

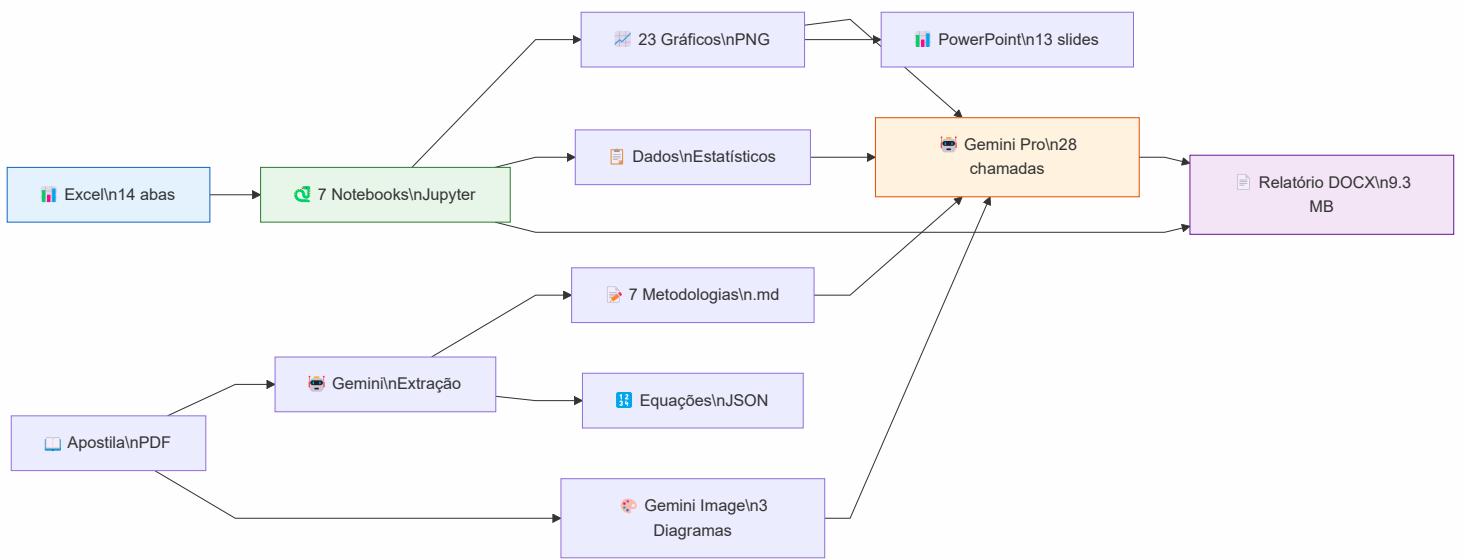
# Pipeline Completo — Auditoria de Medição de Gás Natural

