Documentação Sprint 4

Product Owner
Gustavo Rodrigo Morais Araújo - LinkedIn

Scrum Master
Mayara Carolina da Costa Gomes - LinkedIn

Team Members
Bruna Martins dos Santos - E-mail

Davi Augusto Nascimento Santos - LinkedIn

Juliana Alves Pascuti - LinkedIn

Marcos Vinícius Camargo Santana - LinkedIn

Samantha Giovanna Martins de Lima - LinkedIn

Professor Orientador
Marcus Vinicius do Nascimento

Professor Coorientador
Jean Carlos Lourenço Costa

Objetivo

Este projeto visa oferecer percepções valiosas e soluções para melhorar a eficiência operacional e a rentabilidade da empresa parceira.

- Análise de Produtividade das rotas.
- Criação e modelagem de um banco de dados em SQL.
- Criação de um visualizador de indicadores em BI.
- Aplicação do método de transportes para otimização da distribuição.

Cronograma

Projeto pedagógico fundamentado na Metodologia API (Aprendizagem por Projetos Integradores), visando o ensino e aprendizado. Ele se baseia nos pilares de aprendizado com problemas reais, focando no desenvolvimento de competências, validação externa e mentalidade ágil. Utilizamos estratégias para compreender o problema, conceber uma solução viável durante o desenvolvimento e implementar o MVP (Mínimo Produto Viável).

Link do GitHub

Sprint	Data	Status
Kick Off	13/03/2024	Concluído
1	17/04/2024	Concluída
2	08/05/2024	Concluída
3	29/05/2024	Concluída
4	19/06/2024	Concluída
Feira de Soluções	27/06/2024	A fazer

Mínimo Produto Viável – MVP

Sprint	MVP
1	Um **preview da amostra** de dados permitindo os primeiros inputs e uma análise preliminar das rotas.
2	Propõe-se a criação de um banco de dados dedicado à armazenagem, organização e controle de dados, visando garantir a integridade da base. Tal estrutura possibilitará consultas e exibições detalhadas, adaptáveis às necessidades do usuário. Em paralelo, será desenvolvido um visualizador no Power BI, direcionado aos principais KPI's identificados na sprint anterior.
3	Realização de uma análise detalhada da amostra, incluindo uma **avaliação mês a mês** para fornecer insights sobre as tendências e variações ao longo do tempo. Além disso, criação de um código em Python para identificar o melhor cenário visando a minimização dos custos da empresa, e estabelecimento de metas futuras da organização.
4	Ajustes necessários para alinhar o projeto com a realidade operacional da organização.

Tecnologias da Informação

O **GitHub** é uma plataforma em nuvem para hospedagem e colaboração de código-fonte, amplamente utilizada por desenvolvedores e equipes de projeto. Por meio dela, é possível gerenciar e compartilhar código, conduzir revisões, e rastrear problemas de forma eficiente. Os projetos são armazenados em repositórios, permitindo que os membros da equipe visualizem e editem o código de seus trabalhos.

O **Jira Software** é uma ferramenta de gerenciamento de projetos voltada para equipes de desenvolvimento de software. Ela é essencial no planejamento, gestão e acompanhamento de projetos, tanto ágeis quanto tradicionais.

O **Microsoft Excel** é uma planilha eletrônica amplamente utilizada para organizar, manipular e analisar dados numéricos e textuais. Nele, é possível criar planilhas distribuídas em colunas e linhas, realizar cálculos, criar gráficos e tabelas dinâmicas, além de utilizar uma variedade de funções e fórmulas para processar e analisar os dados.

O **Slack** é uma plataforma de comunicação empresarial desenvolvida para facilitar a conversação entre equipes de trabalho. Projetado para substituir e otimizar a comunicação por e-mail e outras formas internas. É essencial para empresas com equipes distribuídas ou trabalhando remotamente, proporcionando uma comunicação eficiente e colaborativa, independentemente da localização física dos membros da equipe.

O **Power BI** é uma plataforma de análise de negócios que permite aos usuários visualizarem e compartilharem compreensões a partir de dados de maneira rápida e fácil. Com diversas ferramentas para coletar, transformar, visualizar e compartilhar informações de fontes diferentes, locais ou na nuvem.

O **SQL** trata-se de uma linguagem de programação utilizada para gerenciar e manipular banco de dados relacionais. Ele permite que os usuários realizem diversas consultas e operações, como inserir, atualizar, excluir e recuperar dados. É uma ferramenta poderosa para lidar com grandes conjuntos de dados de maneira eficiente. A linguagem SQL é relativamente semelhante entre os principais Sistemas Gerenciadores de Banco de Dados (SGBDs) do mercado.

