**Hackathon 2024 – Equipe Conexão**

Hackathon 2024 da Escola de Engenharia (EE) da Universidade Presbiteriana Mackenzie (UPM)

**Equipe Conexão**

Integrantes:

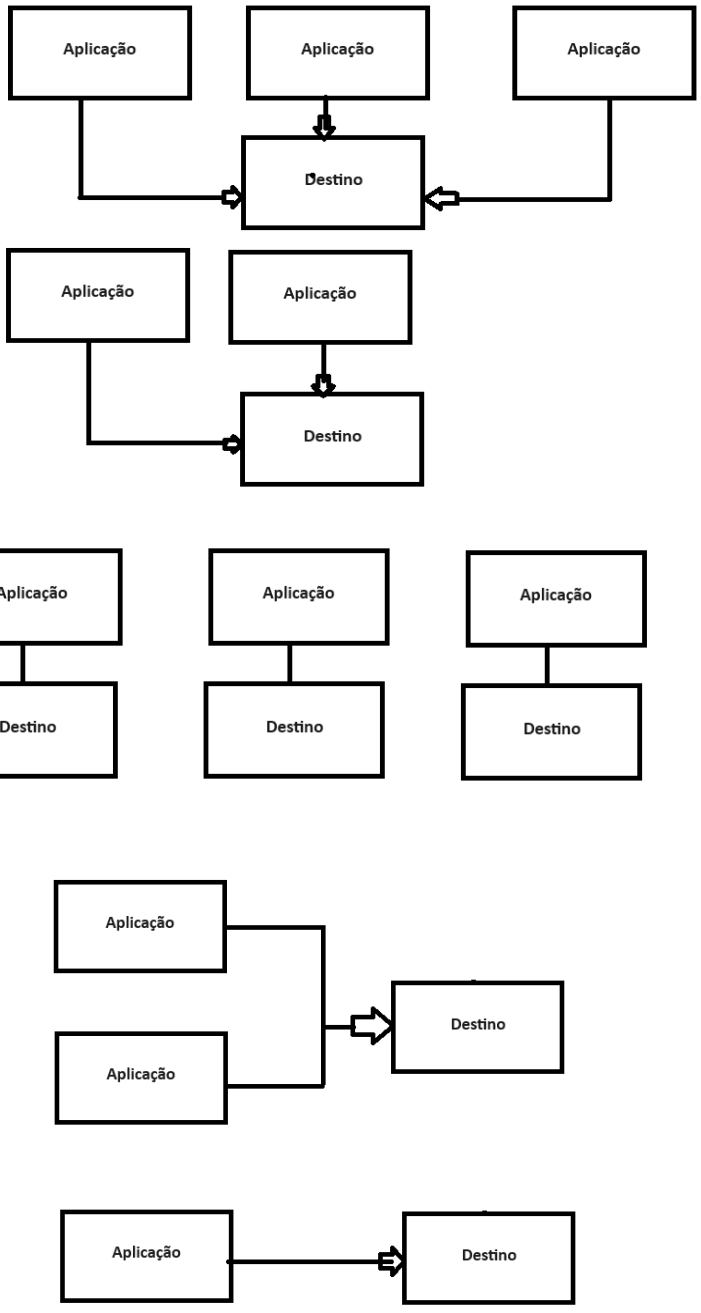
- Amanda Pageú Silva
- Gabriel Akira Wakavaiachi
- Renan Yudi Fukumori

GitHub - https://github.com/gabakiwa/Hackathon

## 1. Construção do Código

No Hackathon 2024, foi proposto o desafio de contribuir para a leitura de dados sobre o destino de diferentes arquivos criando um diagrama a fim de facilitar sua leitura e compreensão.

