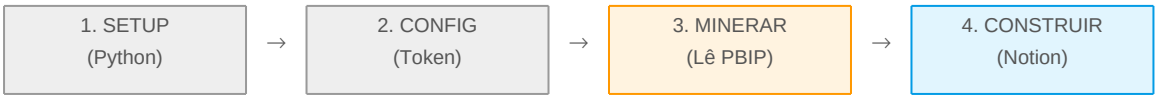
Protocolo Management Data: Operation Setup

Automação de Inventário – Power BI Inventory Genesis v2.0 Elite (Nov/2025)



0. O que é esse Framework?

Primeiro Inventário Automatizado de Projetos de PBI que você está vendo - de graça quero dizer! Uma solução robusta e ELC - Extremely Low Cost

Se você chegou até aqui, provavelmente cansou de:

- Criar ou Atualizar inventário na mão usando planilhas
- Caçar medida perdida no Power BI
- Explicar 300 vezes “de onde vem esse KPI?”

Esse framework faz isso por você. Ele monta, de forma automática:

- Inventário de Tabelas, Colunas, Relacionamentos
- Inventário de Medidas DAX (com metadados)
- Inventário de Páginas e Visuais do relatório e muito mais
- Toda essa inteligência organizada dentro do Notion

Pipeline em 3 camadas + IA opcional:

Minerador (`minerador_pbi.py`)
Lê o projeto `.pbip`, varre TMDL/JSON e monta os arquivos de inventário técnico.

Constructor (`constructor_notion.py`)
Lê esses arquivos (e, se configurado, os arquivos enriquecidos pela IA) e constrói as bases/páginas no Notion.

Pós-processamento (`notion_post_links_ids.py`)
Varre o Notion e cria **links clicáveis** das Medidas nos registros de Páginas/Visuais,

deixando a navegação entre objetos muito mais fluida.

IA opcional
Quando ativada, a IA gera descrições e textos auxiliares das medidas, usando arquivos como:

5. `model_structure.json`
6. `measures_for_ai.csv`
7. `measures_enriched.csv`

O objetivo deste tutorial é simples:
qualquer pessoa, mesmo sem ser de TI, conseguir rodar o pipeline **do zero até o Notion cheio** – com ou sem IA.

---

1. Checklist de Decolagem (Pré-requisitos)

Antes de apertar qualquer botão, confirme se você tem:

- [OK] **Windows com permissão para instalar programas**
- [OK] **Power BI Desktop** atualizado
- [OK] Capacidade de salvar o relatório em formato **Projeto do Power BI (.pbip)**
- [OK] Uma conta no **Notion** (pode ser gratuita)
- [OK] Uma conta no **GitHub** (onde está o repositório)
- [OK] **Python** instalado
- [OK] Um editor de código (recomendado: **VS Code**)

Se qualquer item acima for “não”, resolve primeiro.
Sem isso, o foguete nem liga.

---

2. Instalando as Ferramentas

2.1. Python (o motor)

1. Acesse: https://www.python.org/downloads/
2. Baixe a versão estável mais recente (Windows).
3. Rode o instalador.

4. **Importantíssimo:** na primeira tela, marque a opção
   `[x] Add Python to PATH`
   Se não marcar, nada funciona.
5. Clique em **Install Now** e aguarde.

> *Dica: se já tiver Python instalado, garanta que o comando `python --version` funciona no Terminal (PowerShell ou CMD).*

2.2. VS Code (o painel de controle)

1. Baixe em: https://code.visualstudio.com/
2. Instale normalmente (o velho next, next, next, finish).
3. Abra o VS Code.
4. Vá na aba de extensões (ícone de quadradinhos).
5. Instale a extensão **“Python”** (Microsoft).

---

3. Preparando o Notion (Token + Página)

O constructor e o script de links precisam de um “crachá” para acessar seu Notion.

3.1. Criar a integração (o robô)

1. Acesse: https://www.notion.so/my-integrations
2. Clique em **New integration**.
3. Sugestão de nome: `PowerBI_Inventory`.
4. Selecione o workspace correto no Notion que você quer que o Inventário seja construído ou faça isso depois (detalhes na sessão 3.2, a próxima).
5. Clique em **Submit**.
6. Copie o **Internal Integration Token** (começa com `ntn_...`).

