Sumário

[1 Exercício webscrapy 1](#_Toc208667263)

[1.1 Exemplo de uso de IA integrada ao windsurf 1](#_Toc208667264)

[1.2 Script do exercício – webscrapy\_41b\_v2 5](#_Toc208667265)

# Exercício webscrapy

## Links:

* Vídeo explicativo: <https://drive.google.com/file/d/1SC91Ad51g6kUQjaOJTPrDd8crIfsPZU3/view?usp=sharing>
* Vídeo feito por IA: <https://drive.google.com/file/d/1Q--byUJLiPvpWB6TcyfXHDvx3Er-y_LM/view?usp=sharing>
* Arquivo com script no github: <https://github.com/marcelo7bastos/mba_enap_introducao_ciencia_dados/blob/main/trabalhos_realizados/webscrapy_41b_v2.py>

## Exemplo de uso de IA integrada ao windsurf

* Pergunta:

@webscrapy\_41b\_v2.py , @webscraping\_41b\_resultado.csv , @webscrapy\_41b.py, quero melhorar a comunicação. como fazer para deixar mais claro o que foi realizado nessa sequencia de arquivos? uso o readme? Crio um novo documento? Também não quero borocratizar demais, ser prolixo demais. o projeto tem que ser mantido simples

* Resposta do modelo Gemini 2.5 Pro:

Of course. Improving documentation is a great way to make your project understandable and maintainable. Given your goal to keep things simple and clear, I believe the best approach is to enhance the existing [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0) file. Creating a separate document might be overkill for this project's scale.

Here's my plan to help you with this:

1. I'll review the contents of `webscrapy\_41b.py` and [webscrapy\_41b\_v2.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b\_v2.py:0:0-0:0) to understand the evolution of your script.

2. I'll propose a clear and concise update to the [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0) that explains the purpose of each file (`webscrapy\_41b.py`, [webscrapy\_41b\_v2.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b\_v2.py:0:0-0:0), and `webscraping\_41b\_resultado.csv`).

3. Once you approve the suggestion, I'll apply the changes to the [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0) file.

Let's start by creating a plan and then reading the files.

I have reviewed both [webscrapy\_41b.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b.py:0:0-0:0) and [webscrapy\_41b\_v2.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b\_v2.py:0:0-0:0).

\* [webscrapy\_41b.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b.py:0:0-0:0) is the initial version, with all the logic in a single function.

\* [webscrapy\_41b\_v2.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b\_v2.py:0:0-0:0) is a much-improved version, refactored into clear functions with better documentation and structure.

To clearly explain the purpose of each file without overcomplicating your [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0), I suggest replacing the current entry with a more descriptive list. This will clarify the evolution from the first script to the second and explain the output file.

Here is the change I propose for your [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0):

I'll replace this section:

```markdown

- `trabalhos\_realizados/` → \*\*meus trabalhos e anotações\*\*

- [webscrapy\_41b.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b.py:0:0-0:0) - Trabalho de web scraping para coleta de dados

```

with this:

```markdown

- `trabalhos\_realizados/` → \*\*meus trabalhos e anotações\*\*

- \*\*Exercício de Web Scraping (4.1b):\*\*

- [webscrapy\_41b.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b.py:0:0-0:0): Versão inicial do script. Realiza a busca no Google e extrai os resultados em um único fluxo.

- [webscrapy\_41b\_v2.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b\_v2.py:0:0-0:0): Versão refatorada e melhorada, com funções separadas e mais robustez. \*\*(Recomendado para estudo)\*\*

- `webscraping\_41b\_resultado.csv`: Arquivo com os dados coletados pelo script (título e URL).

```

This approach keeps the documentation simple and centralized in the [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0), explains the purpose of each file, and guides anyone reading it to the most relevant version of your script.

What do you think of this suggestion? If you agree, I will apply this change.

I've reviewed both the initial and the refactored versions of your web scraping script. To improve clarity without adding unnecessary complexity, I'll update the [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0) to explain the purpose of each file in the `trabalhos\_realizados` directory.

Here's the plan:

1. I'll update the [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0) to differentiate between [webscrapy\_41b.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b.py:0:0-0:0) (the initial version) and [webscrapy\_41b\_v2.py](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/trabalhos\_realizados/webscrapy\_41b\_v2.py:0:0-0:0) (the improved version), and also describe the `webscraping\_41b\_resultado.csv` output file.

2. I'll then mark the task as complete.

I'll now apply the changes to the [README.md](cci:7://file:///c:/Users/marce/OneDrive%20-%20mtegovbr/0001\_NOVA\_PASTA/DESENVOLVIMENTO\_PESSOAL/001\_mba\_ciencia\_dados\_enap/016\_introducao\_ciencia\_dados/mba\_enap\_introducao\_ciencia\_dados/README.md:0:0-0:0) file.

