Dashboard Hoteleiro em Python: Uma Solução para Revenue Management

Izairton Oliveira de Vasconcelos

Repositório GitHub:
https://github.com/IOVASCON/hotel_dashboard.git

Resumo

Este artigo descreve o desenvolvimento de um **Dashboard Hoteleiro** em Python, focado em **Revenue Management**, utilizando dados sintéticos gerados por um script Python. O projeto visa fornecer uma ferramenta interativa e personalizável para análise de métricas essenciais, como **Taxa de Ocupação**, **ADR Médio (Average Daily Rate)**, **GOP Médio - Gross Operating Profit** e **GOPPAR (Gross Operating Profit per Available Room)**. O dashboard foi construído exclusivamente em Python, com base em uma base de dados fictícia, também gerada por um script Python, demonstrando a viabilidade e a eficácia de soluções open-source para gestão hoteleira.

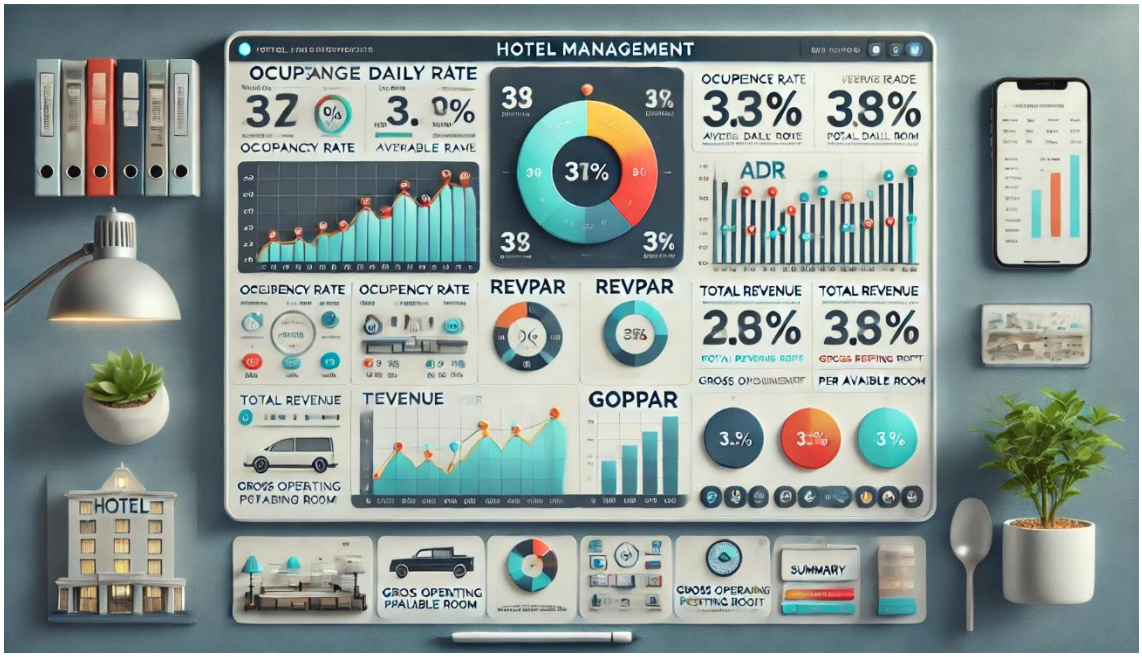

Figura 1: Dashboard desenvolvido em Python para Revenue Management.

Introdução

A gestão hoteleira moderna exige ferramentas robustas para monitorar e otimizar o desempenho financeiro e operacional. O **Revenue Management (Gestão de Receitas)** é uma disciplina crítica nesse contexto, pois permite maximizar a receita por meio da análise de métricas específicas. No entanto, muitas soluções disponíveis no mercado são caras e pouco flexíveis. Este projeto propõe uma alternativa **open-source**, desenvolvida em Python, que combina a geração de dados sintéticos com a criação de um dashboard interativo para análise de métricas.

Objetivo do Projeto

O objetivo principal foi desenvolver um **Dashboard Hoteleiro** em Python, utilizando dados sintéticos gerados por um script Python. O dashboard permite:

1. **Visualização de Métricas:** Exibir métricas de Revenue Management em tempo real.
2. **Filtros Dinâmicos:** Filtrar dados por ano, mês e intervalo de datas.
3. **Gráficos Interativos:** Incluir gráficos de linhas, rosca e barras de progresso para análise visual.
4. **Tomada de Decisão:** Facilitar a análise de desempenho e a identificação de oportunidades de melhoria.

Estrutura do Projeto

O projeto foi organizado em módulos e arquivos Python, seguindo boas práticas de desenvolvimento. A estrutura é a seguinte:

1. Geração de Dados Sintéticos

A base de dados foi gerada por um script Python (**projeto_csv_super**), que simula dados de um hotel fictício chamado **Hotel Luxo**. O script gera um arquivo CSV com informações diárias sobre ocupação, receita, custos e lucros. Essa base de dados é essencial para alimentar o dashboard.

O repositório do projeto pode ser encontrado no GitHub: https://github.com/IOVASCON/projeto_csv_super.git

Uma das grandes vantagens do **script de geração de dados** desenvolvido é sua **maleabilidade e capacidade de adaptação** para diferentes cenários e necessidades analíticas. Seja para um pequeno hotel, uma rede hoteleira de grande porte ou até mesmo para outros segmentos empresariais, a ferramenta permite **gerar**

dados sem restrições, combinando métricas e períodos conforme a necessidade do usuário.

O script não impõe **limites rígidos** à quantidade de registros gerados, permitindo que o usuário:

✅ Simule **anos ou décadas de dados** em uma única execução.
✅ Gere **milhares ou milhões de registros** para testar dashboards em diferentes escalas.
✅ Crie **diferentes granularidades de análise**, desde um único hotel até um conjunto de múltiplas unidades.

O usuário pode definir livremente o período de análise, escolhendo:

- Dados diários, semanais, mensais ou anuais;
- Intervalos curtos, como **um mês específico**, ou períodos longos, como **décadas inteiras** (exemplo: **1980-2024**);
- Simulação de tendências e sazonalidades ao longo dos anos, permitindo prever impactos financeiros e operacionais.

2. Dashboard Hoteleiro

O dashboard foi desenvolvido utilizando as seguintes tecnologias e bibliotecas:

- **Dash:** Framework para criação de dashboards interativos.
- **Dash Bootstrap Components (DBC):** Componentes estilizados para Dash.
- **Pandas:** Manipulação e processamento de dados.
- **Plotly:** Criação de gráficos interativos.
- **Babel:** Formatação de valores monetários.

