**Documentação do Código - Processamento de Arquivos Uber e Integração com Notion**

**Situação problema**

A empresa utiliza Uber Empresas para gerenciar as viagens dos funcionários, mas dependia de que os próprios colaboradores justificassem os gastos, informando o motivo e o centro de custo da despesa. Frequentemente, os funcionários não justificavam as despesas até o fechamento do cartão, o que atrasava o processo de conciliação bancária e criava distorções nos relatórios financeiros da empresa.

**Objetivo**

Este código acessa os servidores SFTP da Uber empresas e a API Do Notion para:

* Obter novos relatórios .CSV de corridas realizadas na Uber empresas
* Criar novas entradas no Notion para as despesas que precisam ser identificadas

**Como funciona**

1. O código acessa o servidor SFTP da Uber para sincronizar os relatórios de viagem diários localmente, no formato CSV.
2. Para cada corrida realizada, é criada uma página no Notion com base no UUID (código de identificação) da corrida.
3. O UUID é usado para evitar a criação de páginas duplicadas no Notion.
4. O usuário é notificado para justificar a despesa e anexar os documentos necessários.

**Requisitos obrigatórios para funcionamento do código**

* Configurações no SFTP: O endereço de IP da máquina do usuário deve estar cadastrado na integração SFTP da Uber Empresas (Configurações > Integrações).
* **Ferramentas instaladas:**
  + Python 3.x
  + Bibliotecas Python: requests, csv, json, subprocess, datetime.
  + Cyberduck CLI (“Duck”) configurado corretamente.
  + Arquivo PEM para autenticação no SFTP.
* **Arquivos de configuração:**
  + notion\_credenciais.json: Contém as credenciais da API do Notion.
  + notion\_users.json: Mapeamento entre e-mails dos usuários e seus respectivos IDs no Notion.
* **Estrutura do CSV: O arquivo deve conter pelo menos as seguintes colunas:**
  + UUID da viagem
  + Data da solicitação
  + Nome e sobrenome
  + E-mail
  + Valor da transação
  + Moeda
  + Serviço.

**Guia de Setup dos Arquivos JSON e Configurações**

**1. Arquivo notion\_credenciais.json**

Este arquivo contém as credenciais necessárias para se conectar à API do Notion. Aqui está um exemplo de como ele deve ser estruturado:

{

"Uber\_Business\_Key": "sua\_chave\_api\_notion\_aqui",

"Registro\_de\_Despesas\_ID": "id\_do\_banco\_de\_dados\_notion\_aqui"

}

**Como obter os valores:**

* **Uber\_Business\_Key**:
  1. Acesse o [Notion](https://www.notion.so/" \t "_blank).
  2. Vá em **Settings & Members** > **Integrations** > **Develop your own integrations**.
  3. Crie uma nova integração e copie o **Internal Integration Token**.
  4. Cole o token no campo Uber\_Business\_Key.
* **Registro\_de\_Despesas\_ID**:
  1. Abra o banco de dados do Notion que você deseja usar.
  2. Na URL do banco de dados, copie o ID (é a parte após https://www.notion.so/ e antes do ?).
  3. Cole o ID no campo Registro\_de\_Despesas\_ID.

**2. Arquivo notion\_users.json**

Este arquivo mapeia os e-mails dos usuários aos seus respectivos IDs no Notion. Aqui está um exemplo:

{

"usuario1@empresa.com": "id\_usuario1\_notion",

"usuario2@empresa.com": "id\_usuario2\_notion"

}

**Como obter os valores:**

* **E-mails**: Use os e-mails dos usuários que realizam as viagens.
* **IDs dos Usuários no Notion**:
  1. Acesse o [Notion](https://www.notion.so/" \t "_blank).
  2. Vá em **Settings & Members** > **Members**.
  3. Para cada usuário, clique no nome e copie o ID da URL (é a parte após https://www.notion.so/ e antes do ?).
  4. Associe o ID ao e-mail correspondente no arquivo JSON.

**3. Arquivo list\_files.bat**

Este arquivo é um script batch usado para listar os arquivos no servidor SFTP da Uber. Aqui está um exemplo de como ele pode ser configurado:

@echo off  
echo ls /from\_uber/trips | sftp -i "C:\Users\keys\private.pem" -oPort=2222 [seu\_usuário@sftp.uber.com](mailto:seu_usuário@sftp.uber.com)

**Como configurar:**

* **Caminho da chave PEM**: Substitua C:\Users\keys\private.pem pelo caminho correto da sua chave privada PEM.
* **Usuário e endereço SFTP**: Substitua usuario@sftp.uber.com:2222 pelo usuário e endereço do SFTP fornecido pela Uber.

**4. Configuração do Duck CLI**

O Duck CLI é usado para sincronizar os arquivos do SFTP com o diretório local. Aqui está como configurá-lo:

**Instalação do Duck CLI:**

1. Baixe e instale o [Cyberduck](https://cyberduck.io/" \t "_blank).
2. Instale o Duck CLI seguindo as instruções no site oficial.

