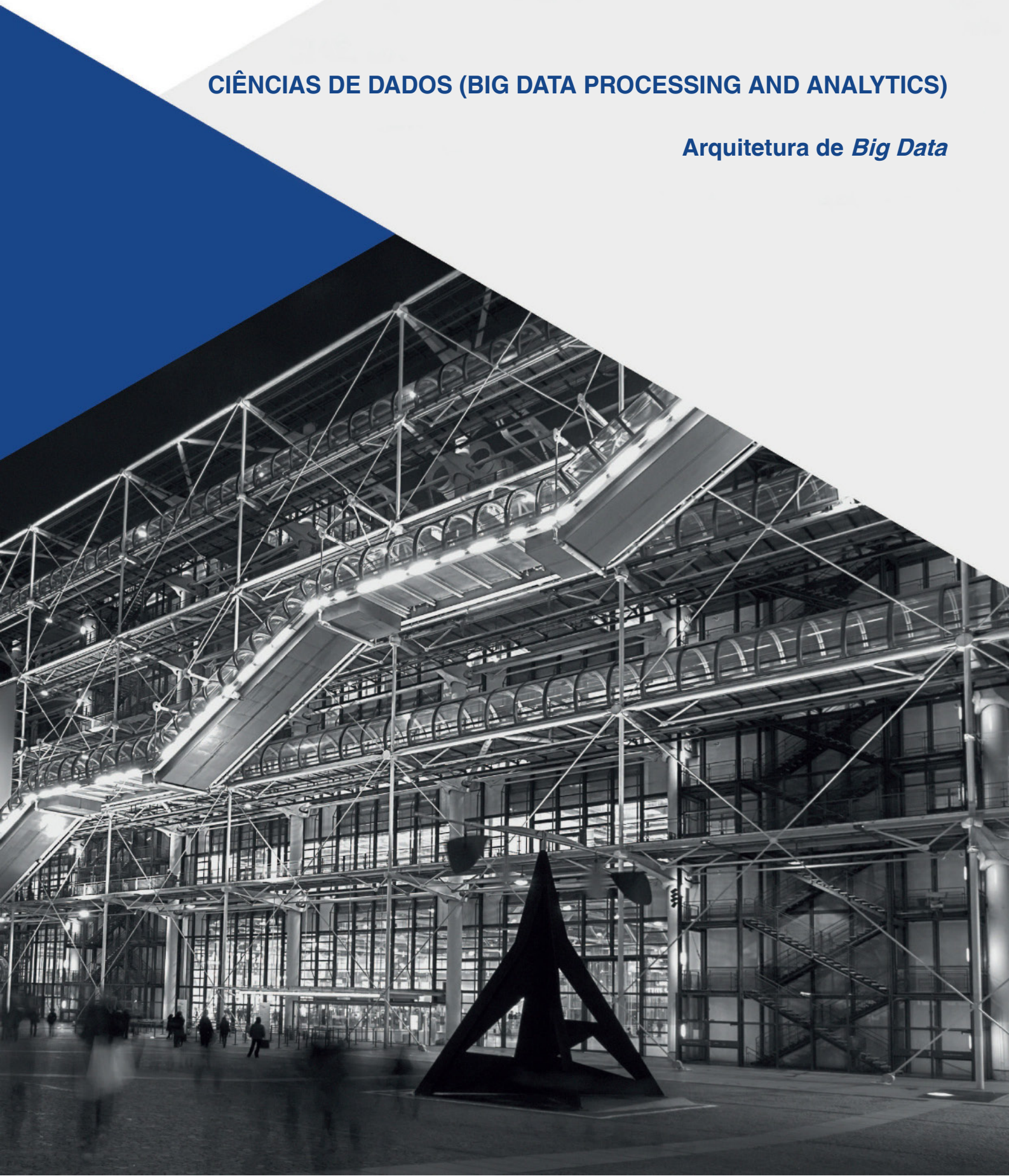
Universidade Presbiteriana Mackenzie

CIÊNCIAS DE DADOS (BIG DATA PROCESSING AND ANALYTICS)

Arquitetura de *Big Data*



Universidade Presbiteriana Mackenzie

Modalidade a distância

RECUPERAÇÃO DA INFORMAÇÃO NA WEB E EM REDES SOCIAIS

Trilha 2 – Recuperação de informação por raspagem: introdução

Professor: Luciano Moreira Camilo e Silva

Sumário

1. Introdução à Trilha 4

2. A Internet 5

2.1. O começo da World Wide Web 6

2.2. Os documentos de Hipertexto – HTML 8

2.3. A biblioteca Python Beautiful Soup 4.0 12

3. Exercício Prático 15

3.1. Conhecendo o conteúdo a ser raspado 15

3.2. Executando o código Python 17

4. Síntese 19

5. Referências 20

1. Introdução à Trilha

Na trilha anterior, estudamos como encontrar documentos a partir de consulta a sistemas de recuperação de informação e de que maneira esses sistemas evoluíram a tal ponto de se tornarem quase onipresentes na vida moderna, por meio dos sistemas de buscas como Google, Baidu ou DuckDuckGo.

Esta segunda trilha pretende ajudá-lo a dar o próximo passo. Uma vez encontrado o documento desejado, como retirar informações dele?

