**Documentação e detalhamento de**

**Software de extração de dados**

Versão 1.0

# **INTRODUÇÃO**

O objetivo deste documento é demonstrar o processo de extração, transformação e carregamento dos dados através de um script utilizando linguagem estruturada de programação. Neste processo, os dados são extraídos através de uma API de dados públicos, estruturados e armazenados num banco de dados local. Após a extração, é utilizado um software de Business Inteligente para gerar informações sobre os dados.

# **DESCRIÇÃO GERAL DO PROJETO**

O objetivo central é extrair os dados da API pública da **receita federal**, estruturar todas as colunas de dados com Python e inserir as informações via conexão com o SQL SERVER, após isso, é feito o processo de transformação dos dados em informação utilizando o Microsoft Power BI Desktop.

Para executar o processo de extração, foi necessário criar funções de limpeza, estruturação e requisição dos dados.

Os dados serão puxados através de um laço de repetição **FOR** contendo uma condição que faz o loop congelar a cada 60 segundos, respeitando o tempo da **API** para uma nova requisição e inseridos no SQL Server.

Após o fim do loop, os dados são lidos no Power BI, é feito as relações de cardinalidade, tratamento dos tipos de dados e transformados em informação num dashboard totalmente interativo.

# **CODIFICAÇÃO**

## **1.1 FERRAMENTAS DE DESENVOLVIMENTO**

* Python v3.8
* Biblioteca pandas, pyodbc, time e requests
* SQL Server Management Studio 18
* Microsoft Power BI Desktop
* Microsoft Excel

## **1.2 CONEXÃO COM O BANCO DE DADOS**

A primeira parte do desenvolvimento consiste na conexão do Python com o SQL Server utilizando a biblioteca pyodbc.

Utilizando uma tupla, declarei as variáveis DRIVER, que se refere ao banco de dados, SERVER, referindo ao banco de dados local, DATABASE, que irá receber os dados e uma opção chamada Trusted\_Connection, que indica a não necessidade de senha para conexão, já que o servidor é local.

Texto

Descrição gerada automaticamente

## **1.3 A API**

Para extrair os dados, utilizei a API pública da receita federal. Seu funcionamento necessita de um dado de entrada, o CNPJ.

Exemplo de entrada:



Exemplo de Saída:

Texto

Descrição gerada automaticamente

A consulta retorna 27 colunas com diversas informações da empresa no modelo json, (JavaScript Object Notation) baseado em chave-valor.

Para puxar as informações, é utilizado o método GET do protocolo HTTP através da biblioteca requests do Python.

Esta API permite 3 consultas por minuto. No caso do limite ser excedido, o código HTTP retornado é o 429.

## **1.4 DADOS DE ENTRADA**

Os dados de entrada serão os CNPJ’s de todas as empresas listadas na B3 contidos num arquivo Excel.

Tabela

Descrição gerada automaticamente

Este arquivo é lido no Python através da biblioteca pandas.



## **1.5 SCRIPT DE REQUISIÇÃO**

O script que puxa os dados, previamente chama o script de limpeza que, ao retornar TRUE, executa o método **requests.get** recebendo como parâmetro o CNPJ.

Esta requisição necessita da biblioteca **requests** do Python.

Exemplo:

**Texto

Descrição gerada automaticamente**

A saída é um **Dicionário** já mostrado na apresentação da **API.**

## **1.6 SCRIPT DE LIMPEZA DE CNPJ**

Antes de fazer as requisições, programei uma função para limpar os dados de CNPJ e fazer uma verificação para evitar possíveis CNPJ’s errados.

Texto

Descrição gerada automaticamente

Utilizei a função replace para remover todos os caracteres inválidos e um list-comprehension para armazenar qualquer informação que não fosse numérica para mostrar ao usuário caso haja.

Após isso, verifico se a quantidade de números está de acordo com o padrão de CNPJ, se estiver, retorno o CNPJ limpo, se não, retorno os possíveis erros para o usuário.

Exemplo:

Texto

Descrição gerada automaticamente

Saídas:

Texto

Descrição gerada automaticamente

## **1.7 SCRIPT PARA TRATAR DADOS COM MULTIPLAS CHAVES**

Algumas partes do Dicionário retornam chaves com múltiplos valores e retornam uma lista ao invés de somente um valor. Para tratar essas informações, fiz uma função especifica para estas chaves e às transformei num Dataframe pandas.

Exemplo de chave-valor:

**"nome":"PETROLEO BRASILEIRO S A PETROBRAS"**

sendo o "nome" a chave e o valor “PETROLEO BRASILEIRO S A PETROBRAS"

Exemplo de chave com múltiplos valores:

**{"atividade\_principal":[{"text":"Comércio atacadista de álcool carburante, biodiesel, gasolina e demais derivados de petróleo, exceto lubrificantes, não realizado por transportador retalhista (T.R.R.)","code":"46.81-8-01"}]**

Para resolver esse problema, escrevi o código abaixo que será executado em duas chaves com múltiplos valores:

A chave qsa contém o nome de todos os principais sócios e cargos da empresa

Texto

Descrição gerada automaticamente

A **atividade** contém os múltiplos serviços prestados pela empresa

Interface gráfica do usuário

Descrição gerada automaticamente com confiança média

## **1.8 SCRIPT PRINCIPAL**

O código principal funcionará num laço FOR que executará em um RANGE de acordo com o número de linhas que o arquivo Excel terá.



