Tutorial Técnico e Estratégico: Pipeline de Dados para o "Radar de Tendências" na KAIRÓS

Autor: Manus AI

Este tutorial detalhado destina-se a desenvolvedores que desejam construir um pipeline robusto e personalizado para alimentar a página "Radar de Tendências" na plataforma SaaS KAIRÓS. O foco está na utilização do n8n.io para orquestração, APIs de scraping para coleta de dados públicos e Supabase (com `pgvector`) para armazenamento e cálculo de afinidade vetorial.

Introdução

A página "Radar de Tendências" da KAIRÓS tem como objetivo identificar tópicos em alta e, crucialmente, calcular a relevância de cada tópico para as pautas e posicionamentos específicos de cada cliente. Diferente das integrações anteriores com APIs oficiais, este pipeline se baseia na coleta de dados públicos através de APIs não oficiais e serviços de scraping. Esta abordagem prioriza a agilidade e o custo-benefício, permitindo acesso a uma gama mais ampla de dados de tendências que talvez não estejam disponíveis através de APIs oficiais.

Aviso de Risco: A Natureza Frágil do Scraping

É fundamental começar este tutorial com um aviso claro sobre os prós e contras desta abordagem. Embora o uso de APIs de scraping e fontes não oficiais ofereça flexibilidade e acesso a dados valiosos, ele vem com um risco inerente de **instabilidade (brittleness)**. Scrapers e APIs não oficiais podem quebrar a qualquer momento devido a alterações na estrutura dos sites-alvo, atualizações nas políticas das plataformas ou mudanças nas APIs subjacentes. Isso pode levar a interrupções na coleta de dados e exigir manutenção contínua.

Prós:

- **Acesso a Dados Amplos:** Permite coletar dados de tendências de plataformas que não oferecem APIs oficiais ou que restringem o acesso a determinados tipos de dados.

- **Custo-Benefício:** Muitas APIs de scraping oferecem planos mais acessíveis em comparação com as APIs oficiais de grandes plataformas, especialmente para volumes de dados específicos.
- **Flexibilidade:** Maior controle sobre quais dados são coletados e como são estruturados.

Contras:

- **Fragilidade:** Alta suscetibilidade a quebras devido a mudanças no layout ou na estrutura dos sites. Isso exige monitoramento constante e adaptação.
- **Conformidade Legal e Termos de Serviço:** O scraping pode violar os termos de serviço de algumas plataformas, o que pode levar a bloqueios de IP ou ações legais. É crucial revisar os termos de cada fonte.
- **Qualidade dos Dados:** A qualidade e a consistência dos dados podem variar. Pode ser necessário um processamento adicional para normalização e limpeza.
- **Manutenção Contínua:** Requer um esforço de manutenção maior para garantir que os scrapers continuem funcionando.

Estratégias de Mitigação:

- **Fontes de Dados Redundantes:** Utilize múltiplas APIs ou métodos de scraping para a mesma fonte, se possível. Se uma falhar, você terá uma alternativa.
- **Logging Robusto:** Implemente um sistema de logging detalhado no n8n para registrar o sucesso ou falha de cada etapa da coleta de dados.
- **Alertas de Falha:** Configure alertas (por e-mail, Slack, etc.) para ser notificado imediatamente em caso de falhas na coleta de dados, permitindo uma resposta rápida.
- **Monitoramento Contínuo:** Monitore regularmente a qualidade e a disponibilidade dos dados coletados.

Todo o processo de coleta, processamento e ranqueamento deve ocorrer exclusivamente no backend (n8n), em um workflow agendado. O frontend apenas consumirá o resultado final já processado, garantindo a segurança e a performance da aplicação.

Parte 1: Estratégia de Coleta e Ferramentas

Esta secção apresenta sugestões de APIs de scraping para cada fonte de dados necessária (Google Trends, Twitter/X, TikTok e Instagram), comparando-as em termos de custo, confiabilidade e facilidade de uso. Também aborda como obter as chaves de API para esses serviços.

