Column Filter

Programa auxiliar ao OnProject: Automação de Lista de Peças

Conteúdo – Guia de demonstração da solução

1. MIT

A solução apresentada consiste num script desenvolvido pela **TeamOn** para colmatar necessidades específicas da **ESI**, pelo que este script é propriedade da **ESI** e não poderá ser replicado, vendido, doado ou partilhado pelas partes envolvidas no seu desenvolvimento sem o consentimento da **ESI**.

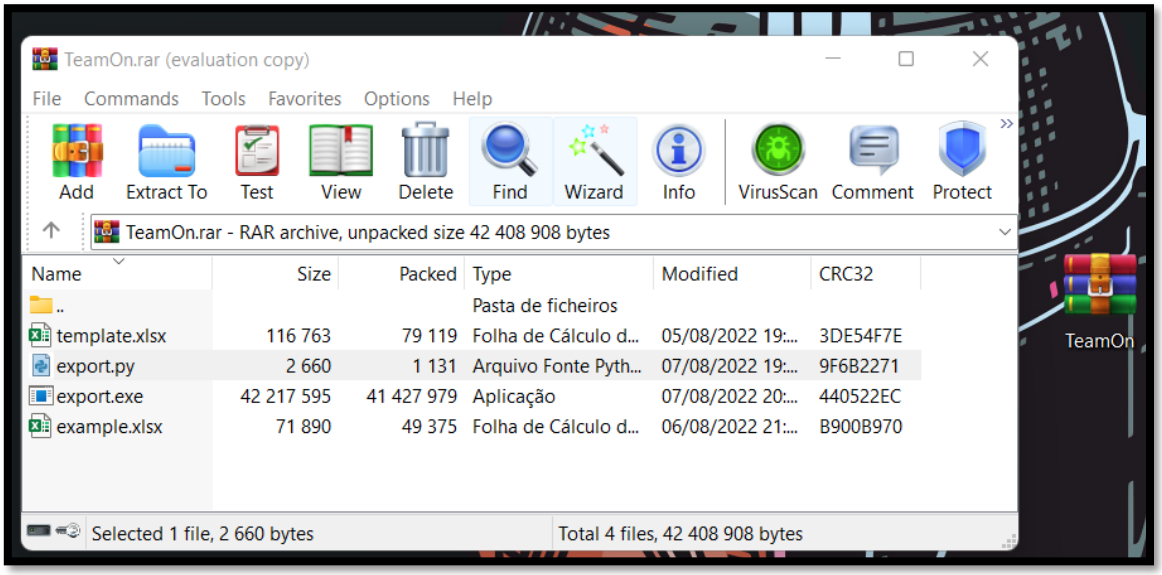
2. Introdução ao *Script* desenvolvido

Este *script* proposto foi desenvolvido em *Python* com o auxílio de três *frameworks* *Pandas*, *Openpyxl* e *Tkinter* por forma a facilitar a manipulação dos dados presentes num ficheiro Excel com tipo .**xlsx** e possibilitar oferecer ao utilizador uma pequena interface. Todo este *script* foi transformado num executável próprio para Windows, tendo como principal objetivo a extração de alguns campos do Excel adquirido através da rule **export_excel**, mais concretamente os campos de *Part Number*, *QTY*, *Description*, *Material*, *Company* e *Category*. Após a aquisição destes campos, todos os dados devem ser colocados num novo Excel com um *template* específico da empresa, estando todos os dados organizados por ordem decrescente segundo os parâmetros de *Category* e *Description*.

Neste manual será demonstrado detalhadamente como instalar e executar este programa.

