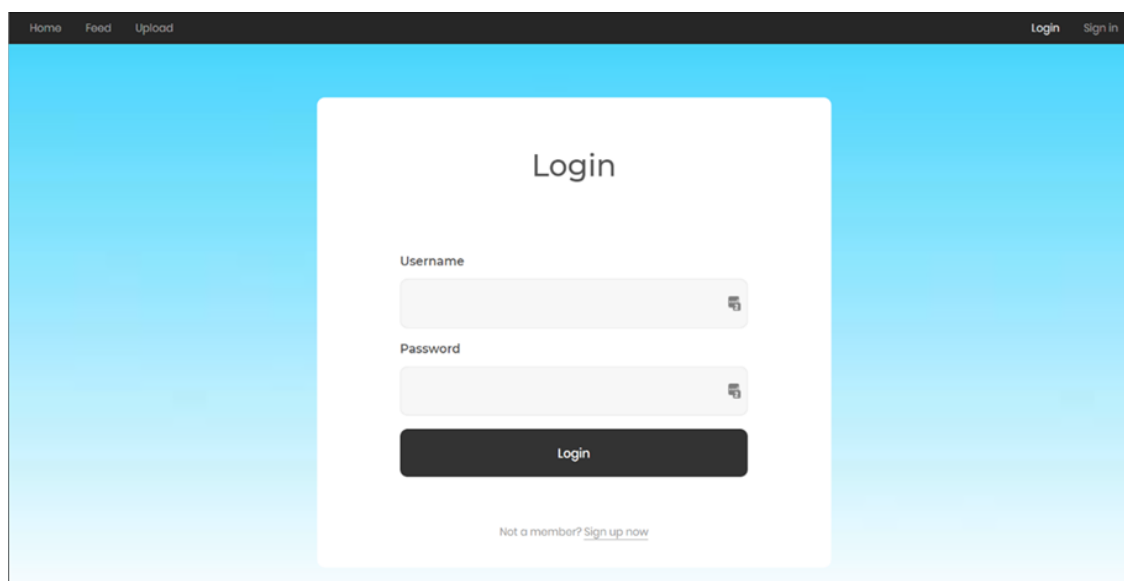


Figura 16: Layout da página com o formulário para o login na conta



Ao preencher este formulário, as credenciais submetidas vão ser comparadas com as credenciais armazenadas na base de dados, tendo em conta que o utilizador é forçado inserir um *username* e uma *password*, pois são dados requeridos pelo servidor, havendo 3 resultados possíveis para a pesquisa efetuada:

- Caso o *username* submetido não exista, aparecerá uma mensagem ao utilizador, a dizer que não existe nenhuma conta com esse *username*, sendo redirecionado para a página do “Sign in”.
- Caso o *username* submetido exista, mas a *password* inserida não seja igual á que está guardada na base de dados, aparecerá uma mensagem avisando de que a *password* inserida está errada.
- Caso os dados inseridos estejam corretos e válidos, o utilizador será redirecionado para uma “HomePage”.

Na página de “/Userupload” [Figura 12], o utilizador tem a possibilidade de fazer upload duma imagem, tendo a condição de, se for mais do que uma imagem, terá de submeter uma foto de cada vez. Para escolher a imagem desejada, terá de carregar no botão “Escolher ficheiro”. Se tentar fazer o upload de ficheiros que não sejam formato de imagem (JPEG, JPG, TIFF, HEIC), irá aparecer um aviso de erro “Ficheiro não permitido”. Depois do utilizador escolher a imagem para submeter, terá a opção de escolher se deseja a imagem publica ou privada para os outros utilizadores. Caso seja privada, a imagem não aparecerá para os outros utilizadores, e apenas aparecerá no seu perfil. Caso seja pública, todos os utilizadores poderão ver e interagir com a respetiva imagem. O utilizador pode também adicionar uma breve descrição/comentário sobre a imagem, de forma a saber o melhor o contexto da imagem submetida. Se não houver erro nenhum, irá aparecer um aviso “Imagem guardada com sucesso” e irá continuar nesta página, caso queira fazer upload de uma nova imagem.

## 5.2. BASE DE DADOS

### 5.2.1. CONCEITO E SUAS FUNCIONALIDADES

“Uma base de dados é uma ferramenta de recolha e organização de informações, que armazenam um conjunto de informação estruturada e relacionada entre si, sobre um determinado tema ou domínio. Permite gerir enormes volumes de dados de modo a facilitar a organização, a manutenção e a pesquisa de dados, bem como outros tipos de operações processados por meios informáticos.” (AprendIS, 2016). Através desta ferramenta, a gestão de qualquer tipo de dados torna-se altamente eficiente e acessível, tornando a sua utilização imprescindível, em qualquer tipo de trabalhos e serviços existentes atualmente.

### 5.2.2. FERRAMENTAS UTILIZADAS

Existem várias ferramentas disponíveis para criar e gerir uma base de dados, tendo ainda a opção de ligação ao servidor através do código programado. Para este efeito, existem várias opções possíveis de softwares de base de dados. Entre os softwares mais utilizados encontram-se o MySQL, Oracle, Microsoft SQL Server, MongoDB e o SQLite. Este último foi escolhido para este projeto. O SQLite é uma base de dados relacional que, á diferença das outras ferramentas do tipo, não armazena informações num servidor. Essa independência acontece porque ele consegue colocar os seus arquivos dentro de si próprio. O SQLite também não precisa de nenhum tipo de configuração. Assim, a sua utilização como base de dados para diferentes tipos de aplicações torna-se, além de fácil, mais fluída, dinâmica e leve. Todo esse potencial tem colocado o SQLite como a 9ª opção de base de dados mais utilizada em todo o mundo. (Souza, 2020)[Figura 17]









Rank			DBMS	Database Model	Score		
Jun 2021	May 2021	Jun 2020			Jun 2021	May 2021	Jun 2020
1.	1.	1.	Oracle +	Relational, Multi-model 	1270.94	+1.00	-72.65
2.	2.	2.	MySQL +	Relational, Multi-model 	1227.86	-8.52	-50.03
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model 	991.07	-1.59	-76.24
4.	4.	4.	PostgreSQL +	Relational, Multi-model 	568.51	+9.26	+45.53
5.	5.	5.	MongoDB +	Document, Multi-model 	488.22	+7.20	+51.14
6.	6.	6.	IBM Db2	Relational, Multi-model 	167.03	+0.37	+5.23
7.	7.	↑ 8.	Redis +	Key-value, Multi-model 	165.25	+3.08	+19.61
8.	8.	↓ 7.	Elasticsearch +	Search engine, Multi-model 	154.71	-0.65	+5.02
9.	9.	9.	SQLite +	Relational	130.54	+3.84	+5.72
10.	10.	↑ 11.	Microsoft Access	Relational	114.94	-0.46	-2.24

Figura 17: Ranking DB-Engines [Fonte: (Db-Engines, 2021)]

### 5.3. SUPORTE DO BACK-END

A linguagem utilizada para programar este projeto foi Python. O motivo da utilização desta linguagem, para além de ter sido atribuído como um requisito, baseia-se também na variedade de módulos existentes que torna mais eficiente a programação do utilizador, quer seja front-end, quer seja back-end. Os módulos utilizados estão presentes na Tabela 1:

Tabela 1: Módulos utilizados

Biblioteca	Utilização
<i>render_template</i>	Retornar um ficheiro html com um template que irá ser apresentado na página web.
<i>redirect</i>	Redirecionar para uma rota em específico definida anteriormente.
<i>request</i>	Requisição de dados submetidos em cada página
<i>session</i>	Partilha de dados de página para página. Ao guardar um valor na variável “session”, esta pode ser acedida em qualquer página, desde que seja adicionada no “return” feito na mudança de página.
<i>os</i>	Permite aceder a funcionalidades do sistema operativo através de código
<i>shutil</i>	Permite mover ficheiros entre diferentes pastas

