Documentação do funcionamento do Bot do Whatsapp de notificações e entrega de relatórios automatizados

Scripts Importantes e Funcionamento Base

Autor: Carlos Vinícius dos Santos

Natal - RN

Funcionamento Geral

O bot do whatsapp utiliza o whatsapp web através do selenium, que é uma ferramenta de automação de tarefas utilizando o navegador, para poder automatizar a entrega via whats de alguns relatórios e também de notificações, ambos gerados através de scripts python, que pegam os dados de diversas fontes, realizam o tratamento e constroem de forma automática a estrutura das notificações e os relatórios em PDF, para posteriormente enviar para o destinatário correto.

Utilizando a funcionalidade de ler a última mensagem recebida o bot é capaz de executar scripts pré definidos utilizando palavras chave para identificar qual deve ser executado, através disso é possível realizar solicitações ao bot a qualquer momento para gerar um relatório ou executar qualquer outra tarefa que pode ser realizada através de scripts.

As notificações são geradas e enviadas à medida que o bot executa periodicamente os scripts, que realizam a captura e análise dos dados para identificar mudanças em nossas bases de dados, com o objetivo de perceber alterações específicas que nos interessam e direcioná-las para quem tem lidar com aquelas informações.

Script - classe_whats

Esse script possui a classe Zapbot, que é responsável por controlar toda a parte do selenium, com métodos que possibilitam toda a manipulação do whatsapp web através do navegador firefox, sendo possível transitar entre conversas, perceber e abrir novas mensagens recebidas, enviar mensagens e arquivos.

Métodos:

__init__ (): no método construtor são realizadas as pré configurações para a inicialização do bot, em que é utilizado um profile, que serve para manter a sessão do whatsweb logada, em seguida o bot inicia e abre a página do whatsapp web onde será possível a partir daí utilizar os outros métodos.

inicia_conversa (numero,mensagem): Inicia uma nova conversa com um novo número e envia uma mensagem inicial, passada por parâmetro, para poder futuramente ter essa nova conversa salva para realizar o envio de notificações ou receber solicitações caso seja permitido a esse número.

abre_conversa (contato): Abre uma conversa já iniciada, simulando o ato de buscar por um contato e clicar em sua aba de conversa, como mostra a figura 1.

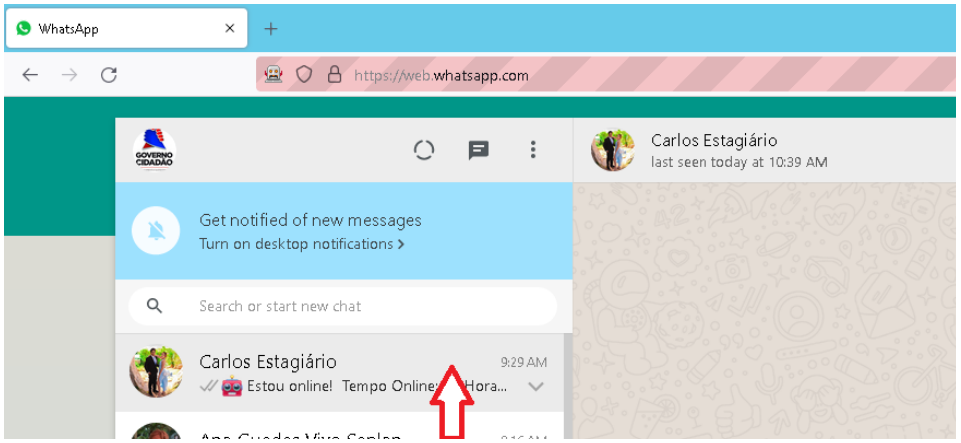

Figura 1 - abre conversa

abre_notificacao (): Verifica se existe alguma nova notificação de mensagem, caso exista ele clica na notificação para abrir a conversa, como mostra a figura 2.

