Relatório Técnico: Automação de Coleta de Dados de Produtos HP no Mercado Livre

---

Lancelot Chagas Rodrigues / 554707

Ana Carolina Martins da Silva / 555762

Kauan Alves Batista / 555082

---

1. Introdução

Este relatório descreve o processo de desenvolvimento de uma solução automatizada para coletar dados de anúncios de produtos da marca HP, com ênfase em cartuchos e toners, na plataforma Mercado Livre. O principal objetivo deste trabalho, inserido no contexto do Challenge Sprint – HP, foi construir uma base de dados robusta que pudesse ser utilizada em análises subsequentes, especialmente na identificação de possíveis casos de pirataria de produtos. A ferramenta desenvolvida é um script em Python que utiliza técnicas de web scraping para extrair e organizar as informações relevantes.

2. Estratégia de Coleta de Dados Adotada

2.1. Plataforma Alvo: Mercado Livre

Nossa escolha recaiu sobre o Mercado Livre (mercadolivre.com.br) devido à sua proeminência como um dos maiores marketplaces no Brasil. A plataforma agrega uma vasta gama de anúncios de produtos HP, tanto de revendedores oficiais quanto de terceiros, o que consideramos essencial para a análise comparativa e a detecção de anomalias que poderiam indicar pirataria.

2.2. Ferramenta Principal: Python com Selenium

Para interagir com o site e extrair os dados, optamos pela combinação de Python e a biblioteca Selenium. A natureza dinâmica do Mercado Livre, com muito conteúdo carregado via JavaScript e interações de usuário complexas (como a expansão de descrições), tornava o scraping direto de HTML estático pouco eficaz. O Selenium nos permitiu automatizar um navegador real (Google Chrome), simulando a navegação humana e acessando o conteúdo renderizado.

É importante mencionar que nossa jornada com o Selenium teve seus percalços. Uma abordagem inicial mais direta e menos cautelosa resultou em um bloqueio temporário de IP, um aprendizado valioso sobre as defesas anti-scraping do site. Isso nos levou a explorar, brevemente, a API oficial do Mercado Livre. No entanto,

as barreiras relacionadas a permissões e a necessidade de registro e aprovação de um aplicativo se mostraram um desvio de tempo considerável para os prazos do projeto. Assim, decidimos retornar ao Selenium, mas desta vez com um foco renovado em implementar técnicas de camuflagem mais sofisticadas e um comportamento de scraping mais "gentil" para evitar detecções.

2.3. Fluxo de Navegação e Interação

O processo de coleta foi desenhado da seguinte forma:

1. **Busca Automatizada:** O script opera a partir de uma lista pré-definida de termos de busca (ex: "Cartucho HP 664 original", "Toner HP compatível"). Para cada termo, ele constrói a URL de busca correspondente.
2. **Coleta de Links:** Ao acessar a página de resultados, o script identifica os links para as páginas individuais dos produtos. Realizamos alguns scrolls automáticos para permitir o carregamento de mais itens via "lazy loading", aumentando o pool de produtos potenciais.
3. **Acesso à Página do Produto:** Com a lista de links, o script visita cada página de produto individualmente.
4. **Expansão da Descrição:** Um desafio comum era obter a descrição completa, muitas vezes oculta atrás de um botão "Ver descrição completa". Incluímos uma lógica para identificar e clicar neste elemento específico (<a data-testid="action-collapsable-target" ...>), utilizando JavaScript para o clique quando necessário, a fim de garantir a captura integral do texto.

2.4. Extração de Dados

A extração dos dados de cada página foi feita mapeando os elementos HTML relevantes através de seletores XPath e CSS.

- **Sincronização:** Utilizamos WebDriverWait para garantir que os elementos estivessem completamente carregados antes de qualquer tentativa de extração, crucial para lidar com o carregamento assíncrono.
- **Seletores Flexíveis:** Para campos como nome do vendedor e avaliações, onde a estrutura HTML podia variar, implementamos uma lista de XPaths alternativos, testados em sequência, para aumentar a robustez da coleta.