> *Guarde esse token com carinho. Ele é a senha do robô e você vai precisar informar durante esse processo.*

3.2. Conectar a integração à página

1. No Notion, crie uma página chamada, por exemplo:
   “HUB – Inventários Power BI”.
2. No canto superior direito da página, clique em **•••**.
3. Vá em **Connections** (ou **Add connections**).
4. Selecione a integração `PowerBI_Inventory` / `PowerBI_Inventory_Genesis`.
5. Confirme.

Sem esse passo, o script consegue ver o workspace, mas **não entra na página**.

3.3. Descobrindo o ID da página

1. Com a página aberta, copie o link da URL.
2. O ID é o trecho final, com 32 caracteres, algo assim:
   `https://www.notion.so/HUB-...-1a2b3c4d5e6f78901234567890abcdef`
3. O que interessa é:
   `1a2b3c4d5e6f78901234567890abcdef`

Anote esse ID. Vamos usar já já.

---

4. Organizando as Pastas

Você pode seguir o padrão que já vem no projeto ou adaptar.
Sugestão simples:

- **PASTA_FRAMEWORK**
  Onde está o repositório clonado / extraído do GitHub.

  Dentro dela, ficam os arquivos principais que você vai usar:

  `pbi_config.json`
- `minerador_pbi.py`
- `constructor_notion.py`
- `notion_post_links_ids.py`

Se quiser separar scripts em uma pasta dedicada (ex.: `C:\Scripts\Automacao_BI`), tudo bem, mas então lembre de:

- Copiar **os scripts** para essa pasta, ou
- Ajustar os caminhos quando rodar os comandos `python ...`.

Para quem está começando, a recomendação é:
não mexer na estrutura do repositório. Só usar como está.

---

5. Configurando o Projeto Power BI

5.0. Habilitando o formato Projeto (.pbip) no Power BI

Para conseguir salvar seu relatório como **Projeto do Power BI (.pbip)**, pode ser que você precise ativar esse recurso:

1. Abra o **Power BI Desktop**.
2. Vá em **Arquivo > Opções e configurações > Opções**.
3. No menu à esquerda, procure por algo como **Recursos em Pré-visualização** ou **Preview features**.
4. Marque a opção relacionada a **"Salvar arquivos como projeto do Power BI"** ou **"Power BI Project (.pbip)"** (o nome exato pode mudar conforme a versão).
5. Clique em **OK**.
6. Feche e reabra o Power BI Desktop.

Depois disso, ao usar **Arquivo > Salvar como**, a opção **"Projeto do Power BI (*.pbip)"** deve aparecer.

5.1. Convertendo o PBIX para PBIP

1. Abra o relatório no **Power BI Desktop**.
2. Vá em **Arquivo > Salvar como**.
3. Selecione uma pasta dedicada para esse projeto, por exemplo:
   `C:\Users\SeuUsuario\Documents\Projetos_PBI\SeuProjeto`
4. No tipo de arquivo, escolha **"Projeto do Power BI (*.pbip)"**.
5. Salve.

Isso cria uma pasta com a estrutura do projeto (arquivos `.pbip`, subpastas, JSON etc.).
Vamos chamar essa pasta de **PASTA_PROJETO_PBIP**.

5.2. Criando o arquivo de configuração do projeto (`pbi_config.json`)

Dentro da **PASTA_PROJETO_PBIP**, crie um arquivo chamado:

`pbi_config.json`

Com o conteúdo base:

```
{ "project_name": "HR Board KPIs", "project_link": "",
"use_ai_enrichment": false, "ai_model": "gemini-2.5-flash" }
```

`project_name`
Nome amigável do projeto (vai para o Notion).
Exemplo: `"HR Board KPIs"`.

`project_link`
Link opcional para esse projeto em outro lugar (App Power BI, SharePoint, etc.).
Pode ficar vazio (`""`) se você não tiver nada pra colocar.

`use_ai_enrichment`

- `false` → **IA desligada.** É o modo mais seguro pra começar.

  `true` → **IA ligada.** O framework vai tentar enriquecer as medidas usando a API do Google.