**Projeto:** Curso ABAR — Medições Inteligentes e Gestão Integrada de Dados

**Período dos dados:** Abril a Setembro de 2025 (183 dias)

**Data de execução:** 11-12 de Fevereiro de 2026

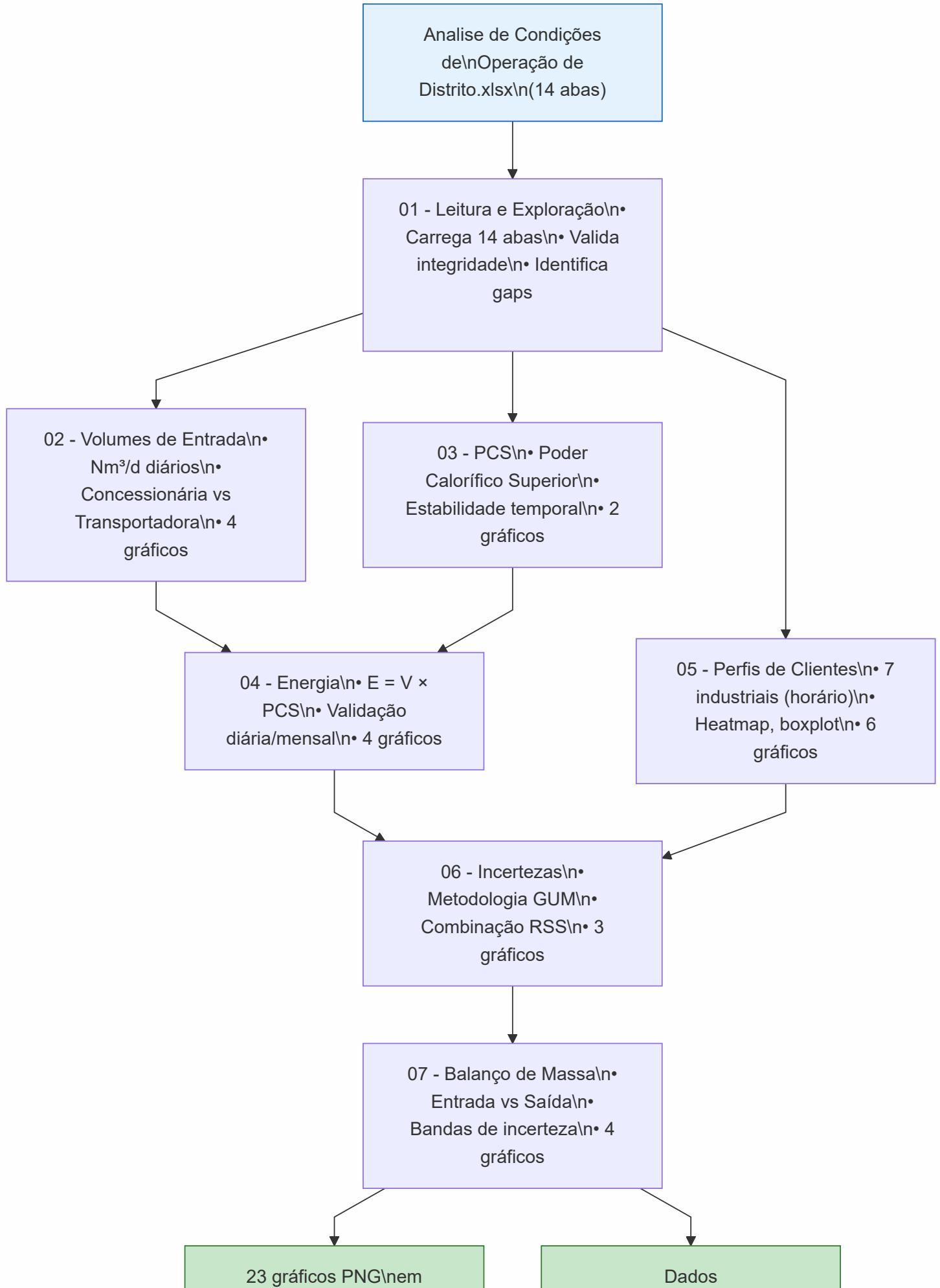
## Visão Geral do Pipeline



## Cronologia Detalhada

### Fase 1 — Notebooks de Análise (11/fev)

**Objetivo:** Explorar, processar e analisar os dados brutos do Excel.

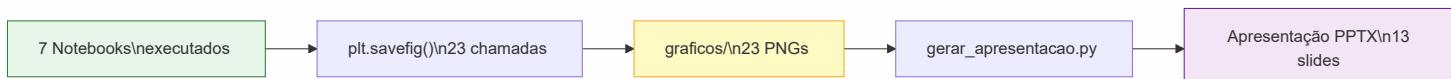




#	Notebook	Conteúdo	Gráficos
1	01_leitura_e_exploracao.ipynb	Carrega 14 abas, valida dados, identifica gaps (Coop Taxi 57% NaN)	1
2	02_analise_volumes_entrada.ipynb	Volumes diários Nm³/d, comparação Concessionária vs Transportadora	4
3	03_analise_pcs.ipynb	Poder Calorífico Superior — estabilidade, distribuição	2
4	04_calculo_energia.ipynb	$E = V \times PCS$ , validação de energia diária e mensal	4
5	05_perfis_clientes.ipynb	7 clientes industriais — perfis horários, heatmap, boxplot	6
6	06_sumario_e_incertezas.ipynb	Incertezas GUM, combinação RSS por tramo e cliente	3
7	07_balanco_massa.ipynb	Balanço entrada vs saída com bandas de incerteza	4

**Correções aplicadas:** Ajustes de usecols/skiprows na leitura do Excel, PCS com espaço no nome da coluna, incerteza 0.0109 vs 1.09%, waterfall com valores negativos, conversão pd.to\_numeric.

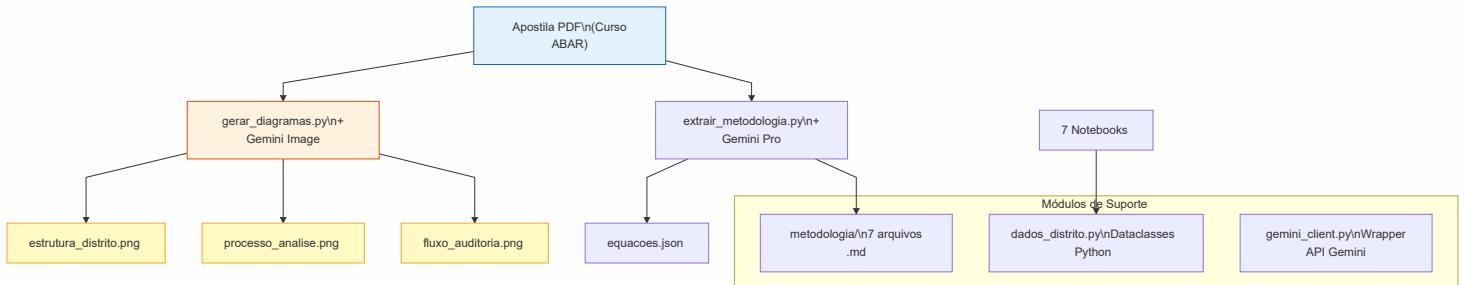
## Fase 2 — Exportação de Gráficos e Apresentação (11-12/fev)



Artefato	Descrição
graficos/ (23 PNGs)	plt.savefig() adicionado a cada gráfico dos notebooks
gerar_apresentacao.py	Script Python que gera PowerPoint automatizado
Apresentacao_Curso_ABAR.pptx	13 slides com gráficos incorporados
requirements.txt	11 dependências (pandas, numpy, matplotlib, openpyxl, python-pptx, etc.)

