WEB SCRAPING: TÉCNICAS E APLICAÇÕES NA EXTRAÇÃO DE DADOS DO SERVIÇO DE DESEMPENHO PARLAMENTAR DE FORTALEZA[1]

Guilherme Vital Rabelo Silva[2]
Rodrigo Viana Castelo Branco[3]

RESUMO

Este trabalho apresenta o desenvolvimento e a implementação de um sistema automatizado de extração de dados (web scraping) para análise do Serviço de Desempenho Parlamentar (SDP) da Câmara Municipal de Fortaleza. O SDP constitui uma verba mensal destinada à gestão de despesas dos gabinetes dos vereadores, incluindo consultorias, correspondências, telefonia, combustível, publicidade e outros serviços. A pesquisa utiliza Python com as bibliotecas Selenium, Beautiful Soup, Pandas, NumPy e Matplotlib para automatizar a coleta, processamento e visualização de dados do portal de transparência, permitindo uma análise sistemática dos gas- tos parlamentares. A metodologia desenvolvida inclui três etapas principais: implementação do sistema de extração, processamento e validação dos dados coletados, e análise dos resulta- dos obtidos. O trabalho demonstra como técnicas de web scraping podem ser aplicadas para promover transparência na gestão pública, permitindo o monitoramento sistemático dos gastos e a identificação de padrões nas despesas parlamentares. Os resultados obtidos evidenciam a eficácia da solução implementada, fornecendo uma base para análises mais aprofundadas do uso dos recursos públicos. Esta pesquisa contribui tanto para o campo técnico do web scraping quanto para a promoção da transparência na administração pública.

Palavras-chave: web scraping; python; transparência pública; análise de dados; serviço de desempenho parlamentar.

ABSTRACT

This paper presents the development and implementation of an automated data extraction sys- tem (web scraping) for analyzing the Parliamentary Performance Service (SDP) of the Fortaleza City Council. The SDP is a monthly allowance for managing expenses of councilors' offices, including consultancies, correspondence, telephony, fuel, advertising, and other services. The research uses Python with Selenium, Beautiful Soup, Pandas, NumPy, and Matplotlib libraries to automate the collection, processing, and visualization of data from the transparency portal, allowing a systematic analysis of parliamentary expenses. The methodology developed includes three main stages: implementation of the extraction system, processing and validation of the collected data, and analysis of the results obtained. The work demonstrates how web scraping techniques can be applied to promote transparency in public management, allowing systematic monitoring of expenses and the identification of patterns in parliamentary expenses. The results obtained demonstrate the effectiveness of the implemented solution, providing a basis for more

[1]Artigo apresentado ao Curso de Sistemas de Informação da Faculdade Multiversa, como requisito parcial para obtenção do Título de Bacharel em Sistemas de Informação, Fortaleza-CE, 2025.

[2]Aluno do Curso de Sistemas de Informação da Faculdade Multiversa, e-mail: guiivital@gmail.com

[3]Orientador - Professor da Faculdade Multiversa, e-mail: rodrigo.viana@multiversa.com

in-depth analysis of the use of public resources. This research contributes both to the technical field of web scraping and to the promotion of transparency in public administration.

Keywords: web scraping; python; public transparency; data analysis; parliamentary performance service.

Data de submissão: 24 maio 2025.
Data de aprovação: 11 junho 2025.

1 INTRODUÇÃO

A era digital trouxe consigo uma quantidade massiva de dados disponíveis na web, criando novas oportunidades e desafios para a coleta e a análise desses dados. Neste contexto, o Web Scraping emerge como uma técnica fundamental que permite a extração automatizada de da- dos estruturados e não estruturados diretamente de páginas web, transformando informações dispersas em insights valiosos.

Como destaca Mitchell (2015, p. 10), "os web scrapers podem acessar lugares que as ferramentas de pesquisa tradicionais não conseguem", permitindo a coleta e análise de dados em uma escala que seria inviável manualmente. Esta capacidade torna-se especialmente relevante quando consideramos a necessidade de monitorar e analisar dados governamentais para promover a transparência pública.

O processo de Web Scraping consiste em três fases principais: análise do website, rastreamento (crawling) e organização dos dados. Esta técnica se diferencia da mineração de dados tradicional, pois foca na coleta automatizada, enquanto a mineração concentra-se na análise posterior utilizando técnicas estatísticas avançadas.

No contexto da transparência governamental, especificamente no monitoramento do desempenho parlamentar, o Web Scraping apresenta-se como uma ferramenta vital para extrair e ana- lisar informações sobre gastos e atividades dos representantes políticos. Esta análise sistemática pode contribuir significativamente para a fiscalização e otimização dos recursos públicos.

Como o uso de técnicas de Web Scraping pode facilitar a análise e promoção da transparência nos gastos públicos relacionados ao Serviço de Desempenho Parlamentar em Fortaleza?

2 OBJETIVO GERAL

Este trabalho tem como objetivo desenvolver um sistema automatizado de extração e análise de dados do Serviço de Desempenho Parlamentar de Fortaleza, utilizando técnicas de Web Scraping, visando promover maior transparência na gestão pública e compreensão dos padrões de gastos parlamentares.

3 OBJETIVOS ESPECÍFICOS

- Implementar um sistema de Web Scraping utilizando Python e suas bibliotecas especiali- zadas para coletar dados do portal de transparência;
- Desenvolver uma estrutura de armazenamento e organização dos dados coletados;
- Analisar padrões de gastos e comportamentos nos dados extraídos;
- Avaliar a eficiência e transparência na utilização dos recursos públicos.

4 JUSTIFICATIVA

A escolha de investigar a aplicação de técnicas de Web Scraping no contexto do Serviço de Desempenho Parlamentar em Fortaleza é justificada pela crescente necessidade de transparência na gestão pública. Em um cenário onde a confiança nas instituições governamentais é frequentemente questionada, o acesso à informação sobre gastos públicos torna-se crucial para promover a prestação de contas e responsabilidade. O Web Scraping se apresenta como uma solução eficaz para coletar dados do portal de transparência, permitindo transformar informações desestruturadas em dados abertos e acessíveis. Essa abordagem não apenas facilita o monitoramento das despesas dos vereadores, mas também empodera os cidadãos, proporcionando ferramentas para que possam fiscalizar e compreender como os recursos públicos estão sendo utilizados. Portanto, este trabalho visa contribuir para um ambiente mais transparente e responsável na administração pública, alinhando-se aos princípios da Lei de Acesso à Informação (LAI).

Este trabalho está estruturado em cinco seções principais. A Introdução apresenta o tema e a relevância do Web Scraping na promoção da transparência pública. Na seção de Objetivos, são delineados o objetivo geral e os objetivos específicos da pesquisa. A Metodologia descreve as etapas adotadas para a implementação do sistema de extração e análise dos dados, detalhando as ferramentas utilizadas, como Python e suas bibliotecas especializadas: Selenium, Beautiful Soup, Pandas, NumPy e Matplotlib. A seção de Desenvolvimento e Resultados apresenta os resultados obtidos com a aplicação do Web Scraping, incluindo a análise dos padrões de gastos dos vereadores e visualizações gráficas das informações extraídas. Por fim, as Considerações Finais discutem as implicações dos resultados e sugerem direções para futuras pesquisas.

5 REFERENCIAL TEÓRICO

Este trabalho está organizado da seguinte forma: inicialmente, é apresentada uma contextualização sobre web scraping e suas aplicações. Em seguida, são discutidas as principais ferramentas e bibliotecas utilizadas para implementação da técnica. Por fim, aborda-se a importância da transparência pública e como o web scraping pode contribuir para este fim através da análise dos dados do Serviço de Desempenho Parlamentar de Fortaleza.

5.1 Aplicações do Web Scraping

Apesar de ainda não ser amplamente conhecida no Brasil e no meio acadêmico nacional, a técnica de web scraping tem sido destacada por diversos autores devido ao seu grande potencial e suas aplicações práticas no dia a dia. A literatura especializada apresenta diferentes perspectivas sobre essa tecnologia e suas possibilidades de aplicação. Mitchell destaca o potencial transformador desta técnica ao afirmar que "[...] os web scrapers podem acessar lugares que as ferramentas de pesquisa tradicionais não conseguem"(Mitchell, 2015, p. 10), permitindo a coleta e análise de dados em uma escala que seria inviável manualmente. Isso é particularmente relevante em contextos como o monitoramento de preços, onde um scraper pode compilar informações de diferentes sites para gerar comparações significativas. No campo da evolução tecnológica, Calo observa que "a evolução da computação, nos últimos anos, incentivou a cri- ação de grandes volumes de dados em todas as áreas de conhecimento”(Calò, 2014, p. 9). O autor enfatiza o esforço significativo para digitalizar e organizar esses dados, destacando a necessidade de novas técnicas de análise para extrair significados valiosos a partir de grandes quantidades de informações brutas. Numa perspectiva mais aplicada, Khder analisa o uso do web scraping em setores específicos como negócios, finanças e marketing, demonstrando como a coleta automatizada de dados pode fornecer insights competitivos (Khder, 2021). A transparência governamental representa outra área onde esta técnica se mostra vital, uma vez que a coleta sistemática de dados sobre gastos públicos permite uma análise crítica das despesas dos representantes políticos, promovendo a responsabilidade e o controle social. No contexto específico deste trabalho, o web scraping será utilizado para coletar dados sobre os gastos dos vereadores de Fortaleza, contribuindo diretamente para a promoção da transparência na

administração pública municipal.