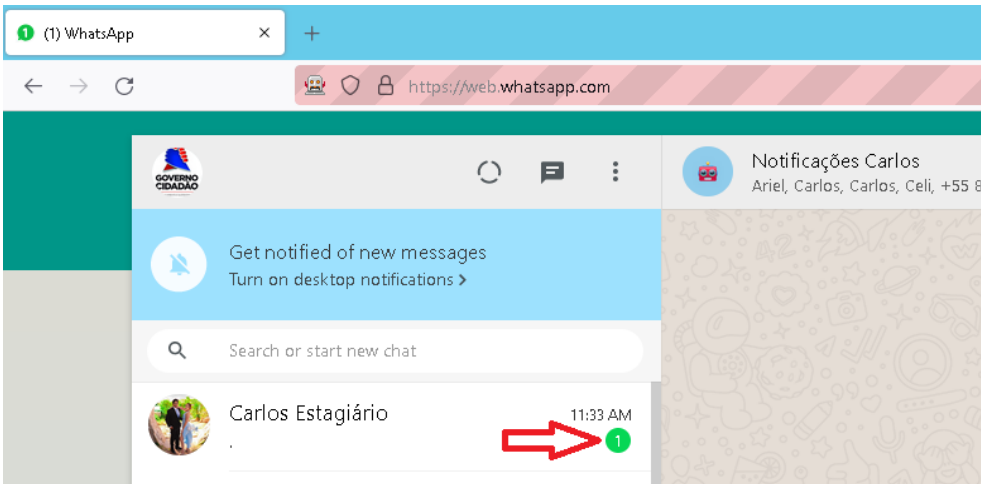

Figura 2 - abre notificação

esta_na_lista (lista): Verifica se o contato que está atualmente com a conversa aberta está na lista de permitidos, para posteriormente validar ou não suas solicitações ao bot.

conversa_aberta (): Retorna qual o nome do grupo ou contato que o bot está com a conversa aberta naquele momento, o nome é capturado de onde mostra a figura 3.

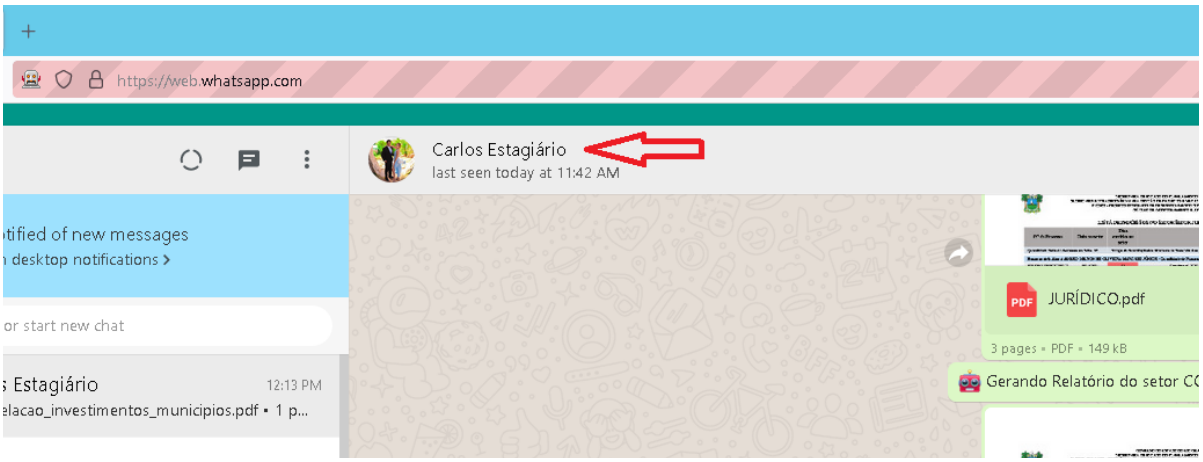

Figura 3 - Conversa Aberta

salvar_print (): Salva um printscreen da tela na pasta Prints, do momento em que o método é chamado.

anexa_media (caminho_arquivo): Anexa um arquivo ao whatsapp web, na conversa que está aberta no momento em que o método é chamado, passando o caminho absoluto, como mostra a figura 4 e 5.