## Fase 3 — Infraestrutura do Relatório (12/fev)

**Objetivo:** Preparar os módulos Python para geração automatizada do relatório DOCX via LLM.



Arquivo	Função
<code>dados_distrito.py</code>	Dataclasses com dados estatísticos dos notebooks (volumes, PCS, energia, clientes, incertezas, balanço)
<code>gemini_client.py</code>	Wrapper da API Gemini — <code>analyze_section()</code> (texto+imagens+thinking) e <code>generate_image()</code>
<code>extrair_metodologia.py</code>	Extrai teoria da apostila PDF via Gemini → 7 .md + <code>equacoes.json</code> em <code>metodologia/</code>
<code>gerar_diagramas.py</code>	Gera 3 diagramas de processo via <code>gemini-3-pro-image-preview</code> → <code>diagramas/</code>

#### Modelos Gemini utilizados:

- `gemini-3-pro-preview` — texto com thinking (análise de seções)
- `gemini-3-pro-image-preview` — geração de imagens (diagramas)

## Fase 4 — Relatório v1/v2 (12/fev)

**Objetivo:** Primeira geração do relatório (monolítica — 1 chamada LLM por seção).

Arquivo	Função
<code>prompts_auditoria.py</code>	9 templates de prompts para as seções
<code>docx_builder.py</code>	Construtor DOCX: capa, TOC, markdown→Word, equações LaTeX→OMML, tabelas, gráficos
<code>gerar_relatorio_auditoria.py</code>	Orquestrador: 9 chamadas Gemini → cache → montagem DOCX

**Resultado:** `Relatorio_Auditoria_Distrito_v2.docx` — 9 seções, 6 tabelas, 23 gráficos, 3 diagramas, equações nativas Word (OMML).

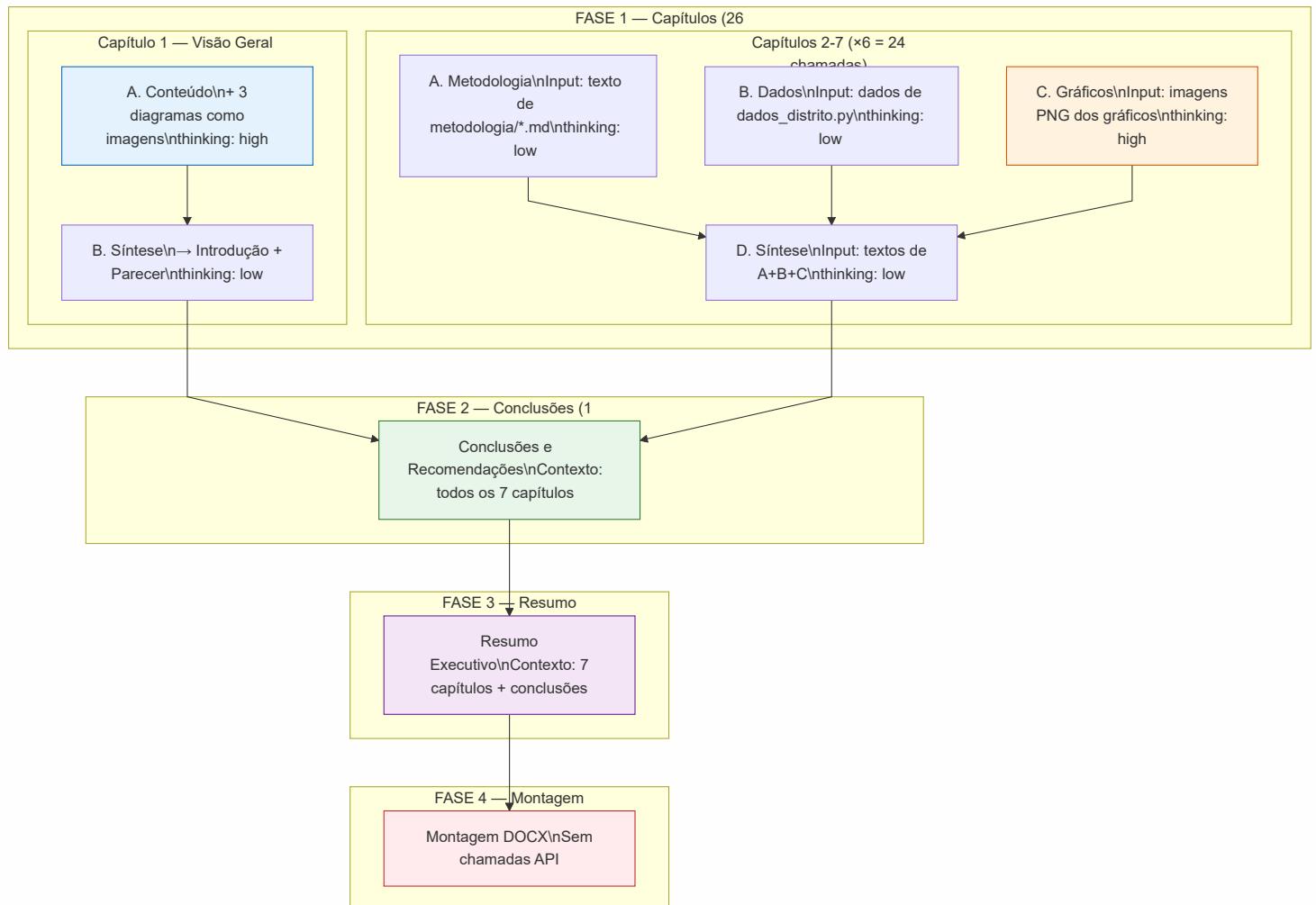
#### Problemas identificados na revisão:

1. Seção 1 inventava 3 gráficos inexistentes (LLM nunca recebeu os diagramas)
2. Gráficos apareciam DEPOIS do texto que os referenciava
3. `clientes_heatmap.png` ausente do DOCX (enviado ao LLM mas não inserido)
4. Resumo Executivo gerado ANTES de Conclusões
5. Geração monolítica causando alucinações
6. Sem estrutura lógica Metodologia → Dados → Análise nos capítulos

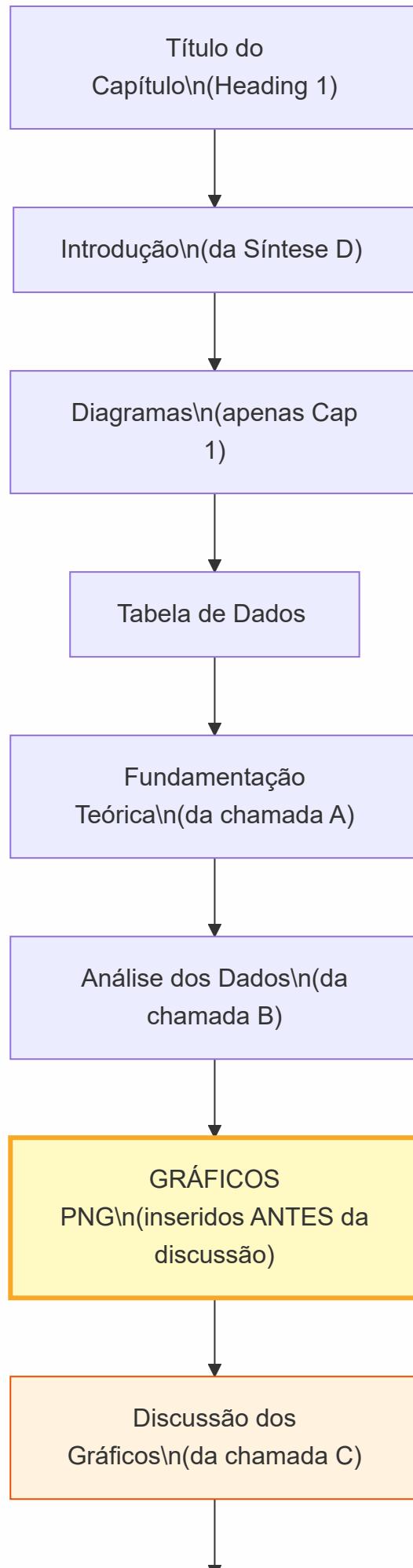
## Fase 5 — Pipeline Segmentado v4 (12/fev)

**Objetivo:** Resolver os 6 problemas estruturais com geração segmentada.

### Arquitetura Segmentada — 28 Chamadas LLM



## **Estrutura DOCX por Capítulo (ordem corrigida)**



## Parecer Regulatório\n(da Síntese D)

## Problemas Corrigidos

#	Problema	Solução
P1	Seção 1 inventava 3 gráficos inexistentes	Diagramas passados como imagens ao LLM via <code>prompt_secao1_conteudo()</code>
P2	Gráficos após o texto que os referencia	<code>add_chapter_structured()</code> insere gráficos ANTES da discussão
P3	<code>clientes_heatmap.png</code> ausente do DOCX	Adicionado como Figura 5.4 no <code>CHAPTER_CONFIG</code>
P4	Resumo Executivo gerado antes de Conclusões	Ordem: Capítulos → Conclusões → Resumo Executivo
P5	Geração monolítica (1 chamada/seção)	4 sub-chamadas por capítulo (segmentado)
P6	Sem estrutura lógica nos capítulos	Ordem fixa: Metodologia → Dados → Gráficos → Síntese