Python é uma linguagem de programação de alto nível, interpretada e de propósito geral. É conhecida por sua sintaxe simples e legibilidade, o que a torna popular entre iniciantes e experientes em programação. Python é amplamente utilizado em uma variedade de domínios, como desenvolvimento web, análise de dados, inteligência artificial e automação de tarefas. Sua flexibilidade e facilidade de aprendizado contribuem para sua popularidade e ampla adoção em diferentes setores.

Pesquisa operacional

Pesquisa operacional (PO) é uma metodologia científica de análise de dados que visa proporcionar uma melhor tomada de decisão, baseada na otimização de processos, aumentando a eficiência do negócio e vem sendo aplicada em diversas áreas, como indústria, comércio e serviços (REIS, 2020).

A PO é dividida em duas abordagens: qualitativa e quantitativa. A abordagem qualitativa é baseada na experiência do tomador de decisão, em geral para problemas mais simples, enquanto a quantitativa é baseada em métodos científicos para solucionar problemas

complexos. Em ambas as abordagens, a PO é aplicada em sistemas reais, utilizando probabilidade ou relações de causalidade.

Existem cinco principais tipos de Pesquisa Operacional aplicados na melhoria de processos de decisão, que são: Programação linear, Teoria dos jogos, Simulação (como a simulação de Monte Carlo), Filas, e Teoria dos Grafos. Este trabalho utilizou a programação linear para realizar a otimização do método de transportes da empresa embarcadora.

A Programação Linear (PL) visa otimizar a utilização de recursos limitados e significa a representação dos aspectos de um problema em forma de um grupo de equações lineares, utilizando-se apenas a matemática básica na etapa de elaboração destas equações, chamada de modelagem do problema. A modelagem é o desenvolvimento de um modelo que simbolize a situação que se quer estudar ou resolver, ou seja, a tradução das características do problema para uma linguagem matemática. Geralmente os problemas de otimização com o uso de Programação Linear tem como objetivo minimizar custos ou maximizar lucros ou faturamento, pois esses são objetivos comuns das organizações (RODRIGUES, 2014).

A elaboração de um modelo de programação linear baseia-se em três etapas (FOGLIATTO, s.d.):

- Etapa 1: Definir as variáveis desconhecidas a serem determinadas (variáveis de decisão) e simbolizá-las através de símbolos algébricos (por exemplo, x e y ou x1 e x2).
- Etapa 2: Elencar todas as restrições do problema e representá-las como equações (=) ou inequações (≤, ≥) lineares em termos das variáveis de decisão definidas na etapa anterior.
- Etapa 3: Identificar o objetivo ou critério de otimização do problema, que pode ser do tipo maximizar ou minimizar, representando-o como uma função linear das variáveis de decisão.

A seguir, será descrito como foi realizada a otimização do método de transporte da empresa embarcadora trabalhada neste projeto.

Otimização com programação linear utilizando o Python

1 – Primeiramente, foi adicionado as bibliotecas e os data frames, modificando e consolidando a base.

```
[ ]  #Bibliotecas para rodar o código
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     !pip install pulp
     from pulp import * #Todas as funções da biblioteca

     from google.colab import files #Fazer o download do arquivo CSV
```