5.2 Ferramentas e Bibliotecas

Para implementar o Web Scraping, utilizaremos a linguagem Python, reconhecida por sua simplicidade e pela vasta gama de bibliotecas que facilitam a extração e manipulação de dados. Neste trabalho, as bibliotecas Selenium e Pandas serão as principais ferramentas utilizadas, complementadas por outras bibliotecas especializadas.

- **Selenium:** Esta biblioteca é essencial para automatizar a interação com navegadores web. O Selenium permite simular ações humanas, como cliques em botões, preenchimento de formulários e navegação entre páginas. Isso é particularmente útil para acessar páginas dinâmicas que carregam conteúdo via JavaScript, onde os dados não estão imediatamente disponíveis no HTML da página inicial. Com o Selenium, podemos programar scripts que realizam essas interações automaticamente, coletando dados de forma eficiente e em larga escala. Além disso, o Selenium é compatível com diversos navegadores, o que proporciona flexibilidade na execução dos scripts.

- **Beautiful Soup 4:** Conforme a documentação oficial (Beautiful Soup, 2024), o Beautiful Soup é uma biblioteca Python para extrair dados de arquivos HTML e XML. Ela fornece métodos simples para navegar, pesquisar e modificar a árvore de análise, criando um kit de ferramentas ideal para projetos de web scraping. A biblioteca trabalha com diferentes parsers, oferecendo maneiras idiomáticas de navegar, pesquisar e modificar a árvore de análise. Ela automaticamente converte documentos para Unicode e, ao salvar, para UTF- 8, facilitando o tratamento de caracteres especiais em diferentes idiomas.

- **Pandas:** Após a coleta dos dados, a biblioteca Pandas desempenha um papel crucial na manipulação e análise dos dados extraídos. Pandas oferece estruturas de dados como DataFrames, que permitem organizar os dados em tabelas de forma intuitiva e eficiente. Com essa biblioteca, podemos realizar operações como filtragem, agregação e transformação dos dados coletados em um formato adequado para análise posterior. A capacidade do Pandas de lidar com grandes volumes de dados torna-o uma ferramenta indispensável para a análise dos gastos parlamentares.

- **NumPy:** De acordo com sua documentação oficial (NumPy, 2024), o NumPy é a biblio- teca fundamental para computação científica em Python. Ela fornece suporte para arrays multidimensionais e matrizes, juntamente com uma grande coleção de funções matemáticas de alto nível para operar nestes arrays. No contexto deste trabalho, o NumPy é utilizado para operações matemáticas e estatísticas sobre os dados extraídos, possibilitando cálculos eficientes sobre grandes conjuntos de dados numéricos, como médias, desvios-padrão e correlações entre diferentes categorias de gastos.

- **Matplotlib:** Conforme descrito em sua documentação (Matplotlib, 2024), o Matplotlib é uma biblioteca abrangente para criação de visualizações estáticas, animadas e interativas em Python. Ela produz figuras de qualidade para publicação em diversos formatos e ambientes. No presente estudo, o Matplotlib é empregado para gerar gráficos e visualizações que facilitam a compreensão dos padrões de gastos, tendências temporais e distribuições de recursos, transformando dados numéricos complexos em representações visuais intuitivas.

A combinação dessas bibliotecas permite uma abordagem robusta para o Web Scraping: enquanto o Selenium e o Beautiful Soup facilitam a coleta automatizada de dados de sites complexos, o Pandas, NumPy e Matplotlib possibilitam a organização, análise e visualização desses dados de maneira eficaz. Essa metodologia não apenas promove maior transparência na gestão pública, mas também permite uma compreensão mais profunda dos padrões de gastos dos vereadores.

5.3 Importância da Transparência Pública

A transparência na gestão dos recursos públicos é fundamental para garantir a confiança da população nas instituições governamentais. A análise dos dados do SDP permitirá identificar padrões nos gastos dos vereadores, promovendo uma maior compreensão sobre como os recursos são alocados e utilizados. Segundo Krotov *et al.,* (2020), a transparência não apenas fortalece a democracia, mas também incentiva um ambiente onde os cidadãos se sentem empoderados para questionar e exigir responsabilidade dos seus representantes.

6 METODOLOGIA

Este trabalho teve como objetivo extrair, analisar e apresentar graficamente os dados do Serviço de Desempenho Parlamentar (SDP) da Câmara Municipal de Fortaleza. O SDP constituiu uma verba mensal destinada ao gerenciamento das despesas operacionais dos gabinetes dos vereadores, abrangendo:

- Consultorias
- Correspondências
- Telefonia
- Combustível
- Impressos
- Publicidade
- Passagens aéreas
- Fretamento de veículos automotores

6.1 Coleta e Estruturação dos Dados

A extração dos dados foi realizada através da técnica de web scraping, utilizando Python com um conjunto completo de bibliotecas especializadas. O Selenium foi empregado para navegação automatizada em páginas dinâmicas, o Beautiful Soup para parsing e extração de dados do HTML, o Pandas para estruturação e manipulação dos dados coletados, o NumPy para operações matemáticas e estatísticas, e o Matplotlib para visualização gráfica dos resultados. Os dados foram extraídos em uma única operação realizada no último semestre de 2024, cobrindo os gastos parlamentares desse período. O código fonte completo do projeto está disponível em um repositório público (Silva, 2025), permitindo a replicação da metodologia e a verificação dos procedimentos utilizados.

6.2 Tratamento e Validação dos Dados

Para garantir a qualidade e consistência dos dados, foram implementadas as seguintes etapas:

- Remoção de dados duplicados através de verificação de registros únicos
- Tratamento de dados ausentes utilizando técnicas de imputação quando apropriado
- Padronização de formatos monetários e datas
- Verificação de consistência através de validação cruzada com relatórios oficiais

- Identificação e tratamento de outliers através de análise estatística

6.3 Análise dos Dados

A análise foi realizada utilizando técnicas estatísticas descritivas e comparativas, incluindo:

- Análise de distribuição dos gastos por categoria
- Comparação de gastos entre vereadores
- Evolução temporal das despesas
- Identificação de padrões de utilização dos recursos

Os dados foram estruturados em arquivos CSV contendo campos como: identificação do parlamentar, data da despesa, categoria do gasto, valor, fornecedor e descrição detalhada.

7 DESENVOLVIMENTO E RESULTADOS

7.1 Implementação do Sistema de Extração

O desenvolvimento deste trabalho foi estruturado em três etapas principais para garantir uma extração e análise eficiente dos dados do Serviço de Desempenho Parlamentar (SDP) de Fortaleza:

- **Coleta Automatizada de Dados**: Implementação de um sistema robusto utilizando Selenium para navegar pelo portal de transparência e extrair dados em larga escala.
- **Processamento e Validação dos Dados**: Tratamento dos dados extraídos, com normalização de formatos, identificação de inconsistências e estruturação para análise.
- **Análise e Visualização**: Geração de insights a partir dos dados coletados, com produção de gráficos e estatísticas descritivas para compreensão dos padrões de gastos.

Essas etapas foram implementadas em módulos distintos no sistema, permitindo a execução independente de cada fase do processo, bem como sua adaptação para diferentes contextos de análise de dados públicos.

7.2 Coleta Automatizada de Dados

Nesta etapa, foi implementado o sistema de Web Scraping utilizando Python, especificamente com as bibliotecas Selenium para automação de navegador e Pandas para estruturação de dados. As atividades realizadas incluíram:

- Implementação do Web Scraping: Utilizou-se Selenium para automatizar a navegação nas páginas do portal de transparência da Câmara Municipal de Fortaleza, identificando e tratando elementos dinâmicos da página.
- Desenvolvimento de Mecanismos Robustos: Foram criadas funções específicas para lidar com a complexidade do site, como (*extrair_vereadores_pagina*, *extrair_gastos_vereador* e *verificar_numero_paginas*), que garantiram uma coleta eficiente mesmo em caso de alterações na estrutura do portal.
- Tratamento de Exceções: Implementação de estruturas try-except para lidar com falhas na conexão, carregamento de páginas ou alterações no HTML, garantindo que o processo de extração fosse resiliente.

- Gerenciamento Temporal: Configuração de esperas explícitas (WebDriverWait) e implíci- tas (time.sleep) para sincronizar adequadamente o carregamento dos elementos dinâmicos antes da extração.
- Estruturação dos Dados: Os dados coletados foram organizados em DataFrames com timestamp de extração, facilitando o controle de versões e análises temporais.