Passo 1: Sugestão e análise comparativa de APIs de Scraping

Para cada plataforma, a escolha da API de scraping dependerá do equilíbrio entre custo, confiabilidade e a facilidade de integração com o n8n. As APIs listadas abaixo são geralmente bem avaliadas e oferecem uma boa combinação desses fatores.

Google Trends

Como o Google Trends não possui uma API oficial pública para acesso programático, a dependência de soluções de scraping é alta. As opções geralmente envolvem APIs de terceiros que encapsulam a lógica de scraping.

API Sugerida	Custo	Confiabilidade	Facilidade de Uso (n8n)	Notas
SerpApi Google Trends API [1]	Baseado em volume de requisições (planos pagos a partir de ~$50/mês para 5k requisições)	Alta (serviço dedicado)	Alta (requisições HTTP simples)	Oferece acesso estruturado aos dados do Google Trends. É uma solução robusta e mantida ativamente.
Bright Data Google Trends Scraper [2]	Baseado em volume de dados e requisições (planos flexíveis)	Alta (grande rede de proxies)	Média (requer configuração de scraper)	Mais complexo de configurar inicialmente, mas oferece alta escalabilidade e resiliência contra bloqueios.

Pytrends (via n8n Code Node) [3]	Gratuito (se hospedado por você)	Média (depende da manutenção do scraper)	Média (requer Python e ambiente customizado)	`Pytrends` é uma biblioteca Python não oficial. Pode ser usada num nó de código personalizado no n8n se você tiver um ambiente Python configurado, mas é mais propenso a quebras.

Recomendação para KAIRÓS: Para começar, o **SerpApi Google Trends API** é uma excelente escolha devido à sua facilidade de uso e confiabilidade. Se a escala aumentar e o custo se tornar um fator, ou se houver necessidade de maior controle, o Bright Data pode ser uma alternativa a ser explorada.

Twitter / X

O acesso à API oficial do X (anteriormente Twitter) é agora significativamente restrito e caro para a maioria dos casos de uso, tornando as APIs de scraping uma alternativa viável para dados públicos.

API Sugerida	Custo	Confiabilidade	Facilidade de Uso (n8n)	Notas
Scrapingdog X (Twitter) Scraping API [4]	Baseado em volume de requisições (planos a partir de ~$30/mês para 200k requisições)	Alta	Alta (requisições HTTP simples)	Projetado para evitar bloqueios e lidar com CAPTCHAs. Oferece dados estruturados de tweets, perfis e tendências.

Apify Twitter Scraper [5]	Baseado em uso de recursos (créditos)	Alta	Média (requer uso da plataforma Apify)	Apify oferece "Actors" pré-construídos para scraping do Twitter. É uma plataforma completa para gerenciamento de scrapers.
Bright Data Twitter Scraper [6]	Baseado em volume de dados e requisições	Alta	Média	Similar ao Bright Data para Google Trends, oferece alta escalabilidade e é bom para grandes volumes de dados.

Recomendação para KAIRÓS: O **Scrapingdog X (Twitter) Scraping API** é uma boa opção para começar, oferecendo um bom equilíbrio entre custo e funcionalidade para dados de tendências. Apify é uma alternativa robusta se você precisar de uma plataforma mais gerenciada para múltiplos scrapers.

TikTok (tendências públicas)

O TikTok oferece APIs oficiais (como a Research API), mas o acesso a dados de tendências públicas em larga escala pode ser mais fácil via APIs de scraping, especialmente para evitar processos de aprovação complexos para cada caso de uso.

API Sugerida	Custo	Confiabilidade	Facilidade de Uso (n8n)	Notas
Data365 TikTok Trends API [7]	Baseado em volume de requisições (planos pagos)	Média-Alta	Alta (requisições HTTP simples)	Focada especificamente em tendências e dados públicos do TikTok. Pode ser uma solução mais direta

				para o seu caso de uso.
Apify TikTok Scraper [8]	Baseado em uso de recursos (créditos)	Alta	Média	Oferece scrapers para vídeos, hashtags e usuários. Boa para coletar dados de tendências por hashtag ou explorar vídeos populares.
Bright Data TikTok Scraper [9]	Baseado em volume de dados e requisições	Alta	Média	Solução de scraping de nível empresarial, ideal para grandes volumes e alta resiliência.