Um ser humano alfabetizado provavelmente conseguirá encontrar a informação desejada com alguma facilidade. A despeito das opções de design gráfico – como escolha de cores, tamanho da fonte, bem como estilo literário e idioma da escrita –, somos treinados nossa vida inteira para encontrar esses dados.

Como fazer uma máquina ler? Como organizar as informações de outra maneira, mais aprazível para o consumo que você pretende fazer? Ao final dessa trilha, você estará apto a responder a todas essas perguntas.

Como na trilha anterior, esta iniciará contando trechos da história da internet, a invenção e a explosão da web na década de 1990.

Na sequência, será apresentada a linguagem que suporta quase todas as informações espalhadas nos aproximadamente dois bilhões de sites existentes na web em 2021 (TOTAL, 2021). É essencial entender a estrutura dos documentos para conseguir raspá-los.[1]

Como ferramental, será introduzida uma biblioteca *Python* chamada *Beautiful Soup*, bastante popular nas atividades de raspagem de dados. Ela possui uma barreira de entrada muito baixa e é ideal para pequenos trabalhos.

A trilha encerrará com um caso prático. Em um site popular, será raspado o endereço de todas as lojas e os organizaremos para serem lidos no MS Excel.

1 "*Data scraping* (do inglês, raspagem de dados) é uma técnica computacional na qual um programa extrai dados de saída legível somente para humanos, proveniente de um serviço ou aplicativo. Os dados extraídos geralmente são minerados e estruturados em um formato padrão como CSV, XML ou JSON." (Raspagem de dados, Wikipédia).

2. A Internet

Hoje, se você questionar a um cidadão médio o que é a internet, principalmente se for da geração Y, Z ou posterior, deve, provavelmente, ouvir que são os sites ou os aplicativos que operam majoritariamente sobre protocolo HTTP.[1]

A internet é maior que a web e começou muito antes, com os primeiros esboços de redes de computadores conectadas sendo escritos ainda no início da década de 1960. Essa história foi brilhantemente descrita, de uma forma muito gostosa de se ler, no artigo de um grupo de pesquisadores, "A brief history of the Internet" (LEINER et al., 1999).

Em um resumo rápido, pesquisadores de diversos laboratórios se juntaram à DARPA, na época, ARPA, para criar uma rede de computadores, a ARPANET.[2] Coisas de Guerra Fria! A ideia era construir uma rede de computadores que não dependesse de um único ponto central e que não encerrasse as comunicações caso um centro de processamento de dados fosse bombardeado.

Com o tempo, o governo americano permitiu que universidades do país e seus aliados se conectassem a esta rede e, posteriormente, também as empresas privadas.

Algo que incentivou o uso e a expansão da internet foram as trocas de documento. Diversos protocolos foram criados ao longo dos anos, seja para troca de mensagens em sistemas de newsgroup ou o e-mail.

Como curiosidade, vejamos alguns dos principais protocolos de troca de informação utilizados antes do advento da web.

- **File Transfer Protocol (FTP) e SSH File Transfer Protocol (SFTP):** um dos mais antigos protocolos da internet, originalmente publicado pela RFC 114 (BHUSHAN, 1971) e, desde então, dezenas de outras RFCs foram publicadas, modernizando o protocolo. Ao contrário de boa parte dos protocolos citados nessa lista, o FTP continua bastante popular, principalmente no caso de troca de arquivos entre computadores, em sua versão SFTP.

1 "O *Hypertext Transfer Protocol*, sigla HTTP (em português, Protocolo de Transferência de Hipertexto) é um protocolo de comunicação (na camada de aplicação segundo o Modelo OSI) utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Ele é a base para a comunicação de dados da World Wide Web." (Hypertext Transfer Protocol, *Wikipédia)*.

2 "*Advanced Research Projects Agency Network*, em português: Rede da Agência para Projetos de Pesquisa Avançada, a primeira rede de computadores, projetada em 1969 para fazer transmissões de dados militares sigilosos e interligação dos departamentos de pesquisa nos Estados Unidos, o ponto inicial da internet, a rede mundial de computadores." (Defense Advanced Research Projects Agency, *Wikipédia*).

• **Simple Mail Transfer Protocol (SMTP):** outro protocolo que sobreviveu ao tempo. Foi proposto inicialmente pela RFC 772 (POSTEL; SLUIZER, 1980)e modernizado ao longo dos anos. Concebido para melhorar a troca de mensagens eletrônicas que, na época, ocorria por troca de arquivos pelo protocolo FTP. Opera em conjunto com os protocoles de recepção de correio eletrônico POP (REYNOLDS, 1984) e IMAP (MELNIKOV; LEIBA, 2021).

• **Network News Transfer Protocol (NNTP):** protocolo aplicacional criado na década de 1980 e atualizado em 2006. Seu principal uso era para a troca de mensagens entre os usuários, principalmente da Usenet. É um protocolo que caiu em desuso pela popularização de serviços similares na Web. Para conhecer um pouco mais sobre esse protocolo que precede a web com forte influência em nossa cultura ocidental, leia este artigo em inglês, “What is Usenet? 5 things you didn’t know about it” (ATHOW, 2018).

