



# PYTHON – AULA 4

## WEB SCRAPING

# AGENDA

## Aula 1

Introdução – conceitos, arquitetura cliente – servidor, Introdução a HTML, CSS e JavaScript, URLLIB e Requests.

## Aula 2

Introdução ao BeautifulSoup, Expressões Lambdas, Expressões regulares (REGEX)

## Aula 3

Beautiful Soup, List Comprehension, Revisitando NUMPY e PANDAS, Introdução ao Selenium

## Aula 4

Selenium 4, Scraping de Imagens, Trabalhando com inputs em pesquisas, Introdução ao Docker.

## Aula 5

Selenium 4, Banco de dados SQLite, projeto final, considerações finais

# WEB SCRAPING

SELENIUM 4  
SCRAPING DE IMAGENS  
INPUTS EM PESQUISAS  
INTRODUÇÃO AO DOCKER

## Web Scraping – Selenium



**Selenium**

- O Selenium é uma biblioteca de testes para manipulação de páginas dinâmicas.
- Automatiza browsers
- Permite simular um usuário real utilizando um navegador
- Mais indicado para sites que contêm muito código Javascript, JQuery, Angular, Vue e React.

# Web Scraping – Selenium


+ Para utilizar o Selenium no Python:


1. Instale a biblioteca pelo gerenciador de pacotes
2. Acesse: <https://pypi.org/project/selenium/>
3. Faça a escolha do driver para seu navegador; ao extrair direcione para a pasta com o Python instalado.

← → ↻ 🔒 pypi.org/project/selenium/

Professor Online Portal FIAP FIAP EAD emailMicrosoftF yahooG Gmail WhatsApp Rico XP B3 Santander Alura GitHub Lattes Discip

🔗 New blog post, on PyPI's \*new\* blog! [Read Now](#)

[tebeka](#)

[titusfortner](#)

---

**Classificadores**

**Development Status**

- [5 - Production/Stable](#)

**Intended Audience**

Other supported browsers will have their own drivers available. Links to some of the more popular browser drivers follow.

Chrome:	<a href="https://chromedriver.chromium.org/downloads">https://chromedriver.chromium.org/downloads</a>
Edge:	<a href="https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/">https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/</a>
Firefox:	<a href="https://github.com/mozilla/geckodriver/releases">https://github.com/mozilla/geckodriver/releases</a>
Safari:	<a href="https://webkit.org/blog/6900/webdriver-support-in-safari-10/">https://webkit.org/blog/6900/webdriver-support-in-safari-10/</a>



## Web Scraping – Selenium

Para localizar a instalação do Python na sua máquina, rode o código abaixo no Prompt de Comando

```
C:\> Prompt de Comando - python

Microsoft Windows [versão 10.0.19044.2728]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\Emerson Abraham>python
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug  1 2022, 21:53:49) [MSC v.1932 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> import sys
>>>
>>> print(os.path.dirname(sys.executable))
C:\Users\Emerson Abraham\AppData\Local\Programs\Python\Python310
>>> _
```

## Web Scraping – Selenium

Vamos acessar a página do Selenium.

```
from selenium import webdriver

driver = webdriver.Chrome(executable_path=r"\chromedriver.exe")

driver.get("http://selenium.dev")

driver.quit()
```



## Web Scraping – Selenium 4

- Com a chegada do Selenium 4, não precisamos mais gerenciar os drivers (versão do navegador)
- Tivemos também algumas mudanças em relação às classes e seus métodos.
- No núcleo do Selenium 4, está o WebDriver, uma interface para escrever conjuntos de instruções que podem ser executados de forma intercambiável em muitos navegadores.
- O Selenium WebDriver é uma recomendação do W3C. Uma API compacta orientada a objetos, que “dirige” o navegador de forma eficaz.

# Web Scraping – Selenium 4

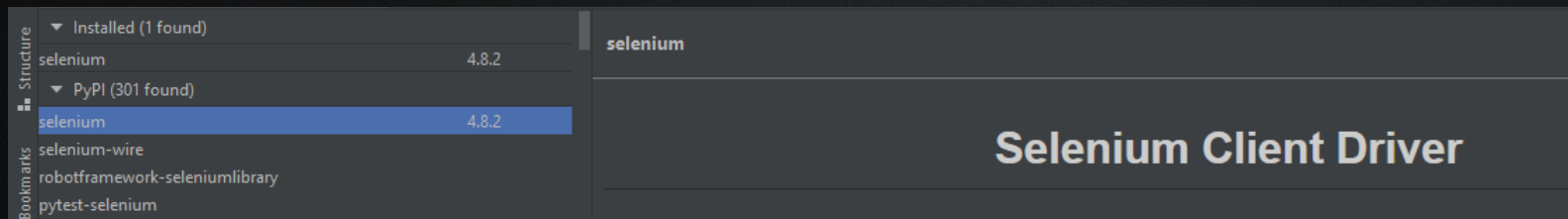
- Instalando / atualizando os recursos na máquina

```
PS C:\Users\Emerson Abraham\PycharmProjects\web_scraping> pip install selenium==4.0.0
Collecting selenium==4.0.0
  Downloading selenium-4.0.0-py3-none-any.whl (954 kB)
    954.3/954.3 kB 3.0 MB/s eta 0:00:00
Requirement already satisfied: urllib3[secure]~=1.26 in c:\users\emerson abraham\appdata\local\pr
1.26.15)
```

```
Terminal: Local × + ▾
Successfully uninstalled selenium-4.8.2
Successfully installed cryptography-39.0.2 pyOpenSSL-23.0.0 selenium-4.0.0 urllib3-secure-extra-0.1.0
PS C:\Users\Emerson Abraham\PycharmProjects\web_scraping> pip3 install webdriver_manager
Collecting webdriver_manager
  Downloading webdriver_manager-3.8.5-py2.py3-none-any.whl (27 kB)
Collecting requests
```