Recomendação para KAIRÓS: O **Data365 TikTok Trends API** parece ser o mais alinhado com a necessidade de tendências públicas. Apify é uma alternativa sólida para uma abordagem mais generalista de scraping do TikTok.

Instagram (hashtags e posts públicos)

O Instagram, parte da Meta, tem restrições significativas em sua API oficial para acesso a dados públicos em massa. APIs de scraping são a principal forma de obter informações sobre hashtags e posts públicos sem depender de autenticação de usuário.

API Sugerida	Custo	Confiabilidade	Facilidade de Uso (n8n)	Notas
Scrapingdog Instagram Scraping API [10]	Baseado em volume de requisições (planos a partir de ~$30/mês para 200k requisições)	Alta	Alta (requisições HTTP simples)	Permite extrair dados de hashtags, perfis públicos e posts. Boa para monitorar tendências por hashtag.

Apify Instagram Scraper [11]	Baseado em uso de recursos (créditos)	Alta	Média	Oferece scrapers para diversos tipos de dados do Instagram, incluindo posts, perfis e hashtags.
Bright Data Instagram Scraper [12]	Baseado em volume de dados e requisições	Alta	Média	Solução de scraping robusta para grandes volumes de dados do Instagram.

Recomendação para KAIRÓS: O **Scrapingdog Instagram Scraping API** é uma escolha prática para começar a coletar dados de hashtags e posts públicos. É fácil de integrar e oferece um bom custo-benefício.

Passo 2: Explicação sobre como obter as chaves de API (API Keys) para esses serviços

Para utilizar qualquer uma das APIs de scraping sugeridas, você precisará obter uma chave de API (API Key) de cada provedor de serviço. O processo geral é bastante similar para a maioria deles:

1. **Registo na Plataforma:** Visite o site do provedor da API (por exemplo, RapidAPI, Scrapingdog, Apify, Bright Data, Data365).
2. **Criação de Conta:** Crie uma conta de utilizador. Muitos oferecem um plano gratuito ou um período de teste para que você possa experimentar o serviço.
3. **Acesso ao Painel de Controle:** Após o registo, você terá acesso a um painel de controle ou dashboard.
4. **Localização da API Key:** Dentro do painel, procure por secções como "API Keys", "Credenciais", "Minhas Aplicações" ou "Configurações". A sua chave de API estará visível

lá. Em alguns casos, você pode precisar gerar uma nova chave.

5. **Gerenciamento de Planos:** Se você precisar de mais requisições ou recursos do que o plano gratuito oferece, precisará fazer upgrade para um plano pago. Certifique-se de entender o modelo de precificação (por requisição, por volume de dados, por tempo de uso, etc.).

Exemplo prático (RapidAPI): Muitos desses provedores de API estão listados no RapidAPI Hub. Se você usar o RapidAPI, o processo é ainda mais centralizado:

1. Vá para https://rapidapi.com/ e crie uma conta.
2. Pesquise pela API desejada (ex: "Google Trends API").
3. Na página da API, você verá a opção para "Subscribe to Test" ou "Pricing". Escolha um plano.
4. A sua API Key (geralmente `X-RapidAPI-Key`) será exibida na página da API, pronta para ser usada nas suas requisições.

Armazenamento Seguro das API Keys: Assim como o Client Secret da Meta e do TikTok, as API Keys desses serviços de scraping são credenciais sensíveis. Elas **NUNCA** devem ser expostas no código do frontend. No n8n, você deve armazená-las de forma segura usando as **Credenciais** (Credentials) do n8n. Isso permite que você use as chaves em seus nós HTTP Request sem expô-las diretamente no workflow, e o n8n as gerencia de forma segura em seu ambiente.