1.1 - Com a abertura dos data frames, foi realizado alguns tratamentos iniciais, como por exemplo, renomear algumas colunas e consolidar em um único data frame.

```
[ ] #Habilitando a abertura de dados que estão no Google Drive (inserindo os dados)
    from google.colab import drive
    drive.mount ('/content/drive')

    file = '/content/drive/My Drive/Fatec/6º Semestre/Tecnologia da Informação aplicada à Logística/API/Dados/'

    #Abrindo os arquivos csv
    Rotas = pd.read_csv(file+ 'Rotas.csv', sep = ';', decimal =',')

    Clientes = pd.read_csv(file+ 'Clientes.csv', sep = ',', decimal = '.')
    Clientes.rename(columns={'LAT': 'LAT_CLIENTES', 'LONG': 'LONG_CLIENTES'}, inplace=True) #Renomenado as colunas latitude e longitute para ficar mais didático a leitura dos dados

    Fabricas = pd.read_csv(file+ 'Fabricas.csv', sep = ',', decimal = '.', encoding='latin1')
    Fabricas.rename(columns={'LAT': 'LAT_FAB', 'LONG': 'LONG_FAB'}, inplace=True) #Df Fabricas
```

```
#Alterando o nome das fabricas e ID dos clientes

Base['CO.Fabrica'] = Base['CO.Fabrica'].replace({3423909: 1, 3403208: 2, 3424402: 3}) #Alterar o nome das fábricas

Base['CO.Cliente'] = Base['CO.Cliente'].astype(str).apply(lambda x: x[2:] if len(x) > 2 else x) #Alterar o ID dos clientes
Base['CO.Cliente'] = Base['CO.Cliente'].astype(int)

print(Base)
```

```
        Dt.Emissao  Dt.Entrega  Mes.Base  Ano.Exec  CO.Fabrica  CO.Cliente  \
0       01/01/2023  05/01/2023         1      2023           1          11
1       02/01/2023  05/01/2023         1      2023           1          11
2       02/01/2023  05/01/2023         1      2023           1          11
3       02/01/2023  06/01/2023         1      2023           1          11
4       03/01/2023  07/01/2023         1      2023           1          11
...            ...         ...       ...       ...         ...         ...
106257  16/11/2023  21/11/2023        11      2023           3          36
106258  23/11/2023  28/11/2023        11      2023           3          36
106259  23/11/2023  28/11/2023        11      2023           3          36
106260  24/11/2023  27/11/2023        11      2023           3          36
106261  01/12/2023  05/12/2023        12      2023           3          36
```

1.2 - Adicionado as colunas de capacidade, produtividade e frete/unidade e abaixando o csv consolidado.

```
[ ]  #Cria nova: coluna condicional (Capacidade dos veículos)
     Base['Capacidade'] = None
     Base.loc[Base['Veiculo'] == 'P24', 'Capacidade'] = 3600
     Base.loc[Base['Veiculo'] == 'P12', 'Capacidade'] = 1800
```

```
[ ]  #Cria nova coluna: Produtividade
     Base['Produtividade'] = pd.to_numeric(Base['Qtd.Transp'] / Base['Capacidade'] * 100).round(2)
     print(Base)
```

Mostrar saída oculta

```
[ ]  #Adicionar coluna: Frete/unidade
     Base['Vlr.Frete'] = Base['Vlr.Frete'].astype(float)
     Base['Frete/unidade'] = (Base['Vlr.Frete'] / Base['Qtd.Transp']).round(2)

     pd.set_option('display.max_columns', None)
     print(Base)
```

Mostrar saída oculta

```
[ ]  # Exportar a base para CSV
     Base.to_csv('Base_tratada.csv', index=False, sep=";", decimal=",")
     files.download('Base_tratada.csv')
```

1.3 - Criando gráfico da distribuição dos Incoterms

Nesta etapa, foi inserido comando para possibilitar a construção de um gráfico responsável por ilustrar o percentual de cada incoterm presente na rota trabalhada.

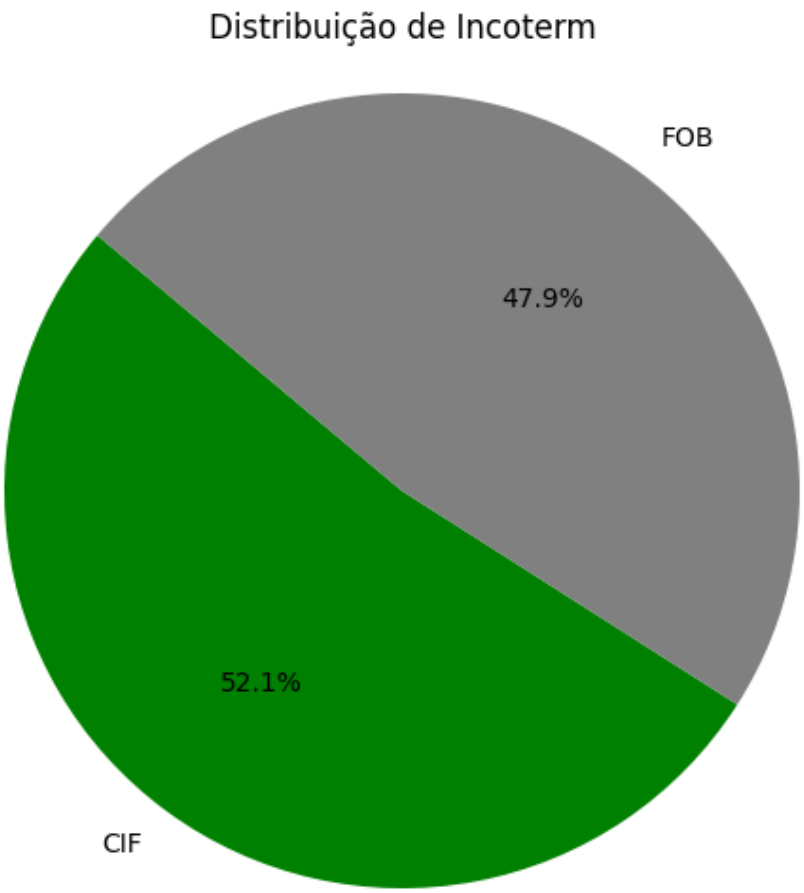
```
[ ]  def plotar_grafico_pizza(Base):

         contagem_incoterm = Base['Incoterm'].value_counts()     # Conta a quantidade de ocorrências de cada tipo de Incoterm

         # Define as cores das fatias
         cores = ['green', 'grey']  # FOB: verde, CIF: cinza

         # Cria o gráfico de pizza com as cores personalizadas e texto em branco
         plt.figure(figsize=(8, 6))
         plt.pie(contagem_incoterm, labels=contagem_incoterm.index, autopct='%1.1f%%', startangle=140, colors=cores)
         plt.title('Distribuição de Incoterm')
         plt.axis('equal')  #Garante que o gráfico de pizza seja desenhado como um círculo.
         plt.show()

     # Chama a função para plotar o gráfico de pizza com base na coluna 'Incoterm' do DataFrame 'Base'
     plotar_grafico_pizza(Base)
```



