Descrição geral:
O *FlightOps Planner* é uma ferramenta de **planejamento operacional de aeroportos** que consome os dados públicos da **ANAC (SIROS API)** para visualizar a **demanda de colaboradores** com base nos voos por aeroporto e período (qualquer aeroporto, qualquer mês).

O sistema combinará **dados de voo (chegadas, partidas, tempos de solo e atendimento)** com **classificações automáticas por tipo de aeronave, natureza do voo e companhia aérea**, para gerar **mapas interativos** e **curvas de demanda de pessoal** por função e por hora.

🧩 Etapas do Projeto (Roadmap)

FASE 1 — Backend e Modelagem de Dados

1. Extração

- Conectar à API da ANAC:
  Endpoint:
  https://sas.anac.gov.br/sas/siros_api/ssimfile?ds_temporada=S25
    - Parâmetro ds_temporada define a temporada (S25, W26, etc.)
- Extrair dados brutos (voos, ICAO, horários, origem/destino, tipo de aeronave etc.)
- Validar e normalizar campos (datas, ICAO, companhias).

2. Casamento de Voos

- Criar lógica em Python para **linkar voos correspondentes**:
    - Mesma **companhia aérea**, **aeroporto** e **tipo de aeronave**.
    - Voo de chegada linka ao voo de partida mais próximo.
- Gerar campos derivados:
    - Tempo de solo (arrival → departure).
    - Slots de atendimento (chegada e partida, conforme regras).

3. Tratamento e Padronização

- Arredondar horários para **slots de 10 minutos**.
- Abrir slots de **atendimento** e **solo** conforme lógica:
    - **Chegada:** -10 min até +30 min

    - **Partida:** -30 min até +10 min
- Classificar:
    - Tipo de aeronave: WIDE, NARROW, ATR, CARGO, MELI, CESNNA.
    - Tipo de voo: DOMÉSTICO / INTERNACIONAL.
    - Tipo de operação: PNT (>4h solo) ou TST (<4h solo).
- Enriquecer com dados complementares:
    - Base de aeronaves, aeroportos, calendário, slots.

4. Armazenamento

- Subir os dados tratados para o **Supabase** (Postgres):
    - Tabelas principais:
        - voos_raw
        - voos_tratados
        - slots_atendimento
        - slots_solo
        - aeroportos, aeronaves, calendario
- Criar queries SQL de suporte:
    - Agregações por hora, dia, semana.
    - Comparativos mês atual vs anterior.

FASE 2 — Visualizações e Frontend

1. Seleção de Parâmetros

- Usuário escolhe:
    - Aeroporto
    - Mês e ano
    - Companhias específicas (ex: apenas LATAM e AZUL)

2. Dashboards Interativos

- **Gráfico de Linha e Área:**

    - Comparativo da média mensal de voos em atendimento e solo — mês atual vs anterior.
- **Mapa 1 (Slots):**
    - Linhas: dias da semana
    - Colunas: slots de 10 min (00:00–23:50)
    - Exemplo: 1(2) = 1 chegada e 2 partidas.
    - Tooltip com detalhes dos voos.
- **Mapa 2 (Heatmap):**
    - Quantidade de voos simultaneamente em atendimento.
    - Tooltip mostra voos no slot.

3. Indicadores e Comparativos

- Total de voos mês atual vs anterior.
- Diferença percentual e normalização (30 dias).
- Semana mais crítica (pico de operação).

4. Curva de Demanda de Equipes

- Baseada no pior cenário de atendimento da semana.
- Distribuição horária (0h–23h).
- Exemplo:
    - 07h: 24 voos em atendimento →
        - 11 NARROW
        - 10 ATR
        - 3 WIDE
- Combinar com **parâmetros de staff padrão**:
    - Definir via tabela configurável (ASG, Aux. Líder, ASA Rampa, ASA Desembarque etc.)
- Geração de total por hora e por turno (A, B, C, D).

FASE 3 — Escalabilidade e Funcionalidades Futuras

- Autenticação de usuários (Supabase Auth).

- Exportar dashboards (PDF/PNG).
- Integração com APIs de clima e estatísticas (ex: passageiros).
- Machine Learning para previsão de demanda futura (baseado em histórico).
- Criação de “cenários simulados” (aumento de voos, horários, feriados etc.).

