DOCUMENTAÇÃO – POKEDEX AUTOMATIZADA COM N8N

AUTOR: SAVYO LIMA PAIXÃO

1. OBJETIVO

O objetivo deste projeto foi automatizar a coleta de informações da **PokéAPI** e organizar esses dados em uma **planilha**, trazendo de forma estruturada:

- Game Index
- Nome
- Tipo
- Pré-evolução
- Evolução
- Evolução final
- URL da cadeia de evolução

2. FERRAMENTAS UTILIZADAS

- **n8n** → para orquestrar toda a automação.
- **PokéAPI** → fonte oficial de informações sobre Pokémon.
- **Google Sheets / Excel** → armazenamento e visualização dos dados.

3. ESTRUTURA DO FLUXO

- O fluxo no **n8n** foi dividido em etapas principais:
- **Entrada da planilha**
  - Leitura da coluna *Game Index* para definir quais Pokémon buscar.
- **Requisição de dados do Pokémon (HTTP Request)**
  - Consulta ao endpoint da PokéAPI para trazer nome, tipo e evoluções.
- **Requisição da cadeia de evolução (HTTP Request)**
  - Busca a URL da cadeia de evolução e retorna todos os possíveis estágios evolutivos.

- **Filtro de dados (Function Node)**
    - Aplicação das regras:
        - Identificar corretamente pré-evolução, evolução e evolução final.
        - Evitar duplicação (ex.: Charizard não mostrar Charizard de novo como evolução).
        - Padronizar saída mesmo em casos de Pokémon sem evolução.
- **Normalizador**
    - Correção dos nomes.
    - Ordenação por *Game Index* em ordem crescente.
    - Preservação de colunas manuais existentes na planilha.
- **Escrita na planilha**
    - Dados atualizados são gravados, sempre em ordem.

NOTA SOBRE A COLETA DE EVOLUÇÕES VIA URL DA POKEAPI

DURANTE OS TESTES INICIAIS, PERCEBI QUE A PRIMEIRA URL DE EVOLUÇÃO (POKEMON-SPECIES) TRAZIA APENAS A PRÉ-EVOLUÇÃO DO POKÉMON, MAS NÃO INFORMAVA A CADEIA COMPLETA DE EVOLUÇÕES.

EXEMPLO: AO CONSULTAR O *CHARMELEON*, A RESPOSTA MOSTRAVA APENAS QUE ELE TINHA COMO PRÉ-EVOLUÇÃO O *CHARMANDER*, MAS NÃO TRAZIA INFORMAÇÕES DE QUE ELE EVOLUI PARA O *CHARIZARD*.

PARA RESOLVER ISSO:

FOI NECESSÁRIO REALIZAR UM SEGUNDO REQUEST PARA A URL DA EVOLUTION-CHAIN, QUE RETORNA TODA A ÁRVORE EVOLUTIVA.

QUANDO ANALISEI ESSA ESTRUTURA, PERCEBI QUE ELA PODERIA CONTER:

PRÉ-EVOLUÇÃO

EVOLUÇÃO DIRETA

EVOLUÇÃO FINAL

E, EM ALGUNS CASOS, MÚLTIPLAS RAMIFICAÇÕES (COMO O EEVEE).

COM ISSO, DEFINI A REGRA DE NÃO LIMITAR A APENAS UMA EVOLUÇÃO, MAS CONSIDERAR TAMBÉM A EVOLUÇÃO FINAL. DESSA FORMA, A POKEDEX FICA MAIS COMPLETA E CONSISTENTE, SEM PERDER DADOS IMPORTANTES DA ÁRVORE.

4. TESTES REALIZADOS

Foram realizados testes com:

- **Charmander (com 2 evoluções)** → Charmander → Charmeleon → Charizard.
- **Eevee (com múltiplas evoluções)** → validação de pré-evolução.
- **Pokémon sem evolução (ex.: Cinderace)** → retorna “Sem evolução” e “Sem evolução final”.

5. REGRAS IMPLEMENTADAS

- Se o Pokémon não tiver evolução → retorna *“Sem evolução”*.
- Se for o último da linha evolutiva → retorna *“Sem evolução final”*.
- Se o nome do Pokémon se repetir em qualquer estágio → retorna *“Sem evolução”*.
- Pré-evolução corrigida (caso como *Leafeon*, que inicialmente aparecia como “Sem pré-evolução”).
- Todos os nomes normalizados → primeira letra maiúscula, hífen e underline substituídos por espaço.

NOTA:

DURANTE OS TESTES SURGIRAM ALGUNS PROBLEMAS QUE EXIGIRAM REGRAS EXTRAS:

REPETIÇÃO DE NOMES

EM ALGUNS CASOS, O MESMO POKÉMON APARECIA COMO SUA PRÓPRIA PRÉ-EVOLUÇÃO OU EVOLUÇÃO FINAL.

PARA CORRIGIR, IMPLEMENTEI UMA REGRA QUE SUBSTITUI O VALOR REPETIDO POR *"SEM EVOLUÇÃO"* OU *"SEM PRÉ-EVOLUÇÃO"*.

