

USO DA LINGUAGEM PYTHON NA AUTOMATIZAÇÃO DE MENSAGENS VIA WHATSAPP E E-MAIL

Laura Fernandes Procópio

Departamento de Informática - Instituto Federal de Educação, Ciência e Tecnologia Mato Grosso – Campus Cuiabá Cel. Octayde Jorge da Silva, Rua Profa. Zulmira Canavarros, 95 – CEP: 78005-200

Laura.p@estudante.ifmt.edu.br

RESUMO

A automatização de tarefas é crucial para melhorar a produtividade e minimizar o tempo dedicado a atividades repetitivas. Python é uma linguagem de programação amplamente reconhecida por sua simplicidade e flexibilidade, tornando-se uma ferramenta eficaz para a criação de scripts de automação e integração com diversas APIs, como as do WhatsApp e Gmail. Este artigo investiga como Python pode ser empregado para automatizar o envio de mensagens via WhatsApp e e-mail, buscando aprimorar a eficiência e reduzir erros associados à comunicação manual. O estudo é estruturado em: revisão das ferramentas e bibliotecas disponíveis para integração com APIs, desenvolvimento de scripts de automação e uma implementação prática que demonstra a eficiência do sistema para envio em massa de mensagens. Espero que este trabalho contribua com os desenvolvedores no intuito de mostrar uma linguagem utilizada para automatizar o envio de mensagens via WhatsApp e E-mail.

Palavras-chave: Automatização de Tarefas; Python; E-mail; WhatsApp; Scripts de Automatização.

1 INTRODUÇÃO

A automatização de tarefas é fundamental para otimizar a produtividade e reduzir o tempo gasto em atividades repetitivas, como destacado por Sweigart (2015): “A automatização reduz a necessidade de intervenção manual e minimiza erros” (p. 12). Python, com sua simplicidade e flexibilidade, se destaca como uma ferramenta poderosa para criar scripts de automação e integrar com diversas APIs, como WhatsApp e Gmail (Matthes, 2019). A automatização reduz erros e melhora a eficiência dos processos de comunicação, o que é crucial em um ambiente onde a pontualidade e a precisão são essenciais (McKinney, 2017). O envio manual de mensagens, por exemplo, é demorado e sujeito a falhas, o que pode comprometer a qualidade e a agilidade da comunicação.

Este artigo aborda a automatização de tarefas com Python e tem como objetivo demonstrar como essa linguagem pode ser utilizada para automatizar o envio de mensagens via WhatsApp e e-mail. A necessidade de automatizar esses processos surge da demanda por uma comunicação mais eficiente e precisa, que garante não apenas a entrega oportuna das mensagens, mas também a redução de erros humanos.

Os objetivos específicos desse trabalho são:

1. Fazer a revisão de Ferramentas e Bibliotecas Disponíveis para integração com APIs de WhatsApp e serviços de e-mail.
2. Desenvolver um Script de Automação em Python: para criar um script que envia mensagens automaticamente.
3. Exemplo Prático: Desenvolver um exemplo prático de envio em lote de mensagens, destacando a eficiência e escalabilidade do sistema de automação.

2 REFERENCIAL TEÓRICO

A automatização de tarefas com Python é valorizada pela sua capacidade de aumentar eficiência e produtividade. Python é conhecido pela simplicidade e extensa biblioteca padrão, facilitando a automação de tarefas rotineiras e integração com APIs e serviços web (Sweigart, 2015; Matthes, 2019). Bibliotecas como `smtplib` e `requests` são essenciais para e-mails e HTTP, respectivamente, enquanto `pandas` e `numpy` ajudam na manipulação de dados e `matplotlib` na visualização (McKinney, 2017).

Python é ideal para tarefas rotineiras pela sua simplicidade e vasto suporte de bibliotecas. A documentação oficial e a comunidade oferecem ferramentas para automação, com destaque para `smtplib` e `requests` (Matthes, 2019).

2.1 ENVIO DE AUTOMATIZADO DE E-MAIL

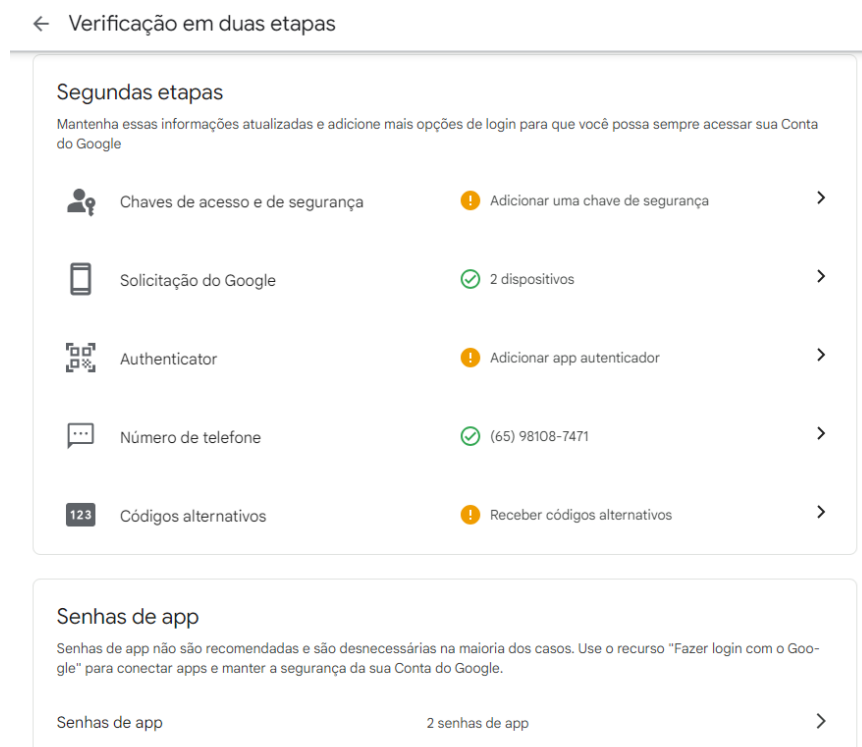
O envio automatizado de e-mails é uma aplicação comum de Python. A biblioteca `smtplib` permite a implementação de clientes SMTP para envio de e-mails programaticamente (Python, n.d.). A integração com a biblioteca `email` permite criar e-mails sofisticados, incluindo conteúdo HTML e anexos (Real Python, n.d.). No entanto, para utilizar o Gmail como servidor SMTP, é necessário configurar a autenticação de dois fatores e gerar uma senha de aplicativo específica

no Google. Isso é fundamental para garantir a segurança das credenciais e permitir que scripts automatizados acessem a conta de e-mail (Google, n.d.).