### 5.3.1. SERVIDOR

No servidor, é utilizado uma funcionalidade disponibilizada pelo *Flask*, que agiliza a navegação entre páginas web. Esta funcionalidade consiste na criação de rotas de navegação, cada uma relacionada com uma página web, tornando mais acessível, a navegação entre páginas. Para a navegação entre as páginas da interface gráfica, foram criadas rotas, estando elas dispostas como é apresentado na Figura 9. Após entrar na sua conta, na rota “/Login”, o utilizador terá acesso a todas as funcionalidades, estando elas disponíveis nas rotas “/UserHome”, /Userupload, “/Userplayer”, “/Grafos”, “/Profile” e “/Metadados”. [Figura 9]

Para verificar as credenciais inseridas no “Login”, é necessário efetuar *request* entre a interface gráfica e o servidor. O protocolo utilizado para a comunicação entre o servidor e a interface gráfica foi o protocolo HTTP. O protocolo HTTP assenta no modelo cliente-servidor e é composto por duas partes: pedido e respostas. O formato correto para os pedidos e respostas HTTP depende da versão do protocolo que o cliente e o servidor estão a utilizar. As versões mais utilizadas na internet são o HTTP/1.0 e o HTTP/1.1. Nos pedidos e respostas, tanto o cliente como o servidor devem indicar a versão HTTP que estão a utilizar. (IBM, 2019).

O pedido é feito por um cliente a um determinado *hostname* que está localizado num servidor. Este pedido tem como objetivo aceder a algum recurso guardado no servidor. Para enviar um pedido, o cliente utiliza vários componentes da URL, que incluiu as informações necessárias para aceder ao recurso pretendido. Um pedido HTTP contém os seguintes elementos:

- A linha de pedido
- O cabeçalho
- Mensagem no corpo, se necessário

A linha de pedido é a primeira linha que aparece na mensagem de pedido e é composta por três componentes:

- O método: É um comando de uma palavra que identifica ao servidor o que é pretendido fazer com recurso pedido. Por exemplo, o servidor pode ser solicitado para enviar algum recurso de volta para o cliente.
- O *path*: Este *path* identifica o recurso que está a ser pedido ao servidor.
- A versão HTTP

Os cabeçalhos HTTP consistem numa mensagem para fornecer ao servidor informações sobre a mensagem, o remetente e a forma de comunicação que o cliente se pretende comunicar. O corpo da mensagem serve para enviar dados adicionais para o servidor, no caso de um pedido com o método *POST*. Na Figura 18, é possível ver o exemplo de um pedido HTTP. (IBM, 2019)

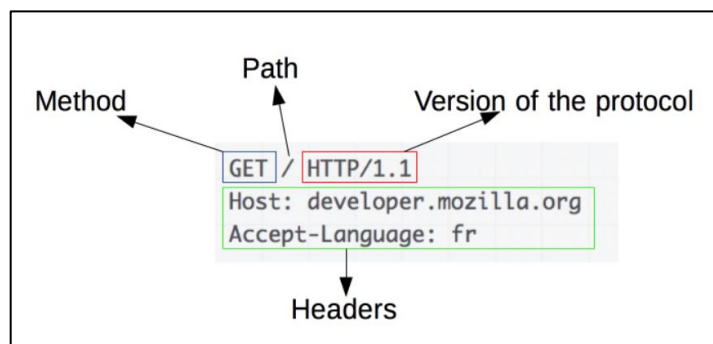


Figura 18: Exemplo de pedido HTTP [Fonte: (MDN Web Docs, 2021)]

A resposta é feita do servidor para o cliente. O principal objetivo da resposta é fornecer ao cliente o recurso que ele solicitou, ou informar se ocorreu algum erro. Uma resposta HTTP contém os seguintes elementos:

- A linha de status
- O cabeçalho HTTP
- Uma mensagem no corpo, se necessário

A linha de *status* é a primeira linha que aparece na mensagem de resposta e contém três elementos:

- A versão HTTP utilizada
- O *status code* que indica o resultado do pedido
- Uma mensagem de *status*, indica se foi *OK* ou erro.

O *status code* tal como foi referido identifica o resultado do pedido e pode assumir os seguintes valores:

- 200 – 299 representa uma resposta bem-sucedida
- 300 – 399 representam um redireccionamento
- 400 – 499 são erros do lado do cliente
- 500 - 599 são erros do lado do servidor

O corpo da mensagem é utilizado para a maioria das respostas porque o recurso pedido vem nesta secção. Quando o cliente apenas pede ao servidor os cabeçalhos, faz um pedido através com o método *HEAD*. Na Figura 19, é possível visualizar o exemplo de uma resposta *HTTP*. (IBM, 2019)

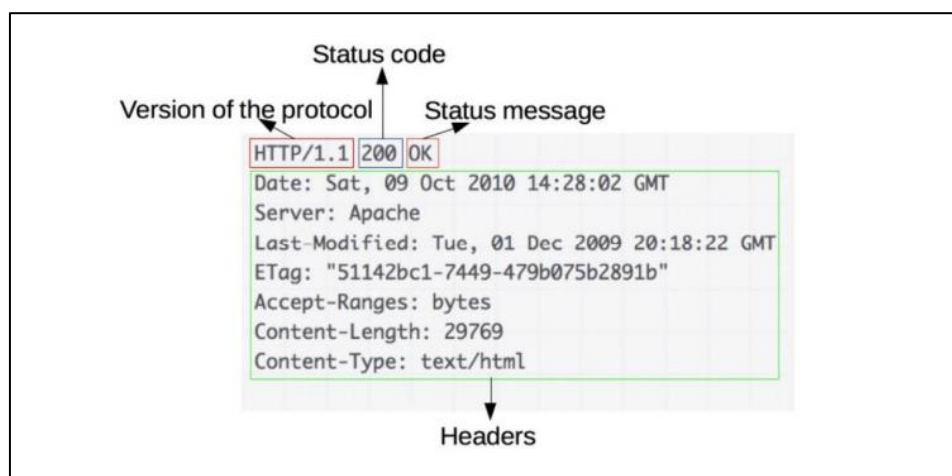


Figura 19: Exemplo de uma resposta HTTP [Fonte: (MDN Web Docs, 2021)]

Os métodos HTTP são responsáveis por indicar que tipo de ação é que se pretende efetuar para um dado recurso, sendo os mais utilizados (MDN Web Docs, 2021):

- *GET*: Pedidos com o método *GET* deve ser retornado o conteúdo solicitado por parte do cliente em que os dados enviados para o servidor são colocados na URL na forma de *query string*.
- *HEAD*: Pedido com o método *HEAD* solicita uma resposta idêntica ao método *GET*, contudo apenas será retornado o cabeçalho.
- *POST*: Este método é utilizado com o mesmo propósito do *GET*, contudo os dados enviados para o servidor serão colocados no corpo do pedido. Este método é utilizado para enviar dados de formulários *HTML*.
- *PUT*: O método *PUT* tem como objetivo substituir os dados do recurso de destino, pelos dados que seguem no pedido.
- *DELETE*: Serve para eliminar um recurso específico

