Cronograma Detalhado: Fase 1 - Coleta e Armazenamento de Dados via API

Duração Estimada: 8 dias (incluindo buffer) **Ferramentas Principais:** Python, FastAPI, Pydantic, Uvicorn, SQLite, Git, GitHub.

Semana 1: Construção e Integração

Dia 1 - Segunda-feira (08/09/2025): Setup do Ambiente e Estrutura do Projeto

- **Objetivo do dia:** Preparar todo o alicerce para o desenvolvimento, garantindo um ambiente de trabalho limpo e versionado.
- **Bloco 1 (1h): Configuração do Ambiente Virtual e Dependências.**
  - **Atividade:** Criar um ambiente virtual (`venv`) para isolar as dependências do projeto.
  - **Comandos:** `python -m venv venv` , `source venv/bin/activate` (ou `venv\Scripts\activate` no Windows).
  - **Ferramentas:** Python, venv.
  - **Atividade:** Instalar as bibliotecas iniciais: FastAPI e Uvicorn (servidor para a API).
  - **Comandos:** `pip install fastapi "uvicorn[standard]"`
- **Bloco 2 (1h): Estruturação de Diretórios e Controle de Versão.**
  - **Atividade:** Criar a estrutura de pastas do projeto (ex: `/api` , `/notebooks` , `/data`).
  - **Atividade:** Iniciar o repositório Git, criar um arquivo `.gitignore` (para ignorar arquivos como `__pycache__` e `venv`) e fazer o primeiro commit.
  - **Ferramentas:** Git.
- **Bloco 3 (1h): Criação do "Hello World" da API.**
  - **Atividade:** Criar o arquivo `main.py` dentro da pasta `/api` com um endpoint raiz (`@app.get("/")`) que retorna uma mensagem simples.
  - **Objetivo:** Validar que a instalação do FastAPI e Uvicorn foi bem-sucedida e que o servidor está rodando.
  - **Ferramentas:** FastAPI, Uvicorn.

Dia 2 - Terça-feira (09/09/2025): Modelagem dos Dados de Entrada

- **Objetivo do dia:** Definir um contrato claro para os dados que a API irá receber, garantindo a qualidade e o formato das informações.
- **Bloco 1 (1h): Análise do Schema dos Dados.**
  - **Atividade:** Abrir o arquivo `train.csv` e mapear todas as colunas e seus tipos de dados (texto, inteiro, decimal).
  - **Atenção:** Notar que a coluna `Class` é uma palavra-chave reservada em Python. Decidir renomeá-la para `flight_class` no nosso modelo de dados.
- **Bloco 2 e 3 (2h): Implementação do Modelo Pydantic.**
  - **Atividade:** Dentro de `main.py` , criar uma classe `Passageiro(BaseModel)` usando a biblioteca Pydantic. Adicionar todos os campos mapeados no bloco anterior como atributos da classe, com seus respectivos tipos (ex: `age: int` , `gender: str`).
  - **Ferramentas:** Pydantic (parte do FastAPI).

Dia 3 - Quarta-feira (10/09/2025): Desenvolvimento do Endpoint de Coleta

- **Objetivo do dia:** Criar o "portão de entrada" que receberá os dados dos passageiros.
- **Bloco 1 e 2 (2h): Desenvolvimento do Endpoint POST.**
  - **Atividade:** Criar um novo endpoint `@app.post("/adicionar_passageiro")` que recebe um objeto do tipo `Passageiro` (a classe Pydantic que você criou).
  - **Atividade:** Inicialmente, a função deste endpoint irá apenas imprimir os dados recebidos no console para fins de teste.
- **Bloco 3 (1h): Teste de Validação Automática.**
  - **Atividade:** Iniciar o servidor Uvicorn e acessar a documentação automática (`/docs`). Usar a interface do Swagger UI para enviar um JSON de teste para o seu novo endpoint e verificar se os dados são impressos corretamente no terminal.
  - **Ferramentas:** FastAPI Docs (Swagger UI).