  `ai_model`
  Nome do modelo de IA que será usado quando `use_ai_enrichment` for `true`.
  Exemplos típicos:

- `"gemini-2.5-flash"` → modelo rápido, mais barato, ótimo pra esse tipo de tarefa. **Recomendado.**
- `"gemini-2.5-pro"` → modelo mais potente, também mais caro. Use só se fizer sentido, mas sinceramente não precisa.

> *Resumindo:*
>
> - *Quer rodar **sem IA**?*
>   *Deixe assim:*
>
> *`"use_ai_enrichment": false`*

> *Quer rodar **com IA**?*
> *Ajuste para:*

- `"use_ai_enrichment": true`
- `"ai_model": "gemini-2.5-flash"` *(sugestão custo/benefício)*

O minerador e o constructor usam esse arquivo como “cartão de visita” do projeto **e** como chave de liga/desliga da IA.

---

6. [SENHA] Configurando o acesso ao Notion nos scripts

Os scripts que falam com o Notion (`constructor_notion.py` e `notion_post_links_ids.py`) precisam saber:

- Qual é o **TOKEN** da integração
- Qual é o **ID da página HUB**

Geralmente, isso é feito com variáveis no topo dos arquivos.

1. Abra o arquivo `constructor_notion.py` no VS Code.
2. No topo do arquivo, procure pelas variáveis de configuração, algo como:

```
NOTION_TOKEN = "ntn_...cole_aqui_seu_token..." NOTION_ROOT_PAGE_ID = "EX: 1a2b3c4d5e6f78901234567890abcdef"
```

1. Substitua:
2. `NOTION_TOKEN` pelo token da integração.
3. `NOTION_ROOT_PAGE_ID` pelo ID da página HUB.
4. Salve o arquivo.

Repita o mesmo ajuste, se necessário, no `notion_post_links_ids.py`, seguindo os comentários dentro do próprio script.

> *Em versões futuras, isso pode migrar para variáveis de ambiente ou um arquivo `.env`. Por ora, manter direto nos scripts deixa mais simples para o usuário leigo.*

---

7. (Opcional) IA para enriquecer as Medidas

Além do inventário “cru”, o framework pode usar **IA (Google)** para enriquecer as medidas DAX com:

- descrições em linguagem natural
- resumos técnicos
- tags ou categorias auxiliares

7.1. Arquivos auxiliares da IA

Durante o processo, alguns arquivos podem aparecer na pasta de saída / trabalho do framework (aquela do seu projeto do PBI):

- `model_structure.json`
- Export da estrutura do modelo (tabelas, colunas, relacionamentos).

  Serve tanto para auditoria técnica quanto como insumo para prompts de IA.

  `measures_for_ai.csv`
- Lista das medidas que serão enriquecidas pela IA.

  Gerado automaticamente a partir do inventário de medidas (você **não** cria na mão).

  `measures_enriched.csv`
- Resultado do enriquecimento.
- Contém as medidas com campos adicionais (descrições, resumos etc.).
- Usado pelo `constructor_notion.py` para preencher colunas no Notion, quando disponível.

Se a IA **não** estiver configurada, o framework ainda funciona:

- O inventário padrão é montado normalmente.
- Esses arquivos podem não ser gerados ou não serão consumidos.
- Você só perde a “camada extra” de texto gerado pela IA.

7.2. Configurando a API do Google (passo a passo simples)

> *Só siga esta parte se você **realmente quer ligar a IA**.*
> *Se a ideia é só testar o framework, pule a IA por enquanto.*

Passo 1 – Criar a chave no Google

1. Vá até o produto do Google que fornece o modelo de IA (Google AI Studio / Gemini).
2. Crie um projeto (se ainda não tiver).
3. Gere uma **API Key** (chave de API).
4. Copie essa chave para um lugar seguro (um bloco de notas temporário, por exemplo).

Passo 2 – Instalar a biblioteca Python da IA

No mesmo terminal onde você vai rodar o framework (VS Code, por exemplo):

```
python -m pip install google-generativeai
```

Essa biblioteca é a "ponte" entre os scripts Python e o modelo Gemini.

Passo 3 – Guardar a API Key numa variável de ambiente (Windows)

A ideia aqui é **não deixar a chave escrita no código**.