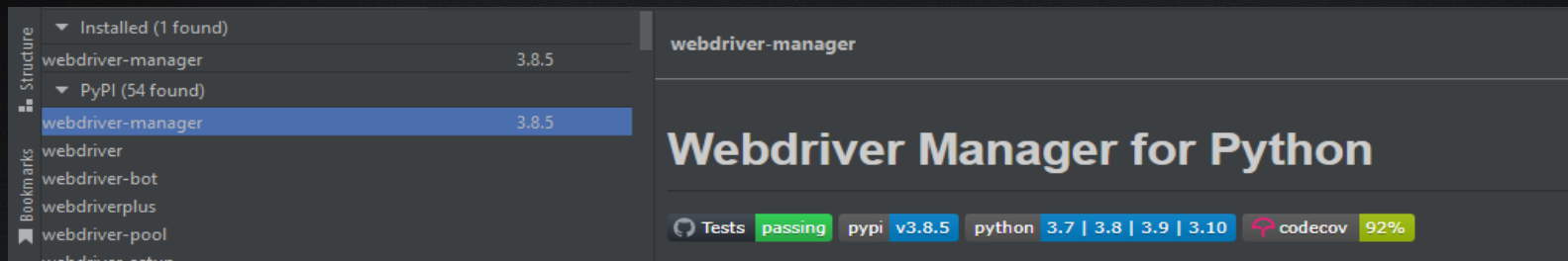
## Web Scraping – Selenium 4

### Instalando / atualizando os recursos no projeto



The screenshot shows the PyCharm IDE interface. On the left, the 'Structure' tool window is open, displaying a list of installed packages. 'selenium' version 4.8.2 is highlighted in blue. Below it, other packages like 'selenium-wire', 'robotframework-seleniumlibrary', and 'pytest-selenium' are listed. The main editor area on the right shows the title 'Selenium Client Driver'.

Package	Version
selenium	4.8.2
selenium-wire	
robotframework-seleniumlibrary	
pytest-selenium	



The screenshot shows the PyCharm IDE interface. On the left, the 'Structure' tool window is open, displaying a list of installed packages. 'webdriver-manager' version 3.8.5 is highlighted in blue. Below it, other packages like 'webdriver', 'webdriver-bot', 'webdriverplus', 'webdriver-pool', and 'webdriver-setup' are listed. The main editor area on the right shows the title 'Webdriver Manager for Python'. Below the title, there is a status bar with 'Tests passing', 'pypi v3.8.5', 'python 3.7 | 3.8 | 3.9 | 3.10', and 'codecov 92%'.

Package	Version
webdriver-manager	3.8.5
webdriver	
webdriver-bot	
webdriverplus	
webdriver-pool	
webdriver-setup	

## Web Scraping – Selenium 4

### Classe By

Esta classe tem as estratégias de localização. Vamos utilizar um `find_element` genérico e passamos as estratégias.

#### selenium.webdriver.common.by

The By implementation.

##### Classes

`By` Set of supported locator strategies.

`class selenium.webdriver.common.by.By`

Set of supported locator strategies.

`CLASS_NAME = 'class name'`

`CSS_SELECTOR = 'css selector'`

`ID = 'id'`

`LINK_TEXT = 'link text'`

`NAME = 'name'`

`PARTIAL_LINK_TEXT = 'partial link text'`

`TAG_NAME = 'tag name'`

`XPATh = 'xpath'`

# Web Scraping – Selenium 4

## Exemplo InfoMoney



The screenshot shows the Chrome DevTools Elements panel. The HTML structure of the page is displayed. A red circle highlights the following HTML elements:

```
<div class="table-header"> flex
  <i class="material-icons up">arrow_upward</i>
  <h2>Maiores altas</h2>
</div>
<table id="high"> ... </table>
</div>
<div class="table-quote"> ... </div>
</div>
<div class="col-12 p-0 widget-footer"> ... </div> flex
</div>
<div class="row mt-5 default_Small"> ... </div> flex
<div class="row mt-5 default_Small"> ... </div> flex
</div>
<!-- Caso exclusivo pra HOME versão DESKTOP. Essa config não converte para
mobile e trás 2 blocos de anúncio -->
<div class="col-12 col-lg-4 VERTICAL_AF_2 container-ads"> ... </div> flex
<div class="row mt-5 widget-columnists"> ... </div> flex
```

# Web Scraping – Selenium 4

```
from selenium import webdriver
from selenium.webdriver.common.by import By

#Chrome
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

#faz a instância do driver apropriado
driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install()))

driver.get("https://www.infomoney.com.br/")

dados_infomoney = driver.find_element(By.ID, "high").text

print(dados_infomoney)
```



