✈️ CASE STUDY: AERO OPS PREDICTOR (Logistics Intelligence)

Cliente: SkyCargo Logistics (Transporte de Cargas Críticas) **Nível:** Pleno/Sênior **Sprint:** 1 Semana **Tecnologias:** Python (Geospatial), SQL (Postgres/MySQL), Power BI.

---

1. O Problema de Negócio (The Pain Point)

A **SkyCargo** transporta órgãos para transplante e peças urgentes de maquinário. Atualmente, eles dependem do painel da Infraero para saber se o voo chegou. Porém, esse painel é atualizado manualmente e muitas vezes mostra "No Horário" mesmo quando o avião está enfrentando uma tempestade a 200km do destino.

A Consequência: As ambulâncias e caminhões ficam esperando na pista, perdendo tempo crítico.

O Pedido: O Diretor de Operações quer uma "Torre de Controle Própria". Ele quer um sistema que calcule o **ETA (Estimated Time of Arrival)** real, cruzando a posição física do avião com as condições climáticas do destino.

---

2. A Missão Técnica

Você deve desenvolver o **Motor de Predição** desse sistema. O protótipo visual (Streamlit) já foi aprovado. Agora, precisamos da engenharia robusta por trás.

📡 Fase 1: Coleta & Geometria (Python + APIs)

Você usará duas APIs públicas: **OpenSky Network** (Tráfego Aéreo) e **Open-Meteo** (Clima).

Desafio de Engenharia:

1. **Rastreamento em Loop:** Crie um script `flight_tracker.py` que monitore um voo específico (ex: voos chegando em Guarulhos/GRU ou Galeão/GIG) a cada 5 minutos.
2. **Cálculo Geodésico:** O mundo não é plano. Não use Pitágoras!
    - Use a biblioteca `geopy` para calcular a distância exata entre a Lat/Lon do avião e a Lat/Lon da pista de pouso.

3. **Tratamento de Falhas:** A API da OpenSky é instável. Seu código deve ter um `try/except` robusto (se a API falhar, não pode derrubar o processo, deve tentar de novo ou logar o erro).

🧠 Fase 2: O Algoritmo de Predição (Regra de Negócio)

Não basta saber onde o avião está. Você deve prever o atraso. Implemente a seguinte lógica no seu código Python:

- **ETA Base:** `Distância Restante (km) / Velocidade Atual (km/h)`.
- **Fator "Vento de Proa":** Se o vento no destino for > 30 km/h (contra o pouso), adicione **+10 min** ao ETA.
- **Fator "Pista Molhada":** Se precipitação no destino > 0.5mm, adicione **+15 min** (aproximação lenta).
- **Alerta de Desvio:** Se a altitude cair drasticamente (> 5000 pés) longe do aeroporto, gerar flag de "Emergência/Pouso Não Programado".

🗄️ Fase 3: Data Warehouse (SQL)

Os dados coletados devem ser persistidos para auditoria. Modele e popule as seguintes tabelas:

Tabela: `FACT_VOO_TELEMETRIA` | Coluna | Tipo | Descrição | | :--- | :--- | :--- | | `id_log` | INT (PK) | Chave única | | `callsign` | VARCHAR | Ex: GLO1542 | | `latitude` | DECIMAL | Posição atual | | `longitude` | DECIMAL | Posição atual | | `velocidade_kmh` | FLOAT | Velocidade convertida | | `distancia_destino` | FLOAT | Distância calculada (Geopy) | | `timestamp_coleta` | DATETIME | Hora da leitura |

Tabela: `FACT_CONDICOES_POUSO` | Coluna | Tipo | Descrição | | :--- | :--- | :--- | | `id_clima` | INT (PK) | Chave única | | `aeroporto_destino` | VARCHAR | Ex: SBGR (Guarulhos) | | `vento_velocidade` | FLOAT | Extraído da Open-Meteo | | `chuva_mm` | FLOAT | Extraído da Open-Meteo | | `risco_calculado` | VARCHAR | 'Baixo', 'Médio', 'Crítico' |

---

3. Visualização (Power BI)

O Dashboard final deve responder:

1. **Mapa de Rastreio:** Plotar os pontos de Latitude/Longitude coletados no SQL para desenhar a rota real que o avião fez.
2. **Análise de Velocidade:** Um gráfico de linha mostrando a velocidade do avião ao longo do tempo. (Ele manteve a velocidade de cruzeiro ou teve que frear antes da hora?).
3. **Matriz de Risco:** Qual a porcentagem de voos que enfrentaram chuva no destino na última semana?

---

4. Critérios de Avaliação (Sênior)

- ✅ **Precisão Matemática:** A conversão de m/s (API) para km/h (Negócio) está correta? O cálculo de distância usou a curvatura da Terra (`geodesic`)?
- ✅ **Robustez da API:** O script continua rodando mesmo se a internet cair por 1 minuto? (Uso de `try/except` e logs).
- ✅ **Enriquecimento de Dados:** Você conseguiu cruzar o dado do avião com o dado do clima no mesmo registro do banco de dados?
- ✅ **SQL Limpo:** Uso correto de chaves e tipos de dados (não salvar latitude como texto!).

---

5. Dica do Especialista

> *"Aviões não voam em linha reta perfeita e o clima muda rápido. O diferencial deste case é o **Tempo Real**. Seu sistema deve ser capaz de dizer: 'O avião acelerou para recuperar o atraso' ou 'O avião entrou em padrão de espera (ficou dando voltas) devido à chuva'."*

Links Úteis para o Aluno

- **Ferramenta Base (Protótipo):** https://aeroopsradarcase.streamlit.app/
- **Docs OpenSky:** https://openskynetwork.github.io/opensky-api/
- **Docs Open-Meteo:** https://open-meteo.com/en/docs
- **Lib Geopy:** https://geopy.readthedocs.io/