1. No Windows, pesquise por **"Variáveis de ambiente"** no menu Iniciar.
2. Abra **Editar as variáveis de ambiente do sistema**.
3. Clique em **Variáveis de ambiente...**.
4. Em **Variáveis de usuário**, clique em **Novo...**.
5. Preencha:
6. **Nome da variável:** `GEMINI_API_KEY`
7. **Valor da variável:** cole aqui a chave que você copiou no Passo 1.
8. Confirme em **OK** em todas as janelas.

Depois disso, qualquer script Python que faça algo como:

```
import os api_key = os.getenv("GEMINI_API_KEY")
```

vai conseguir ler a chave sem você precisar editar o código.

Passo 4 – Ligar a IA no `pbi_config.json`

No arquivo `pbi_config.json` (dentro da PASTA_PROJETO_PBIP), ajuste:

```
{ "project_name": "HR Board KPIs", "project_link": "",
"use_ai_enrichment": true, "ai_model": "gemini-2.5-flash" }
```

- Se `use_ai_enrichment` estiver `true` **e** a variável `GEMINI_API_KEY` existir **e** a biblioteca `google-generativeai` estiver instalada, o framework vai tentar enriquecer as medidas.
- Se qualquer uma dessas coisas faltar ou der problema, o comportamento esperado é o script:
- logar um erro/aviso,
- e seguir o fluxo base **sem IA**, pra não travar o pipeline.

> *Dica de custo:*
> *- Comece com `"gemini-2.5-flash"` → rápido e mais barato.*
> *- Só vá para `"gemini-2.5-pro"` se você realmente precisar de respostas mais sofisticadas e estiver confortável com custo maior.*

---

8. Hora de rodar o pipeline

8.1. Abrindo o terminal na pasta do framework

1. Abra o **VS Code**.
2. Vá em **File > Open Folder...** e selecione a **PASTA_FRAMEWORK**
   (a pasta onde estão `minerador_pbi.py`, `constructor_notion.py` etc.).
3. Abra o terminal integrado: `Ctrl + Shift + '` (ou `Ctrl + '`).

Você deve ver algo como:

```
PS C:\Users\SeuUsuario\Documents\powerbi-inventory-genesis>
```

Se o terminal estiver em outra pasta, use o comando `cd` para ir até a PASTA_FRAMEWORK, por exemplo:

```
cd C:\Users\SeuUsuario\Documents\powerbi-inventory-genesis
```

8.2. Instalando dependências (primeira vez)

No terminal (já apontando para a PASTA_FRAMEWORK), rode:

```
python -m pip install --upgrade pip python -m pip install requests markdown xhtml2pdf
```

- `requests` → usado para falar com a API do Notion.
- `markdown` e `xhtml2pdf` → podem ser usados em módulos de documentação / suporte.

Se você **ligou a IA** (seção 7):

```
python -m pip install google-generativeai
```

Se já estiver tudo instalado, esses comandos só vão confirmar/atualizar.

8.3. Missão 1 – Minerar o PBIP

1. Garanta que o terminal está na **PASTA_FRAMEWORK**
   (se não estiver, use `cd` como mostrado em 8.1).
2. No terminal, rode:

```
python minerador_pbi.py
```

Esse processo é mais rápido, só se ajeita na cadeira aí e aguarda ...

O script pode:

- Perguntar o caminho da **PASTA_PROJETO_PBIP**, ou
- Ler esse caminho de alguma configuração interna.

Siga as instruções exibidas na tela.

Ao final, você deve ver algo como:

```
--- MINERADOR CONCLUÍDO --- Arquivos gerados em: ...
```

Esses arquivos normalmente incluem:

- Estrutura do modelo (tabelas, colunas, relacionamentos)
- Inventário de medidas
- Informações de páginas e visuais

- E, quando a IA está ativa, arquivos auxiliares como `model_structure.json`, `measures_for_ai.csv` e `measures_enriched.csv`.

8.4. Missão 2 – Construir o inventário no Notion

1. Com o terminal ainda na **PASTA_FRAMEWORK**, rode:

```
python constructor_notion.py
```

Agora você pode sair da frente da tela e ir pegar um café, respirar, encher sua garrafinha de água, rs ...