No caso do “Login”, é efetuado um pedido *POST* para que seja enviado as credenciais para o servidor, para que elas sejam comparadas com os dados guardados na base de dados. [Figura 20]

```
115 @app.route("/Login", methods= [ "GET", "POST"])
116 def home():
117     if request.method == "POST":
118         user= request.form.get("email_login")
119         password= request.form.get("pass_login")
120
121         user_data = validar_user(user)
122         #pass_test = user_data[1]
123
124         if user_data:
125             pass_test = user_data[2]
126             if pass_test == password:
127                 print('Login com sucesso')
128                 session['user'] = user
129                 return redirect('/UserHome')
130             else:
131                 flash('Password incorreta')
132                 print('Password incorreta')
133                 #return render_template("Home.html")
134                 return redirect("/Home")
135         else:
136             flash('Esse user não existe')
137             print('Esse user não existe')
138         return redirect("/Login")
139
140     return render_template("Login.html")
141
```

Figura 20: Request *POST* ao servidor

No extrato de código da Figura 20, é enviado um pedido *POST* para o servidor, enviando as credenciais submetidas pelo utilizador, de modo a serem comparadas com os dados guardados e confirmar as credenciais. Para haver essa confirmação, foi criada a função “validar\_user”, que está presente na linha 121. Esta função procura o *user* inserido pelo utilizador, na base de dados e, caso o utilizador exista, retorna a linha inteira relativa aos dados do utilizador, incluindo o *user* e a sua *password*. Depois, a *password* retornada pela base de dados é comparada com a *password* inserida pelo utilizador na interface gráfica. [Figura 21]

```
260 def validar_user(username):
261
262     conn = sqlite3.connect('basedados.db')
263     c = conn.cursor()
264     print("Connected to SQLite")
265     try:
266
267         sql = "SELECT * FROM USER WHERE email = '{}'.format(username)"
268         c.execute(sql) #executar o comando
269         dados = c.fetchone()
270         conn.commit()
271         conn.close()
272
273     except sqlite3.Error as error:
274         print("Failed to insert data into sqlite table", error)
275     finally:
276         if conn:
277             conn.close()
278             print("The sqlite connection is closed")
279
280     return(dados)
```

Figura 21: Interação com a base de dados: função "validar\_user"

Caso não seja nada retornado pela função “validar\_user”, significa que não existe o *username* inserido pelo utilizador, aparecendo o aviso “Esse *user* não existe” Caso a *password* seja diferente da *password* retorna pela função “validar\_user”, significa que a *password* inserida pelo utilizador está errada, aparecendo o aviso “Password incorreta”. Em último caso, ou seja, se estiver tudo correto, o utilizador é redirecionado para a rota “/UserHome”, onde pode aceder a todas as funcionalidades disponíveis.



## 5.4. EXTRAÇÃO E UTILIZAÇÃO DOS METADADOS

Para a extração e utilização dos metadados, as imagens submetidas passam por um processo com várias etapas, como é demonstrado na Figura 22.

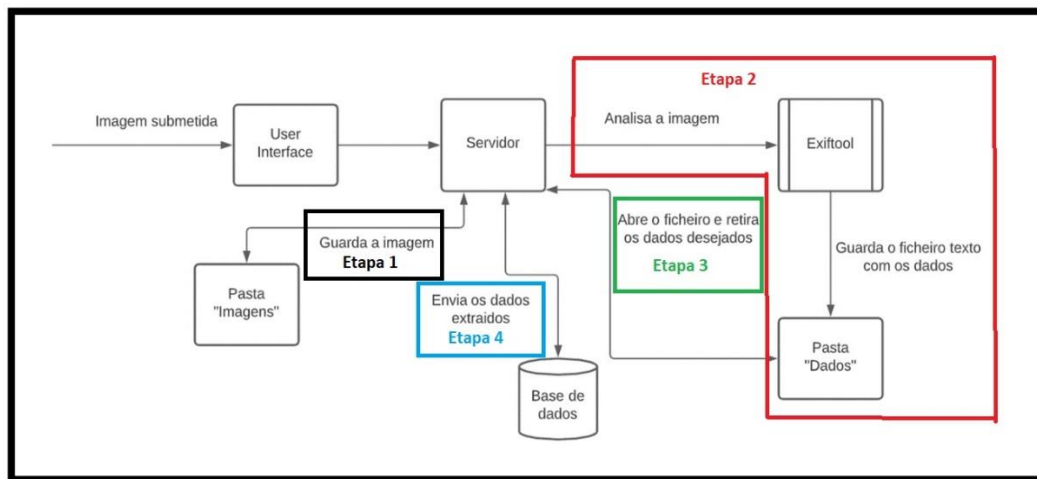


Figura 22: Esquema de extração e armazenamento dos metadados

### 5.4.1. EXECUÇÃO E ANÁLISE DAS IMAGENS

Para a análise e extração dos dados das imagens submetidas foi integrado o software “Exiftool” (Harvey, 2021). Trata-se de um programa de software gratuito e de código aberto para ler e editar metadados de ficheiros de imagem, áudio, vídeo e PDF.

Após a imagem ser submetida, vai ser guardada numa pasta, donde vai ser analisada através do código presente no servidor. No código, é utilizado um módulo chamado “OS”, que dá a possibilidade de interagir com ferramentas do sistema operativo de forma independente, ou seja, através apenas do código presente do servidor. Neste caso, este modulo é utilizado para a execução da ferramenta “Exiftool”. O módulo permite a chamada da linha de comandos, onde vai ser executado a ferramenta para a análise das imagens submetidas. Durante a sua execução, é utilizado uma ferramenta disponibilizada pelo software “Exiftool”, que dá a possibilidade de guardar todos os dados extraídos da imagem submetida num ficheiro texto com o mesmo nome da imagem analisada. Para isso, o nome do executável tem de ser alterado de “exiftool(-k).exe” (nome original) para “exiftool (-a -u -g1 -w txt).exe”. [Figura 23]

Executable Name	Operation
exiftool(-k).exe	Print meta information in window and pause before terminating.
exiftool(-k -a -u -g1 -w txt).exe	Generate output ".txt" files with detailed meta information.

Figura 23: Nomes do executável "exiftool" e suas funcionalidades [Fonte: (Harvey, 2021)]

Após a criação do ficheiro de texto, este ficheiro tem de ser analisado através do código presente no servidor, recolhendo os dados desejados e enviando-os para a base de dados, em que estão armazenados toda a informação das imagens submetidas.

#### 5.4.2. ANÁLISE DO FICHEIRO DOS DADOS

Após a abertura do ficheiro com os dados extraídos na etapa 2, a procura dos dados desejados será feita através do nome da *tag* atribuída a cada informação guardada na imagem. Para isso, é utilizado o método “*.startswith()*”, que procura uma linha que começa com uma palavra/frase desejada, que neste caso, são os nomes das *tags* dos dados desejados.[Figura 24]

```
#Make
if linha_break.startswith("Make") :
    Testemarca = True
    #print(linha_break)
    registro = linha_break.split()
    #print(registro)
    Make = registro[2]
    print (Make)
```

Figura 24: Excerto do código de análise do ficheiro dos dados da imagem

Desta forma, todas as *tags* vão ser pesquisadas no ficheiro com os dados extraídos, sendo cada um destes guardados numa variável. Este processo é repetido para cada um dos dados pretendidos, guardando todas as variáveis num dicionário criado. Um dicionário consiste numa coleção de pares de valores-chave. Cada par de valores-chave mapeia a chave para o seu valor associado. (Sturtz, s.d.) No dicionário criado, vão ser criados pares de valores-chaves, em que cada par guarda uma informação extraída do ficheiro analisado, como é demonstrado na Figura 25.

```
dic={
    "Make" : Make,
    "Model" : Model,
    "FileType" : FileType,
    "Res" : Resolution,
    "Data" : Data,
    "Time" : Time,
    "Comentário" : Comentario[0],
    "Latitude" : Latitude,
    "Longitude" : Longitude
}
#print(dic)
return dic
```

Figura 25: Dicionário criado para o armazenamento dos dados extraídos

Após a extração dos dados, o dicionário vai ser utilizado para enviar os dados extraídos para a base de dados para serem armazenados e organizados nas tabelas criadas.