## Arquivos Reescritos

Arquivo	Mudança
<code>prompts_auditoria.py</code>	6 system prompts especializados + <code>CHAPTER_CONFIG</code> dict + funções genéricas de prompt
<code>docx_builder.py</code>	<code>add_chapter_structured()</code> com ordem correta de elementos
<code>gerar_relatorio_auditoria.py</code>	<code>ChapterResult</code> dataclass, cache granular (28 .md), pipeline 4 fases

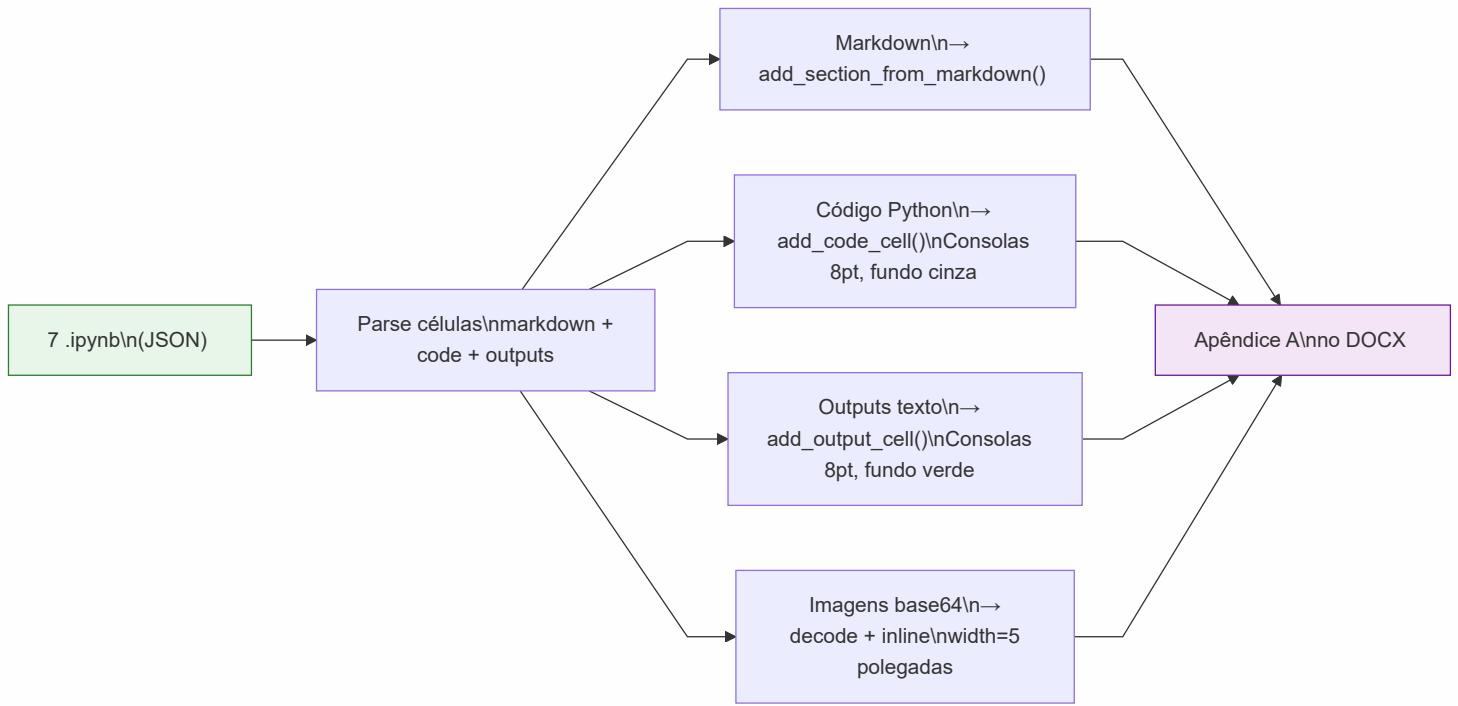
## Cache Granular

```
cache/
├── cap1_a_conteudo.md      # Cap 1 – Conteúdo (com diagramas)
├── cap1_b_sintese.md       # Cap 1 – Introdução + Parecer
├── cap2_a_metodologia.md   # Cap 2 – Fundamentação teórica
├── cap2_b_dados.md         # Cap 2 – Análise dos dados
├── cap2_c_graficos.md      # Cap 2 – Discussão dos gráficos
├── cap2_d_sintese.md       # Cap 2 – Introdução + Parecer
├── ...
├── cap7_d_sintese.md       # (mesmo padrão para Cap 3-7)
└── conclusoes.md           # Conclusões e Recomendações
└── resumo_executivo.md     # Resumo Executivo
```

Permite `--resume` (retoma de onde parou) e `--montar` (remonta DOCX sem chamar API).