Para nossa resolução nós utilizamos o recurso *Collaboratory* do Google e com a linguagem python. De início, visualizamos a base de dados e, em um primeiro momento, cogitamos que poderia ser observado de três formas distintas, conforme demonstrado a seguir:



As duas primeiras imagens representam uma variação visual de um diagrama de Árvore de Decisão. Contudo, ao cogitar a quantidade robusta dos dados, chegou-se a conclusão de que a visualização poderia ficar poluída ou de difícil leitura (por exemplo, imagens muito largas). Desse modo, considerou-se que a terceira opção é a que tem maior versatilidade e adaptação para os moldes necessários.

Por meio dos códigos criamos dois diagramas, um demonstrando as movimentações dos arquivos de uma pasta origem para uma pasta destino e um outro demonstrando a movimentação da pasta origem para a pasta backup.

Para o teste do código utilizamos a base de dados disponibilizada pela Hiperstream, a seguir mostramos o resultado dos diagramas utilizando esses dados.

Como é possível observar temos o nome da origem e o nome do destino de forma de visualização fácil. Demonstramos também como alguns dos arquivos não possuem nenhuma pasta destino.

Para as pastas Backup também temos a fácil visualização da pasta origem e a pasta backup e podemos ver que todas as pastas tem um backup correspondente.

A primeira ideia que tivemos foi colocar as pastas origem junto as pastas backup, pois como dito anteriormente toda origem possui um backup, porém essa ideia foi descartada por conta de as caixas de texto possuir tanto a origem, backup e aplicação, logo um excesso de informação em apenas uma bolha, o que poderia confundir a pessoa visualizando a imagem.

Outra sugestão foi a utilização de ramificações na pasta origem para tanto a pasta backup, quanto para a pasta destino, porém ao analisarmos o grafo resultado, concluímos que a imagem estava com um tamanho muito largo e inadequado para uso.

Decidimos na criação de dois diagramas pois dessa maneira é mais fácil analisar quais são as pastas destino e quais são as pastas backup, sem que a figura seja larga demais e impossibilite a visualização. Dessa maneira podemos visualmente diferenciar uma da outra e analisar cada diagrama de forma mais relevante para o objetivo.

# 2. Código

## 2.1. Código Origem-Destino

```
import graphviz as gz
```

```
import pandas as pd
import re

url                                                                    =
"https://raw.githubusercontent.com/gabakiwa/Hackathon/main/baseparateste-
1.csv"
Base = pd.read_csv(url)

grafo = {}
dot = gz.Digraph(graph_attr={'rankdir': 'LR'})
for _, row in Base.iterrows():
    aplicacao = re.sub(r"[^\w:/-]", "", row["Nome"])
    origem = re.sub(r"\W", "", row["PastaOrigem"])
    destino    =      re.sub(r"\W",    "",    row["PastaDestino"])    if    not
pd.isnull(row['PastaDestino']) else ""

    if backup is not None:

        if origem not in grafo:
            grafo[origem] = []

            label = f"{origem}\n{row['Nome']}"
            dot.node(origem, label=label)

        grafo[origem].append(destino)

        dot.node(destino)

for k, v in grafo.items():
    for n in v:
        dot.edge(k, n)

dot.attr(rank='same')
for i, node in enumerate(grafo.keys()):
    dot.attr(rank='same', _attributes={'rank': 'same', 'same': f'{node};'})

dot.render('OrigemDestino', format='pdf', cleanup=True)
print("Diagrama do grafo dividido em camadas verticais gerado com sucesso
como 'OrigemDestino.pdf'")
```

