CENTRO UNIVERSITÁRIO CENTRAL PAULISTA
INTELIGÊNCIA ARTIFICIAL

GISMAR PEREIRA BARBOSA

ESTUDO DE CASO: MAPEAMENTO DE UM SISTEMA DE LOGÍSTICA

RIO CLARO, 2021

GISMAR PEREIRA BARBOSA

ESTUDO DE CASO: MAPEAMENTO DE UM SISTEMA DE LOGÍSTICA

Estudo de caso (experimento) apresentado à disciplina de Inteligência Artificial, ministrada pelo professor Dr. Erik Aceiro Antônio no Centro Universitário Central Paulista, UNICEP, na cidade de Rio Claro - SP. A apresentação deste estudo de caso (experimento), conforme solicitado, tem peso 100% na nota da P1 e T1 de 2021.

Rio Claro, 2021

RESUMO

Este estudo de caso está sendo realizado com o objetivo de entender como ocorre o fluxo de processos (geral) em um sistema logístico de entregas, assim como, analisar e empregar técnicas de manipulação de dados (estatísticas e analíticas), para, baseado nas informações e artefatos disponíveis, inferir sobre o que poderia ser realizado para melhoria do fluxo, como um todo, com a utilização de automação computacional em alguns pontos específicos do processo. Além do mais, também empregar técnicas de IA, que possam agregar ao processo mais “performance”. Dentro das técnicas de IA, as que se poderiam aplicar ao estudo de caso, seriam a Smart Data Discovery (ciclo de BI automatizado) para questões de gerenciamento de dados do negócio, visando os níveis táticos e estratégicos, e a Análise Preditiva, auxiliando no ciclo de processos (BPMN), para criar uma automação inteligente e eficiente nas etapas do fluxo de negócios. Adequando-se a uma metodologia experimental (teste de hipóteses), com carácter qualitativo (buscando uma melhor solução para um determinado conjunto de problemas), o estudo será conduzido, inicialmente analisando e modificando, dentro do necessário os documentos de fluxo de processo disponibilizado para o trabalho. E, posteriormente, será utilizado um arquivo de dados (com informações coletados do fluxo de processo de negócios), para inserir métricas de avaliação (MTTF, MTTR, MTBF e disponibilidade), e assim, inferir soluções baseados nos dados analisados. Esse procedimento se assemelha ao que é realizado em ambientes profissionais, quando se é feito uma sugestão e/ou solicitação de mudança de processo, buscando melhorias como um todo. Como resultados, considerações e/ou conclusões, é verificado, que o processo possui muitos problemas de performance (perda ritmo, tempos altos de parada etc), e que este podem estar relacionado com processos manuais (realizados por pessoas), demonstrando que automação destes, com uso das técnicas de IA já descritas anteriormente, poderiam ser eficazes para aumento de produtividade e melhoria do processo como um todo.

Palavras-chave: IA; MTTF; MTTR; MTBF; Disponibilidade; Processos; Sistema Logístico; BPMN.

SUMÁRIO

INTRODUÇÃO

Este estudo de caso está sendo desenvolvido com a premissa da avaliação e verificação do fluxo de um sistema de logística de entrega de mercadorias. Por meio deste, a intenção é, após a análise estatística e analítica da cadeia de processos existentes no objeto de estudos (fluxo de processos de negócios), inferir opções e/ou sugestões de melhorias para aplicação no processo (do ponto de vista de negócios).

Ao longo deste documento, serão demonstradas alguma técnicas de manipulação de dados, buscando entendê-los, para encontrar informações implícitas nos dados, sobre o processo logístico, que possam ou não, suportar a mudança e/ou melhoria em tais processos.

1. DESENVOLVIMENTO

Este estudo de caso será conduzido com o foco acadêmico (contexto orientado ao experimento voltado para o aprendizado).

Atualmente a Inteligência Artificial (IA) está diretamente relacionada com nossas atividades diárias. Nesse contexto, é importante a) compreender; b) decidir qual tecnologia aplicar e c) conceber uma ideia de uso da Inteligência Artificial (IA).

Orientado as premissas deste estudo de caso, será realizada uma atividade experimental com embasamento estatístico e analítico, sobre os dados fornecidos para esta atividade.

O contexto principal (e hipotético) baseia-se em um sistema logístico de transporte de mercadorias que possui problemas (em níveis de negócios e estruturação de processos) e, este necessita que sejam levantadas soluções, para que o processo com um todo seja melhorado.

Inicialmente, o modelo de negócios do referido sistema está disposto na Figura 1 (vide APÊNDICE I), e o arquivo de dados pode ser encontrado no arquivo “Fontes de Dados” (vide APÊNDICE II).

E destes dados informados acima, iniciamos nosso estudo de caso (experimento).

1.1 OBJETIVO GERAL

Baseado na análise, aplicação e modelagem de dados, o objetivo é encontrar informações que sustentem as modificações e melhorias no sistema de logística de transporte disposto.

1.1.1 Objetivos específicos

De maneira específica, os objetivos podem ser divididos conforme as questões colocadas e solicitadas como atividades:

1. Mapear e descrever através de um fluxo de atividades as principais atividades desenvolvidas no dia-a-dia de trabalho de um sistema de Logística de Entregas de Mercadorias, como disponível na Figura 1 (vide APÊNDICE I).

2. Usar o diagrama BPMN com a Ferramenta (vide link: https://lucid.app/) - Acrescente se necessário outros pontos.

3. A partir desse mapeamento é possível identificar lacunas e eventualmente falhas processuais pois algumas delas envolvem atividades manuais. Nesse contexto, é importante medir (mensurar) essas atividades (vide link: https://databinteligencia.com.br/indicadores-de-manutencao-o-que-sao-e-como-calcular-o-mttr-e-mtbf/). Desta forma utilizar métricas MTTF, MTTR, MTBF e Disponibilidade.

4. Construir uma planilha no Excel (ou em uma linguagem de processamento como o Python) para medir as métricas solicitadas (com base no arquivo dataset.csv, disponibilizado para o estudo de caso).

5. A partir da avaliação inicial do item 3 indique qual técnica de IA melhor poderia ser incorporado ao processo definido no item 2:

- Análise de investimento em ações / projetos de inovação;
- Predição de índices financeiros;
- Análise de créditos;
- Aplicação na área médica (realização de pré-diagnósticos);
- Controle de estoque;
- Predição da concentração cáustica (produção de alumina);
- Previsão de preço do adubo;
- Controle de dosagem de cloro em Estação de Tratamento de Água;
- Contribuição da IA para o desenvolvimento de tecnologias assistivas;
- Aplicação em gestão e previsão de vendas;

- Seleção de fornecedores;
- Controle de tráfego urbano.

6. Desenhe novamente o diagrama do item 2 agora com a indicação da solução proposta do item 5.

1.2 METODOLOGIA

Este estudo de caso possui uma metodologia experimental (teste de hipóteses), com carácter qualitativo (buscando uma melhor solução para um determinado conjunto de problemas).

O estudo será conduzido, inicialmente analisando e modificando, dentro do necessário os documentos de fluxo de processo disponibilizado para o trabalho. E, posteriormente, será utilizado um arquivo de dados (com informações coletados do fluxo de processo de negócios), para inserir métricas de avaliação (MTTF, MTTR, MTBF e disponibilidade), e assim, inferir soluções baseados nos dados analisados.

O passo a passo será baseado nas premissas traçadas para esta atividade (vide 1.1.1 Objetivos específicos). Mas, de forma, resumida, iremos utilizar o software “diagramas.net” (também conhecido como “draw.io”, instalado em máquina local, vide link: https://app.diagrams.net/) para criar e atualizar os fluxogramas de processos e BPMN. Para modelar dados e calcular as métricas, iremos utilizar o software “JupyterLab” (instalado em máquina local, utilizado via browser, vide link: https://jupyter.org/). Para escrita deste relatório, o “LibreOffice Writer” (instalado em máquina local, vide link: https://www.libreoffice.org/discover/libreoffice/). Além disso, toda documentação estará presente no Github (vide link: https://github.com/gismarb/projeto-IA-p1).

1.3 PROCEDIMENTOS EXPERIMENTAIS

Os procedimento, irão de encontro com o que já foi citado nos objetivo (vide 1.1.1 Objetivos específicos), assim como na forma da condução do experimento (vide 1.2 METODOLOGIA).