****até aqui as imagens do Python estão iguais - JU

Parte 2 – Pesquisa Operacional (Aplicação do Método de Transportes)

Após realização de tratamento dos dados necessários, foi desenvolvida a etapa da pesquisa operacional para resolução do problema.

2.1 - Criando problemática e inserindo os dados das restrições

Nesta etapa, foi inserida as restrições do problema, sendo estas: “Capacidade” e “Demanda”. Foi aplicada também o tipo de problema a ser solucionado, definido como: “Minimização”. Foram inseridas as variáveis de decisão, para que o código desenvolvesse a solução baseado nos parâmetros descritos.

```
[ ] #Inserindo os dados das restrições: Capacidade
    Capacidade = pd.read_csv(file+ 'Capacidade.csv', sep = ';', decimal = '.')
    Capacidade['CO.Fabrica'] = Capacidade['CO.Fabrica'].replace({3423909: 1, 3403208: 2, 3424402: 3}) #Alterar o nome das fábricas

    #Inserindo os dados das restrições: Demanda
    Demanda = pd.read_csv(file+ 'Demanda.csv', sep = ';', decimal = '.')
    Demanda['CO.Cliente'] = Demanda['CO.Cliente'].astype(str).apply(lambda x: x[2:] if len(x) > 2 else x) #Alterar o ID dos clientes
    Demanda['CO.Cliente'] = Demanda['CO.Cliente'].astype(int)
    Demanda['Demanda_ajustada'] = (Demanda['Demanda'] * 0.521).round(decimals=0).astype(int) #Criando a coluna com a demanda ajustada de acordo com os dados CIF

    print(Capacidade)
    print(Demanda)

    #Criando e definindo o tipo de problema
    problem = LpProblem('Minimizacao_de_custos_operacionais', LpMinimize)
```

```
[ ] # Criando as variáveis de decisão
    x = LpVariable.dicts("Envios", [(i, j) for i in Capacidade['CO.Fabrica'] for j in Demanda['CO.Cliente']],
                         lowBound=0, cat='Continuous')
```

2.2 - Criando as restrições do problema

Nessa etapa foram aplicadas as restrições para dar continuidade ao problema a ser solucionado.

```
[ ] # Adicionando as restrições de capacidade das fábricas
    for i, row in Capacidade.iterrows():
        problem += lpSum(x[(row['CO.Fabrica'], j)] for j in Demanda['CO.Cliente']) <= row['Capacidade'], f"Capacidade_Fabrica_{row['CO.Fabrica']}"

    # Adicionando as restrições de demanda dos clientes
    for j, row in Demanda.iterrows():
      problem += lpSum(x[(i, row['CO.Cliente'])] for i in Capacidade['CO.Fabrica']) == row['Demanda_ajustada'], f"Demanda_Cliente_{row['CO.Cliente']}"
```