⚙️ Stack Técnica

Camada	Tecnologia
Backend	Python (FastAPI ou Streamlit para protótipo)
Banco	Supabase (Postgres)
Data Pipeline	Pandas + Requests + SQLAlchemy
Frontend	Next.js / React + Tailwind / Plotly / ECharts
Visualizações	Heatmap + Line Chart + Tooltips dinâmicos
Hospedagem	Vercel (frontend) + Supabase (backend)

🎯 Objetivo Final

Criar uma ferramenta **modular, gratuita e pública** que permite:

- Visualizar o comportamento da malha aérea de qualquer aeroporto.
- Analisar a **curva de demanda operacional** com base real.
- Estimar **recursos humanos necessários** por turno, por tipo de voo e por cenário.

RESUMO GERAL DO PROJETO

Visão Geral

Nome de trabalho: FlightOps Planner
Objetivo: Estimar e visualizar demanda operacional de equipes em aeroportos do Brasil, usando dados da ANAC (SIROS API) para qualquer aeroporto e qualquer mês/temporada.
Saída principal:

1. mapas interativos por slots de 10 min (chegadas/partidas + heatmap de "voos em atendimento"),
2. comparativos mês atual x anterior (linhas/área),
3. curva de demanda por hora (0–23h) agregada por tipo de aeronave e convertida em *headcount* por função e por turno.

Fonte de Dados

API: https://sas.anac.gov.br/sas/siros_api/ssimfile?ds_temporada={temporada}

- temporada exemplo: S25, W26, etc.
- A resposta contém malha em formato "similar a SSIM"; os campos podem variar de acordo com a temporada.
- Precisamos normalizar para um esquema interno.

Bases auxiliares (internas):

- aeroportos (IATA/ICAO, nome, UF, fuso opcional),
- aeronaves (IATA act type → classe: WIDE, NARROW, ATR, CARGO, MELI (73C GOL), CESNNA/outros leves),
- calendario (datas, dia da semana, flags de feriado),
- slots_10m (00:00–23:50, incremento de 10 min),
- cia_aerea (códigos e nomes, opcional).

Regras de Negócio (core)

1) Casamento (link) de voos

Para cada aeroporto AP dentro do mês/temporada alvo:

- Para um **voo de chegada** V_in (cia = X, act_type = Y, data/hora chegada arredondada para slot de 10 min):
    - Encontrar a **partida candidata** V_out mais próxima **no mesmo aeroporto**, **mesma companhia** e **mesmo tipo de aeronave**.
    - Janela de busca padrão: [chegada, chegada + 36h] (parametrizável).
    - Critério de escolha: **menor diferença de tempo**; se empate, escolher menor número de passageiros ou número do voo mais próximo (heurística secundária).
    - Se **não houver partida** compatível na janela → o voo fica **sem link** (solo termina no fim do dia ou marcado como aberto).

Observação: o link é usado para medir **tempo de solo** e gerar **faixas de atendimento**.

2) Slots (arredondamento)

- Todo horário é **arredondado para slots de 10 min**:
    - Ex.: 10:25 → **10:30**, 10:04 → **10:00** (regra: arredondar para o mais próximo; em caso de empate, arredondar **para cima**).
    - Este comportamento é global e configurável.

3) Atendimento e Solo

Dado um voo linkado (chegada A e partida D):

- **Atendimento de Chegada**: de A - 10 min até A + 30 min.
- **Atendimento de Partida**: de D - 30 min até D + 10 min.
- **Tempo de Solo**: de A até D.

Para voos **sem link de partida**:

- Gerar atendimento de chegada normalmente.
- Solo pode ser considerado “aberto” até A + 3h (parâmetro) ou ignorado para métricas de PNT/TST (decisão: usar 3h default).

4) Classificações

- **Tipo de aeronave (classe)**:
    - CARGO se a cia ou act_type estiver marcado como cargueiro.
    - MELI para Gol 73C (regra específica).
    - WIDE para act types 777, 787, 330, 350, 767 etc.

    - NARROW para 737, 320 etc.
    - ATR para ATRs (72/42).
    - CESNNA/“Leves” para monomotores e equivalentes.
- **PNT/TST** (pela duração do solo):
    - PNT se solo > 4h; caso contrário TST.
- **Doméstico/Internacional**: por origem/destino (mesma nacionalidade = DOM).

Arquitetura & Stack

- **ETL/Backend:** Python (Pandas, Requests, SQLAlchemy).
- **Banco:** Supabase (Postgres).
- **API:** FastAPI (ou PostgREST do próprio Supabase + RPC).
- **Frontend:** Next.js + React + Tailwind. Gráficos: ECharts ou Plotly.
- **Hospedagem:** Vercel (front) + Supabase (DB/Storage/Auth).

Modelo de Dados (Postgres / Supabase)

Tabelas principais

voos_raw

Dados crus normalizados da ANAC (granularidade: trecho/registro).