# Exemplo Funds Explorer

```
from selenium import webdriver
from selenium.webdriver.common.by import By

#Chrome
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

#faz a instância do driver apropriado
driver =
webdriver.Chrome(service=Service(ChromeDriverManager().install()))

driver.get("https://www.fundsexplorer.com.br/funds/rbds11")
```

## Web Scraping – Selenium 4

The screenshot shows a web browser at the URL `fundsexplorer.com.br/funds/rbds11`. The page displays information for the fund **RBDS11**, including its name **RB CAPITAL DESENV. RESID. II FDO INV IMOB - FII** and various performance metrics like **Liquidez Diária** (83 negociações), **Último Rendimento** (R\$ 2,35), and **Valor Patrimonial** (R\$ 19,91). The browser's developer tools are open, showing the HTML structure. A context menu is visible over the fund name **RBDS11**, indicating its role as a **section-title** and **heading**. The HTML structure shows the following elements:

```
<header></header>
<section id="funds-show">
  <section id="head" class="rbds11-header">
    <div class="container">
      ::before
      <div class="row">
        ::before
        <div class="col-md-12 col-xs-12">
          <div class="section-dividen"></div>
          <div class="ticker-wrapper"> flex
            <h1 class="section-title">RBDS11</h1>
          </div>
          <h3 class="section-subtitle">RB CAPITAL DESENV. RESID. II FDO INV IMOB - FII</h3>
          <div id="stock-price-wrapper"> </div>
        </div>
      </div>
    </div>
  </section>
</section>
```

```
#find_element_by_tag_name
titulo = driver.find_element(By.TAG_NAME, "h1").text
```

## Web Scraping – Selenium 4

The screenshot shows a web browser window with the URL `fundsexplorer.com.br/funds/rbds11`. The page displays financial data for "RB CAPITAL DES" and "FII IMOB - FII". A context menu is open over the price "R\$ 3,61", showing properties like `span.price`, `64.31 x 28.56`, `Color`, `Font`, `Margin`, `ACCESSIBILITY`, `Contrast`, `Name`, `Role`, and `Keyboard-focusable`. The developer tools are open on the right, showing the `Elements` tab with the following HTML structure:

```
<section id="head" class="roussi-header">
  <div class="container">
    ::before
    <div class="row">
      ::before
      <div class="col-md-12 col-xs-12">
        <div class="section-divider"></div>
        <div class="ticker-wrapper">
          <h3 class="section-subtitle">RB CAPITAL DESENV. RESID. II FDO INV IMOB - FII</h3>
          <div id="stock-price-wrapper">
            <div id="stock-price">
              <span class="price">R$ 3,61 </span> == $0
              <span class="percentage negative">-0,28% </span>
            </div>
          </div>
        </div>
      </div>
    </div>
  </section>
```

The bottom of the page shows a table with financial metrics:

Liquidez Diária	Último Rendimento	Dividend Yield	Patrimônio Líquido	Valor Patrimonial
83 negociações	R\$ 2,35	0,00%	R\$ 2,5 mi	R\$ 19,91

```
#find_element_by_class_name
```

```
preco = driver.find_element(By.CLASS_NAME, "price").text
```

## Web Scraping – Selenium 4

The screenshot shows a web browser displaying the fundsexplorer.com.br website. The main content area features a blue header with the 'fundsexplorer' logo and a search bar. Below the header, the fund 'RBDS11' is highlighted in a large blue box, with the description 'RB CAPITAL DESENV. RESID. II FDO INV IMOB - FII'. A price tag shows 'R\$ 3,61' with a '-0,28%' change. Below this, a table of indicators is visible:

Indicador	Valor
Liquidez Diária	83 negociações
Último Rendimento	R\$ 2,35
Dividend Yield	0,00%
Valor Patrimonial	R\$ 2,5 mi
Valor	R\$ 19,91

The right side of the image shows the browser's developer tools, specifically the 'Elements' panel. It displays the DOM tree with a context menu open over a  element. The menu options include 'Add attribute', 'Edit as HTML', 'Duplicate element', 'Delete element', 'Cut', 'Copy', 'Paste', 'Copy element', 'Copy outerHTML', 'Copy selector', 'Copy JS path', 'Copy styles', 'Copy XPath', 'Copy full XPath', 'Hide element', 'Force state', 'Break on', 'Expand recursively', and 'Collapse children'. The 'Copy selector' option is highlighted.

```
#find_element_by_css_selector
```

```
dividend_yield = driver.find_element(By.CSS_SELECTOR, "#main-indicators-  
carousel > div > div > div:nth-child(3) > span.indicator-value").text
```

## Web Scraping – Selenium 4

The screenshot shows a web browser at the URL `fundsexplorer.com.br/funds/rbds11`. The page displays information for the fund **RBDS11**, categorized as **RB CAPITAL DESENV. RESID. II FDO INV IMOB – FII**. Key metrics shown include a daily liquidity of **83** negotiations, a net asset value of **R\$ 2,35**, and a daily return of **0,00%**. The browser's developer tools are open, showing the HTML structure. A context menu is open over the HTML, with the **Copy XPath** option highlighted. The HTML structure shows a carousel of indicators, with the selected indicator being **Liquidez Diária** with a value of **83**.

```
#find_element_by_xpath
```

```
ultimo_rendimento = driver.find_element(By.XPATH, '//*[@id="main-indicators-carousel"]/div/div/div[2]/span[2]').text
```

## Web Scraping – Selenium 4

### find\_element e find\_elements

- `find_element` devolve um elemento único
- Ex: `driver.find_element(By.CLASS_NAME, "value").text`
- `find_elements` devolve uma lista de elementos
- Ex: `driver.find_elements(By.CLASS_NAME, "value")[0].text`
- Estes métodos podem ser utilizados com os demais atributos da classe `By`.