## Fase 6 — Apêndice com Notebooks (12/fev)

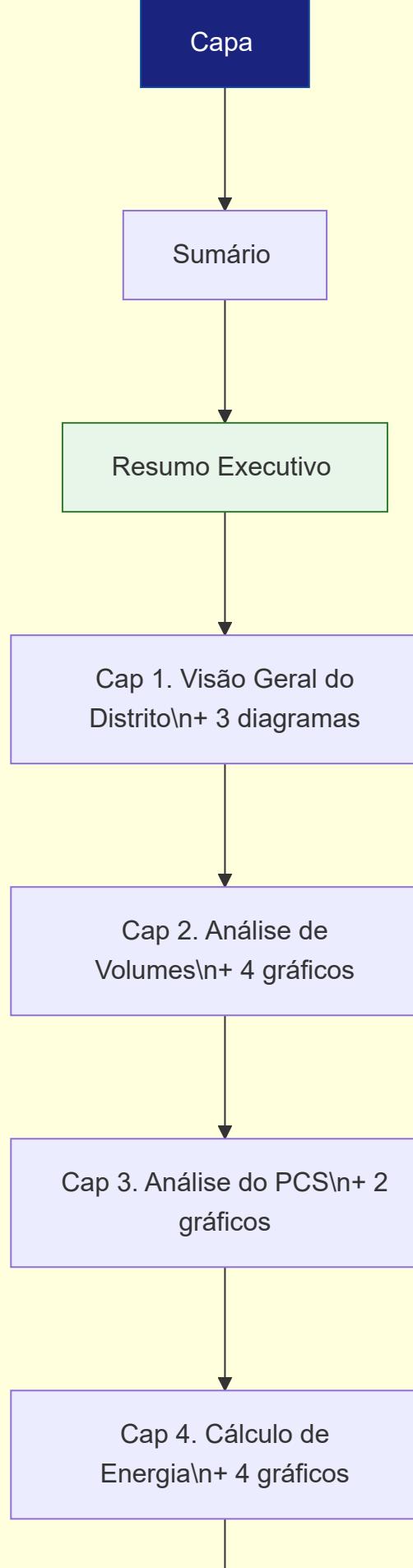
**Objetivo:** Incluir código-fonte e resultados dos 7 notebooks como anexo.

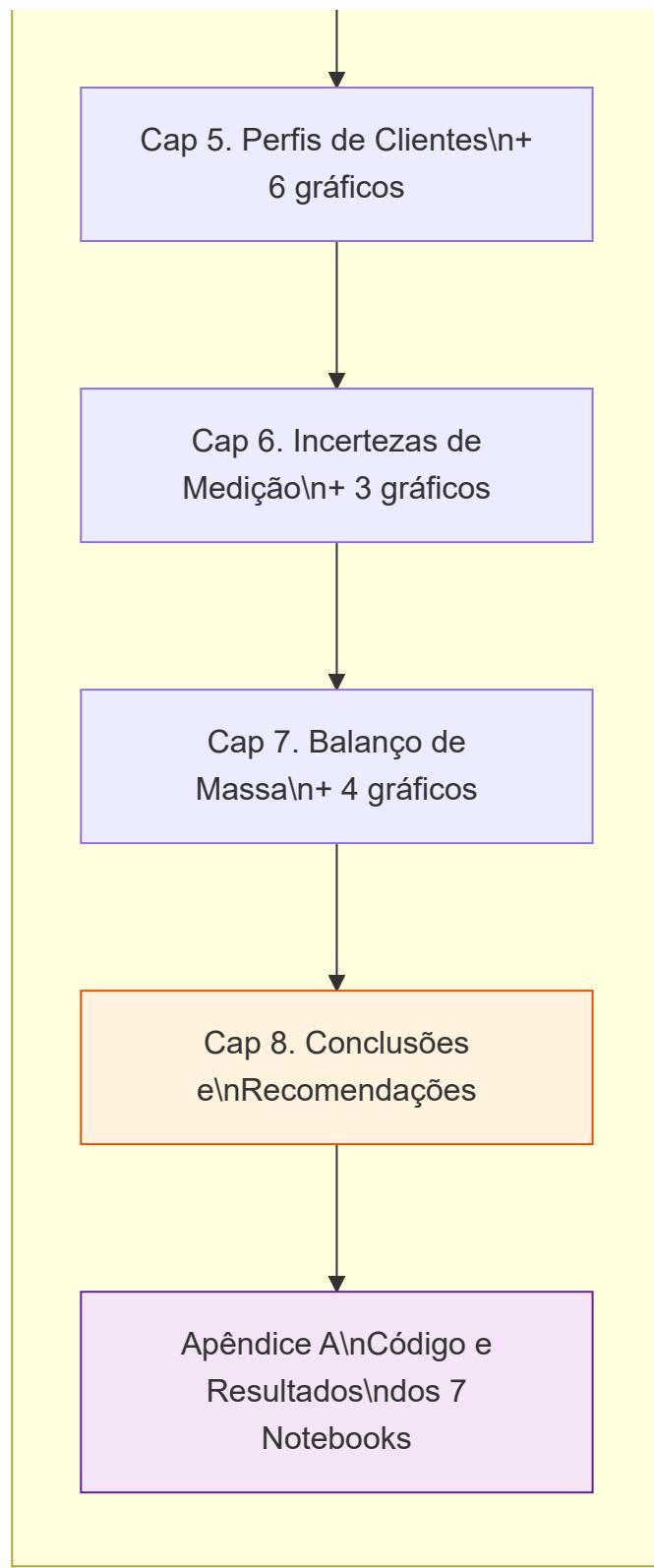


Tipo de célula	Formatação no DOCX
Markdown	Texto normal (headings, bullets, bold/italic)
Código Python	Consolas 8pt, fundo #F5F5F5, borda esquerda azul, label In [N]:
Saída texto	Consolas 8pt, fundo #F0F8F0, borda esquerda verde, label Out:
Imagem (gráfico)	Decodificada de base64, centralizada, width=5"