- id (PK), temporada, cia, act_type, origem, destino, aeroporto_operacao (o AP que estamos analisando),
- dt_partida_utc, dt_chegada_utc (antes do arredondamento),
- numero_voo, natureza (DOM/INT, se disponível), assentos_previstos (opcional), payload (jsonb).

voos_tratados

Agrega lógica de link + arredondamento.

- id (PK), raw_id,
- aeroporto, cia, act_type, classe_aeronave,

- chegada_slot (timestamp arredondado), partida_slot (timestamp arredondado ou null),
- solo_min (int), pnt_tst (PNT/TST), dom_int (DOM/INT),
- link_status (linked | no_departure | no_arrival),
- numero_voo_in, numero_voo_out.

slots_atendimento

Uma linha por **voo x slot de atendimento**.

- id (PK), voo_id (FK voos_tratados), aeroporto, cia, classe_aeronave,
- slot_ts (timestamp 10 min), fase (ARR | DEP),
- dia_semana (1–7), ano_mes (YYYY-MM), turno (A/B/C/D via regra abaixo),
- dom_int, pnt_tst.

slots_solo

Uma linha por **voo x slot de solo**.

- id (PK), voo_id, slot_ts, aeroporto, cia, classe_aeronave, dia_semana, ano_mes.

param_staff_por_classe

Configurações de headcount por **classe de aeronave**.

- classe_aeronave (PK),
- ASG, AUX_LIDER, ASA_RAMPA, ASA_TRIAGEM, ASA_DESEMB, OPER_EQUIP (inteiros).

Será editável via UI.

Auxiliares

- aeronaves_ref(act_type, classe_aeronave, is_cargo bool, is_meli bool, ...)
- aeroportos_ref(iata, icao, nome, uf, timezone)
- calendario_ref(dt, ano_mes, semana_mes, dia_semana, feriado bool)
- turnos_ref(nome, hora_inicio, hora_fim) — Ex.: A=00–05:59, B=06–11:59, C=12–17:59, D=18–23:59 (configurável).

ETL — Passo a Passo

1. **Extract:** baixar temporada (S25, W26...) e parsear registros.
2. **Normalize:** padronizar campos, mapear companhias, act types, aeroportos.
3. **Round:** arredondar horários para slots de 10 min (regra global).
4. **Link:** executar algoritmo de casamento (chegada → partida).
5. **Derive:** calcular solo, PNT/TST, DOM/INT, classe de aeronave.
6. **Expand:** gerar linhas em slots_atendimento (janelas ARR/DEP) e slots_solo (A→D).
7. **Load:** gravar em Supabase, com *upserts* idempotentes por temporada+aeroporto+mes.

Pseudocódigo do link:

```
for each arrival in arrivals[aeroporto, cia, act_type, dia]:
  cands = departures.filter(
    aeroporto==arrival.aeroporto and
    cia==arrival.cia and
    act_type==arrival.act_type and
    partida_slot >= arrival.chegada_slot and
    partida_slot <= arrival.chegada_slot + timedelta(hours=36)
  )
  if cands:
    best = argmin_cands_by(partida_slot - chegada_slot,
tie_breakers=[abs(numero_voo_out-numero_voo_in)])
    link(arrival, best)
  else:
    mark_no_departure(arrival)
```

Expansão de slots (exemplo chegada):

```
window_arr = [chegada_slot - 10min, chegada_slot + 30min]
for ts in slots_10m_in(window_arr):
  insert(slots_atendimento, voo_id, ts, fase='ARR', classe, cia, aeroporto, etc)
```

Consultas/Agregações (SQL)

1) Mapa “Chegadas/Partidas por slot”

- Eixo X = slots 10 min (00:00–23:50), Eixo Y = dia da semana (seg–dom).
- Célula mostra qtd_chegadas e qtd_partidas no slot.

```
-- chegadas
SELECT date_trunc('day', slot_ts) AS dia, EXTRACT(DOW FROM slot_ts) AS dow,
    date_part('hour', slot_ts) AS hh, date_part('minute', slot_ts) AS mm,
    COUNT(*) AS chegadas
FROM slots_atendimento
WHERE aeroporto = :AP
 AND ano_mes = :YYYYMM
 AND fase = 'ARR'
GROUP BY 1,2,3,4;

-- partidas (fase='DEP') idem;
```

No frontend, mesclar por slot (ARR e DEP) e exibir como chegadas(partidas), ex.: 1(2).