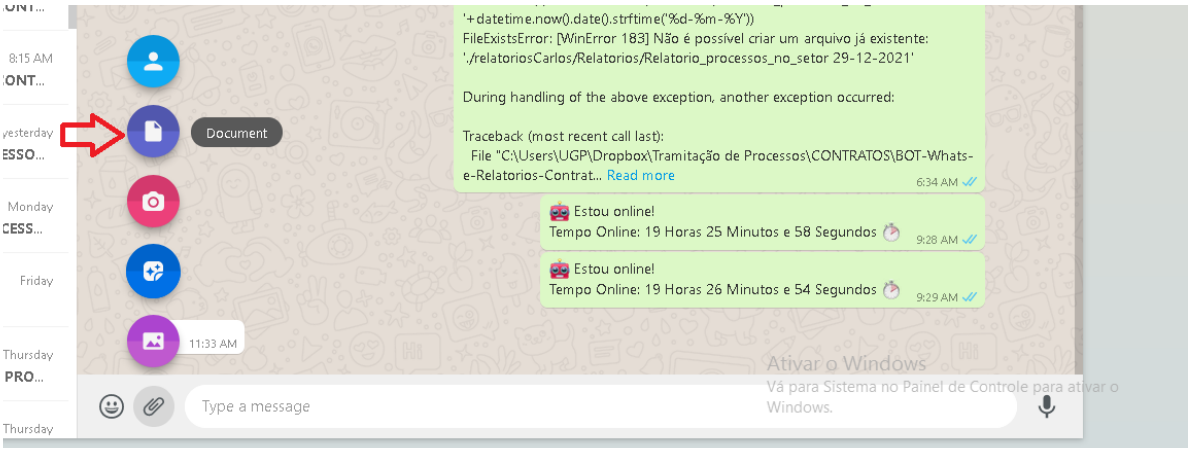

Figura 4 - Selecionando documento para anexar

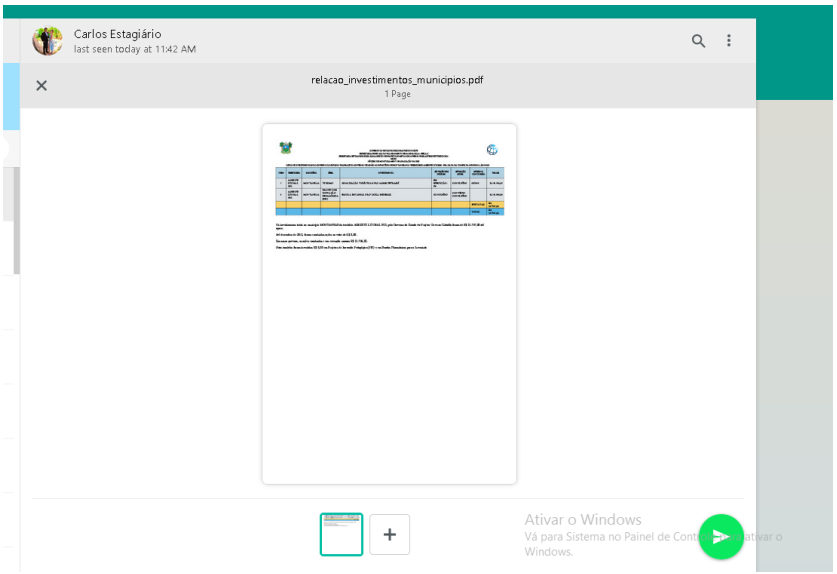

Figura 5 - Documento Anexado

envia_media (): Envia o arquivo anexado pelo método anterior, como mostra a figura 6.