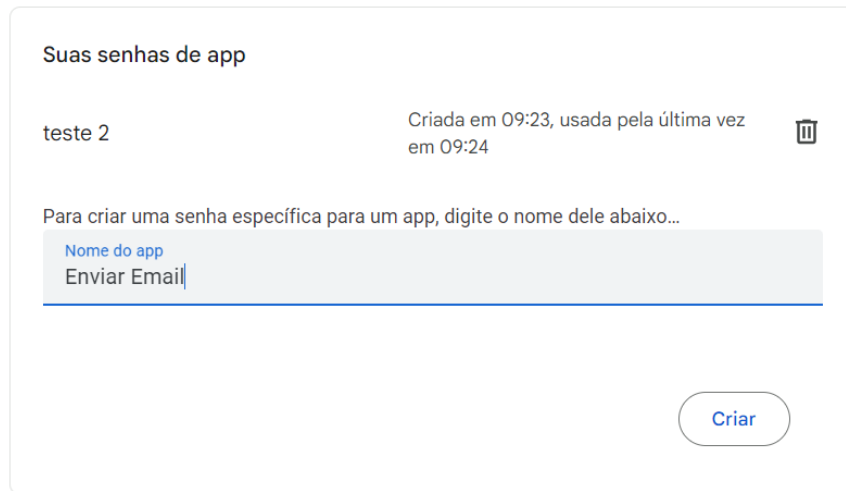
Fonte: Figura do Autor(a)

Figura 2 - Tela de Senhas de Aplicativo na Autenticação de Dois Fatores do Google



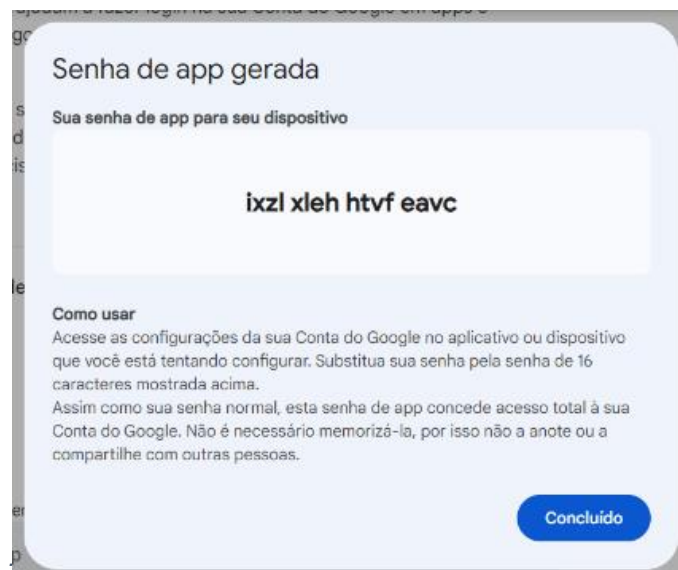
Fonte: Figura do Autor(a)

Figura 3 - Tela com Lista de Senhas de Aplicativo na Autenticação de Dois Fatores do Google



Fonte: Figura do Autor(a)

Figura 4 - Senha de Aplicativo Gerada



Fonte: Figura do Autor(a)

Além disso, existem APIs especializadas para envio de e-mails:

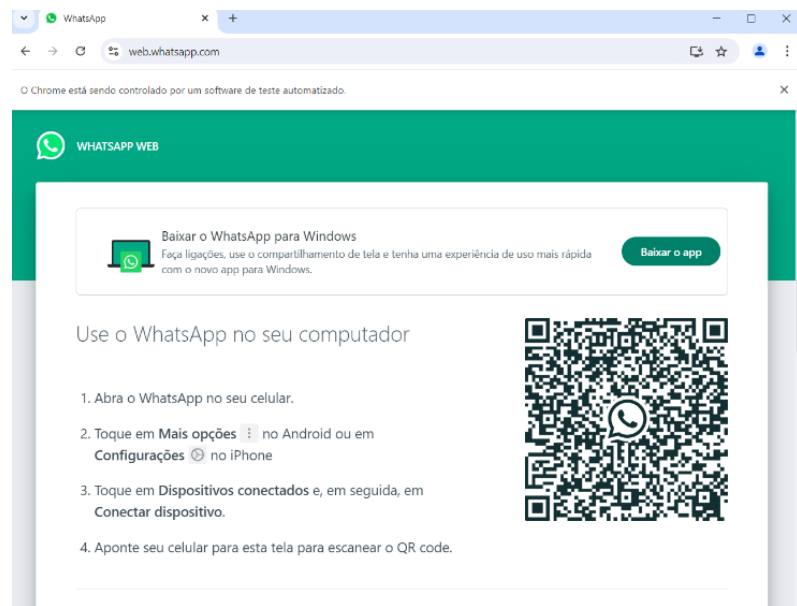
API SendGrid: Um serviço de e-mail transacional pago, conhecido por sua escalabilidade e facilidade de integração com várias plataformas.

API Mailgun: Outra API paga popular para envio de e-mails, conhecida por sua flexibilidade e recursos avançados.

2.2 ENVIO DE AUTOMATIZADO DE MENSAGENS VIA WHATSAPP

A API do Twilio permite o envio automatizado de mensagens pelo WhatsApp, embora tenha algumas restrições, como a necessidade de um número aprovado e limitações de taxa. Como alternativa, a biblioteca Selenium pode ser utilizada para automatizar a interface do WhatsApp Web.