2) Heatmap “Voos em atendimento simultâneo”

- Conta **linhas em slots_atendimento** por slot.

```
SELECT date_trunc('day', slot_ts) AS dia, EXTRACT(DOW FROM slot_ts) AS dow,
    date_part('hour', slot_ts) AS hh, date_part('minute', slot_ts) AS mm,
    COUNT(*) AS voos_em_atendimento
FROM slots_atendimento
WHERE aeroporto = :AP AND ano_mes = :YYYYMM
GROUP BY 1,2,3,4;
```

3) Linhas/Área “Média de atendimento e solo”

- Série 1: média de voos em atendimento por slot ao longo do mês.

- Série 2: média de voos em solo por slot ao longo do mês.
- Fazer para **mês alvo** e **mês anterior**, e plotar ambas curvas.

```
-- atendimento por slot/hora (agrupar por hora para o gráfico)
SELECT date_part('hour', slot_ts) AS hora,
    AVG(cnt) AS media_mes
FROM (
 SELECT slot_ts, COUNT(*) AS cnt
 FROM slots_atendimento
 WHERE aeroporto=:AP AND ano_mes=:YYYYMM
 GROUP BY slot_ts
) t
GROUP BY 1
ORDER BY 1;
```

Repetir para slots_solo e para o YYYYMM_anterior.

4) Normalização para 30 dias

Para comparar meses diferentes:

valor_normalizado = (total_voos_no_mes / qtd_dias_do_mes) * 30

diff_% = (norm_mes_atual - norm_mes_anterior) / norm_mes_anterior

5) Semana mais crítica

- Encontrar a **semana (ISO week)** com **pico máximo** de voos_em_atendimento.

```
SELECT week, MAX(qtd) AS pico
FROM (
 SELECT to_char(slot_ts, 'IYYY-IW') AS week, slot_ts, COUNT(*) AS qtd
 FROM slots_atendimento
 WHERE aeroporto=:AP AND ano_mes=:YYYYMM
 GROUP BY 1,2
) s
```

```
GROUP BY 1
ORDER BY pico DESC
LIMIT 1;
```

6) Curva de Demanda (por hora)

- Selecionar **pior cenário dentro da semana mais crítica** por **hora**:

```
WITH semana_crit AS (
  SELECT ... -- (usar a consulta acima para obter a ISO week alvo)
),
atend_semana AS (
  SELECT slot_ts, COUNT(*) qtd
  FROM slots_atendimento s
  JOIN semana_crit w ON to_char(s.slot_ts,'IYYY-IW') = w.week
  WHERE s.aeroporto=:AP
  GROUP BY 1
),
por_hora AS (
  SELECT date_part('hour', slot_ts) AS hora, MAX(qtd) AS pico_hora
  FROM atend_semana
  GROUP BY 1
)
SELECT * FROM por_hora ORDER BY hora;
```

- **Quebrar por classe de aeronave** no pico (para distribuir equipes):

```
SELECT date_part('hour', slot_ts) AS hora, classe_aeronave, MAX(qtd) pico
FROM (
  SELECT slot_ts, classe_aeronave, COUNT(*) qtd
  FROM slots_atendimento s
  WHERE to_char(slot_ts,'IYYY-IW')=:WEEK AND aeroporto=:AP
  GROUP BY 1,2
```

) t

GROUP BY 1,2;

- Multiplicar pico por parâmetros de param_staff_por_classe para obter **headcount por função**.

API (contratos sugeridos)

Base /api

- GET /aeroportos → lista.
- GET /temporadas → exemplos: S25, W26.
- POST /etl/run { temporada } → roda ingestão e tratamento (admin).
- GET /resumo?ap=REC&mes=2025-12&cias=AZUL,LATAM
    - retorna:
        - totais mês e mês anterior + normalização 30 dias,
        - semana crítica,
        - séries para gráfico (atendimento/solo — mês e mês anterior).
- GET /mapa_slots?ap=REC&mes=2025-12&cias=AZUL,LATAM
    - retorna matriz por dia-da-semana x slot 10 min com chegadas, partidas e lista de voos para tooltip.
- GET /heatmap_atendimento?ap=REC&mes=2025-12&cias=AZUL,LATAM
    - retorna contagem por dia/slot e lista de voos (para tooltip).
- GET /curva_demanda?ap=REC&mes=2025-12&cias=AZUL,LATAM
    - retorna por hora:
        - pico_total, por_classe[{classe: pico}],
        - headcount_por_funcao[{funcao: qtd}] (aplicando param_staff_por_classe).
- GET /param_staff e PUT /param_staff (configuração por classe).

Filtro por cias: todas as rotas filtram cia IN (...) tanto em voos_tratados quanto nos slots_*.