Código 1: Código para extrair os dados

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from bs4 import BeautifulSoup
import pandas as pd
import time
import os
from datetime import datetime
import re

#configuracao do selenium
def configurar_driver():
    options = Options()
    #options.add_argument("--headless")
    options.add_argument("--no-sandbox")
    options.add_argument("--disable-dev-shm-usage")

    driver = webdriver.Chrome(options=options)
    return driver

#analisa estrutura da pagina
def analisar_pagina(soup, nome_vereador=None):
    print(f"\n{'='*50}")
```

```
    print(f"ANÁLISE DA PÁGINA{' de ' + nome_vereador if nome_vereador
        else ''}")
    print(f"{'='*50}")

    tabelas = soup.find_all('table')
    print(f"Total de tabelas encontradas: {len(tabelas)}")

    for i, tabela in enumerate(tabelas):
        print(f"\nTabela {i+1}:")
        print(f"Classes: {tabela.get('class', 'Sem classes')}")

        thead = tabela.find('thead')
        if thead:
            colunas = [th.text.strip() for th in thead.find_all('th')]
            print(f"Colunas: {colunas}")
        else:
            print("Cabeçalho não encontrado")

        tbody = tabela.find('tbody')
        if tbody:
            linhas = tbody.find_all('tr')
            print(f"Número de linhas: {len(linhas)}")
            if linhas:
                primeira_linha = linhas[0]
                colunas = primeira_linha.find_all('td')
                if colunas:
                    valores = [col.text.strip() for col in colunas]
                    print(f"Exemplo de linha: {valores}")
        else:
            print("Corpo da tabela não encontrado")

    print(f"{'='*50}\n")

#converte texto para numero
def texto_para_numero(texto):
    if not texto or texto.strip() == '':
        return 0.0

    texto = re.sub(r'[R$\s]', '', texto)
    texto = texto.replace('.', '').replace(',', '.')

    try:
        return float(texto)
    except ValueError:
        print(f"Erro ao converter '{texto}' para número")
        return 0.0

#encontra tabela de gastos
def encontrar_tabela_gastos(soup):
    for tabela in soup.find_all('table'):
        thead = tabela.find('thead')
        if thead:
            cabecalhos = [th.text.strip() for th in thead.find_all('th')]
            cabecalhos_esperados = ["Especificação", "Credor", "CNPJ"]
            if all(cabecalho in cabecalhos for cabecalho in
                cabecalhos_esperados):
                return tabela
```

```
    tabela = soup.find('table', {'class': 'table table-striped'})
    if tabela:
        return tabela

    tabela = soup.find('table', {'class': 'table'})
    if tabela:
        return tabela

    tabelas = soup.find_all('table')
    if tabelas:
        tabela_mais_provavel = None
        max_linhas = 0

        for tabela in tabelas:
            tbody = tabela.find('tbody')
            if tbody:
                linhas = tbody.find_all('tr')
                if len(linhas) > max_linhas:
                    max_linhas = len(linhas)
                    tabela_mais_provavel = tabela

        return tabela_mais_provavel

    return None

#extrai gastos do vereador
def extrair_gastos_vereador(driver, ano, mes, nome_vereador, link):
    try:
        print(f"      Acessando link: {link}")
        driver.get(link)

        WebDriverWait(driver, 10).until(
            EC.presence_of_element_located((By.TAG_NAME, "table"))
        )

        html_source = driver.page_source
        soup = BeautifulSoup(html_source, 'html.parser')

        analisar_pagina(soup, nome_vereador)

        tabela = encontrar_tabela_gastos(soup)

        if not tabela:
            print(f"Tabela não encontrada para {nome_vereador}")
            return []

        dados_gastos = []

        tbody = tabela.find('tbody')
        if not tbody:
            print(f"Corpo da tabela não encontrado para {nome_vereador}")
            return []

        for linha in tbody.find_all('tr'):
            colunas = linha.find_all('td')
```

```
            if len(colunas) >= 5:
                try:
                    especificacao = colunas[0].text.strip()
                    credor = colunas[1].text.strip()
                    cnpj = colunas[2].text.strip()

                     valor_total = texto_para_numero(colunas[3].text.strip
                        ())
                    saldo = texto_para_numero(colunas[4].text.strip())

                    dados_gastos.append({
                        'Ano': ano,
                        'Mês': mes,
                        'Vereador': nome_vereador,
                        'Especificação': especificacao,
                        'Credor': credor,
                        'CNPJ': cnpj,
                        'Valor Total': valor_total,
                        'Saldo': saldo
                    })

                except Exception as e:
                     print(f"Erro ao processar linha para {nome_vereador}:
                         {str(e)}")
                    print(f"Conteúdo da linha: {[col.text.strip() for col
                        in colunas]}")

        if not dados_gastos:
            print(f"Nenhum dado extraído para {nome_vereador}")

            print("Tentando extração manual...")

            linhas_selenium = driver.find_elements(By.CSS_SELECTOR, "
                table.table tbody tr")
            if linhas_selenium:
                print(f"Encontradas {len(linhas_selenium)} linhas via
                    Selenium")

                for linha in linhas_selenium:
                    try:
                        colunas = linha.find_elements(By.TAG_NAME, "td")
                        if len(colunas) >= 5:
                            especificacao = colunas[0].text.strip()
                            credor = colunas[1].text.strip()
                            cnpj = colunas[2].text.strip()
                            valor_total = texto_para_numero(colunas[3].
                                text.strip())
                            saldo = texto_para_numero(colunas[4].text.
                                strip())

                            dados_gastos.append({
                                'Ano': ano,
                                'Mês': mes,
                                'Vereador': nome_vereador,
                                'Especificação': especificacao,
                                'Credor': credor,
                                'CNPJ': cnpj,
```

```
                                    'Valor Total': valor_total,
                                    'Saldo': saldo
                                })
                    except Exception as e:
                        print(f"Erro na extração manual: {str(e)}")

        print(f"      Extraídos {len(dados_gastos)} registros de gastos para {nome_vereador}")
        return dados_gastos

    except Exception as e:
        print(f"Erro ao extrair gastos de {nome_vereador}: {str(e)}")
        return []

#verifica numero de paginas
def verificar_numero_paginas(driver, ano, mes):
    url = f"https://portaltransparencia.cmfor.ce.gov.br/despesas/sdp?&page=1&ano={ano}&mes={mes}&nome="
    driver.get(url)

    WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.CLASS_NAME, "pagination"))
    )

    try:
        paginacao = driver.find_element(By.CLASS_NAME, "pagination")
        links_paginas = paginacao.find_elements(By.TAG_NAME, "a")

        if links_paginas:
            numeros_paginas = []
            for link in links_paginas:
                texto = link.text.strip()
                if texto.isdigit():
                    numeros_paginas.append(int(texto))

            if numeros_paginas:
                return max(numeros_paginas)

        tabela = driver.find_element(By.TAG_NAME, "tbody")
        if tabela:
            linhas = tabela.find_elements(By.TAG_NAME, "tr")
            if linhas:
                return 1

    except Exception as e:
        print(f"Erro ao verificar número de páginas: {str(e)}")

    return 3

#extrai vereadores de uma pagina
def extrair_vereadores_pagina(driver, ano, mes, pagina):
    url = f"https://portaltransparencia.cmfor.ce.gov.br/despesas/sdp?&page={pagina}&ano={ano}&mes={mes}&nome="

    print(f"    Acessando URL: {url}")
    driver.get(url)

```

```
    try:
        WebDriverWait(driver, 15).until(
            EC.presence_of_element_located((By.TAG_NAME, "tbody"))
        )

        time.sleep(2)

        soup = BeautifulSoup(driver.page_source, 'html.parser')
        tabela = soup.find('tbody')

        if not tabela:
            print(f"    AVISO: Tabela não encontrada na página {pagina} do mês {mes}/{ano}")
            return [], []

        vereadores = []
        links = []

        for linha in tabela.find_all('tr'):
            colunas = linha.find_all('td')
            if len(colunas) >= 4:
                ano_vereador = colunas[0].text.strip()
                mes_vereador = colunas[1].text.strip()
                nome_vereador = colunas[2].text.strip()
                link_element = colunas[3].find('a')

                if link_element and 'href' in link_element.attrs:
                    link = "https://portaltransparencia.cmfor.ce.gov.br" + link_element['href']
                    vereadores.append(nome_vereador)
                    links.append(link)

        print(f"    Encontrados {len(vereadores)} vereadores na página {pagina}")
        return vereadores, links

    except Exception as e:
        print(f"    ERRO ao extrair vereadores da página {pagina}: {str(e)}")
        return [], []

#coleta dados dos vereadores
def coletar_dados_vereadores(meses, ano):
    driver = configurar_driver()

    todos_gastos = []

    try:
        for mes in meses:
            print(f"\nColetando dados de {mes}/{ano}...")

            try:
                num_paginas = verificar_numero_paginas(driver, ano, mes)
                print(f" Detectadas {num_paginas} páginas para o mês {mes}/{ano}")
            except Exception as e:
                print(f" Erro ao verificar número de páginas: {str(e)}")
```