3. Material Fornecido

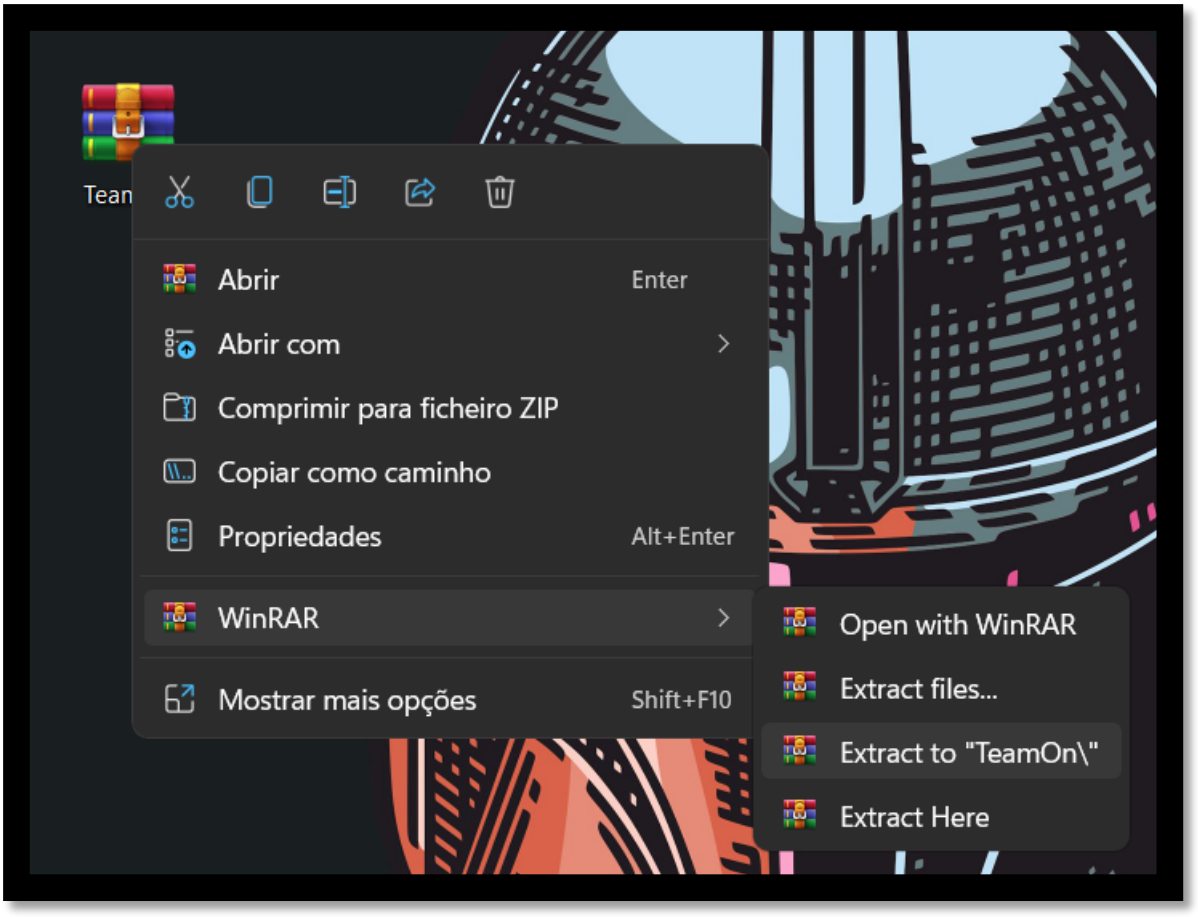
- Executável **.exe** com o *script*
- Ficheiro **.py** do *script*
- Ficheiros **.xlsx** exemplo, contendo entre os mesmo o Excel **Template**, que contém a formatação (cores, tipos de letra, tamanho de letra,…)
- Manual de instalação e execução do programa



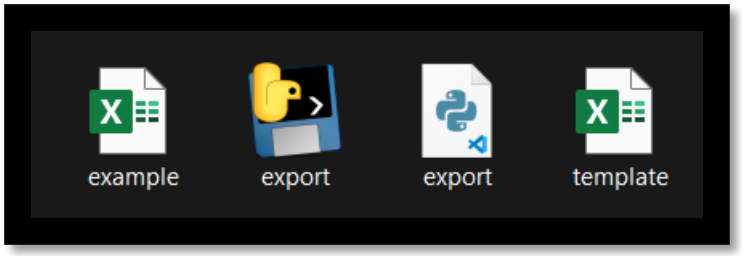
4. Instalação

O único passo a ser realizado é:

- Fazer a extração de todo o material para uma pasta, com nome à escolha, ou para o ambiente de trabalho.



