GRANDES PASSOS DA IMPLEMENTAÇÃO

(versão prática e objetiva, pronta para virar roteiro de execução)

PARTE 1 — Implementação LOCAL

1. Criar a estrutura de pastas do projeto

Onde isso aparece no seu plano: início da página 1 do *Plano do Projeto* (estrutura do repo) .

Tarefas:

- Criar `src/ingestion_api` , `src/crawler` , `src/transformations` , `src/metadata` , `src/analysis` .
- Criar `docs/` com os templates dos documentos.
- Criar `cloud/` com arquivos vazios de instruções.

---

2. Implementar módulo de METADATA (DynamoDB abstraction, mas por enquanto local)

Onde está no plano: seção **Metadados de ingestão** (pág. 10–11).

O que fazer localmente:

- Criar funções genéricas:
  - `start_run()` , `end_run()`
  - `save_checkpoint(source, value)`
  - `load_checkpoint(source)`
- No ambiente local, você pode mockar DynamoDB (arquivo JSON local).

---

3. Implementar INGESTÃO da World Bank API (RAW)

Onde isso está no plano:

- Seção RAW 1.1 (pág. 4)
- Seção Incremental ingestion – estratégia detalhada (pág. 5)

Tarefas:

1. Função para listar anos disponíveis para o indicador.
2. Carregar checkpoint `last_year_loaded_world_bank`.
3. Ingerir somente anos > checkpoint.
4. Salvar arquivo RAW no formato JSONL.
5. Calcular `record_hash`.

---

4. Implementar PROCESSAMENTO da World Bank API

Onde isso está no plano: seção 2.1 (pág. 5)

Tarefas:

- Converter JSON bruto para schema:
  - `country_code`, `country_name`, `year`, `gdp_per_capita_usd`, etc.
- Garantir tipagem correta.
- Criar DataFrame particionado por ano.
- Salvar como PARQUET (processed).

---

5. Implementar o CRAWLER da Wikipedia (RAW)

Onde está no plano: seção RAW 1.2 (pág. 5)

Tarefas:

- Baixar HTML.
- Identificar tabela correta.
- Detectar e tratar inconsistências:
  - footnotes (`[a]`), `–`, valores faltantes etc.
- Salvar HTML + JSON "sujo" em RAW.
- Calcular `record_hash`.

---

6. Implementar PROCESSAMENTO da tabela de CO_2

Onde está no plano:

- seção 2.2 + subseções (pág. 6–9)

Tarefas:

1. **Criar função** `normalize_name()`.
2. **Despivotar 2000 e 2023** (2.2.1).
3. **Criar tabela de mapping de países** (2.2.2 e 2.2.3):
   - mapping base via World Bank
   - aplicar overrides
4. Popular campos:
   - `country_code`
   - `country_name`
   - `year`
   - `co2_tons_per_capita`
5. Salvar PARQUET.

---

7. Implementar a CURATED LAYER

Onde está no plano: seção 3.1 (pág. 10–11)

Tarefas:

- Join por `(country_code, year)`.
- Calcular campo derivado `co2_per_1000usd_gdp`.
- Preencher campos de auditoria:
  - `first_ingestion_run_id`
  - `last_update_run_id`
  - `last_update_ts`
- Salvar particionado por ano + snapshot_date.

---

8. Implementar o Analytical Output

Onde está no plano: início do documento (Analytical Output) e seção Analytics no S3 (pág. 12–13)

Tarefas:

1. **Scatter (gdp_vs_co2_scatter.png)**.

2. **Tabela correlation_summary.csv**:
    - correlação anual
    - top5 países mais/menos eficientes (co_2 por 1000 USD).

---

9. Implementar o "ORQUESTRAÇÃO LOCAL" (entrypoint local)

Onde isso se encaixa: subentendido na seção AWS Lambda Flow (pág. 14) — você apenas simula localmente.

Tarefas:

- Criar script `run_pipeline_local.py`.
- Rodar em sequência:
    1. Ingest API
    2. Process API
    3. Crawl Wikipedia
    4. Process Wikipedia
    5. Build mapping
    6. Build curated
    7. Generate analytics
    8. Save metadata

✅ PARTE 2 — Preparar para levar à AWS (infra + adaptação do código)

10. Criar bucket S3 e projetar a estrutura de prefixos

Onde isso está no plano: seção Storage → S3 (pág. 12–13)

Tarefas:

- Criar bucket `env-econ-pipeline-data`.
- Criar pastas:

- `raw/`
- `processed/`
- `curated/`
- `analytics/`

---

11. Criar tabela DynamoDB real

Onde está no plano: seção 4. Metadata & Incremental (pág. 14–16).

Tarefas:

- Criar tabela `env_econ_pipeline_metadata`.
- Criar itens iniciais:
  - checkpoint API
  - checkpoint WIKI

---

12. Ajustar caminho dos arquivos para S3

Após criação do bucket:

- substituir caminhos locais por S3
- usar `boto3` para upload/download.

---

13. Criar função AWS Lambda

Onde está no plano: seção Compute → AWS Lambda (pág. 13–14).

Tarefas:

- Criar Lambda `env-econ-pipeline-lambda` (Python 3.11).
- Empacotar dependências (layer ou ZIP).
- Criar handler:
  - chamar mesma sequência do `run_pipeline_local.py`.

---

14. Criar IAM Role do Lambda

Onde está no plano: seção IAM (pág. 16–18).

Permissões necessárias:

- S3 read/write
- DynamoDB read/write
- CloudWatch logs

---

15. Criar regra EventBridge (agendamento diário)

Onde está no plano: seção Orchestration / EventBridge (pág. 14).

Tarefas:

- Criar regra:
  - `env-econ-pipeline-daily`
  - `cron(0 2 * * ? *)`
- Apontar para o Lambda.

---

16. Testar execução na AWS

Teste completo:

- executar manualmente
- verificar logs no CloudWatch
- conferir arquivos no S3
- conferir checkpoints atualizados no DynamoDB

---

17. Preencher os documentos finais

Onde está no case: seção Deliverables (pág. 3–5 do Case.pdf)

Onde está no plano: docs/ (pág. 1 do *Plano do Projeto*)

Documentos a finalizar:

- README.md
- docs/final_documentation.md (com o diagrama)
- docs/presentation.pdf
- cloud/instructions.md

🎯 RESUMO SUPER OBJETIVO

Aqui em 10 passos:

1. Criar estrutura do projeto
2. Implementar metadata
3. Ingestão API (RAW)
4. Processamento API (PROCESSED)
5. Crawler Wikipedia (RAW)
6. Processamento Wikipedia (PROCESSED)
7. Curation (join + indicadores)
8. Analytical outputs
9. Orquestração local
10. Infra + deploy AWS (S3, DynamoDB, Lambda, EventBridge)
11. Documentação + apresentação