• **Internet Relay Chat (IRC):** protocolo criado em 1988 e padronizado publicamente pela RFC 1459 (OIKARINEN; REED, 1993), para ser utilizado como sistema de bate-papo pela universidade de Oulu, na Finlândia. Foi bastante popular até o início dos anos 2000, quando as salas de bate-papo na web começaram a se popularizar. Para conhecer a história do IRC no Brasil, leia este artigo: “10 anos sem Brasnet: se você usou o mIRC, você está ficando velho” (GNIPPER, 2017)

• **Gopher:** protocolo aplicacional criado no mesmo momento do HTTP, no início dos anos 1990, também com o objetivo de trocar documentos por toda a internet. Era um modelo simples, não exigia conhecimento prévios e conseguia, a seu modo, criar uma teia de documentos interligados. Teve uma expansão rápida, superando, no início, o crescimento do protocolo concorrente World Wide Web (www). Porém, a capacidade de replicar as funcionalidades no Gopher nos navegadores do www, além de uma decisão da universidade de Minnesota de cobrar taxa de licenciamento para seu uso, afugentou os usuários (GIHRING, 2016). (Gihring, 2016)

2.1. O começo da World Wide Web

No início dos anos 1990, a internet já havia se popularizado entre os acadêmicos, e isso permitiu uma evolução gigantesca no ecossistema, como dito anteriormente.

Foi utilizando um dos grupos da Usenet que Tim Berners-Lee publicou, pela primeira vez, o que ele imaginava ser a web. Uma réplica da mensagem pode ser acessada por este

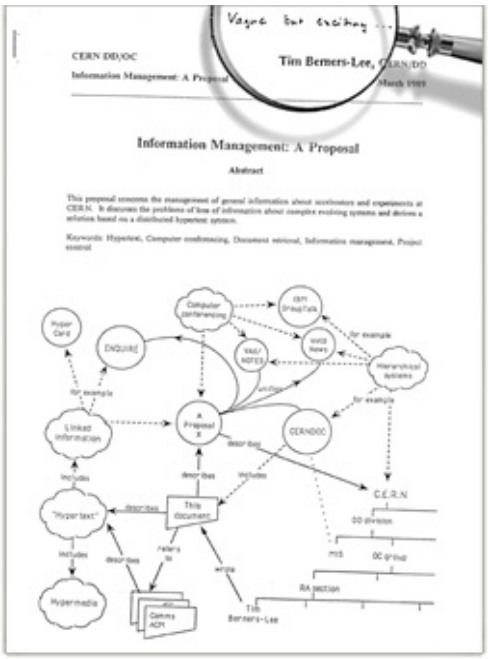
link: <https://www.w3.org/People/Berners-Lee/1991/08/art-6484.txt>.

A ideia de Berners-Lee era trabalhar com o conceito de hiperligações em documentos. Ou seja, alguns trechos do documento teriam ligações para outros documentos, criando, assim, uma teia. Levando em escala global, teríamos uma ampla teia mundial de documentos ou, em inglês, World Wide Web.

Para demonstrar sua ideia, o inventor criou várias ferramentas: o protocolo de transferência de hipertexto (HTTP), a linguagem de marcação de hipertexto (HTML, mais detalhes no item 2.2), o padrão de identificador universal de recursos (URI, em inglês) e o primeiro navegador, editor e servidor de documentos HTTP.

A história dessa invenção, considerada uma das maiores do século passado, pode ser lida em um livro do próprio criador da web, Weaving the Web: the original design and ultimate destiny of the World Wide Web (BERNERS-LEE, 2000).

Figura 1 – Proposta original de Tim Berners-Lee



Fonte: CERN (http://info.cern.ch/Proposal.html).

2.2. Os documentos de Hipertexto – HTML

HTML é o acrônimo, em inglês, para HyperText Markup Language, ou seja, uma Linguagem de Marcação para Hipertextos. Hipertextos são documentos que possuem hiperligações, ou seja, possuem ligações para outros documentos.

Um exemplo fácil de imaginar é uma agenda de contatos estruturada por uma coleção de documentos, na qual cada documento de pessoa possui uma tabela com nome, telefone e nome da empresa onde ela trabalha. O texto onde está escrito o nome da empresa possui uma ligação para um documento com todos os funcionários desta empresa, o qual, por sua vez, tem uma ligação para a página de cada funcionário.

A linguagem de marcação permite acrescentar informações para o programa que interpretará esse documento, porém, sem mostrar na tela do usuário esses comandos. Essas marcações permitem ao aplicativo alterar a formação de cores, tamanho e estilos de fontes, entre outras características. Adicionalmente, a marcação sinaliza características do texto marcado, tais como a identificação de um cabeçalho, um título, uma tabela ou um texto simples.

O HTML juntou os dois conceitos, permitindo a criação de documentos bastante elaborados, com títulos e subtítulos, corpo de texto, listas e tabelas, além, evidentemente, de permitir a ligação desse documento com qualquer outro documento na internet, pela identificação de seu URI.

Não é o objetivo deste e-book, porém, apresentar a totalidade de marcações existentes e sempre crescentes do HTML. Para isso, convido você a ler as matérias referenciadas da MDN, “JavaScript” e “HTML: Linguagem de Marcação de Hipertexto”.

Entretanto, para auxiliá-lo na raspagem de textos e familiarizá-lo com o assunto, apresentaremos as principais etiquetas e estrutura do HTML.