Figura 5 - Interface do WhatsApp Web



Fonte: Figura do Autor(a)

2.3 TESTES E VALIDAÇÃO

Para a implementação prática das técnicas de automação discutidas, várias ferramentas e bibliotecas Python podem ser utilizadas. A tabela abaixo resume algumas das principais bibliotecas e suas funcionalidades:

BIBLIOTECA	FUNCIONALIDADE
'SMTPLIB'	Envio de e-mails via protocolo SMTP
'TWILIO'	Envio de Mensagens via WhatsApp
'PANDAS'	Manipulação e análise de dados
'SENDGRID'	Envio de e-mails com recursos avançados
'MAILGUN'	Envio de e-mails com suporte a automação de campanhas

Tabela 1 Biblioteca Python

3 MATERIAL E MÉTODOS

Este artigo realiza uma pesquisa aplicada com foco na implementação prática de técnicas de automatização utilizando Python. A pesquisa é de natureza exploratória e descritiva, visando demonstrar a aplicação de ferramentas e métodos de automação na prática. A abordagem adotada permite a avaliação da eficácia e eficiência das técnicas de automatização para o envio de mensagens via WhatsApp e e-mail.

3.1 MATERIAIS UTILIZADOS

Para a execução deste estudo, foram utilizados os seguintes materiais:

Computador: Um dispositivo com sistema operacional Windows foi empregado para o desenvolvimento e testes das soluções propostas.

Python: A versão 3.12 da linguagem Python foi utilizada como base para a implementação dos scripts de automação.

Bibliotecas Python: As bibliotecas `smtplib` e `Gmail` foram empregadas para o envio automatizado de e-mails. Para a automação do WhatsApp Web, foi utilizada a biblioteca `Selenium`.

Editor de Código: O Visual Studio Code foi o ambiente de desenvolvimento integrado (IDE) escolhido para a escrita e depuração do código.

Editor de Planilhas: O Microsoft Excel foi utilizado para a manipulação e organização dos dados em dois formatos: CSV (Comma-Separated Values) para o envio de e-mails e XLSX (Excel Workbook) para o envio via WhatsApp.

Conexão à Internet: Uma conexão estável à internet foi necessária para o funcionamento das automações e testes de envio de mensagens.

3.2 MÉTODOS

A seguir, descrevemos os métodos empregados para a implementação da automatização de envio de mensagens.

3.1.1 Envio Automatizado de E-mails

Configuração do Servidor SMTP: Configuração do servidor SMTP para envio de e-mails e Autenticação do usuário no servidor SMTP.

Figura 6 - Código de Configuração SMTP em Python

```
49 s = smtplib.SMTP('smtp.gmail.com: 587')
50 s.starttls()
```

Fonte: Figura do Autor(a)

Criação da Mensagem de E-mail: Utilização da biblioteca email para criar e formatar a mensagem e Inclusão de texto simples ou HTML no corpo do e-mail.

Figura 7 - Código de Importação da Biblioteca email.message em Python

```
4 import email.message
```

Fonte: Figura do Autor(a)

Figura 8 - Visão Geral do Código Usando a Biblioteca email.message com Corpo do E-mail em HTML

```
7 email_enviados = []
8 email_enviados_antes = []
9 email_assunto = 'AUTOMATIZAÇÃO EFICIENTE DE TAREFAS COM PYTHON: AUTOMATIZADO DE MENSAGENS VIA WHAS-TSAPP E E-MAIL'
10 email_corpo = '''
11 A automatização de tarefas é crucial para melhorar a produtividade e reduzir o tem-po gasto com atividades repetitivas.
12 Python, com sua flexibilidade e simplicidade, oferece uma plataforma robusta para criar scripts de automação e integrar
13 com di-versas APIs, como WhatsApp e Gmail.
14
15 Python disponibiliza uma vasta gama de bibliotecas e pacotes, como smtplib para envio de e-mails e requests para
16 interações com APIs. Essas ferramentas permitem criar soluções personalizadas, aumentar a eficiência e minimizar erros.
17
18 A comunicação eficaz é vital em qualquer organização. O envio manual de mensa-gens é demorado e sujeito a erros,
19 o que pode comprometer a qualidade da comu-nicação e a eficiência dos processos. Automatizar o envio de mensagens pode
20 ga-rantir uma comunicação mais precisa e oportuna.
21
22 Este artigo explora como Python pode ser utilizado para automatizar o envio de mensagens via WhatsApp e e-mail.
23 A estrutura do artigo é a seguinte:
24
25 1. Ferramentas e Bibliotecas Disponíveis: Revisão das principais ferramentas e bibliotecas para integração com APIs
26 de WhatsApp e serviços de e-mail.
27 2. Desenvolvimento do Script de Automação: Guia passo a passo para de-senvolver um script Python que envia mensagens
28 automaticamente.
29 3. Exemplo Prático: Demonstração de uma implementação real para envio em massa de mensagens, destacando a eficiência
30 e a escalabilidade do sistema de automação.
31
32 ...'''
```

Fonte: Figura do Autor(a)

Envio do E-mail: Utilização da função sendmail da biblioteca smtplib para enviar a mensagem ao destinatário.

Importação do CSV: Para ler arquivos no formato CSV (Comma-Separated Values), que são arquivos separados por vírgula, foi utilizada a função de importação adequada, facilitando a extração e manipulação dos dados necessários para o envio de e-mails.

Figura 9 - Código de Importação da Biblioteca CSV em Python

```
1 # Ler arquivo csv
2 import csv
```

Fonte: Figura do Autor(a)

Figura 10 - Visão Geral do Código Usando a Biblioteca csv Demonstrando a Leitura de Arquivo

```
72 # Le arquivo de dados que contem o email da pessoa
73 with open('enviaremail.csv','r',encoding='utf8',newline='\r\n') as arquivo:
74     linha = csv.reader(arquivo)
75     lPessoas = [ x for x in linha]
76
```

Fonte: Figura do Autor(a)

Figura 11 - Arquivo em CSV

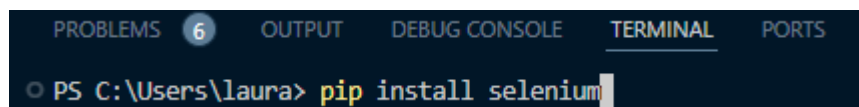
	A	B
1	João Silva,joao.silva@example.com	
2	Ana,ana.souza@corporate.org	
3	Fernanda,fernanda.lima@mailservice.com	

Fonte: Figura do Autor(a)

Envio Automatizado de Mensagens via WhatsApp

Automação com Selenium: Utilização da biblioteca Selenium para a automação do envio de mensagens no WhatsApp Web.