O que esse script faz:

- Lê os arquivos gerados pelo minerador (e, se existirem, os arquivos enriquecidos pela IA).
- Cria ou atualiza bancos de dados na página HUB do Notion.
- Preenche as tabelas com:
- projetos
- tabelas
- colunas
- medidas DAX
- visuais, páginas etc.

Ao final, espere por uma mensagem de sucesso, algo como:

```
SUCESSO FINAL – Inventário atualizado no Notion.
```

Se aparecer mensagem de erro, consulte a seção de **Erros Comuns**.

8.5. Missão 3 – Criar links clicáveis das Medidas no Notion

Para facilitar a navegação, o framework conta com um passo extra de pós-processamento.

1. Ainda na **PASTA_FRAMEWORK**, rode:

```
python notion_post_links_ids.py
```

Esse script:

- Lê o banco de **Medidas** no Notion e captura a URL de cada medida.
- Lê o banco de **Páginas/Visuais**, onde estão listados os IDs das medidas usadas em cada página.
- Atualiza um campo de texto no Notion (por exemplo, `Medidas · Links`) transformando cada ID em um **link clicável** para a respectiva medida.

Resultado prático:

- Na página de cada visual/página do relatório, você consegue clicar no ID da medida e cair direto no registro técnico da medida no Notion.

Se esse script não for executado, o inventário continua válido, mas você perde essa navegação “one-click”.

---

9. O que você deve ver no Notion

Depois de rodar o `constructor_notion.py` e o `notion_post_links_ids.py`, volte na página HUB:

Você deve encontrar, por exemplo:

- Um banco de dados de **Projetos** (um registro por projeto PBIP)
- Um banco de dados de **Tabelas**
- Um banco de dados de **Medidas**
- Um banco de dados de **Visuais/Páginas**
- Relacionamentos entre esses bancos (links / relations do Notion)
- Em Páginas/Visuais, uma coluna com **IDs de Medidas clicáveis**, apontando para os registros de Medidas

A ideia é que o Notion vire **o cérebro documental** dos seus relatórios Power BI.

---

10. ■ Erros comuns e como resolver

Alguns problemas clássicos:

1. **Python não é reconhecido**
2. Mensagem: `'python' não é reconhecido como um comando interno...`

3. Causa: opção **Add Python to PATH** não marcada na instalação.

   Caminho rápido:

   - Reinstalar o Python marcando o PATH, ou
   - Usar `py` no lugar de `python` nos comandos.

   Erro de conexão com o Notion (401 / 403)

   Verifique se:

   - O token começa com `ntn_`.
   - O token está correto no(s) script(s).
   - A página HUB está conectada à integração certa.

   Nada aparece no Notion

   Confirme se:

   - O minerador gerou os arquivos de saída.
   - O caminho que o constructor está usando aponta para a saída certa.
   - O `pbi_config.json` existe na pasta do projeto PBIP.

   Links de Medidas não aparecem ou não funcionam

   Garanta que:

   - O `constructor_notion.py` rodou sem erro.
   - O `notion_post_links_ids.py` foi executado depois.
   - O token e o ID da página HUB nesse script estão corretos.

   Problemas com IA (erros de API)

12. Verifique se:

    - A API Key do Google está correta.
    - A biblioteca Python usada pela IA (`google-generativeai`) está instalada.
    - Você configurou a variável `GEMINI_API_KEY` exatamente com a chave gerada.

13. Se estiver com pressa, desative a IA no `pbi_config.json` e rode só o pipeline base.

---

11. Debriefing

Se tudo rodou certo, você ganhou:

- Um inventário completo do seu modelo Power BI dentro do Notion
- Medidas documentadas, com possibilidade de enriquecimento por IA
- Links clicáveis entre Páginas/Visuais e Medidas
- Um pipeline replicável para qualquer novo projeto PBIP
- A base para uma governança séria de DAX, Tabelas e Visuais

Agora, você decide:

- Rodar isso em mais projetos
- Ajustar o schema do Notion
- Contribuir com melhoria de scripts e documentação (chama o Jimmy pra bater aquele papo)
- Levar essa automação pra dentro da sua equipe

Você oficialmente deixou de ser “refém do PBIX” para virar **Comandante dos Dados**.
Agora respira fundo e vai tomar outro café. café■ rs

Powered by Data Management Team - HAVAS Brazil – Power BI Inventory Genesis