Como uma forma de roteiro, seguem:

1. Analisar fluxograma de processo disponibilizado sobre o sistema de logística de entrega de mercadorias, como mostra Figura 1 (vide APÊNDICE I);

2. Mapear no fluxo, problemas existentes e já identificados, como mostra Figura 2 (vide APÊNDICE I);

3. Redesenhar fluxo, com correções necessárias, como mostra Figura 3 (vide APÊNDICE I);

4. Criar uma proposta de fluxo de negócios (mapeamento BPMN), como mostra Figura 4 (vide APÊNDICE I);

5. Mapear no fluxo BPMN proposto, pontos de atenção e condução (execução) humana, como mostra Figura 5 (vide APÊNDICE I);

6. Modelar dados e calcular métricas (MTTF, MTTR, MTBF e Disponibilidade) , baseados na planilha e/ou arquivo CSV disponibilizado, conforme arquivo “Fonte de Dados” (vide APÊNDICE II), demonstrando os códigos utilizados, conforme arquivos “Métricas de Avaliação” e “Métricas de Avaliação e Resultados” (vide APÊNDICE II);

7. Identificar e propor técnicas de IA, baseadas nas soluções demonstradas anteriormente (vide 1.1.1 Objetivos específicos), para serem implementadas nos processos de negócios;

8. Redesenhar o mapeamento BPMN, com possíveis alterações necessárias, como mostra Figura 6 (vide APÊNDICE I);

9. Escrever relatório sobre o experimento e estudo de caso;

10. Criar repositório no Github, e disponibilizar o conteúdo (vide link: https://github.com/gismarb/projeto-IA-p1).

1.4 RESULTADOS

Baseados nos dados fornecidos (fluxograma de processos e arquivos de dados de atividades), é possível inferir que o processo como um todo tem muitas falhas. Estas são localizadas em várias partes do processo.

Inicialmente, ao analisar o fluxograma representado na Figura 1 (vide APÊNDICE I), é possível verificar que o mesmo está incompleto: não existe uma continuidade do processo após o mesmo aguardar pela inspeção da Receita Federal. Outro erro que podemos verificar é, a inexistência, também, da continuidade do processo, caso a mercadoria seja do tipo “Correspondência”. Desta forma, existe a necessidade de remapear este fluxo, para corrigir estas estas falhas, demonstradas na Figura 2 (vide APÊNDICE I). De forma lógica, é inferido e adicionado novos pontos de controle no fluxograma, como mostra a Figura 3 (vide APÊNDICE I) para solucionar os referidos problemas.

Com o fluxograma macro de processo organizados, é possível inferir uma modelagem mais específica, utilizando o BPMN, como mostra as Figuras 4 e 5 (vide APÊNDICE I), identificando as 5 linhas de processos principais: recebimento, pré-triagem, triagem, inspeção da Receita Federal, devolução e entrega; assim como os pontos do processo, onde temos ocorrência de trabalho manual (pessoas).

Com os processos modelados, é possível analisar os dados com maior acurácia.

Analisando os dados de entregas, pré triagem e raio X, presentes no arquivo “Fonte de Dados” (vide APÊNDICE II), disponibilizado em formato CSV (dataset.csv) e, inferindo métricas, assim como ajustes nas modelagem dos dados, é possível encontrar as informações presentes na Tabela 1 (vide Tabela 1: Demonstração dos Resultados obtidos na aplicação das Métricas.):

Tabela 1: Demonstração dos Resultados obtidos na aplicação das Métricas.

Processo	MTTF	MTTR	MTBF	Disponibilidade
Entrega	8h16min	36h28min	8h33min	18.99%
Pré Triagem	8h56min	38h	9h14min	19.56%
Raio X	8h12min	36h20min	8h28min	18.93%

Fonte: elaborado pelo próprio autor (2021).

Os dados referentes e utilizados para inferir os resultado presentes na Tabela 1 (vide Tabela 1: Demonstração dos Resultados obtidos na aplicação das Métricas.), podem ser encontrados no arquivos “Métricas de Avaliação” e “Métricas

de Avaliação e Resultados” (vide APÊNDICE II) e demais códigos, presentes no repositório Github (vide link: https://github.com/gismarb/projeto-IA-p1).

Todos os dados presentes neste relatório, podem ser conferidos também no Github (vide link: https://github.com/gismarb/projeto-IA-p1), onde estão presentes todos os artefatos utilizados na realização de estudo de caso e/ou experimento.

2. CONCLUSÕES E RECOMENDAÇÕES

Indo direto ao ponto, é possível concluir, primariamente que: levando em consideração a quantidade de dados que fornecidas para o estudo de caso, não podemos afirmar de fato, ou propor uma solução correta para os problemas. Ainda necessitamos de informações como metas de atendimentos, tempos totais diários para operações, tempo de WIP (work in progress), tipo de pardas existentes (quais são os motivos de paradas ?), dentre outras informações.

Tendo a primaria conclusão, agora, baseado nos dados disponibilizados, coletados e modelados, podemos perceber que as falhas nos processo são identificadas tanto no modelo simbólico dos processos, quanto no analítico. E, baseado nestes dados (e somente nestes analisados e no corpo de deste estudo de caso), podemos identificar:

- Tempo de paradas em demasia (e aqui, assumimos que todos estes tempos são problemas de manutenção);
- Processo não contínuo: o que pode demonstrar a existência de WIP, ou processo assíncrono;
- Perda de ritmo;
- Disponibilidade baixa, para um processo em escala de “linha de produção”.

Levando em consideração, que de acordo com a Figura 5 (vide APÊNDICE I), os principais pontos de cada processo, são operados em processo manuais (pessoas), e são baseados em ações repetitivas, uma primeira alternativa como solução seria automatização deste processos, como mostra a Figura 6 (vide APÊNDICE I):

- Uso de esteiras automáticas;
- Scanners e sensores de presença, peso, superfície e outros equipamentos, com tecnologia de IoT e fornecendo dados para análise preditiva (IA);

- Atendimento eletrônico (sistemas robôs) nos pontos de controle do processo, utilizando a predição para tomada de decisões;

- Automatização de envio de follow-ups e mensagens, baseadas em análise preditiva e recursos de IoT.

Essas soluções, com uso da automação computadorizada (IoT, sensores inteligentes), e os Sistemas de Predição (uma IA criada a partir de análise de dados e processos da empresa, criando tendências para tomadas de decisão), citados anteriormente, nos pontos específicos mostrados na Figura 6 (vide APÊNDICE I), onde as interações são repetitivas e “mecânicas”, poderiam solucionar problemas de performance citados anteriormente.

Outra solução adequada, seria, buscando o melhor controle de resultados, assim como entender a métricas e traçar indicadores para metas, o uso de Smart Data Discovery (uso automatizado do BI, aplicando técnicas de IA). Com isso, e as demais soluções já indicadas, em nível tático e estratégico, seria possível a tomadas de decisões assertivas para melhorias no processo como um todo.

Novamente, as proposições citadas neste documento, são única e exclusivamente relacionadas aos materiais de uso do mesmo.