Dia 4 - Quinta-feira (11/09/2025): Configuração do Banco de Dados

- **Objetivo do dia:** Preparar o local onde os dados coletados serão armazenados de forma persistente.
- **Bloco 1 (1h): Desenho da Tabela SQL.**
  - **Atividade:** Escrever a instrução `CREATE TABLE` em SQL para a tabela `passageiros`. As colunas devem corresponder exatamente aos campos do modelo Pydantic.
  - **Ferramentas:** SQL.
- **Bloco 2 e 3 (2h): Criação de um Script de Inicialização do Banco.**
  - **Atividade:** Dentro do `main.py` (ou em um arquivo separado), criar uma função `init_db()` que se conecta a um banco de dados SQLite e executa o comando `CREATE TABLE` que você desenhou.
  - **Objetivo:** Garantir que a tabela exista antes que a API tente inserir dados nela.
  - **Ferramentas:** Python (biblioteca `sqlite3`).

Dia 5 - Sexta-feira (12/09/2025): Integração Final da API com o Banco de Dados

- **Objetivo do dia:** Conectar as duas pontas: fazer com que os dados que chegam na API sejam efetivamente salvos no banco de dados.
- **Bloco 1, 2 e 3 (3h): Implementação da Lógica de Inserção.**
  - **Atividade:** Modificar o endpoint `@app.post("/adicionar_passageiro")`.
  - **Passos:**
    1. Remover a linha `print()`.
    2. Adicionar o código para conectar-se ao banco de dados SQLite.
    3. Escrever a instrução `INSERT INTO passageiros ...` usando os dados recebidos do objeto `passageiro`.
    4. Implementar `try...except` para tratamento de erros durante a inserção.
    5. Confirmar (`commit`) a transação e fechar a conexão.
    6. Retornar uma mensagem de sucesso em JSON.

Semana 2: Testes, Documentação e Encerramento da Fase

Dia 6 - Segunda-feira (15/09/2025): Testes de Integração

- **Objetivo do dia:** Garantir que o fluxo completo (recebimento de dados -> salvamento no banco) está funcionando perfeitamente e sem erros.
- **Bloco 1 e 2 (2h): Testes Funcionais.**
  - **Atividade:** Usar a documentação (`/docs`) para enviar múltiplos registros de passageiros válidos.
  - **Atividade:** Usar uma ferramenta de visualização de banco de dados (como o "DB Browser for SQLite") para abrir o arquivo `.db` e verificar se os dados foram inseridos corretamente, com os tipos de dados certos.
- **Bloco 3 (1h): Testes de Validação.**
  - **Atividade:** Tentar enviar dados inválidos (ex: um campo obrigatório faltando, uma idade como texto). Observar como a API retorna um erro de validação (422 Unprocessable Entity), comprovando que o Pydantic está funcionando.

Dia 7 - Terça-feira (16/09/2025): Documentação e Versionamento

- **Objetivo do dia:** Documentar o trabalho realizado nesta fase e garantir que o código esteja seguro e bem explicado no GitHub.
- **Bloco 1 e 2 (2h): Elaboração do README.md.**
  - **Atividade:** Iniciar a redação do arquivo `README.md` do projeto.
  - **Seções para esta fase:** "Sobre o Projeto", "Fase 1: API de Coleta de Dados", "Como Configurar o Ambiente" (incluindo `pip install -r requirements.txt`), "Como Executar a API".
- **Bloco 3 (1h): Finalização e Commit no GitHub.**
  - **Atividade:** Criar o arquivo `requirements.txt` (`pip freeze > requirements.txt`).
  - **Atividade:** Revisar todo o código, adicionar comentários onde for necessário, e enviar tudo para o seu repositório no GitHub com uma mensagem de commit clara (ex: "Completes Phase 1 - Data Collection API with SQLite integration").
  - **Ferramentas:** Git, GitHub.