$\bullet \quad + \quad \bullet$

# Web Scraping – Selenium 4

## find\_element e find\_elements

Notou algum padrão ?

**TAE11 - TAESA**  
HOME > Ações > TAE11

**VALOR ATUAL**  
**R\$ 35,00**

MIN. 52 SEMANAS  
**R\$ 31,68**

MIN. MÊS  
R\$ 34,50

DIVIDEND YIELD ②  
**17,69 %**  
ÚLTIMOS 12 MESES  
R\$ 6,1899

MAX. 52 SEMANAS  
**R\$ 39,69**

MAX. MÊS  
R\$ 37,25

VALORIZAÇÃO (12M)  
**↓ -8,16%**  
MÊS ATUAL  
**↓ -1,77%**

Context menu for R\$ 39,69:  
strong.value 76.06 × 28  
Color #000000DE  
Font 28px -apple-system, BlinkMacSystemFon...  
Margin 7px 0px  
ACCESSIBILITY  
Contrast Aa 15.21 ✓  
Name  
Role strong  
Keyboard-focusable

Elements Console Sources Network

```
</div>
</div>
</div>
<div class="info w-50 w-md-33 w-lg-20 border-md-0 border-lg-1">
  <div>
    <div title="Valor máximo das últimas 52 semanas">
      <h3 class="title m-0">Máx. 52 semanas</h3>
      <span class="icon">R$</span>
      <strong class="value">39,69</strong>
    </div>
    <div class="d-flex justify-between">
      <div>
        <div class="info w-50 w-md-50 w-lg-20">
        <div class="info w-50 w-md-50 w-lg-20">
      </div>
    </div>
  </div>
```

Styles Computed Layout Event Listeners DOM Breakpoints Properties Accessibility

... tween.flex-wrap div.info.special.w-100.w-md-33.w-lg-20 div.d-md-inline-block div strong.va

## Web Scraping – Selenium 4

### find\_element e find\_elements

A **class value** é igual para todos os elementos do container. Neste caso, se utilizarmos o **find\_element** teremos apenas o primeiro elemento

```
dados_status_invest = driver.find_element(By.CLASS_NAME, "value").text
```

O ideal é utilizar o **find\_elements** (retorna uma lista), informando a posição que desejamos recuperar.

```
valor_atual = driver.find_elements(By.CLASS_NAME, "value")[0].text  
valor_min = driver.find_elements(By.CLASS_NAME, "value")[1].text  
valor_max = driver.find_elements(By.CLASS_NAME, "value")[2].text  
dividend_yield = driver.find_elements(By.CLASS_NAME, "value")[3].text  
valorizacao = driver.find_elements(By.CLASS_NAME, "value")[4].text
```

- **Web Scraping – Selenium 4**
- **find\_element e find\_elements**
- + .
- + •Faça prints customizando as respostas
- 

```
print("Valor atual: ", valor_atual)
print("Valor Min 52 semanas: ", valor_min)
print("Valor Max 52 semanas: ", valor_max)
print("DY 12 meses: ", dividend_yield)
print("Valorização: ", valorizacao)
```

```
Valor atual: 35,00
Valor Min 52 semanas: 31,68
Valor Max 52 semanas: 39,69
DY 12 meses: 17,69
Valorização: -8,16%
```

## Web Scraping – Selenium 4

### Exercícios

1. Inspecione a página do **Status Invest** e faça scraping dos índices Bovespa, CDI e IPCA
2. Faça uma busca avançada em ações. Selecione um setor e filtre. Encontre a tabela (final da página) e faça o scraping dos tickers, preços e valores de mercado.
3. Escolha um papel e faça o scraping de todos os dados (Indicadores)




- **Web Scraping – Selenium 4**
- **Baixando imagens**

- +
  - +
    - Vamos inspecionar a página IMDB no carousel de fotos do filme Dungeons & Dragons. Ao clicar sobre as imagens encontramos o container; podemos copiar o xPath.