No caso, o laço executará os códigos abaixo 254 vezes.

A variável response receberá o retorno do Dicionário, conforme explicado no script de requisição.

Cada empresa possui um Ativo, o segundo código cria uma nova chave e recebe o ativo referente a empresa que está sendo executada no loop.

Exemplo:

response[‘código’] == ‘PETR4’

Texto

Descrição gerada automaticamente

Logo em seguida é feita uma verificação na quantidade de chaves do dicionário. Caso haja menos que 26, significa que o CNPJ não possui informações na base de dados da API. O CNPJ que estiver inválido é inserido na lista cnpjInvalido.

Conforme citado no tópico A API, só é possível fazer 3 requisições a cada 60 segundos. Então desenvolvi uma lógica que faz o laço de repetição aguardar 60 segundos a cada 3 requisições usando a seguinte lógica:

Primeiro criei uma lista com todos os múltiplos de 3 até 254 (que é o total de linhas do Excel mostrado em DADOS DE ENTRADA)





E criei uma condição que verificará se o número atual do loop principal é um múltiplo de 3.

Através do código

**Texto

Descrição gerada automaticamente com confiança baixa**

Se a variável c for um múltiplo de 3, a condição abaixo será verdadeira e o laço congelará por 60 segundos, respeitando o tempo da API para uma nova requisição.

Sem esta condição, ocorreria o seguinte erro:

Texto

Descrição gerada automaticamente

Se a condição for falsa, o loop entrará no código principal.

Primeiro, o laço vai inserir no SQL SERVER as informações dos principais sócios da empresa que estará no loop e seu ramo de atividades. Em cada loop, será criado uma chave de ID para cada tabela para fazer relacionamentos posteriormente.

Logo em seguida, todas as 26 colunas de informação do Dicionário serão inseridas.

Texto

Descrição gerada automaticamente

Após a inserção dos dados no Banco de dados, o código segue a mesma lógica dos 60 segundos explicado acima:

Texto

Descrição gerada automaticamente com confiança média

Exemplo do código em Execução:

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

Resultado no SQL Server:Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Após todos os loops, o código retornará todos os CNPJ’s que forem inválidos

# **TRANSFORMANDO DADOS NO MICROSOFT POWER BI**

## **1.1 Limpeza dos dados**

Após a conexão com o banco de dados, os dados passam por uma fase de transformação e verificação de possíveis falhas de estruturação.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

A API disponibilizou os dados da coluna capital\_social com a pontuação monetário no modelo americano (usando ponto no lugar de vírgula) e a coluna está com o tipo VARCHAR no banco de dados.

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente com confiança média

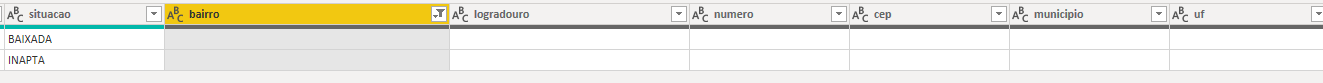
Para resolver isso, substituí o ponto por vírgula e em seguida transformei o tipo da coluna em número decimal fixo, assim transformando a coluna em valor monetário.

Interface gráfica do usuário, Tabela

Descrição gerada automaticamente

Uma imagem contendo Interface gráfica do usuário

Descrição gerada automaticamente

Duas empresas estavam INAPTAS e sem informações, então foram removidas manualmente. 

## **1.2 RELAÇÕES DE CARDINALIDADE ENTRE TABELAS**

Conforme citado no **SCRIPT PRINCIPAL**, para cada loop foi criado um **ID** para cada uma das informações inseridas no banco de dados.

A tabela de **pessoas** contém os nomes dos principais sócios das empresas, representando uma cardinalidade de **MUITOS SÓCIOS PARA UMA EMPRESA.** Já a tabela de **atividades** contém **atividade\_principal** e **atividade\_secundaria**, mas estas informações estão contidas na mesma linha, apenas em colunas diferentes, portando, a cardinalidade é de **UM PARA UM**.

Tabela pessoas:

Tabela

Descrição gerada automaticamente

Tabela atividades:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

## **1.3 CRIANDO O DASHBOARD**

As informações contidas no dashboard são apenas para demonstrar a criação da informação, pois todo o dashboard tem seu propósito, e no momento, quero apenas ilustrar o resultado.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

O Dashboard possui filtros múltiplos e permite selecionar UF, MUNICÍPIO e BAIRRO. Tudo de forma interativa

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamenteA Matriz de dados possui uma barra de rolagem que pode retornar a informação que for necessária para o usuário.

# **FINALIZAÇÃO DO PROJETO**

## **1.1 RESULTADO DO PROJETO**

Programa Python de extração de dados:

Banco de Dados Relacional com três tabelas de dados com informações de 249 empresas extraídas da API pública da receita federal.

Dashboard automatizado de informações sobre as empresas.

## **1.2 INÍCIO E FIM**

Início: 25/05

Finalização: 03/06

## **1.3 DESENVOLVEDORES**

Todas as funcionalidades foram desenvolvidas por Gustavo do Carmo Lima.

https://www.linkedin.com/in/gustavo-do-carmo-lima-568629163/