```
                num_paginas = 3
                print(f" Usando valor padrão: {num_paginas} páginas")

            gastos_mes = []

            for pagina in range(1, num_paginas + 1):
                print(f"\n Processando página {pagina} de {num_paginas}
                    para o mês {mes}/{ano}...")

                vereadores, links = extrair_vereadores_pagina(driver, ano
                    , mes, pagina)

                if not vereadores:
                    print(f" AVISO: Nenhum vereador encontrado na página
                        {pagina} do mês {mes}/{ano}")
                    continue

                print(f" Encontrados {len(vereadores)} vereadores na pá
                    gina {pagina}")

                for i, (vereador, link) in enumerate(zip(vereadores,
                    links)):
                    print(f"    Processando vereador {i+1}/{len(
                        vereadores)}: {vereador}")
                    gastos = extrair_gastos_vereador(driver, ano, mes,
                        vereador, link)
                    gastos_mes.extend(gastos)

                    time.sleep(1.5)

            print(f" Total de {len(gastos_mes)} registros de gastos
                extraídos para o mês {mes}/{ano}")
            todos_gastos.extend(gastos_mes)

        df = pd.DataFrame(todos_gastos)

        if not os.path.exists('dados'):
            os.makedirs('dados')

        data_atual = datetime.now().strftime("%Y%m%d_%H%M%S")
        caminho_arquivo = f"dados/gastos_vereadores_{ano}_{data_atual}.
            csv"
        df.to_csv(caminho_arquivo, index=False, encoding='utf-8-sig')

        print(f"\nDados salvos com sucesso em {caminho_arquivo}")
        print(f"Total de {len(todos_gastos)} registros extraídos para
            todos os meses")
        return caminho_arquivo

    finally:
        driver.quit()

#execucao principal
```

```
338 if __name__ == "__main__":
339     meses_semestre = list(range(7, 13))
340     ano = 2024
341
```

```
    caminho_arquivo = coletar_dados_vereadores(meses_semestre, ano)
    print(f"Webscraping concluído. Dados salvos em {caminho_arquivo}")
```

Fonte: Elaborado pelo autor.

7.3 Processamento e Validação

Após a coleta, os dados passaram por um rigoroso processo de tratamento e validação implementado no módulo de análise, para garantir sua qualidade e consistência. As etapas incluíram:

- Padronização dos Formatos Monetários: Utilização de expressões regulares para remover caracteres especiais (R$, pontos) e converter vírgulas em pontos decimais, normalizando os valores monetários para cálculos precisos.
- Tratamento de Dados Ausentes: Implementação de verificações para identificar e tratar valores nulos, especialmente em campos críticos como valores e identificação de credores.
- Agregação e Sumarização: Criação de estruturas agregadas por diferentes dimensões de análise (vereador, mês, categoria de despesa, credor) para facilitar as análises comparati- vas.
- Validação Estatística: Cálculo de estatísticas descritivas (média, soma, desvio padrão) para identificar outliers e padrões anormais nos dados.
- Visualização para Validação: Geração de gráficos exploratórios para verificação visual de inconsistências e confirmação da integridade dos dados.

Após a extração, os dados são processados e analisados utilizando o Pandas, que permite manipulação avançada e geração de insights estatísticos:

Código 2: Código para processamento e análise dos dados.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os
import glob
from datetime import datetime
import matplotlib.ticker as mtick

#formatacao de numeros
pd.options.display.float_format = 'R$ {:.2f}'.format
plt.style.use('ggplot')
plt.rcParams['font.family'] = 'sans-serif'
plt.rcParams['font.sans-serif'] = ['Arial', 'DejaVu Sans']
plt.rcParams['axes.unicode_minus'] = False

#carrega arquivo mais recente
def carregar_dados():
    arquivos = glob.glob('dados/gastos_vereadores_*.csv')

    if not arquivos:
        print("Nenhum arquivo de dados encontrado. Execute primeiro o
            script de scraping.")
```

```
            return None

        arquivo_mais_recente = max(arquivos, key=os.path.getmtime)
        print(f"Carregando dados do arquivo: {arquivo_mais_recente}")

        df = pd.read_csv(arquivo_mais_recente, encoding='utf-8-sig')
        return df

    #formata valores em reais
    def formatar_valor_reais(x, pos):
        return f'R$ {x:,.0f}'

    #gera estatisticas basicas
    def gerar_estatisticas_basicas(df):
        if df is None or df.empty:
            print("Sem dados para análise.")
            return

        print("\n=== ESTATÍSTICAS BÁSICAS ===")

        gastos_por_mes = df.groupby('Mês')['Valor Total'].sum().
            reset_index()
        print("\nTotal de gastos por mês:")
        for _, row in gastos_por_mes.iterrows():
            print(f"Mês {int(row['Mês'])}: R$ {row['Valor Total']:,.2f}")

        gastos_por_vereador = df.groupby('Vereador')['Valor Total'].sum()
            .sort_values(ascending=False)
        print("\nTop 10 vereadores com maiores gastos:")
        for vereador, valor in list(gastos_por_vereador.items())[:10]:
            print(f"{vereador}: R$ {valor:,.2f}")

        gastos_por_categoria = df.groupby('Especificação')['Valor Total'
            ].sum().sort_values(ascending=False)
        print("\nPrincipais categorias de gastos:")
        for categoria, valor in list(gastos_por_categoria.items())[:5]:
            print(f"{categoria}: R$ {valor:,.2f}")

        gastos_por_credor = df.groupby('Credor')['Valor Total'].sum().
            sort_values(ascending=False)
        print("\nPrincipais credores:")
        for credor, valor in list(gastos_por_credor.items())[:5]:
            print(f"{credor}: R$ {valor:,.2f}")

    #gera graficos de analise
    def gerar_graficos(df):
        if df is None or df.empty:
            print("Sem dados para gerar gráficos.")
            return

        if not os.path.exists('graficos'):
            os.makedirs('graficos')

        data_atual = datetime.now().strftime("%Y%m%d_%H%M%S")

        plt.rcParams['figure.figsize'] = (12, 8)
        plt.style.use('ggplot')
```

```
        sns.set_palette("colorblind")

        formato_real = mtick.FuncFormatter(formatar_valor_reais)

        #grafico de gastos por mes
        plt.figure(figsize=(12, 7))
        gastos_por_mes = df.groupby('Mês')['Valor Total'].sum().
            reset_index()
        ax = sns.barplot(x='Mês', y='Valor Total', data=gastos_por_mes,
            palette='viridis')
        plt.title('Total Gasto por Mês', fontsize=18)
        plt.xlabel('Mês', fontsize=14)
        plt.ylabel('Valor Total', fontsize=14)
        plt.xticks(rotation=0, fontsize=12)
        plt.yticks(fontsize=12)
        ax.yaxis.set_major_formatter(formato_real)
        for i, p in enumerate(ax.patches):
            valor = gastos_por_mes.iloc[i]['Valor Total']
            ax.annotate(f'R$ {valor:,.0f}',
                        (p.get_x() + p.get_width() / 2., p.get_height()),
                        ha = 'center', va = 'bottom', fontsize=11)
        plt.tight_layout()
        plt.savefig(f'graficos/gastos_por_mes_{data_atual}.png', dpi=300)
        plt.close()

        #ranking de vereadores
        plt.figure(figsize=(14, 10))
        top_vereadores = df.groupby('Vereador')['Valor Total'].sum().
            nlargest(15).reset_index()
        ax = sns.barplot(x='Valor Total', y='Vereador', data=
            top_vereadores, palette='viridis')
        plt.title('Total por Vereador (Top 15)', fontsize=18)
        plt.xlabel('Valor Total', fontsize=14)
        plt.ylabel('Vereador', fontsize=14)
        plt.yticks(fontsize=12)
        plt.xticks(fontsize=12)
        ax.xaxis.set_major_formatter(formato_real)
        for i, p in enumerate(ax.patches):
            valor = top_vereadores.iloc[i]['Valor Total']
            ax.annotate(f'R$ {valor:,.0f}',
                        (p.get_width(), p.get_y() + p.get_height()/2),
                        ha = 'left', va = 'center', fontsize=11)
        plt.tight_layout()
        plt.savefig(f'graficos/total_por_vereador_{data_atual}.png', dpi
            =300)
        plt.close()

        #distribuicao por categoria
        plt.figure(figsize=(14, 10))
        top_categorias = df.groupby('Especificação')['Valor Total'].sum()
            .nlargest(7).reset_index()
        total = top_categorias['Valor Total'].sum()
        top_categorias['Porcentagem'] = (top_categorias['Valor Total'] /
            total) * 100

        plt.pie(top_categorias['Valor Total'],
                labels=[f"{cat} (R$ {val:,.0f})" for cat, val in zip(
```

```
                    top_categorias['Especificação'], top_categorias['Valor
                     Total'])],
                autopct='%1.1f%%', startangle=90, shadow=True,
                wedgeprops={'edgecolor': 'white', 'linewidth': 1.5})
        plt.axis('equal')
        plt.title('Distribuição de Gastos por Categoria (Top 7)',
            fontsize=18)
        plt.tight_layout()
        plt.savefig(f'graficos/distribuicao_por_categoria_{data_atual}.
            png', dpi=300)
        plt.close()

        #principais credores
        plt.figure(figsize=(14, 8))
        top_credores = df.groupby('Credor')['Valor Total'].sum().nlargest
            (10).reset_index()
        ax = sns.barplot(x='Valor Total', y='Credor', data=top_credores,
            palette='viridis')
        plt.title('Top 10 Credores', fontsize=18)
        plt.xlabel('Valor Total', fontsize=14)
        plt.ylabel('Credor', fontsize=14)
        plt.yticks(fontsize=12)
        plt.xticks(fontsize=12)
        ax.xaxis.set_major_formatter(formato_real)
        for i, p in enumerate(ax.patches):
            valor = top_credores.iloc[i]['Valor Total']
            ax.annotate(f'R$ {valor:,.0f}',
                        (p.get_width(), p.get_y() + p.get_height()/2),
                        ha = 'left', va = 'center', fontsize=11)
        plt.tight_layout()
        plt.savefig(f'graficos/top_credores_{data_atual}.png', dpi=300)
        plt.close()

        #evolucao de gastos
        plt.figure(figsize=(14, 8))
        top5_vereadores = df.groupby('Vereador')['Valor Total'].sum().
            nlargest(5).index
        df_top5 = df[df['Vereador'].isin(top5_vereadores)]

        pivot_df = df_top5.pivot_table(index='Mês', columns='Vereador',
            values='Valor Total', aggfunc='sum')

        ax = pivot_df.plot(marker='o', linewidth=2.5, markersize=8)

        plt.title('Evolução de Gastos por Vereador (Top 5)', fontsize=18)
        plt.xlabel('Mês', fontsize=14)
        plt.ylabel('Valor Total', fontsize=14)
        plt.grid(True, linestyle='--', alpha=0.7)
        plt.legend(title='Vereador', loc='best', fontsize=12)
        plt.xticks(fontsize=12)
        plt.yticks(fontsize=12)
        ax.yaxis.set_major_formatter(formato_real)
        plt.tight_layout()
        plt.savefig(f'graficos/evolucao_gastos_{data_atual}.png', dpi
            =300)
        plt.close()

