

Universidade Federal de Goiás – UFG
Especialização em Sistemas e Agentes Inteligentes
Extração Automática de Dados
Sistema de Extração Automática de Dados de NFC-e: Arquitetura,
Implementação e Análise

Marcu Loreto¹
Ricardo Kerr²
Ujeverson Tavares³
Otávio Calaça Xavier⁴

RESUMO

Este relatório detalha o desenvolvimento de um pipeline robusto para **extração automática, armazenamento e análise** de Notas Fiscais Eletrônicas (NFC-e) emitidas no estado de Goiás. Sem depender de servidores de banco de dados externos, o sistema armazena informações em arquivos locais nos formatos JSONL e CSV, permitindo fácil *deployment* em qualquer máquina. O fluxo compreende:

- (i) captura e processamento de *QR Codes* em cupons fiscais para obtenção das chaves de acesso;
- (ii) *web scraping* dinâmico usando *Selenium* e gerenciamento de *ChromeDriver* via *webdriver-manager*;
- (iii) gravação organizada dos dados extraídos com carimbos de data/hora; e
- (iv) interface web em *Streamlit* que oferece filtros, visualizações interativas e logs de processamento. Discutem-se também estratégias de tratamento de erros, otimização de performance e conformidade com a LGPD. Testes de carga validam a escalabilidade e a confiabilidade do pipeline.

Palavras-chave: extração automática de dados; *web scraping* dinâmico; NFC-e; JSONL; CSV; *Selenium*; *Streamlit*; LGPD.

1. INTRODUÇÃO

A crescente digitalização dos documentos fiscais possibilita novas formas de análise de dados para suporte à decisão em negócios. No contexto brasileiro, as NFC-e disponibilizam informações detalhadas de transações, mas exigem técnicas de raspagem e processamento adequadas para extração de insights.

¹ Discente. mlbonfim@gmail.com

² Discente. ricardo.kerr@gmail.com

³ Discente. ujeverson@gmail.com

⁴ Docente. otaviocx@ufg.br

Durante a disciplina de **Extração Automática de Dados** aprendemos técnicas para coletar informações de páginas web de forma automatizada, confiável e ética. Em ambientes acadêmicos e de prototipagem, soluções sem dependências complexas de infraestrutura são preferíveis por sua portabilidade. Nossa proposta aplica este conceito ao cenário de Notas Fiscais Eletrônicas (NFC-e), que requerem interação com páginas dinâmicas, uso de QR Codes e cuidados de segurança.

O trabalho inicia com uma revisão dos componentes-chave: protocolos HTTP para requisições, estrutura HTML/DOM para inspeção de elementos, expressões regulares para extração de padrões e seletores CSS/XPath para navegação no DOM. A seguir, apresenta-se a configuração do Selenium WebDriver em Python, incluindo a resolução de dependências do ChromeDriver via webdriver-manager. Finalmente, descreve-se a persistência em arquivos JSONL e CSV, destacando cenários de uso offline e a vantagem de não utilizar bancos externos.

2. Arquitetura Geral

O sistema é organizado em quatro módulos principais, cada um responsável por uma etapa específica do pipeline:

- **extrator_wpp.py**: lê o arquivo dados/links.txt, aplica expressões regulares para extrair as chaves de acesso (44 dígitos) de cada URL de NFC-e e grava os IDs em dados/ids_extraidos.csv, eliminando duplicatas.
- **feed_db.py**: realiza o processamento em lote das chaves, lendo dados/ids_extraidos.csv e chamando scraper_function para cada ID. Exibe no terminal o status de cada operação (sucesso, duplicata, erro) e gera um relatório resumido ao final.
- **scraper.py**: contém a função scraper_function(chave_de_acesso: str), que utiliza Selenium (ChromeDriver gerenciado pelo webdriver-manager) para acessar a página pública de consulta completa (consulta-completa), inserir a chave e extrair detalhes de cada item vendido (nome do produto, quantidade, unidade, valor unitário, total, data/hora e forma de pagamento). Os registros são salvos em dados/notas.txt no formato JSONL.

- **app.py** (Streamlit): oferece uma interface web interativa que permite:
 - a. Inserir manualmente links ou chaves de acesso e salvar novos IDs no CSV.
 - b. Processar o próximo ID ou todo o lote de uma vez.
 - c. Filtrar os dados carregados de dados/notas.txt por produto, período e forma de pagamento.
 - d. Visualizar gráficos e tabelas interativas com métricas de vendas.