Assistir a Dungeons & Dragons: Wrath of the Dragon God

**div.ipc-sub-grid.ipc-sub-grid--page-span-2.ipc-sub-grid--nowrap.ipc-shoveler\_grid** 363 × 145.5

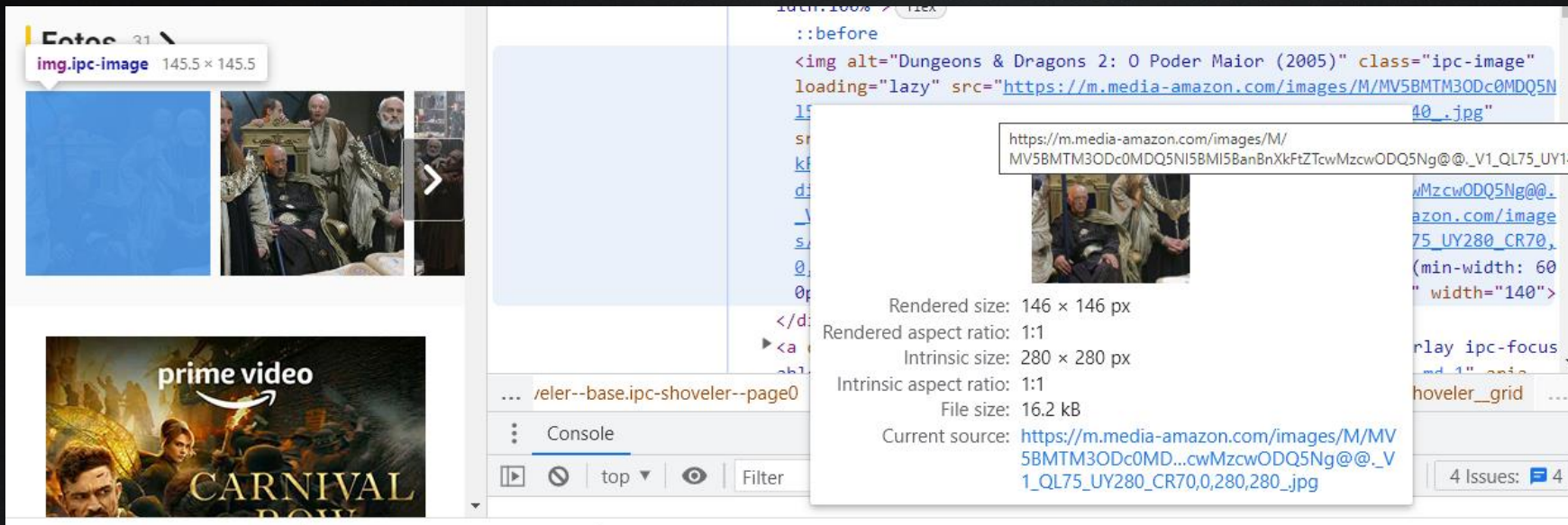


```
</div>
...
<div class="ipc-sub-grid ipc-sub-grid--page-span-2 ipc-sub-grid--nowrap ipc-shoveler_grid" data-testid="shoveler-items-container">
  <div class="ipc-photo ipc-photo--base ipc-photo--dynamic-width photos-image ipc-sub-grid-item ipc-sub-grid-item--span-2" role="group">
    <div class="ipc-media ipc-media--photo ipc-image-media-ratio--photo ipc-media--base ipc-media--photo-m ipc-photo__photo-image ipc-media_img" style="width:100%">
      
    </div>
  </div>
```



- **Web Scraping – Selenium 4**
- **Baixando imagens**

Ao inspecionar as imagens, percebemos que elas usam a **tag img**



## Web Scraping – Selenium 4

### Baixando imagens

- Primeiramente, vamos importar as bibliotecas necessárias e instanciar o driver.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
import urllib.request

driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))

driver.get("https://www.imdb.com/title/tt0406728/?ref_=tt_sims_tt_i_1")
driver.implicitly_wait(10) #segundos
```

- Note que como as imagens podem demorar para carregar, vamos utilizar o método `implicitly_wait`

## • Web Scraping – Selenium 4

### • Baixando imagens

- • Posteriormente, capturamos o Xpath e salvamos em uma variável. Por meio desta variável buscamos as imagens específicas com o `find_elements`.

```
div_imagens = driver.find_element(By.XPATH,  
'//*[@id="__next"]/main/div/section[1]/div/section/div/div[1]/section[2]/div[2]/div[2]')  
imagem = div_imagens.find_elements(By.TAG_NAME, "img")[5]  
src = imagem.get_attribute("src")  
print(src)  
  