## Produto Final

*Estrutura do Relatório DOCX (9.3 MB)*





### *Inventário de Gráficos por Capítulo*

Capítulo	Gráficos	Arquivos
----------	----------	----------

Cap 1	3 diagramas	estrutura_distrito.png, fluxo_auditoria.png, processo_analise.png
-------	----------------	---

Cap 2	4	vol_entrada_serie.png, vol_entrada_diferencias.png, vol_entrada_histograma.png, vol_entrada_boxplot.png
-------	---	--

Capítulo	Gráficos	Arquivos
Cap 3	2	pcs_serie.png, pcs_histograma.png
Cap 4	4	energia_serie.png, energia_diferencias.png, energia_mensal.png, energia_scatter.png
Cap 5	6	clientes_participacao.png, clientes_serie.png, clientes_perfil_horario.png, clientes_heatmap.png, clientes_pressao_temp.png, clientes_boxplot.png
Cap 6	3	incertezas_barras.png, incertezas_rss.png, incertezas_contribuicao.png
Cap 7	4	balanco_barras.png, balanco_waterfall.png, balanco_bandas.png, balanco_dashboard.png
<b>Total</b>	<b>26</b>	23 gráficos + 3 diagramas

## Inventário Completo de Artefatos

Tipo	Quantidade
Notebooks Jupyter	7
Gráficos PNG	23
Diagramas PNG	3
Tabelas de dados no DOCX	6
Equações OMML nativas	~30+
Chamadas API Gemini (texto)	28
Chamadas API Gemini (imagem)	3
Arquivos Python	8
Arquivos de metodologia (.md)	7
Arquivos de cache (.md)	28
Relatório DOCX final	9.3 MB
Apresentação PPTX	13 slides

## Arquivos Python do Projeto

```

📁 Cursos ABAR de Dados/
|__ Analise de Condições de Operação de Distrito.xlsx ← Dados brutos

```

└── APOSTILA COMPLETA_Curso ABAR_(...).pdf	← Apostila teórica
└── 01_leitura_e_exploracao.ipynb	← Notebook 1: Leitura
└── 02_analise_volumes_entrada.ipynb	← Notebook 2: Volumes
└── 03_analise_pcs.ipynb	← Notebook 3: PCS
└── 04_calculo_energia.ipynb	← Notebook 4: Energia
└── 05_perfis_clientes.ipynb	← Notebook 5: Clientes
└── 06_sumario_e_incertezas.ipynb	← Notebook 6: Incertezas
└── 07_balanco_massa.ipynb	← Notebook 7: Balanço
└── dados_distrito.py	(~200 linhas) Dataclasses com dados estatísticos
└── gemini_client.py	(~150 linhas) Wrapper API Gemini (texto + imagem)
└── extrair_metodologia.py	(~100 linhas) Extração de teoria do PDF
└── gerar_diagramas.py	(~145 linhas) Geração de diagramas via Gemini Image
└── prompts_auditoria.py	(~450 linhas) 6 system prompts + CHAPTER_CONFIG
└── docx_builder.py	(~920 linhas) Construtor DOCX completo
└── gerar_relatorio_auditoria.py	(~400 linhas) Orquestrador principal (pipeline)
└── gerar_apresentacao.py	(~200 linhas) Gerador de PowerPoint
└── corrigir_notebooks.py	(~100 linhas) Correções automatizadas
└── requirements.txt	Dependências do projeto
└── graficos/	← 23 PNGs exportados dos notebooks
└── diagramas/	← 3 PNGs gerados pelo Gemini Image
└── metodologia/	← 7 .md + equacoes.json extraídos do PDF
└── cache/	← 28 .md (cache granular das chamadas LLM)
└── Relatorio_Auditoria_Distrito_v4.docx	← PRODUTO FINAL (9.3 MB)
└── Apresentacao_Curso_ABAR.pptx	← Apresentação (13 slides)