### 2.2. Código Origem-BackUp

```
import graphviz as gz
```

```
import pandas as pd
import re

url                                                                    =
"https://raw.githubusercontent.com/gabakiwa/Hackathon/main/baseparateste-
1.csv"
Base = pd.read_csv(url)

grafo = {}
dot = gz.Digraph(graph_attr={'rankdir': 'LR'})
for _, row in Base.iterrows():
    aplicacao = re.sub(r"[^\w:/-]", "", row["Nome"])
    origem = re.sub(r"\W", "", row["PastaOrigem"])
    backup    =       re.sub(r"\W",    "",    row["PastaBackup"])    if    not
pd.isnull(row['PastaBackup']) else ""

    if backup is not None:

        if origem not in grafo:
            grafo[origem] = []

            label = f"{origem}\n{row['Nome']}"
            dot.node(origem, label=label)

        grafo[origem].append(backup)

        dot.node(backup)

        grafo[origem].append(backup)

        dot.node(backup)

for k, v in grafo.items():
    for n in v:
        dot.edge(k, n)

dot.attr(rank='same')
for i, node in enumerate(grafo.keys()):
    dot.attr(rank='same', _attributes={'rank': 'same', 'same': f'{node};'})

dot.render('OrigemBackup', format='pdf', cleanup=True)
print("Diagrama do grafo dividido em camadas verticais gerado com sucesso
como 'OrigemBackup.pdf'")
```

## 3. Observações do Código

Na linha dot = gz.Digraph(graph_attr={‘rankdir’ : ‘LR’}), para a análise da atribuição do layout hierarquico, vale ressaltar que, por melhor visualização do grafo gerado, se deu preferência ao LR (*Left to Right*) em detrimento de TB (*Top to Bottom*). Em uma visualização orientada na vertical se tem maior controla da largura da imagem, o que facilita sua manipulação caso necessário.

Ademais, optou-se, para a construção do grafo, pela biblioteca GraphViz pela sua praticidade e adaptação ao código elaborado. Foram aventadas outras alternativas, como MatPlotLib e NetworkX, para execução, mas, devido ao layout limitado que ambas oferecem e que não eram do propósito de apresentação, foram excluídas.

Já nas linhas origem = re.sub(r"\W", "", row["PastaOrigem"]) e destino= e.sub(r"\W", "", row["PastaDetino"], para a relação Origem-Destino, e backup = re.sub(r"\W", "", row["PastaBackup"] Origem-Backup, devido a presença do caractere "\" na base de dados, se teve dificuldade em encontrar uma alternativa possível de contornar o erro de leitura que se tem no código caso não exclua. Para tanto, foram realizados, infelizmente sem sucesso, testes para contornar o problema.

Uma das alternativas mais 'plausíveis' foi a consideração da substituição de "\" por, exclusivamente, “_", uma vez demais caracteres impossibilitavam a conclusão do código. O problema identificado foi a perda da formatação adequada do grafo após a geração do PDF. Nas linha if backup is not None:, se tem por objetivo o auxílio para que não exista conflito de informações e/ou erro no código caso se identifique uma eventual célula vazia.

Por fim, em dot.render('OrigemBackup', format='pdf', cleanup=True), ao pensar na praticidade e melhor visualização, optou-se pelo formato .PDF. Assim, espera-se facilitar não somente a utilidade do grafo, mas também no compartilhamento dos arquivos uma vez que, por vezes, pode se ter o cenário de dificuldade para abertura da imagem.

## 4. Considerações Finais

Durante a elaboração do desafio, considerou-se a praticidade, usabilidade e objetividade da visualização das informações sob a perspectiva de um cliente. Ou seja, do princípio de que, a partir da imagem, as relações são facilmente identificáveis e possíveis de se compreender.

Ademais, aventou-se a possibilidade de que, com o auxílio do grafo gerado, a empresa possa explicar o fluxo das informações/aplicação com facilidade. Os PDFs foram estruturados de modo que comportem número maior de informações, sem ter restrição de eventual dificuldade de visualização por ‘poluição’ das imagens.

A escolha das bibliotecas, disposição dos dados e dimensões foram levados em considerações de modo a contemplar a necessidade da empresa de transmitir as informações para os seus clientes de modo efetivo e eficaz. Além disso, destaca-se a versatilidade dos códigos, de fácil atualização e emissão de novos arquivos quando necessário.