🐾 MVP Chatbot e Análise de Dados para Hotel de Pets

🌟 Visão Geral do Projeto

Este projeto representa um **Produto Mínimo Viável (MVP)** completo de um sistema inteligente para um hotel de pets, combinando **análise de dados**, um **chatbot interativo** e uma **API RESTful** robusta com uma **interface web moderna e profissional**. Desenvolvido como parte de um desafio para Analista de IA & Dados Jr., o objetivo principal é demonstrar a capacidade de extrair insights de dados, automatizar respostas a perguntas de negócio e construir uma solução escalável, testável e visualmente atraente.

O cenário é um hotel de pets fictício em São Paulo, onde a receita provém de estadias e vendas de produtos. A solução proposta visa auxiliar a equipe interna do hotel a tomar decisões mais informadas e a otimizar suas operações através de uma interface moderna e intuitiva.

✨ Funcionalidades Principais

O projeto abrange as seguintes funcionalidades:

1. **Análise de Dados e Insights:**
    - Processamento de dados de vendas, estadias, produtos e métodos de pagamento.
    - Geração de visualizações gráficas para identificar tendências e padrões.
    - Simulação de queries SQL usando Pandas para demonstração.
2. **Chatbot Interativo (MVP):**
    - Capacidade de responder a perguntas de negócio predefinidas.

    - Retorna queries SQL correspondentes que seriam executadas em um banco de dados real.
    - Interface web moderna com chat em tempo real.

3. **API RESTful com Flask:**
    - Encapsula a lógica do chatbot e a simulação de queries em uma API web.
    - Permite a integração com outras aplicações (front-ends, sistemas internos).
    - Endpoints para interação com o chatbot, execução de queries simuladas, listagem de perguntas e health check.
    - Configuração de CORS para permitir requisições de diferentes origens.

4. **Interface Web Moderna:**
    - Design system profissional com paleta de cores suave e tipografia elegante.
    - Chat interativo com animações suaves e feedback visual.
    - Tema claro/escuro com toggle automático.
    - Design totalmente responsivo para desktop, tablet e mobile.
    - Sistema de notificações toast e indicadores de status em tempo real.

5. **Testes Automatizados com Pytest:**
    - Conjunto abrangente de testes unitários e de integração.
    - Garante a confiabilidade, robustez e manutenibilidade do código.
    - Cobre casos de sucesso, falha, validação de entrada e tratamento de erros.

6. **Documentação Completa:**
    - Relatório técnico detalhado com análises e metodologia.
    - Guia de implementação passo a passo.
    - README abrangente para o GitHub.

🛠️ Tecnologias Utilizadas

- **Python 3.11:** Linguagem de programação principal.
- **Flask (v3.0.3):** Microframework web para construção da API RESTful.

- **Flask-CORS (v6.0.1):** Extensão para habilitar Cross-Origin Resource Sharing na API.
- **Pandas (v2.2.2):** Biblioteca para manipulação e análise de dados.
- **OpenPyXL (v3.1.5):** Biblioteca para leitura de arquivos `.xlsx`.
- **Matplotlib & Seaborn:** Bibliotecas para criação de visualizações de dados.
- **Pytest (v8.4.2):** Framework de testes para Python.
- **HTML5, CSS3, JavaScript ES6+:** Tecnologias web modernas para a interface.
- **Poppins Font:** Tipografia moderna e elegante.
- **CSS Custom Properties:** Sistema de design consistente com variáveis CSS.