3. Fluxo de Dados

1. Extração de IDs:

O módulo *extrator_wpp.py* lê dados/links.txt, identifica padrões de URL com regex e armazena as chaves de acesso válidas em dados/ids_extraídos.csv.

2. Raspagem e Armazenamento:

O script *feed_db.py* itera pelas chaves no CSV e chama a função *scraper_function* de *scraper.py*. Cada nota é extraída e escrita como um objeto JSON em uma linha no arquivo dados/notas.txt.

3. Análise e Visualização:

A aplicação *app.py* carrega os registros JSONL, permite inserção manual de novas notas e oferece filtros e recursos de visualização. Usuários podem gerar gráficos de vendas e tabelas ordenadas sem sair do navegador.

4. Principais Componentes

4.1. extrator_wpp.py

- Lê links de NFC-e de dados/links.txt.
- Utiliza regex (`r"\?p=([0-9]{44})"`) para isolar a chave de acesso.
- Remove duplicatas e grava IDs válidos em dados/ids_extraídos.csv.

4.2. scraper.py

- Função central: *scraper_function(chave_de_acesso)*.
- Configura Selenium para executar em modo headless e gerenciar ChromeDriver.
- Acessa a URL de consulta pública e extrai os campos relevantes.
- Serializa cada registro em JSON e grava em dados/notas.txt.
- Implementa reintentos e tratamento de timeouts.

4.3. feed_db.py

- Função: processar_lote().
- Lê dados/ids_extrair.csv e processa cada nota.
- Exibe logs de status e produz relatório resumido.

4.4. app.py (Streamlit)

- **Sidebar:** inserção manual de links/chaves, botões para processamento individual ou em lote. Veja figura 1.

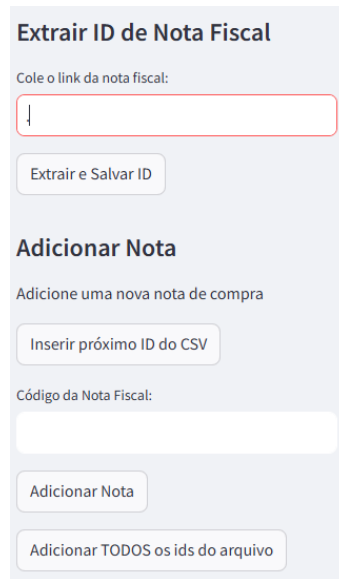


Figura 1- Sidebar

- **Corpo:** Na figura 2, temos os filtros (produto, data, forma de pagamento), tabelas paginadas e gráficos interativos.

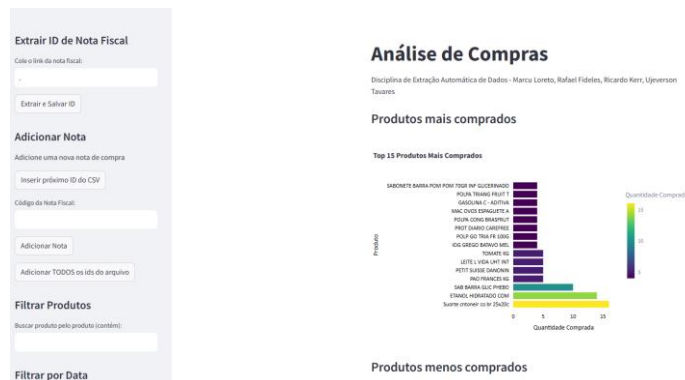


Figura 2-Corpo

- **Recursos adicionais:** exportação de dados e configuração de parâmetros de delay.