A estrutura do dashboard é modular, com os seguintes arquivos principais:

1. app.py

- **Função Principal:** Define a estrutura do dashboard e a lógica de filtragem de dados.
- **Funções de Layout:** Cria os componentes visuais, como filtros, cards, gráficos e tabelas.
- **Callbacks:** Atualiza o conteúdo do dashboard dinamicamente com base nas interações do usuário.

2. components/

- **card.py:** Cria cards de métricas (ex.: Receita Total, Ocupação Média).

- **progress_list.py:** Cria barras de progresso para metas (ex.: Taxa de Ocupação, RevPAR).
- **table.py:** Cria tabelas interativas (ex.: Reservas Recentes).

3. **data_processing/process_data.py**

- **Função calculate_metrics:** Calcula métricas de Revenue Management (Ocupação, ADR, RevPAR, etc.) e as adiciona ao DataFrame.
- **Função prepare_and_filter_data:** Filtra os dados com base em ano, mês e intervalo de datas.

4. **visualizations/charts.py**

- **Funções de Gráficos:** Cria gráficos de linhas, rosca e outros visuais para análise de dados.

5. **data/hotel_luxo_jan2000_dez2024.csv**

- **Base de Dados:** Arquivo CSV com dados sintéticos do Hotel Luxo.

Funcionalidades do Dashboard

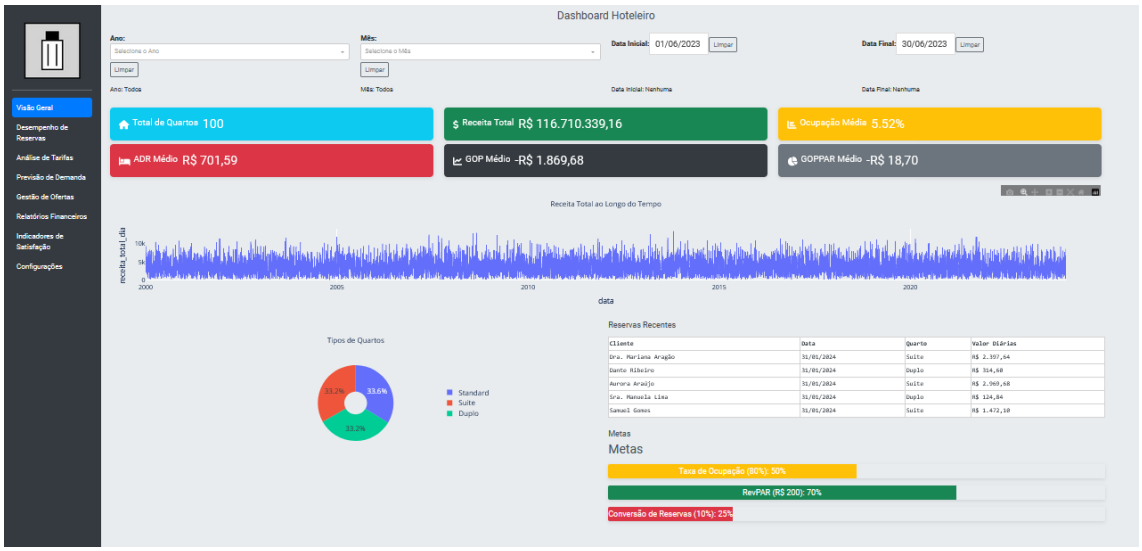

Figura 2: Visão geral do Dashboard Hoteleiro, com filtros de período e exibição de métricas-chave como Total de Quartos, Receita Total, Ocupação Média, ADR Médio, GOP Médio e GOPPAR Médio, além de visualização do progresso em relação às metas de Taxa de Ocupação, RevPAR e Conversão de Reservas.

1. **Filtros Dinâmicos**
   - **Ano, Mês e Intervalo de Datas:** Permite ao usuário selecionar um período específico para análise.
   - **Botões de "Limpar":** Reseta os filtros para visualizar todos os dados.

2. **Cards de Métricas**
   - Exibe métricas agregadas, como **Receita Total**, **Ocupação Média**, **ADR Médio**, **GOP Médio** e **GOPPAR Médio**.

3. **Gráficos Interativos**
   - **Gráfico de Linhas:** Mostra a evolução da Receita Total ao longo do tempo.
   - **Gráfico de Rosca:** Exibe a distribuição dos tipos de quartos ocupados.

4. **Tabela de Reservas Recentes**
   - Lista as últimas reservas, com detalhes como nome do cliente, data e valor das diárias.

5. **Barras de Progresso**
   - Indica o progresso em relação a metas pré-definidas (ex.: Taxa de Ocupação de 80%).

Utilidade do Projeto

O **Dashboard Hoteleiro** desenvolvido neste projeto é uma ferramenta poderosa para gestores hoteleiros, analistas de Revenue Management e estudantes. Ele oferece:

1. **Flexibilidade:** Totalmente personalizável, podendo ser adaptado para diferentes hotéis ou cenários.
2. **Custo Zero:** Solução open-source, sem custos de licenciamento.
3. **Interatividade:** Gráficos e filtros dinâmicos facilitam a análise de dados.
4. **Base para Expansão:** Pode ser integrado com APIs de sistemas hoteleiros para dados em tempo real.

Análise das Métricas de Performance Hoteleira: Interpretação e Impacto na Tomada de Decisão

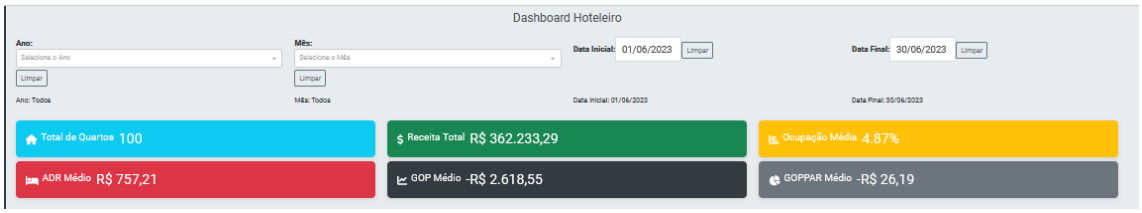

Figura 3: Painel de controle com métricas de desempenho hoteleiro para o período de 01/06/2023 a 30/06/2023, evidenciando a necessidade de otimização da ocupação e resultados operacionais.

A análise de indicadores-chave de desempenho (KPIs) no setor hoteleiro é fundamental para orientar decisões estratégicas e operacionais. As seis métricas exibidas no painel de controle da imagem revelam importantes aspectos sobre a performance do empreendimento, merecendo uma análise aprofundada.