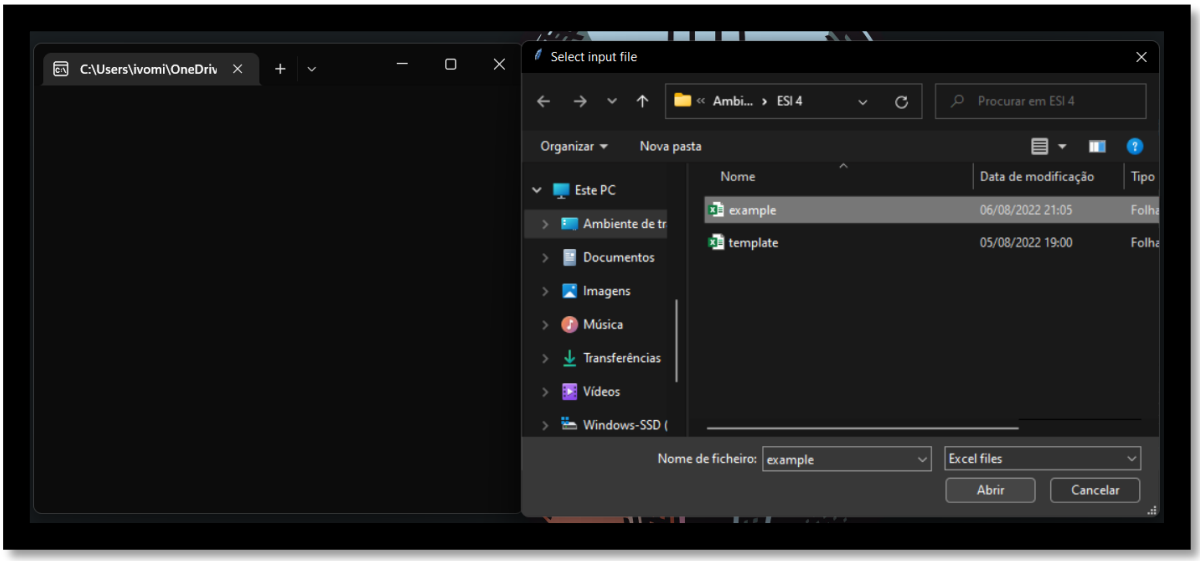
O aspeto dos ficheiros será deste tipo:



5. Execução

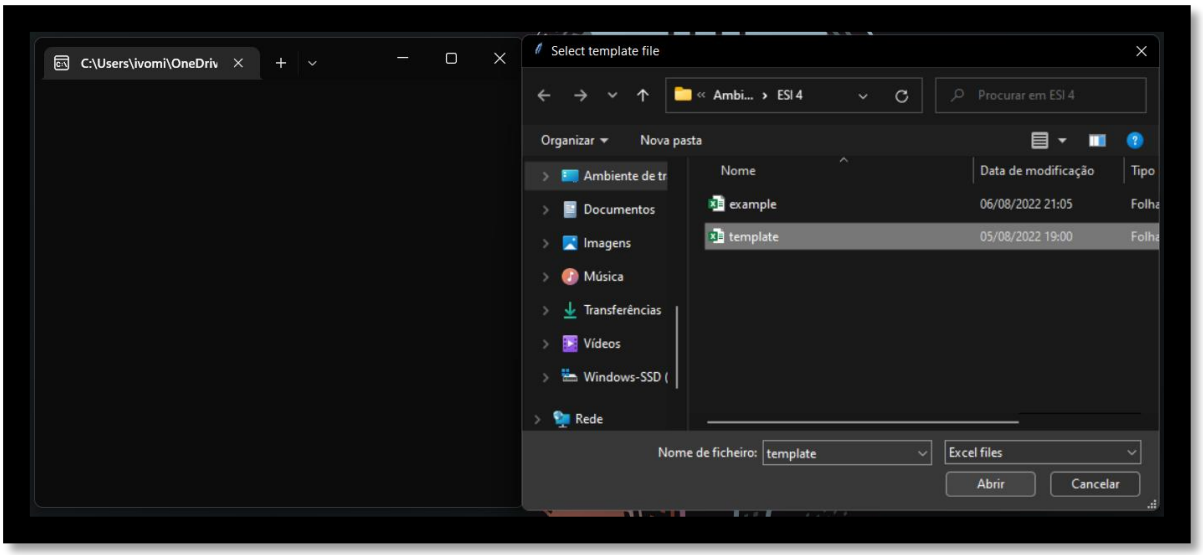
O ficheiro *export* já contém todas as bibliotecas e *utilities* necessários para o seu funcionamento. A única coisa a fazer, tal como qualquer executável, é fazer um *double click* sobre o mesmo.

