**DOCUMENTAÇÃO OFICIAL**

BOTS RESPONSÁVEIS PELA EXPORTAÇÃO DE DADOS DO SICONNET E ALIMENTAÇÃO DE DASHBOARDS COTRASA

*Atualizado em 08/05/2024*

Sumário

[CÓDIGOS GLOBAIS 2](#_Toc165988911)

[ViewFolder 2](#_Toc165988912)

[Library 7](#_Toc165988913)

[BOTS INDIVIDUAIS - Serviços 12](#_Toc165988914)

[Service Pack 12](#_Toc165988915)

[Apuração Débito Interno 12](#_Toc165988916)

[Garantia 13](#_Toc165988917)

[Ordens de Serviço em Aberto 13](#_Toc165988918)

[Ordens de Serviço Abertas em um Determinado Período 14](#_Toc165988919)

[Ordens de Serviço Fechadas em um Determinado Período 14](#_Toc165988920)

[Ranking de Clientes 15](#_Toc165988921)

[Aproveitamento Tempo Mecânicos 15](#_Toc165988922)

[Checklist 16](#_Toc165988923)

[Orçamento 16](#_Toc165988924)

[BOTS INDIVIDUAIS – Estoque 17](#_Toc165988925)

[Itens Não Movimentados 17](#_Toc165988926)

[Vendas, Lucros e Comissões 17](#_Toc165988927)

[Posição Geral Estoque 18](#_Toc165988928)

[Relação de Pedidos Por Fornecedor 18](#_Toc165988929)

[BOTS INDIVIDUAIS – Faturamento 19](#_Toc165988930)

[Tempo Permanência 19](#_Toc165988931)

[Origem Serviços Aplicados 19](#_Toc165988932)

[Ordens de Serviço em Aberto - SINTÉTICO 20](#_Toc165988933)

[Ordens de Serviço Abertas em um Determinado Período - SINTÉTICO 20](#_Toc165988934)

[Ordens de Serviço Fechadas em um Determinado Período - SINTÉTICO 21](#_Toc165988935)

*Obs: Os bots rodam todos os dias úteis às 5:30 da manhã.*

# CÓDIGOS GLOBAIS

Códigos que são usados ao decorrer de todos os códigos pois ficam no caminho raiz do bot

## ViewFolder

*O script é responsável por renomear os arquivos que são baixados pelos bots e transferir para os diretórios corretos (o desenvolvimento do padrão foi realizado individualmente baseado na necessidade de cada relatório)*

from ..Library.library import \*

*# ------------------------------------- SUPER ------------------------------------- #*

class MyHandler(*FileSystemEventHandler*):

    def \_\_init\_\_(self, src\_folder, dest\_folder):

*super*().\_\_init\_\_()

        self.src\_folder = src\_folder

        self.dest\_folder = dest\_folder

        self.log\_file = os.path.join(dest\_folder, "log", "log.txt")  *# Caminho do arquivo de log*

        self.last\_working\_day = get\_last\_working\_day()  *# Obter o último dia útil*

*# Criar a pasta de log, se não existir*

        log\_folder = os.path.join(dest\_folder, "log")

        if not os.path.exists(log\_folder):

            os.makedirs(log\_folder)

    def \_log(self, message):

        with open(self.log\_file, "a") as f:

            f.write(f"{time.strftime('%Y-%m-%d %H:%M:%S')} - {message}\n")

    def on\_modified(self, event):

        if event.is\_directory:

            return

        filename = os.path.basename(event.src\_path)

        time.sleep(5)

        if filename.lower().startswith("service") and filename.lower().endswith(".txt"):

            self.\_process\_service(filename)

        elif filename.lower().startswith("rankingclientes") and filename.lower().endswith(".xml"):

            self.\_process\_ranking\_clientes(filename)

        elif filename.lower().startswith("aproveittempomecanico") and filename.lower().endswith(".xml"):

            self.\_process\_aproveittempomec(filename)

        elif filename.lower().startswith("checklist") and filename.lower().endswith(".xml"):

            self.\_process\_checklist(filename)

        elif filename.lower().startswith("pedidoporfornecedor") and filename.lower().endswith(".xml"):

            self.\_process\_pedido\_fornecedor(filename)

        elif filename.lower().startswith("osemaberto\_sintetico") and filename.lower().endswith(".xml"):

            self.\_process\_os\_em\_aberto(filename)

        else:  *# Lógica geral para outros tipos de arquivo (xml e txt)*

            self.\_process\_generic(filename)