1. Total de Quartos: A Base da Capacidade Hoteleira

Conceito: O total de quartos representa a capacidade máxima de hospedagem do hotel, ou seja, o número absoluto de unidades habitacionais disponíveis para ocupação.

Importância: Embora aparentemente simples, este número é a base para calcular diversas outras métricas de desempenho e serve como limitador superior para a receita potencial do empreendimento.

Análise no Contexto do Dashboard: O valor de 100 quartos indica a escala de operações do hotel, permitindo contextualizar outros indicadores. Por exemplo, uma alta receita total pode ser menos impressionante se o hotel tiver um número significativamente maior de quartos em comparação com concorrentes.

Impacto na Tomada de Decisão:

- **Planejamento Estratégico:** A capacidade total de quartos influencia decisões de expansão ou otimização do mix de unidades (tipos de quartos).
- **Gestão de Receitas:** Este número serve como denominador em métricas como RevPAR e GOPPAR, auxiliando na avaliação da eficiência na geração de receita por unidade disponível.
- **Controle de Custos:** Conhecer a capacidade total permite dimensionar adequadamente os custos operacionais, buscando otimizar a relação custo/quarto disponível.
- **Previsão de Demanda:** Ajuda a modelar a demanda sazonal e planejar estratégias de ocupação em diferentes períodos do ano.

Em resumo, o "Total de Quartos" é um dado fundamental para o planejamento e controle da gestão hoteleira, sendo essencial para a interpretação correta de outros indicadores de desempenho.

2. Receita Total (R$ 362.233,29)

Conceito: Representa o valor total de receitas geradas pelo estabelecimento no período analisado, incluindo hospedagem, alimentos e bebidas, eventos e demais serviços.

Fórmula: Soma de todas as receitas geradas pelo hotel

Análise da Situação: Embora o valor pareça substancial isoladamente, quando analisado em conjunto com a baixa ocupação e os resultados negativos de GOP, revela-se insuficiente para cobrir os custos operacionais do estabelecimento.

Impacto na Tomada de Decisão: Este indicador deve orientar a revisão da estrutura de receitas, buscando oportunidades para diversificar fontes de faturamento além da hospedagem tradicional, como eventos corporativos, pacotes especiais ou serviços complementares.

3. Taxa de Ocupação (4,87%)

Conceito: Representa a porcentagem de quartos ocupados em relação ao total de unidades disponíveis no período analisado.

Fórmula: (Número de Quartos Ocupados ÷ Total de Quartos Disponíveis) × 100

Análise da Situação: O valor de 4,87% indica uma ocupação extremamente baixa, muito abaixo dos parâmetros considerados saudáveis para o setor (que geralmente variam entre 60% e 80%). Esta baixa ocupação é o fator mais preocupante revelado pelo dashboard e provavelmente a principal causa dos resultados financeiros negativos.

Impacto na Tomada de Decisão: Uma taxa de ocupação tão reduzida sinaliza necessidade urgente de revisão das estratégias de marketing, política de preços, canais de distribuição e posicionamento de mercado. Decisões imediatas devem focar em ações para aumentar a demanda e atrair mais hóspedes.

4. ADR Médio (R$ 757,21)

Conceito: Average Daily Rate ou Diária Média - representa o valor médio cobrado por quarto ocupado.

Fórmula: Receita Total de Hospedagem ÷ Número de Quartos Ocupados

Análise da Situação: O valor relativamente elevado de R$ 757,21 sugere um posicionamento de categoria superior do hotel. É um ponto positivo, indicando que os quartos que estão sendo vendidos geram boa receita unitária.

Impacto na Tomada de Decisão: O bom ADR aponta que o problema não está necessariamente no valor cobrado, mas na quantidade de quartos vendidos. Decisões estratégicas devem considerar possíveis ajustes temporários de preço para estimular a ocupação, mantendo o equilíbrio para não desvalorizar o produto.

5. GOP Médio (-R$ 2.618,55)

Conceito: Gross Operating Profit ou Lucro Operacional Bruto Médio - representa o lucro operacional antes de despesas fixas como aluguel, seguros, impostos sobre propriedade etc.

Fórmula: (Receita Total - Despesas Operacionais) ÷ Período de Tempo Analisado

Análise da Situação: O valor negativo de R$ 2.618,55 indica que o hotel está operando com prejuízo significativo, onde as despesas operacionais superam consideravelmente as receitas. Essa situação é insustentável a médio e longo prazo.

Impacto na Tomada de Decisão: Um GOP negativo exige ações corretivas imediatas em duas frentes: aumento de receitas (através de maior ocupação) e redução de custos operacionais, incluindo possível reestruturação de quadro de pessoal, renegociação com fornecedores e otimização de processos.

6. GOPPAR Médio (-R$ 26,19)

Conceito: Gross Operating Profit Per Available Room - representa o lucro operacional bruto por quarto disponível, independentemente de estar ocupado ou não.

Fórmula: GOP Total ÷ Número Total de Quartos Disponíveis no Período

Análise da Situação: O valor negativo de R$ 26,19 indica que cada quarto disponível no hotel está gerando prejuízo operacional, reforçando a gravidade da situação financeira.

Impacto na Tomada de Decisão: Este indicador é particularmente útil para comparar a eficiência operacional entre propriedades similares e períodos diferentes. Seu valor negativo reforça a necessidade de ações corretivas urgentes, podendo justificar decisões mais drásticas como fechamento temporário de andares para reduzir custos operacionais.

7. RevPAR (R$ 36,87)

Conceito: Revenue Per Available Room ou Receita por Quarto Disponível - combina a ocupação e a diária média para medir a eficiência da gestão de receitas.

Fórmula: Receita Total de Hospedagem ÷ Total de Quartos Disponíveis
ou alternativamente: ADR × Taxa de Ocupação (em decimal)

Análise da Situação: O valor baixo de R$ 36,87 resulta da combinação do bom ADR com a baixíssima ocupação, confirmando o diagnóstico de subaproveitamento da capacidade instalada.

Impacto na Tomada de Decisão: O RevPAR é um dos indicadores mais importantes na hotelaria por combinar preço e volume. Seu valor baixo indica necessidade de estratégias que equilibrem melhor esses dois componentes, como revenue management mais agressivo com tarifas dinâmicas.