POKÉMON COM MÚLTIPLAS EVOLUÇÕES

A API DA POKEAPI PODE RETORNAR RAMOS DIFERENTES (EXEMPLO: EEVEE PODE EVOLUIR PARA VÁRIAS FORMAS).

PARA SIMPLIFICAR, DEFINIMOS QUE SÓ SERÁ EXIBIDA A PRÉ-EVOLUÇÃO DIRETA E A EVOLUÇÃO FINAL MAIS DISTANTE, EVITANDO LISTAR TODAS.

POKÉMON JÁ NO ESTÁGIO FINAL

EXEMPLO: CHARIZARD.

NESSE CASO, NÃO FAZ SENTIDO O MOSTRAR COMO EVOLUÇÃO FINAL DE SI MESMO, ENTÃO A SAÍDA É *"SEM EVOLUÇÃO FINAL"*.

6. REQUISITOS FUNCIONAIS E NÃO FUNCIONAIS

Requisitos Funcionais

- Coletar os dados principais do Pokémon: nome, tipo, pré-evolução, evolução, evolução final e URL de evolução.
- Normalizar os dados, garantindo capitalização correta e padronização de nomes (hífens e underlines substituídos por espaços).
- Ordenar automaticamente todos os registros pelo campo *game_index* em ordem crescente.
- Evitar duplicação de informações, especialmente em casos onde o nome do Pokémon poderia se repetir como pré-evolução ou evolução.
- Preservar colunas manuais já existentes na planilha, garantindo que não sejam sobrescritas pelo fluxo.

Requisitos Não Funcionais

- Implementar resiliência contra mudanças na PokéAPI, validando a estrutura dos dados antes do processamento.
- Garantir modularidade no fluxo do n8n, permitindo manutenção e futuras expansões sem necessidade de reestruturação completa.
- Assegurar clareza e padronização na apresentação dos dados, com inicial maiúscula em todas as palavras relevantes.
- Considerar escalabilidade do workflow para lidar com um volume maior de registros de Pokémon.
- Possibilitar monitoramento e notificações em caso de falhas, assegurando confiabilidade no processo.

7. DIFICULDADES E SOLUÇÕES

- **Ordem da planilha** → resolvido com normalizador que apaga e recria os dados em ordem crescente.
- **Nomes repetidos** → adicionada regra para substituir por *“Sem evolução”*.

- **API com múltiplos caminhos de evolução** → simplificado para pegar apenas a cadeia principal (padrão linear).
- **Manutenção futura** → adicionada função de validação para detectar mudanças no formato da PokéAPI.

NOTA SOBRE O APRENDIZADO DA FERRAMENTA N8N

NO INÍCIO DO TESTE, TIVE BASTANTE DIFICULDADE EM USAR O N8N, POIS ERA A PRIMEIRA VEZ QUE EU TINHA CONTATO COM A FERRAMENTA. NÃO SABIA EXATAMENTE COMO OS NÓS FUNCIONAVAM NEM COMO MANIPULAR OS DADOS ENTRE ELES.

PARA SUPERAR ISSO, DEDIQUEI UM TEMPO EXTRA ESTUDANDO POR CONTA PRÓPRIA: ASSISTI A VÍDEOS TUTORIAIS E CONSULTEI A DOCUMENTAÇÃO OFICIAL PARA APRENDER O BÁSICO DE CADA NÓ (FUNCTION, MERGE, HTTP REQUEST ETC.).

ESSE PROCESSO DE APRENDIZADO EXIGIU PACIÊNCIA E PERSISTÊNCIA, MAS ME PERMITIU CONSTRUIR CONFIANÇA COM A FERRAMENTA E, PRINCIPALMENTE, CONCLUIR O FLUXO DO TESTE COM TODAS AS REGRAS QUE DEFINI AO LONGO DO DESENVOLVIMENTO.

ALÉM DISSO, APROVEITEI AS DIFICULDADES COMO UMA OPORTUNIDADE PARA EVOLUIR MINHA LÓGICA DE AUTOMAÇÃO, APRENDENDO NÃO SÓ A "FAZER FUNCIONAR", MAS TAMBÉM A PENSAR EM BOAS PRÁTICAS (COMO NORMALIZAÇÃO, ORDENAÇÃO E TRATAMENTO DE EXCEÇÕES).

8. RESULTADOS

- Planilha final contém todos os Pokémon testados organizados.
- Fluxo exportado em JSON (POKEDEX.json) permite reimportar e rodar em outro n8n.
- Processo escalável: basta adicionar novos *Game Index* que os dados serão atualizados automaticamente.

9. PRÓXIMOS PASSOS

- **Expandir as informações capturadas**: incluir habilidades, estatísticas, nível de evolução e geração de cada Pokémon, enriquecendo a base de dados.
- **Implementar monitoramento automático**: criar mecanismos para detectar alterações na estrutura da PokéAPI e ajustar os fluxos de forma preventiva.

- **Adicionar notificações automáticas**: integrar com canais de comunicação (Slack/Email) para alertar em caso de falhas ou inconsistências no fluxo.
- **Realizar testes em escala**: validar o desempenho do fluxo com um volume maior de registros, garantindo estabilidade e confiabilidade.