Dia 8 - Quarta-feira (17/09/2025): Buffer e Revisão Final

- **Objetivo do dia:** Um dia de segurança para resolver pendências, corrigir bugs ou refinar o código.
- **Bloco 1, 2 e 3 (3h): Atividades Flexíveis.**
  - **Opção 1 (Se houver problemas):** Use este tempo para depurar e consertar qualquer erro encontrado nos dias anteriores.
  - **Opção 2 (Se tudo estiver OK):** Use este tempo para refatorar o código (melhorar nomes de variáveis, organizar funções) ou para aprimorar a documentação.
  - **Opção 3 (Se estiver adiantado):** Comece a pesquisar e planejar as bibliotecas para a próxima fase (Pandas, Scikit-learn).

Cronograma Detalhado: Fase 2 - Construção do Modelo de Machine Learning

Duração Estimada: 10 dias (incluindo buffer) **Ferramentas Principais:** Python, Jupyter Notebook, Pandas, Matplotlib, Seaborn, Scikit-learn, Git, GitHub.

Semana 1: Análise e Preparação dos Dados

Dia 9 - Quinta-feira (18/09/2025): Carregamento e Exploração Inicial (EDA)

- **Objetivo do dia:** Fazer o primeiro contato com os dados, entendendo sua estrutura, qualidade e principais características.
- **Bloco 1 (1h): Configuração do Ambiente de Análise.**
  - **Atividade:** Instalar as bibliotecas de análise: `pip install notebook pandas matplotlib seaborn scikit-learn`.
  - **Atividade:** Iniciar um novo Jupyter Notebook no diretório `/notebooks` do projeto. Carregar o arquivo `train.csv` em um DataFrame do Pandas.
- **Bloco 2 e 3 (2h): Inspeção Inicial do DataFrame.**
  - **Atividade:** Realizar a "sondagem" inicial dos dados.
  - **Comandos:** `df.info()` (para ver tipos e nulos), `df.describe()` (para estatísticas de colunas numéricas), `df.isnull().sum()` (para quantificar dados faltantes), `df.columns` (para listar colunas).
  - **Atividade:** Documentar as primeiras descobertas em células de Markdown no notebook (ex: "Identificamos 310 valores nulos na coluna 'Arrival Delay in Minutes'").

Dia 10 - Sexta-feira (19/09/2025): Análise Exploratória Visual - Parte 1

- **Objetivo do dia:** Transformar números em gráficos para entender a distribuição dos dados e o perfil dos passageiros.
- **Bloco 1 (1.5h): Análise Univariada Numérica.**
  - **Atividade:** Criar histogramas para entender a distribuição das principais variáveis numéricas, como `Age` e `Flight Distance`.
  - **Ferramentas:** Matplotlib/Seaborn (`sns.histplot`, `sns.boxplot`).
- **Bloco 2 (1.5h): Análise Univariada Categórica.**
  - **Atividade:** Criar gráficos de barras para visualizar a contagem das variáveis categóricas, como `Gender`, `Customer Type`, `Class` e, mais importante, a variável alvo `satisfaction`.
  - **Ferramentas:** Seaborn (`sns.countplot`).

Dia 11 - Segunda-feira (22/09/2025): Análise Exploratória Visual - Parte 2

- **Objetivo do dia:** Aprofundar a análise, buscando relações entre as variáveis e a satisfação do cliente.
- **Bloco 1 e 2 (2h): Análise Bivariada.**
  - **Atividade:** Investigar como as características dos passageiros influenciam na satisfação. Criar gráficos de barras agrupados.
  - **Perguntas a responder com gráficos:** Passageiros da `Business class` estão mais satisfeitos? O `Type of Travel` (negócios vs. pessoal) impacta a satisfação?
  - **Ferramentas:** Seaborn (`sns.countplot` com o parâmetro `hue='satisfaction'`).
- **Bloco 3 (1h): Análise de Correlação.**
  - **Atividade:** Criar um heatmap de correlação entre todas as variáveis numéricas para identificar se existem relações lineares fortes entre elas.
  - **Ferramentas:** Pandas (`df.corr()`) e Seaborn (`sns.heatmap`).