Conclusão sobre as Métricas Negativas

Os valores negativos do GOP e GOPPAR são consequência direta da combinação entre baixa ocupação (4,87%) e custos operacionais elevados. Mesmo com uma diária média atrativa (R$ 757,21), o volume de quartos vendidos é insuficiente para gerar receita que cubra os custos fixos e variáveis do empreendimento.

Esta situação frequentemente ocorre em períodos de baixa sazonalidade ou em destinos que enfrentam crises de demanda, mas também pode indicar problemas estruturais no modelo de negócio, como dimensionamento inadequado da estrutura operacional ou posicionamento de mercado equivocado.

A reversão deste quadro demandará uma abordagem integrada que combine ações de marketing para aumentar a demanda, estratégias de Revenue Management (Gestão de Receitas) para otimizar preços e ocupação, e medidas de contenção de custos alinhadas com o nível de atividade atual do empreendimento.

Análise das Metas de Desempenho Hoteleiro: Indicadores Essenciais para Gestão Estratégica

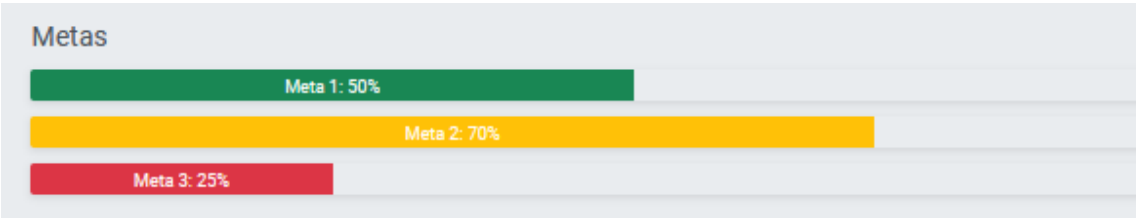

Figura 4: Painel de acompanhamento de metas estratégicas do hotel, ilustrando o progresso em direção aos objetivos de Taxa de Ocupação (Meta 1 - 50%), RevPAR (Meta 2 - 70%) e Taxa de Conversão de Reservas (Meta 3 - 25%). A imagem evidencia a necessidade de ações corretivas para impulsionar a conversão de reservas.

As metas visualizadas no dashboard representam indicadores críticos de performance que orientam a gestão hoteleira para

resultados objetivos. Através de barras de progresso coloridas, estas métricas traduzem visualmente o desempenho atual em relação aos objetivos estabelecidos, permitindo diagnósticos rápidos e ações corretivas.

Meta 1: Taxa de Ocupação (50%) – Barra Verde

Conceito: A taxa de ocupação mede a porcentagem de quartos ocupados em relação ao total disponível, sendo um dos indicadores mais tradicionais e fundamentais da hotelaria.

Análise da Situação: O progresso de 50% em direção à meta estabelecida indica um cenário de desempenho intermediário. Considerando que a ocupação real reportada no dashboard é de apenas 4,87%, podemos inferir que a meta total foi estabelecida em aproximadamente 10-11% para o período analisado, um valor bastante conservador.

Impacto na Tomada de Decisão: Este indicador sinaliza que, mesmo com expectativas modestas para o período, o hotel está conseguindo atingir apenas metade do objetivo de ocupação. A coloração verde da barra sugere que, dentro do contexto sazonal, este desempenho está dentro de parâmetros aceitáveis, porém ainda demanda atenções para atingimento completo.
Aplicação Estratégica: Gestores devem focar em estratégias de marketing direcionadas, promoções específicas para o período e análise dos canais de distribuição para melhorar este indicador.

Meta 2: Revenue per Available Room - RevPAR (70%) – Barra Amarela

Conceito: O RevPAR representa a receita gerada por quarto disponível, combinando ocupação e diária média em um único indicador que mede a eficiência da gestão de receitas do hotel.

Análise da Situação: Com 70% da meta atingida, este é o indicador com melhor desempenho entre os três apresentados. O valor satisfatório desta métrica, mesmo com baixa ocupação, sugere que o hotel está conseguindo manter diárias médias (ADR) relativamente altas, compensando parcialmente a baixa ocupação.

Impacto na Tomada de Decisão: O bom desempenho relativo nesta meta orienta decisões de manutenção da política de preços, indicando que o hotel está conseguindo preservar seu valor percebido mesmo em períodos de baixa demanda.

Aplicação Estratégica: A estratégia de Revenue Management deve continuar priorizando valor sobre volume, com ajustes táticos que não comprometam o posicionamento do estabelecimento.

Meta 3: Taxa de Conversão de Reservas (25%) – Barra Vermelha

Conceito: A taxa de conversão mede a eficiência do funil de vendas, representando a porcentagem de interessados (visitantes do site, solicitações de orçamento etc.) que efetivamente concretizam a reserva.

Análise da Situação: Com apenas 25% da meta atingida, este indicador apresenta o desempenho mais preocupante. A barra vermelha sinaliza que o resultado está significativamente abaixo dos parâmetros considerados aceitáveis, indicando problemas graves no processo de conversão de leads em reservas confirmadas.

Impacto na Tomada de Decisão: Este resultado demanda análise urgente e ações corretivas no processo de vendas, desde a experiência do usuário nos canais digitais até as práticas dos agentes de reserva e central de atendimento.

Aplicação Estratégica: Recomenda-se uma auditoria completa do funil de conversão, identificando pontos de atrito que estejam levando ao abandono do processo de reserva, além de treinamento intensivo para equipes de vendas.

Integração das Metas no Contexto Operacional

A análise conjunta das três metas revela um cenário onde o hotel mantém razoável capacidade de preservar valor (Meta 2 - RevPAR), mas enfrenta dificuldades tanto na atração de volume suficiente de hóspedes (Meta 1 - Ocupação) quanto, principalmente, na conversão de interessados em reservas efetivas (Meta 3 - Conversão).

Este diagnóstico sugere que o empreendimento possui um produto de valor reconhecido pelo mercado, mas enfrenta desafios significativos em seu modelo comercial e processos de vendas, exigindo uma revisão abrangente das estratégias de marketing e canais de distribuição.

A visualização através de barras de progresso oferece vantagens significativas para gestão, como:

- Rápida assimilação do desempenho atual
- Priorização clara de áreas que demandam intervenção
- Comunicação eficiente com equipes multidisciplinares
- Acompanhamento dinâmico da evolução dos indicadores ao longo do tempo

Para maximizar os resultados, recomenda-se a revisão periódica das metas estabelecidas, garantindo que sejam desafiadoras, porém

atingíveis, e que reflitam adequadamente a realidade sazonal e competitiva do mercado em que o empreendimento está inserido.