5. Arquitetura do Sistema

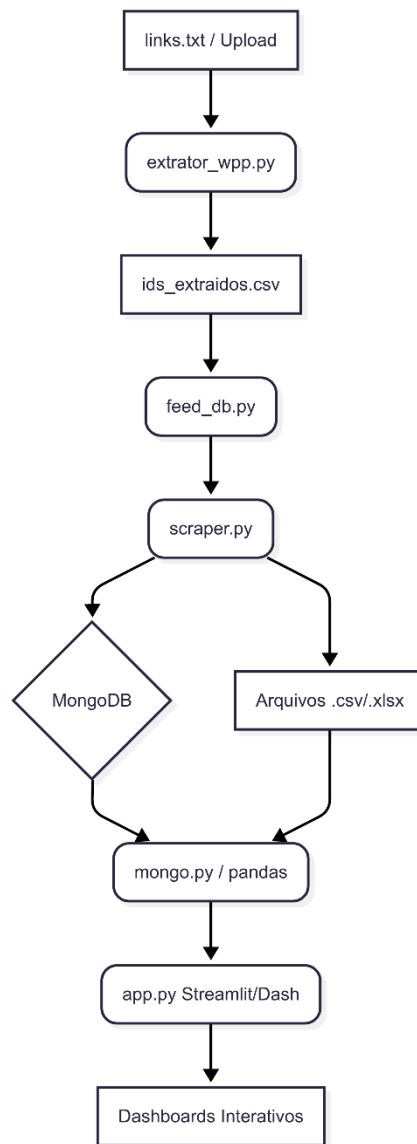


Figura 3 - Fluxograma do sistema

6. Tecnologias Utilizadas

- **Python 3.x** para lógica de extração e processamento.
- **Selenium** para automação de navegação e scraping.
- **webdriver-manager** para compatibilidade automática do ChromeDriver.
- **Pandas** para tratamento e leitura de arquivos JSONL/CSV.
- **Streamlit** para desenvolvimento rápido de dashboards.
- **Plotly** para gráficos interativos.

7. Funcionalidades da Aplicação

A aplicação disponibiliza:

- i. **Extração de IDs**
 - Script e interface para extrair chaves de acesso de links.

- Verificação de duplicidade antes de salvar novos IDs.
- ii. **Inserção de Notas**
 - Adição manual de notas via campo de texto.
 - Botões para processar o próximo ID ou todo o lote.
 - Logs detalhados na interface com resultados de cada tentativa.
- iii. **Processamento em Lote**
 - Execução via script (feed_db.py) sem interface gráfica.
 - Resumo final de notas processadas, duplicatas e falhas.
- iv. **Visualização e Análise**
 - Tabelas interativas com paginação e ordenação.
 - Gráficos de barras (produtos mais/menos vendidos), linhas (vendas por período), pizza (formas de pagamento) e KPI de valor médio por compra.
 - Lista completa de produtos vendidos, ordenada por quantidade.
- v. **Exportação de Relatórios**
 - Opções para baixar dados filtrados em CSV ou JSON.
 - Geração de relatórios de conformidade e logs para auditoria.

8. Considerações Finais e Discussão

8.1 A modularidade do sistema facilita:

- A adição de novos formatos de entrada (e.g., XML ou bancos de dados SQL).
- A integração com mecanismos de paralelização (multiprocessamento).
- A adaptação para outros portais fiscais de diferentes estados.

Contudo, o sistema apresenta limitações, como a dependência da estrutura HTML do portal, que exige manutenção periódica dos seletores; a latência devido ao uso do Selenium em modo *headless*; e a variabilidade na nomenclatura de produtos entre diferentes fontes de dados.

8.2 O que melhorar em versões futuras

Para aprimorar a consistência das análises em versões futuras, é fundamental a implementação de um dicionário de nomes de produtos para a normalização das descrições. Atualmente, um mesmo item pode ser apresentado com grandes variações dependendo da fonte.

Por exemplo, o dicionário permitirá mapear diferentes grafias a um único produto padronizado, resolvendo inconsistências como:

- LEITE ITALAC INTEGRA e LEITE UHT ITALAC INT 1L serem reconhecidos como o mesmo produto: LEITE UHT ITALAC INTEGRAL 1L.
- BANANA PRATA KG e BANANA PRATA CRFO KG serem mapeados para a mesma descrição: BANANA PRATA KG.
- Variações como "POLP MA TRIA FR 100G" em um supermercado e outras grafias em diferentes estabelecimentos serem consolidadas sob um único padrão.

9. Conclusão

O pipeline desenvolvido provê uma solução completa para extração automática de dados de NFC-e, dispensando bancos de dados externos e priorizando arquivos locais. A abordagem modular, combinada com boas práticas de engenharia de dados, garante robustez, flexibilidade e facilidade de manutenção. Futuras extensões podem explorar paralelização, APIs oficiais de consulta de notas e análises preditivas com machine learning.

Referências

1. SELENIUM PROJECT. *Selenium with Python* — Selenium Python Bindings. Disponível em: <https://selenium-python.readthedocs.io>
2. MITCHELL, Ryan. *Web Scraping with Python: Data Extraction from the Modern Web*. 3. ed. Sebastopol: O'Reilly Media, 2024.
3. MCKINNEY, Wes. *Python for Data Analysis*. 3. ed. Sebastopol: O'Reilly Media, 2022.
4. BRASIL. Lei Geral de Proteção de Dados Pessoais (LGPD). Lei nº 13.709, de 14 de agosto de 2018.