Nota: o mesmo demora algum tempo a inicializar, cerca de 10 segundos pois tem de carregar todos os packages que utiliza, que embora pareçam poucos são muito poderosos e, portanto, pesados.

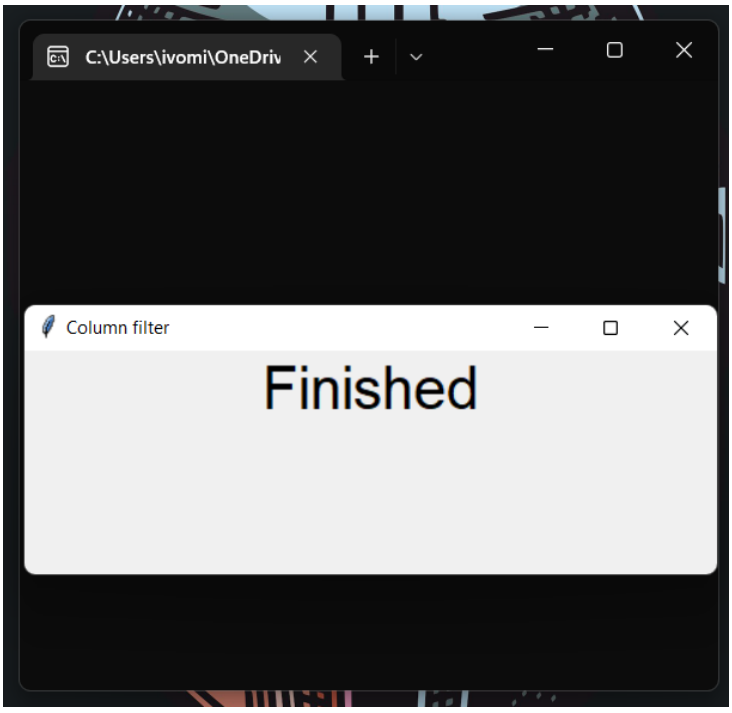


Podemos verificar que são abertos um terminal, que irá notificar o utilizador caso algum erro aconteça e uma nova aba, na localização onde esta o executável, pedindo ao *user* a seleção de um ficheiro Excel **.xlsx** do qual se quer extrair informação.

Selecionado o ficheiro em questão, neste caso o **example.xlxs**, deve ser depois premido o botão “Abrir".

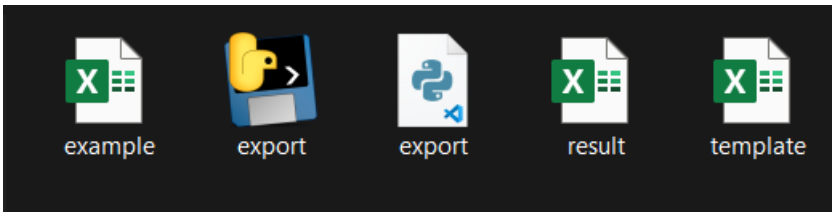


Após essa seleção a janela anterior é substituída por uma nova janela na qual deve ser selecionado o *template* para o qual queremos migrar os dados, tendo este *template* de ser também um Excel com a extensão de **.xlsx**.