CONCLUSÃO

Este projeto demonstra a viabilidade de desenvolver soluções robustas e interativas para Revenue Management (Gestão de Receitas) utilizando Python. O **Dashboard Hoteleiro** é uma ferramenta eficaz para análise de métricas essenciais, com potencial para ser expandida e adaptada para diferentes necessidades. A combinação de geração de dados sintéticos e criação de dashboards interativos abre portas para novas aplicações e pesquisas na área de gestão hoteleira.

Embora o estudo de caso apresentado tenha sido desenvolvido com dados sintéticos, **os scripts criados são perfeitamente aplicáveis a situações reais**. A lógica implementada para a geração de dados, o cálculo das métricas financeiras e a estruturação do dashboard seguem os mesmos princípios utilizados em sistemas de **Revenue Management** reais.

A contribuição deste artigo vai além da simples modelagem: ele demonstra que a metodologia utilizada pode ser diretamente aplicada para **hotéis que buscam otimizar sua gestão financeira.** Com pequenas adaptações, os scripts podem ser alimentados com **dados reais** extraídos de sistemas de gestão hoteleira (PMS – Property Management Systems) e usados para:

- **Automação da análise de indicadores-chave (KPIs);**
- **Monitoramento contínuo da rentabilidade do hotel;**
- **Otimização de estratégias de precificação e ocupação;**
- **Redução de custos operacionais através da análise de despesas;**
- **Identificação de padrões de sazonalidade e comportamento de hóspedes.**

Assim, este projeto **não é apenas uma simulação acadêmica**, mas sim uma ferramenta prática que pode ser incorporada a operações hoteleiras reais. **A única mudança necessária é substituir os dados sintéticos pelos dados coletados do hotel**, garantindo que as decisões sejam embasadas em informações precisas e personalizadas para cada negócio.

Com essa abordagem, hotéis podem **transformar seus dados em insights valiosos**, impulsionando a tomada de decisão estratégica e melhorando sua performance financeira de maneira sustentável.

Próximos Passos

1. **Integração com APIs:** Conectar o dashboard a sistemas hoteleiros para dados em tempo real.
2. **Expansão de Métricas:** Adicionar novas métricas e gráficos para análises mais detalhadas.
3. **Interface Gráfica (GUI):** Desenvolver uma interface gráfica para facilitar o uso por não programadores.
4. **Publicação como Pacote Python:** Disponibilizar o projeto como um pacote Python para facilitar a instalação e o uso.

Referências

- **Dash Documentation:** https://dash.plotly.com/
- **Pandas Documentation:** https://pandas.pydata.org/
- **Plotly Documentation:** https://plotly.com/python/

Apêndice: Código-Fonte Completo
A seguir, apresentamos os scripts Python utilizados no projeto.