#### 5.4.3. INTERAÇÃO COM A BASE DE DADOS

Como já foi referido, de forma a poupar espaço e otimizar o funcionamento da base de dados para cada imagem submetida, a informação extraída irá ser verificada e comparada com os dados já guardados em cada tabela específica. Para que esta verificação seja feita, sempre que o servidor receba dados novos, compara-os aos dados armazenados em cada tabela específica. Se os dados recebidos forem novos, irá ser criada uma linha nova e, consequentemente um *id* novo, com a informação nova. Esse *id* irá ser, depois, utilizado na tabela principal para relacionar o tópico com a imagem submetida. Caso contrário, a informação irá ser dispensada, utilizando assim, só o *id* da informação já presente da tabela em específico. [Figura 26]

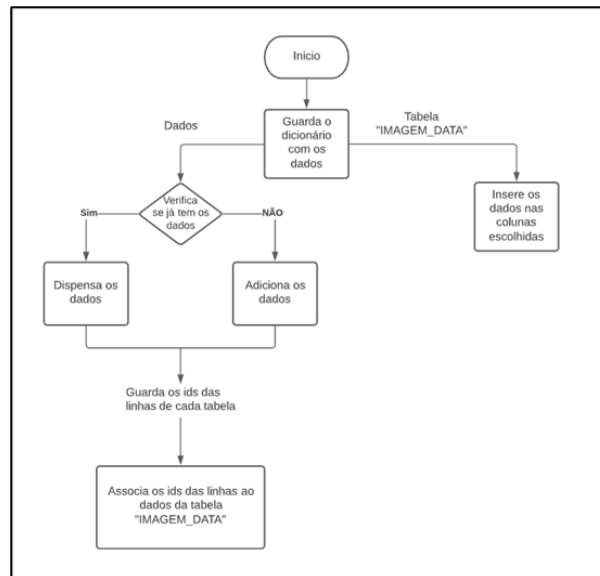


Figura 26: Interação com a base de dados

Para iniciar a interação, terá de ser feita a conexão da base com o servidor, tendo sido utilizado o módulo “sqlite3”, e a função “sqlite.connect (“nome da tabela”)”. [Figura 27]

```

1 import sqlite3
2 from sqlite3 import Error
3
4 conn = sqlite3.connect("basedados.db")
5 c = conn.cursor()
6
7 sql= ''' INSERT INTO nome_tabela (coluna1 ,coluna2) VALUES ('{}','{}') '''.format( val1, val2)
8
9 c.execute(sql)
10 conn.commit()
11 conn.close()
12

```

Figura 27: Conexão do servidor á base

## 5.5. GRAFOS

Para complementar a experiência de visualização e interação do utilizador com as imagens, foi adicionado um elemento gráfico de modo que seja possível uma melhor compreensão e acessibilidade de navegação entre as imagens. Este elemento vai conter uma importância crescente com o aumento dos dados na plataforma, que pode prejudicar a experiência do utilizador, oferecendo ao utilizador, um melhor ponto de vista para com toda a informação disponibilizada na interface. Este elemento gráfico é chamado de grafo e, através dele, irá ser possível ver as relações entre as várias imagens, quais é que se relacionam mais ou menos, entre outros fatores. Para a criação e implementação destes gráficos na interface web foi utilizado da biblioteca D3.js (Bostock, 2021), pois permite a

manipulação de ficheiros em formato de dados, interagindo com o web browser de forma a apresentar o conteúdo visualmente num formato interativo e dinâmico

### 5.5.1. CONTEXTO

Os diagramas de rede (também chamados grafos) são diagramas que mostram interconexões entre um conjunto de entidades, simbolizando que estão relacionados de alguma forma entre elas. Cada entidade é representada por um nó (ou vértice). As ligações entre nós são representadas através de ligações (ou bordas). [Figura 28]

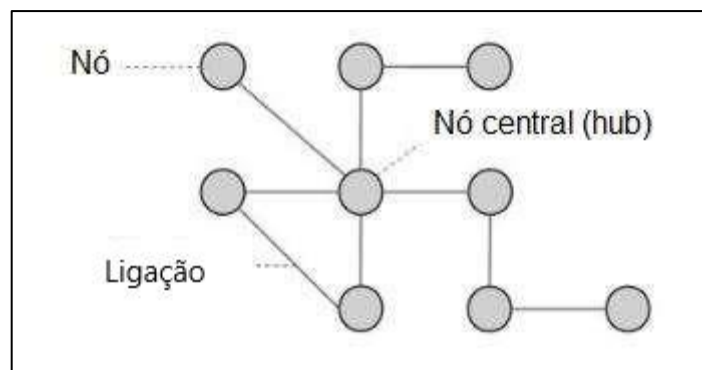


Figura 28: Estrutura gráfica de um grafo [Fonte: (Denise Cristiane Santos, 2016), editada]

Neste projeto, as entidades (ou nós) irão ser imagens, havendo um nó central que será a imagem selecionada, e os restantes nós, as imagens que estarão relacionadas com a anterior. Depois, dependendo da força da ligação entre o nó central e cada um dos nós, a ligação entre ambos irá ser mais fina ou mais espessa, sendo mais fraca ou mais forte, respetivamente. [Figura 29]

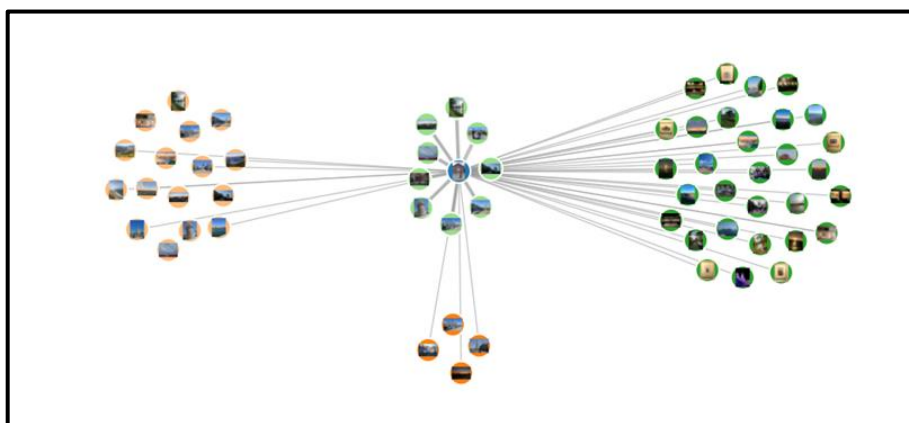


Figura 29: Grafo de exemplo das relações entre as imagens

### 5.5.2. D3.JS

Para a criação e apresentação destes diagramas na interface web, foi utilizado uma ferramenta chamada D3.js, por causa das funcionalidades no processo de criação, edição e implementação de qualquer tipo de diagrama numa interface web. D3 (Data-Driven Documents ou D3.js) é uma biblioteca JavaScript para visualização de dados usando padrões web. (Bostock, 2021)