UI/Gráficos (especificação)

1) Header — Filtros

- **Seletores:** Aeroporto, Mês/Ano, Companhias (multi-select).
- **KPIs:** total voos mês, total mês anterior, variação %, total normalizado 30d e variação %.
- **Card "Semana Crítica"** (ISO week + pico da semana).

2) Gráfico Linhas/Área (comparativo)

- **X:** horas (0–23).
- **Séries:**
    - Atendimento (mês atual), Atendimento (mês anterior),
    - Solo (mês atual), Solo (mês anterior).
- **Tooltip:** hora → médias correspondentes (n médio de voos).
- **Legenda:** habilitar/ocultar séries.

3) Mapa Chegadas/Partidas (grade 7 x 144)

- **Y:** dias da semana (Seg→Dom).
- **X:** slots de 10 min (00:00 → 23:50).
- **Célula:** string c(p), ex.: 1(2).
- **Tooltip (on hover):**
    - Lista dos voos no slot:
        - Voo: G3 1023 | ARR 10:20 | Orig: SSA | Classe: NARROW
        - Voo: G3 2044 | DEP 10:20 | Dest: BSB | Classe: NARROW
    - Se possível, incluir DOM/INT, PNT/TST.

4) Heatmap "Voos em Atendimento"

- Mesmo grid (7 x 144).
- **Cor** proporcional à contagem (0 → claro; alto → mais escuro).
- **Label opcional** com o número.
- **Tooltip:** lista de voos em atendimento no slot (idêntico ao mapa 3, porém só "em atendimento").

5) Curva de Demanda por Hora

- **Tabela/Gráfico de barras (0–23):**
    - pico_total por hora na **semana crítica**.
    - Abaixo, **distribuição por classe** (ex.: 11 NARROW, 10 ATR, 3 WIDE).
    - **Headcount por função** (resultado após multiplicar pela config):
        - ASG: 34, Aux.Líder: 8, ASA Rampa: 52, ….
- **Controles:** editar parâmetros por classe (abre modal que altera param_staff_por_classe).

Tooltips — formato de payload

Para todos os endpoints de mapa, incluir campo tooltipItems por célula:


```
{
 "slot": "2025-12-07T10:20:00Z",
 "chegadas": 1,
 "partidas": 2,
 "tooltipItems": [
  { "fase": "ARR", "voo": "LA1234", "origem": "SSA", "chegada": "10:20", "classe": "NARROW", "dom_int": "DOM", "pnt_tst": "TST" },
  { "fase": "DEP", "voo": "LA2044", "destino": "BSB", "partida": "10:20", "classe": "NARROW", "dom_int": "DOM", "pnt_tst": "TST" }
 ]
}
```

Turnos (cálculo)

- Padrão sugerido (configurável):
    - **A**: 00:00–05:59
    - **B**: 06:00–11:59
    - **C**: 12:00–17:59
    - **D**: 18:00–23:59

- Para **dimensionamento por turno**, pegar **o maior headcount por função** dentro do intervalo do turno (pior caso).

Regras de Filtro (front/back)

- **Aeroporto**: filtra por aeroporto em todas as consultas.
- **Mês/Ano**: filtra por ano_mes = 'YYYY-MM'.
- **Companhias**: cia IN (:lista) aplicada em voos_tratados, slots_atendimento, slots_solo.
- **Classe**: filtros adicionais opcionais.

Aceite (Acceptance Criteria)

1. **ETL** roda para uma temporada (ex.: S25) e popula voos_tratados, slots_atendimento, slots_solo.
2. **Link de voos**: ≥95% dos voos com chegada possuem partida linkada quando operacionalmente plausível.
3. **Arredondamento**: todos horários convertidos a slots de 10 min.
4. **Mapas** exibem dados coerentes com as regras de atendimento/solo e tooltips listam voos corretos.
5. **Comparativo mês x anterior** funciona com normalização para 30 dias.
6. **Semana crítica** identificada e **curva de demanda** apresentada por hora, com headcount por função usando param_staff_por_classe.
7. **Parâmetros por classe** editáveis na UI e persistidos no DB.
8. **Filtros por campanha/mês/aeroporto/cias** alteram todas as visualizações de forma consistente.

Performance & Qualidade

- **Indexação DB:** slots_atendimento(slot_ts, aeroporto, ano_mes, cia), voos_tratados(aeroporto, ano_mes, cia).
- **Paginação/virtualização no front** para grids 7 x 144.
- **Cache** por (ap, mês, cias) no Supabase/Edge Functions quando possível.
- **Testes**:

- Unitários do arredondamento/link/expansão,
- Smoke tests dos endpoints,
- Testes visuais (número de células e tooltips exemplo).