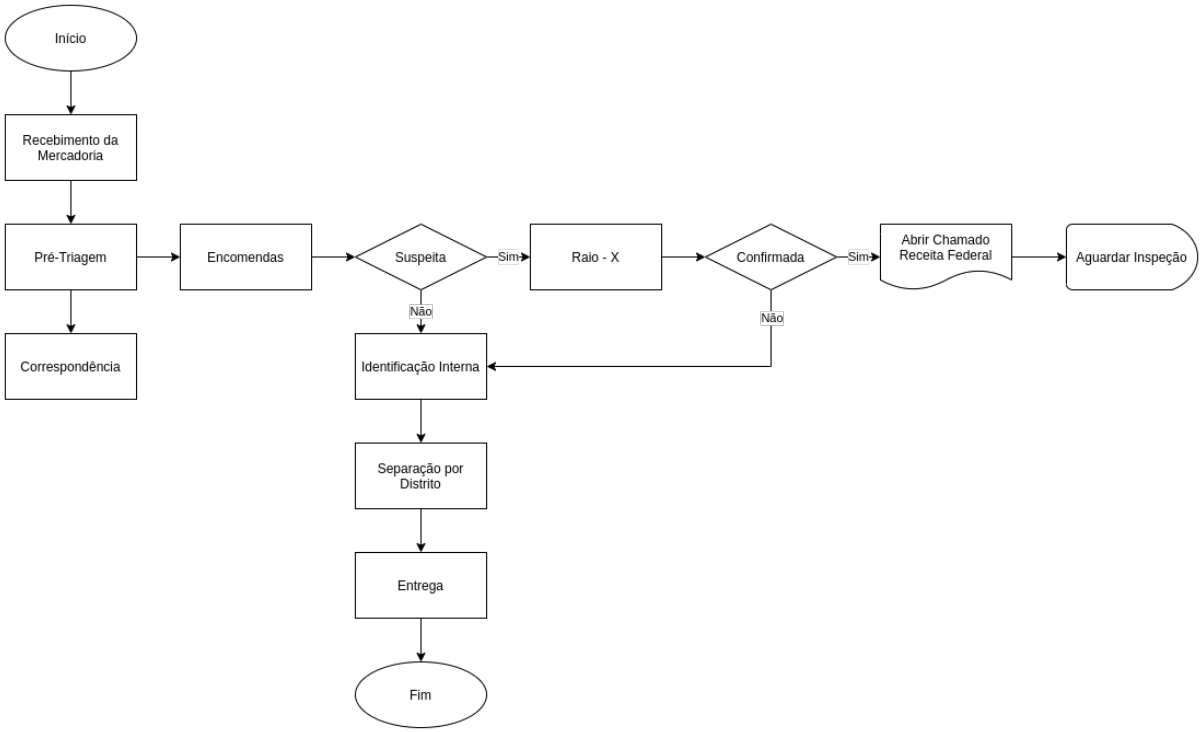
Outra situação é que, as proposições de melhorias não estão relacionadas dentre as solicitadas (vide 1.1.1 Objetivos específicos), pois estas, as solicitadas, não se aplicam ao processo e problema em questão neste estudo de caso.

Desta forma, o desenho final do BPMN, como mostra na Figura 6 (vide APÊNDICE I), seria uma solução macro para o problema de performance existente no sistema de logística, em conjunto com as demais soluções já propostas anteriormente (especificadas).

APÊNDICE I

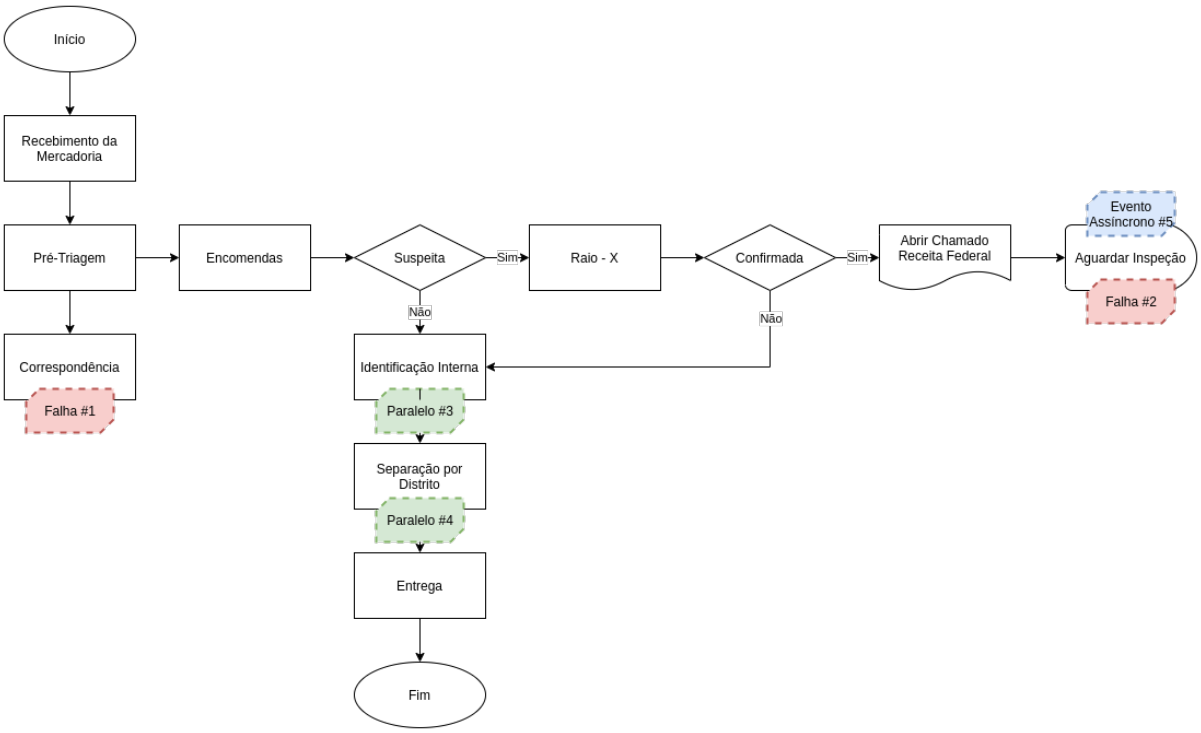
O apêndice I deste estudo de caso contém as imagens do experimento referenciadas neste relatório.

Figura 1: Fluxo Original Processo Logístico.



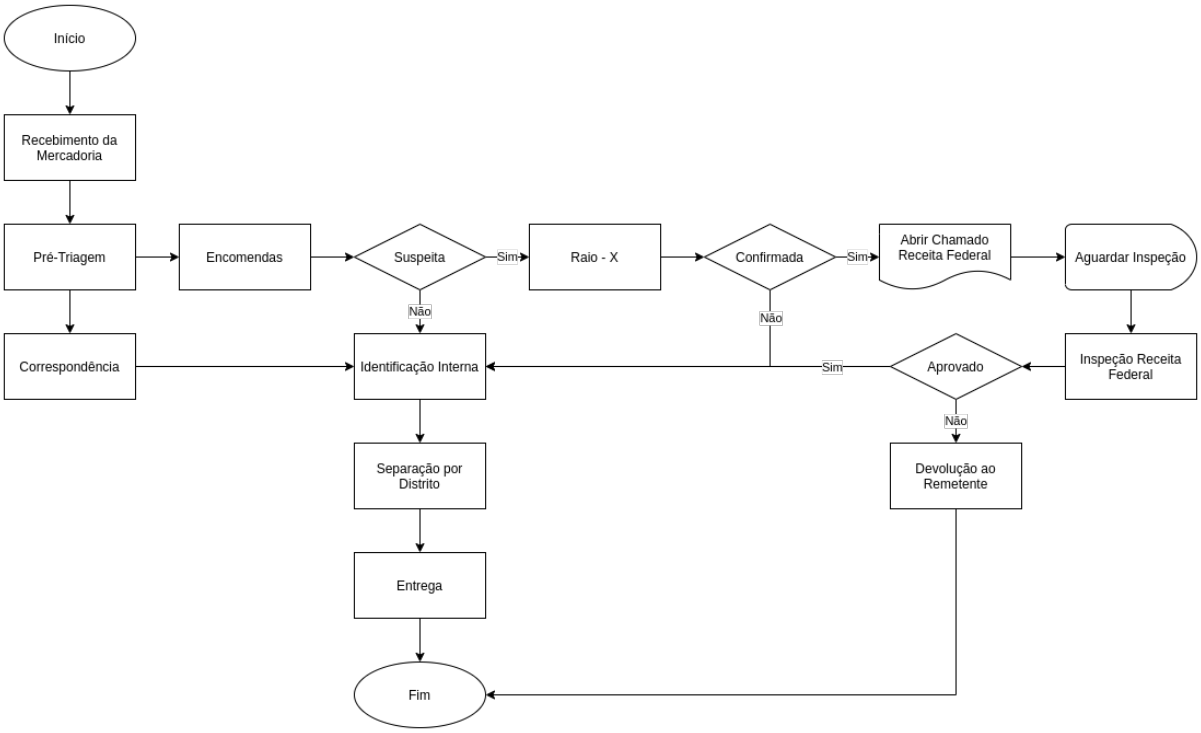
Fonte: adaptado do fluxo fornecido pelo Professor (2021).

Figura 2: Mapeando obervações no Fluxo do Processo Logístico.