Dia 12 - Terça-feira (23/09/2025): Pré-processamento de Dados - Parte 1

- **Objetivo do dia:** Limpar e preparar o terreno para a construção do modelo.
- **Bloco 1 (1h): Tratamento de Dados Faltantes.**
  - **Atividade:** Definir uma estratégia para a coluna `Arrival Delay in Minutes`. Com base na análise, decidir se é melhor preencher com a média, a mediana, ou outro valor. Implementar a decisão.
  - **Ferramentas:** Pandas (`df.fillna()`).
- **Bloco 2 (2h): Seleção de Features e Separação.**
  - **Atividade:** Remover colunas que não serão usadas no modelo (ex: `Unnamed: 0`, `id`).
  - **Atividade:** Separar o DataFrame em duas variáveis: `X` (contendo todas as colunas de características) e `y` (contendo apenas a coluna alvo, `satisfaction`).

Dia 13 - Quarta-feira (24/09/2025): Pré-processamento de Dados - Parte 2

- **Objetivo do dia:** Transformar os dados para um formato que o algoritmo de Machine Learning consiga entender.
- **Bloco 1, 2 e 3 (3h): Construção de um Pipeline de Pré-processamento.**
  - **Atividade:** Usar o `ColumnTransformer` do Scikit-learn para aplicar transformações diferentes em colunas diferentes.
  - **Passos:**
    1. Identificar as colunas numéricas e categóricas.
    2. Aplicar `StandardScaler` nas colunas numéricas (para padronizar a escala).
    3. Aplicar `OneHotEncoder` nas colunas categóricas (para transformá-las em números).
  - **Ferramentas:** Scikit-learn (`ColumnTransformer`, `StandardScaler`, `OneHotEncoder`).

Semana 2: Modelagem, Avaliação e Encerramento

Dia 14 - Quinta-feira (25/09/2025): Treinamento do Modelo

- **Objetivo do dia:** Dividir os dados e treinar nosso primeiro modelo preditivo.
- **Bloco 1 (1h): Divisão em Treino e Teste.**
  - **Atividade:** Dividir `X` e `y` em conjuntos de treino e teste (ex: 80% para treino, 20% para teste) para que possamos avaliar o modelo de forma justa.
  - **Ferramentas:** Scikit-learn (`train_test_split`).
- **Bloco 2 e 3 (2h): Treinamento do Modelo.**
  - **Atividade:** Integrar o `ColumnTransformer` e um algoritmo de ML em um `Pipeline` completo.
  - **Atividade:** Escolher um modelo robusto como `RandomForestClassifier` e treiná-lo com os dados de treino (`pipeline.fit(X_train, y_train)`).
  - **Ferramentas:** Scikit-learn (`Pipeline`, `RandomForestClassifier`).

Dia 15 - Sexta-feira (26/09/2025): Avaliação de Performance

- **Objetivo do dia:** Medir o quão bom nosso modelo é em fazer previsões em dados que ele nunca viu.
- **Bloco 1 e 2 (2h): Métricas de Classificação.**
  - **Atividade:** Usar o modelo treinado para fazer previsões no conjunto de teste (`pipeline.predict(X_test)`).
  - **Atividade:** Calcular as principais métricas de avaliação: Acurácia, Precisão, Recall e F1-Score. Gerar um `classification_report`.
- **Bloco 3 (1h): Análise da Matriz de Confusão.**
  - **Atividade:** Gerar e visualizar uma matriz de confusão para entender os tipos de erros que o modelo está cometendo.
  - **Ferramentas:** Scikit-learn (`classification_report`, `confusion_matrix`, `ConfusionMatrixDisplay`).