Referências

[1] SerpApi. *Google Trends API*. Disponível em: https://serpapi.com/google-trends-api

[2] Bright Data. *Google Trends Scraper*. Disponível em: https://brightdata.com/products/serp-api/google-search/trends

[3] GitHub. *Pytrends*. Disponível em: https://github.com/GeneralMills/pytrends

[4] Scrapingdog. *X (Twitter) Scraping API*. Disponível em: https://www.scrapingdog.com/blog/best-twitter-scraper/

[5] Apify. *Twitter Scraper*. Disponível em: https://apify.com/scrapers/twitter

[6] Bright Data. *Twitter Scraper*. Disponível em: https://brightdata.com/products/web-scraper/twitter

[7] Data365. *TikTok Trends API*. Disponível em: https://data365.co/blog/tiktok-trends-api

[8] Apify. *TikTok Scraper*. Disponível em: https://apify.com/clockworks/tiktok-scraper

[9] Bright Data. *TikTok Scraper*. Disponível em: https://brightdata.com/products/web-scraper/tiktok

[10] Scrapingdog. *Instagram Scraping API*. Disponível em: https://www.scrapingdog.com/blog/instagram-scraper/

[11] Apify. *Instagram Scraper*. Disponível em: https://apify.com/apify/instagram-scraper

[12] Bright Data. *Instagram Scraper*. Disponível em: https://brightdata.com/products/web-scraper/instagram

Parte 2: O Workflow Principal no n8n (O Coração da Lógica)

Esta secção detalha a construção do workflow central no n8n, que será responsável por orquestrar a coleta, processamento, enriquecimento com IA e personalização dos dados de tendências. Este workflow é o "coração" da funcionalidade "Radar de Tendências" da KAIRÓS.

Passo 2.1: Gatilho e Input Dinâmico (Personalização desde o início)

O workflow deve ser acionado periodicamente para garantir que os dados de tendências estejam sempre atualizados. Além disso, a personalização é um requisito fundamental, o que significa que o workflow deve começar por obter as pautas e temas específicos de cada cliente.

1. **Nó Schedule (Agendador):** Adicione um nó `Schedule` como o ponto de partida do seu workflow. Configure-o para rodar na frequência desejada, por exemplo, a cada 4 horas (`Every 4 Hours`). Isso garantirá que o sistema esteja sempre a monitorizar as tendências mais recentes.

    - **Configuração do Nó Schedule:**
        - **Mode:** `Interval`
        - **Value:** `4`
        - **Unit:** `Hours`

2. **Nó Supabase (Consulta de Pautas do Cliente):** Após o nó `Schedule`, adicione um nó `Supabase`. Este nó será usado para consultar a base de dados vetorial do seu Supabase e buscar as palavras-chave, temas e "pautas" principais de cada cliente. Esta consulta deve retornar uma lista de clientes e as suas respetivas pautas, que servirão como input dinâmico para as buscas nas APIs de scraping.

    - **Configuração do Nó Supabase:**
        - **Operation:** `Read`
        - **Resource:** `Table` (ou o nome da sua tabela de clientes/pautas)
        - **Table Name:** `clientes_pautas` (ou o nome da sua tabela)
        - **Filters:** Se você tiver muitos clientes, pode querer filtrar por clientes ativos ou por aqueles que optaram por esta funcionalidade. Para este tutorial, assumiremos que você está a buscar todas as pautas relevantes.
        - **Output:** A saída deste nó deve ser uma lista de objetos, onde cada objeto contém o ID do cliente e as suas pautas (por exemplo, um array de strings ou um objeto com temas e embeddings).

Passo 2.2: Extração de Dados (Em Paralelo)

Para otimizar o tempo de execução e garantir a coleta de dados de múltiplas fontes, você deve criar ramos paralelos no seu workflow do n8n. Cada ramo será responsável por

chamar uma das APIs de scraping que você selecionou na Parte 1, usando os temas do cliente como parâmetros de busca.

1. **Nó Split In Batches (Dividir em Lotes):** Se o seu nó Supabase retornar múltiplos clientes, adicione um nó `Split In Batches` após o nó Supabase. Configure-o para processar um cliente por vez (`Batch Size: 1`). Isso permitirá que os ramos paralelos sejam executados para cada cliente individualmente.