Figura 12 - Instalação da biblioteca Selenium



```
PS C:\Users\laura> pip install selenium
```

Fonte: Figura do Autor(a)

Interface Utilizada: O desenvolvimento e a execução dos scripts foram realizados utilizando o Visual Studio Code (VS Code), um editor de código-fonte amplamente utilizado para desenvolvimento Python devido à sua interface intuitiva e extensões úteis.

Importação da Biblioteca Pandas: A biblioteca Pandas foi importada para a manipulação de arquivos no formato XLSX. Esta biblioteca é essencial para o tratamento de dados tabulares, permitindo operações como leitura, escrita, e transformação de dados com facilidade e eficiência.

Figura 13 - Código de Importação da Biblioteca Pandas em Python

```
1 import pandas as pd
```

Fonte: Figura do Autor(a)

Figura 14 - Visão Geral do Código Usando a Biblioteca Pandas Demonstrando a Leitura de Arquivo

```
10 # Carregar contatos de um arquivo Excel
11 contatos_df = pd.read_excel("Enviar.xlsx")
12 print(contatos_df)
```

Fonte: Figura do Autor(a)

Figura 15 - Arquivo em XLSX

	A	B	C
1	Pessoa	Número	Mensagem
2	Elton	+5567999999999	Texto Qualquer
3	Katia	+5567999999999	Texto Qualquer

Fonte: Figura do Autor(a)

3.1.2 Testes e Validação

Para assegurar a qualidade do código, validamos a funcionalidade dos scripts em um ambiente de produção limitado para garantir que as mensagens fossem enviadas corretamente. Os códigos a seguir de exemplo prático foram os da biblioteca Selenium e smtplib, mas é possível usar as outras bibliotecas comentadas anteriormente.

Exemplo Prático: Envio de Mensagens via WhatsApp com Selenium

Figura 16 - Código Python Automatização WhatsApp part1

```

1 import pandas as pd
2 from selenium import webdriver
3 from selenium.webdriver.common.by import By
4 from selenium.webdriver.common.keys import Keys
5 from selenium.webdriver.support.ui import WebDriverWait
6 from selenium.webdriver.support import expected_conditions as EC
7 import time
8 import urllib.parse
9
10 # Carregar contatos de um arquivo Excel
11 contatos_df = pd.read_excel("Enviar.xlsx")
12 print(contatos_df)
13
14 try:
15     # Inicializar o navegador com as configurações padrão
16     navegador = webdriver.Chrome()
17     navegador.get("https://web.whatsapp.com/")
18
19     # Esperar até que o WhatsApp Web esteja carregado
20     WebDriverWait(navegador, 60).until(EC.presence_of_element_located((By.ID, "side")))
21
22     # Envio das mensagens
23     for i, row in contatos_df.iterrows():
24         pessoa = row["Pessoa"]
25         numero = row["Número"]
26         mensagem = row["Mensagem"]
27
28         # Preparar o texto da mensagem
29         texto = urllib.parse.quote(mensagem)
30         link = f"https://web.whatsapp.com/send?phone={numero}&text={texto}"
31
32

```

Fonte: Figura do Autor(a)

Figura 17 - Código Python Automatização WhatsApp part1

```

31
32 # Navegar para o link do WhatsApp com o número e mensagem
33 navegador.get(link)
34
35 try:
36     # Esperar até que o botão de enviar esteja visível
37     # A localização do botão pode variar dependendo da versão do WhatsApp Web
38     enviar_button = WebDriverWait(navegador, 60).until(
39         EC.element_to_be_clickable((By.XPATH, '//span[@data-icon="send"]'))
40     )
41
42     # Verificar se o botão de enviar está visível e habilitado
43     if enviar_button.is_displayed() and enviar_button.is_enabled():
44         print(f"Enviando mensagem para {pessoa} ({numero})")
45         enviar_button.click()
46
47     # Esperar um pouco antes de enviar a próxima mensagem
48     time.sleep(10)
49
50     else:
51         print(f"Botão de enviar não está interativo para {pessoa} ({numero})")
52
53 except Exception as e:
54     print(f"Erro ao enviar mensagem para {pessoa} ({numero}): {e}")
55
56 navegador.quit()
57
58 except Exception as e:
59     print(f"Erro ao inicializar o navegador: {e}")
60

```

Fonte: Figura do Autor(a)

Explicação do Código:

1. **Carregar Contatos:** O script começa carregando uma planilha Excel contendo informações de contato e mensagens.
2. **Inicialização do Navegador:** O Selenium abre o navegador e acessa o WhatsApp Web.
3. **Esperar o Carregamento:** O script aguarda o carregamento completo da interface do WhatsApp Web.
4. **Envio de Mensagens:** Para cada contato, o script cria um link de envio de mensagem e o abre no navegador. Após isso, aguarda a visibilidade do botão de envio e realiza o clique para enviar a mensagem.

Exemplo Prático 1: Envio de Mensagens via Email com smtplib

Figura 18 - Código Python Automatização E-mail part1

```

1  # ler arquivo csv
2  import csv
3  import smtplib
4  import email.message
5  from time import sleep
6
7  email_enviados = []
8  email_enviados_antes = []
9  email_assunto = 'AUTOMATIZAÇÃO EFICIENTE DE TAREFAS COM PYTHON: AUTOMATIZADO DE MENSAGENS VIA WHATSAPP E E-MAIL'
10 email_corpo = '''
11 A automatização de tarefas é crucial para melhorar a produtividade e reduzir o tempo gasto com atividades repetitivas.
12 Python, com sua flexibilidade e simplicidade, oferece uma plataforma robusta para criar scripts de automação e integrar
13 com diversas APIs, como WhatsApp e Gmail.
14
15 Python disponibiliza uma vasta gama de bibliotecas e pacotes, como smtplib para envio de e-mails e requests para
16 interações com APIs. Essas ferramentas permitem criar soluções personalizadas, aumentar a eficiência e minimizar erros.
17
18 A comunicação eficaz é vital em qualquer organização. O envio manual de mensagens é demorado e sujeito a erros,
19 o que pode comprometer a qualidade da comunicação e a eficiência dos processos. Automatizar o envio de mensagens pode
20 garantir uma comunicação mais precisa e oportuna.
21
22 Este artigo explora como Python pode ser utilizado para automatizar o envio de mensagens via WhatsApp e e-mail.
23 A estrutura do artigo é a seguinte:
24
25 1. Ferramentas e Bibliotecas Disponíveis: Revisão das principais ferramentas e bibliotecas para integração com APIs
26    de WhatsApp e serviços de e-mail.
27 2. Desenvolvimento do Script de Automação: Guia passo a passo para desenvolver um script Python que envia mensagens
28    automaticamente.
29 3. Exemplo Prático: Demonstração de uma implementação real para envio em massa de mensagens, destacando a eficiência
30    e a escalabilidade do sistema de automação.
31
32 ...
33
34 def enviar_email(to) -> None:
35     # caso o email ja tenha sido enviado, retorna
36     if to in email_enviados+email_enviados_antes:
37         print(f'Email enviado anteriormente - {to}')
38         return
39
40     msg = email.message.Message()
41     msg['Subject'] = email_assunto
42     msg['From'] = 'laura.fg@colaborador.ifmt.edu.br' # seu email
43     msg['To'] = to
44
45     password = 'lpfs mecj qecj doqj' # Senha gerada pelo google
46     # msg.add_header('Content-Type', 'text/html')
47     msg.set_payload(email_corpo)
48

