# **Documentação Técnica Comprovei Sac**

# Resumo

Este script em Python busca dados de uma API e os salva em um arquivo CSV. É um script de coleta de dados usado para baixar e processar dados do serviço Comprovei SAC. Ele recupera os dados a partir de uma data inicial até uma data atual especificada pelo usuário, os armazena temporariamente em um arquivo CSV e então realiza o tratamento desses dados.

# Dependências

As seguintes bibliotecas Python são necessárias para executar este script:

* requests
* pandas
* zipfile
* io
* re
* os
* xml.etree.ElementTree
* logging
* datetime
* argparse
* pathlib
* dotenv
* requests.auth

# Requisitos

Antes de executar o script, certifique-se de ter os seguintes requisitos atendidos:

* Python 3.x instalado no sistema.
* Bibliotecas Python.

Você pode instalar as bibliotecas usando o comando **pip install -r requirements.txt**

# Configurações

O script usa um arquivo ".env" para configurar variáveis de ambiente, tais como nome de usuário, senha, URL do serviço Comprovei SAC e diretórios onde os dados serão armazenados.  
Crie um arquivo **config.env** no diretório raiz do projeto e defina as seguintes variáveis de ambiente:

USERNAME=<seu\_nome\_de\_usuário>  
PASSWORD=<sua\_senha>  
DATADIR=<diretório\_de\_dados>  
CSV\_DATA\_DIR=<diretório\_de\_saida\_csv>  
CSV\_DATA\_DIR\_BI=<diretório\_de\_saida\_csv\_para\_BI>

* Substitua **<seu\_nome\_de\_usuário>** pelo seu nome de usuário no Comprovei SAC.
* Substitua **<sua\_senha>** pela sua senha no Comprovei SAC.
* Defina **<diretório\_de\_dados>** como o diretório onde os arquivos de dados serão armazenados.
* Defina **<diretório\_de\_saida\_csv>** como o diretório onde os arquivos CSV de saída serão armazenados.
* Defina **<diretório\_de\_saida\_csv\_para\_BI>** como o diretório onde os arquivos CSV de saída para Business Intelligence serão armazenados.

# Uso

1. Abra um terminal ou prompt de comando no diretório raiz do projeto.
2. Execute o script fornecendo as datas iniciais e atuais como argumentos:

python comprovei.py <data\_inicial> <data\_atual>

* + Substitua **<data\_inicial>** pela data inicial desejada no formato 'YYYY-MM-DD' ou use as palavras-chave 'hoje', 'ontem' ou 'tres' para obter as datas correspondentes automaticamente.
  + Substitua **<data\_atual>** pela data atual desejada no formato 'YYYY-MM-DD' ou use as palavras-chave 'hoje' para obter a data atual automaticamente.

1. Aguarde o processo ser concluído. O script realizará as seguintes etapas:
   * Autenticará na API do Comprovei SAC usando as credenciais fornecidas.
   * Solicitará os dados do Comprovei SAC com base nas datas especificadas.
   * Baixará um arquivo ZIP contendo os dados solicitados.
   * Extrairá os arquivos CSV do arquivo ZIP e os concatenará.
   * Processará e limpará os dados.
   * Exportará os dados processados para arquivos CSV nos diretórios especificados.
   * Gerará um arquivo de log chamado **Log.log** para registrar informações sobre a execução do script.
2. Verifique os arquivos CSV gerados nos diretórios definidos nas variáveis de ambiente **CSV\_DATA\_DIR** e **CSV\_DATA\_DIR\_BI**.

**Observações**

* Certifique-se de ter as credenciais corretas para autenticação na API do Comprovei SAC.
* O script está configurado para processar dados de até 10 dias anteriores à data atual. Se desejar alterar esse período, você pode modificar o código-fonte do script.

# Funções

**create\_login\_payload(data\_inicial, data\_atual)**

Esta função cria o payload necessário para a autenticação e solicitação dos dados. Recebe como parâmetros a data inicial e a data final para a coleta dos dados.

**autenticar\_e\_solicitar\_dados(data\_inicial, data\_atual)**

Esta função realiza a autenticação na API Comprovei SAC e solicita os dados. Se a resposta da API for bem sucedida, a função retorna a resposta. Caso contrário, um erro é registrado no log.

**processar\_csv()**

Esta função processa os arquivos CSV baixados, os lê e armazena em um DataFrame. Todos os arquivos são concatenados em um único DataFrame.

**drop\_duplicates(df\_concatenado)**

Esta função remove as duplicatas do DataFrame concatenado, usando a coluna 'Documento' e 'Chave' como critérios.

**save\_output(df\_concatenado)**

Esta função salva o DataFrame processado em um arquivo CSV.

**clean\_directory(directory, keep\_file)**

Esta função limpa o diretório especificado, excluindo todos os arquivos, exceto o arquivo mantido.

# Exceções

O script trata algumas exceções. Por exemplo, ao autenticar e solicitar dados, se a URL não for encontrada, o script registra um erro no log. Além disso, ao baixar o arquivo ZIP, se a URL do arquivo ZIP não for encontrada, uma exceção é levantada.