### 5.5.3. IMPLEMENTAÇÃO E INTEGRAÇÃO NA INTERFACE GRÁFICA

#### 5.5.3.1. Requisitos de integração

A utilização da biblioteca D3.js, mais propriamente dos grafos, neste projeto consiste na apresentação das relações existentes entre as várias imagens submetidas. De modo que isto seja possível, foi necessário ter acesso à informação necessária e de forma organizada para uma correta interpretação da biblioteca utilizada. Para isso foi preciso efetuar alguma pesquisa de código para que fosse possível criar a estrutura, adicionar os dados e guardar o ficheiro de forma correta e apta para a respetiva utilização. Para realização deste requisito, foi necessário a criação de um ficheiro de formato JSON. Este ficheiro JSON foi estruturado de uma forma específica [Figura 30], tendo sido utilizado código para a respetiva criação e organização do documento.

```
{
  "links": [
    {
      "source": Nó de início,
      "target": Nó de destino,
      "value": valor referente á força da ligação
    }
  ],
  {
    "nodes": [
      {
        "group": Grupo em que o nó vai estar presente,
        "id": Nome do nó
      }
    ]
  }
}
```

Figura 30: Estrutura do ficheiro JSON

A estrutura do documento está dividida em 2 partes: “Nodes” e “Links”. Cada parte irá representar uma lista, sendo apresentado cada elemento da respetiva lista. Cada elemento irá conter um dicionário com a informação necessária para a implementação do grafo. Na primeira lista, “Nodes”, cada dicionário irá conter a informação de cada nó que irá ser criado, sendo a informação: [Tabela 2]

Tabela 2: Ficheiro JSON: Elemento "Nodes"

<b>Id:</b>	O nome/numero que irá identificar cada nó
<b>Group</b>	O grupo em que cada nó irá pertencer, normalmente identificados por um número.

Na segunda lista, “Links”, cada dicionário irá conter a informação necessária para efetuar as ligações entre os nós, sendo a informação: [Tabela 3]

Tabela 3: Ficheiro JSON: Elemento “Links”

<b>Source</b>	Id do nó, a partir do qual vai ser efetuada a ligação
<b>Target</b>	Id do nó, a qual vai ser efetuada a ligação
<b>Value</b>	Intensidade da ligação, que será refletida na espessura da ligação entre os nós

Para a abertura e interpretação do ficheiro JSON, foi utilizada a função `d3.json(parâmetro1, parâmetro2)`, onde irá conter 2 parâmetros:

- **Parâmetro 1:** Localização do ficheiro json com os dados necessários
- **Parâmetro 2:** Função onde irá ser criado e implementado todos os elementos do grafo

```

114
115 d3.json("/static/json/file.json", function(error, graph) {
116     if (error) throw error;
117
118     // CRIAR "LINKS"
119     var link = svg.selectAll("line")
120         .data(graph.links)
121         .enter().append("line")
122
123     link
124         .attr("class", "link")
125         .attr("stroke-width", function(d) { return Math.sqrt(d.value); });
126

```

Figura 31: Excerto da utilização "d3.json()"

#### 5.5.3.2. Criação e implementação dos grafos

Após a abertura do ficheiro JSON com os dados necessários, foi criada a estrutura do grafo, definindo as características dos nós e as ligações e as respetivas interações visuais. Para isso, foi criado um elemento <svg> na página html, que vai conter o grafo e todos os elementos relacionados. No elemento <svg> estão definidas as dimensões desejadas para o grafo. Após a criação do elemento <svg> e da sua customização, irão ser criados os elementos “nodes” e “links” que estarão interligados ao elemento <svg>. Cada um destes elementos foi customizado de acordo com o problema proposto, de forma que as características de cada elemento são as seguintes:

Links:

- **Classe “link”**: Classe que contem alguns parâmetros de customização pré-definidos no início do documento
- **Parâmetro “stroke-width”**: Parâmetro que irá definir a espessura da ligação, consoante a variável “value” presente no ficheiro JSON

Nodes:

- **Classe “nodes”**: Classe que contem alguns parâmetros de customização pré-definidos no início do documento
- **Classe “circle”**: Classe que adiciona a forma circular aos nós, e contem alguns parâmetros para customização da figura circular
- **Classe “image”**: Classe que adiciona as imagens no nó ao qual estas estão associadas, e contem alguns parâmetros para customização do posicionamento da imagem no nó



Após a criação de cada elemento, foi feito a atribuição dos dados do ficheiro JSON aos respetivos elementos do grafo, para que sejam feitas as ligações pretendidas entre os elementos. Para efetuar essas atribuições, é utilizado a função “.data()”, inserindo no parâmetro, a variável “graph” seguido do nome da lista que quer ler do ficheiro JSON, consoante o elemento em que está a ser inserido, como está apresentado na Figura 32.

```
// CRIAR "LINKS"
var link = svg.selectAll(".link")
    .data(graph.links)
    .enter().append("line")

// CRIAR "NODES"
var node = svg.selectAll(".nodes")
    .data(graph.nodes)
    .enter().append('g')
```

Figura 32: Extração dos dados do ficheiro JSON

Para uma melhor interação do utilizador, foram adicionados alguns parâmetros ao grafo, sendo eles:

- **Zoom:** É utilizado a função d3.zoom(), adicionando, de seguida, alguns parâmetros para customizar a interação do zoom na interface web
- **Legenda:** É utilizado a função d3.legendColor(), para relacionar os grupos criados, às cores utilizadas para identificar cada grupo, e personalizar a secção onde irá aparecer a legenda( Título, Tamanho que vai ocupar ... ). É utilizado também o d3.scaleOrdinal(), para identificar cada grupo criado, atribuindo um nome para cada um dos grupos. Cada nome irá corresponder á relação que esse grupo tem com a imagem central.
- **Parâmetro “. on (click)”:** Função para definir uma interação específica com os nós, que neste caso, é no momento em que o nó é selecionado com o rato.
  - Através deste parâmetro, quando a interação é feita, irá abrir uma janela com a imagem associada ao nó, apresentando uma tabela com os respetivos metadados

- **Parâmetro “. on (mouseover)”**: Função para definir uma interação específica com os nós, que neste caso, é no momento que o rato passar por cima do nó.
  - Através deste parâmetro, quando a interação é feita, irá aparecer uma nota com a indicação da relação que o nó selecionado tem com o nó central.

## 6. TESTES E RESULTADOS

Para a testar o funcionamento do sistema apresentado no relatório, foram realizados 3 testes:

1. Determinar e avaliar o tempo do processo de upload da imagem e da análise, extração e armazenamento dos dados obtidos pelo servidor
2. Após a escolha de uma imagem, aceder ao grafo relativo á imagem selecionada e verificar se as relações estabelecidas estão corretas, comparando os metadados de cada imagem.
3. Após a criação dos grafos, demonstrar todas as interações possíveis com os elementos presentes.

Para o primeiro teste, o utilizador vai precisar de criar uma conta e de conectar-se à mesma, para conseguir realizar o upload de imagens para o sistema. Após cumprir estes requisitos, já vai ser possível realizar o teste proposto. Após se conectar á sua conta, o utilizador terá de ir para a página de “/Userupload”, e preencher os campos necessários para realizar o upload da imagem pretendida. Quando o upload for feito, um cronometro irá medir o tempo necessário para todo o processo de análise, extração e armazenamento da imagem e dos dados na base de dados, obtendo, assim, o tempo pretendido para a respetiva avaliação. O cronometro irá ser exibido na consola do servidor, apresentando também o momento de começo e fim da contagem, seguido do resultado obtido, como é demonstrado na Figura 33.

```
Começou
Connected to SQLite
The sqlite connection is closed
3
IMG_7012.JPG
1 output files created
./static/img/imagens/IMG_7012.JPG
Connected to SQLite
Esta marca já existe
Esta resolução já existe
Este formato já existe
INSERT INTO IMAGEM_DATA ( Marca_ID, Resolucao_ID , Formatos_ID , Hora, Data, GPS_Latitude, GPS_Longitude, User_Comment
) VALUES("2" , "23" , "3" , "11:39:59.49" , "2019:09:04" , " " , " " , "Campo de concentração, polonia")
Dados inseridos na base de dados com sucesso
The sqlite connection is closed
Imagem guardada com sucesso
Acabou
Time Lapsed = 0:0:0.5774681568145752
```

Figura 33: Processo de upload de uma imagem e a sua cronometragem

O resultado do teste foi positivo, considerando que o intervalo de confiança esperado seria um tempo dentro dos milissegundos. O tempo obtido foi 0.577 segundos, concluindo que o tempo obtido está dentro do intervalo desejado.

