

FATEC – FACULDADE DE TECNOLOGIA DE RIO CLARO  
CURSO: Inteligência Artificial

## **RELATÓRIO DE PESQUISA – WEB SCRAPING EM PYTHON (FONTE: GEMINI)**

**GRUPO: SOUL CARE  
ALUNO: Vladimir**

## 1. INTRODUÇÃO

O presente relatório registra, de forma técnica e acadêmica, o conteúdo de uma pesquisa realizada na plataforma Google Gemini sobre a aplicação de Web Scraping em Python. Esta pesquisa foi elaborada como suporte ao Projeto “Feeling AI”, desenvolvido pelo grupo SOUL CARE no curso de Inteligência Artificial da FATEC.

O estudo base (“Exemplo de Web Scraping em Python”) aborda conceitos fundamentais de raspagem de dados, descrevendo métodos para extrair e processar informações a partir de páginas web. O material analisado foi produzido com o modelo Gemini 2.5 Flash em 8 de outubro de 2025, às 18h18, e publicado na mesma data às 18h23, disponível em: <https://gemini.google.com/share/08d42e6fbe84>.

## 2. DESENVOLVIMENTO

2.1 Síntese do Conteúdo Original O conteúdo fornecido pelo Gemini introduz o Web Scraping com as bibliotecas “requests” (requisições HTTP) e “Beautiful Soup” (parsing HTML). O exemplo inicial ilustra a extração do título de uma página (tag title) e, em seguida, apresenta a extração de múltiplos parágrafos (tag p). Por fim, exibe um exemplo personalizado de coleta de comentários de uma empresa, com organização em DataFrame (pandas) e exportação para CSV.

2.2 Boas Práticas e Considerações Técnicas As recomendações incluem: uso de cabeçalhos “User-Agent”; tratamento de exceções de rede (try/except com requests.exceptions); validação da presença de elementos HTML (evitando AttributeError); respeito aos Termos de Serviço; e, quando necessário, uso de ferramentas como Selenium para páginas com carregamento dinâmico em JavaScript.

2.3 Aplicação ao Feeling AI A partir do roteiro do Gemini, o pipeline proposto para o Projeto Feeling AI é: (1) Coleta (requests) → (2) Parsing (BeautifulSoup) → (3) Normalização (limpeza de strings e datas) → (4) Estruturação (pandas DataFrame) → (5) Exportação (CSV/JSON) → (6) Integração com o módulo de Análise de Sentimentos do projeto.

Instalação de dependências (pip)

```
# Bash
pip install requests beautifulsoup4
pip install pandas
```

## Exemplo básico - Extrair o título da página (scraper\_exemplo.py)

```
# Python
import requests
from bs4 import BeautifulSoup

# 1. Definir a URL que será raspada (scraped)
url_alvo = "http://www.pudim.com.br/"

# 2. Fazer a requisição HTTP
try:
    resposta = requests.get(url_alvo)
    resposta.raise_for_status() # Verifica status 200
except requests.exceptions.RequestException as e:
    print(f"Erro ao acessar a URL: {e}")
    raise SystemExit

# 3. Criar o objeto Beautiful Soup para analisar o HTML
soup = BeautifulSoup(resposta.content, 'html.parser')

# 4. Extrair o <title>
titulo_pagina = soup.title.text if soup.title else "(sem título)"

# 5. Imprimir o resultado
print("-" * 30)
print(f"URL: {url_alvo}")
print(f"Título da Página Extraído: {titulo_pagina}")
print("-" * 30)
```

### Como executar

```
# Bash
python scraper_exemplo.py
```

## Extração de múltiplos parágrafos (tag p)

```
# Python (continuação do exemplo básico)
# 3. Criar o objeto Beautiful Soup (já criado acima)
# 4. Extrair os parágrafos
paragrafos = soup.find_all('p')

print("\nParágrafos Encontrados:")
for p in paragrafos:
    print(f"- {p.get_text(strip=True)}")
```

## Exemplo personalizado - Busca de comentários de uma empresa (hipotético) (scraper\_comentarios.py)

```
# Python
import requests
from bs4 import BeautifulSoup
import pandas as pd

# 1. URL de avaliações (EXEMPLO HIPOTÉTICO)
url_avaliacoes = "https://www.site-de-avaliacoes.com.br/empresa/minha-empresa/reviews"
```

```

# Cabeçalhos para simular navegador real
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 "
                  "(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36"
}

lista_comentarios = []
print(f"Iniciando scraping da URL: {url_avaliacoes} (HIPOTÉTICO)")

# 2. Requisição HTTP
try:
    resposta = requests.get(url_avaliacoes, headers=headers, timeout=20)
    resposta.raise_for_status()
except requests.exceptions.RequestException as e:
    print(f"ERRO: Não foi possível acessar a URL. Detalhes: {e}")
    raise SystemExit

# 3. Parser HTML
soup = BeautifulSoup(resposta.content, "html.parser")

# 4. Encontrar blocos de comentários (ajuste as classes para o site real)
blocos = soup.find_all("div", class_="review-box")

if not blocos:
    print("Nenhum bloco de comentário encontrado com a classe 'review-box'. Verifique o")
else:
    print(f"Total de {len(blocos)} comentários encontrados.")
    for bloco in blocos:
        try:
            texto = bloco.find("p", class_="review-text").get_text(strip=True)
            data = bloco.find("span", class_="review-date").get_text(strip=True)
            nota_el = bloco.find("span", class_="rating-value")
            nota = nota_el.get_text(strip=True).split("/")[0].strip() if nota_el else "N/A"
            lista_comentarios.append({"Comentário": texto, "Data": data, "Nota": nota})
        except AttributeError:
            # Caso algum elemento não exista em um bloco específico
            continue

# 5. Salvar em CSV com pandas
if lista_comentarios:
    df = pd.DataFrame(lista_comentarios)
    df.to_csv("comentarios_empresa.csv", index=False, encoding="utf-8")
    print("Scraping concluído. Dados salvos em comentarios_empresa.csv")
else:
    print("Nenhum dado válido foi extraído.")

```

6. REFERÊNCIA GEMINI (Google). Exemplo de Web Scraping em Python.  
 Criado com 2.5 Flash em 8 out. 2025, às 18h18; publicado em 8 out. 2025, às 18h23. Disponível em: <https://gemini.google.com/share/08d42e6fbe84>. Acesso em: [inserir data de acesso].