**Documentação do Sistema — Projeto Aéreo (Fatura Export + Comparativos)**

Esta documentação consolida **como o sistema funciona hoje**, onde ficam as **regras de filtragem**, **quais colunas e abas** são usadas, **como estender/adaptar** quando os formatos mudam, e **quais arquivos** você deve alterar para cada tipo de ajuste.

**Visão Geral**

**Fluxo funcional**

1. **Upload de PDF da fatura** (rota Fatura Export)  
   → extração da tabela (páginas 2–fim), normalização de cabeçalhos e linhas, identificação do Tipo\_Serviço, cálculo de Frete\_Peso, e cache (cache/<file\_id>.feather).
2. **Upload da planilha de Acordos (.xlsx)**  
   → leitura das abas de acordos:
   * **Reservado LUFT - Bases** (apenas para RESERVADO MEDS)
   * **Abas “Est. …”** (para RESERVADO MEDS fallback e para ESTANDAR …, via códigos)
3. **Comparação** PDF × Acordos  
   → Frete\_Peso vs Frete\_Acordo (ou *Simulação de Acordado* = Frete\_Acordo \* Peso Taxado)  
   → cálculo de Diferenca, Dif\_abs, Dif\_%, Status, Observacao e Fonte\_Tarifa.
4. **Exibição** (DataTable + cards de KPIs) e **Export Excel** (formatado).
5. **Histórico**: uploads/outputs carimbados com timestamp, listagem e download.

**Estrutura do Projeto (pastas/arquivos relevantes)**

* app.py — cria app Flask, registra blueprints.
* config.py — **configuração central** (Paths, Tuning: tolerância, epsilon).
* routes/
  + main.py — **Home** (index com tiles das ferramentas).
  + fatura\_export.py — **Fatura Export** (upload PDF → compare).
  + historicos\_faturas.py — **Histórico** (lista e baixa arquivos de uploads/ e outputs/).
* repositories/
  + pdf\_repository.py — **extração do PDF** (pdfplumber/Camelot), normalização e limpeza.
* services/
  + comparison\_service.py — **regras de comparação** (equivalente ao table\_compare.py antigo).
  + formatting.py — arredondamentos/formatos HTML/Excel.
* formats/
  + reserved\_bases.py — transformação **Reservado LUFT - Bases** para tidy.
  + stations\_generic.py — transformação para **Est. …** (tratamento por código ou por múltiplos códigos válidos).
* utils/
  + files.py — ensure\_dirs, allowed\_file.
* templates/
  + base.html — layout base (navbar, CDN, mensagens).
  + index.html — home com cards/tiles.
  + fatura\_index.html — tela de upload da Fatura Export.
  + compare.html — tabela comparativa + KPIs.
  + historicos.html — listagem de uploads/outputs.
* static/css/
  + base.css — estilos globais.
  + theme.css — temas/overrides (inclui ajuste do contraste dos botões DataTables).

**Extração do PDF (regras e pontos de ajuste)**

Arquivo: repositories/pdf\_repository.py

**Regras principais**

* **Páginas**: apenas da **página 2 em diante** (pages="2-end" quando Camelot; pdfplumber itera e pula a 1ª).
* **Engine**: tenta **Camelot** (flavor="stream", edge\_tol=500) e, se não conseguir, cai para **pdfplumber**.
* **Cabeçalhos**:
  + Constrói header a partir da **primeira linha** de cada tabela.
  + Aplica **substituições** para colunas coladas (ex.: "TaxasPeso" → "Taxas|Peso") e quebra por |.
  + Mapeia para nomes padronizados usando **HEADER\_ALIASES**.  
    Exemplos:
    - origem → Origem
    - v. frete, vlr frete → Vlr Frete
    - peso taxado / peso + taxado → Peso Taxado
    - … e outros (veja o dicionário no arquivo).
* **Reparo de linhas mescladas**: \_fix\_merged\_data\_rows
  + Heurística: quando Origem ou Documento está NaN **e** a **1ª célula** tem **texto longo**, divide por **múltiplos espaços** (\s{2,}) e redistribui as colunas.
* **Limpeza de linhas ruído**: \_drop\_noise\_rows
  + Remove linhas com Sub Total, Copyright, Total.
  + Mantém apenas linhas com **Documento contendo dígitos** ou linhas cabeçalho de serviço (e.g., RESERVADO, ESTANDAR, VELOZ), para poder **ffill**.
* **Identificação do serviço por bloco**: \_add\_service\_type\_and\_clean
  + Linhas de “header de serviço” com qualquer de:  
    ['RESERVADO MEDS', 'ESTANDAR 10 BASICO', 'ESTANDAR 2 BASICO', 'ESTANDAR 2 MEDS', 'VELOZ']
  + Preenche Tipo\_Serviço para as linhas subsequentes com **ffill**.
* **Números**:
  + Coerção de Vlr Frete, Outras Taxas, Peso Taxado, Vlr Total, Vlr Advalorem para numérico (remove vírgula de milhar etc.).
* **Documento**: limpa caracteres não numéricos.
* **Frete\_Peso**: se Peso Taxado > 0, calcula Frete\_Peso = Vlr Frete / Peso Taxado.
* **Ordenação final de colunas**: TARGET\_COLS primeiro, extras depois.