📂 Estrutura do Projeto

```
. (diretório raiz do projeto)
├── chatbot.py                      # Lógica central do chatbot (versão
standalone)
├── chatbot_api.py                  # Implementação da API RESTful Flask
├── simulador_sql.py                # Script para simular execução de queries
SQL com Pandas
├── test_chatbot.py                 # Testes unitários para a lógica do chatbot
├── test_api.py                     # Testes unitários e de integração para a
API Flask
├── conftest.py                     # Configurações e fixtures globais do
Pytest
├── requirements.txt                # Lista de dependências do Python
├── Conjuntodedados.xlsx            # Conjunto de dados original do hotel de
pets
├── index.html                   # Interface web moderna (versão aprimorada)
├── style.css                    # Estilos CSS modernos e responsivos
├── script_v2.js                    # JavaScript avançado com funcionalidades
completas
└── README.md                 # Este arquivo README
```

🚀 Como Executar o Projeto

Siga os passos abaixo para configurar e executar o projeto em seu ambiente local.

Pré-requisitos

Certifique-se de ter o **Python 3.11** (ou versão compatível) instalado em sua máquina. Você pode verificar sua versão com:

```
python3 --version
```

1. Clonar o Repositório

Primeiro, clone este repositório para sua máquina local:

```
git clone <URL_DO_SEU_REPOSITORIO>
cd <nome_do_diretorio_do_projeto>
```

2. Instalar Dependências

Instale todas as bibliotecas Python necessárias usando o `pip` e o arquivo `requirements.txt`:

```
pip install -r requirements.txt
```

3. Preparar o Arquivo de Dados

Certifique-se de que o arquivo `Conjuntodedados.xlsx` esteja presente no diretório raiz do projeto. Este arquivo é essencial para a execução das simulações de queries e para a API.

4. Executar Componentes do Projeto

a) Interface Web Moderna (Recomendado)

Para a melhor experiência, use a interface web moderna:

1. **Inicie a API Flask:** `bash python chatbot_api.py`
2. **Abra a interface web:** Abra o arquivo `index_v2.html` em qualquer navegador moderno ou use um servidor local: `bash # Usando Python para servir arquivos estáticos python -m http.server 8000 # Acesse http://localhost:8000/index_v2.html`
3. **Interaja com o chatbot:**
4. Use os botões de perguntas rápidas coloridos

5. Digite suas próprias perguntas no campo de input
6. Monitore as estatísticas em tempo real na sidebar
7. Alterne entre tema claro e escuro

b) Chatbot Standalone

Para interagir com a versão básica do chatbot via linha de comando:

```
python chatbot.py
```

c) API RESTful (Flask)

Para testar a API diretamente:

```
python chatbot_api.py
```

A API será executada em `http://0.0.0.0:5000`. Exemplos de uso:

- **Verificar status da API:** `bash curl http://localhost:5000/health`
- **Fazer uma pergunta ao chatbot:** `bash curl -X POST http://localhost:5000/chat \ -H "Content-Type: application/json" \ -d '{"pergunta": "Qual o total de vendas de produtos por tipo de pagamento?"}'`

d) Simular Queries SQL

Para ver a simulação das queries SQL usando Pandas:

```
python simulador_sql.py
```

5. Executar Testes Automatizados

Execute os testes para garantir que tudo está funcionando:

```
# Executar todos os testes
pytest

# Executar testes com saída detalhada
pytest -v

# Executar apenas os testes do chatbot
pytest test_chatbot.py -v

# Executar apenas os testes da API
pytest test_api.py -v
```

🌐 Interface Web Moderna

Características do Design

A interface web foi completamente redesenhada com foco na experiência do usuário e na estética moderna:

🎨 Design System Profissional

- **Paleta de Cores:** Tons de azul suave (#0ea5e9) com cinzas neutros para máxima legibilidade
- **Tipografia:** Poppins, uma fonte moderna e elegante com diferentes pesos
- **Espaçamentos:** Sistema de grid consistente com espaçamentos harmoniosos
- **Sombras:** Sombras sutis e em camadas para profundidade visual
- **Bordas:** Cantos arredondados consistentes para um visual suave