Dia 16 - Segunda-feira (29/09/2025): Salvando o Modelo para Produção

- **Objetivo do dia:** "Empacotar" nosso modelo treinado para que ele possa ser usado na aplicação final.
- **Bloco 1 e 2 (2h): Persistência do Pipeline.**
  - **Atividade:** Salvar o objeto `pipeline` inteiro (que contém tanto o pré-processamento quanto o modelo treinado) em um arquivo.

- **Ferramentas:** Biblioteca `joblib` (`joblib.dump`).

- **Bloco 3 (1h): Teste de Carregamento.**
  - **Atividade:** Em uma nova célula (ou script), carregar o modelo salvo (`joblib.load`) e fazer uma previsão em um único exemplo para garantir que o processo de salvamento/carregamento funcionou.

Dia 17 - Terça-feira (30/09/2025): Documentação e Versionamento

- **Objetivo do dia:** Limpar e documentar todo o processo de análise e modelagem para o GitHub.
- **Bloco 1 e 2 (2h): Limpeza e Comentários no Notebook.**
  - **Atividade:** Revisar todo o Jupyter Notebook, remover códigos de teste desnecessários, e adicionar explicações claras em células de Markdown para cada etapa da análise. Contar a "história" dos dados.
- **Bloco 3 (1h): Commit no GitHub.**
  - **Atividade:** Atualizar o `README.md` com uma nova seção descrevendo a Fase 2.
  - **Atividade:** Fazer o commit do notebook finalizado (`.ipynb`), do modelo salvo (`.joblib`) e do `requirements.txt` atualizado para o repositório.

Dia 18 - Quarta-feira (01/10/2025): Buffer e Refinamento

- **Objetivo do dia:** Dia de segurança para refinar o modelo ou a análise.
- **Bloco 1, 2 e 3 (3h): Atividades Flexíveis.**
  - **Opção 1 (Refinamento):** Tentar um algoritmo diferente (ex: `XGBoost`) para ver se consegue uma performance melhor.
  - **Opção 2 (Ajuste de Hiperparâmetros):** Pesquisar e aplicar técnicas como `GridSearchCV` para encontrar os melhores parâmetros para o seu modelo.
  - **Opção 3 (Revisão):** Revisar todo o trabalho da fase, garantindo que a documentação está clara e o código é legível.

Cronograma Detalhado: Fase 3 - Modelo em Produção (Dashboard Interativo)

Duração Estimada: 4 dias **Ferramentas Principais:** Python, Streamlit, Pandas, Joblib, Git, GitHub.

Dia 19 - Quinta-feira (02/10/2025): Estrutura do Dashboard e Carregamento do Modelo

- **Objetivo do dia:** Montar o esqueleto da aplicação web e garantir que nosso modelo treinado possa ser carregado com sucesso.
- **Bloco 1 (1h): Setup do Ambiente e Aplicação Básica.**
  - **Atividade:** Instalar o Streamlit: `pip install streamlit`.
  - **Atividade:** Criar o arquivo `dashboard.py`.
  - **Atividade:** Escrever a estrutura inicial da aplicação com título e texto (`st.title`, `st.header`, `st.write`) e executá-la com `streamlit run dashboard.py` para validar a instalação.
- **Bloco 2 e 3 (2h): Carregamento do Modelo e Início da UI.**
  - **Atividade:** Escrever uma função para carregar seu arquivo `pipeline.joblib`. **Dica de especialista:** Use o decorador `@st.cache_resource` nesta função para que o modelo seja carregado na memória apenas uma vez, tornando a aplicação muito mais rápida.
  - **Atividade:** Criar uma barra lateral (`st.sidebar`) para organizar os inputs do usuário. Começar a adicionar os primeiros widgets (ex: `st.selectbox` para `Customer Type` e `Class`).