**Onde mexer se o PDF mudar**

* **Novos nomes de cabeçalho** → **HEADER\_ALIASES**  
  Adicione regex novas para mapear para as colunas padrão.
* **Novos “headers” de serviço** → lista service\_keywords em \_add\_service\_type\_and\_clean.  
  Ex.: incluir ESTANDAR 5 BASICO, EXPRESSO, etc.
* **Heurística de reparo** → \_fix\_merged\_data\_rows (limiar de tamanho, regex do splitter).
* **Páginas** → em extract\_tables\_from\_pdf (mudar range "2-end").
* **Cálculo de Frete\_Peso** → \_add\_frete\_peso (se mudar a lógica, por exemplo usar tarifa mínima).

**Interpretação de Serviços (mapeamento)**

Arquivo: services/comparison\_service.py (helpers portados do antigo table\_compare.py)

* \_service\_code\_from\_tipo(Tipo\_Serviço) **normaliza** e mapeia:
  + **RESERVADO MEDS** → RESMD
  + **ESTANDAR 2 MEDS**, **ESTANDAR 10 BASICO**, **ESTANDAR 2 BASICO** → **ST2MD** *(por ora)*
* **Códigos válidos** para buscar nas abas “Est.” (ordem importa):  
  VALID\_CODES\_ST2 = ["ST2MD", "MD/PE", "MEDICAMENTOS", "MD"]
* **Por quê?**  
  Nas “Est. …” os tratamentos variam de nome e data de conferência; por enquanto **todos os “ESTANDAR …”** do PDF procuram tarifas da família **ST2MD** (ou equivalentes “MEDICAMENTOS”, “MD/PE”, “MD”).

**Extensão futura:** se “ESTANDAR 10 BASICO” passar a ter um **código próprio** nas planilhas, você só altera esse mapeamento e/ou cria um coletor dedicado.

**Planilha de Acordos — Regras de Leitura**

**1) Aba “Reservado LUFT - Bases” (somente RESERVADO MEDS)**

Arquivo: formats/reserved\_bases.py

* Estrutura esperada:
  + **Linha 1** (linha 0 no pandas) = **cabeçalho**; a 1ª coluna vira Destino, e as demais são **as Origem do cabeçalho**.
  + Demais linhas → valores por **(Destino, Origem)**.
* Transformação:
  + “Gira” a matriz com melt para o formato *tidy*:
  + Destino | Origem | Frete\_Acordo | Fonte\_Tarifa
  + Converte Frete\_Acordo para número via \_to\_num (aceita R$, vírgula, ponto, e **corrige data** tipo 13/08/2025 → 13.8).
  + Remove NaN e dupes por (Origem, Destino).
  + Marca Fonte\_Tarifa = "Reservado LUFT - Bases - RESMD".

**Onde mexer se mudar o layout desta aba?**

* Trocar o **nome da aba** (constante BAS\_SHEET).
* Se a **primeira linha não for cabeçalho**, ajuste a leitura (p.ex. skiprows, ou detectar o header de outra forma).
* Se inverterem **Origem/Destino**, mude a lógica da criação do tidy.