try:  
    urllib.request.urlretrieve(src, r"C:\Users\Emerson  
Abraham\PycharmProjects\web_scraping\teste.jpg")  
    print("Imagem copiada")  
except:  
    print("Erro")
```

## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

- Formulários de busca possuem um campo tipo **input**
- O atributo **name** é muito utilizado em campos do tipo **input** que estão em formulários (**form**).
- Com o **find\_element(By.name)** conseguimos buscar o elemento e enviar um dado para o campo.

**Nota:** tenha preferência pelo ID nas buscas, pois este normalmente é um campo único e evita *exceptions*

- **Web Scraping – Selenium 4**
- **Trabalhando com inputs em pesquisas.**



- +
  - Ocomon é um sistema opensource para gestão de chamados (Helpdesks e Service Desks).
  - Surgiu em 2002 pelo programador Franque Custódio e foi assumido pelo Analista Flávio Ribeiro, que transformou a ferramenta básica em um robusto sistema de ocorrências.

Algumas funções do módulo de Ocorrências:

- +
  - abertura de chamados de suporte
  - busca rápida de informações referentes ao equipamento;
  - acompanhamento do andamento do processo;
  - relatórios gerenciais;
  - controle de SLAs;

Algumas funções do módulo de Inventário:

- cadastro detalhado das informações
- histórico de mudanças (de localidades) dos equipamentos;
- controle de licenças de softwares;
- estatísticas técnicas e gerenciais do parque de equipamentos;



## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.



- Vamos fazer scraping do sistema OCOMON, simulando o processo de acesso, busca avançada de ocorrências e extração da tabela para um arquivo .csv.
- Para este trabalho, vamos usar uma imagem do Docker.



# Docker



- Docker é uma plataforma open-source para desenvolvimento, envio e execução de aplicativos de forma virtualizada.
- Utiliza os containers para isolar os processos das nossas dependências a nível de disco, memória, processamento e rede, onde normalmente cada container deve ser responsável por apenas um processo que irá ser executado.
- Os containers são rodados a partir de imagens, um template com instruções e configurações para a criação do container.

<https://docs.docker.com/get-docker/>

## Instalando o Docker



- Acesse: <https://docs.docker.com/get-docker/>
- Requisitos:
- Virtualização habilitada (gerenciador de tarefas – desempenho)
- Win 10 pró (64 bits) com Hypervisor
- Instalar a versão desktop ou toolbox (para quem não cumpre os requisitos)
- Docker roda em uma máquina virtual bem leve chamada Alpine.

## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

- Após a instalação e execução do Docker, abra o terminal e execute o comando.
- Será feito o download da imagem e seus requisitos.
- Acesse o navegador e digite na barra de endereços.
- Teste o sistema com usuário e senha admin

 **OCOMON**

Início Notícias

Para testar a versão 4.0 – Se você tem o [docker](#) instalado, basta executar o seguinte comando em seu terminal:

```
docker run -it --name ocomon_4 -p 8000:80 flaviorib/ocomon-demo-4.0:20220108 /bin/ocomon
```

Em seguida basta abrir o seu navegador e digitar na barra de endereços:

```
localhost:8000
```

É pronto! Você já está com uma instalação do OcoMon prontinha para testes com os seguintes usuários cadastrados:

Usuário	Senha	Descrição
admin	admin	Nível de administração do sistema
operador	operador	Operador padrão – nível 1
operador2	operador	Operador padrão – nível 2
abertura	abertura	Usuário apenas para abertura de ocorrências

- **Web Scraping – Selenium 4**
- **Trabalhando com inputs em pesquisas.**

- Agora que temos o sistema rodando em Docker, vamos inspecionar os elementos da página. O campo input USUARIO possui ID = user

input#user.input100 548 x 78  
Padding 0px 26px  
ACCESSIBILITY  
Name  
Role textbox  
Keyboard-focusable ✓

Usuário

Senha

☐ Memorizar meu nome de usuário

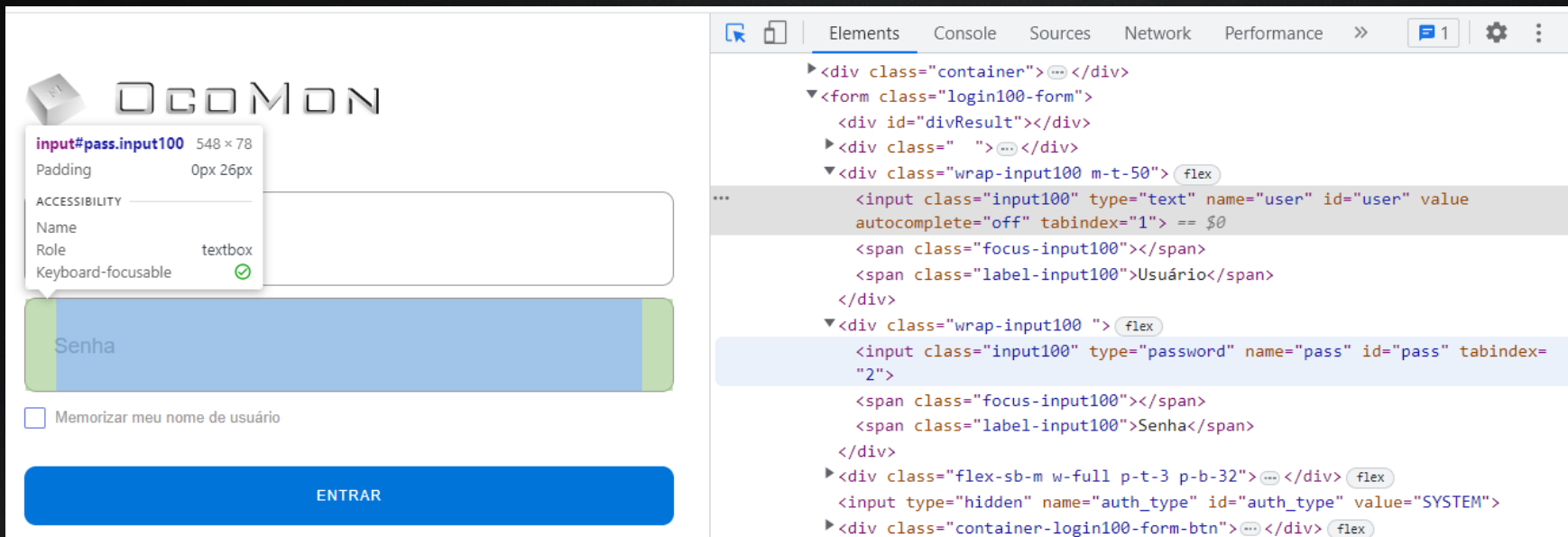
ENTRAR