2. **Nó Merge (Mesclar):** Adicione um nó `Merge` após o nó `Split In Batches` . Este nó será usado para mesclar os resultados de todos os ramos paralelos de coleta de dados para cada cliente. Configure-o para `Merge By: Item` e `Mode: Merge by position` .

3. **Ramos Paralelos (Nós HTTP Request):** Conecte múltiplos nós `HTTP Request` ao nó `Split In Batches` . Cada nó `HTTP Request` representará uma chamada a uma API de scraping diferente (Google Trends, Twitter/X, TikTok, Instagram). Use as credenciais seguras do n8n para as API Keys.

   - **Configuração de cada Nó HTTP Request:**

     - **Método (Method):** Geralmente `GET` ou `POST` , dependendo da API.

     - **URL:** A URL do endpoint da API de scraping.

     - **Headers:** Inclua os cabeçalhos necessários, como `X-RapidAPI-Key` ou `Authorization` , usando as credenciais do n8n.

     - **Query Parameters / Body Parameters:** Use expressões para passar os temas do cliente obtidos do nó Supabase. Por exemplo, se o nó Supabase retornar `{{ $json.pautas[0] }}` , você pode usar isso como um parâmetro de busca na API.

   - **URL:** `https://serpapi.com/search`

   - **Query Parameters:**

     - `engine` : `google_trends`

     - `q` : `{{ $json.pautas[0] }}` (ou itere sobre as pautas se a API suportar múltiplas buscas)

    - `api_key`: Use uma Credencial do n8n para a sua chave SerpApi.
- **URL:** `https://api.scrapingdog.com/twitter`
- **Query Parameters:**
    - `api_key`: Use uma Credencial do n8n para a sua chave Scrapingdog.
    - `query`: `{{ $json.pautas[0] }}`
    - `type`: `search`
- **Iteração sobre Pautas:** Se uma API de scraping não suportar múltiplas palavras-chave numa única requisição, você pode precisar de um nó `Loop` ou `Item Lists` para iterar sobre as pautas de cada cliente e fazer uma requisição separada para cada uma. Os resultados de cada iteração seriam então mesclados.
- **Tratamento de Erros:** Considere adicionar nós `Try/Catch` em cada ramo paralelo para lidar com falhas de API individualmente, sem interromper todo o workflow.

Passo 2.3: Transformação (Limpeza e Normalização)

Os dados retornados por diferentes APIs de scraping virão em formatos variados. Para que possam ser processados de forma consistente nos passos seguintes, é crucial limpá-los e normalizá-los numa estrutura de objeto unificada. Isso será feito usando um nó `Code` (JavaScript) no n8n.

1. **Nó Code (Transformação de Dados):** Adicione um nó `Code` após o nó `Merge` (que mesclou os resultados dos ramos paralelos). Este nó receberá os dados brutos de todas as fontes para um determinado cliente.
    - **Lógica no Nó Code (JavaScript):** O script JavaScript dentro deste nó deve iterar sobre os itens de entrada (que são os resultados das diferentes APIs de scraping) e transformá-los num formato padronizado. O objetivo é criar uma estrutura como: `{ titulo: "...", snippet: "...", fonte: "Twitter", url: "...", data: "..." }`.

Passo 2.4: Enriquecimento com IA (Gerando o Insight)

Com os dados de tendências limpos e normalizados, o próximo passo é usar a inteligência artificial para extrair insights acionáveis. O objetivo é identificar o tema central emergente, resumir em uma frase concisa e extrair entidades-chave para um "card de tendência" que o utilizador verá.

1. **Nó Aggregate (Agregação de Dados):** Antes de enviar para a IA, é útil agregar todos os dados normalizados de todas as fontes para um único cliente. Adicione um nó `Aggregate` após o nó `Code` de transformação.
    - **Configuração do Nó Aggregate:**
        - **Mode:** `Merge by index`
        - **Output:** `All`
2. **Nó de IA (ex: Gemini):** Adicione um nó de IA, como o nó `OpenAI` (que pode ser configurado para usar o Gemini via API, se o n8n tiver essa integração ou se você usar um nó HTTP Request para a API do Gemini). Este nó receberá o conjunto de dados agregados e aplicará um prompt estratégico.
    - **Configuração do Nó de IA (Exemplo com nó OpenAI/HTTP Request para Gemini API):**
        - **Model:** `gemini-pro` (ou o modelo Gemini apropriado)
        - **Prompt:** O prompt é crucial para guiar a IA. Ele deve ser construído dinamicamente, incluindo os títulos e snippets dos dados normalizados.
    - **Output:** A saída deste nó será um objeto JSON com o tema central, o resumo da tendência e as entidades-chave, prontos para serem usados no card de tendência.