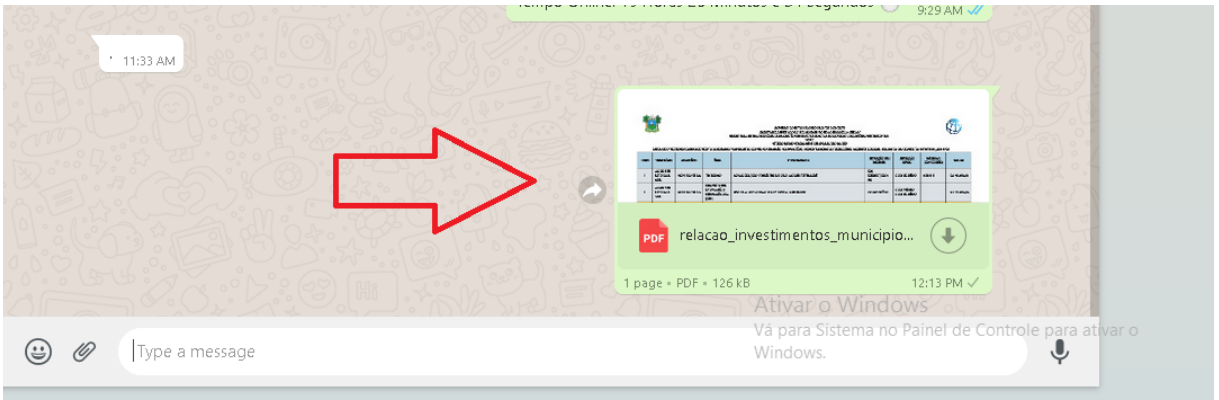

Figura 6 - Documento Enviado

envia_lista_arquivos (caminho,arquivos): Anexa e envia uma lista de arquivos que estão no mesmo diretório.

ultima_msg(): Lê e retorna a última mensagem recebida, na conversa que está aberta no momento em que o método é chamado, como na figura 7.

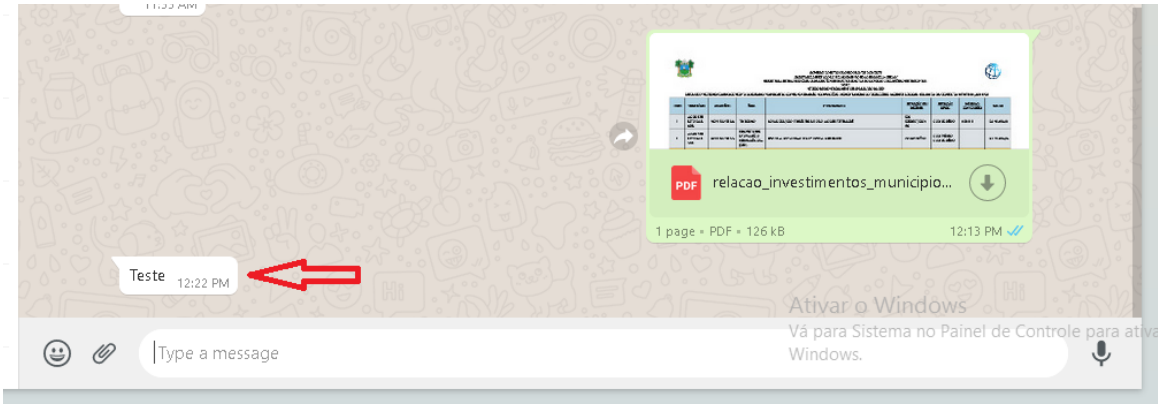

Figura 7 - última mensagem recebida

envia_msg (msg): Envia uma mensagem de texto (string), na conversa que está aberta no momento em que o método é chamado.

get_driver (): Retorna a variável driver do Selenium.

reiniciar (self): Finaliza o driver do selenium, abre um novo terminal com o bot executando e por fim finaliza a si mesmo (sys.exit(0)), reiniciando o bot.

finalizar (self): Finaliza o driver do selenium e finaliza a si mesmo (sys.exit(0)).

Script - classe_comandos

A classe comandos é responsável por conter todos os **métodos** que serão chamados para executar os **comandos**, cada comando possui um método na classe Comandos, chamando outros scripts, que são importados no classe_comandos para poderem ser chamados quando necessário, e utilizando uma instância da classe Zapbot para enviar as mensagens e arquivos.