**2) Abas “Est. …” (RESERVADO fallback e ESTANDAR)**

Arquivo: formats/stations\_generic.py

Existem **dois modos** de coletar:

**2A) Por tratamento único (ex.: RESMD)**

* Onde usamos: fallback para RESERVADO MEDS quando não achou na Base.
* Regras:
  + Localiza colunas por regex:
    - ORIGEM\_PAT = r"^\s\*origem\s\*$"
    - DESTINO\_PAT = r"^\s\*destino\s\*$"
    - TRATAMENTO\_PATS (variações de “Tratamento cadastrado em sistema conferido” com/sem **data**).
  + Seleção de **coluna de tarifa** por padrões que dão preferência a **reajuste + código** (ex.: Tarifas com Reajuste Março 2025 ST2MD) e mantêm **fallbacks**:
    - Função \_tarifa\_patterns\_for(code) gera os regex.
    - Fallbacks genéricos: reajuste.\*, tarifas?.\*reajuste.\*.
  + Filtro por **trat\_code**: normaliza texto e procura **token** (remove espaços/pontuação).
  + pick\_tarifa(row): pega a **primeira** entre as colunas candidatas que virar número por \_to\_num.
  + Resulta em tidy:
  + Origem | Destino | Frete\_Acordo | Fonte\_Tarifa (ex.: "Est. RJ - RESMD")

**2B) Por múltiplos códigos válidos (ESTANDAR)**

* Onde usamos: para **qualquer linha** do PDF que mapeia para ST2MD.
* Regras:
  + Em cada linha da aba “Est. …”, verifica **qual código válido** (em VALID\_CODES\_ST2) **aparece** como **tokens** (evita MD casar com RESMD).
  + Seleção de **coluna de tarifa** como acima, preferindo **ST2MD** (ou o prefer\_tarifa\_code que você passar).
  + Fonte\_Tarifa fica como "<Nome da aba> - <Código que casou>" (ex.: Est. GYN - MEDICAMENTOS).

**Onde mexer quando as abas variam:**

* **Nome/variações do “Tratamento cadastrado em sistema conferido …”**  
  → edite **TRATAMENTO\_PATS** (regex).  
  Exemplos reais que você citou:
  + “Tratamento cadastrado em sistema conferido”
  + “Tratamento cadastrado em sistema conferido **17/02/25**”
* **Nome da coluna de tarifa (“Reajuste”)**  
  → adicione/edite padrões em **\_tarifa\_patterns\_for**.  
  Exemplos reais:
  + “Tarifas com Reajuste Março 2025”
  + “Tarifas com Reajuste Março 2025 **ST2MD**”
  + “Valor por Kg - **Novo Portfólio**” *(pode virar fallback se necessário)*
* **Novos códigos válidos**  
  → liste em **VALID\_CODES\_ST2** e/ou mude o mapeamento em \_service\_code\_from\_tipo.

**Regras da Comparação**

Arquivo: services/comparison\_service.py

**Entradas críticas**

* **Do PDF** (colunas padronizadas pela extração):  
  Tipo\_Serviço, Origem, Destino, Data, Documento,  
  Vlr Frete, Peso Taxado, (opcionais: Vlr Total, Outras Taxas, Vlr Advalorem),  
  **calculada**: Frete\_Peso = Vlr Frete / Peso Taxado.
* **Dos Acordos (tidy)**:  
  Origem, Destino, Frete\_Acordo, Fonte\_Tarifa.

**Como decide qual tabela usar**

1. **Se Tipo\_Serviço mapeia para RESMD** (RESERVADO MEDS):
   * **1º** procura na **“Reservado LUFT - Bases”**.
   * **Se não achar** (Frete\_Acordo = NaN), faz **fallback** nas **abas “Est. …”** por RESMD.
2. **Se Tipo\_Serviço mapeia para ST2MD** (ESTANDAR 2 MEDS / 10 BASICO / 2 BASICO):
   * **Ignora** “Reservado LUFT - Bases”.
   * Procura **somente** nas **abas “Est. …”** por **qualquer um** de VALID\_CODES\_ST2  
     (ordem importa), **preferindo** colunas de tarifa que tenham ST2MD.