Inicialmente, é preciso compreender que um documento HTML é estruturado como uma árvore, chamada de Modelo de Documento por Objeto ou, em sua sigla em inglês, DOM.

O raiz dessa árvore, do ponto de vista do documento, é um nó chamado document. A partir dele, você consegue navegar por toda a estrutura do documento.

O HTML se utiliza de um conjunto de elementos chamado tag ou etiqueta. Cada etiqueta é responsável por uma marcação e é escrito entre os caracteres de maior e menor:

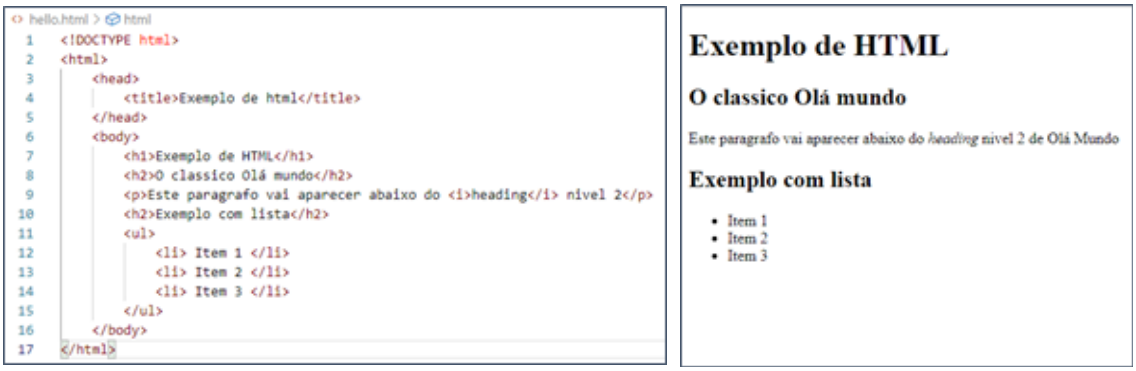
<etiqueta>. No HTML, as etiquetas são separadas em dois grupos:

- **Etiquetas simples ou únicas:** são as etiquetas geralmente desassociada de textos e que não permitem que outros elementos o tenham como nó superior. Exemplos são
 para adicionar uma quebra de linha, <hr> para adicionar uma linha horizontal e <input> para permitir a entrada de informação pelo usuário.

- **Etiquetas pareadas:** diferentemente das etiquetas simples, as etiquetas pareadas se utilizam de representação de abertura e fechamento do elemento na forma de <etiqueta> [algum conteúdo] </etiqueta>. Essas etiquetas permitem que outros elementos, representados por novos conjuntos de etiquetas simples ou pareadas, sejam adicionados o tendo como nó superior. Note que a quantidade de folhas na árvore é ilimitada, e você pode adicionar etiquetas dentro de etiquetas, dentro de etiquetas até que seu documento esteja representado da forma desejada. As etiquetas mais comuns são <h1></h1> para títulos, <h2...6></h2...6> para subtítulos, para separar elementos logicamente em seu documento, para iniciar um novo parágrafo de texto e <a> para adicionar uma ligação de hipertexto para outro documento.

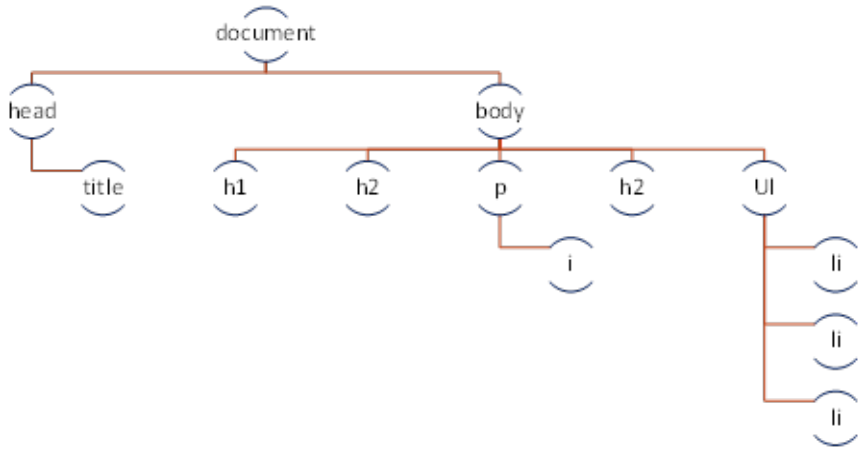
Para ficar mais claro, veja o clássico exemplo “Olá Mundo” de HTML e sua respectiva árvore.

Figura 2 – Código HTML e sua renderização no Google Chrome



Fonte: Elaborada pelo autor.

Figura 3 – Representação gráfica da árvore da Figura 2



Fonte: Elaborada pelo autor.

