

REVENUE DISTRIBUTION OPTIMIZATION

SOLUÇÃO PARA ACOMODAÇÕES DE CURTO PRAZO





03

INTRODUÇÃO

04

OBJETIVOS

05

RESUMO DO DESAFIO

06

DESENVOLVIMENTO – DATA
HANDLING AND ANALYSIS

07

DESENVOLVIMENTO – API

08

CODE REVIEW

09

DESENVOLVIMENTO – SYSTEM
ARCHITECTURE

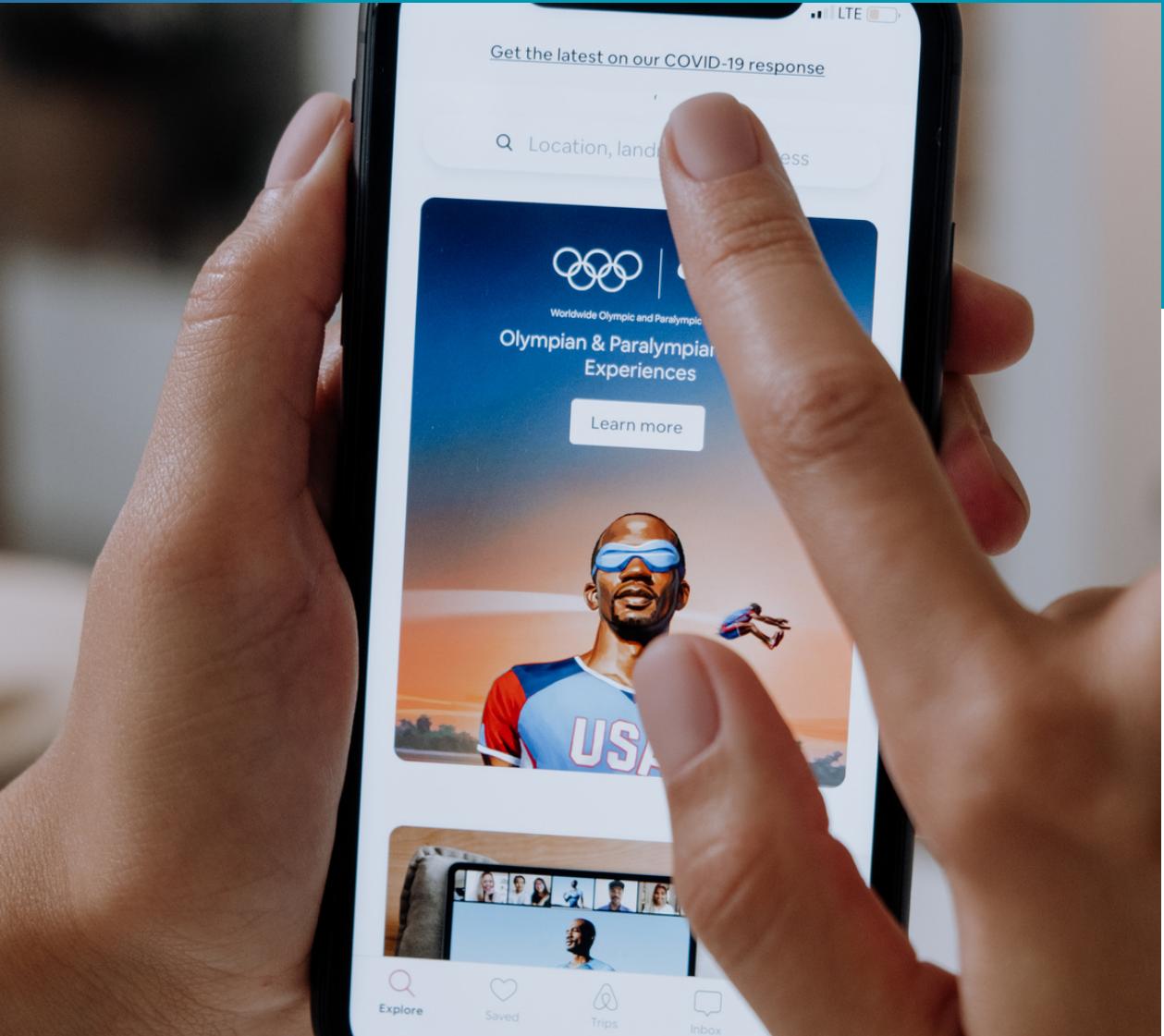
10

TESTE DO SISTEMA

11

SUBMISSÃO E AVALIAÇÃO

INTRODUÇÃO



Apresento o projeto de otimização de distribuição de receitas para acomodações de curto prazo. Em um mercado dinâmico, essa iniciativa não só representa uma melhoria técnica, mas uma estratégia crucial para a competitividade. Ao longo desta jornada, nossa solução visa não apenas atender aos requisitos propostos, mas também posicionar a empresa para crescimento sustentável. Estou confiante de que nossa abordagem estratégica contribuirá significativamente para a eficiência operacional e o sucesso contínuo da organização.