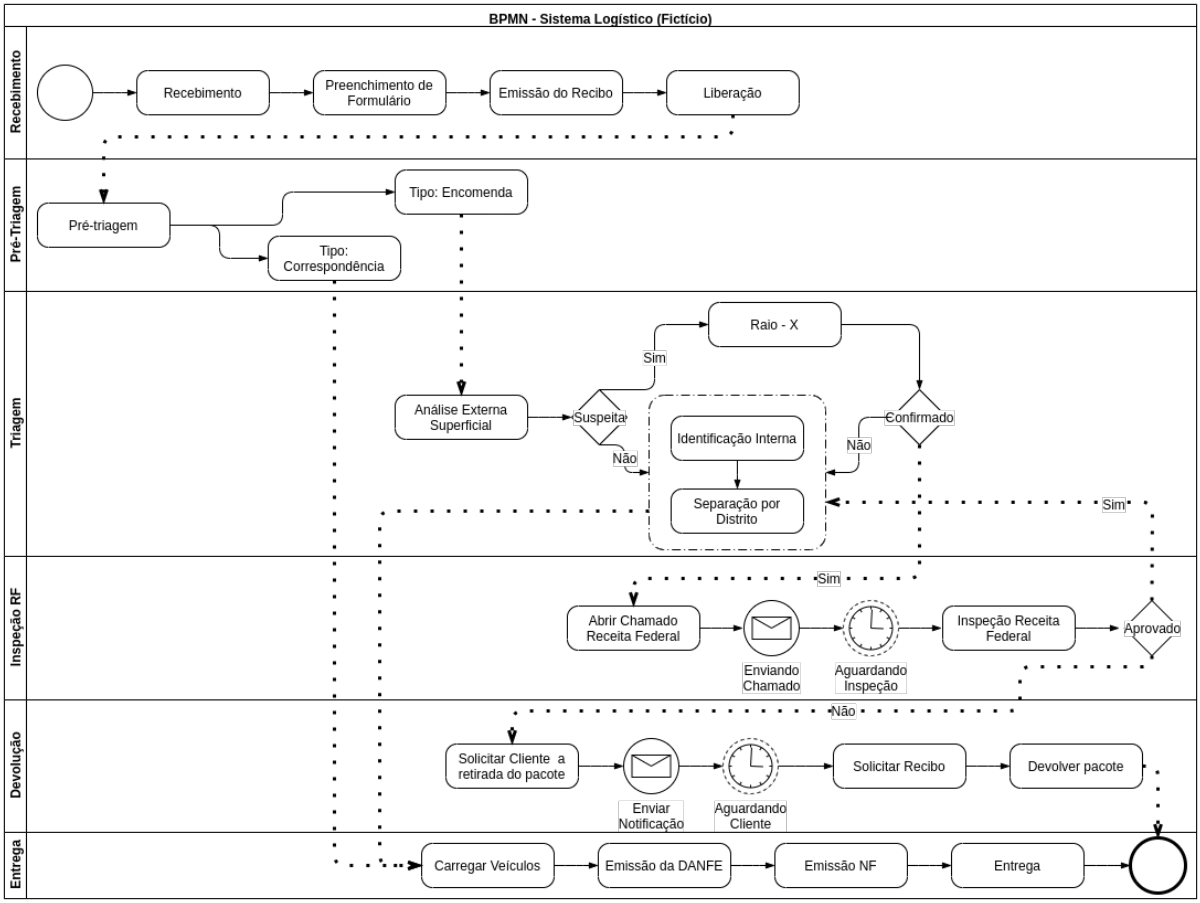
Fonte: elaborado pelo próprio autor (2021).

Figura 3: Fluxo Original (atualizado) Processo Logístico.



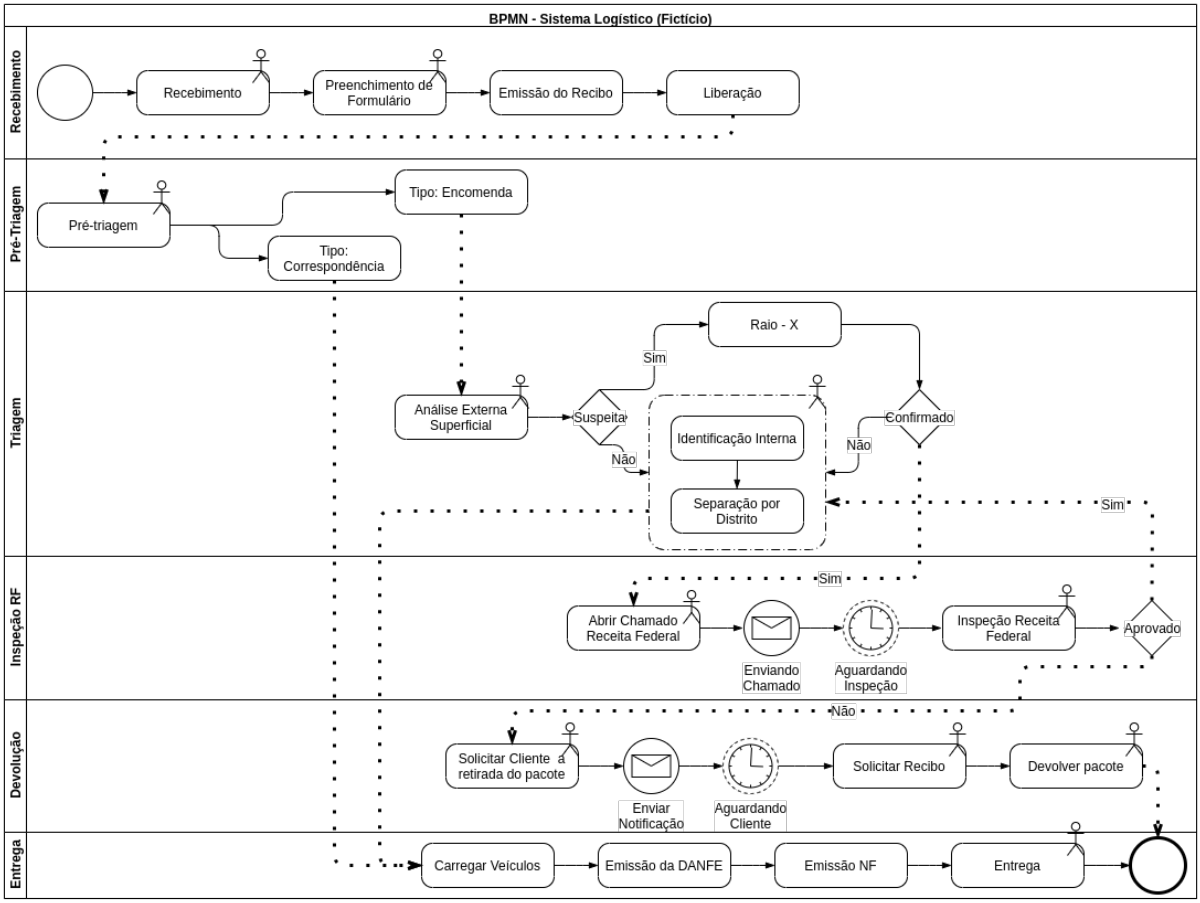
Fonte: elaborado pelo próprio autor (2021).

Figura 4: Fluxo BPMN (inicial) Processo Logístico.