Dia 20 - Sexta-feira (03/10/2025): Interface do Usuário e Coleta de Dados

- **Objetivo do dia:** Finalizar a interface de entrada de dados e garantir que a aplicação consiga capturar todas as informações do usuário.
- **Bloco 1 e 2 (2h): Finalização do Formulário de Input.**
  - **Atividade:** Adicionar todos os widgets restantes na barra lateral para cada uma das features que seu modelo precisa para fazer uma previsão (ex: `st.slider` para `Age`, `st.number_input` para `Flight Distance`, etc.).
  - **Atividade:** Organizar os widgets com subtítulos (`st.subheader`) para uma melhor experiência do usuário.
- **Bloco 3 (1h): Coleta e Estruturação dos Inputs.**

- **Atividade:** Adicionar um botão de ação (`st.button('Prever Satisfação')`).
- **Atividade:** Escrever a lógica que, ao clicar no botão, coleta os valores de todos os widgets e os organiza em um DataFrame do Pandas de uma única linha. **Atenção:** Os nomes das colunas deste DataFrame devem ser *exatamente* os mesmos que o modelo espera.

Fim de Semana (04/10 e 05/10):

- Este é um período crucial. Recomendo fortemente usar algumas horas aqui para garantir que a interface esteja 100% funcional ou para adiantar a lógica de previsão. O prazo está apertado.

Dia 21 - Segunda-feira (06/10/2025): Lógica de Previsão e Apresentação do Resultado

- **Objetivo do dia:** Implementar a chamada ao modelo e exibir a previsão de forma clara e intuitiva para o usuário.
- **Bloco 1 e 2 (2h): Implementação da Lógica de Previsão.**
  - **Atividade:** Dentro da lógica do botão, passar o DataFrame de input para o método `pipeline.predict()` para obter a previsão ("satisfied" ou "neutral or dissatisfied").
  - **Atividade:** (Opcional, mas recomendado) Usar também o método `pipeline.predict_proba()` para obter a probabilidade da previsão, o que adiciona um nível de confiança ao resultado.
- **Bloco 3 (1h): Exibição Dinâmica do Resultado.**
  - **Atividade:** Exibir o resultado na tela principal.
  - **Sugestão:** Se a previsão for "satisfied", mostre uma mensagem de sucesso com `st.success()`. Se for "neutral or dissatisfied", use `st.warning()`.
  - **Atividade:** Use `st.metric()` ou `st.progress()` para mostrar a probabilidade da previsão de forma visual e profissional.

Dia 22 - Terça-feira (07/10/2025): RETA FINAL - PREPARAÇÃO PARA ENTREGA

- **Objetivo do dia:** Focar exclusivamente em revisar, finalizar e empacotar o projeto para a submissão. **Não desenvolva código novo hoje.**
- **Bloco 1 (1h): Revisão Geral e Testes Finais.**
  - **Atividade:** Execute todos os componentes do seu projeto uma última vez: a API, o notebook de treinamento e, principalmente, o dashboard. Garanta que tudo funciona como esperado.
- **Bloco 2 (1h): Documentação Final (README.md).**
  - **Atividade:** Atualize o `README.md` no GitHub com a seção final, explicando como executar o dashboard (`streamlit run dashboard.py`). Garanta que todas as instruções, do início ao fim, estejam claras.
- **Bloco 3 (1h): Storytelling e Empacotamento.**
  - **Atividade:** Grave um vídeo curto (usando Loom, OBS Studio, etc.) demonstrando o projeto. Siga o roteiro do *storytelling*: apresente o problema de negócio, mostre o dashboard funcionando, e explique brevemente os resultados do modelo.
  - **Atividade:** Crie o arquivo `.txt` final contendo o link para o seu repositório no GitHub e o link para o vídeo no YouTube (ou outra plataforma).
  - **Atividade: Submeta o projeto.**