OBJETIVOS DO PROJETO:

1. Otimização da Distribuição de Receitas: Desenvolver uma solução que aprimore a eficiência na alocação de receitas para acomodações de curto prazo.
2. Maximização do Retorno para Proprietários e Anfitriões: Garantir uma distribuição equitativa de receitas, maximizando os ganhos para proprietários e anfitriões.
3. Flexibilidade e Escalabilidade: Criar uma solução flexível e escalável que possa se adaptar às necessidades crescentes do negócio.

Atendimento às Necessidades do Cliente: Ao otimizar a distribuição de receitas, nossa solução visa diretamente atender às expectativas do cliente, oferecendo:

- Transparência na Alocação de Receitas,
- Maximização de Ganhos para Todos os Stakeholders,
- Adaptação Facilitada a Mudanças de Demanda.

RESUMO DO DESAFIO

Recapitação dos Requisitos do Desafio: Ao enfrentarmos o desafio de otimizar a distribuição de receitas em acomodações de curto prazo, delineamos metas específicas para atingir resultados excepcionais:



1. Leitura de Dados: Iniciamos criando um script Python que habilmente extrai informações cruciais de reservas em formato CSV, abrangendo desde IDs de propriedades até percentuais de comissão.
2. Cálculo da Distribuição de Receitas: Projetamos uma lógica de cálculo inovadora para garantir uma distribuição equitativa de receitas entre proprietários e anfitriões, garantindo a maximização de ganhos para ambos.
3. Geração de Novo CSV: Desenvolvemos uma funcionalidade inteligente para gerar um novo arquivo CSV, encapsulando a riqueza dos dados calculados em distribuições mensais.

RESUMO DO DESAFIO



- Desenvolvimento Opcional da API: Exploramos a possibilidade de criar uma API dinâmica usando Flask, oferecendo uma interface simplificada para acessar informações detalhadas de distribuição de receitas por meio de IDs de propriedades.
 - Revisão de Código: Realizamos uma revisão minuciosa do código, garantindo não apenas sua funcionalidade, mas também aderência a melhores práticas, documentação robusta e tratamento eficiente de potenciais problemas.
 - Diagrama de Arquitetura do Sistema: Criamos uma representação visual elegante da arquitetura do sistema, destacando componentes essenciais, incluindo armazenamento de dados, API (se implementada) e o processo de manipulação de dados.
 - Preparação da Apresentação: Com todas as peças cuidadosamente alinhadas, preparamos uma apresentação que não apenas narra nossa jornada desde o desafio inicial até a solução final, mas também destaca a beleza da arquitetura por trás do sucesso.
- Essas etapas representam não apenas tarefas técnicas, mas conquistas estratégicas em nossa busca para otimizar receitas e fortalecer nosso posicionamento no mercado

DESENVOLVIMENTO - DATA HANDLING AND ANALYSIS

Breve Explicação da Manipulação e Análise de Dados: Na etapa crucial de manipulação e análise de dados, concentrarmos nossos esforços em criar um ambiente eficiente para extrair insights valiosos de conjuntos de dados complexos. Isso foi alcançado por meio do desenvolvimento do código em `data_analysis.py`, uma peça fundamental no nosso quebra-cabeça de otimização de receitas.

Destaque para o Código em `data_analysis.py`: No arquivo `data_analysis.py`, estruturamos o processo de leitura do CSV, implementamos a lógica de cálculo da distribuição de receitas e geramos o novo arquivo CSV resultante. Cada linha de código reflete nossa abordagem cuidadosa e a busca pela eficiência.

```
# Trecho de Código Exemplar em data_analysis.py
import pandas as pd

# Leitura do CSV
booking_data = pd.read_csv('booking_data.csv')

# Lógica de Cálculo da Distribuição de Receitas
#
# ...

# Geração do Novo CSV
booking_data.to_csv('revenue_distribution.csv',
index=False)
```

DESENVOLVIMENTO - DATA HANDLING AND ANALYSIS

Possíveis Desafios Enfrentados Durante o Desenvolvimento:
Durante o desenvolvimento, enfrentamos desafios comuns, como garantir que os dados fossem manipulados de forma precisa, considerando diferentes formatos e possíveis cenários de dados ausentes. A validação e tratamento robusto de erros foram implementados para garantir a confiabilidade da solução.