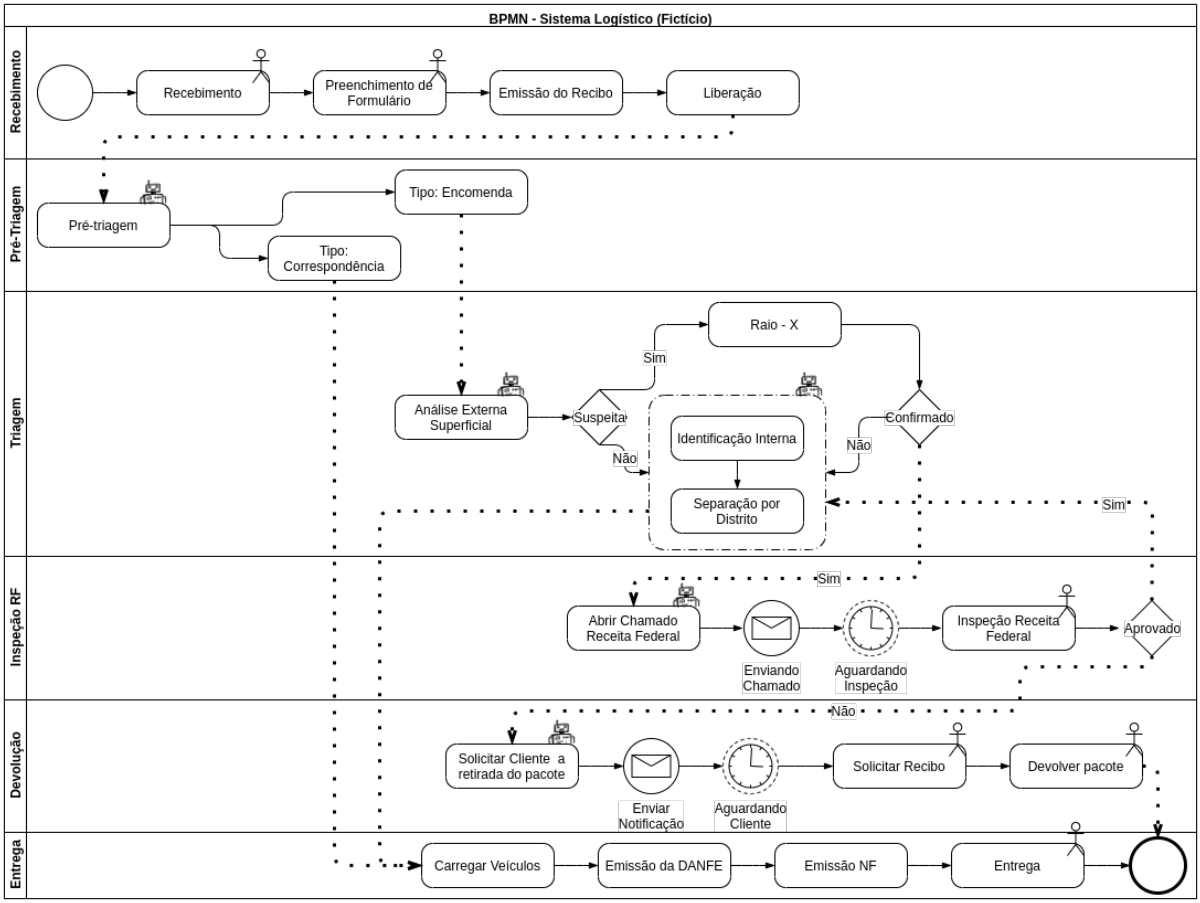
Fonte: elaborado pelo próprio autor (2021).

Figura 5: Fluxo BPMN (usuário) Processo Logístico.



Fonte: elaborado pelo próprio autor (2021).

Figura 6: Fluxo BPMN (IA e automação) Processo Logístico.



Fonte: elaborado pelo próprio autor (2021).

APÊNDICE II

O apêndice II corresponde ao dados utilizados neste estudo de caso. Aqui, estão os arquivos de dados, planilha de manipulação e extração de informações (arquivos utilizados no JupyterLab), baseados nos dados inferidos por esta fonte de dados (arquivo CSV).

Lista de arquivos presentes neste Apêndice, presentes também no Github (vide link: https://github.com/gismarb/projeto-IA-p1):

- Métricas de Avaliação: arquivo PDF referente as métricas utilizadas para avaliar os indicadores solicitados (MTTF, MTTR, MTBF e Disponibilidade). O arquivo markdown pode ser encontrado no Github (vide link: https://github.com/gismarb/projeto-IA-p1) do estudo de caso;

- Métricas de Avaliação e Resultados: PDF referente as métricas utilizadas para avaliar os indicadores solicitados (MTTF, MTTR, MTBF e Disponibilidade). O arquivo markdown pode ser encontrado no Github (vide link: https://github.com/gismarb/projeto-IA-p1) do estudo de caso;

- Fonte de Dados: arquivo dataset.csv (em anexo, estará uma versão do mesmo em PDF). Este arquivo também pode ser encontrado no Github(vide link: https://github.com/gismarb/projeto-IA-p1).

In [ ]:
```
# Carregando Pandas para uso nos calculos das planilhas
import pandas as pd
from datetime import datetime
# Carregando arquivo.csv em memória (para manipulaçaõ)
# Em meu caso, eu estou modificando o "df = pandas.read_csv('../Database/dataset.csv')"
# Eu estou utilizando uma função lambda para usar um parse no campo datas
dateparse = lambda x: datetime.strptime(x, '%H:%M:%S')
df = pd.read_csv('../Database/dataset.csv', parse_dates=True, date_parser=dateparse)
df.head(31)
```

In [ ]:
```
#
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Recebimento de Mercadoria
#
```

In [ ]:
```
# Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_recebimento_mercad
```

In [ ]:
```
df.sort_values(by=['tempo_inicio_recebimento_mercadoria'])
```

In [ ]:
```
tempo_inicio_recebimento_mercadoria = df['tempo_inicio_recebimento_mercadoria']
```

In [ ]:
```
tempo_fim_recebimento_mercadoria = df['tempo_fim_recebimento_mercadoria']
```

In [ ]:
```
tempo_inicio_recebimento_mercadoria = pd.to_datetime(tempo_inicio_recebimento_mercadoria)
```

In [ ]:
```
tempo_fim_recebimento_mercadoria = pd.to_datetime(tempo_fim_recebimento_mercadoria)
```

In [ ]:
```
# Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
```

In [ ]:
```
tempo_total_disponivel_recebimento_mercadoria = tempo_fim_recebimento_mercadoria[7] - tempo_inicio_recebimento_mercadoria[15]
```

In [ ]:
```
tempo_total_disponivel_recebimento_mercadoria
```

In [ ]:
```
tempo_duracao_recebimento_mercadoria = tempo_fim_recebimento_mercadoria - tempo_inicio_recebimento_mercadoria
```

In [ ]:
```
tempo_duracao_recebimento_mercadoria
```

In [ ]:
```
# Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

In [ ]:
```
tempo_total_utilizado_recebimento_mercadoria = tempo_duracao_recebimento_mercadoria.sum()
```

In [ ]:
```
tempo_total_utilizado_recebimento_mercadoria
```

In [ ]:
```
# Usando uma variável para armazernar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

In [ ]:
```
tempo_total_parada_recebimento_mercadoria = tempo_total_disponivel_recebimento_mercadoria - tempo_total_utilizado_recebimento_mercador
```

In [ ]:
```
tempo_total_parada_recebimento_mercadoria
```

In [ ]:
```
# Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

In [ ]:
```
qtde_paradas_recebimento_mercadorias = tempo_duracao_recebimento_mercadoria.count() - 1
```

In [ ]:
```
qtde_paradas_recebimento_mercadorias
```

In [ ]:
```
# Calculando o MTTF das Colunas de Recebimento
```

In [ ]:
```
mttf_recebimento_mercadoria = tempo_duracao_recebimento_mercadoria.mean()
```

In [ ]:
```
mttf_recebimento_mercadoria
```

In [ ]:
```
# Calculando o MTTR das Colunas de Recebimento
```

In [ ]:
```
mttr_recebimento_mercadoria = tempo_total_parada_recebimento_mercadoria / qtde_paradas_recebimento_mercadorias
```

In [ ]:
```
mttr_recebimento_mercadoria
```

In [ ]:
```
# Calculando o MTBF das Colunas de Recebimento
```

In [ ]:
```
mtbf_recebimento_mercadoria = (tempo_total_disponivel_recebimento_mercadoria - tempo_total_parada_recebimento_mercadoria) / qtde_parad
```

In [ ]:
```
mtbf_recebimento_mercadoria
```

In [ ]:
```
# Calculando a Disponibilidade das Colunas de Recebimento
```

In [ ]:
```
disponibilidade_recebimento_mercadoria = mtbf_recebimento_mercadoria / (mttr_recebimento_mercadoria + mtbf_recebimento_mercadoria)
```

In [ ]:
```
disponibilidade_recebimento_mercadoria
```

In [ ]:
```
#
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Pré Triagem
#
```

In [ ]:
```
# Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_pre_triagem)
```

In [ ]:
```
df.sort_values(by=['tempo_inicio_pre_triagem'])
```

In [ ]:
```
tempo_inicio_pre_triagem = df['tempo_inicio_pre_triagem']
```

In [ ]:
```
tempo_fim_pre_triagem = df['tempo_fim_pre_triagem']
```

In [ ]:
```
tempo_inicio_pre_triagem = pd.to_datetime(tempo_inicio_pre_triagem)
```

In [ ]:
```
tempo_fim_pre_triagem = pd.to_datetime(tempo_fim_pre_triagem)
```

In [ ]:
```
# Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
# O uso da operação "+ pd.Timedelta("1 days")" é para corrigir erro do valor negativo.
```

In [ ]:
```
tempo_total_disponivel_pre_triagem = (tempo_fim_pre_triagem[15] - tempo_inicio_pre_triagem[1]) + pd.Timedelta("1 days")
```

In [ ]:
```
tempo_total_disponivel_pre_triagem
```

In [ ]:
```
tempo_duracao_pre_triagem = tempo_fim_pre_triagem - tempo_inicio_pre_triagem
```

In [ ]:
```
tempo_duracao_pre_triagem
```

In [ ]:
```
# Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