Para o segundo teste, o utilizador será de seleccionar uma imagem na “/UserHome”, para, de seguida, utilizar a funcionalidade dos grafos, de modo que se possível a realização do teste proposto. Após a interação com a imagem e solicitação do grafo, este será apresentado com todas as imagens que, de alguma forma, estarão relacionadas com a imagem seleccionada. Para testar a veracidade das ligações criadas entre os diferentes nós do grafo, vão ser escolhidas 3 imagens para comparar os seus metadados com os metadados da imagem central, e verificar se as ligações do grafo estão corretas. A Figura 34 contém a imagem que se encontra no nó central e os respetivos metadados.



Figura 34: Imagem presente no nó central do grafo e os seus metadados

De forma a demonstrar a veracidade das ligações vão ser demonstrados os metadados de cada uma das imagens-teste, sendo o primeiro tópico comparado vai ser o utilizador. A Figura 35 contém uma imagem que foi retirada do grupo “Utilizador”

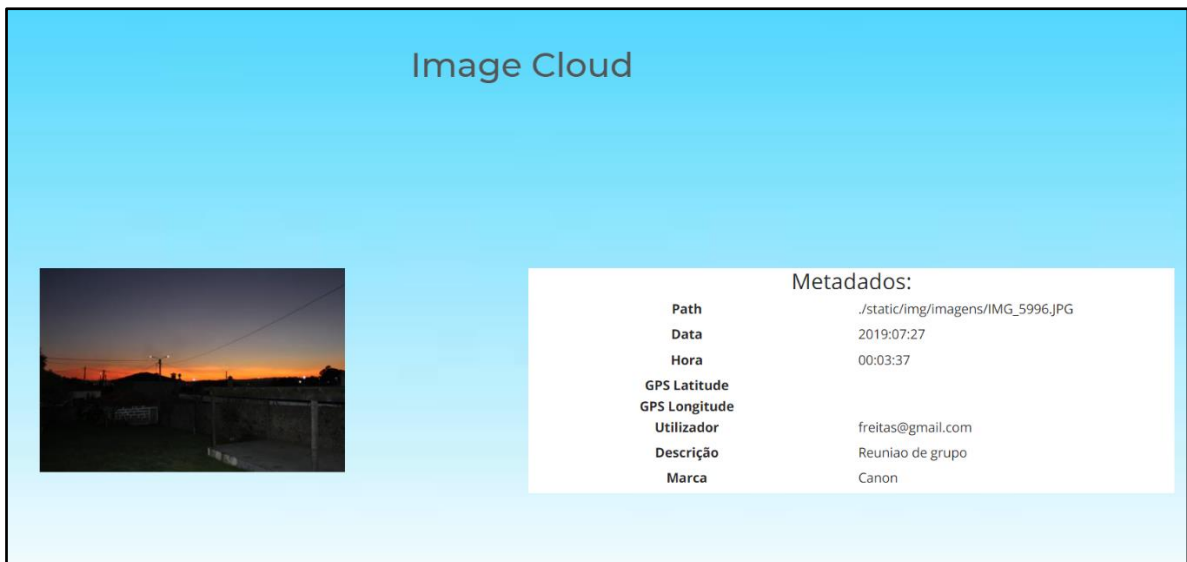


Figura 35: Imagem de teste: Grupo “Utilizador”

Como se pode verificar na Figura 35, a conta utilizada para fazer o upload das duas imagens foi a mesma, provando a correta ligação entre os nós. O segundo tópico avaliado foi a hora de criação da foto. Este tópico foi utilizado com uma condição pré-definida, de forma que fosse possível comparar, de alguma forma, todas as imagens. Essa condição consiste na predefinição de um intervalo de tempo em relação á imagem selecionada, sendo esse intervalo de 3 horas, antes e depois da hora utilizada como referência. Deste modo, os testes realizados foram efetuados com esta condição definida. A Figura 36 contém uma imagem retirada do grupo “Hora”.

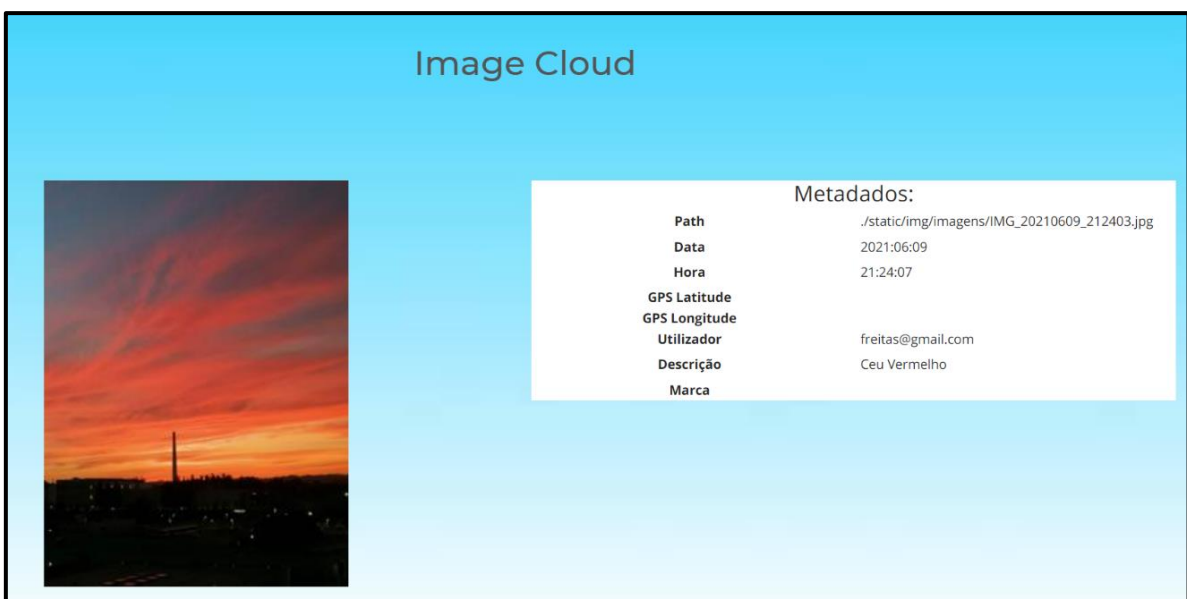


Figura 36: Imagem de teste: Grupo “Hora”

Através dos metadados apresentados na Figura 36, foi possível verificar que o nó correspondente á imagem presente na figura está corretamente ligado nó central, estando dentro do intervalo de confiança estabelecido. Por último, o terceiro tópico avaliado foi a marca do dispositivo em que a imagem foi tirada. A Figura 37 contém uma imagem retirada do grupo “Marca”.