**Configuração do Comando:**

No código, o comando Duck está definido como:

duck --username “seu\_username” --synchronize "sftp://sftp.uber.com:2222/from\_uber/trips/" "C:\\Users \\keys\\UberTrips" --identity "C:\\Users \\keys\\private.pem"

* **seu\_username**: Substitua pelo seu nome de usuário no SFTP da Uber.
* **sftp://sftp.uber.com:2222/from\_uber/trips/**: Este é o caminho remoto no SFTP da Uber.
* **C:\\Users\\ keys\\UberTrips**: Este é o diretório local onde os arquivos serão baixados.
* **--identity**: Substitua pelo caminho da sua chave privada PEM.

**5. Estrutura do Diretório Local**

O script espera que os arquivos CSV estejam em um diretório específico. Aqui está um exemplo de como organizar os arquivos:

C:\Users \keys\

├── private.pem

├── list\_files.bat

├── notion\_credenciais.json

├── notion\_users.json

└── UberTrips\

├── daily\_trips-2023\_10\_01.csv

├── daily\_trips-2023\_10\_02.csv

└── ...

**Passo a passo do funcionamento**

**Descrição Geral**

**Este script automatiza o processo de:**

1. Sincronização de arquivos CSV de corridas da Uber via SFTP utilizando o Duck CLI.
2. Processamento de arquivos CSV contendo dados de corridas.
3. Criação de entradas na API do Notion com base nos dados processados.

**Principais Funcionalidades**

1. **Sincronização de Arquivos via Duck CLI:**
   * Conecta ao servidor SFTP da Uber.
   * Sincroniza arquivos do servidor com um diretório local.
2. **Leitura e Processamento de Arquivos CSV:**
   * Lê arquivos gerados nos últimos 10 dias.
   * Ignora linhas de cabeçalho e processa dados transacionais.
3. **Criação de Entradas no Notion:**
   * Gera entradas na base de dados do Notion com informações extraídas dos arquivos CSV.
   * Verifica duplicidade utilizando UUIDs existentes na base de dados.

**Dependências**

* **Python Bibliotecas:**
  + requests
  + os
  + json
  + csv
  + subprocess
  + msvcrt
  + datetime
* **Ferramentas Externas:**
  + Duck CLI: Para sincronização de arquivos via SFTP.

**Configuração**

1. **Arquivos Necessários:**
   * notion\_credenciais.json: Contém chaves de autenticação da API do Notion e IDs da base de dados.
   * notion\_users.json: Mapeia e-mails para IDs de usuários no Notion.
   * list\_files.bat: Script Batch para listar arquivos no SFTP.
2. **Caminhos e Comandos Importantes:**
   * sftp\_batch\_file: Caminho para o script Batch.
   * duck\_command: Comando utilizado pelo Duck CLI para sincronização.
3. **Estrutura de Arquivos CSV:**
   * O script espera que o CSV tenha pelo menos 45 colunas e que os dados transacionais comecem na linha 7.

**Funções do Código**

1. **retrieve\_all\_notion\_uuids():**
   * Recupera todos os UUIDs existentes no banco de dados do Notion.
   * Utiliza paginação para buscar até 100 entradas por vez.
2. **create\_notion\_entry(row, existing\_entries):**
   * Cria uma entrada no Notion para um UUID específico.
   * Verifica duplicidade com base nos UUIDs existentes.
3. **process\_csv\_files(directory):**
   * Processa arquivos CSV de um diretório especificado.
   * Filtra arquivos pelos últimos 10 dias.
   * Lê e converte os dados do CSV para dicionários.
   * Ignora linhas inválidas ou arquivos fora do formato esperado.
4. **main():**
   * Define o diretório de arquivos CSV e chama a função para processar os arquivos.

**Fluxo do Script**

1. **Sincronização de Arquivos:**
   * Executa o script Batch para listar arquivos no servidor.
   * Executa o comando do Duck CLI para sincronização.
2. **Processamento de Dados:**
   * Filtra arquivos CSV recentes.
   * Lê e processa os dados do CSV.
   * Cria entradas no Notion para transações únicas.
3. **Criação no Notion:**
   * Formata dados para conformidade com a API do Notion.
   * Envia requisições para criar entradas.

**Erros e Exceções**

* Erro ao buscar UUIDs no Notion: Tratado com requests.exceptions.RequestException.
* Falhas ao criar entradas: Erros logados e ignorados.
* Formato inválido de data ou arquivo CSV: Linhas/arquivos ignorados.

**Observações**

* **Sincronização Segura:**
  + A autenticação via SFTP utiliza uma chave privada PEM.
* **Intervalo de Arquivos:**
  + Apenas arquivos gerados nos últimos 10 dias são processados.