**Cálculos**

* Frete\_Peso (PDF) e Frete\_Acordo (planilha) são **valores por Kg**.
* **Simulação do Acordado** (para KPI): Frete\_Acordo \* Peso Taxado  
  → “quanto deveria ter sido cobrado, se seguissem a tabela”.
* Diferenca = Frete\_Peso - Frete\_Acordo
* Dif\_abs = abs(Diferenca)
* Dif\_% = ((Frete\_Peso / Frete\_Acordo) - 1) \* 100 (se Frete\_Acordo > 0)
* Status (com base na **tolerância** de %, **padrão 1.0**):
  + Frete\_Acordo **NaN** → Sem tarifa
  + senão, abs(Dif\_%) ≤ tolerancia\_pct → Dentro da tolerância
  + caso contrário → Fora da tolerância
* Observacao:
  + **Somente** quando Frete\_Acordo estiver NaN:  
    “Sem tarifa encontrada nas Estações para (Origem, Destino).”

**Arredondamento e apresentação**

Arquivo: services/formatting.py

* Zera “lixo” numérico de ponto flutuante: valores com |x| < EPSILON (default **1e-9**) viram 0.0.
* Arredonda Diferenca, Dif\_abs, Dif\_% para **2 casas**.
* Em HTML: Dif\_% vira **string com “%”** ("12,34%").
* Em Excel: Dif\_% é gravada como **fração** (0.1234) com formato %.

**Onde ajustar**: config.py → Tuning

* TOLERANCIA\_PCT (p.ex. 1.0)
* EPSILON (p.ex. 1e-9)

**KPIs (cards) no compare.html**

* **Total Cobrado (Vlr Frete)** = sum(Vlr Frete)
  + mostra **quanto foi cobrado na fatura**.
* **Simulação do Acordado** = sum(Frete\_Acordo \* Peso Taxado)
  + ignora linhas **sem tarifa**.
* **Total Diferença (R$)** = sum( (Frete\_Peso - Frete\_Acordo) \* Peso Taxado )  
  *(se você preferir diferença monetária total)*
  + **Observação:** No corpo da tabela, Diferenca é por Kg. Para o KPI fazer sentido monetário, multiplicamos por Peso Taxado antes de somar.
* **% Fora da Tolerância** (ignora “Sem tarifa”):
  + count(Status == "Fora da tolerância") / count(Status ∈ {"Dentro","Fora"}) \* 100
* **% Sem Tarifa**:
  + count(Status == "Sem tarifa") / total\_linhas \* 100

Se preferir manter “Total Diferença” como **por Kg**, documente isso no título. Recomendação: KPI **monetário** soma Diferenca \* Peso Taxado.

**“Fonte\_Tarifa”**

A coluna Fonte\_Tarifa **sempre diz** onde o valor foi encontrado:

* Para Base de Reservado:  
  Reservado LUFT - Bases - RESMD
* Para Estações:  
  Est. <NOME> - <CÓDIGO QUE CASA> — ex.: Est. RJ - ST2MD, Est. GYN - MEDICAMENTOS.

Isto ajuda a auditar divergências e entender a origem do preço.

**Histórico de Faturas**

Arquivo: routes/historicos\_faturas.py

* **Carimbo de data/hora** nos nomes dos arquivos (SP):
  + Upload do PDF: YYYY-MM-DD\_HH-MM-SS\_<fileid>.pdf
  + Upload dos acordos: YYYY-MM-DD\_HH-MM-SS\_<fileid>\_acordos.xlsx
  + Output comparativo: YYYY-MM-DD\_HH-MM-SS\_<fileid>\_comparativo.xlsx
* Página **/historico/faturas** lista uploads/ e outputs/ em **ordem decrescente de data**, com **tamanho** e **modificação** e botão **Baixar**.
* Downloads seguros: sanitiza filename para evitar **path traversal**.

**Onde mexer**:

* Formato do timestamp → função \_now\_stamp() no fatura\_export.py.
* Ordenação/colunas exibidas → historicos.html.