2.3 - Criando a função objetivo

Após aplicação das restrições do problema, foi desenvolvida a função objetivo que o código deve atingir, com intuito de minimizar o custo total de transporte.

```
# Criando um dicionário a partir do DataFrame Custo_mod
custo_dict = Custo_mod['Custo_Medio_Ajustado'].to_dict()

# Criando e definindo o tipo de problema
problem = LpProblem('Minimizacao_de_custos_operacionais', LpMinimize)

# Criando as variáveis de decisão
x = LpVariable.dicts("Envios", [(i, j) for i in Capacidade.index for j in Demanda.index],
                     lowBound=0, cat='Continuous')

# Adicionando as restrições de capacidade das fábricas
for i, row_capacidade in Capacidade.iterrows():
    problem += lpSum(x[(i, j)] for j in Demanda.index) <= row_capacidade['Capacidade'], f"Capacidade_Fabrica_{i}"

# Adicionando as restrições de demanda dos clientes
for j, row_demanda in Demanda.iterrows():
    problem += lpSum(x[(i, j)] for i in Capacidade.index) >= row_demanda['Demanda'], f"Demanda_Cliente_{j}"

# Definindo a função objetivo de forma explícita como uma expressão linear
objective_function = lpSum(x[(i, j)] * custo_dict.get(f'X{i}{str(j).zfill(2)}', 0) for i in Capacidade.index for j in Demanda.index)

# Adicionando a função objetivo ao problema
problem += objective_function, "Custo_Total_Transporte"
```

2.4. Resolução

Feita a elaboração da função objetivo, foi aplicado o comando responsável pela resolução do problema.

```
[ ]  # Resolver o problema
     problem.solve()

     # Verificando o status da solução
     print("Status:", LpStatus[problem.status])

     # Exibindo o custo total de transporte
     print("Custo Total de Transporte:", value(problem.objective))

     # Exibindo as quantidades enviadas de cada fábrica para cada cliente
     for i in Capacidade['CO.Fabrica']:
         for j in Demanda['CO.Cliente']:
             if x[(i, j)].varValue > 0:
                 print(f"Enviar {x[(i, j)].varValue} unidades da fábrica {i} para o cliente {j}")
```

```
     # Inicializando listas para os dados
     resultado = []

     # Adicionando os resultados à lista de dados
     for i in Capacidade['CO.Fabrica']:
         for j in Demanda['CO.Cliente']:
             if x[(i, j)].varValue > 0:
                 resultado.append([i, j, x[(i, j)].varValue])

     # Criando o dataframe
     resultado = pd.DataFrame(resultado, columns=['Fabrica', 'Cliente', 'Unidades_enviadas'])

     #Criar o novo índice no formato Xij como uma nova coluna
     resultado['Xij'] = 'X' + resultado['Fabrica'].astype(str) + resultado['Cliente'].astype(str).str.zfill(2)

     # Remover as colunas 'Fabrica' e 'Cliente'
     resultado.drop(['Fabrica', 'Cliente'], axis=1, inplace=True)

     # Reordenar as colunas para colocar 'Xij' antes de 'Unidades Enviadas'
     resultado = resultado[['Xij', 'Unidades_enviadas']]

     # Realizar o merge (junção) dos DataFrames usando 'Xij' como chave
     resultado_com_custo = pd.merge(resultado, Custo_mod, on='Xij', how='left')

     print(resultado_com_custo)
```

```
     #Download do resultado para validação
     resultado_com_custo.to_csv('Otimizado.csv', index=False, sep=";", decimal=',')
     files.download('Otimizado.csv')
```

REFERÊNCIAS BIBLIOGRÁFICAS

FOGLIATTO, Flavio. *Pesquisa Operacional*. Apostila. [s.d.]. Disponível em: < http://www.producao.ufrgs.br/arquivos/disciplinas/382_po_apostila_completa_mais_livro.pdf >. Acesso em: 18 jun. 2024.

REIS, Tiago. Pesquisa operacional. 2020. Disponível em: <https://www.suno.com.br/artigos/pesquisa-operacional/#:~:text=Pesquisa%20operacional%20(tamb%C3%A9m%20conhecida%20pela,meio%20da%20otimiza%C3%A7%C3%A3o%20de%20processos.>. Acesso em: 18 jun. 2024.

RODRIGUES, Luís Henrique et al. *Pesquisa Operacional – Programação Linear Passo a Passo: Do Entendimento do Problema à Interpretação da Solução*. Editora Unisinos, 2014. Disponível em: <http://biblioteca.asav.org.br/vinculos/000045/000045c5.pdf>. Acesso em: 18 jun. 2024.