*# ------------------------------------- ARQUIVOS SERVIÇOS ------------------------------------- #*

    def \_rename\_service\_pack(self, filename):

        service\_file\_number = ''.join(filter(*str*.isdigit, filename))

        return f"service{service\_file\_number} - {self.last\_working\_day.strftime('%m.%Y')}{os.path.splitext(filename)[1]}"

    def \_rename\_ranking\_clientes(self, filename):

        ranking\_file\_number = ''.join(filter(*str*.isdigit, filename))

        return f"RankingClientes{ranking\_file\_number} - {self.last\_working\_day.strftime('%d.%m.%Y')}{os.path.splitext(filename)[1]}"

    def \_rename\_checklist(self, filename):  *# Nova função para renomear o checklist*

        checklist\_file\_number = ''.join(filter(*str*.isdigit, filename))

        return f"Checklist{checklist\_file\_number} - {self.last\_working\_day.strftime('%d.%m.%Y')}{os.path.splitext(filename)[1]}"

    def \_rename\_aproveittempomec(self, filename):

        return f"AproveitTempoMecanico - {self.last\_working\_day.strftime('%m.%Y')}{os.path.splitext(filename)[1]}"

    def \_process\_service(self, filename):

        time.sleep(3)

        new\_filename = self.\_rename\_service\_pack(filename)

        new\_file\_path = os.path.join(self.dest\_folder, "ServicePack", new\_filename)

        try:

            time.sleep(5)

            shutil.move(os.path.join(self.src\_folder, filename), new\_file\_path)

            print(f"Arquivo {filename} movido para a pasta ServicePack e renomeado para {new\_filename}")

            self.\_log(f"Arquivo {filename} movido para a pasta ServicePack e renomeado para {new\_filename}")  *# Adiciona log*

        except (*PermissionError*, *FileNotFoundError*) as e:

            print(f"Erro ao processar arquivo {filename}: {e}")

            self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

        except *Exception* as e:

            print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

            self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

    def \_process\_ranking\_clientes(self, filename):

        time.sleep(3)

        new\_filename = self.\_rename\_ranking\_clientes(filename)

        new\_file\_path = os.path.join(self.dest\_folder, "RankingClientes", new\_filename)

        try:

            time.sleep(5)

            shutil.move(os.path.join(self.src\_folder, filename), new\_file\_path)

            print(f"Arquivo {filename} movido para a pasta RankingClientes e renomeado para {new\_filename}")

            self.\_log(f"Arquivo {filename} movido para a pasta RankingClientes e renomeado para {new\_filename}")  *# Adiciona log*

        except (*PermissionError*, *FileNotFoundError*) as e:

            print(f"Erro ao processar arquivo {filename}: {e}")

            self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

        except *Exception* as e:

            print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

            self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

    def \_process\_aproveittempomec(self, filename):

        time.sleep(3)

        aproveit\_tempo\_mec\_folder = os.path.join(self.dest\_folder, "AproveitTempoMecanico")

        new\_filename = self.\_rename\_aproveittempomec(filename)

        new\_file\_path = os.path.join(aproveit\_tempo\_mec\_folder, new\_filename)

        if not os.path.exists(aproveit\_tempo\_mec\_folder):

            os.makedirs(aproveit\_tempo\_mec\_folder)

        try:

            shutil.move(os.path.join(self.src\_folder, filename), new\_file\_path)

            print(f"Arquivo {filename} movido para {aproveit\_tempo\_mec\_folder} e renomeado para {new\_filename}")

            self.\_log(f"Arquivo {filename} movido para {aproveit\_tempo\_mec\_folder} e renomeado para {new\_filename}")  *# Adiciona log*

        except (*PermissionError*, *FileNotFoundError*) as e:

            print(f"Erro ao processar arquivo {filename}: {e}")

            self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

        except *Exception* as e:

            print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

            self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

    def \_process\_checklist(self, filename):

        time.sleep(3)

        new\_filename = self.\_rename\_checklist(filename)

        new\_file\_path = os.path.join(self.dest\_folder, "Checklist", new\_filename)

        try:

            time.sleep(5)

            shutil.move(os.path.join(self.src\_folder, filename), new\_file\_path)

            print(f"Arquivo {filename} movido para a pasta Checklist e renomeado para {new\_filename}")

            self.\_log(f"Arquivo {filename} movido para a pasta Checklist e renomeado para {new\_filename}")  *# Adiciona log*

        except (*PermissionError*, *FileNotFoundError*) as e:

            print(f"Erro ao processar arquivo {filename}: {e}")

            self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

        except *Exception* as e:

            print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

            self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

*# ------------------------------------- ARQUIVOS ESTOQUE------------------------------------- #*

    def \_rename\_pedido\_fornecedor\_pack(self, filename):

        pedido\_fornecedor\_file\_number = ''.join(filter(*str*.isdigit, filename))

        last\_working\_day = get\_last\_working\_day()  *# Função da biblioteca*

        return f"PedidoPorFornecedor{pedido\_fornecedor\_file\_number} - {last\_working\_day.strftime('%m.%Y')}{os.path.splitext(filename)[1]}"

    def \_rename\_os\_em\_aberto\_s(self, filename):

        return "OSemAberto\_Sintetico"

    def \_process\_pedido\_fornecedor(self, filename):

        pedido\_fornecedor\_new\_filename = self.\_rename\_pedido\_fornecedor\_pack(filename)

        pedido\_fornecedor\_new\_file\_path = os.path.join(self.dest\_folder, "PedidoPorFornecedor", pedido\_fornecedor\_new\_filename)

        try:

            shutil.move(os.path.join(self.src\_folder, filename), pedido\_fornecedor\_new\_file\_path)

            print(f"Arquivo {filename} movido para a pasta PedidoPorFornecedor e renomeado para {pedido\_fornecedor\_new\_filename}")

            self.\_log(f"Arquivo {filename} movido para a pasta PedidoPorFornecedor e renomeado para {pedido\_fornecedor\_new\_filename}")  *# Adiciona log*

        except (*PermissionError*, *FileNotFoundError*) as e:

            print(f"Erro ao processar arquivo {filename}: {e}")

            self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

        except *Exception* as e:

            print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

            self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

    def \_process\_os\_em\_aberto(self, filename):

        os\_em\_aberto\_s\_folder = os.path.join(self.dest\_folder, "OSemAberto\_Sintetico")

        os\_em\_aberto\_s\_new\_filename = self.\_rename\_os\_em\_aberto\_s(filename)

        os\_em\_aberto\_s\_new\_file\_path = os.path.join(os\_em\_aberto\_s\_folder, os\_em\_aberto\_s\_new\_filename + ".xml")

        if not os.path.exists(os\_em\_aberto\_s\_folder):

            os.makedirs(os\_em\_aberto\_s\_folder)

        try:

            shutil.move(os.path.join(self.src\_folder, filename), os\_em\_aberto\_s\_new\_file\_path)

            print(f"Arquivo {filename} movido para {os\_em\_aberto\_s\_folder} e renomeado para {os\_em\_aberto\_s\_new\_filename}")

            self.\_log(f"Arquivo {filename} movido para {os\_em\_aberto\_s\_folder} e renomeado para {os\_em\_aberto\_s\_new\_filename}")  *# Adiciona log*

        except (*PermissionError*, *FileNotFoundError*) as e:

            print(f"Erro ao processar arquivo {filename}: {e}")

            self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

        except *Exception* as e:

            print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

            self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

*# ------------------------------------- GENERICOS ------------------------------------- #*

    def \_process\_generic(self, filename):

        time.sleep(3)

        if filename.lower().endswith((".xml", ".txt")):

            folder\_name = os.path.splitext(filename)[0]

            new\_folder\_path = os.path.join(self.dest\_folder, folder\_name)

            if not os.path.exists(new\_folder\_path):

                os.makedirs(new\_folder\_path)

            new\_filename = f"{folder\_name}{self.last\_working\_day.strftime('%d.%m.%Y')}{os.path.splitext(filename)[1]}"

            new\_file\_path = os.path.join(new\_folder\_path, new\_filename)

            try:

                shutil.move(os.path.join(self.src\_folder, filename), new\_file\_path)

                print(f"Arquivo {filename} movido para {new\_folder\_path} e renomeado para {new\_filename}")

                self.\_log(f"Arquivo {filename} movido para {new\_folder\_path} e renomeado para {new\_filename}")  *# Adiciona log*

            except (*PermissionError*, *FileNotFoundError*) as e:

                print(f"Erro ao processar arquivo {filename}: {e}")

                self.\_log(f"Erro ao processar arquivo {filename}: {e}")  *# Adiciona log de erro*

            except *Exception* as e:

                print(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")

                self.\_log(f"Ocorreu um erro ao mover o arquivo {filename}: {e}")  *# Adiciona log de erro*