```

```
        #correlacao gastos e saldo
        plt.figure(figsize=(14, 8))
        correlacao_df = df.groupby('Vereador').agg({
            'Valor Total': 'sum',
            'Saldo': 'mean'
        }).reset_index()

        plt.figure(figsize=(14, 8))
        ax = sns.scatterplot(data=correlacao_df, x='Valor Total', y='
            Saldo',
                             size='Valor Total', sizes=(100, 500), alpha=0.7)

        for i, row in correlacao_df.iterrows():
            plt.annotate(row['Vereador'],
                         (row['Valor Total'], row['Saldo']),
                         xytext=(7, 0),
                         textcoords='offset points',
                         fontsize=9)

        plt.title('Correlação entre Gastos e Saldo Médio por Vereador',
            fontsize=18)
        plt.xlabel('Valor Total Gasto', fontsize=14)
        plt.ylabel('Saldo Médio', fontsize=14)
        plt.xticks(fontsize=12)
        plt.yticks(fontsize=12)
        ax.xaxis.set_major_formatter(formato_real)
        ax.yaxis.set_major_formatter(formato_real)
        plt.tight_layout()
        plt.savefig(f'graficos/correlacao_gastos_saldo_{data_atual}.png',
            dpi=300)
        plt.close()

        #mapa de calor
        plt.figure(figsize=(16, 12))

        top20_vereadores = df.groupby('Vereador')['Valor Total'].sum().
            nlargest(20).index
        df_top20 = df[df['Vereador'].isin(top20_vereadores)]

        heatmap_data = df_top20.pivot_table(index='Vereador', columns='Mê
            s',
                                            values='Valor Total', aggfunc='
                                                sum', fill_value=0)

        heatmap_data = heatmap_data.reindex(df.groupby('Vereador')['Valor
            Total'].sum().nlargest(20).index)

        ax = sns.heatmap(heatmap_data, cmap='viridis', annot=True, fmt='
            .0f',
                         linewidths=.5, cbar_kws={'label': 'Valor Total
                             Gasto (R$)'})

        plt.title('Mapa de Calor: Gastos por Vereador e Mês', fontsize
            =18)
        plt.xlabel('Mês', fontsize=14)
        plt.ylabel('Vereador', fontsize=14)
        plt.tight_layout()
```

```
        plt.savefig(f'graficos/mapa_calor_{data_atual}.png', dpi=300)
        plt.close()

        print(f"\nGráficos salvos na pasta 'graficos' com o timestamp {data_atual}")

    #analise detalhada dos dados
    def analise_detalhada(df):
        if df is None or df.empty:
            print("Sem dados para análise detalhada.")
            return

        print("\n=== ANÁLISE DETALHADA ===")

        media_por_vereador = df.groupby('Vereador')['Valor Total'].mean().sort_values(ascending=False)
        print("\nMédia de gastos por vereador (top 5):")
        for vereador, valor in list(media_por_vereador.items())[:5]:
            print(f"{vereador}: R$ {valor:,.2f}")

        variacao_gastos = df.groupby('Vereador')['Valor Total'].agg(['sum', 'mean', 'std']).sort_values(by='sum', ascending=False)
        variacao_gastos['std'] = variacao_gastos['std'].fillna(0)

        print("\nVariação de gastos (top 5 por total de gastos):")
        for vereador, row in list(variacao_gastos.iterrows())[:5]:
            print(f"{vereador}:")
            print(f" Total: R$ {row['sum']:,.2f}")
            print(f" Média: R$ {row['mean']:,.2f}")
            print(f" Desvio Padrão: R$ {row['std']:,.2f}")

        print("\nTipo de gasto mais comum por vereador (top 10 vereadores):")
        top10_vereadores = list(df.groupby('Vereador')['Valor Total'].sum().sort_values(ascending=False).head(10).index)

        for vereador in top10_vereadores:
            df_vereador = df[df['Vereador'] == vereador]
            gasto_principal = df_vereador.groupby('Especificação')['Valor Total'].sum().sort_values(ascending=False).head(1)
            if not gasto_principal.empty:
                categoria = gasto_principal.index[0]
                valor = gasto_principal.values[0]
                print(f"{vereador}: {categoria} (R$ {valor:,.2f})")

        if len(df['Mês'].unique()) > 1:
            print("\nComparação de gastos entre os meses:")
            for vereador in top10_vereadores:
                df_vereador = df[df['Vereador'] == vereador]
                if len(df_vereador['Mês'].unique()) > 1:
                    gastos_mensais = df_vereador.groupby('Mês')['Valor Total'].sum()
                    mes_min = gastos_mensais.idxmin()
                    mes_max = gastos_mensais.idxmax()

                    print(f"{vereador}:")
                    print(f" Mês com menor gasto: {int(mes_min)} (R$ {
```

```
                    gastos_mensais[mes_min]:,.2f})")
                print(f" Mês com maior gasto: {int(mes_max)} (R$ {
                    gastos_mensais[mes_max]:,.2f})")
                print(f" Variação: {((gastos_mensais[mes_max] /
                    gastos_mensais[mes_min]) - 1) * 100:.2f}%")

#funcao principal
def main():
    print("=== ANÁLISE DE GASTOS DOS VEREADORES DE FORTALEZA ===")

    df = carregar_dados()

    if df is not None:
        print(f"\nDados carregados com sucesso: {len(df)} registros")

        print("\nInformações do DataFrame:")
        print(f"Colunas: {', '.join(df.columns)}")
        print(f"Período: Meses {df['Mês'].min()} a {df['Mês'].max()}
            de {df['Ano'].iloc[0]}")
        print(f"Total de vereadores: {df['Vereador'].nunique()}")
        print(f"Total de categorias de gastos: {df['Especificação'].
            nunique()}")
        print(f"Total de credores: {df['Credor'].nunique()}")
        print(f"Valor total dos gastos: R$ {df['Valor Total'].sum()
            :,.2f}")

        gerar_estatisticas_basicas(df)

        analise_detalhada(df)

        gerar_graficos(df)

        print("\nAnálise concluída!")

if __name__ == "__main__":
    main()
```

Fonte: Elaborado pelo autor.