Passo 2.5: Personalização (Cálculo do "Nível de Afinidade")

Esta é a etapa que torna o "Radar de Tendências" verdadeiramente personalizado. Você calculará a relevância de cada tendência para as pautas específicas de cada cliente usando embeddings vetoriais e similaridade de cosseno.

1. **Nó de Embeddings (API ou outro nó de IA):** Para calcular a similaridade de cosseno, você precisará de um vetor (embedding) para o resumo da tendência gerado pela IA. Se

o seu modelo de IA (Gemini) não retornar embeddings diretamente, você precisará de um nó adicional para gerar o embedding do `resumo_tendencia`.

- **Configuração do Nó de Embeddings (Exemplo com nó OpenAI/HTTP Request para API de Embeddings):**
  - **Model:** `text-embedding-ada-002` (ou um modelo de embedding apropriado)
  - **Input:** `{{ $json.resumo_tendencia }}` (o resumo da tendência gerado no passo anterior)
  - **Output:** Um vetor numérico (embedding) para o resumo da tendência.

2. **Nó Supabase (Busca de Similaridade de Cosseno):** Agora, use o nó Supabase novamente para executar uma busca de similaridade de cosseno na sua base de dados vetorial (`pgvector`). Você comparará o vetor do tema da tendência com os vetores das pautas do cliente (que foram obtidos no Passo 2.1).

- **Configuração do Nó Supabase (Busca de Similaridade):**
  - **Operation:** `Custom Query` (ou `RPC` se você tiver uma função Supabase para isso)
  - **Query:** Utilize uma consulta SQL que use a função de similaridade de cosseno do `pgvector`.
  - `pauta_embedding` : A coluna na sua tabela Supabase que armazena os embeddings das pautas do cliente.
  - `trend_embedding` : O embedding do resumo da tendência gerado no passo anterior.
  - `client_id` : O ID do cliente atual, passado através do workflow.
  - `1 - (A <=> B)` : Esta é a fórmula para calcular a similaridade de cosseno usando o operador de distância de cosseno (`<=>`) do `pgvector`. O resultado é um valor entre 0 e 1, onde 1 é a maior similaridade.
- **Output:** A saída deste nó será uma lista de pautas do cliente com a sua `affinity_score` para a tendência atual.

3. **Nó Code (Mapeamento do Nível de Afinidade):** Adicione um nó `Code` para processar a `affinity_score` e mapeá-la para um rótulo compreensível (ex: "Alta Afinidade", "Média Afinidade", "Baixa Afinidade").

    - **Lógica no Nó Code (JavaScript):**

Passo 2.6: Rankeamento e Carregamento (Salvando o Resultado Final)

O passo final do workflow é ranquear as tendências com base na sua relevância geral e no nível de afinidade com o cliente, e então salvar os resultados finais no seu banco de dados Supabase para que o frontend possa consumi-los.

1. **Nó Code (Lógica de Rankeamento):** Adicione um nó `Code` para implementar a lógica de ranqueamento final. Você pode usar uma fórmula de pontuação ponderada que combine o volume de menções (se disponível nas APIs de scraping) com o `Nível de Afinidade`.

    - **Lógica no Nó Code (JavaScript):**

2. **Nó Sort (Ordenação):** Se você estiver a processar múltiplas tendências para um único cliente, adicione um nó `Sort` para ordenar as tendências pela `pontuacao_final` em ordem decrescente.

3. **Nó Supabase (Salvar Tendências):** Adicione um nó `Supabase` para salvar a lista final de tendências (já ranqueadas e com o nível de afinidade) em uma tabela relacional simples no seu banco de dados. O frontend irá ler apenas desta tabela, garantindo que os dados já estão processados e prontos para exibição.