⚡ Funcionalidades Avançadas

- **Tema Claro/Escuro:** Toggle automático com persistência no localStorage
- **Notificações Toast:** Sistema elegante de notificações não-intrusivas
- **Animações Suaves:** Transições CSS otimizadas para uma experiência fluida
- **Status em Tempo Real:** Monitoramento da conexão com a API e estatísticas de uso
- **Design Responsivo:** Adaptação perfeita para desktop, tablet e mobile

🔧 **Experiência do Usuário**

- **Chat Interativo:** Interface de chat moderna com avatares e formatação de mensagens
- **Perguntas Rápidas:** Botões coloridos com ícones e descrições para acesso rápido
- **Feedback Visual:** Indicadores de carregamento, estados de erro e confirmações
- **Acessibilidade:** Contraste adequado, navegação por teclado e textos alternativos

Componentes da Interface

Header Moderno

- Branding elegante com ícone personalizado
- Status de conexão discreto
- Toggle de tema com ícone animado

Seção de Chat

- Área de mensagens com scroll suave
- Avatares diferenciados para usuário e assistente
- Exibição de queries SQL em blocos de código formatados
- Estatísticas de uso no cabeçalho

Perguntas Rápidas

- Cards coloridos com ícones emoji
- Títulos descritivos e subtítulos explicativos
- Hover effects e animações de clique

Sidebar Informativa

- Cards organizados com informações do sistema
- Estatísticas em tempo real
- Guia de uso passo a passo

- Design em grid responsivo

Campo de Input

- Input moderno com placeholder animado
- Contador de caracteres com mudança de cor
- Botão de envio com ícone SVG
- Dicas de uso na parte inferior

📊 Dashboard Executivo no Power BI

O projeto inclui um Dashboard Executivo no Power BI, permitindo à equipe do hotel analisar dados de forma visual e interativa.

📈 Resultados e Insights

As análises e visualizações geradas fornecem insights valiosos:

- **Vendas por Tipo de Pagamento:** Identifica métodos de pagamento preferidos
- **Produtos Mais Vendidos:** Destaca itens de maior demanda para gestão de estoque
- **Custo de Estadias por Pet:** Oferece visão sobre receita por cliente para programas de fidelidade

🚀 Próximos Passos e Melhorias Futuras

Este MVP serve como base sólida para futuras expansões:

1. **Integração com Banco de Dados Real:** Migração para PostgreSQL/MySQL
2. **Processamento de Linguagem Natural Avançado:** Implementação de spaCy/NLTK
3. **Autenticação e Autorização:** Sistema de login com JWT/OAuth
4. **Deploy em Nuvem:** Containerização com Docker e deploy na AWS/GCP/Azure
5. **Monitoramento e Logging:** Ferramentas de observabilidade em produção

6. **Dashboards Dinâmicos:** Geração de relatórios sob demanda
7. **Integração com CRM:** Conexão com sistemas de gestão de clientes
8. **Análise Preditiva:** Modelos de ML para previsão de demanda

🧪 Testes e Qualidade

O projeto inclui 21 testes automatizados que garantem:

- **Cobertura Funcional:** Todos os endpoints da API e funções do chatbot
- **Casos de Erro:** Tratamento adequado de entradas inválidas
- **Integração:** Comunicação correta entre componentes
- **Performance:** Tempos de resposta dentro dos limites aceitáveis

🤝 Contribuição

Contribuições são bem-vindas! Este projeto demonstra:

- **Análise de Dados:** Processamento e visualização de dados de negócio
- **Desenvolvimento Backend:** API RESTful com Flask e testes automatizados
- **Desenvolvimento Frontend:** Interface web moderna e responsiva
- **Documentação Técnica:** Relatórios e guias detalhados
- **Boas Práticas:** Código limpo, testes e documentação

📄 Licença

Este projeto está licenciado sob a Licença MIT. Veja o arquivo `LICENSE` para mais detalhes.

Desenvolvido com ❤️ para demonstrar habilidades em Análise de Dados e Desenvolvimento Full-Stack