In [ ]:
```
tempo_total_utilizado_pre_triagem = tempo_duracao_pre_triagem.sum() + pd.Timedelta("1 days")
```

In [ ]:
```
tempo_total_utilizado_pre_triagem
```

In [ ]:
```
# Usando uma variável para armazernar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

In [ ]:
```
tempo_total_parada_pre_triagem = tempo_total_disponivel_pre_triagem - tempo_total_utilizado_pre_triagem
```

In [ ]:
```
tempo_total_parada_pre_triagem
```

In [ ]:
```
# Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

In [ ]:
```
qtde_paradas_pre_triagem = tempo_duracao_pre_triagem.count() - 1
```

In [ ]:
```
qtde_paradas_pre_triagem
```

In [ ]:
```
# Calculando o MTTF das Colunas de Pré Triagem
```

In [ ]:
```
# Correção do item 15 da nossa lista (-1 days +00:05:00).
# A correção é necessária para desprezar o valor de "-1 days" do resultado. Correto é "00:05:00"
# Observação-1: a multiplicação por -1 (*-1) não resolvia o problema, por que estamos trabalhando com tipo data/hora
# Observação-2: esta solução deve ser executada uma única vez. Caso necessário uma segunda ou mais execuções
#               será necessário "limpar todas as saídas" e rexecutar todos os calculos novamente.
```

In [ ]:
```
x_tempo_duracao_pre_triagem = tempo_duracao_pre_triagem
```

In [ ]:
```
x_tempo_duracao_pre_triagem[15] = x_tempo_duracao_pre_triagem[15] + pd.Timedelta("1 days")
```

In [ ]:
```
mttf_pre_triagem = x_tempo_duracao_pre_triagem.mean()
```

In [ ]:
```
mttf_pre_triagem
```

In [ ]:
```
# Calculando o MTTR das Colunas de Pré Triagem
```

In [ ]:
```
mttr_pre_triagem = tempo_total_parada_pre_triagem / qtde_paradas_pre_triagem
```

In [ ]:
```
mttr_pre_triagem
```

In [ ]:
```
# Calculando o MTBF das Colunas de Pré Triagem
```

In [ ]:
```
mtbf_pre_triagem = (tempo_total_disponivel_pre_triagem - tempo_total_parada_pre_triagem) / qtde_paradas_pre_triagem
```

In [ ]:
```
mtbf_pre_triagem
```

In [ ]:
```
# Calculando a Disponibilidade das Colunas de Pré Triagem
```

In [ ]:
```
disponibilidade_pre_triagem = mtbf_pre_triagem / (mttr_pre_triagem + mtbf_pre_triagem)
```

In [ ]:
```
disponibilidade_pre_triagem
```

In [ ]:
```
#
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Raio X
#
```

In [ ]:
```
# Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_raio_x)
```

In [ ]:
```
df.sort_values(by=['tempo_inicio_raio_x'])
```

In [ ]:
```
tempo_inicio_raio_x = df['tempo_inicio_raio_x']
```

In [ ]:
```
tempo_fim_raio_x = df['tempo_fim_raio_x']
```

In [ ]:
```
tempo_inicio_raio_x = pd.to_datetime(tempo_inicio_raio_x)
```

In [ ]:
```
tempo_fim_raio_x = pd.to_datetime(tempo_fim_raio_x)
```

In [ ]:
```
# Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
```

In [ ]:
```
tempo_total_disponivel_raio_x = (tempo_fim_raio_x[4] - tempo_inicio_raio_x[27]) + pd.Timedelta("1 days")
```

In [ ]:
```
tempo_total_disponivel_raio_x
```

In [ ]:
```
tempo_duracao_raio_x = tempo_fim_raio_x - tempo_inicio_raio_x
```

In [ ]:
```
# Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

In [ ]:
```
tempo_total_utilizado_raio_x = tempo_duracao_raio_x.sum()
```

In [ ]:
```
tempo_total_utilizado_raio_x
```

In [ ]:
```
# Usando uma variável para armazernar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

In [ ]:
```
tempo_total_parada_raio_x = tempo_total_disponivel_raio_x - tempo_total_utilizado_raio_x
```

In [ ]:
```
tempo_total_parada_raio_x
```

In [ ]:
```
# Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

In [ ]:
```
qtde_paradas_raio_x = tempo_duracao_raio_x.count() - 1
```

In [ ]:
```
qtde_paradas_raio_x
```

In [ ]:
```
# Calculando o MTTF das Colunas de Raio X
```

In [ ]:
```
mttf_raio_x = tempo_duracao_raio_x.mean()
```

In [ ]:
```
mttf_raio_x
```

In [ ]:
```
# Calculando o MTTR das Colunas de Raio X
```

In [ ]:
```
mttr_raio_x = tempo_total_parada_raio_x / qtde_paradas_raio_x
```

In [ ]:
```
mttr_raio_x
```

In [ ]:
```
# Calculando o MTBF das Colunas de Raio X
```

In [ ]:
```
mtbf_raio_x = (tempo_total_disponivel_raio_x - tempo_total_parada_raio_x) / qtde_paradas_raio_x
```

In [ ]:
```
mtbf_raio_x
```

In [ ]:
```
# Calculando a Disponibilidade das Colunas de Raio X
```

In [ ]:
```
disponibilidade_raio_x = mtbf_raio_x / (mttr_raio_x + mtbf_raio_x)
```

In [ ]:
```
disponibilidade_raio_x
```

In [ ]:
```

```

In [1]:

```
# Carregando Pandas para uso nos calculos das planilhas
import pandas as pd
from datetime import datetime
# Carregando arquivo.csv em memória (para manipulaçaõ)
# Em meu caso, eu estou modificando o "df = pandas.read_csv('../Database/dataset.csv')"
# Eu estou utilizando uma função lambda para usar um parse no campo datas
dateparse = lambda x: datetime.strptime(x, '%H:%M:%S')
df = pd.read_csv('../Database/dataset.csv', parse_dates=True, date_parser=dateparse)
df.head(31)
```

Out[1]:

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
5	04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
7	22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
15	01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
25	04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02

In [2]:
```
#
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Recebimento de Mercadoria
#
```

In [3]:
```
# Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_recebimento_mercad
```

In [4]:
```
df.sort_values(by=['tempo_inicio_recebimento_mercadoria'])
```

Out[4]:

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
15	01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
5	04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
25	04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
29	19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34

In [5]:
```
tempo_inicio_recebimento_mercadoria = df['tempo_inicio_recebimento_mercadoria']
```

In [6]:
```
tempo_fim_recebimento_mercadoria = df['tempo_fim_recebimento_mercadoria']
```

In [7]:
```
tempo_inicio_recebimento_mercadoria = pd.to_datetime(tempo_inicio_recebimento_mercadoria)
```

In [8]:
```
tempo_fim_recebimento_mercadoria = pd.to_datetime(tempo_fim_recebimento_mercadoria)
```

In [9]:
```
# Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
```

In [10]:
```
tempo_total_disponivel_recebimento_mercadoria = tempo_fim_recebimento_mercadoria[7] - tempo_inicio_recebimento_mercadoria[15]
```

In [11]:
```
tempo_total_disponivel_recebimento_mercadoria
```

Out[11]:
```
Timedelta('0 days 21:45:38')
```

In [12]:
```
tempo_duracao_recebimento_mercadoria = tempo_fim_recebimento_mercadoria - tempo_inicio_recebimento_mercadoria
```

In [13]:
```
tempo_duracao_recebimento_mercadoria
```

Out[13]:
```
0    0 days 00:14:00
1    0 days 00:14:00
2    0 days 00:09:00
3    0 days 00:02:00
4    0 days 00:08:00
5    0 days 00:03:00
6    0 days 00:14:00
7    0 days 00:03:00
8    0 days 00:04:00
9    0 days 00:02:00
10   0 days 00:13:00
11   0 days 00:00:00
12   0 days 00:12:00
13   0 days 00:10:00
14   0 days 00:13:00
15   0 days 00:04:00
16   0 days 00:07:00
17   0 days 00:11:00
18   0 days 00:03:00
19   0 days 00:08:00
20   0 days 00:13:00
21   0 days 00:15:00
22   0 days 00:07:00
23   0 days 00:11:00
24   0 days 00:05:00
25   0 days 00:04:00
```