Comandos de Relatórios:

Os comandos de relatórios em geral seguem um formato padrão, em que é passada a mensagem que foi obtida através do bot, utilizando o método ultima_mensagem() mostrado anteriormente, que servirá para que sejam obtidos os parâmetros que serão repassados para o script que gera o relatório.

```python
def gerar_situacao_geral(self, mensagem):
    """ Relatório situação dos aditivos de revisão prévia. ->exemplos:!gerar_situacao_GERAL,consultor:CARLOS"""
    self.whats.envia_msg('🗎 Gerando Relatório de Situação GERAL dos aditivos!')

    consultor,tipo_aditivo,dias_max,ues = self.filtrar_lista(["consultor:",
                                                              "tipo_aditivo:",
                                                              "dias_max:",
                                                              "ues:"],mensagem)

    caminho,arquivo = gerar_relatorio_aditivos_GERAL(consultor,tipo_aditivo,dias_max,ues)
    self.whats.envia_lista_arquivos(caminho,[arquivo])
```

Figura 8 - Código do comando gerar_situacao_GERAL

Como é possível notar na figura 8 existe logo de início uma marcação que mostra qual a função daquele método e também um exemplo de comando para seu uso, em seguida é utilizado a instância do Zapbot para enviar uma mensagem de feedback informando que o relatório será gerado. Para realizar a captura dos parâmetros é utilizado o método filtrar_lista, da própria classe Comandos, em seguida eles são repassados chamando a função que irá gerar o relatório.

Para poder enviar o arquivo PDF do relatório já gerado é utilizada a função do zapbot envia_lista_arquivos, citada anteriormente, passando como parâmetro o caminho e o nome do arquivo que foi obtido como retorno da função que cria esse relatório.

Comandos Gerais:

Os comandos gerais são todos os que não são para gerar relatórios, existem comandos de configuração para o próprio bot, comandos de ajuda ao usuário ou também de solicitação de notificações. Alguns exemplos dos comandos mais úteis e importantes são:

enviar_notificacoes_e_mensagens_agendadas (): Executa todos os scripts de notificações e envia todas as mensagens que foram agendadas para antes daquela data e hora que o comando foi executado.

print (contato_printar): Printa a conversa com algum contato específico e envia a imagem para quem solicitou.

restart (): Reinicia o bot utilizando o método reiniciar da classe Zapbot.

agendar_mensagem (destino, conteudo, data): Agenda uma mensagem de texto para determinado contato, destino, com o conteúdo da mensagem e a data passados por parâmetro.

exibir_comandos (numero_comando, buscar): Envia uma mensagem de texto para a conversa que está aberta com uma lista contendo todos os comandos criados até o momento, essa lista é lida do script Bot_whats_geral e salva automaticamente toda vez que o bot é iniciado. Pode ser passado como parâmetro o número identificador do comando, dessa forma será enviado na mensagem somente aquele comando, ou também pode ser passado no parâmetro "buscar" uma palavra que servirá como filtro, de forma que somente os comandos com aquelas palavras irão ser enviados na mensagem.

exibir_comandos_exemplos (numero_comando, buscar): Funciona exatamente como o comando anterior com a diferença de que esse trás não só o comando como também exemplos de parâmetros que servirão como guia para quem não lembra de como deve ser usado o comando.

exibir_historico (): Envia uma mensagem de texto para a conversa aberta com um histórico dos 10 últimos comandos utilizados.

exibir_dados_processo (tabela, processo): Envia uma mensagem de texto para a conversa aberta com informações detalhadas sobre determinado processo de licitação ou contratos.

report(): Envia um da conversa aberta para um contato cadastrado como responsável pela manutenção do bot, esse comando serve para informar o responsável sobre possíveis falhas ou bugs no funcionamento do bot.

exibir_status(): Esse comando serve para ter um feedback rápido do bot, ele envia uma mensagem de texto para a conversa aberta dizendo a quanto tempo está executando.

Script - Bot_whats_geral

O script que será executado para iniciar todo o funcionamento do bot, ele é responsável por gerenciar todo o loop de funcionamento, chamando os outros scripts sempre que necessário. Nele existe uma estrutura em loop que procura por novas mensagens em suas conversas, utilizando **abre_notificacao (),** caso encontre uma nova mensagem ele irá ler, utilizando **ultima_msg(),** para depois comparar com os comandos já existentes, caso aquela mensagem seja um comando será chamado um dos métodos da **classe_comandos** referente a aquele comando que foi dado.