8 ANÁLISE DOS RESULTADOS

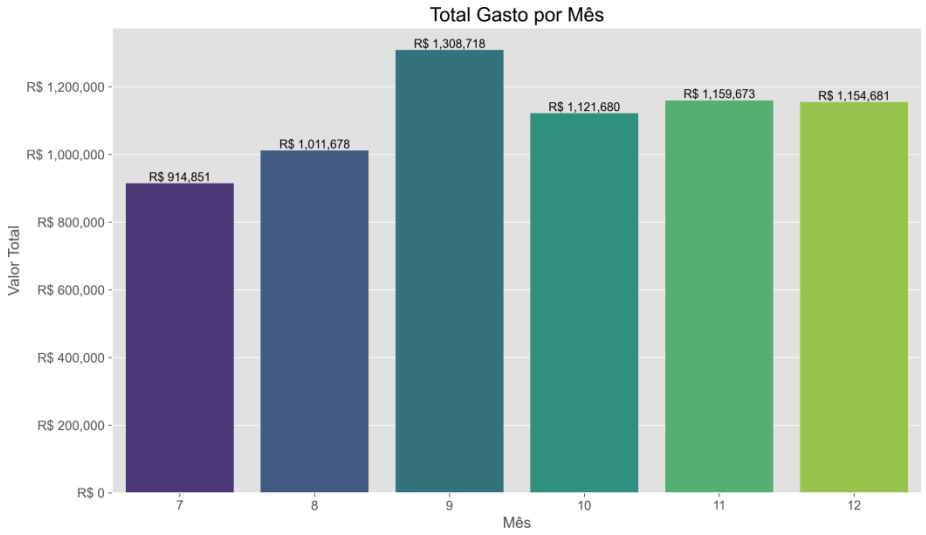
A análise dos dados coletados revelou informações significativas sobre os padrões de gastos do Serviço de Desempenho Parlamentar (SDP) da Câmara Municipal de Fortaleza. A extração e processamento das informações por meio do sistema desenvolvido permitiram identificar tendências, distribuições de recursos e comportamentos consistentes ao longo do período analisado. A seguir, são apresentados os principais resultados obtidos, organizados em categorias analíticas específicas.

8.1 Análise Temporal dos Gastos

A análise temporal dos gastos do SDP revelou um padrão de crescimento no segundo semestre de 2024, com variações significativas entre os meses, conforme ilustrado na Figura 1. O valor total das despesas no período foi de aproximadamente R$ 6,67 milhões, distribuídos de forma não uniforme ao longo dos seis meses analisados.

Os dados demonstram um acentuado aumento nos gastos no mês de setembro (R$ 1.308.717,55), representando um crescimento de 29,3% em relação ao mês de agosto (R$ 1.011.678,04).

Figura 1: Evolução dos gastos mensais do Serviço de Desempenho Parlamentar

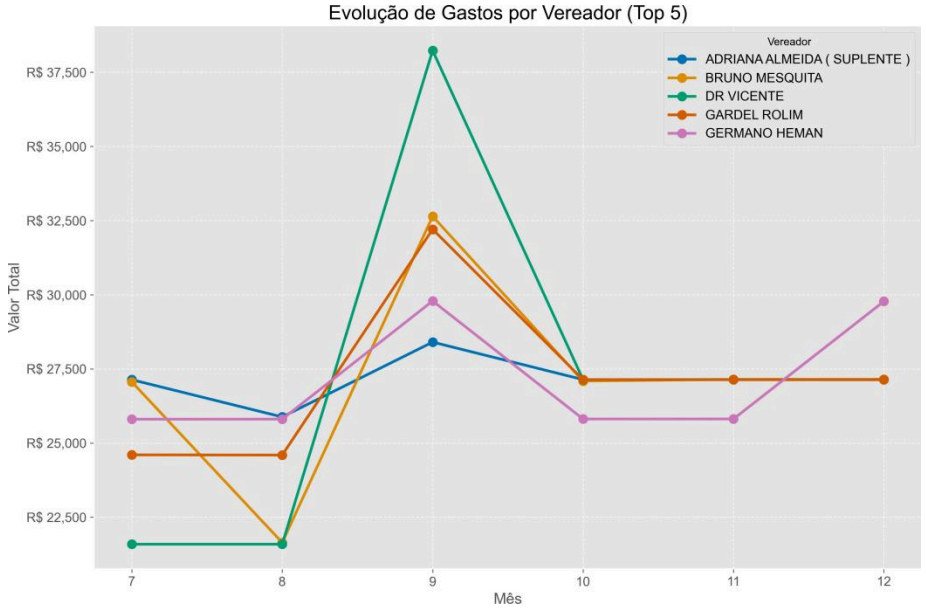


Fonte: Elaborado pelo autor

Este pico de gastos pode estar relacionado ao período pré-eleitoral, considerando-se que setembro antecede o primeiro turno das eleições municipais no Brasil. Nos meses subsequentes, observou-se uma estabilização relativa dos gastos em aproximadamente R$ 1,15 milhão mensal.

A evolução temporal dos gastos por parlamentar, ilustrada na Figura 2, demonstra que, embora exista uma variação entre diferentes mandatos, há um padrão relativamente constante de utilização dos recursos ao longo do período, com pequenas oscilações individuais que não alteram significativamente o perfil geral de despesas da Câmara.

Figura 2: Evolução temporal dos gastos por parlamentar (top 5 maiores gastos)



Fonte: Elaborado pelo autor

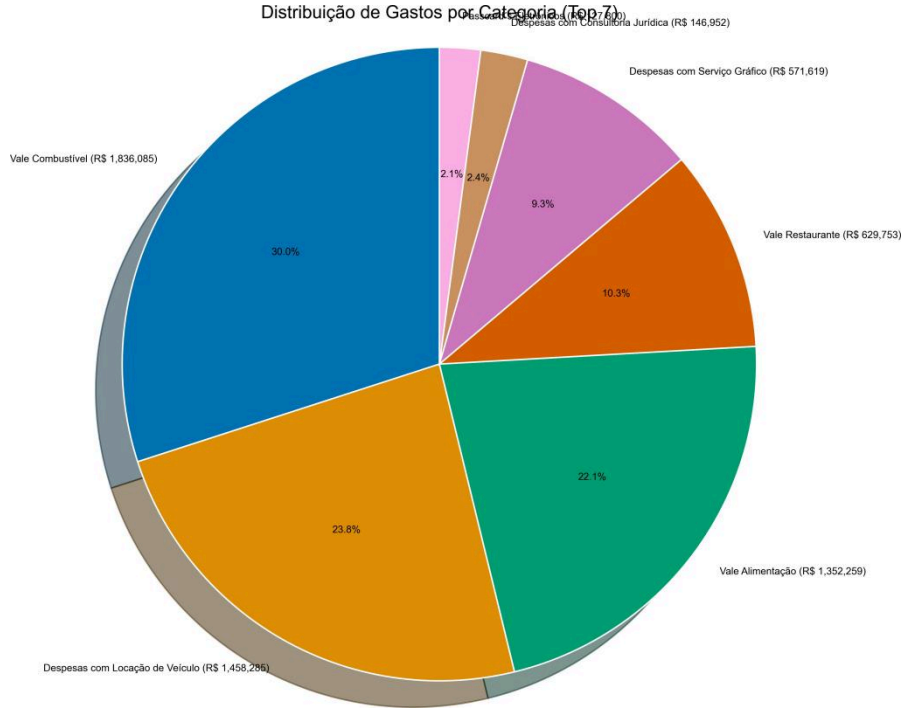
8.2 Distribuição por Categorias de Despesas

A análise da distribuição de gastos por categoria revelou concentrações significativas em áreas específicas, conforme ilustrado na Figura 3. As cinco principais categorias de despesas representam aproximadamente 83,7% do total de gastos do SDP no período analisado, evidenciando uma clara priorização na alocação dos recursos.

Os dados mostram que as despesas com "Vale Combustível"constituem a maior parcela dos gastos, totalizando R$ 1.836.085,30 (27,6% do total), seguidas por "Despesas com Locação de Veículo"com R$ 1.458.284,74 (21,9%) e "Vale Alimentação"com R$ 1.352.259,02 (20,3%). Considerando-se conjuntamente as categorias relacionadas a transporte (combustível e locação de veículos), observa-se que estas representam 49,5% dos gastos totais, evidenciando a predominância dos custos de mobilidade na composição das despesas parlamentares.