*# ------------------------------------- EXECUÇÃO ------------------------------------- #*

if \_\_name\_\_ == "\_\_main\_\_":

    src\_folder = "C:\\Users\\Automação\\Downloads"

    dest\_folder = "Y:\\CORPORATIVO\\INTELIGENCIA\\BI\\PowerBI - Geral Cotrasa\\Arquivos"

    folder\_to\_watch = src\_folder

    event\_handler = MyHandler(src\_folder, dest\_folder)

    observer = Observer()

    observer.schedule(event\_handler, folder\_to\_watch, recursive=False)

    observer.start()

    stop\_file = "stop.txt"

    try:

        while not os.path.exists(stop\_file):

            time.sleep(1)

    except *KeyboardInterrupt*:

        pass

    observer.stop()

    observer.join()

## Library

A *library* não é nada mais nada menos que a *biblioteca* de funções globais, ou seja, funções que são utilizadas ao decorrer dos códigos e também é responsável pelas importações de bibliotecas utilizadas nos códigos em um lugar só.

*# Bibliotecas para automação e interação com páginas da web*

import pyautogui

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.common.by import By

from selenium.common.exceptions import TimeoutException, NoSuchElementException

from selenium.webdriver.support.ui import WebDriverWait

from selenium.webdriver.support import expected\_conditions as EC

from webdriver\_manager.chrome import ChromeDriverManager

from selenium.webdriver.chrome.service import Service

from selenium.webdriver.chrome.options import Options

from apscheduler.schedulers.background import BackgroundScheduler

from apscheduler.triggers.cron import CronTrigger

from flask\_apscheduler import APScheduler

*# Bibliotecas para manipulação de dados*

import pandas as pd

import os

import xml.etree.ElementTree as ET

import requests

*# Bibliotecas para manipulação de data e hora*

from datetime import datetime, timedelta

import time

import holidays as hdays

import schedule

*# Bibliotecas para criação de aplicativos web*

from flask import Flask, render\_template, request, jsonify

from flask\_socketio import SocketIO, send

from webdriver\_manager.chrome import ChromeDriverManager

*# Bibliotecas para monitoramento de arquivos e diretórios*

import shutil

from watchdog.observers import Observer

from watchdog.events import FileSystemEventHandler

*# Biblioteca para manipulação de sockets e rede*

import socket

*# Biblioteca para execução de código em threads*

from threading import Thread

*# Biblioteca para ambiente de producao*

from waitress import serve

*# Inicialização do aplicativo Flask*

app = Flask(\_\_name\_\_, template\_folder="templates", static\_folder="static")

*# Função para obter o primeiro e o último dia do mês*

def get\_first\_last\_days\_of\_month():

    data\_atual = datetime.now()

    first\_day = data\_atual.replace(day=1)

    last\_day = first\_day + pd.offsets.MonthEnd(0)

    return first\_day, last\_day

*# Função para verificar se uma data é fim de semana*

def is\_weekend(date):

    return date.weekday() >= 5

*# Função para verificar se uma data é feriado*

def is\_holiday(date, holidays\_instance):

    return date in holidays\_instance

*# Função para obter o calendário filtrado sem fins de semana e feriados*

def get\_filtered\_calendar():

    data\_atual = datetime.now()

    start\_date = data\_atual - timedelta(days=365)

    end\_date = data\_atual - timedelta(days=1)

    calendar = pd.bdate\_range(start=start\_date, end=end\_date)

    holidays\_instance = hdays.HolidaysAS()

    filtered\_calendar = [

        date

        for date in calendar

        if not is\_holiday(date, holidays\_instance) and not is\_weekend(date)