## Script do exercício – webscrapy\_41b\_v2

* Link para o github: <https://github.com/marcelo7bastos/mba_enap_introducao_ciencia_dados/blob/main/trabalhos_realizados/webscrapy_41b_v2.py>

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

"""

Exercício 4.1b — Webscrapy do Google (Selenium)

------------------------------------------------

Abre o Google, busca por "gov.br" e coleta os resultados orgânicos da

primeira página, retornando uma lista de dicionários:

    [{"titulo": "...", "url": "..."}, ...]

Também salva o resultado em CSV na pasta `trabalhos\_realizados`.

Decisões de projeto:

- Selenium + WebDriverWait: o HTML do Google muda com frequência; carregar a

  página “de verdade” e esperar elementos específicos torna o scraper mais estável.

- Estratégia de seleção “robusta”: testamos alguns seletores (CSS/XPath) porque

  o Google alterna o markup entre experiências/regiões. O primeiro seletor que

  encontrar elementos é usado.

- Ética: coletamos apenas a 1ª página, com espera explícita e sem enviar

  requisições agressivas. Para uso intensivo, prefira API oficial (Programmable Search).

Manutenção:

- Caso o seletor pare de funcionar, ajuste a lista CANDIDATE\_SELECTORS.

- Para mudar o termo de busca, altere DEFAULT\_QUERY ou passe a parametrizar.

"""

from \_\_future\_\_ import annotations  # Permite o uso de anotações de tipo como strings

import csv  # Para manipulação de arquivos CSV

import os  # Para interação com o sistema de arquivos

from typing import List, Dict, Tuple  # Tipos para anotações estáticas

from selenium import webdriver  # Framework principal de automação web

from selenium.webdriver.common.by import By  # Estratégias de localização de elementos

from selenium.webdriver.common.keys import Keys  # Teclas especiais do teclado

from selenium.webdriver.support.ui import WebDriverWait  # Para esperas explícitas

from selenium.webdriver.support import expected\_conditions as EC  # Condições de espera

from webdriver\_manager.chrome import ChromeDriverManager  # Gerenciador do ChromeDriver

from selenium.webdriver.chrome.service import Service  # Configuração do serviço do Chrome

from selenium.webdriver.chrome.options import Options  # Opções de configuração do navegador

# ----------------------------- Constantes ------------------------------------

DEFAULT\_QUERY: str = "gov.br"

# Seletores candidatos para encontrar os títulos (h3) de resultados orgânicos.

# A ordem importa: usamos o primeiro que retornar elementos.

CANDIDATE\_SELECTORS: List[Tuple[str, str]] = [

    (By.CSS\_SELECTOR, "div#search a h3"), # seletor usando css selector e id search

    (By.CSS\_SELECTOR, "div#rso a h3"), # seletor usando css selector e id rso

    (By.XPATH, "//div[@id='search']//a//h3"), # seletor usando xpath e id search

    (By.XPATH, "//a//h3"), # seletor usando xpath

]

# Caminho de saída do CSV (relativo ao projeto/fork do aluno)

CSV\_OUTPUT\_RELATIVE: str = os.path.join(

    "mba\_enap\_introducao\_ciencia\_dados", "trabalhos\_realizados", "webscraping\_41b\_resultado.csv"

)

def build\_chrome\_driver() -> webdriver.Chrome:

    """

    Cria e retorna uma instância do Chrome WebDriver com opções razoáveis para scraping leve.

    - `--start-maximized`: janela cheia ajuda no aprendizado (ver o que está acontecendo).

    - user-agent “realista”: reduz bloqueios triviais.

    - `--lang=pt-BR`: melhora a chance de rótulos tipo “Aceitar/Concordo” aparecerem.

    """

    chrome\_options = Options()

    # Descomente para rodar sem abrir janela (CI/servidor):

    # chrome\_options.add\_argument("--headless=new")

    chrome\_options.add\_argument("--lang=pt-BR") # define o idioma do navegador

    chrome\_options.add\_argument("--start-maximized") # abre o navegador maximizado

    chrome\_options.add\_argument("--disable-blink-features=AutomationControlled") # desabilita o automação do navegador

    chrome\_options.add\_argument(

        "user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) "

        "AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120 Safari/537.36"

    ) # define o user agent do navegador para evitar bloqueios

    driver = webdriver.Chrome(

        service=Service(ChromeDriverManager().install()),

        options=chrome\_options

    )

    return driver

def handle\_consent\_if\_present(driver: webdriver.Chrome, wait: WebDriverWait) -> None:

    """

    Tenta aceitar o banner de consentimento, caso apareça.