## Pipeline de Execução (Comandos)

```
# 1. Instalar dependências
pip install -r requirements.txt

# 2. Executar notebooks (gera gráficos em graficos/)
jupyter nbconvert --to notebook --execute 01_leitura_e_exploracao.ipynb
jupyter nbconvert --to notebook --execute 02_analise_volumes_entrada.ipynb
# ... (repetir para 03-07)

# 3. Extrair metodologia da apostila
python extrair_metodologia.py --api-key SUA_CHAVE_GEMINI

# 4. Gerar diagramas de processo
python gerar_diagramas.py --api-key SUA_CHAVE_GEMINI

# 5. Gerar relatório completo (28 chamadas API, ~18 min)
python gerar_relatorio_auditoria.py --api-key SUA_CHAVE_GEMINI

# 5b. Retomar de onde parou (usa cache)
python gerar_relatorio_auditoria.py --api-key SUA_CHAVE_GEMINI --resume
```

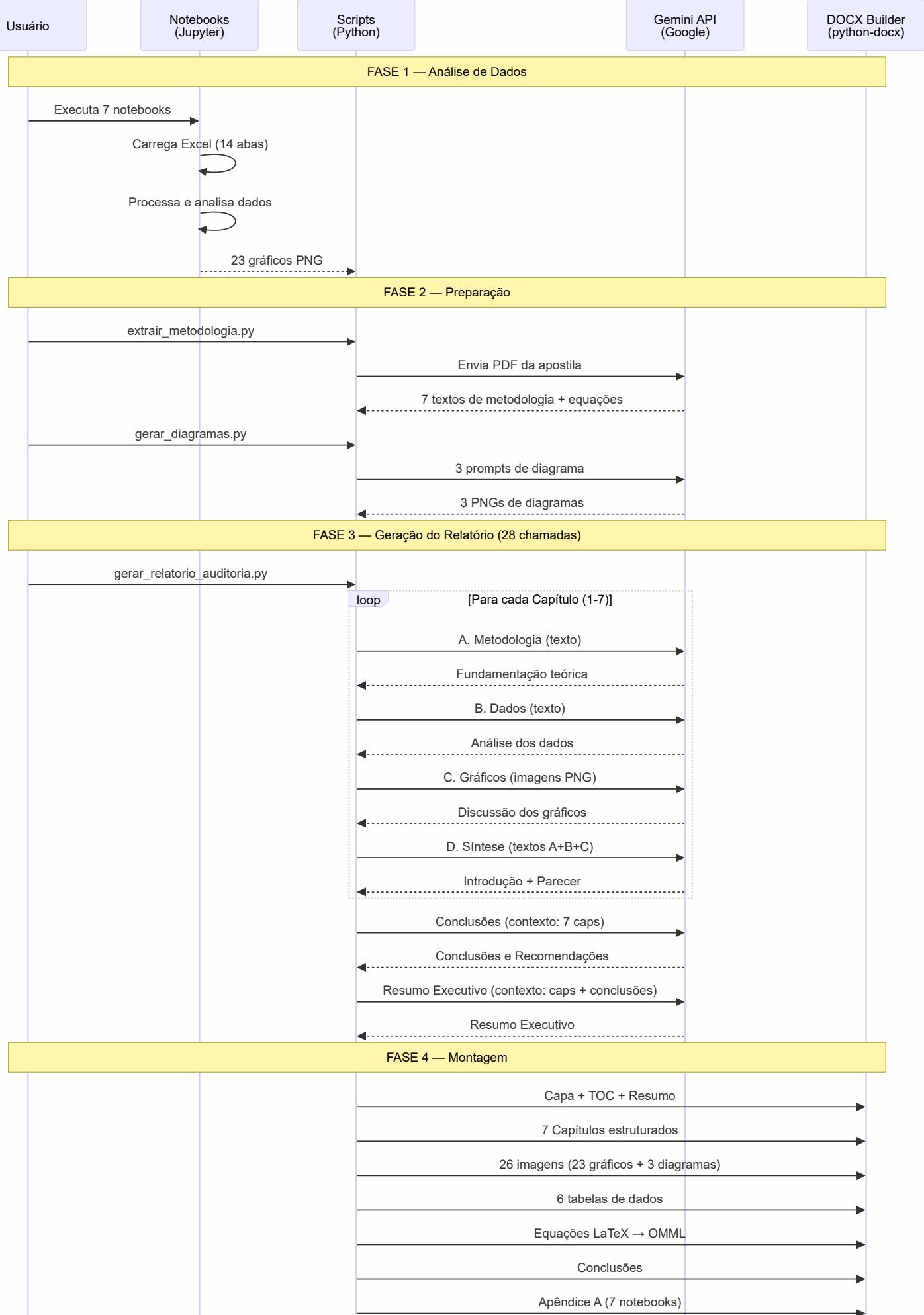
```
# 5c. Apenas remontar DOCX sem chamar API (usa cache completo)
```

```
python gerar_relatorio_auditoria.py --api-key SUA_CHAVE_GEMINI --montar
```

```
# 6. Gerar apresentação PowerPoint
```

```
python gerar_apresentacao.py
```

## Fluxo Completo End-to-End



Relatorio\_Auditoria\_Distrito\_v4.docx (9.3 MB)

Usuário

Notebooks  
(Jupyter)

Scripts  
(Python)

Gemini API  
(Google)

DOCX Builder  
(python-docx)