```

Fonte: Figura do Autor(a)

Figura 19 - Código Python Automatização E-mail part1

```

48
49 s = smtplib.SMTP('smtp.gmail.com: 587')
50 s.starttls()
51 # Login Credentials for sending the mail
52 s.login(msg['From'], password)
53 s.sendmail(msg['From'], [msg['To']], msg.as_string().encode('utf-8'))
54
55 email_enviados.append(to)
56 print(f'Email enviado - {to}')
57 # salva o email no arquivo de email enviado
58 with open('emailenviado.txt', 'a') as arquivo:
59     arquivo.writelines(f'{to}\n')
60
61 # abre arquivo de emails enviados
62 try:
63     with open('emailenviado.txt', 'r') as arquivo:
64         linha = arquivo.readlines()
65         email_enviados_antes = [email.replace('\n', '') for email in linha]
66 except Exception as error:
67     # cria o arquivo de email enviado, caso nao tenha
68     with open('emailenviado.txt', 'w') as arquivo:
69         ...
70
71 # Le arquivo de dados que contem o email da pessoa
72 with open('enviaremail.csv', 'r', encoding='utf8', newline='\r\n') as arquivo:
73     linha = csv.reader(arquivo)
74     lPessoas = [x for x in linha]
75
76 # percorre a lista de emails
77 for pessoa in lPessoas:
78     # chama a rotina de envio de email passando o email
79     enviar_email(pessoa[1])
80     sleep(2)

```

Fonte: Figura do Autor(a)

Explicação do Código:

1. Importação de Bibliotecas e Definição de Variáveis:

- O código importa bibliotecas necessárias para o envio de e-mails e manipulação de arquivos (csv, smtplib, email.message, time.sleep).

- Define listas para armazenar e-mails já enviados e os e-mails enviados anteriormente.
- Define o assunto e o corpo do e-mail que será enviado.

2. Função `enviar_email(to)`:

- **Verificação de E-mail Duplicado:** A função verifica se o e-mail já foi enviado anteriormente, consultando as listas de e-mails enviados. Se o e-mail já estiver na lista, a função retorna sem enviar o e-mail novamente.
- **Criação da Mensagem:** Se o e-mail ainda não foi enviado, a função cria uma mensagem de e-mail com o assunto, remetente e destinatário especificados. O corpo da mensagem é definido.
- **Envio do E-mail:** Conecta-se ao servidor SMTP do Gmail, inicia a criptografia com `s.starttls()`, faz login com as credenciais fornecidas e envia o e-mail para o destinatário.
- **Registro do E-mail Enviado:** Após o envio, o endereço do destinatário é adicionado à lista de e-mails enviados e registrado em um arquivo de texto `emailenviado.txt` para evitar duplicações futuras.

3. Leitura de E-mails Enviados Anteriormente:

- O código tenta abrir o arquivo `emailenviado.txt` para ler os endereços de e-mail que já foram enviados. Se o arquivo não existir, ele é criado vazio.

4. Leitura da Lista de Destinatários:

- Abre o arquivo `enviaremail.csv`, que contém os endereços de e-mail dos destinatários.
- Lê a lista de e-mails e percorre cada endereço, chamando a função `enviar_email()` para enviar o e-mail.
- Introduz uma pausa de 2 segundos entre os envios para evitar sobrecarregar o servidor SMTP.

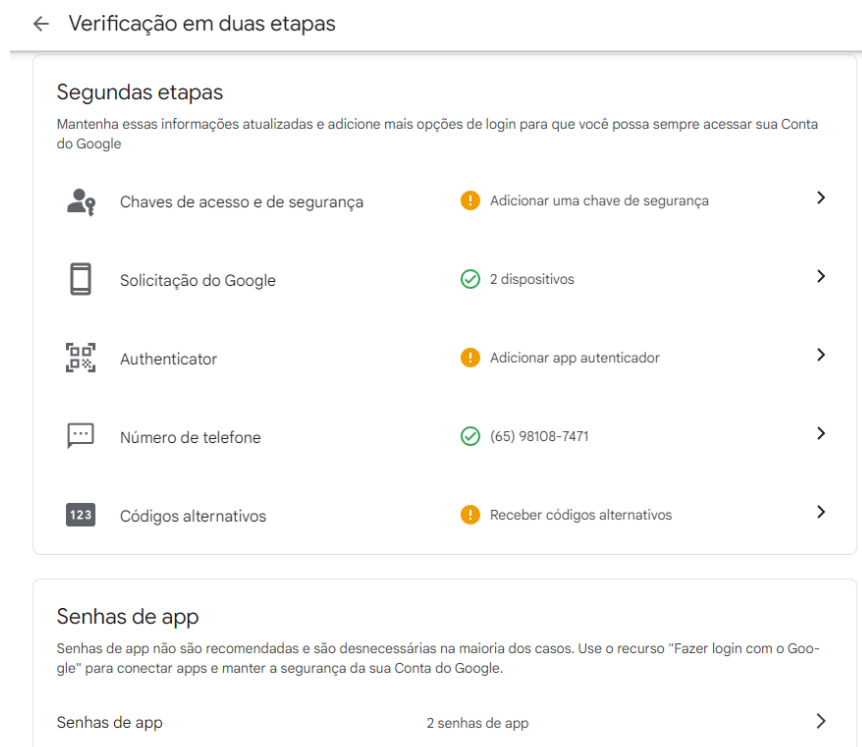
4 RESULTADO E DISCUSSÕES

4.1 RESULTADOS DA AUTOMATIZAÇÃO DE E-MAILS

A implementação da automação de e-mails utilizando a biblioteca smtplib demonstrou ser eficiente e robusta. A configuração do servidor SMTP, a criação e o envio de mensagens funcionaram conforme o esperado. A integração com o Gmail, utilizando autenticação de dois fatores e senha de aplicativo, garantiu a segurança das credenciais e a funcionalidade do envio automático. O código foi capaz de enviar mensagens para uma lista de contatos com sucesso e verificar e-mails duplicados para evitar envios repetitivos.