Por fim, mas não menos importante, as etiquetas HTML possuem algumas propriedades. Algumas obrigatórias, dependendo do elemento, algumas opcionais. E, no desenvolvimento web moderno, o programador pode incluir propriedades adicionais em suas etiquetas para que outros programadores, ou ele mesmo, possam usufruir.

```
<etiqueta propriedade1="valor1" propriedade2="valor2">
    [algum conteúdo ou outros etiquetas]
</etiqueta>
```

As principais propriedades são:

- **href:** utilizada dentro das etiquetas de ligação <a> para indicar qual documento aquele link deve seguir:

```
<a href="http://www.mackenzie.br">Visite o site do Mackenzie</a>
```

- **src:** utilizada dentro de algumas etiquetas, como <img>, apontando para a origem do recurso. Também podem ser utilizadas as propriedades width para largura e height para altura da imagem:

```
<img src="path/to/img.png">
```

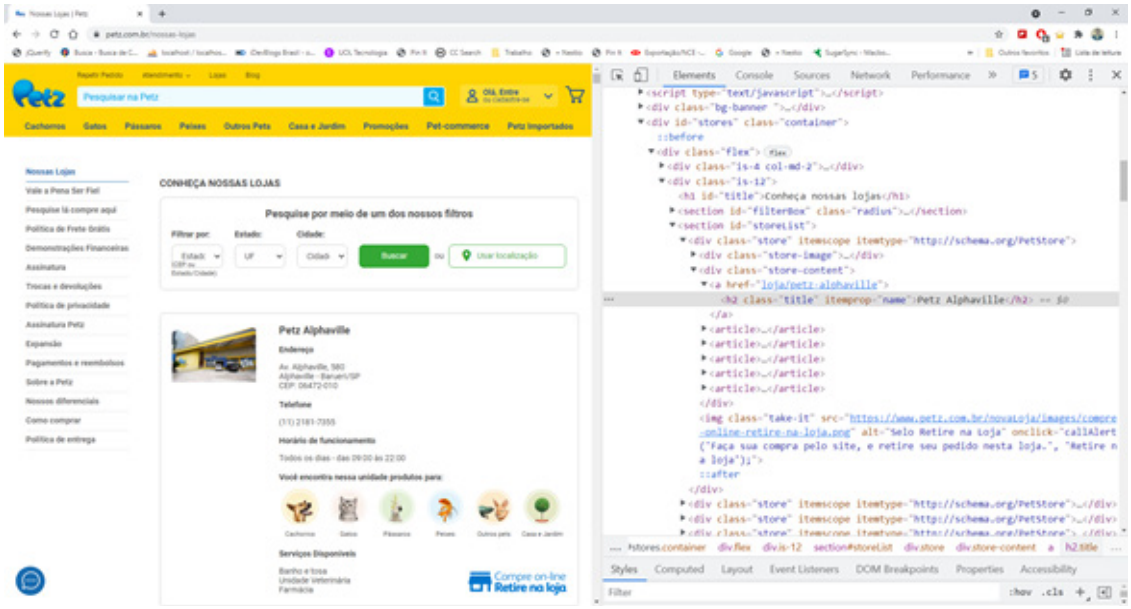
• **type e value:** utilizadas nas etiquetas de formulário <input> para determinar o tipo (text, password, hidden, checkbox, radio, button) e o valor inicial (text e password), ou os valores associados ao elemento (demais casos). No caso do tipo button, os valores possíveis são reset (voltar os elementos do formulário ao valor padrão, ignorando alterações do usuário) e sumbit (enviar o formulário para processamento):

```
<form action="/">
    <input type="text" value="seu e-mail">
    <input type="password" value="">
    <input type="button" value="submit">
</form>
```

• **id:** um dos atributos mais importantes de se observar em raspagem de tela. A propriedade ID cria uma identificação para a etiqueta, o que facilita a navegação e a seleção do elemento. Cuidado, porém, pois, pelo padrão HTML, essa identificação pode não ser única.

• **class:** O atributo class, por sua vez, é utilizado principalmente por designers para aplicar um conjunto de estilos pré-determinado ao elemento e é muito útil para repetir elementos visuais na tela. Observando esse atributo, você conseguirá encontrar, com maior facilidade, a lógica da diagramação do site e utilizar essa informação em sua lógica de raspagem de tela.

Figura 4 – Uma página real da internet e as etiquetas usadas com id e class



Fonte: Elaborada pelo autor.

2.3. A biblioteca Python Beautiful Soup 4.0

A biblioteca *Beautiful Soup* foi criada em 2004 pelo cientista da computação estadunidense Leonard Richardson. O objetivo do desenvolvedor foi elaborar uma ferramenta para pequenos projetos de ganho rápido no dia a dia.

Como se pode imaginar, escrever um documento HTML com algumas falhas de sintaxe é algo fácil de acontecer. Pode ser uma etiqueta com propriedades inválidas e que será interpretado pelo navegador como uma propriedade adicionada pelo desenvolvedor ou erros na ordem de abertura e fechamento de uma etiqueta.

Esses erros são extremamente comuns. De acordo com um e-mail escrito pelo suíço Ian Hickson, um dos maiores especialistas em padrão web, em um teste conduzido por ele em 2006, avaliou-se que mais de 93% dos documentos falharam quanto à correção da sintaxe HTML.[3]

3 Uma cópia dos e-mails trocados pode ser lida pelo site da W3C em <https://lists.w3.org/Archives/Public/www-tag/2006Aug/0048.html>

Dada essa característica, criou-se um jargão no mundo web de chamar as etiquetas presentes em um documento real da web de sopa de etiquetas. Alguns dos erros, por serem comuns, passaram a ser interpretados pelos motores dos navegadores atuais.