3. Estrutura e Componentes do Código

O scraper foi desenvolvido em Python 3, apoiando-se nas seguintes bibliotecas chave:

- **Selenium (selenium):** Núcleo da automação do navegador.

- **WebDriver Manager (webdriver-manager):** Gerencia o download e a configuração do ChromeDriver de forma automática.
- **Pandas (pandas):** Essencial para a organização dos dados em uma estrutura tabular (DataFrame) e para a exportação final para CSV.
- **Re (re):** Utilizada para limpeza de texto e manipulação de strings, principalmente na função clean_price.
- **Time e Random:** Para introduzir pausas estratégicas e aleatórias, tornando a navegação menos robotizada.

3.1. Funções Principais

- **setup_driver():** Configura e retorna uma instância do WebDriver (Chrome). É aqui que implementamos as principais técnicas de camuflagem:
    - User-Agent de navegador comum.
    - Desativação de flags de automação do Chrome.
    - Modificação de propriedades do navigator (como navigator.webdriver).
    - Opções como --disable-gpu e --headless (para execução em segundo plano).
- **clean_price(price_str_original):** Responsável por normalizar strings de preço. Remove símbolos monetários, trata pontos como separadores de milhar, converte vírgulas decimais para pontos e tenta interpretar formatos textuais (ex: "153 reais e 10 centavos") para retornar um valor float limpo. A precisão desta função foi crucial para garantir que os centavos fossem corretamente representados.
- **extract_review_count(count_text):** Uma função auxiliar simples para extrair o número de avaliações de strings formatadas (ex: "(3084 opiniões)").
- **scrape_mercado_livre_product(url, driver):** O coração do scraper. Para uma dada URL de produto:
    - Primeiro, tenta lidar com overlays de cookies que possam interceptar cliques.
    - Extrai título, preço (usando clean_price), vendedor (com múltiplos XPaths de fallback), nota e número de avaliações.

    - Implementa a lógica de encontrar e clicar no botão "Ver descrição completa" usando o seletor //a[@data-testid='action-collapsable-target'...] fornecido.
    - Coleta a descrição, já com quebras de linha substituídas por espaços.
    - Retorna um dicionário com os dados do produto.
- **search_mercado_livre_and_get_links(search_term, driver, max_links):**
    - Constrói a URL de busca.
    - Navega, tenta fechar banners de cookie na página de busca, e realiza scrolls.
    - Coleta os URLs dos produtos, filtrando por links válidos.
- **save_to_csv(data_list, filename):**
    - Converte a lista de dicionários de produtos em um DataFrame Pandas.
    - Formata a coluna de preço para o padrão "R$ XXX,YY" para o CSV final.
    - Salva os dados no arquivo CSV especificado, utilizando ; como separador.

3.2. Fluxo de Execução Principal (if __name__ == '__main__':)

1. Define-se uma lista de termos de busca (search_queries_list = ["Cartucho HP original", "Cartucho HP compativel", ...]).
2. O usuário informa quantos produtos devem ser raspados por cada termo.
3. O script itera sobre cada termo da lista: a. Inicia um driver para a busca, coleta os links dos produtos e fecha esse driver. b. Para cada link de produto obtido: i. Inicia um novo driver (para isolamento). ii. Chama scrape_mercado_livre_product para extrair os dados. iii. Adiciona os dados à lista principal. iv. Fecha o driver do produto. v. Pausas são aplicadas entre produtos e entre termos de busca.
4. Finalmente, todos os dados acumulados são salvos em um único arquivo CSV.

4. Campos Coletados (Estrutura do CSV)

O arquivo CSV gerado contém as seguintes colunas para cada produto:

- **link_anuncio:** URL da página do produto.
- **titulo:** Título do anúncio.
- **preco:** Preço formatado (ex: "R$ 153,10").
- **vendedor:** Nome do vendedor/loja.
- **avaliacao_nota:** Nota média de avaliação.
- **avaliacao_numero:** Quantidade de avaliações.
- **descricao:** Descrição do produto (quebras de linha substituídas por espaços).