a) hotel_dashboard\app.py

```
import dash
from dash import dcc, html, dash_table
from dash.dependencies import Input, Output
import pandas as pd
import dash_bootstrap_components as dbc
from babel.numbers import format_currency

# Módulos auxiliares
from components import card, table, progress_list
from utils import helpers
from data_processing import process_data
from visualizations import charts

# ------------------------------------------------
# CONFIGURAÇÕES
# ------------------------------------------------

DATA_FILE = "data/hotel_luxo_jan2000_dez2024.csv"

MENU_ITEMS = [
    {"label": "Visão Geral", "href": "/"},
    {"label": "Desempenho de Reservas", "href": "/desempenho-reservas"},
    {"label": "Análise de Tarifas", "href": "/analise-tarifas"},
    {"label": "Previsão de Demanda", "href": "/previsao-demanda"},
    {"label": "Gestão de Ofertas", "href": "/gestao-ofertas"},
    {"label": "Relatórios Financeiros", "href": "/relatorios-
financeiros"},
    {"label": "Indicadores de Satisfação", "href": "/satisfacao"},
    {"label": "Configurações", "href": "/configuracoes"},
]

def load_data():
    """
    Carrega o DataFrame do arquivo CSV e processa; se falhar, encerra.
    """
    df = process_data.load_and_process_data(DATA_FILE)
    if df is None:
        print("Erro ao carregar os dados. O aplicativo não pode
continuar.")
        exit()
    return df

def calculate_metrics(df):
    """
```

```
    Calcula métricas (Ocupacao, ADR, RevPAR etc.) e as adiciona ao
DataFrame.
    """
    try:
        df['Ocupacao'] =
helpers.calcular_ocupacao(df['quartos_ocupados_dia'],
df['total_quartos'])
        df['ADR'] = helpers.calcular_adr(df['receita_quartos_dia'],
df['quartos_ocupados_dia'])
        df['RevPAR'] = helpers.calcular_revpar(df['ADR'], df['Ocupacao'])
        df['TRevPAR'] = helpers.calcular_trevpar(df['receita_total_dia'],
df['total_quartos'])
        df['GOP'] = df['lucro_operacional_bruto_dia']
        df['GOPPAR'] = df['goppar_dia']
        return df
    except Exception as e:
        print(f"Erro ao calcular as métricas: {e}")
        exit()

def prepare_table_data(df):
    """
    Retorna apenas 5 reservas recentes (nome_cliente, data,
tipo_de_quarto, valor_total_diarias).
    Converte datas e formata valores para BRL.
    """
    data_tabela = df[['nome_cliente', 'data', 'tipo_de_quarto',
'valor_total_diarias']].copy()
    data_tabela['data'] = pd.to_datetime(data_tabela['data'])
    data_tabela = data_tabela.sort_values(by='data', ascending=False)
    data_tabela['data'] = data_tabela['data'].dt.strftime('%d/%m/%Y')
    data_tabela['valor_total_diarias'] =
data_tabela['valor_total_diarias'].apply(
        lambda x: format_currency(x, 'BRL', locale='pt_BR')
    )
    return data_tabela.head(5)

def prepare_donut_chart_data(df):
    """
    Prepara dados para um gráfico de rosca, contando 'tipo_de_quarto'.
    """
    data_rosca = df['tipo_de_quarto'].value_counts().reset_index()
    data_rosca.columns = ['Tipo', 'Quantidade']
    return data_rosca

df = load_data()
df = calculate_metrics(df)

def format_date_br(date_str):
    """
```

```
    Converte data ISO (YYYY-MM-DD) para DD/MM/YYYY. Se None, retorna
'Nenhuma'.
    """
    if date_str:
        try:
            d = pd.to_datetime(date_str)
            return d.strftime('%d/%m/%Y')
        except:
            return 'Formato Inválido'
    return 'Nenhuma'

def create_sidebar():
    """
    Cria o menu lateral (sidebar) fixo à esquerda,
    com altura total da tela (100vh) e um padding básico.
    """
    sidebar = html.Div([
        html.Div([
            html.Img(src=app.get_asset_url('logo.webp'),
                     style={'width': '80%', 'margin': '10px auto',
'display': 'block'}),
            html.Hr(style={'borderColor': '#fff'})
        ], style={'textAlign': 'center'}),

        dbc.Nav(
            [dbc.NavLink(item["label"], href=item["href"],
active="exact") for item in MENU_ITEMS],
            vertical=True,
            pills=True,
        ),
    ],
    className="bg-primary text-white",
    style={
        'height': '100vh',
        'position': 'fixed',
        'top': 0,
        'left': 0,
        'padding': '20px',
        'margin': '0px',
        'width': '220px'  # Aumentei ligeiramente a largura
    })
    return sidebar

def prepare_and_filter_data(df, selected_year=None, selected_month=None,
start_date=None, end_date=None):
    """
    Filtra o DataFrame com base em:
    - Ano (selected_year)
    - Mês (selected_month)
```

```
    - Data Inicial e Data Final (start_date, end_date)
    """
    df_filtered = df.copy()

    if selected_year:
        df_filtered = df_filtered[df_filtered['ano'] == selected_year]
    if selected_month:
        df_filtered = df_filtered[df_filtered['mes'] == selected_month]

    df_filtered['data'] = pd.to_datetime(df_filtered['data'])

    if start_date and end_date:
        start_parsed = pd.to_datetime(start_date)
        end_parsed = pd.to_datetime(end_date)
        df_filtered = df_filtered[(df_filtered['data'] >= start_parsed) &
(df_filtered['data'] <= end_parsed)]

    return df_filtered

def create_main_content(df, selected_year=None, selected_month=None,
start_date=None, end_date=None):
    """
    Cria o conteúdo principal do dashboard.
    Ajustamos tamanhos, margens e paddings para evitar sobreposição
    e deixar o layout mais limpo.
    """
    df_filtered = prepare_and_filter_data(df, selected_year,
selected_month, start_date, end_date)

    # Exemplo de cálculo de métricas
    total_quartos = df_filtered['total_quartos'].iloc[0] if not
df_filtered.empty else 0
    receita_total = df_filtered['receita_total_dia'].sum()
    receita_total_fmt = format_currency(receita_total, 'BRL',
locale='pt_BR')

    ocupacao_media_val = df_filtered['Ocupacao'].mean() if not
df_filtered.empty else 0
    ocupacao_media = f"{ocupacao_media_val:.2f}%"

    adr_medio_val = df_filtered['ADR'].mean() if not df_filtered.empty
else 0
    adr_medio = format_currency(adr_medio_val, 'BRL', locale='pt_BR')

    gop_medio_val = df_filtered['GOP'].mean() if not df_filtered.empty
else 0
    gop_medio = format_currency(gop_medio_val, 'BRL', locale='pt_BR')
```

```
    goppar_medio_val = df_filtered['GOPPAR'].mean() if not
df_filtered.empty else 0
    goppar_medio = format_currency(goppar_medio_val, 'BRL',
locale='pt_BR')

    # Tabela e Gráfico de Rosca
    df_tabela_local = prepare_table_data(df_filtered)
    df_rosca_local = prepare_donut_chart_data(df_filtered)

    rosca_graph = dcc.Graph(
        id='grafico-rosca',
        figure=charts.create_pie_chart(df_rosca_local, names='Tipo',
values='Quantidade', title='Tipos de Quartos'),
        style={'height': '350px'}  # Aumentar a altura
    )

    tabela_reservas = dash_table.DataTable(
        data=df_tabela_local.to_dict('records'),
        columns=[
            {'name': 'Cliente', 'id': 'nome_cliente'},
            {'name': 'Data', 'id': 'data'},
            {'name': 'Quarto', 'id': 'tipo_de_quarto'},
            {'name': 'Valor Diárias', 'id': 'valor_total_diarias'}
        ],
        style_cell={'textAlign': 'left', 'fontSize': '14px'},
        style_header={'backgroundColor': 'white', 'fontWeight': 'bold'}
    )

    # Metas (Progress Bars) - Exemplo
    metas_progress = progress_list.create_progress_list([
        {
            "label": "Taxa de Ocupação (80%)",
            "value": 50,
            "color": "warning",
            "style": {"fontSize": "16px", "height": "25px"}
        },
        {
            "label": "RevPAR (R$ 200)",
            "value": 70,
            "color": "success",
            "style": {"fontSize": "16px", "height": "25px"}
        },
        {
            "label": "Conversão de Reservas (10%)",
            "value": 25,
            "color": "danger",
            "style": {"fontSize": "16px", "height": "25px"}
        }
    ])
```

```
    main_content = html.Div([
        html.H1('Dashboard Hoteleiro', className="text-center mb-4",
style={'fontSize': '24px'}),

        # Linha de Filtros
        dbc.Row([
            # Filtro: Ano
            dbc.Col([
                html.Label("Ano:", style={'fontWeight': 'bold'}),
                dcc.Dropdown(
                    id='ano-dropdown',
                    options=[{'label': ano, 'value': ano} for ano in
sorted(df['ano'].unique())],
                    placeholder="Selecione o Ano",
                    style={'fontSize': '14px'}
                ),
                dbc.Button("Limpar", id='btn-limpar-ano',
color='secondary', outline=True, size="sm", className="mt-1")
            ], md=3),

            # Filtro: Mês
            dbc.Col([
                html.Label("Mês:", style={'fontWeight': 'bold'}),
                dcc.Dropdown(
                    id='mes-dropdown',
                    options=[{'label': mes, 'value': mes} for mes in
sorted(df['mes'].unique())],
                    placeholder="Selecione o Mês",
                    style={'fontSize': '14px'}
                ),
                dbc.Button("Limpar", id='btn-limpar-mes',
color='secondary', outline=True, size="sm", className="mt-1")
            ], md=3),

            # Filtro: Data Inicial
            dbc.Col([
                html.Label("Data Inicial:", style={'fontWeight':
'bold'}),
                dcc.DatePickerSingle(
                    id='start-date',
                    placeholder='Selecione a Data Inicial',
                    display_format='DD/MM/YYYY',
                    persistence=True,
                    persistence_type='local',
                    style={'fontSize': '14px'}
                ),
                dbc.Button("Limpar", id='btn-limpar-start',
color='secondary', outline=True, size="sm", className="mt-1")
```

```
            ], md=3),

            # Filtro: Data Final
            dbc.Col([
                html.Label("Data Final:", style={'fontWeight': 'bold'}),
                dcc.DatePickerSingle(
                    id='end-date',
                    placeholder='Selecione a Data Final',
                    display_format='DD/MM/YYYY',
                    persistence=True,
                    persistence_type='local',
                    style={'fontSize': '14px'}
                ),
                dbc.Button("Limpar", id='btn-limpar-end',
color='secondary', outline=True, size="sm", className="mt-1")
            ], md=3),
        ], className="mb-3"),

        # Texto de Filtros Selecionados
        dbc.Row([
            dbc.Col(html.P(f"Ano: {selected_year if selected_year else
'Todos'}", style={'fontSize': '14px'}), md=3),
            dbc.Col(html.P(f"Mês: {selected_month if selected_month else
'Todos'}", style={'fontSize': '14px'}), md=3),
            dbc.Col(html.P(f"Data Inicial: {format_date_br(start_date)}",
style={'fontSize': '14px'}), md=3),
            dbc.Col(html.P(f"Data Final: {format_date_br(end_date)}",
style={'fontSize': '14px'}), md=3)
        ], className="mb-3"),

        # Cards de Resumo (2 Linhas para 6 Cards)
        # Linha 1: "Total de Quartos", "Receita Total", "Ocupação Média"
        dbc.Row([
            dbc.Col(card.create_card("Total de Quartos", total_quartos,
color="info", icon="fas fa-home"), md=4),
            dbc.Col(card.create_card("Receita Total", receita_total_fmt,
color="success", icon="fas fa-dollar-sign"), md=4),
            dbc.Col(card.create_card("Ocupação Média", ocupacao_media,
color="warning", icon="fas fa-chart-bar"), md=4),
        ], className="mb-2"),

        # Linha 2: "ADR Médio", "GOP Médio", "GOPPAR Médio"
        dbc.Row([
            dbc.Col(card.create_card("ADR Médio", adr_medio,
color="danger", icon="fas fa-bed"), md=4),
            dbc.Col(card.create_card("GOP Médio", gop_medio,
color="primary", icon="fas fa-chart-line"), md=4),
            dbc.Col(card.create_card("GOPPAR Médio", goppar_medio,
color="secondary", icon="fas fa-chart-pie"), md=4),
```

```
        ], className="mb-3"),

        # Gráfico de Linhas (Receita total)
        dcc.Graph(
            id='grafico-principal',
            figure=charts.create_line_chart(df_filtered, x='data',
y='receita_total_dia', title='Receita Total ao Longo do Tempo'),
            style={'height': '300px'}
        ),

        # Gráfico de Rosca e Tabela + Metas
        dbc.Row([
            dbc.Col(
                rosca_graph,
                xs=12, sm=12, md=6, lg=6,
                style={'padding': '5px'}
            ),
            dbc.Col([
                html.H3("Reservas Recentes", className="mb-3",
style={'fontSize': '18px'}),
                tabela_reservas,
                html.Br(),
                html.H4("Metas", className="mb-2", style={'fontSize':
'18px'}),
                metas_progress
            ],
                xs=12, sm=12, md=6, lg=6,
                style={'padding': '5px'}
            )
        ], className="mb-3"),
    ], style={'marginLeft': '20px', 'marginRight': '20px', 'marginTop':
'10px'})

    return main_content

app = dash.Dash(
    __name__,
    external_stylesheets=[dbc.themes.BOOTSTRAP,
                          "https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0/css/all.min.css"],
    suppress_callback_exceptions=True
)

def create_sidebar():
    """
    Menu lateral fixo (sidebar) com as opções de MENU_ITEMS.
    Ajustamos a largura (width=220px) para evitar sobreposição.
    """
    sidebar = html.Div([
```

```
        html.Div([
            html.Img(src=app.get_asset_url('logo.webp'), style={'width':
'80%', 'margin': '10px auto', 'display': 'block'}),
            html.Hr(style={'borderColor': '#fff'})
        ], style={'textAlign': 'center'}),
        dbc.Nav(
            [dbc.NavLink(item["label"], href=item["href"],
active="exact") for item in MENU_ITEMS],
            vertical=True,
            pills=True
        ),
    ], className="bg-primary text-white",
    style={
        'height': '100vh',
        'position': 'fixed',
        'top': 0,
        'left': 0,
        'padding': '20px',
        'margin': '0px',
        'width': '220px'
    })
    return sidebar

app.layout = dbc.Container([
    dbc.Row([
        # Menu lateral (2 colunas)
        dbc.Col(
            create_sidebar(),
            width=2,
            style={'padding': '0px', 'margin': '0px'}
        ),
        # Conteúdo principal (10 colunas)
        dbc.Col(
            id='main-content',
            width=10,
            children=create_main_content(df),
            style={'padding': '0px', 'margin-left': '220px'}  # Ajuste p/
não sobrepor o menu fixo
        )
    ], style={'margin': '0px'}),
], fluid=True, style={'margin': '0px', 'padding': '0px'})

# CALLBACKS

@app.callback(
    Output('main-content', 'children'),
    [Input('ano-dropdown', 'value'),
     Input('mes-dropdown', 'value'),
     Input('start-date', 'date'),
```

```
     Input('end-date', 'date')]
)
def update_main_content(selected_year, selected_month, start_date,
end_date):
    """Atualiza o conteúdo principal do dashboard ao mudar qualquer
filtro."""
    return create_main_content(df, selected_year, selected_month,
start_date, end_date)

@app.callback(
    Output('ano-dropdown', 'value'),
    Input('btn-limpar-ano', 'n_clicks')
)
def limpar_ano(n_clicks):
    if n_clicks:
        return None
    return dash.no_update

@app.callback(
    Output('mes-dropdown', 'value'),
    Input('btn-limpar-mes', 'n_clicks')
)
def limpar_mes(n_clicks):
    if n_clicks:
        return None
    return dash.no_update

@app.callback(
    Output('start-date', 'date'),
    Input('btn-limpar-start', 'n_clicks')
)
def limpar_data_inicial(n_clicks):
    if n_clicks:
        return None
    return dash.no_update

@app.callback(
    Output('end-date', 'date'),
    Input('btn-limpar-end', 'n_clicks')
)
def limpar_data_final(n_clicks):
    if n_clicks:
        return None
    return dash.no_update

if __name__ == "__main__":
    app.run_server(debug=True)
```