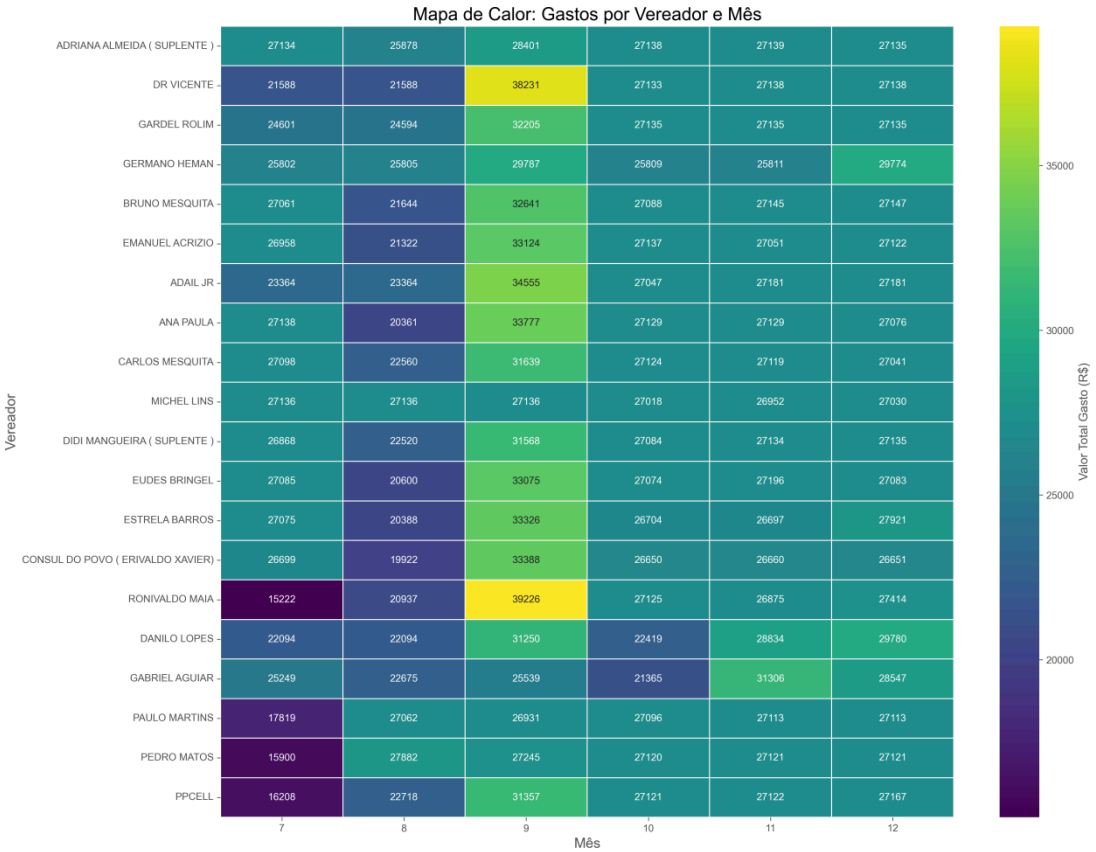
A análise dos gastos por categoria por parlamentar, apresentada na Figura 4, permite identificar padrões individuais de utilização dos recursos, demonstrando preferências distintas entre os mandatos na alocação das verbas do SDP.

Figura 3: Distribuição dos gastos por categoria



Fonte: Elaborado pelo autor

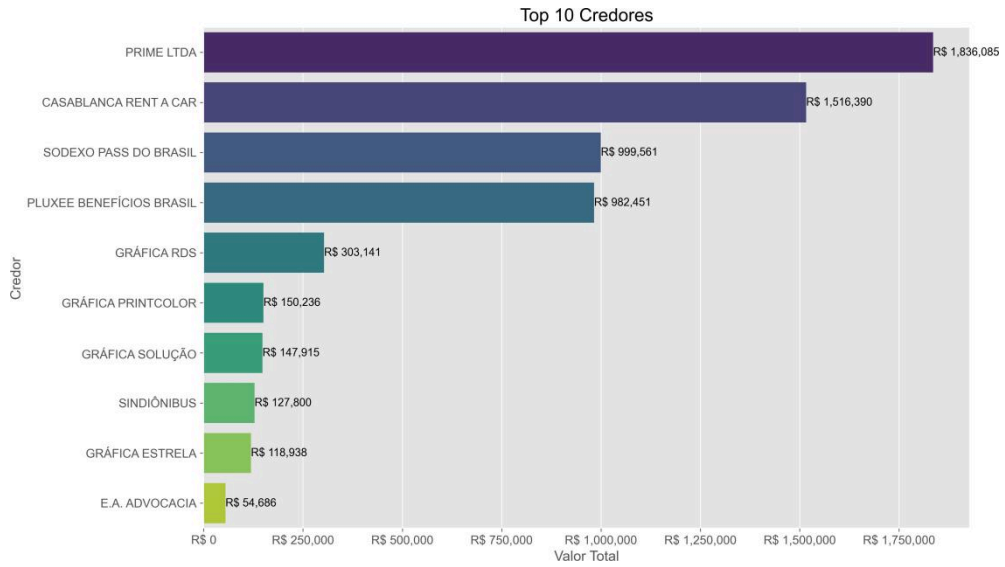
Figura 4: Mapa de calor: Gastos por parlamentar e mês



Fonte: Elaborado pelo autor.

8.3 Análise de Credores e Fornecedores

Figura 5: Distribuição dos gastos por principais credores



Fonte: Elaborado pelo autor.

A análise dos credores e fornecedores que recebem recursos do SDP revela uma alta concentração em poucos prestadores de serviços. Conforme ilustrado na Figura 5, os cinco principais credores receberam aproximadamente 82,5% do total de recursos destinados ao SDP no período analisado.

A empresa "PRIME LTDA", fornecedora de vales combustível, lidera o ranking de cre- dores com R$ 1.836.085,30 (27,6% do total), seguida pela locadora "CASABLANCA RENT A CAR"com R$ 1.516.389,74 (22,8%) e pelas empresas de benefícios alimentação "SODEXO PASS DO BRASIL"e "PLUXEE BENEFÍCIOS BRASIL", que juntas receberam R$ 1.982.012,10

(29,8%). Esta concentração levanta questões importantes sobre a diversificação de fornecedo- res e potenciais oportunidades para otimização de custos através de processos licitatórios mais amplos.

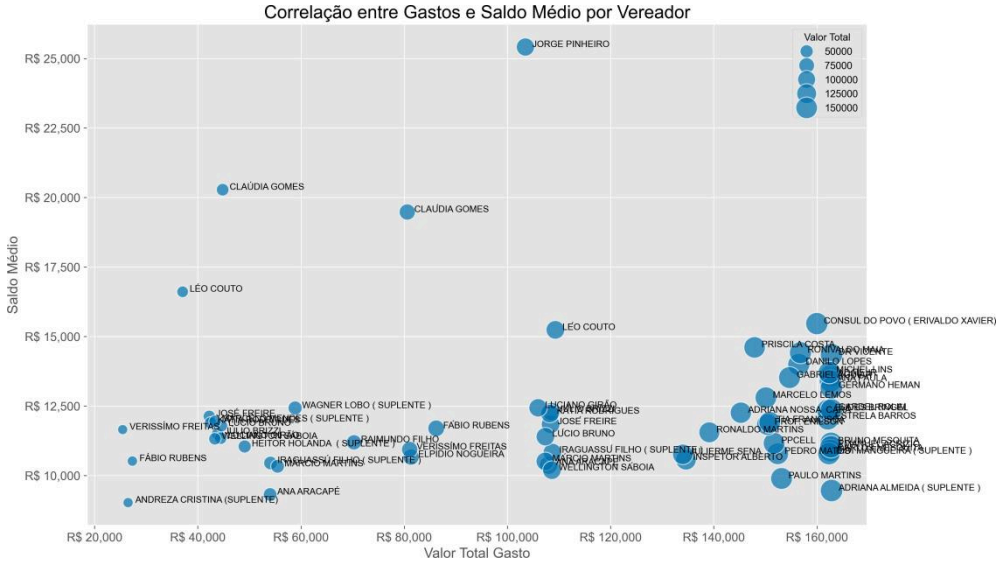
8.4 Correlações e Padrões de Utilização

A análise de correlações entre os gastos totais e os saldos médios, ilustrada na Figura 6, permite identificar diferentes perfis de utilização dos recursos do SDP entre os parlamentares.

Os dados revelam uma variação significativa nos padrões de utilização dos recursos, com alguns parlamentares mantendo saldos médios mais elevados enquanto outros utilizam a quase totalidade dos recursos disponíveis. A análise estatística demonstrou que, embora os valores totais de gastos sejam relativamente uniformes entre os parlamentares (aproximadamente R$ 162 mil no semestre para os dez maiores gastos), há diferenças consideráveis na distribuição temporal desses recursos, com desvios-padrão variando de R$ 2.349,14 a R$ 3.716,72 entre os cinco principais gastos.