**Front-end**

* **DataTables** (CDN) para tabela comparativa e histórico.
* **Botões**: Copy/CSV/Excel com contraste corrigido (overrides em static/css/theme.css).
* **Cards**: KPIs calculados no template (JS) a partir do conteúdo da tabela (ou via dataset JSON, se preferir evoluir).
* **Layouts**:
  + base.html — placeholders {% block %}, navbar com links para **Fatura Export** e **Histórico**.
  + index.html — tiles: “Fatura Export”, “Histórico”.
  + fatura\_index.html — formulário de upload PDF → depois de extrair, formulário para upload do .xlsx.
  + compare.html — tabela + filtros e **cards**.
  + historicos.html — tabs Uploads/Outputs com DataTables.

**Onde mexer quando algo muda**

**1) PDF (nomes de colunas, páginas, headers de serviço)**

* repositories/pdf\_repository.py
  + **Cabeçalhos** → HEADER\_ALIASES, \_normalize\_header
  + **Detectar bloco de serviço** → service\_keywords em \_add\_service\_type\_and\_clean
  + **Páginas** → extract\_tables\_from\_pdf
  + **Reparo de linhas** → \_fix\_merged\_data\_rows (regex/split)
  + **Cálculo de Frete\_Peso** → \_add\_frete\_peso

**2) Planilha — Reservado LUFT - Bases**

* formats/reserved\_bases.py
  + Nome da **aba**, posição do **cabeçalho**, inversão **Origem/Destino**.
  + Parsing de valores “data como número” → \_to\_num (no comparison\_service — é compartilhado).

**3) Planilha — Estações**

* formats/stations\_generic.py
  + Colunas **Origem/Destino** → ORIGEM\_PAT / DESTINO\_PAT
  + Coluna de **Tratamento** → TRATAMENTO\_PATS (inclua variantes com data)
  + Colunas **Tarifa/Reajuste** → \_tarifa\_patterns\_for(code) (adicione versões que você ver)
  + Casamento de **códigos** por token → VALID\_CODES\_ST2, \_find\_matching\_code

**4) Mapa Serviço → Código**

* services/comparison\_service.py → \_service\_code\_from\_tipo e VALID\_CODES\_ST2

**5) Regras de tolerância e arredondamento**

* config.py → Tuning.TOLERANCIA\_PCT, Tuning.EPSILON
* services/formatting.py → arredondamentos, % para HTML/Excel

**6) KPIs**

* templates/compare.html
  + Fórmulas dos cards: confirme se quer **por Kg** ou **monetário** (recomendado monetário).
  + Se preferir, mover os cálculos para backend e injetar no template.

**Principais Colunas (glossário)**

* **Do PDF extraído**
  + Tipo\_Serviço — cabeçalho de bloco (ffill), p.ex. RESERVADO MEDS, ESTANDAR 2 MEDS.
  + Origem / Destino
  + Data — data do conhecimento/nota.
  + Documento — número fiscal/CT-e.
  + Vlr Frete — **valor cobrado** (R$).
  + Peso Taxado — peso utilizado para tarifação.
  + Frete\_Peso — Vlr Frete / Peso Taxado (R$/Kg).
  + (opcionais) Outras Taxas, Vlr Total, Vlr Advalorem, Tipo de Cte etc.
* **Dos Acordos (tidy)**
  + Origem / Destino
  + Frete\_Acordo — **valor/kg acordado** (R$/Kg).
  + Fonte\_Tarifa — aba + código onde achou a tarifa.
* **Do Comparativo**
  + Diferenca — Frete\_Peso - Frete\_Acordo (**por Kg**).
  + Dif\_abs — abs(Diferenca).
  + Dif\_% — % sobre o acordo.
  + Status — “Dentro”, “Fora”, “Sem tarifa”.
  + Observacao — mensagens de falta de tarifa.

**Exemplos de Casos Reais (como o sistema decide)**

* **GIG → MAO** com Tipo\_Serviço = RESERVADO MEDS
  1. Procura em **Reservado LUFT - Bases** (Origem no cabeçalho, Destino na lateral).
  2. Se **não tiver**, procura em **Est. RJ / Est. GYN / …** filtrando Tratamento … conferido por RESMD.
  3. A coluna de tarifa pode ter nomes ligeiramente diferentes, mas os regex pegam:
     + “Tarifas com Reajuste Março 2025 **ST2MD**”
     + “Tarifas com Reajuste Março 2025”
     + “Reajuste …”
* **GIG → MAO** com Tipo\_Serviço = ESTANDAR 2 MEDS
  1. **Ignora** a base de Reservado.
  2. Procura nas Estações **qualquer** entre ["ST2MD","MD/PE","MEDICAMENTOS","MD"].
  3. Fonte\_Tarifa mostra **qual** bateu (ex.: Est. RJ - ST2MD).