b) utils\helpers.py

```python
def calcular_ocupacao(quartos_ocupados, total_quartos):
    """
    Calcula a taxa de ocupação.
    """
    try:
        return (quartos_ocupados / total_quartos) * 100
    except ZeroDivisionError:
        return 0  # Retorna 0 se total_quartos for zero

def calcular_adr(receita_quartos_dia, quartos_ocupados_dia):
    """
    Calcula a diária média (ADR).
    """
    try:
        return receita_quartos_dia / quartos_ocupados_dia
    except ZeroDivisionError:
        return 0  # Retorna 0 se quartos_ocupados for zero

def calcular_revpar(adr, ocupacao):
    """
    Calcula a receita por quarto disponível (RevPAR).
    """
    return adr * (ocupacao / 100)

def calcular_trevpar(receita_total, total_quartos):
    """
    Calcula a receita total por quarto disponível (TRevPAR).
    """
    try:
        return receita_total / total_quartos
    except ZeroDivisionError:
        return 0  # Retorna 0 se total_quartos for zero

def calcular_gop(receita_total, custos_operacionais):
    """
    Calcula o lucro operacional bruto (GOP).
    """
    return receita_total - custos_operacionais

def calcular_goppar(gop, total_quartos):
    """
    Calcula o lucro operacional bruto por quarto disponível (GOPPAR).
    """
    try:
        return gop / total_quartos
    except ZeroDivisionError:
        return 0  # Retorna 0 se total_quartos for zero
```