Leonard Richardson criou um interpretador de documentos HTML que conseguiu decodificar esses documentos mal escritos e, da melhor forma possível, montar a árvore DOM de tais documentos. Daí o nome ***Beautiful Soup***.

Como uma biblioteca Python, pode ser instalada com o comando pip.

```
1.  pip install beautifulsoup4
```

O funcionamento da biblioteca é bem simples e direto. Basta inicializá-la e passar uma string contendo o texto do documento que deseja interpretar.

```
1.  from bs4 import BeautifulSoup
2.  soup = BeautifulSoup(html_doc_string, 'html.parser')
```

O texto pode vir de um arquivo local em seu computador ou da internet. Para abrir um arquivo local, utilize os comandos do próprio Python.

```
1.  from bs4 import BeautifulSoup
2.  with open('Nossas Lojas _ Petz.html', 'r') as fp:
3.    soup = BeautifulSoup(fp, 'html.parser')
```

Para abrir um arquivo com origem na internet, use a biblioteca request.

```
1.  import requests
2.  from bs4 import BeautifulSoup
3.
4.  url = 'https://www.google.com'
5.  res = requests.get(url)
6.  html_doc_string = res.text
7.  soup = BeautifulSoup(html_doc_string, 'html.parser')
```

Terminada a etapa de carregar o documento dentro da biblioteca, podemos utilizar as funções de busca. Há diversas funções dentro da biblioteca, porém a mais poderosa é a função find_all. A partir dela, você poderá raspar qualquer website. Caso queira se aprofundar no tema, recomendo a leitura de ***Beautiful Soup*** Documentation (RICHARDSON, 2019).

O uso mais imediato é com o soup.find_all("tag"). Nesta configuração, a biblioteca buscará, recursivamente, do nó raiz até todas as folhas a tag especificada, retornando uma lista Python. Essa lista pode ser, então, iterada em um loop, como no exemplo abaixo.

```
1.  from bs4 import BeautifulSoup
2.
3.  with open('Institucional - Cobasi.html', 'r') as fp:
4.    soup = BeautifulSoup(fp, 'html.parser')
5.
6.  for store in soup.find_all("h2"):
7.    print(store.get_text())
```

Um documento HTML pode ser formado por centenas ou milhares de etiquetas, muitas, repetidas. A biblioteca permite que você filtre elementos também pelos seus atributos, inclusive os não padronizados. Para isso, basta adicionar, na consulta, qual propriedade quer filtrar como nesse exemplo: soup.find_all("id=valor").

Você também pode combinar essas propriedades, passando como um objeto adicional à etiqueta. É útil para filtrar etiquetas de um tipo de classe como no exemplo soup.find_all("div", {"class": "artigo"}). Você pode combinar quantos atributos desejar.

Algumas últimas dicas iniciais: pode-se passar uma expressão regular como no exemplo: soup.find_all(re.complile("^b")), solicitar que a busca não seja recursiva: soup.find_all("id=valor", recursive=False) e encadear buscas: soup.find_all("id=valor").find_all("a").

3. Exercício Prático

O objetivo desse exercício é obter uma lista organizada com todos os endereços das unidades da Petz, tradicional rede de pet shop, com dezenas de lojas. É um caso real, no qual foi solicitada uma lista de todas as lojas da cidade de são Paulo para uma ação de marketing de uma empresa que lida com esse público.

As lojas estão endereçadas no sítio da internet da rede em https://www.petz.com.br/nossas-lojas, como pode ser visto na Figura 4.

3.1. Conhecendo o conteúdo a ser raspado

Antes de iniciar a codificação de seu programa de raspagem da web, é importante conhecer o estilo em que ele foi criado e como foi organizado. Lembre-se: seu programa lidará com o texto em HTML e não com a imagem renderizada que você enxerga na tela.

Aqueles que desejarem ver a árvore do documento, semelhante ao mostrado na Figura 3, podem utilizar ferramentas disponíveis na própria internet,[1] porém isso não será necessário.

Os navegadores modernos, tanto Chrome quanto Edge, trazem ferramentas integradas de desenvolvimento chamada *DevTools*.

Uma forma de conhecer todo o código fonte do site é utilizar o menu de contexto (geralmente acessado com o botão direito do mouse) e clicar na opção **exibir código fonte** da página. Essa função abrirá, em outra aba, todo o código HTML que está sendo renderizado.

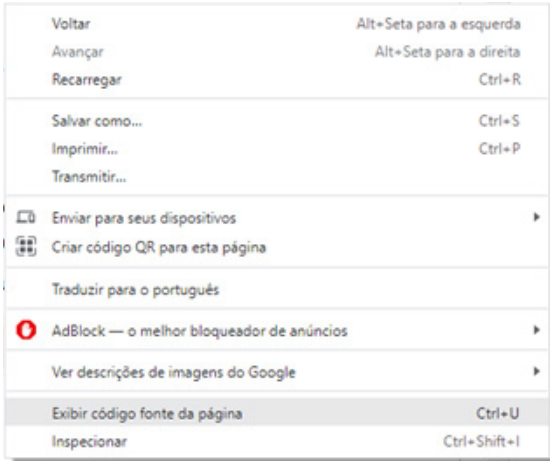
Uma opção melhor é utilizar o item de menu **inspecionar** (vide Figura 5). Esta opção abrirá, compartilhada com a mesma aba da página, a ferramenta *DevTools*. Adicionalmente, a ferramenta o levará para o nó exato correspondente ao elemento que estava sobre o ponteiro no mouse no momento do clique.