```
Elements Console Sources Network Performance >> 1 ⚙ ⋮ ×  
<!DOCTYPE html>  
<html lang="pt-BR">  
  <head>...</head>  
  <body style="background-color: #666666;">  
    <div class="limiter">  
      <div class="container-login100"> flex  
        <div class="wrap-login100"> flex  
          <div class="modal " id="modal" tabindex="-1" style="z-index:9001!important">  
            ...</div>  
          <div class="modal" id="modalRecovery" tabindex="-1" style="z-index:9001!important">...</div>  
          <div class="container">...</div>  
          <form class="login100-form">  
            <div id="divResult"></div>  
            <div class=" ">...</div>  
            <div class="wrap-input100 m-t-50"> flex  
              ...  
              <input class="input100" type="text" name="user" id="user" value  
                autocomplete="off" tabindex="1"> == $0  
              <span class="focus-input100"></span>
```

# Web Scraping – Selenium 4

## Trabalhando com inputs em pesquisas.

- O campo input SENHA possui ID = pass



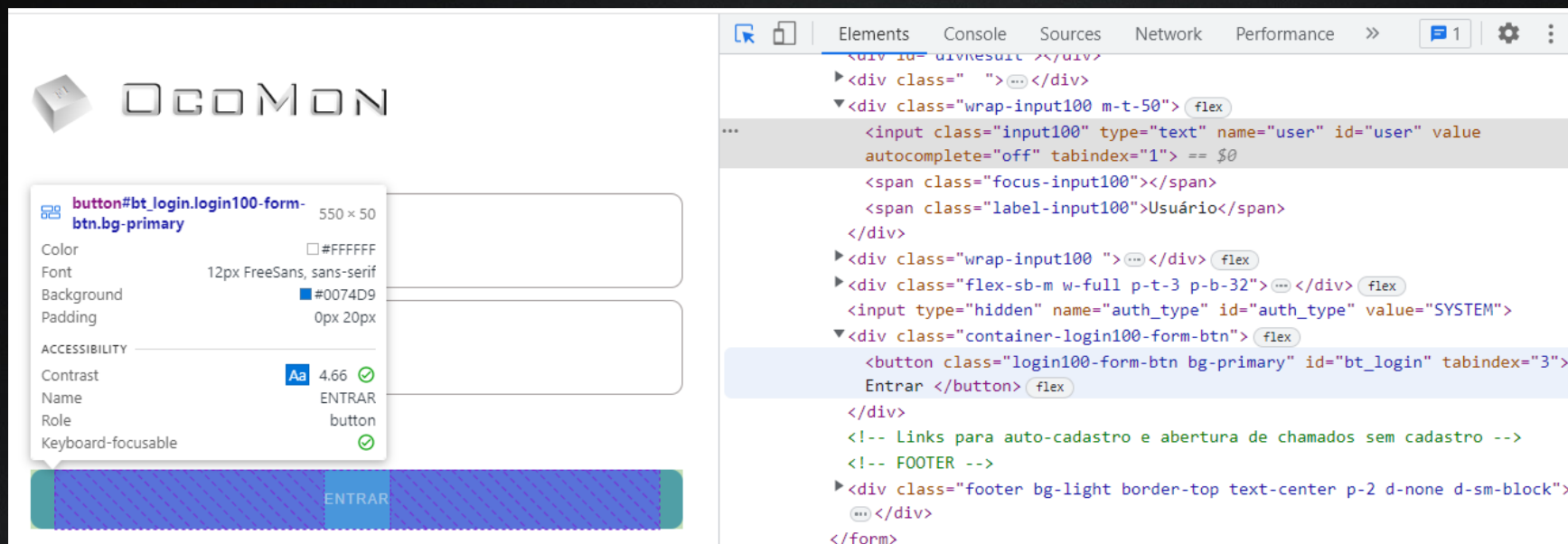
The image shows a web application interface on the left and its developer tools on the right. The web application has a header with a logo and the text "O COMON". Below the header is a login form with two input fields: "Usuário" (User) and "Senha" (Password). The "Senha" field is highlighted with a blue background. Below the inputs is a checkbox labeled "Memorizar meu nome de usuário" (Remember my username) and a blue button labeled "ENTRAR" (Enter).

The developer tools on the right show the "Elements" tab. The DOM tree is expanded to the "login100-form" element. The "input" element with class "input100" and type "password" is selected, showing its attributes: `name="pass" id="pass" tabindex="2"`. The "input" element with class "input100" and type "text" is also visible, showing its attributes: `name="user" id="user" value="" autocomplete="off" tabindex="1"`.

## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

O campo input BOTAO possui ID = btn\_login



The image displays a web application interface on the left and its underlying HTML structure in a browser's developer tools on the right.

**Web Application Interface (Left):**

- Logo: OCOMON
- Form Fields:
  - Input field for user name (ID: user).
  - Input field for password (ID: auth\_type).
  - Submit button (ID: btn\_login) labeled "ENTRAR".
- Accessibility Panel (bottom left):
  - button#bt\_login.login100-form-btn.bg-primary: 550 x 50
  - Color: #FFFFFF
  - Font: 12px FreeSans, sans-serif
  - Background: #0074D9
  - Padding: 0px 20px
  - ACCESSIBILITY: Contrast 4.66 (passing), Name ENTRAR, Role button, Keyboard-focusable (passing).

**Developer Tools (Right):**

- Elements panel shows the HTML structure of the login form.
- Selected element: `<div class="wrap-input100 m-t-50">`
- HTML Structure (simplified):