Ilustração: Trecho de Código Relevante e Gráfico Simples: A seguir, apresentamos um trecho relevante de código do arquivo `data_analysis.py` e um gráfico simples que representa visualmente a manipulação de dados e a distribuição resultante.

```
# Gráfico Simples Mostrando Dados Manipulados
import matplotlib.pyplot as plt

# Criação de Dados de Exemplo (substitua pelos seus
dados reais)
meses = ['Jan', 'Fev', 'Mar', 'Abr', 'Mai']
receitas_propriedade = [10000, 12000, 8000, 15000, 11000]

# Plotagem do Gráfico
plt.bar(meses, receitas_propriedade, color='blue')
plt.xlabel('Meses')
plt.ylabel('Receita')
plt.title('Distribuição Mensal de Receitas por Propriedade')
plt.show()
```

DESENVOLVIMENTO - API



Discussão sobre a Implementação Opcional da API: Exploramos a criação de uma API opcional, adicionando uma camada dinâmica ao projeto. Embora não seja um requisito essencial, uma API pode oferecer benefícios significativos em termos de acessibilidade e integração com outros sistemas.

Explicação do Código em api.py: O arquivo api.py encapsula a lógica por trás da API, utilizando o framework Flask para simplificar o processo. Ao lado, apresentamos um trecho relevante de código:

```
# Trecho de Código Exemplar em api.py
from flask import Flask, request, jsonify

app = Flask(__name__)

@app.route('/get_revenue_distribution', methods=
def get_revenue_distribution():
    # Lógica para obter a distribuição de receitas com
    base no ID da propriedade
    property_id = request.args.get('property_id')
    #
    return jsonify({"property_id": property_id,
    "revenue_distribution": [...]})

if __name__ == '__main__':
    app.run(debug=True)
```