Outra vantagem de utilizar essa ferramenta é a possibilidade de encontrar informações

1 Procure por *HTML Dom Tree Visualization*. Uma ferramenta disponível pode ser acessada em <https://software.hixie.ch/utilities/js/live-dom-viewer/>. Cuidado com o tamanho da árvore, utilize apenas os trechos em que tiver alguma dúvida a respeito da organização para evitar que seu navegador pare de responder.

detalhadas sobre o elemento e, navegando pelo ***DevTools***, de conseguir o mesmo resultado ao passar o mouse sobre os nós da árvore HTML. O navegador renderizará retângulos coloridos opacos, permitindo que você veja o que cada nó representa no documento.

Figura 5 – Menu de contexto do Google Chrome 92



Fonte: Elaborada pelo autor.

Por fim, e não menos importante, aqueles confortáveis em utilizar ECMAScript, vulgo javascript, podem utilizar o console do DevTools para fazer uma primeira inspeção no código.

O comando mais básico para encontrar os dados via console são

o el = document.querySelector(param) para buscar o primeiro elemento que corresponde ao parâmetro digitado, e

el = document.querySelectorAll(param) para retornar uma lista com todos os elementos que correspondem à pesquisa do parâmetro.

Para a consulta, existem alguns operadores especiais, para identificar se você está buscando, sempre entre aspas (note o uso dos sinais . e #):

- **etiqueta:** el = document.querySelectorAll("div");
- **classes:** el = document.querySelectorAll(".store-content");
- **id:** el = document.querySelector("#title").

Para contar quantos elementos existem na lista, basta usar el.length.

A web está repleta de referências, guias e tutorias de javascript àqueles que desejam se aprofundar, sendo os materiais no Mozilla Developer Network os mais simples e completos.

3.2. Executando o código Python

O código Python está sendo apresentado para execução em um caderno de Python (*Jupyter Notebook*), mas pode ser utilizado em qualquer arquivo texto, com a extensão .py e rodado localmente. Como serão utilizados dados remotos, é importante verificar se você possui conexão com a internet.

Como entre a elaboração desse material e a execução desse exercício pode ocorrer atualizações na página do exemplo, uma cópia dela pode ser encontrada para download.

Como sugestão, abra o endereço https://colab.research.google.com/ e use uma conta gratuita do Google para executar esse exemplo, criando um novo notebook.

Passo 1

Agora que conhecemos o código, a primeira etapa é importar o HTML em uma variável string do Python. Para isso, basta usar a biblioteca request, passando a URL (um tipo específico de URI, para localização de documentos na web). Caso queira visualizar o conteúdo no próprio jupyter notebook, basta colocar na última linha o nome da variável (opcional).

```
1.  import requests
2.
3.  url = 'https://www.petz.com.br/nossas-lojas'
4.  res = requests.get(url)
5.  html_page = res.text
```

Passo 2

Importamos a biblioteca Beautiful Soup 4 e a inicializamos com o texto gerado no item anterior. Identificamos para a biblioteca que se trata de um html, de forma que ela possa trabalhar com o DOM. Não se preocupe com a falta de mensagens na tela.

```
1.  from bs4 import BeautifulSoup
2.  soup = BeautifulSoup(html_page, 'html.parser')
```

Passo 3

Esta etapa é a peça central da raspagem de tela. Pela inspeção que fizemos no tópico 4.1, foi possível identificar o padrão. Todas as lojas estão em etiquetas div com a classe store. Caso você queira dar mais uma inspecionada, execute o código soup.find_all("div",

{“class”: “store”})[0].

Novamente, pela nossa inspeção, vimos que o desenvolvedor utilizou atributos personalizados na construção da tela itemprop, o que será bastante útil em nossa raspagem.

Por estarmos trabalhando com textos, alguns artifícios foram utilizados para remover espaços e quebras de linha,[2] como forma de tratamento dos textos, pois são indesejados para nosso propósito.

Por fim, todos os dados são salvos em um dicionário do Python, de nome stores, declarado no início do código. Caso queira ver o resultado da raspagem, acrescente a variável stores na última linha do seu código, lembrando de colocá-la na mesma identação do seu loop.

```
1.  import re
2.  stores = []
3.  for index, store in enumerate(soup.find_all("div", {"class": "store"}), start=1):
4.      name = store.find("h2", {"itemprop": "name"}).get_text().strip()
5.      address = store.find("p", {"itemprop": "address"})
6.      address = address.get_text()
7.      address = re.sub(' +', ' ', address)
8.      address = " ".join([s for s in address.strip().splitlines(False) if s.strip()])
9.      addressLoc = store.find("span", {"itemprop": "addressLocality"}).get_text().strip()
10.     region = store.find("span", {"itemprop": "addressRegion"}).get_text().strip()
11.     telephone = store.find("p", {"itemprop": "telephone"}).get_text().strip()
12.     openingHours = store.find("p", {"itemprop": "openingHours"}).get_text().strip()
13.     row = { 'name': name,                          'address':address,
14.             'telephone': telephone,                'openingHours':openingHours,
15.             'addressLocality': addressLoc,         'region': region}
16.     stores.append(row)
```

Passo 4

Para facilitar a manipulação do dado raspado, este é convertido ao formato CSV, que pode ser lido nativamente pelo MS Excel, entre outras aplicações.