```
<div id="divresult"></div>
<div class="wrap-input100 m-t-50">
  <input class="input100" type="text" name="user" id="user" value=""
    autocomplete="off" tabindex="1">
  <span class="focus-input100"></span>
  <span class="label-input100">Usuário</span>
</div>
<div class="wrap-input100">
  <input type="hidden" name="auth_type" id="auth_type" value="SYSTEM">
</div>
<div class="container-login100-form-btn">
  <button class="login100-form-btn bg-primary" id="bt_login" tabindex="3">
    Entrar
  </button>
</div>
<!-- Links para auto-cadastro e abertura de chamados sem cadastro -->
<!-- FOOTER -->
<div class="footer bg-light border-top text-center p-2 d-none d-sm-block">
</div>
</form>
```



## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

Vamos iniciar a implementação fazendo os imports necessários e instanciando o driver.

```
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from bs4 import BeautifulSoup
import pandas as pd
from time import sleep
```

```
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
driver.get("http://localhost:8000/ocomon-4.0RC1/login.php")
```

- Web Scraping – Selenium 4
- Trabalhando com inputs em pesquisas.
- + .
- + .
  - Na sequência buscamos os elementos pelo ID.
  - A função **send\_keys** preenche o campo específico.
  - A função **click** executa o função de clicar no botão de entrar
  - Utilizamos a função **sleep** para aguardar a página carregar

```
user = driver.find_element(By.ID, "user")
password = driver.find_element(By.ID, "pass")
btn_login = driver.find_element(By.ID, "bt_login")

user.send_keys("admin")
password.send_keys("admin")
btn_login.click()
sleep(3)
```

## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

#### Exercícios

Faça as buscas por:

- Ocorrência
- Filtro avançado – use **By.LINK\_TEXT**, "Fitro avançado"
- Após o filtro faça a troca de frame

```
frame = driver.find_element(By.ID, "iframeMain")  
driver.switch_to.frame(frame)
```

- Mês corrente
- Data abertura
- Botão pesquisa.

## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

```
filtro = driver.find_element(By.LINK_TEXT, "Filtro avançado")
filtro.click()
frame = driver.find_element(By.ID, "iframeMain")
driver.switch_to.frame(frame)

sleep(3)
mes = driver.find_element(By.ID, "current_month")
mes.click()

data = driver.find_element(By.ID, "data_abertura_from")
data.send_keys("01/01/2020")

btn_pesquisa = driver.find_element(By.ID, "idSearch")
btn_pesquisa.click()
sleep(3)
```

## Web Scraping – Selenium 4

### Trabalhando com inputs em pesquisas.

- Para finalizar vamos extrair os dados da tabela e converter para .CSV com o Pandas.

```
tabela = driver.find_element(By.ID,  
"table_tickets_queue")  
conteudo = tabela.get_attribute("outerHTML")  
soup = BeautifulSoup(conteudo, "html.parser")  
ocorrencias = soup.find(name="table")  
  
data_frame = pd.read_html(str(ocorrencias))[0]  
data_frame.to_csv("ocorrencias.csv", sep=";",  
index=False)
```

## BIBLIOGRAFIA BÁSICA<sup>+</sup>

- + BEAZLEY, David. **Python Essential Reference**, 2009.
- + • BEK, ANDY. **The Ultimate Web Scraping With Python Bootcamp 2023**.
- BHARGAVA, ADITYA Y. **Entendendo Algoritmos. Um guia ilustrado para programadores e outros curiosos**. São Paulo: Ed. Novatec, 2017
- OCOMOM. Disponível em: <https://ocomonphp.sourceforge.io/>, acessado em 03/2023
- + | • DOCKER. Disponível em : <https://www.docker.com/>, acessado em 03/2023
- DOWNEY, ALLEN B. **Pense em Python. Pense como um cientista da computação**. São Paulo: Ed. Novatec, 2016
- DUMS, Anderson F. **Como logar em uma página web e extrair os dados de uma tabela utilizando python e selenium**, Youtube, 2022
- KOPEC, DAVID. **Problemas clássicos de ciência da computação com Python**. São Paulo: Ed. Novatec, 2019
- MCKINNEY, WILLIAM WESLEY. **Python para análise de dados. Tratamento de dados com Pandas, Numpy e Ipython**. São Paulo: Ed. Novatec, 2018
- MITCHELL, RYAN. **Web Scraping com Python. Coletando mais dados na web moderna**. São Paulo: Ed. Novatec, 2019
- W3Schools. Disponível em: [https://www.w3schools.com/python/python\\_lists\\_comprehension.asp](https://www.w3schools.com/python/python_lists_comprehension.asp), acesso: 03/2023



# OBRIGADO



## FIAP

Copyright © 2023 | Professor Dr. Emerson R. Abraham

Todos os direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor/autor.