Decidimos não persistir com a extração separada de "dados_extras" em colunas distintas, pois a estrutura dessa informação variava muito e já estava contida, de forma satisfatória para análise inicial, dentro do campo "descricao".

5. Desafios Enfrentados e Soluções Adotadas

O desenvolvimento deste scraper apresentou alguns desafios significativos:

- **Detecção e Bloqueio pelo Site:**
    - **Problema:** Nossa primeira abordagem mais direta com Selenium levou a um bloqueio temporário de IP, o que nos forçou a repensar a estratégia.
    - **Solução:** Implementamos um conjunto de técnicas de camuflagem (User-Agent, opções do Selenium para evitar detecção de automação, delays variáveis, e sessões de navegador isoladas para cada produto e busca). Isso se mostrou crucial para permitir coletas mais longas e estáveis. A tentativa de usar a API oficial do Mercado Livre foi descartada devido a complexidades de acesso e registro.
- **Conteúdo Dinâmico e Interações Complexas:**
    - **Problema:** Muitas informações, incluindo a descrição completa, não estavam imediatamente disponíveis no HTML inicial, sendo carregadas ou reveladas por JavaScript. O clique no botão "Ver descrição completa" era frequentemente interceptado por banners de consentimento de cookies.
    - **Solução:** O Selenium foi essencial. Usamos WebDriverWait para sincronizar com o carregamento dos elementos. Para o botão de descrição, utilizamos o seletor específico fornecido (//a[@data-testid='action-collapsable-target']) e implementamos uma rotina

para tentar fechar overlays de cookie antes do clique, além de tentar o clique via JavaScript como primeira opção.

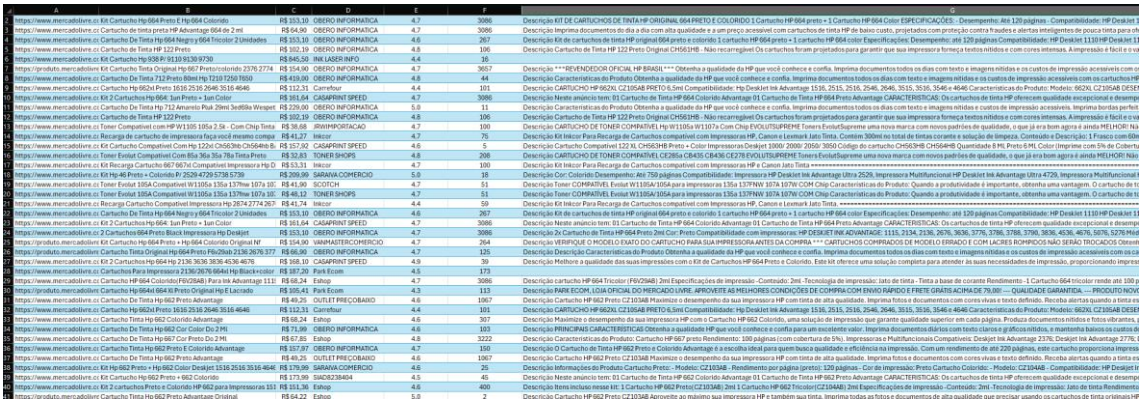
- **Variações no HTML (Seletores):**
    - **Problema:** A estrutura HTML das páginas de produto e de busca pode variar sutilmente ou mudar com o tempo.
    - **Solução:** Para campos como "vendedor" e "avaliações", utilizamos uma lista de XPaths alternativos. Os seletores foram refinados iterativamente com base nos testes e no HTML observado, como os exemplos que identifiquei para o nome do vendedor (<button class="ui-pdp-seller__link-trigger-button ...>).
- **Paginação dos Resultados de Busca:**
    - **Problema:** As páginas de busca apresentam resultados paginados.
    - **Solução Parcial:** O script atual simula alguns scrolls para baixo na página de resultados, o que carrega mais alguns produtos via "lazy loading". Uma navegação completa por múltiplas páginas de resultados não foi implementada nesta fase, mas seria uma melhoria para coletas mais extensivas. O número de links por busca é limitado pelo usuário.
- **Consistência dos Dados (Preço):**
    - **Problema:** A extração inicial do preço, especialmente os centavos, apresentava inconsistências, às vezes resultando em valores incorretos (ex: 153.0 em vez de 153.10).
    - **Solução:** A função clean_price foi significativamente aprimorada para lidar com diferentes formatos de string de preço, incluindo a interpretação de "X reais e Y centavos" comumente encontrada em atributos aria-label, garantindo a correta inclusão dos centavos no valor float final.
- **Erros de GPU e TensorFlow Lite:**
    - **Observação:** Durante a execução, logs do console como ERROR:gpu[...]Automatic fallback to software WebGL... e Created TensorFlow Lite XNNPACK delegate... foram notados. Estes são mensagens de baixo nível do navegador e não impactaram diretamente a funcionalidade do scraper, especialmente com a opção --disable-gpu ativa.