    ]

    return filtered\_calendar

*# Função para obter o último dia útil*

def get\_last\_working\_day():

    filtered\_calendar = get\_filtered\_calendar()

    if filtered\_calendar:

        return filtered\_calendar[-1]

    return None

*# Função para formatar uma data no formato "ddMMyyyy"*

def format\_date(date):

    return date.strftime("%d%m%Y")

*# Função para inicializar o navegador Chrome*

def initialize\_browser():

*# Configura as opções do Chrome*

    chrome\_options = Options()

    chrome\_options.add\_argument("user-data-dir=C:\\Users\\Automação\\AppData\\Local\\Google\\User Data\\Default")

    chrome\_options.add\_argument("--disable-blink-features=AutomationControlled")

    chrome\_options.add\_argument("--disable-extensions")

    chrome\_options.add\_argument("--disable-gpu")

    chrome\_options.add\_argument("--disable-dev-shm-usage")

    chrome\_options.add\_argument("--no-sandbox")

*# Configura preferências para downloads*

    prefs = {"safebrowsing.enabled": "false"}

    chrome\_options.add\_experimental\_option("prefs", prefs)

*# Inicia o serviço do Chrome*

    downloadBrowser = Service(ChromeDriverManager().install())

*# Inicia o navegador Chrome com as opções configuradas*

    browser = webdriver.Chrome(service=downloadBrowser, options=chrome\_options)

    browser.maximize\_window()

    return browser

*# Função para fazer login em uma página da web*

def login(browser):

    browser.get("https://siconnet.scania.com.br/sicomnet4/newcotrasa/sicomweb.gen.gen.apl.Sicomweb.cls")

    wait = WebDriverWait(browser, 20)

    try:

        element = wait.until(

            EC.visibility\_of\_element\_located((By.XPATH, '//\*[@id="control\_11"]'))

        )

*#element.send\_keys("CTSAUT")*

        element.send\_keys("CTSBOT")

    except TimeoutException:

        error\_message = "Não foi possível acessar o SiconNet"

        return render\_template(

            "interface.html",

            resultado\_geralestoque="Erro na Execução",

            error\_message=error\_message,

        )

*#browser.find\_element(By.XPATH, '//\*[@id="control\_15"]').send\_keys("Cotrasa2023\*\*")*

    browser.find\_element(By.XPATH, '//\*[@id="control\_15"]').send\_keys("Cotrasa2024\*\*")

    browser.find\_element(By.ID, 'botaoLogin').click()

*# Função para alternar para uma janela específica do navegador*

def switch\_to\_window(browser, url):

    browser.current\_window\_handle

    wids = browser.window\_handles

    for window in wids:

        browser.switch\_to.window(window)

        if url in browser.current\_url == url:

            break

*# Função para alternar para uma janela específica do navegador com tentativas de retry*

def switch\_to\_window\_with\_retry(browser, url\_esperado):

    while True:

        try:

            janelas = browser.window\_handles

            for janela in janelas:

                browser.switch\_to.window(janela)

                if url\_esperado in browser.current\_url:

                    return

            time.sleep(10)

        except *Exception* as e:

            print("Erro:", e)

            time.sleep(10)

*# Função para encontrar uma janela específica do navegador*

def find\_window(browser, url):

    wids = browser.window\_handles

    for window in wids:

        browser.switch\_to.window(window)

        if url in browser.current\_url == url:

            break

*# Definindo num\_downs*

num\_downs1 = 1

num\_downs2 = 2

num\_downs3 = 3

num\_downs4 = 4

num\_downs5 = 5

num\_downs6 = 6

num\_downs7 = 7

num\_downs8 = 8

num\_downs9 = 9

num\_downs10 = 10

num\_downs11 = 11

num\_downs12 = 12

num\_downs13 = 13

num\_downs14 = 14

num\_downs15 = 15

num\_downs16 = 16

num\_downs17 = 17

num\_downs18 = 18

num\_downs19 = 19

num\_downs20 = 20

*# Definindo num\_tabs*

num\_tabs1 = 1

num\_tabs2 = 2

num\_tabs3 = 3

num\_tabs4 = 4

num\_tabs5 = 5

num\_tabs6 = 6

num\_tabs7 = 7

num\_tabs8 = 8

num\_tabs9 = 9

num\_tabs10 = 10

num\_tabs11 = 11

num\_tabs12 = 12

num\_tabs13 = 13

num\_tabs14 = 14

num\_tabs15 = 15

num\_tabs16 = 16

num\_tabs17 = 17

num\_tabs18 = 18

num\_tabs19 = 19

num\_tabs20 = 20

# BOTS INDIVIDUAIS - Serviços

Os prints ilustram os filtros exatamente no mesmo formato em que os bots trabalham.