```
26   0 days 00:05:00
27   0 days 00:14:00
28   0 days 00:09:00
29   0 days 00:11:00
dtype: timedelta64[ns]
```

In [14]:
```
# Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

In [15]:
```
tempo_total_utilizado_recebimento_mercadoria = tempo_duracao_recebimento_mercadoria.sum()
```

In [16]:
```
tempo_total_utilizado_recebimento_mercadoria
```

Out[16]:
```
Timedelta('0 days 04:08:00')
```

In [17]:
```
# Usando uma variável para armazernar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

In [18]:
```
tempo_total_parada_recebimento_mercadoria = tempo_total_disponivel_recebimento_mercadoria - tempo_total_utilizado_recebimento_mercador
```

In [19]:
```
tempo_total_parada_recebimento_mercadoria
```

Out[19]:
```
Timedelta('0 days 17:37:38')
```

In [20]:
```
# Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

In [21]:
```
qtde_paradas_recebimento_mercadorias = tempo_duracao_recebimento_mercadoria.count() - 1
```

In [22]:
```
qtde_paradas_recebimento_mercadorias
```

Out[22]:
```
29
```

In [23]:
```
# Calculando o MTTF das Colunas de Recebimento
```

In [24]:
```
mttf_recebimento_mercadoria = tempo_duracao_recebimento_mercadoria.mean()
```

In [25]:
```
mttf_recebimento_mercadoria
```

Out[25]:
```
Timedelta('0 days 00:08:16')
```

In [26]:
```
# Calculando o MTTR das Colunas de Recebimento
```

In [27]:
```
mttr_recebimento_mercadoria = tempo_total_parada_recebimento_mercadoria / qtde_paradas_recebimento_mercadorias
```

In [28]:
```
mttr_recebimento_mercadoria
```

Out[28]:
```
Timedelta('0 days 00:36:28.206896551')
```

In [29]:
```
# Calculando o MTBF das Colunas de Recebimento
```

In [30]:
```
mtbf_recebimento_mercadoria = (tempo_total_disponivel_recebimento_mercadoria - tempo_total_parada_recebimento_mercadoria) / qtde_parad
```

In [31]:
```
mtbf_recebimento_mercadoria
```

Out[31]:
```
Timedelta('0 days 00:08:33.103448275')
```

In [32]:
```
# Calculando a Disponibilidade das Colunas de Recebimento
```

In [33]:
```
disponibilidade_recebimento_mercadoria = mtbf_recebimento_mercadoria / (mttr_recebimento_mercadoria + mtbf_recebimento_mercadoria)
```

In [34]:
```
disponibilidade_recebimento_mercadoria
```

Out[34]:
```
0.18994613086859172
```

In [35]:
```
#
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Pré Triagem
#
```

In [36]:
```
# Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_pre_triagem)
```

In [37]:
```
df.sort_values(by=['tempo_inicio_pre_triagem'])
```

Out[37]:

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
25	04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
7	22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
29	19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30
5	04:24:12	04:27:12	21:25:22	21:35:22	06:27:22	06:51:22

In [38]:
```
tempo_inicio_pre_triagem = df['tempo_inicio_pre_triagem']
```

In [39]:
```
tempo_fim_pre_triagem = df['tempo_fim_pre_triagem']
```

In [40]:
```
tempo_inicio_pre_triagem = pd.to_datetime(tempo_inicio_pre_triagem)
```

In [41]:
```
tempo_fim_pre_triagem = pd.to_datetime(tempo_fim_pre_triagem)
```

In [42]:
```
# Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
# O uso da operação "+ pd.Timedelta("1 days")" é para corrigir erro do valor negativo.
```

In [43]:
```
tempo_total_disponivel_pre_triagem = (tempo_fim_pre_triagem[15] - tempo_inicio_pre_triagem[1]) + pd.Timedelta("1 days")
```

In [44]:
```
tempo_total_disponivel_pre_triagem
```

Out[44]:
```
Timedelta('0 days 22:50:22')
```

In [45]:
```
tempo_duracao_pre_triagem = tempo_fim_pre_triagem - tempo_inicio_pre_triagem
```

In [46]:
```
tempo_duracao_pre_triagem
```

Out[46]:
```
0      0 days 00:04:00
```

```
1      0 days 00:07:00
2      0 days 00:06:00
3      0 days 00:09:00
4      0 days 00:04:00
5      0 days 00:10:00
6      0 days 00:10:00
7      0 days 00:11:00
8      0 days 00:13:00
9      0 days 00:03:00
10     0 days 00:15:00
11     0 days 00:12:00
12     0 days 00:12:00
13     0 days 00:05:00
14     0 days 00:07:00
15   -1 days +00:05:00
16     0 days 00:07:00
17     0 days 00:05:00
18     0 days 00:07:00
19     0 days 00:10:00
20     0 days 00:08:00
21     0 days 00:06:00
22     0 days 00:10:00
23     0 days 00:15:00
24     0 days 00:14:00
25     0 days 00:03:00
26     0 days 00:14:00
27     0 days 00:10:00
28     0 days 00:14:00
29     0 days 00:12:00
dtype: timedelta64[ns]
```

In [47]:
```
# Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

In [48]:
```
tempo_total_utilizado_pre_triagem = tempo_duracao_pre_triagem.sum() + pd.Timedelta("1 days")
```

In [49]:
```
tempo_total_utilizado_pre_triagem
```

Out[49]:
```
Timedelta('0 days 04:28:00')
```

In [50]:
```
# Usando uma variável para armazernar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

In [51]:
```
tempo_total_parada_pre_triagem = tempo_total_disponivel_pre_triagem - tempo_total_utilizado_pre_triagem
```

In [52]:
```
tempo_total_parada_pre_triagem
```

Out[52]: Timedelta('0 days 18:22:22')