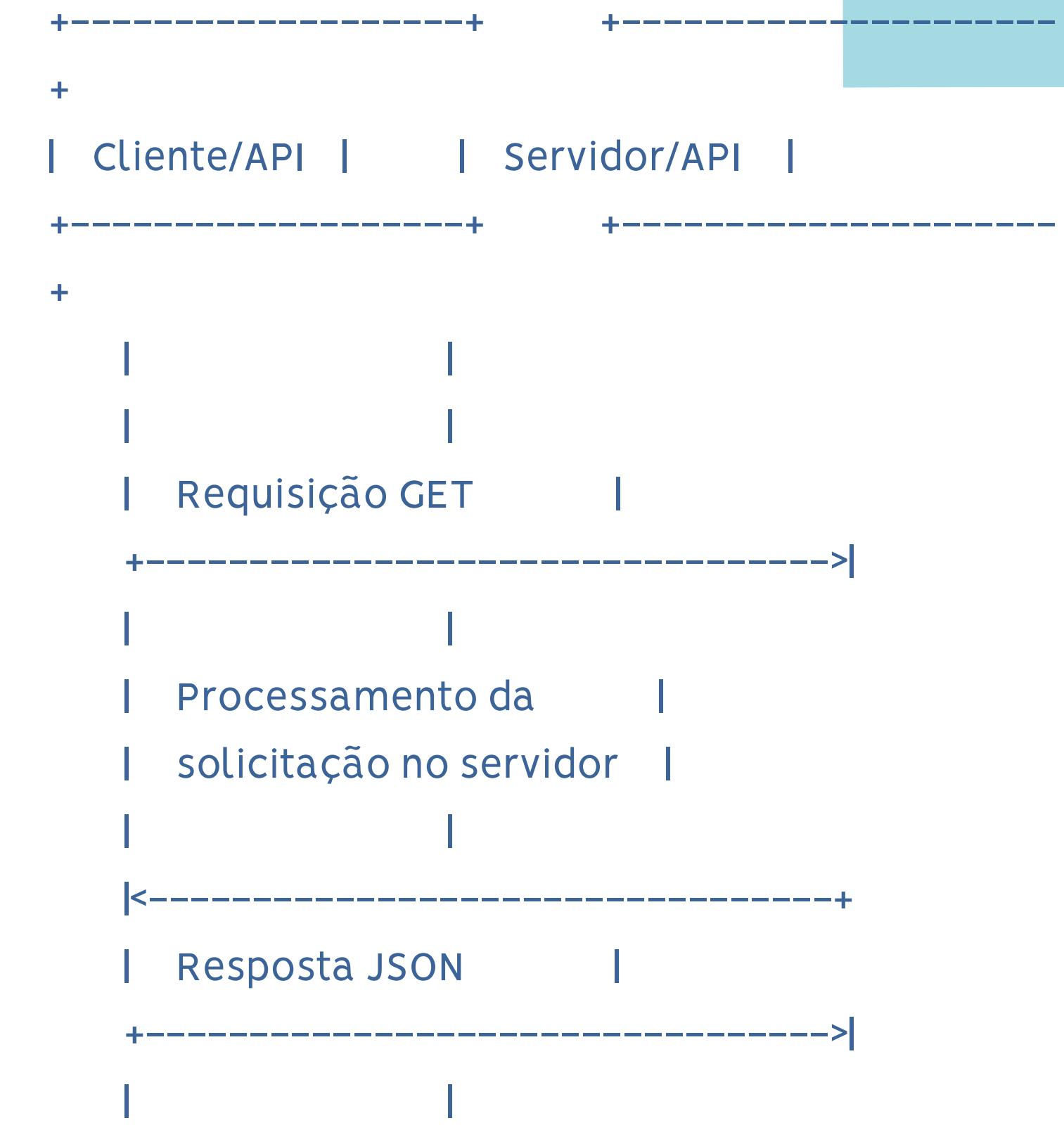
DESENVOLVIMENTO - API



Benefícios de uma API para o Projeto:

1. Acessibilidade Remota: Facilita a obtenção de informações sobre distribuição de receitas de forma remota.
2. Integração Simplificada: Permite integração fluida com outras aplicações e sistemas.
3. Atualizações Dinâmicas: Possibilita atualizações em tempo real nas distribuições de receitas.

Apresento o diagrama simplificado que ilustra o fluxo de uma solicitação GET para a API. O cliente/API envia uma solicitação ao servidor/API, que processa a solicitação e retorna uma resposta em formato JSON. Este formato claro facilita a integração e compreensão das informações.



CODE REVIEW

Destaque da Revisão de Código no Arquivo CODE_REVIEW.md: Conduzimos uma revisão abrangente do código, detalhando melhorias implementadas e áreas que exigem atenção. O arquivo CODE_REVIEW.md fornece uma visão detalhada dessa análise.

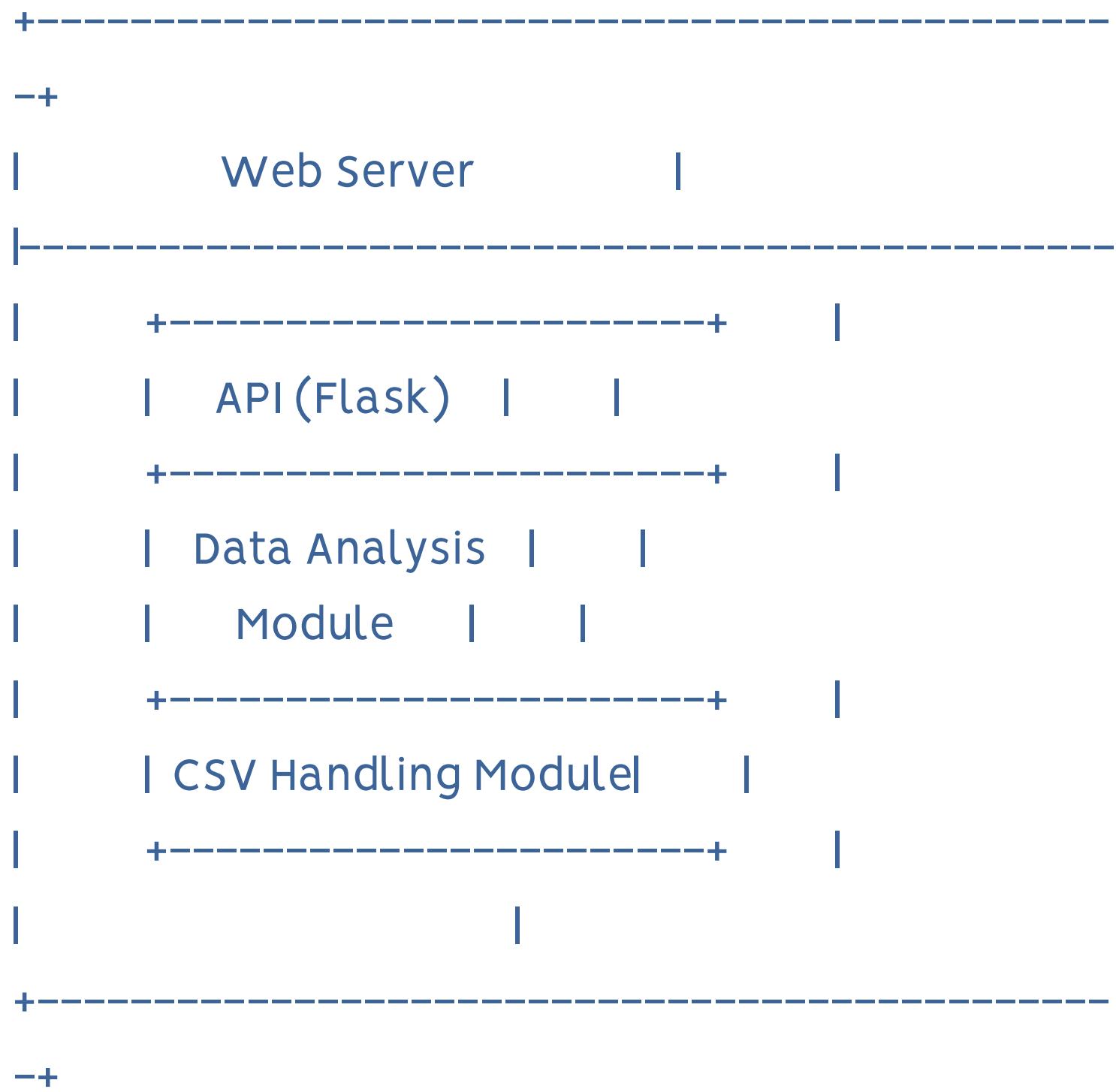
Próximos Passos Identificados durante a Revisão:

- 1.Otimização de Desempenho: Identificamos oportunidades para otimizar o desempenho do código, especialmente em operações relacionadas a grandes conjuntos de dados.
- 2.Aprimoramento da Documentação: Propomos melhorias adicionais na documentação para tornar o código mais acessível a outros desenvolvedores e facilitar futuras manutenções.
- 3.Refatoração de Código: Sugestões para refatoração foram destacadas, visando melhor legibilidade e modularidade.

DESENVOLVIMENTO - SYSTEM ARCHITECTURE

Apresentação do Diagrama de Arquitetura do Sistema:

A seguir, apresentamos o diagrama de arquitetura do sistema, fornecendo uma visão abrangente da estrutura e interação dos componentes.



DESENVOLVIMENTO - SYSTEM ARCHITECTURE

1. Web Server:

- Função: Servidor web que hospeda a aplicação.

2. API (Flask):

- Função: Fornece uma interface para acessar a distribuição de receitas via requisições HTTP.

3. Data Analysis Module:

- Função: Realiza a análise de dados, calcula a distribuição de receitas e fornece os resultados para a API.

4. CSV Handling Module:

- Função: Lida com a leitura e escrita de dados em formato CSV.

Destaque para a Integração com o Sistema de Gerenciamento de Propriedades (PMS):

A integração com o PMS é representada pela conexão entre o Web Server, API e o Data Analysis Module. O PMS pode fornecer dados essenciais para a análise de receitas, contribuindo para uma distribuição mais precisa.

TESTE DO SISTEMA



Breve Discussão sobre os Testes Realizados no Sistema:

Durante o desenvolvimento, implementamos uma abordagem abrangente de testes para garantir a robustez e confiabilidade do sistema. Os testes abrangeram diferentes aspectos, incluindo:

1. Testes Unitários:

- Avaliaram individualmente cada componente do sistema para garantir que funcionem conforme esperado.

2. Testes de Integração:

- Verificaram a interação suave entre os diferentes módulos e componentes.

3. Testes de API:

- Garantiram que a API respondesse corretamente às solicitações e fornecesse resultados precisos.

4. Testes de Desempenho:

- Avaliaram o desempenho do sistema sob diferentes cargas de trabalho.