## Service Pack

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**ServicePack** é o único relatório extraído com a extensão .txt e é extraído por filial e com filtro para período mensal. Não há iteração em loop; cada aba é aberta e baixada sequencialmente.

O nome do arquivo é manipulado para incluir o número da filial, baseado na ordem de extração.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O relatório **ServicePack** é baixado com o nome **"service"** concatenado com o número da filial e a data no formato mês.ano (por exemplo: **service09 – 05.2024**).

Os arquivos das 7 filiais são então movidos para a pasta **ServicePack.**

## Apuração Débito Interno

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Apuração Débito Interno** é filtrado diariamente, é baixado somente um relatório sem o filtro de filiais, pois o relatório traz todas as filiais em um só relatório e não afeta as rotinas do BI.

O relatório ApurDebitoInterno é baixado com o nome ApurDebitoInterno concatenado com a data com dia, mês e ano **(exemplo: ApurDebitoInterno30.04.2024)**

*O arquivo único é jogado para a pasta* ***ApurDebitoInterno***

## Garantia

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Garantia** é filtrada diariamente, resultando na obtenção de um único relatório sem o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como Garantia, seguido pela data no formato dia.mês.ano (por exemplo: **Garantia30.04.2024**), e então é movido para a pasta **Garantia**.

## Ordens de Serviço em Aberto

***Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente***

**Ordens de Serviço em Aberto** não possuem filtro de data, pois o relatório engloba todas as Ordens de Serviço em aberto no sistema no momento da geração. É baixado apenas um relatório sem o filtro de filiais, já que o relatório abrange todas as filiais em um único documento, sem interferir nas operações do BI.

O relatório é nomeado como **OSemAberto** e é então movido para a pasta **OSemAberto**.

## Ordens de Serviço Abertas em um Determinado Período

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente com confiança média

**Ordens de Serviço Abertas em um Determinado Período** são filtradas diariamente, resultando na obtenção de um único relatório sem o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **OSemAbertoO**, seguido pela data no formato dia.mês.ano (por exemplo: **OSemAbertoO30.04.2024**), e é então movido para a pasta **OSemAberto**.

## Ordens de Serviço Fechadas em um Determinado Período

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Ordens de Serviço Fechadas em um Determinado Período** são filtradas diariamente, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **OSFechadasO,** seguido pela data no formato dia.mês.ano (por exemplo: **OSFechadasO30.04.2024**), e é então movido para a pasta **OSFechadasO.**

## Ranking de Clientes

***Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente***

**Ranking Clientes** passa por um processo de filtragem diário, onde um loop é executado para manter os dados intactos, exceto pela variável da filial, que é alterada a cada iteração. Isso permite a geração de um relatório para cada filial.

O relatório é nomeado como **RankingClientes,** seguido pela filial e data no formato dia.mês.ano (por exemplo: **RankingClientes09 - 30.04.2024**), então os 7 relatórios para cada uma das filiais são então movidos para a pasta **RankingClientes.**

## Aproveitamento Tempo Mecânicos

***Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente***

**Aproveitamento Tempo Mecânicos** são filtradas mensalmente (primeiro e último dia do mês), resultando na geração de um único relatório que traz todas as filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **AproveitTempoMecanico,** seguido pela data no formato mês.ano (por exemplo: **AproveitTempoMecanico - 04.2024**), e é então movido para a pasta **AproveitTempoMecanico.**

## Checklist

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Checklist** passa por um processo de filtragem diário, onde um loop é executado para manter os dados intactos, exceto pela variável da filial, que é alterada a cada iteração. Isso permite a geração de um relatório para cada filial. Apenas um relatório é baixado, sem o filtro de filiais, já que o relatório engloba todas as filiais em um único documento, sem afetar as operações do BI.