**Pitfalls tratados**

* **Datas digitadas no lugar de tarifa** (ex.: 13/08/2025)  
  → \_to\_num interpreta como 13.8 (dia.mês), garantindo padronização (13,8 → 13.8).
* **Resíduo numérico** (ex.: 1,77635683940025E-15)  
  → EPSILON zera e **arredonda** para 2 casas.
* **MD casando com RESMD**  
  → matching por **tokens** (evita falsos positivos).
* **Origem/Destino com espaços/acentos/minúsculas**  
  → \_std\_text e \_strip\_accents normalizam tudo para comparar.

**Extensões Futuras (como fazer direito)**

* **Novos serviços** (ex.: VELOZ):
  + Ensine o **PDF** a reconhecer o bloco (adicionar a palavra-chave na lista de serviço).
  + Diga **para onde buscar no Acordo** (novo coletor formats/… e lógica em comparison\_service).
* **Tarifa Mínima / Faixas**:
  + Encaminhar regras no comparison\_service (aplicar max(Frete\_Acordo \* Peso, TarifaMinima)).
* **Server-side DataTables**:
  + Se o volume crescer, migrar os cálculos de KPIs para backend e servir **JSON** paginado.
* **Auditoria**:
  + Guardar o **par de fontes** usado (Base ou Est., qual aba/coluna) e o **hash** do arquivo.

**Testes Rápidos (sanidade)**

1. **PDF com 2+ serviços** (RESERVADO + ESTANDAR)
   * Verifique se ambos aparecem no comparativo (antes só vinha RESERVADO).
2. **Caso GIG → MAO** nos dois serviços
   * Checar que **Fonte\_Tarifa** difere conforme o serviço.
   * Checar Dif\_% arredondada e Status conforme tolerância.
3. **Linhas sem tarifa**
   * Status = “Sem tarifa” e o card % Sem tarifa > 0.
4. **Export Excel**
   * Dif\_% como **percentual** (0.00%).

**Configurações Importantes (onde estão)**

Arquivo: config.py (exemplo típico)

from dataclasses import dataclass

from pathlib import Path

@dataclass

class Paths:

BASE\_DIR: Path

UPLOAD\_DIR: Path

OUTPUT\_DIR: Path

CACHE\_DIR: Path

@dataclass

class Tuning:

TOLERANCIA\_PCT: float = 1.0 # % (ex.: 1.0 = ±1%)

EPSILON: float = 1e-9 # zera resíduo numérico

@dataclass

class AppConfig:

paths: Paths

tuning: Tuning = Tuning()

* **Mudar tolerância** → Tuning.TOLERANCIA\_PCT.
* **Apertar/afrouxar arredondamento** → Tuning.EPSILON.

**Dependências realmente usadas**

Essencial:

* Flask, Jinja2 (templates)
* pandas, numpy
* openpyxl (leitura .xlsx)
* pyarrow (feather)
* pdfplumber (+ pdfminer.six), Pillow
* XlsxWriter (export Excel)
* tzdata (Windows/zoneinfo)

Opcional:

* camelot-py (+ opencv-python-headless) — fallback de extração.

**Conclusão**

* **Pontos-chaves para manter o sistema “à prova de variação”** são os **regex** de detecção de colunas e serviços, e a camada de normalização (\_to\_num, tokens, acentos).
* **Quando algo “não casar”** (ex.: uma nova coluna de Reajuste com nome diferente), 90% das vezes basta **ajustar os padrões** em:
  + repositories/pdf\_repository.py → HEADER\_ALIASES, service\_keywords
  + formats/stations\_generic.py → TRATAMENTO\_PATS, \_tarifa\_patterns\_for
  + services/comparison\_service.py → VALID\_CODES\_ST2, \_service\_code\_from\_tipo

Se quiser, posso gerar um **checklist de mudança** (um guia passo-a-passo) para quando chegar **uma nova planilha ou layout de PDF** — é só me mandar exemplos de nomes de colunas/tab e eu já deixo os padrões prontos.