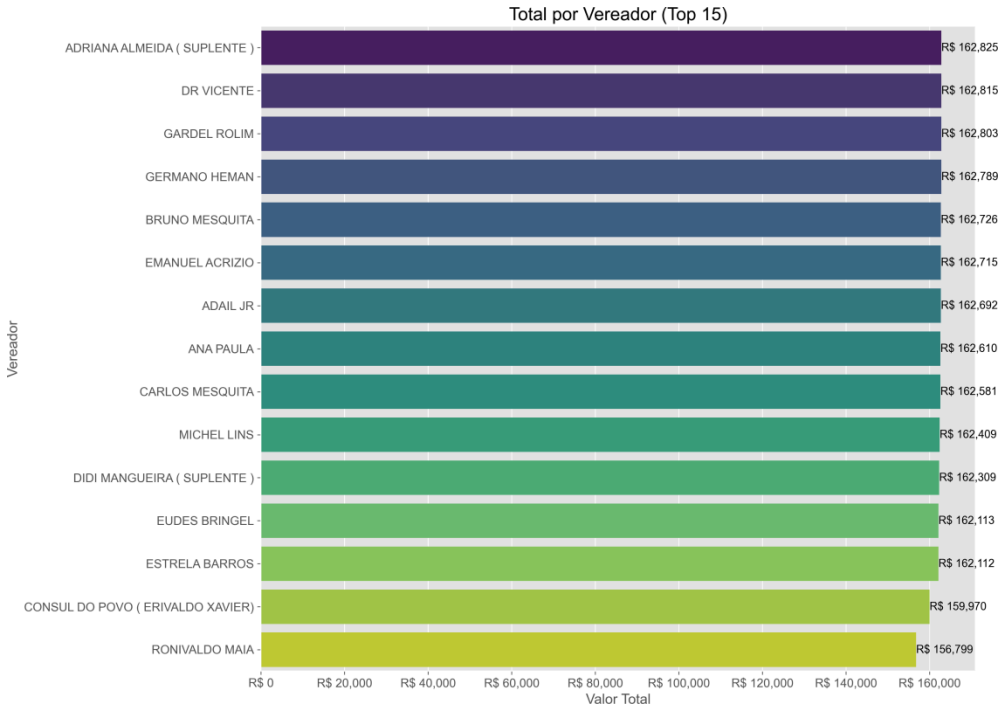
O ranking total de gastos por parlamentar, apresentado na Figura 7, demonstra uma distri- buição relativamente equilibrada dos recursos entre os diferentes mandatos, sugerindo que os limites estabelecidos para o SDP são utilizados de forma praticamente integral pela maioria dos parlamentares.

Figura 6: Correlação entre gastos totais e saldo médio por parlamentar



Fonte: Elaborado pelo autor.

Figura 7: Total de gastos por parlamentar (top 15)



Fonte: Elaborado pelo autor.

8.5 Considerações sobre os Resultados

A análise dos dados do SDP da Câmara Municipal de Fortaleza revela padrões consistentes de utilização dos recursos públicos, com concentração em categorias específicas e fornecedores determinados. Os resultados obtidos corroboram as observações de Krotov *et al.,* (2020) sobre a importância da transparência nos gastos governamentais e alinham-se às recomendações de Barbosa e Cavalcanti (2020) quanto à necessidade de análises sistemáticas para compreensão das dinâmicas financeiras na administração pública.

A predominância de gastos relacionados a transporte e alimentação sugere uma priorização destas áreas na atuação parlamentar, possivelmente refletindo necessidades logísticas e operacionais dos gabinetes. Entretanto, a alta concentração de recursos em poucos fornecedores indica potenciais oportunidades para otimização de custos através de processos licitatórios mais amplos.

Essa concentração de recursos em poucos fornecedores, especialmente nas áreas de combustível, locação de veículos e alimentação, pode suscitar questionamentos sobre a competitividade dos processos de contratação e a busca por melhores condições de preço e serviço. Embora a análise não permita afirmar irregularidades, a falta de diversificação pode, em tese, limitar a eficiência na alocação dos recursos públicos e reduzir o potencial de economia para a administração. Investigar os modelos de contratação e a variedade de fornecedores em futuras análises poderia trazer insights adicionais sobre a otimização desses gastos.

As análises temporais e comparativas implementadas pelo sistema de web scraping permitiram identificar padrões sazonais e variações individuais que seriam dificilmente detectáveis por métodos manuais de coleta e análise de dados, demonstrando o valor agregado pela automação destes processos para a promoção da transparência pública, conforme destacado por Calò (2014).

9 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a implementação de um sistema de web scraping para a extração e análise de dados do Serviço de Desempenho Parlamentar (SDP) da Câmara Municipal de Fortaleza, demonstrando a eficácia desta técnica para promover a transparência na gestão pública. O objetivo geral foi alcançado, evidenciado pelo desenvolvimento bem-sucedido de um sistema automatizado que utilizou Python e suas bibliotecas Selenium, Beautiful Soup, Pandas, NumPy e Matplotlib. Este sistema permitiu a coleta estruturada, o processamento eficiente e a visualização clara dos dados dos gastos parlamentares, criando uma base de dados organizada que possibilitou análises detalhadas dos padrões de despesas.

A implementação do sistema resultou em um processo estruturado de coleta e análise dos dados, promovendo a transparência pública através da automatização da coleta de informações. Além disso, desenvolveu-se uma metodologia replicável que pode ser aplicada em outros contextos de monitoramento de gastos públicos, contribuindo para uma gestão mais transparente e eficiente dos recursos públicos.

Entretanto, é importante reconhecer as limitações deste trabalho. A dependência da qualidade dos dados disponíveis no portal de transparência pode afetar a precisão das análises. Além disso, o sistema desenvolvido está restrito ao contexto da Câmara Municipal de Fortaleza, o que pode limitar sua aplicabilidade em outras localidades ou esferas governamentais. A variação na estrutura das páginas web também pode exigir ajustes frequentes nos scripts de scraping, o que pode ser um desafio contínuo.

Para futuros trabalhos, sugere-se a ampliação da pesquisa para incluir outros municípios ou esferas do governo, permitindo uma comparação mais abrangente sobre os padrões de gastos públicos. Além disso, a exploração de técnicas avançadas de análise de dados, como aprendizado

de máquina, poderia enriquecer as interpretações dos dados coletados. A criação de interfaces amigáveis para visualização dos dados também poderia facilitar o acesso e a compreensão das informações por parte do público em geral.

Em suma, embora o portal de transparência já disponibilize os dados publicamente, este trabalho agregou valor ao demonstrar como o web scraping pode transformar dados brutos em informações mais acessíveis e analisáveis. Ao automatizar a coleta, estruturar os dados e gerar visualizações claras, o sistema desenvolvido facilita a identificação de padrões, tendências e concentrações de gastos que dificilmente seriam percebidas manualmente, contribuindo efetivamente para uma maior compreensão e fiscalização do uso dos recursos públicos. Assim, estabeleceu-se um caminho para futuras investigações e melhorias na gestão pública através da análise sistemática dos gastos parlamentares.

REFERÊNCIAS

BARBOSA, A. B. G.; CAVALCANTI, A. B. Web scraping e análise de dados. In: Anais do CONAPESC - Congresso Nacional de Pesquisa e Ensino em Ciências, Campina Grande. Realize Editora. Disponível em: https://www.editorarealize.com.br/editora/anais/conapesc/2020/TRABALHO_EV138_MD4_SA24_ID1284_24112020001516.pdf. Acesso em: 15 out. 2024.

BEAUTIFUL SOUP. Beautiful soup: Documentação em português. Disponível em: https://www.crummy.com/software/BeautifulSoup/bs4/doc.ptbr/. Acesso em: 15 abr. 2025.

CALÒ, A. Extração e análise de informações jurídicas públicas. Monografia (Bacharelado em Ciência da Computação) - Instituto de Matemática e Estatística, Universidade de São Paulo. Disponível em: https://bccdev.ime.usp.br/tccs/2014/sandro/Monografia.pdf. Acesso em: 18 out. 2024.

CÂMARA MUNICIPAL DE FORTALEZA. Portal da transparência – despesas sdp. Disponível em: https://portaltransparencia.cmfor.ce.gov.br/despesas/sdp. Acesso em: 10 mai. 2025.

CÂMARA MUNICIPAL DE FORTALEZA. Sistema de apoio ao processo legislativo – sapl. Disponível em: https://sapl.fortaleza.ce.leg.br. Acesso em: 15 abr. 2025.

KHDER, M. A. Web scraping or web crawling: State of art, techniques, approaches and application. International Journal of Advance Soft Computing and Applications, v. 13, n. 3, p. 144-162. Disponível em: http://www.i-csrs.org/Volumes/ijasca/2021.3.11.pdf. Acesso em: 20 out. 2024.

KROTOV, V.; JOHNSON, L.; SILVA, L. Tutorial: Legality and ethics of web scraping. Communications of the Association for Information Systems, v. 47, p. 1-37. Disponível em: https://digitalcommons.murraystate.edu/faculty/86/. Acesso em: 22 out. 2024.

MATPLOTLIB. Matplotlib: Visualization with python. Disponível em: https://matplotlib.org/stable/index.html. Acesso em: 15 abr. 2025.

MITCHELL, R. Web Scraping com Python: Coletando Mais Dados da Web Moderna. 2. ed. O'Reilly Media.

NUMPY. Numpy: The fundamental package for scientific computing with python. Disponível em: https://numpy.org/doc/stable/. Acesso em: 15 abr. 2025.

SILVA, G. V. R. Sdp scraping: Sistema de extração e análise de dados do serviço de desempenho parlamentar de fortaleza. Disponível em: https://github.com/guivtl/sdp_scraping. Acesso em: 10

jan. 2025.

TRIBUNAL DE CONTAS DO ESTADO DO CEARÁ. Relatório de fiscalização orçamentária 2023. Technical report, TCE-CE, Fortaleza. Disponível em: https://www.tce.ce.gov.br. Acesso em: 28 abr. 2025.