O relatório é nomeado como **Checklist,** seguido pela filial e data no formato dia.mês.ano (por exemplo: **Checklist09 - 30.04.2024**), então os 7 relatórios para cada uma das filiais são então movidos para a pasta **Checklist.**

## Orçamento

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Orçamento** é filtrado diariamente, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **Orcamento,** seguido pela data no formato dia.mês.ano (por exemplo: **Orcamento30.04.2024**), e é então movido para a pasta **Orcamento.**

# BOTS INDIVIDUAIS – Estoque

## Itens Não Movimentados

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Itens Não Movimentados** é filtrado sempre com os últimos 180 dias, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **ItensNaoMovMaisXDias,** seguido pela data no formato dia.mês.ano (por exemplo: **ItensNaoMovMaisXDias30.04.2024**), e é então movido para a pasta **ItensNaoMovMaisXDias.**

## Vendas, Lucros e Comissões

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

**Vendas, Lucros e Comissões** é filtrado sempre com a data de ontem, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **VendaLucroComissoes,** seguido pela data no formato dia.mês.ano (por exemplo: **VendaLucroComissoes30.04.2024**), e é então movido para a pasta **VendaLucroComissoes.**

## Posição Geral Estoque

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Posição Geral Estoque** é filtrado sempre com a data de ontem, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **PosicaoGeralEstoque,** seguido pela data no formato dia.mês.ano (por exemplo: **PosicaoGeralEstoque30.04.2024**), e é então movido para a pasta **PosicaoGeralEstoque.**

## Relação de Pedidos Por Fornecedor

***Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente***

**Pedido Por Fornecedor** passa por um processo de filtragem mensal (primeiro e último dia do mês), é realizada a extração todos os dias, ou seja, o arquivo do mês sempre é substituído, nos filtros um loop é executado para manter os dados intactos, exceto a variável filial, que é alterada a cada iteração. Isso permite a geração de um relatório para cada filial.

O relatório é nomeado como **PedidoPorFornecedor,** seguido pela filial e data no formato mês.ano (por exemplo: **PedidoPorFornecedor09 - 04.2024**), então os 7 relatórios para cada uma das filiais são então movidos para a pasta **PedidoPorFornecedor.**

# BOTS INDIVIDUAIS – Faturamento

## Tempo Permanência

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Tempo Permanência** é filtrado sempre com a data de ontem, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **TempoVeiculoOficina,** seguido pela data no formato mês.ano (por exemplo: **TempoVeiculoOficina - 04.2024**), e é então movido para a pasta **TempoVeiculoOficina.**

## Origem Serviços Aplicados

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Origem Serviços Aplicados** é filtrado sempre com a data de ontem, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **OrigemServAplicados,** seguido pela data no formato mês.ano (por exemplo: **OrigemServAplicado - 04.2024**), e é então movido para a pasta **OrigemServAplicados..**

## Ordens de Serviço em Aberto - SINTÉTICO

***Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente***

**Ordens de Serviço em Aberto - Sintético** são filtradas diariamente, pois o relatório engloba todas as Ordens de Serviço em aberto no sistema no momento da geração. É baixado apenas um relatório sem o filtro de filiais, já que o relatório abrange todas as filiais em um único documento, sem interferir nas operações do BI.

O relatório é nomeado como **OSemAberto\_Sintetico** e é então movido para a pasta **OSemAberto\_Sintetico**.

## Ordens de Serviço Abertas em um Determinado Período - SINTÉTICO

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Ordens de Serviço Abertas em um Determinado Período - Sintético** são filtradas diariamente, resultando na obtenção de um único relatório sem o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **OSemAbertoO\_Sintetico**, seguido pela data no formato dia.mês.ano (por exemplo: **OSemAbertoO\_Sintetico30.04.2024**), e é então movido para a pasta **OSemAberto\_Sintetico**.

## Ordens de Serviço Fechadas em um Determinado Período - SINTÉTICO

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Ordens de Serviço Fechadas em um Determinado Período - Sintético** são filtradas diariamente, resultando na geração de um único relatório que mantem em vazio o filtro de filiais. Isso ocorre porque o relatório abrange todas as filiais em um único documento, sem impactar as operações do BI.

O relatório é nomeado como **OSFechadasO\_Sintetico,** seguido pela data no formato dia.mês.ano (por exemplo: **OSFechadasO\_Sintetico30.04.2024**), e é então movido para a pasta **OSFechadasO\_Sintetico.**