Lista de Permitidos:

Os comandos só serão aceitos se aquele contato ou grupo que o enviou estiver cadastrado em uma lista predefinida, que se encontra em uma planilha do excel, localizada em "Planilhas\Configurações BOT\Contatos_e_grupos_permitidos.xlsx", para ter a segurança e a certeza de seus dados não serem requisitados por alguém não autorizado.

Horário de Serviço:

As notificações são enviadas a cada período de tempo predeterminado se a hora no momento estiver entre os intervalos de horário de serviço, para evitar notificações que seriam ignoradas, o horário de serviço é predeterminado com o uso de variáveis de início e término de expediente, também é definido utilizando uma variável quais os dias da semana em que não haverá expediente, como por exemplo sábado e domingo, como mostra a figura 9.

```
# horário de serviço configurações
agora = datetime.now()
hora_inicio_expediente = datetime(agora.year,agora.month,agora.day,hour=7,minute=45)
hora_fim_expediente = datetime(agora.year,agora.month,agora.day,hour=18,minute=0)
dias_sem_expediente = ['Saturday' ,'Sunday']
```

Figura 9 - Configurações horário de serviço

Comandos Agendados:

É possível realizar também o agendamentos de comandos que serão executados semanalmente ou diariamente através de uma planilha no excel, que está na pasta localizada em "Planilhas\Configurações BOT\Comandos - Agendados.xlsx", que irá servir como cadastro para esses agendamentos, armazenando qual o comando que será utilizado; para qual contato será enviado o resultado do uso daquele comando; em quais horários aquele comando será executado e também em quais dias da semana, como mostra a figura 10.

Comandos	Contato	Hora	Dias
!gerar_juridico,agendado:True	Carlos Est	05:50,06:0	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
!gerar_contratos,agendado:True	Carlos Est	06:10,18:1	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
!gerar_setor:PRESTAÇÃO DE CONTAS SUBPROJETOS,agendado:True	Carlos Est	06:20,18:2	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
!gerar_setor:ENGENHARIA ESTRUTURANTE,agendado:True	Carlos Est	06:30,18:3	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday
!status	Carlos Est	09:28,09:2	Sunday,Monday,Tuesday,Wednesday,Thursday,Friday,Saturday

Figura 10 - Excel Comandos Agendados

Esses comandos serão percebidos pelo bot no horário em que foram agendados, e o contato que está definido será aberto, como se o comando fosse enviado por ele naquele momento.

Loop de Funcionamento Geral:

O loop irá funcionar continuamente a cada segundo, procurando por novos comandos e executando os scripts de notificações a cada determinado período de tempo, caso aconteça algum erro não previsto durante a execução daquela iteração do loop, será enviada uma mensagem de erro para o contato que gerou aquele erro e uma mensagem mais completa e explicativa para um contato predefinido para realizar a correção dos erros.

Funcionamento do loop:

- **While(True)** - Loop infinito, o trecho de código dentro dele irá executar sem fim até que ele seja finalizado por algum motivo.
    - **Verifica se está em horário de trabalho -** pega hora naquele momento e verifica se está no horário de trabalho definido nas configurações, guardando o resultado em uma variável.
    - **Lê última mensagem -** Irá utilizar o método **ultima_msg()** para ler a ultima mensagem enviada pela conversa que está aberta no momento.
    - **Verifica os agendamentos** - Verifica se nos agendamentos existe algum para aquele momento.
    - **Verifica se aquele contato está na lista dos permitidos**
        - **Verifica última mensagem -** Verifica se última mensagem é um comando e chama o método correspondente da **classe_comandos.**

    - **Envia novas notificações se houver -** a cada 5 minutos executa os scripts que procuram por notificações e as envia se houver alguma.
    - **Verifica se houve algum erro dentro do loop -** Caso houve algum erro será enviado para o contato de manutenção e para o contato que gerou aquele erro.