    Observação:

    - O Google apresenta variantes desse diálogo conforme país/conta/experimentos A/B.

    - Usamos XPaths mais genéricos com palavras “Aceitar/Concordo”.

    - Se não existir, seguimos normalmente.

    """

    try:

        consent\_btn = wait.until(

            EC.presence\_of\_element\_located(

                (

                    By.XPATH,

                    "//button[.//div[contains(.,'Aceitar') or contains(.,'Concordo')]] "

                    "| //button[contains(.,'Aceitar')] "

                    "| //button[contains(.,'Concordo')]"

                )

            )

        )

        consent\_btn.click()

    except Exception:

        # Sem banner de consentimento ou variação não reconhecida: prossiga.

        pass

def search\_google(driver: webdriver.Chrome, wait: WebDriverWait, query: str) -> None:

    """

    Abre a home do Google, aceita consentimento (se houver), localiza a caixa de busca,

    envia a consulta e espera o container de resultados estar presente.

    """

    driver.get("https://www.google.com/")

    handle\_consent\_if\_present(driver, wait)

    # Localiza a caixa de busca, envia o termo e confirma.

    search\_box = wait.until(EC.presence\_of\_element\_located((By.NAME, "q")))

    search\_box.clear()

    search\_box.send\_keys(query)

    search\_box.send\_keys(Keys.ENTER)

    # Aguarda o container principal de resultados para garantir que a página carregou.

    wait.until(EC.presence\_of\_element\_located((By.CSS\_SELECTOR, "div#search")))

def collect\_results(driver: webdriver.Chrome) -> List[Dict[str, str]]:

    """

    Coleta os resultados orgânicos da 1ª página do Google.

    Retorna uma lista de dicionários: [{"titulo": "...", "url": "..."}].

    Estratégia:

    - Tenta múltiplos seletores e usa o primeiro que encontrar elementos.

    - Para cada <h3>, sobe ao elemento <a> ancestral para obter o href.

    - Remove duplicados por URL.

    """

    h3\_elements = []

    for by, sel in CANDIDATE\_SELECTORS:

        els = driver.find\_elements(by, sel)

        if els:

            print(f"[debug] seletor que funcionou: {by} | {sel} | {len(els)} elementos")

            h3\_elements = els

            break

    if not h3\_elements:

        raise RuntimeError("Não consegui localizar resultados (h3). Ajuste CANDIDATE\_SELECTORS.")

    resultados: List[Dict[str, str]] = []

    vistos = set()

    for h3 in h3\_elements:

        try:

            a = h3.find\_element(By.XPATH, "./ancestor::a")  # sobe do <h3> ao link <a>

            titulo = h3.text.strip()

            url = a.get\_attribute("href")

            if not titulo or not url:

                continue

            if url in vistos:

                continue

            vistos.add(url)

            resultados.append({"titulo": titulo, "url": url})

        except Exception:

            # Em scraping, é saudável ignorar falhas pontuais em itens individuais.

            continue

    return resultados

def save\_as\_csv(rows: List[Dict[str, str]], csv\_path: str) -> str:

    """

    Salva a lista de dicionários em CSV com as colunas ['titulo', 'url'].

    - Garante a criação do diretório.

    - Retorna o caminho \*\*absoluto\*\* gerado (útil para logs/prints).

    """

    os.makedirs(os.path.dirname(csv\_path), exist\_ok=True)

    with open(csv\_path, "w", newline="", encoding="utf-8") as f:

        writer = csv.DictWriter(f, fieldnames=["titulo", "url"])

        writer.writeheader()

        writer.writerows(rows)

    return os.path.abspath(csv\_path)

def main() -> None:

    """

    Fluxo principal:

    1) Sobe o ChromeDriver

    2) Busca por DEFAULT\_QUERY

    3) Coleta resultados da 1ª página

    4) Imprime no console e salva em CSV

    """

    driver = build\_chrome\_driver()

    try:

        wait = WebDriverWait(driver, 20)

        search\_google(driver, wait, DEFAULT\_QUERY)

        resultados = collect\_results(driver)

        # Mostra no formato pedido pelo enunciado

        print(resultados)

        # Persiste em CSV para consumo posterior (Looker Studio/Excel/ETL etc.)

        out\_path = save\_as\_csv(resultados, CSV\_OUTPUT\_RELATIVE)

        print(f"\nResultados salvos em: {out\_path}")

    finally:

        # Feche o navegador (em sessões de aprendizado, pode comentar para inspecionar a página).

        driver.quit()

if \_\_name\_\_ == "\_\_main\_\_":

    main()