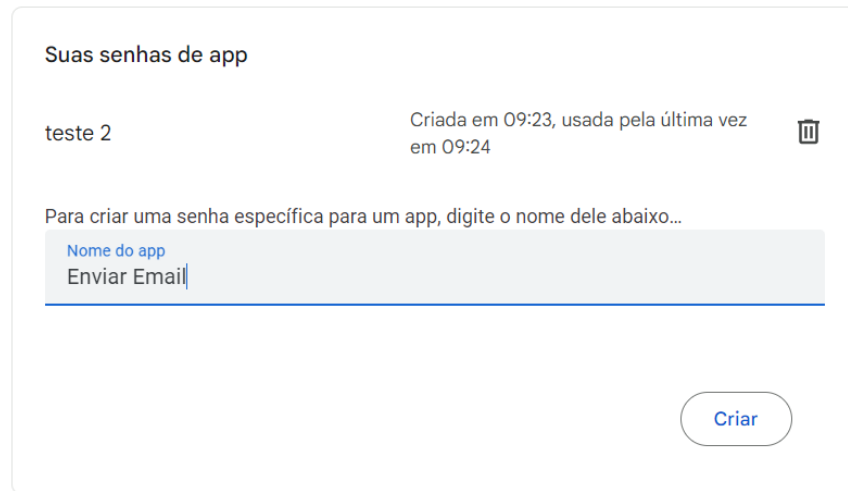
No entanto, algumas dificuldades foram encontradas. A autenticação de dois fatores do Gmail é essencial; sem ela, o código não funcionará. A autenticação gera uma senha temporária válida por 30 dias, após os quais uma nova senha deve ser gerada.

Figura 180 - Tela de Senhas de Aplicativo na Autenticação de Dois Fatores do Google



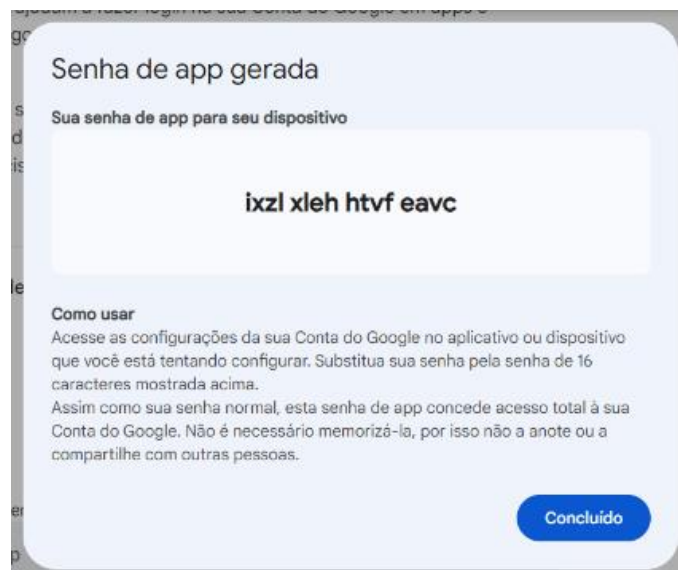
Fonte: Figura do Autor(a)

Figura 21 - Tela com Lista de Senhas de Aplicativo na Autenticação de Dois Fatores do Google



Fonte: Figura do Autor(a)

Figura 22 - Senha de Aplicativo Gerada



Fonte: Figura do Autor(a)

Além disso, houve um problema com o formato dos arquivos CSV. O Excel pode exportar arquivos CSV, mas muitos arquivos usam ponto e vírgula (;) em vez de vírgula (,). Caso o arquivo esteja separado por ponto e vírgula, é necessário substituí-lo manualmente por vírgula no Bloco de Notas para evitar erros no código.

Figura 23 - Arquivo CSV

	A	B
1	João Silva,joao.silva@example.com	
2	Ana,ana.souza@corporate.org	
3	Fernanda,fernanda.lima@mailservice.com	

Fonte: Figura do Autor(a)

Figura 24 - Arquivo CSV com Erro

	A	B
1	Laura Email Pessoal;laurafprocopio@gmail.com	
2	Laura Email Estudante;laura.p@estudante.ifmt.edu.br	
3	Laura Email Colaborador;laura.f@colaborador.ifmt.edu.br	

Fonte: Figura do Autor(a)

O código lê o arquivo CSV, inicia o envio das mensagens e gera um arquivo com todos os e-mails enviados. Caso algum e-mail não seja encontrado, o código envia uma notificação para o e-mail utilizado e exibe o erro no terminal. Se o código for executado novamente e um e-mail já tiver sido enviado, o terminal informará que o e-mail já foi enviado, evitando envios duplicados, a menos que a lista inicial seja excluída.