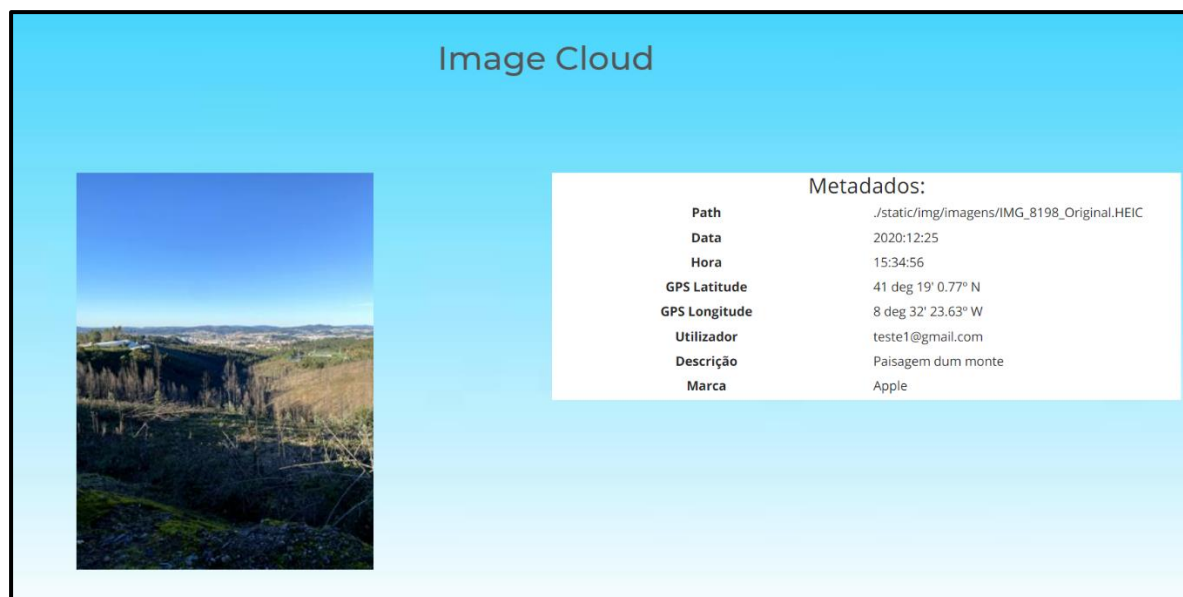


Figura 37: Imagem de teste: Grupo "Marca do dispositivo"

Mais uma vez, os metadados da imagem apresentada na Figura 37 comprovam a correta ligação feita entre o nó da imagem de teste e o nó central do grafo. Concluindo, através dos testes apresentados, comprova-se a veracidade e exatidão das ligações estabelecidas entre os nós do grafo.

No terceiro teste, irá ser demonstrado as interações possíveis num grafo, demonstrando a respetiva interação. A primeira interação será o *drag*, ou seja, quando o utilizador arrasta qualquer nó presente no grafo. [Figura 38]

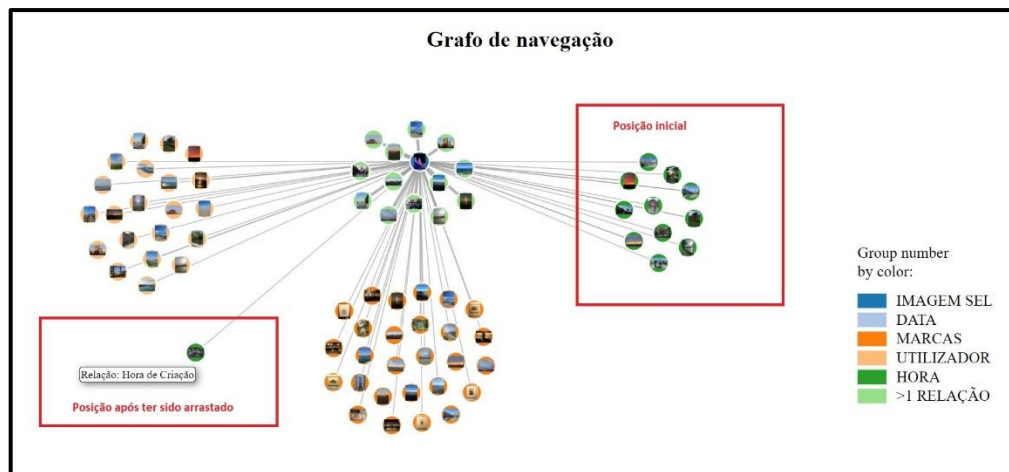


Figura 38: Interação de *drag*, alterando o nó de posição

A segunda interação será o *zoom*, ou seja, o utilizador pode ampliar os elementos do grafo.

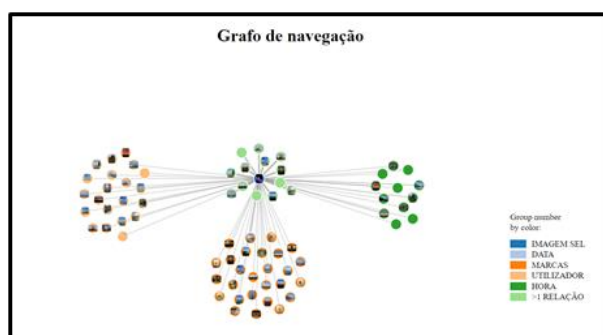


Figura 39: Grafo com o zoom inicial

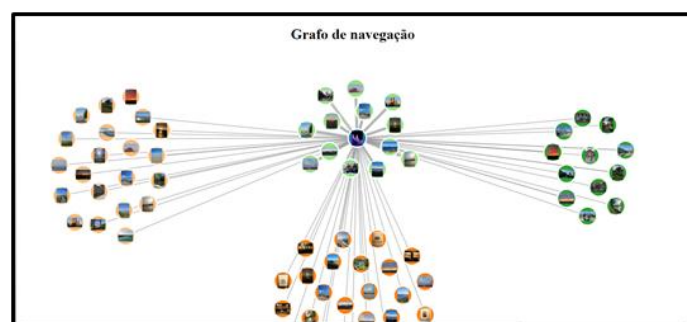


Figura 40: Grafo ampliado

Na Figura 39, está presente o grafo original que é apresentado na interface, estando com o zoom inicialmente escolhido. Na Figura 40, está presente o mesmo grafo, mas que foi ampliado, de forma a poder ver-se melhor as imagens nos nós e as ligações entre estes. A terceira interação será o *mouseover*, ou seja, quando o utilizador passa o rato por cima de um nó, aparecerá uma nota com a relação do respetivo nó com o nó central. [Figura 41]

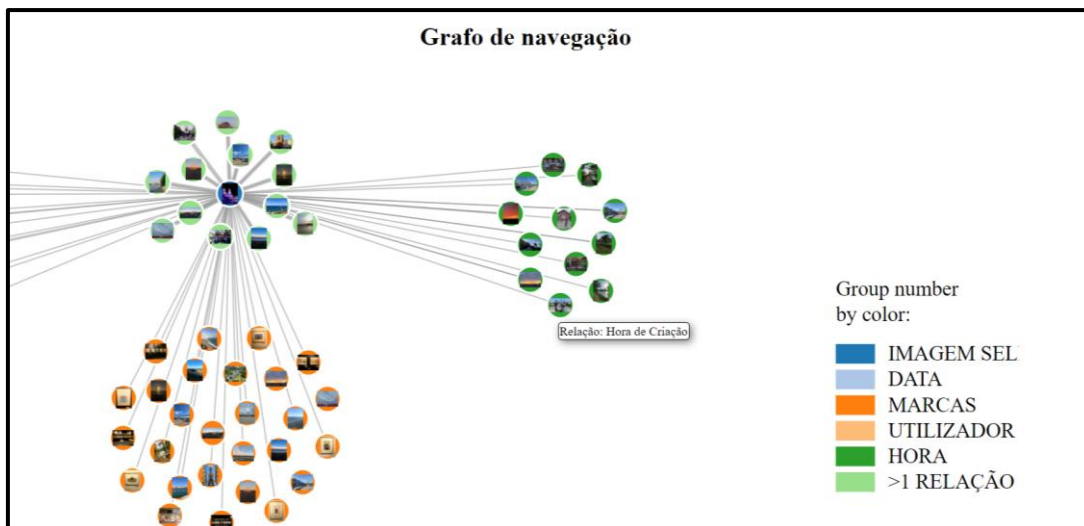


Figura 41: Interação do *mouseover* e a respetiva relação do nó selecionado

Na Figura 41, é apresentado a relação entre o nó selecionado e o nó central, como é provado através da legenda do grafo. Todas as imagens de cada grupo irão apresentar a respetiva relação com o nó central. A última interação será o *mouseclick*, ou seja, quando o utilizador carrega num nó, abrirá um separador novo, onde aparecerá a imagem correspondente ao nó e os respetivos metadados da imagem. [Figura 42]



Figura 42: Separador aberto através da interação *mouseclick*

Na Figura 42, está apresentado um exemplo do separador aberto, após o nó correspondente a esta imagem ter sido selecionado pelo utilizador.