Uma vez que tudo correu bem e não se deu nenhum tipo de erro pois ambos os ficheiros seguiam a estrutura e tinham o formato correto o utilizador é notificado com uma mensagem de *Finished* que indica que o ficheiro resultado foi criado com sucesso.

Podemos então fechar tanto a mensagem como o terminal e verificar que esse ficheiro resultado tem o nome de *result* e encontra-se na mesma localização que o executável.



6. Cuidados a ter em conta

- O *script* apenas deve ser executado sobre o sistema operativo Windows.
- O programa apenas é executável se o ficheiro de Excel tiver o tipo mais recente que é Livro do Excel (**.xlsx**) e não Livro do Excel 97-2003 (**.xls**).
- Todas as colunas de *Part Number*, *QTY*, *Description*, *Material*, *Company* e *Category* devem estar presentes no Excel do qual pretendemos extrair informação.
- O ficheiro de *Template* deve ter somente uma *Sheet* denominada como Projeto, contendo as colunas sobre a seguinte ordem Código, QT, Descrição, Material, Marca, Operação e Data. Devendo ainda ter linhas suficientes para poder acolher um projeto com uma grande quantidade de peças e partes.
- Apenas se pode selecionar um ficheiro de entrada para cada input, primeiramente deve ser selecionado o Excel do qual queremos extrair a informação e depois o *Template*.

7. Apêndice

7.1. Conteúdo do *Script*

```
#Interface
from tkinter import *
import tkinter.font as font
from tkinter.filedialog import askopenfilename

#Excel
import pandas as pd
import openpyxl
from openpyxl.utils.dataframe import dataframe_to_rows

def file_is_not_valid(dictionary, key):
    if key not in dictionary:
        return True
    return False
    ...

def get_excel_filename(dictionary, key, title='Select a File'):
    while file_is_not_valid(dictionary, key):
        dictionary[key] = askopenfilename(initialdir='.',
                                          title=title,
                                          filetypes=[("Excel files", ".xlsx")])
    return dictionary

def execute_program(files):
    # Variables for source file, worksheets, and empty dictionary for dataframes
    spreadsheet_file = pd.ExcelFile(files['input_filename'])
    worksheets = spreadsheet_file.sheet_names
    # Template File
    wb = openpyxl.load_workbook(files['template_filename'])
    ws = wb['Projeto']
    # Skip 2 rows and start writing from row 3 - first two are headers in template file
    rownumber = 2

    for sheet_name in worksheets:
        df = pd.read_excel(spreadsheet_file, sheet_name)
        # Getting only the columns asked: "Part Number","QTY","Description","Material",
        # "Company","Category"
        df = df[["Part Number", "QTY", "Description", "Material", "Company", "Category"]]
        # Organizing info:
        # 1º By Category
        # 2º By Description
```

```
      df = df.sort_values(['Category', 'Description'],
                          ascending=[False, False])
      appended_data = df.to_dict()
      # Read all rows from df, but don't read index or header
      rows = dataframe_to_rows(df, index=False, header=False)
      for r_idx, row in enumerate(rows, 1):
         for c_idx, value in enumerate(row, 1):
            # Write to cell, but after rownumber + row index
            ws.cell(row=r_idx + rownumber, column=c_idx, value=value)
       # Move the rownumber to end, so next worksheet data comes after this sheet's
data
      rownumber += len(df)
   wb.save('result.xlsx')

windows = Tk()
windows.withdraw()
windows.title("Column filter")
windows.geometry("500x150")

files = {}
get_excel_filename(files, 'input_filename', title='Select input file')
get_excel_filename(files, 'template_filename', title='Select template file')

windows.deiconify()
processing = Label(windows, text="Processing...", font=font.Font(size=30))
finished = Label(windows, text="Finished", font=font.Font(size=30))

processing.pack(fill='both')
execute_program(files)
processing.pack_forget()
finished.pack()

windows.mainloop()
```