# Interação com a API e processamento de dados

O script realiza uma requisição POST para a API, usando a autenticação básica HTTP e um payload criado pela função **create\_login\_payload**.

Se a requisição for bem-sucedida, o script utiliza uma expressão regular para encontrar a URL do arquivo ZIP no texto da resposta. O arquivo ZIP é então baixado e extraído para um diretório específico.

Os arquivos CSV extraídos são processados: o script lê cada um dos arquivos, armazena-os em DataFrames e, em seguida, concatena todos os DataFrames em um único DataFrame.

O script então remove as duplicatas desse DataFrame e o salva em um arquivo CSV. Após isso, todos os arquivos do diretório, exceto o arquivo CSV final, são excluídos.

# FAQ

1. **Pergunta**: O que faz o script em Python mencionado no documento?   
   **Resposta**: Este script em Python busca dados de uma API e os salva em um arquivo CSV. Ele é usado para baixar e processar dados do serviço Comprovei SAC. Ele recupera os dados a partir de uma data inicial até uma data atual especificada pelo usuário, os armazena temporariamente em um arquivo CSV e então realiza o tratamento desses dados. (Página 1)
2. **Pergunta**: Quais bibliotecas Python são necessárias para executar este script? **Resposta**: As bibliotecas Python necessárias para executar este script incluem requests, pandas, zipfile, io, re, os, xml.etree.ElementTree, logging, datetime, argparse, pathlib, dotenv e requests.auth. (Página 1)
3. **Pergunta**: Quais são os requisitos para executar o script?   
   **Resposta**: Antes de executar o script, certifique-se de ter Python 3.x instalado no sistema e todas as bibliotecas Python necessárias. Você pode instalar as bibliotecas usando o comando pip install -r requirements.txt. (Página 1)
4. **Pergunta**: Como o script é configurado?   
   **Resposta**: O script usa um arquivo ".env" para configurar variáveis de ambiente, tais como nome de usuário, senha, URL do serviço Comprovei SAC e diretórios onde os dados serão armazenados. (Página 1)
5. **Pergunta**: Quais funções o script contém?  
   **Resposta**: O script contém várias funções, incluindo create\_login\_payload, autenticar\_e\_solicitar\_dados, processar\_csv, drop\_duplicates, save\_output e clean\_directory. Cada uma dessas funções desempenha um papel específico no processo de coleta, processamento e armazenamento de dados. (Página 3)
6. **Pergunta**: Como o script interage com a API e processa os dados?   
   **Resposta**: O script realiza uma requisição POST para a API, usando a autenticação básica HTTP e um payload criado pela função create\_login\_payload. Se a requisição for bem-sucedida, o script utiliza uma expressão regular para encontrar a URL do arquivo ZIP no texto da resposta. O arquivo ZIP é então baixado e extraído para um diretório específico. Os arquivos CSV extraídos são processados: o script lê cada um dos arquivos, armazena-os em DataFrames e, em seguida, concatena todos os DataFrames em um único DataFrame. O script então remove as duplicatas desse DataFrame e o salva em um arquivo CSV. Após isso, todos os arquivos do diretório, exceto o arquivo CSV final, são excluídos. (Página 3)
7. **Pergunta**: Como usar o script?   
   **Resposta**: Para usar o script, abra um terminal ou prompt de comando no diretório raiz do projeto. Execute o script fornecendo as datas iniciais e atuais como argumentos. Aguarde o processo ser concluído. O script realizará várias etapas, incluindo autenticação na API do Comprovei SAC, solicitação dos dados, download de um arquivo ZIP contendo os dados solicitados, extração dos arquivos CSV do arquivo ZIP e concatenação deles, processamento e limpeza dos dados, exportação dos dados processados para arquivos CSV nos diretórios especificados e geração de um arquivo de log para registrar informações sobre a execução do script. (Página 2)
8. **Pergunta**: Quais são as observações importantes sobre o uso do script?  
   **Resposta**: Certifique-se de ter as credenciais corretas para autenticação na API do Comprovei SAC. O script está configurado para processar dados de até 10 dias anteriores à data atual. Se desejar alterar esse período, você pode modificar o código-fonte do script. (Página 2)
9. **Pergunta**: Como configurar as variáveis de ambiente no arquivo .env?   
   **Resposta**: Substitua <seu\_nome\_de\_usuário> pelo seu nome de usuário no Comprovei SAC, <sua\_senha> pela sua senha no Comprovei SAC, <diretório\_de\_dados> como o diretório onde os arquivos de dados serão armazenados, <diretório\_de\_saida\_csv> como o diretório onde os arquivos CSV de saída serão armazenados e <diretório\_de\_saida\_csv\_para\_BI> como o diretório onde os arquivos CSV de saída para Business Intelligence serão armazenados. (Página 2)
10. **Pergunta**: O que acontece se a URL não for encontrada durante a autenticação e solicitação de dados?   
    **Resposta**: Se a URL não for encontrada durante a autenticação e solicitação de dados, o script registra um erro no log. Além disso, ao baixar o arquivo ZIP, se a URL do arquivo ZIP não for encontrada, uma exceção é levantada. (Página 3)