# 7. CONCLUSÕES

O principal objetivo deste projeto era abordar uma tecnologia pouco conhecida como os metadados presentes nas imagens, interligando-a com toda a tecnologia de gestão de base de dados e programação de interfaces gráficas, associando-as de uma forma acessível e cativante para o utilizador. A implementação do projeto baseia-se em duas secções: Servidor e interface gráfica.

Na secção do servidor, para a implementação dos metadados, foram feitos vários testes para encontrar qual seria o melhor método para a análise e extração dos dados presentes nas imagens analisadas, tendo apenas havido algumas dificuldades na escolha dos dados necessários para o projeto. Após esta escolha, o processo de extração e envio dos dados para a base de dados comprovou-se adequado para o problema. Na implementação da base de dados, a organização das tabelas provou-se útil e correta, pois facilitou o armazenamento e organização dos dados, evitando o armazenamento de informação repetida e tornando a pesquisa mais rápida e precisa.

Por fim, na secção da interface gráfica, o objetivo era proporcionar uma plataforma de navegação e gestão de imagens acessível ao utilizador, de forma a que permitisse 2 funcionalidades principais: O upload de imagens e dos respetivos parâmetros do formulário solicitado para submeter a imagem. Nesta funcionalidade, existe melhorias que podem ser realizadas no futuro, como a possibilidade de submeter mais do que uma imagem ao mesmo tempo, havendo alterações a fazer no armazenamento e análise das respetivas imagens. A segunda funcionalidade é a navegação pelas imagens submetidas na plataforma, disponibilizando, ao utilizador, funcionalidades como a filtragem das imagens através de tópicos relevantes, como a hora e a data da criação da imagem, podendo ser expandido para mais tópicos futuramente, a visualização das relações entre imagens, através de grafos para que o utilizador possa navegar e interagir com as imagens de uma forma mais acessível e intrínseca, melhorando a experiência visual da plataforma. Este elemento gráfico pode também ser melhorado futuramente, adicionando mais interações possíveis e tornando ainda mais acessível e apelativo para o utilizador.

Concluindo, o projeto realizado funcionou de forma correta, cumprindo todos os requisitos e objetivos inicialmente propostos, embora houvesse aspetos que poderiam ser melhorados futuramente.

## *Referências Documentais*

- AprendIS. (24 de Janeiro de 2016). *AprendIS*. Obtido em 14 de Agosto de 2021, de [http://aprendis.gim.med.up.pt/index.php/Bases\\_de\\_Dados](http://aprendis.gim.med.up.pt/index.php/Bases_de_Dados)
- Bostock, M. (2021). *D3.js*. Obtido de <https://d3js.org>
- Cagle, K. (26 de Feb de 2019). Obtido de Forbes: <https://www.forbes.com/sites/cognitiveworld/2019/02/26/the-value-of-metadata/?sh=1836deff6d30>
- Db-Engines*. (Agosto de 2021). Obtido em 31 de Agosto de 2021, de <https://db-engines.com/en/ranking>
- Denise Cristiane Santos, J. S. (Abril de 2016). *Researchgate*. Obtido em 31 de Agosto de 2021, de [https://www.researchgate.net/publication/305896242\\_UTILIZACAO\\_DE\\_MEDIDAS\\_DE\\_CENTRALIDADE\\_NA\\_ANALISE\\_DE\\_REDES\\_SOCIAIS\\_ONLINE\\_CONTEXTUALIZADA\\_NA\\_INTERATIVIDADE\\_ENTRE\\_MARCAS\\_E\\_CONSUMIDORES](https://www.researchgate.net/publication/305896242_UTILIZACAO_DE_MEDIDAS_DE_CENTRALIDADE_NA_ANALISE_DE_REDES_SOCIAIS_ONLINE_CONTEXTUALIZADA_NA_INTERATIVIDADE_ENTRE_MARCAS_E_CONSUMIDORES)
- Dietmar Wueller, R. F. (November de 2007). *Image Engineering*. Obtido em 31 de Agosto de 2021, de [https://www.image-engineering.de/content/library/conference\\_papers/before\\_2009/statistic\\_analysis\\_millions\\_photos.pdf](https://www.image-engineering.de/content/library/conference_papers/before_2009/statistic_analysis_millions_photos.pdf)
- Farias, P. (24 de Maio de 2021). *Tudocelular.com*. Obtido em 31 de Agosto de 2021, de <https://www.tudocelular.com/curiosidade/noticias/n174861/como-identificar-rostos-marcadores-google-fotos.html>
- Harvey, P. (12 de Agosto de 2021). *Exiftool*. Obtido em 31 de Agosto de 2021, de <https://exiftool.org/#running>
- IBM. (27 de Junho de 2019). Obtido em 03 de Setembro de 2021, de <https://www.ibm.com/docs/en/cics-ts/5.1?topic=concepts-http-protocol>
- MANSUROV, N. (24 de Abril de 2020). *protograpylife*. Obtido em 29 de Julho de 2020, de <https://photographylife.com/what-is-exif-data>
- MDN Web Docs. (2021). Obtido em 3 de Setembro de 2021, de <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>
- Pelski, S. (Junho de 2005). *Oracle*. Obtido de [https://docs.oracle.com/cd/B19306\\_01/appdev.102/b14302/ch\\_metadata.htm](https://docs.oracle.com/cd/B19306_01/appdev.102/b14302/ch_metadata.htm)
- Photometadata*. (2021). Obtido em 30 de Agosto de 2021, de Photometadata: <http://www.photometadata.org/META-101-metadata-types>

- Riley, J. (2017, Janeiro 01). *NISO*. Retrieved from [https://groups.niso.org/apps/group\\_public/download.php/17446/Understanding%20Metadata.pdf](https://groups.niso.org/apps/group_public/download.php/17446/Understanding%20Metadata.pdf)
- Singhal, A. (16 de Maio de 2012). Introducing the Knowledge Graph: things, not strings. Obtido de <https://blog.google/products/search/introducing-knowledge-graph-things-not/>
- Souza, I. d. (24 de Novembro de 2020). *Rockcontent*. Obtido de <https://rockcontent.com/br/blog/sqlite/>
- Sturtz, J. (s.d.). *Real Python*. Obtido em 01 de Setembro de 2021, de <https://realpython.com/python-dicts/>
- Tesic, J. (01 de Agosto de 2005). Metadata practices for consumer photos. *Metadata practices for consumer photos*, p. 92. Obtido em 19 de Agosto de 2021, de <https://ieeexplore.ieee.org/abstract/document/1490501>