c) data_processing\process_data.py

```python
import pandas as pd

def load_and_process_data(file_path):
    """
    Carrega os dados do arquivo CSV e retorna um DataFrame.
    """
    try:
        df = pd.read_csv(file_path)
        return df
    except FileNotFoundError:
        print(f"Erro: Arquivo '{file_path}' não encontrado.")
        return None
    except Exception as e:
        print(f"Erro ao carregar o arquivo: {e}")
        return None
```

d) componentes\table.py

```python
import dash_bootstrap_components as dbc
from dash import html

def create_table(title, df):
    """
    Cria uma tabela com título e os dados do DataFrame.
    """
    return html.Div([
        html.H3(title, className="mb-3"),
        dbc.Table.from_dataframe(
            df,
            striped=True,
            bordered=False,
            hover=True,
            responsive=True,  # Adiciona responsividade
            className="table-sm shadow"  # Adiciona sombra e tamanho
pequeno
        )
    ])
```

e) componentes\progress_list.py

```python
import dash_bootstrap_components as dbc
from dash import html
```

```
def create_progress_list(items):
    """
    Cria uma lista de barras de progresso.
    """
    return html.Div([
        html.H3("Metas", className="mb-3"),
        *[dbc.Progress(
            value=item["value"],
            color=item["color"],
            label=f'{item["label"]}: {item["value"]}%',
            className="mb-3 shadow"  # Adiciona sombra
        ) for item in items]
    ])
```

f) componentes\card.py

```
import dash_bootstrap_components as dbc
from dash import html

def create_card(title, value, color="primary", icon=None):
    """
    Cria um card com título, valor e, opcionalmente, um ícone.
    """
    card_body_children = []

    # Adiciona o ícone se fornecido
    if icon:
        card_body_children.append(html.I(className=f'{icon} me-2 fa-lg',
style={'color': 'white'}))

    # Adiciona o título e o valor
    card_body_children.extend([
        html.H5(title, className="card-title text-white"),  # Título
menor
        html.H3(value, className="card-text text-white",
style={'textAlign': 'right', 'marginLeft': '10px'})  # Valor menor e
alinhado à direita
    ])

    return dbc.Card(
        dbc.CardBody(card_body_children, className="d-flex align-items-
center"),  # Alinha verticalmente
        color=color,
        inverse=True,
        className="shadow h-100"  # Adiciona sombra e ocupa todo o espaço
vertical
    )
```

g) assets\styles.css

```css
body {
    font-family: 'Roboto', sans-serif;
    background-color: #e9ecef;
    color: #343a40;
}

/* Estilizar o menu lateral */
.bg-primary {
    background-color: #343a40 !important;
}

.text-white {
    color: #fff !important;
}

/* Links do menu lateral */
.nav-link {
    color: #fff !important;
    padding: 10px;
    border-radius: 5px;
}

.nav-link:hover {
    background-color: rgba(255, 255, 255, 0.1);
}

.nav-pills .nav-link.active {
    background-color: #007bff;
}

/* Centralizar título do gráfico */
.plotly .title {
    text-align: center;
}

/* Estilos para os cards */
.card {
    border: none;
    border-radius: 0.5rem;
    box-shadow: 0 0.15rem 0.5rem rgba(0, 0, 0, 0.1) !important;
}

.card-body {
    padding: 1.25rem;
}

/* Estilos para a tabela */
```

```css
.table {
    background-color: #fff;
    border-radius: 0.5rem;
    box-shadow: 0 0.15rem 0.5rem rgba(0, 0, 0, 0.1) !important;
}

.table-sm th,
.table-sm td {
    padding: 0.5rem;
    text-align: left;
}

/* Ajustes de responsividade */
@media (max-width: 768px) {
    .main-content {
        margin-left: 0;
        padding: 10px;
    }

    .bg-primary {
        position: static;
        height: auto;
    }
}

/* Estilos gerais */
.shadow {
    box-shadow: 0 0.15rem 0.5rem rgba(0, 0, 0, 0.1) !important;
}

h1, h2, h3, h4, h5, h6 {
    color: #495057;
}

/* Ajustes adicionais para o layout */
.mb-4 {
    margin-bottom: 1.5rem !important;
}

/* Icones nos cards */
.card-body i {
    vertical-align: middle;
}

/* Estilo para a lista de progresso */
.progress {
    height: 2rem !important; /* Ajuste conforme o tamanho desejado */
    border-radius: 0.25rem;
}
```

```css
/* Aumentar a altura (e alinhar texto) na barra interna */
.progress-bar {
    font-size: 18px !important;
    line-height: 2rem !important; /* para centralizar o texto
verticalmente */
}

/* Estilo para o texto da meta */
.goal-text {
    font-size: 1rem;
    color: #6c757d;
}
```

Siga-me no LinkedIn: www.linkedin.com/comm/mynetwork/discovery-see-all?usecase=PEOPLE_FOLLOWS&followMember=izairton-oliveira-de-vasconcelos-a1916351

Minha Newsletter, o link para assinar:https://www.linkedin.com/build-relation/newsletter-follow?entityUrn=7287106727202742273

https://www.linkedin.com/pulse/dashboard-hoteleiro-em-python-uma-solu%25C3%25A7%25C3%25A3o-para-izairton-6l3tf

https://github.com/IOVASCON/hotel_dashboard.git