Figura 25 - Leitura do arquivo CSV demonstrada no Terminal

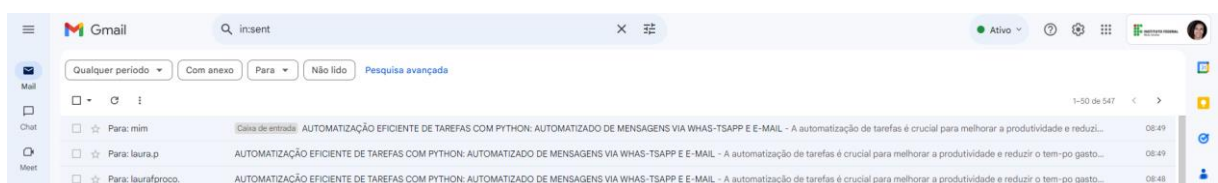
```

PS D:\Metodologia de Pesquisa Cientifica\testes em python> & "d:/Metodologia de Pesquisa Cientifica/testes em python/myenv/Scripts/python.exe" "d:/Metodologia de Pesquisa Cientifica/testes em python/enviaremailsmtplib.py"
Email enviado - laurafprocopio@gmail.com
Email enviado - laura.p@estudante.ifmt.edu.br
Email enviado - laura.f@colaborador.ifmt.edu.br
PS D:\Metodologia de Pesquisa Cientifica\testes em python>

```

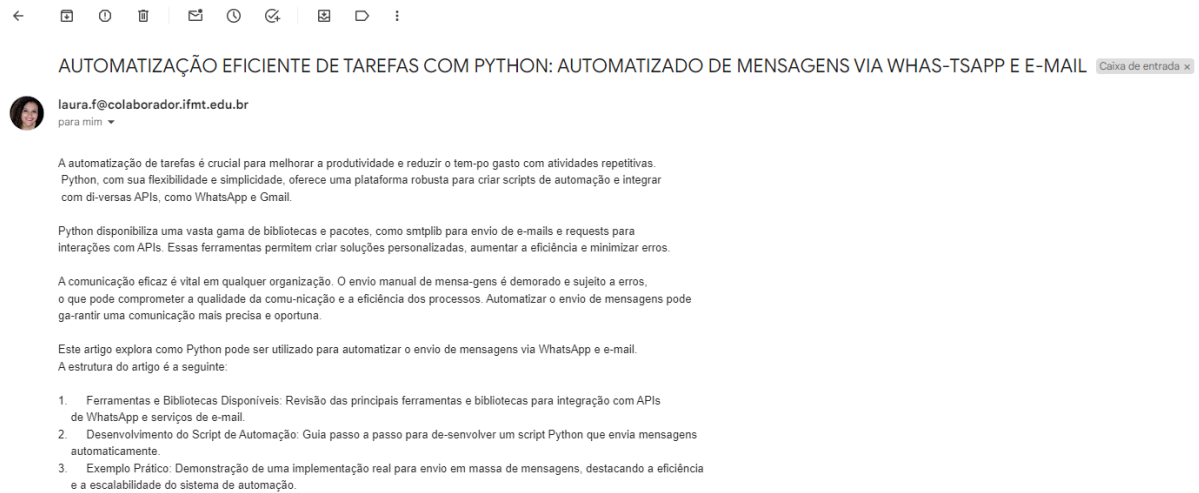
Fonte: Figura do Autor(a)

Figura 26 - Demonstração de Que Todos os E-mails Foram Enviados, Exibido no E-mail do Remetente



Fonte: Figura do Autor(a)

Figura 27 - Demonstração do E-mail Recebido pelo Destinatário



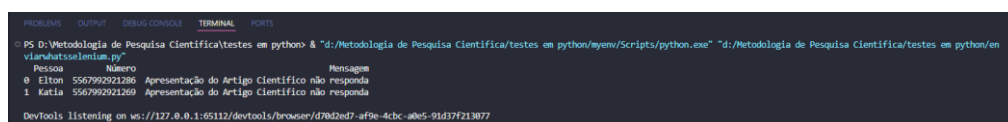
Fonte: Figura do Autor(a)

4.2 RESULTADOS DA AUTOMATIZAÇÃO DE MENSAGENS VIA WHATSAPP

A automação de mensagens via WhatsApp foi realizada utilizando a biblioteca Selenium para interagir com a interface do WhatsApp Web. O uso do Selenium permitiu o envio de mensagens para múltiplos contatos de forma automatizada, apesar de exigir mais configuração inicial e manutenção contínua devido às possíveis mudanças na interface do WhatsApp Web. É importante destacar que o Google Chrome e o WhatsApp Web devem estar atualizados para a versão mais recente; caso contrário, o código pode gerar erros. Esse foi uma das dificuldades encontradas no caminho

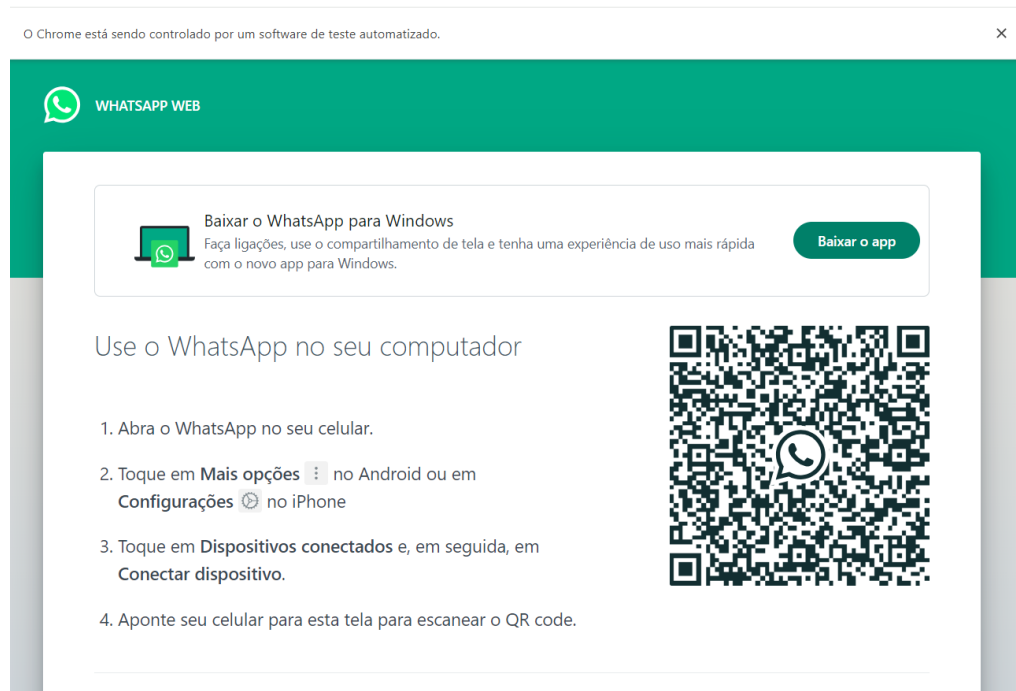
Como podemos observar nos resultados, a automação com Selenium se mostrou eficaz e funcional. No entanto, o WhatsApp impõe algumas restrições em relação ao envio de mensagens em lote, permitindo o envio de até 1000 mensagens por dia. Conforme demonstrado no código, o processo começa com a leitura de um arquivo no formato XLSX, onde são extraídos o nome, número de telefone e a mensagem a ser enviada. Em seguida, o WhatsApp Web é aberto automaticamente pelo script. Após a conexão ao WhatsApp Web, o script inicia o envio das mensagens.