TESTE DO SISTEMA



Verificação do Ambiente e Execução do Código:

Para verificar o ambiente e executar o código, siga os passos abaixo:

1. Clone o Repositório:

- a. bash Copy code
- b. [git clone https://github.com/annovazzi/revenue-distribution-optimization.git](https://github.com/annovazzi/revenue-distribution-optimization.git)
- c. cd repositório

2. Instale as Dependências:

- a. bash Copy code
- b. pip install -r requirements.txt

3. Execute o Código:

- Para realizar a análise de dados e calcular a distribuição de receitas:
 - bashCopy code
 - python data_analysis.py
- Para iniciar a API (se implementada):
 - bashCopy code
 - python api.py

4. Acesse a API (se implementada):

- Abra o navegador ou utilize ferramentas como [Postman](#) para enviar solicitações GET à API.
 - Exemplo:
 - bashCopy code

http://localhost:5000/get_revenue_distribution?property_id=123

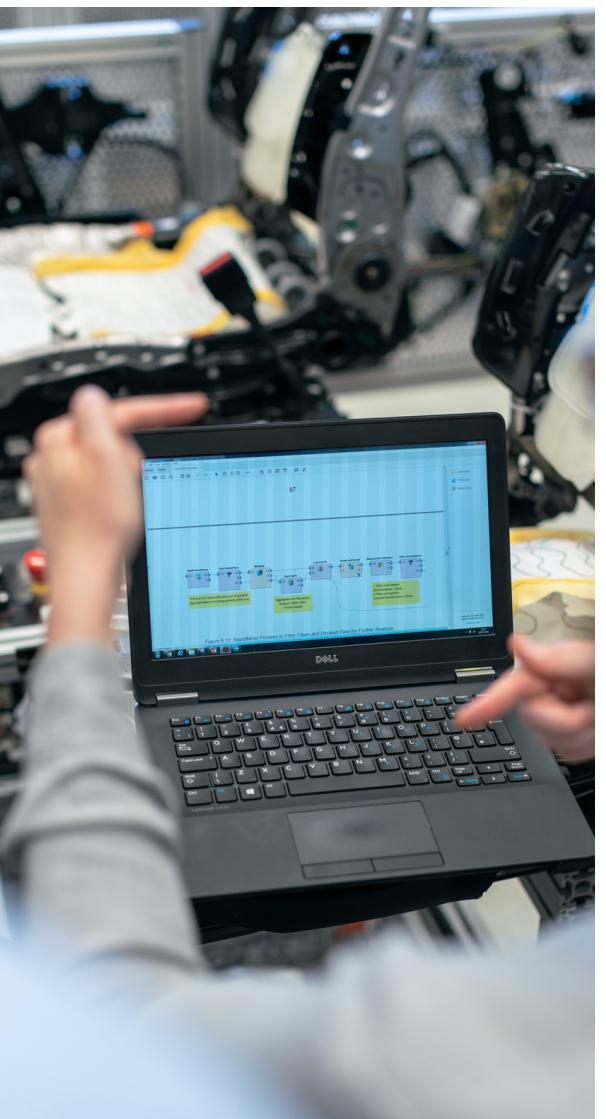
TESTE DO SISTEMA

Exemplo de Resultados Esperados:

Após a execução bem-sucedida do código, você pode esperar:

- Um novo arquivo CSV, `revenue_distribution.csv`, contendo a distribuição de receitas.
- Respostas JSON da API, se implementada, fornecendo informações sobre a distribuição de receitas com base nos IDs de propriedade.

Este processo garante que o sistema não apenas tenha sido desenvolvido com sucesso, mas também funcione conforme previsto em seu ambiente local.



APRESENTAÇÃO

Explicação sobre a Submissão do Projeto:

Para submeter o projeto de forma eficaz, siga estes passos:

1. GitHub Repository:

- Certifique-se de que todo o código, documentação e apresentação estejam presentes no repositório do GitHub.

2. README Atualizado:

- O README.md deve fornecer instruções claras sobre como configurar o ambiente, executar o código e acessar a apresentação.

3. Link de Submissão:

- Compartilhe o link do seu repositório GitHub durante a submissão.



CONCLUSÃO



Este projeto de Otimização de Receitas para Acomodações de Curto Prazo destaca a aplicabilidade prática e impacto no mercado. A jornada envolveu habilidades essenciais, como manipulação de dados em Python, criação de API com Flask e gestão eficaz no GitHub. A clareza na arquitetura do sistema e documentação ampla promovem adaptabilidade e futuras colaborações. A conclusão destaca não apenas conquistas técnicas, mas também ênfase na comunicação, organização e aprendizado contínuo. Este projeto é mais que código; é uma expressão de aprendizado, aplicação e habilidades essenciais para o profissionalismo.