Para executar a conversão, importamos a biblioteca csv do próprio Python e utilizamos a manipulação de arquivos para escrevê-lo. Note que, para o cabeçalho, usamos as chaves do nosso dicionário.

```
1.  import csv
2.  keys = stores[0].keys()
3.  with open('stores.csv', 'w', newline='')  as output_file:
4.    dict_writer = csv.DictWriter(output_file, keys)
5.    dict_writer.writeheader()
6.    dict_writer.writerows(stores)
```

2 “Na computação, line feed (LF), nova linha ou quebra de linha é um caractere de controle que indica que uma linha deve ser acrescentada.” (Nova linha, Wikipédia).

4. Síntese

Esta trilha continuou a jornada de recuperação de informação da Web, ensinando você a identificar os elementos apresentados nos documentos em seu formato HTML e a capturar somente os dados que são de interesse do projeto.

Foi apresentada a biblioteca ***Beautiful Soup*** 4, a qual, apesar de não seguir de forma integral os padrões estabelecidos pelas linguagens na manipulação do objeto de modelo do documento, é extremamente versátil para manipulação rápida em pequenas tarefas.

O uso dessas ferramentas permite diminuir risco operacional e ganhar velocidade na obtenção das informações disponíveis em documentos pelos mais diversos times.

Com provocação, deixamos alguns questionamentos a serem discutidos em nossa atividade de fórum.

- Como você poderia utilizar a raspagem de tela no seu dia a dia?
- E fora do seu dia a dia, onde você imagina o uso dessa técnica para obtenção de informação?
- Quais negócios você conhece hoje na internet que dependem da raspagem de dados?
- Você imagina algum novo negócio sendo possível pela raspagem de informações na Web?

5. Referências

ATHOW, D. What is Usenet? 5 things you didn't know about it. ***ThechRadar***, 16 maio 2018. Disponível em: <https://www.techradar.com/news/what-is-usenet-5-things-you-didnt-know-about-it>. Acesso em: 16 set. 2021.

BERNERS-LEE, T. ***Weaving the Web***: the original design and ultimate destiny of the World Wide Web. New York: Harper Business, 2000.

BHUSHAN, A. File Transfer Protocol. ***RFC 114***, abr. 1971. Disponível em: <https://www.rfc-editor.org/info/rfc114>. Acesso em: 16 set. 2021.

GIHRING, T. The rise and fall of the Gopher protocol. ***Minnpost***, 11 ago. 2016. Disponível em: <https://www.minnpost.com/business/2016/08/rise-and-fall-gopher-protocol/>. Acesso em: 24 set. 2021.

GNIPPER, P. 10 anos sem Brasnet: se você usou o mIRC, você está ficando velho. ***Canal Tech***, 14 jul. 2017. Disponível em: <https://canaltech.com.br/curiosidades/10-anos-sem-brasnet-se-voce-usou-o-mirc-voce-esta-ficando-velho-97294/>. Acesso em: 16 set. 2021.

HTML: LINGUAGEM de Marcação de Hipertexto. ***Mozilla Developer Network Web Docs***, [s.d.]. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 23 set. 2021.

JAVASCRIPT. ***Mozilla Developer Network Web Docs***, [s.d.]. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 23 set. 2021.

LEINER, B. M.; CERF, V. G.; CLARK, D. D.; KAHN, R. E.; KLEINROCK, L.; LYNCH, D. C.; POSTEL, J.; ROBERTS, L. G.; WOLFF, S. A brief history of the Internet. ***Cornell University,*** 23 jan. 1999. Disponível em: <https://arxiv.org/abs/cs/9901011>. Acesso em: 16 set. 2021.

MELNIKOV, A.; LEIBA, B. Internet Message Access Protocol (IMAP) – Version 4rev2. ***RFC Editor,*** 2021. Disponível em: <https://www.rfc-editor.org/rfc/rfc9051.html>. Acesso em: 16 set. 2021.

OIKARINEN, J.; REED, D. Internet Relay Chat Protocol. ***RFC 1459***, maio 1993. Disponível em: <https://www.rfc-editor.org/info/rfc1459>. Acesso em: 16 set. 2021.

POSTEL, J.; SLUIZER, S. Mail Transfer Protocol. *RFC 772*, set. 1980. Disponível em: <https://rfc-editor.org/rfc/rfc772.txt>. Acesso em: 16 set. 2021.

REYNOLDS, J. K. Post Office Protocol. *RFC 918*, out. 1984. Disponível em: <https://www.rfc-editor.org/rfc/pdfrfc/rfc918.txt.pdf>. Acesso em: 16 set. 2021.

RICHARDSON, L. *Beautiful Soup Documentation*: Release 4.4.0. 2019. Disponível em: <https://beautiful-soup-4.readthedocs.io/_/downloads/en/latest/pdf/>. Acesso em: 24 set. 2021.

TOTAL number of websites, 2021. Disponível em: <https://www.internetlivestats.com/total-number-of-websites>. Acesso em: 16 ago. 2021.