Figura 19 - Leitura do arquivo XLSX demonstrada no Terminal



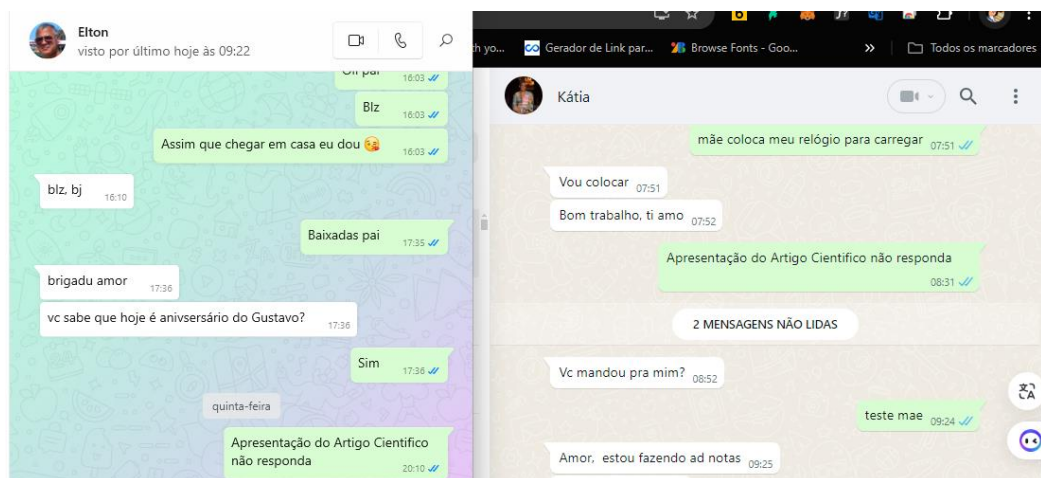
Fonte: Figura do Autor(a)

Figura 20 - Interface do WhatsApp Web



Fonte: Figura do Autor(a)

Figura 21 - Resultado dos Envios



Fonte: Figura do Autor(a)

5 CONSIDERAÇÕES FINAIS

A automatização de tarefas, como o envio de mensagens via e-mail e WhatsApp, desempenha um papel crucial na melhoria da produtividade e na redução de erros associados à comunicação manual. A implementação de scripts de automação utilizando Python demonstrou ser uma abordagem eficiente e robusta para lidar com essas tarefas.

Automatização de E-mails: A utilização da biblioteca `smtplib` para automatizar o envio de e-mails provou ser eficaz. A configuração do servidor SMTP e a integração com o Gmail, através da autenticação de dois fatores e senhas de aplicativo, garantiu a segurança das credenciais e a funcionalidade do sistema de envio automático. Apesar dos desafios encontrados, como a necessidade de gerar novas senhas temporárias e ajustar o formato dos arquivos CSV, o sistema conseguiu enviar e-mails com sucesso e evitar duplicações. O código desenvolvido é capaz de ler listas de contatos, enviar mensagens e registrar e-mails enviados para prevenir envios repetitivos.

Automatização de Mensagens via WhatsApp: A automação do envio de mensagens pelo WhatsApp foi realizada utilizando a biblioteca Selenium, permitindo a interação com a interface do WhatsApp Web. Esta abordagem, embora eficaz, apresentou desafios como a necessidade de manutenção contínua devido a possíveis alterações na interface do WhatsApp Web. Além disso, o sistema enfrenta restrições impostas pelo WhatsApp, como o limite de envio de até 1000 mensagens por dia. A automação foi realizada com base na leitura de arquivos XLSX, extraindo informações de contatos e mensagens para envio automático.

Em resumo, a implementação de scripts de automação em Python para e-mails e WhatsApp demonstra a viabilidade de melhorar a eficiência e reduzir o erro humano em processos de comunicação. As ferramentas e técnicas discutidas oferecem uma base sólida para futuras melhorias e adaptações, destacando o potencial do Python na automação de tarefas rotineiras e na integração com diversas APIs.

6 REFERÊNCIA

FONSECA JUNIOR, C. da S.; FONSECA, E. A. de A. F.; SARMENTO, G. H. B. *Uso do Python na automação empresarial*. Universidade do Mato Grosso do Sul. Disponível em <https://periodicos.ufms.br/index.php/EIGEDIN/article/view/16946/11595>. Acesso em: s.d.

REIS, P. L. *Automatize Tarefas Maçantes com Python*. São Paulo: Editora TechBooks, 2019. Disponível em: [https://github.com/free-educu/books/blob/main/books/Livro%20de%20Python%20\(Automatize%20tarefas%20ma%C3%A7antes\).pdf](https://github.com/free-educu/books/blob/main/books/Livro%20de%20Python%20(Automatize%20tarefas%20ma%C3%A7antes).pdf).

TWILIO. *Twilio API Documentation*. Disponível em: <https://www.twilio.com/docs/whatsapp>. Acesso em: s.d.

GOOGLE. *Gmail API Documentation*. Disponível em:
<https://developers.google.com/gmail/api>. Acesso em: s.d.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). *NBR 14724:2002 - Informação e documentação - Trabalhos acadêmicos - Apresentação*. Rio de Janeiro, 2002.

PYTHON SOFTWARE FOUNDATION. *Python Documentation*. Disponível em:
<https://docs.python.org>. Acesso em: s.d.

KORN, J. *Introduction to Python's SMTP Library*. Disponível em:
<https://realpython.com/python-send-email/>. Acesso em: s.d.

SELENIUM. *Selenium WebDriver Documentation*. Disponível em:
<https://www.selenium.dev/documentation/en/>. Acesso em: s.d.