In [53]:
```
# Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

In [54]:
```
qtde_paradas_pre_triagem = tempo_duracao_pre_triagem.count() - 1
```

In [55]:
```
qtde_paradas_pre_triagem
```

Out[55]: 29

In [56]:
```
# Calculando o MTTF das Colunas de Pré Triagem
```

In [57]:
```
# Correção do item 15 da nossa lista (-1 days +00:05:00).
# A correção é necessária para desprezar o valor de "-1 days" do resultado. Correto é "00:05:00"
# Observação-1: a multiplicação por -1 (*-1) não resolvia o problema, por que estamos trabalhando com tipo data/hora
# Observação-2: esta solução deve ser executada uma única vez. Caso necessário uma segunda ou mais execuções
#               será necessário "limpar todas as saídas" e rexecutar todos os calculos novamente.
```

In [58]:
```
x_tempo_duracao_pre_triagem = tempo_duracao_pre_triagem
```

In [59]:
```
x_tempo_duracao_pre_triagem[15] = x_tempo_duracao_pre_triagem[15] + pd.Timedelta("1 days")
```

In [60]:
```
mttf_pre_triagem = x_tempo_duracao_pre_triagem.mean()
```

In [61]:
```
mttf_pre_triagem
```

Out[61]: Timedelta('0 days 00:08:56')

In [62]:
```
# Calculando o MTTR das Colunas de Pré Triagem
```

In [63]:
```
mttr_pre_triagem = tempo_total_parada_pre_triagem / qtde_paradas_pre_triagem
```

In [64]:
```
mttr_pre_triagem
```

Out[64]:
```
Timedelta('0 days 00:38:00.758620689')
```

In [65]:
```
# Calculando o MTBF das Colunas de Pré Triagem
```

In [66]:
```
mtbf_pre_triagem = (tempo_total_disponivel_pre_triagem - tempo_total_parada_pre_triagem) / qtde_paradas_pre_triagem
```

In [67]:
```
mtbf_pre_triagem
```

Out[67]:
```
Timedelta('0 days 00:09:14.482758620')
```

In [68]:
```
# Calculando a Disponibilidade das Colunas de Pré Triagem
```

In [69]:
```
disponibilidade_pre_triagem = mtbf_pre_triagem / (mttr_pre_triagem + mtbf_pre_triagem)
```

In [70]:
```
disponibilidade_pre_triagem
```

Out[70]:
```
0.19556809612983905
```

In [71]:
```
#
# Etapas para tratar os dados e realizar os cálculos - MTTF, MTTR, MTBF e Disponibilidade - Tempo de Raio X
#
```

In [72]:
```
# Primeiramente, iremos Ordenar o valores de recebimento, para criar uma sequência de dados crescente (tempo_inicio_raio_x)
```

In [73]:
```
df.sort_values(by=['tempo_inicio_raio_x'])
```

Out[73]:

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
4	14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
27	16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
23	18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
15	01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
18	21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
26	16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
19	09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
10	12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
2	03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
5	04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
6	04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
24	04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
8	13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
16	22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
7	22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
9	14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
14	07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
17	12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
28	10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
20	03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
22	18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
3	14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
0	05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
1	07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
11	10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
13	17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11

	tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
12	03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
21	17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
29	19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30

In [74]:
```
tempo_inicio_raio_x = df['tempo_inicio_raio_x']
```

In [75]:
```
tempo_fim_raio_x = df['tempo_fim_raio_x']
```

In [76]:
```
tempo_inicio_raio_x = pd.to_datetime(tempo_inicio_raio_x)
```

In [77]:
```
tempo_fim_raio_x = pd.to_datetime(tempo_fim_raio_x)
```

In [78]:
```
# Usando uma variável para armazenar o valores de Tempo Total Disponível (1ª tempo inicial - último tempo final)
# Com a subtração acima, é possível ter um valor de tempo disponível
```

In [79]:
```
tempo_total_disponivel_raio_x = (tempo_fim_raio_x[4] - tempo_inicio_raio_x[27]) + pd.Timedelta("1 days")
```

In [80]:
```
tempo_total_disponivel_raio_x
```

Out[80]:
```
Timedelta('0 days 21:39:49')
```

In [81]:
```
tempo_duracao_raio_x = tempo_fim_raio_x - tempo_inicio_raio_x
```

In [82]:
```
# Usando uma variável para armazenar Tempo Total Utilizado (somando todos os valores da sequência de tempo de duração)
# Com esta operação, é possível ter o valor final de tempo de atividade (produção)
```

In [83]:
```
tempo_total_utilizado_raio_x = tempo_duracao_raio_x.sum()
```

In [84]:
```
tempo_total_utilizado_raio_x
```

Out[84]:
```
Timedelta('0 days 04:06:00')
```

In [85]:
```
# Usando uma variável para armazernar o Tempo Total de Paradas (subtraindo do Tempo Total Disponível o Tempo Total Utilizado)
```

In [86]:
```
tempo_total_parada_raio_x = tempo_total_disponivel_raio_x - tempo_total_utilizado_raio_x
```

In [87]:
```
tempo_total_parada_raio_x
```

Out[87]:
```
Timedelta('0 days 17:33:49')
```

In [88]:
```
# Usando uma variável para armazenar as ocorrências de paradas (quantidade de registro -1, nos mostra todas as paradas da série)
```

In [89]:
```
qtde_paradas_raio_x = tempo_duracao_raio_x.count() - 1
```

In [90]:
```
qtde_paradas_raio_x
```

Out[90]:
```
29
```

In [91]:
```
# Calculando o MTTF das Colunas de Raio X
```

In [92]:
```
mttf_raio_x = tempo_duracao_raio_x.mean()
```

In [93]:
```
mttf_raio_x
```

Out[93]:
```
Timedelta('0 days 00:08:12')
```

In [94]:
```
# Calculando o MTTR das Colunas de Raio X
```

In [95]:
```
mttr_raio_x = tempo_total_parada_raio_x / qtde_paradas_raio_x
```

In [96]:
```
mttr_raio_x
```

Out[96]:
```
Timedelta('0 days 00:36:20.310344827')
```

In [97]:
```
# Calculando o MTBF das Colunas de Raio X
```

In [98]:
```
mtbf_raio_x = (tempo_total_disponivel_raio_x - tempo_total_parada_raio_x) / qtde_paradas_raio_x
```

In [99]:
```
mtbf_raio_x
```

Out[99]:
```
Timedelta('0 days 00:08:28.965517241')
```

In [1...
```
# Calculando a Disponibilidade das Colunas de Raio X
```

In [1...
```
disponibilidade_raio_x = mtbf_raio_x / (mttr_raio_x + mtbf_raio_x)
```

In [1...
```
disponibilidade_raio_x
```

Out[1...
```
0.18925745938522484
```

In [ ]:
```

```

dataset

tempo_inicio_recebimento_mercadoria	tempo_fim_recebimento_mercadoria	tempo_inicio_pre_triagem	tempo_fim_pre_triagem	tempo_inicio_raio_x	tempo_fim_raio_x
05:16:21	05:30:21	18:53:43	18:57:43	16:43:04	16:44:04
07:51:07	08:05:07	01:12:47	01:19:47	17:55:17	17:59:17
03:28:10	03:37:10	16:28:54	16:34:54	05:45:50	05:45:50
14:54:25	14:56:25	17:58:10	18:07:10	14:33:26	14:48:26
14:49:53	14:57:53	06:19:11	06:23:11	00:40:29	00:43:29
04:24:12	04:27:12	21:25:22	21:35:22	06:37:33	06:51:33
04:33:30	04:47:30	02:30:53	02:40:53	08:01:21	08:05:21
22:50:12	22:53:12	13:24:58	13:35:58	10:29:12	10:36:12
13:44:59	13:48:59	09:16:28	09:29:28	09:26:00	09:36:00
14:50:50	14:52:50	13:37:44	13:40:44	10:48:19	10:52:19
12:32:16	12:45:16	15:15:16	15:30:16	05:41:14	05:52:14
10:54:34	10:54:34	16:26:16	16:38:16	20:19:28	20:30:28
03:16:24	03:28:24	13:05:14	13:17:14	22:33:44	22:43:44
17:09:17	17:19:17	13:54:10	13:59:10	20:21:11	20:36:11
07:32:06	07:45:06	12:40:02	12:47:02	11:38:40	11:52:40
01:07:34	01:11:34	23:58:09	00:03:09	05:11:24	05:14:24
22:23:57	22:30:57	09:43:03	09:50:03	09:33:34	09:38:34
12:28:29	12:39:29	14:01:29	14:06:29	12:05:57	12:11:57
21:55:29	21:58:29	07:52:06	07:59:06	05:16:42	05:26:42
09:38:08	09:46:08	20:29:04	20:39:04	05:29:26	05:40:26
03:53:04	04:06:04	16:15:00	16:23:00	13:11:08	13:23:08
17:33:54	17:48:54	15:00:34	15:06:34	22:46:14	22:51:14
18:11:42	18:18:42	16:30:01	16:40:01	13:57:02	14:01:02
18:26:26	18:37:26	01:54:48	02:09:48	04:18:59	04:31:59
04:42:27	04:47:27	15:06:39	15:20:39	08:54:14	09:06:14
04:48:58	04:52:58	04:44:45	04:47:45	23:08:56	23:18:56
16:57:28	17:02:28	08:40:10	08:54:10	05:27:54	05:35:54
16:41:53	16:55:53	16:24:08	16:34:08	03:03:40	03:05:40
10:04:45	10:13:45	18:51:15	19:05:15	13:04:02	13:19:02
19:26:42	19:37:42	20:57:19	21:09:19	23:06:30	23:13:30