6. Evidências de Execução

- **Exemplo de Log de Console Durante a Execução:**

```
--- Raspando Produto 10 de 10 (Termo: 'Cartucho HP 662'): https://produto.mercadolivre.com.br/MLB-3927029467-cartucho-tinta-hp-662-preto-advantage-original-_JM ---

DevTools listening on ws://127.0.0.1:54170/devtools/browser/8e58dd6e-b715-4dd0-a3eb-6f4d52d3ac34
Pausando por 3.01s antes de carregar o produto...
Created TensorFlow Lite XNNPACK delegate for CPU.

Dados coletados para este produto:
  link_anuncio: https://produto.mercadolivre.com.br/MLB-3927029467-cartucho-tinta-hp-662-preto-advantage-original-_JM
  titulo: Cartucho Tinta Hp 662 Preto Advantage Original
  preco: 64.22
  vendedor: Eshop
  avaliacao_nota: 5.0
  avaliacao_numero: 2
  descricao: Descrição Cartucho HP 662 Preto CZ103AB Aproveite ao máximo sua impressora HP e também sua tinta. Imprima todas as fotos e documentos de alta qualidade que precisar usando os cartuchos de tinta ori
ginais HP, que ajudam a garantir que sua impressora HP ofereça os resultados consistentes que você espera. Conclua facilmente seus projetos. Alertas precisos podem ajudar a garantir que você não fique sem tinta
 na hora errada. Crie documentos atraentes com os cartuchos de tinta HP, projetados para oferecer cores intensas e texto nítido e preciso. - Rendimento Preto 120 paginas - Tecnologia de impressão: Jato de Tinta
 - Capacidade/Rendimento: 2 ml Impressoras Compatíveis: - HP Deskjet Ink Advantage 2516 - HP Deskjet Ink Advantage 3516 - HP Deskjet Ink Advantage 1516 - HP Deskjet Ink Advantage 3546 - HP Deskjet Ink Advantage
 2546 - HP Deskjet Ink Advantage 4646 - HP Deskjet Ink Advantage 2646 Ver descrição completa
Fechando navegador para produto 10...

Fim do processamento para o termo: 'Cartucho HP 662'
Dados salvos com sucesso em ml_produtos_hp_coletados_20250525_162728.csv (separador: ';')

--- Processo de scraping completo. Tempo total: 19.59 minutos ---
PS G:\Desktop\cartuchos_hp>
```

- **Exemplo do Arquivo CSV Gerado:**



7. Conclusão

O script de web scraping desenvolvido se mostrou uma ferramenta funcional para a coleta automatizada de dados de produtos do Mercado Livre. Através de um processo iterativo de desenvolvimento e superação de desafios como detecção pelo site e variações de HTML, conseguimos implementar uma solução que extrai os campos de dados solicitados e os organiza em um formato CSV útil para análises futuras. As técnicas de camuflagem e a extração robusta de informações, como o clique para expandir a descrição, foram essenciais para o sucesso da coleta. Embora a extração de "dados extras" de forma estruturada diretamente do HTML tenha sido abandonada em favor de uma descrição mais completa, o dataset gerado atende aos requisitos primários do projeto.