    - **Configuração do Nó Supabase (Salvar):**

        - **Operation:** `Insert` ou `Upsert` (se você quiser atualizar tendências existentes)
        - **Resource:** `Table`
        - **Table Name:** `tendencias_personalizadas` (ou o nome da sua tabela)
        - **Data:** Mapeie os campos da saída do nó anterior para as colunas da sua tabela. Inclua:

- `client_id`
- `tema_central`
- `resumo_tendencia`
- `entidades_chave` (pode ser armazenado como JSONB)
- `affinity_score`
- `affinity_level`
- `pontuacao_final`
- `data_coleta` (timestamp da coleta)

Parte 3: O que Enfatizar no Tutorial

Além da construção técnica do workflow, é crucial abordar as melhores práticas e considerações operacionais para garantir a robustez, a manutenibilidade e a eficiência do seu pipeline de dados de tendências. Esta secção destaca as áreas que merecem atenção especial.

Modularidade: A Importância de Construir o Workflow de Forma Flexível

Dado o risco inerente de instabilidade (brittleness) das APIs de scraping, a modularidade é a sua maior aliada. Construir o workflow de forma que seja fácil adicionar, substituir ou desativar uma fonte de dados se uma API de scraper falhar é fundamental para a resiliência do sistema.

- **Uso de Sub-Workflows ou Funções:** No n8n, considere encapsular a lógica de coleta de dados para cada fonte (Google Trends, Twitter/X, TikTok, Instagram) em sub-workflows separados ou em nós `Function` bem definidos. Isso permite que você isole a lógica de cada scraper. Se um scraper falhar, você pode desativar apenas o sub-workflow correspondente sem afetar o restante do pipeline.

- **Nós de Roteamento (Switch/If):** Utilize nós `Switch` ou `If` para controlar quais ramos de coleta de dados são ativados. Por exemplo, você pode ter uma variável de ambiente no n8n que habilita ou desabilita a coleta de dados de uma fonte específica. Isso é útil para testes ou para desativar temporariamente uma fonte problemática.
- **Padronização da Saída:** Como enfatizado na Parte 2.3, a normalização dos dados é crucial. Ao garantir que todos os scrapers produzam uma saída padronizada, você facilita a integração de novas fontes e a substituição de antigas, pois os nós subsequentes (enriquecimento com IA, personalização) não precisarão ser alterados.
- **Documentação Interna:** Mantenha uma documentação clara dentro do n8n (usando nós `Note`) explicando a finalidade de cada parte do workflow e como lidar com falhas específicas de cada scraper. Isso é vital para a equipe de manutenção.

Gestão de Chaves: Como Usar as Credenciais do n8n para Armazenar API Keys de Forma Segura

A segurança das suas API Keys é primordial. Expor essas chaves no código-fonte ou em variáveis de ambiente não seguras é um risco grave. O n8n oferece um sistema robusto de Credenciais (Credentials) para gerenciar isso.

- **Armazenamento Centralizado:** Sempre utilize o recurso de `Credentials` do n8n para armazenar todas as suas API Keys (SerpApi, Scrapingdog, Data365, etc.). Isso as mantém fora do seu workflow visual e as armazena de forma criptografada no ambiente do n8n.
- **Reutilização:** Uma vez configurada uma credencial, ela pode ser reutilizada em múltiplos nós `HTTP Request` e em diferentes workflows, garantindo consistência e facilitando a rotação de chaves, se necessário.
- **Variáveis de Ambiente:** Para configurações que não são credenciais sensíveis, mas que podem mudar entre ambientes (desenvolvimento, produção), utilize variáveis de ambiente no n8n. Por exemplo, URLs de endpoints de API que podem variar.
- **Princípio do Menor Privilégio:** Conceda apenas as permissões mínimas necessárias para cada API Key. Se uma API oferece diferentes tipos de chaves ou escopos, use o mais restritivo possível para a tarefa em questão.

Controle de Custos: Monitorando o Uso de APIs Pagas e Nós de IA

APIs de scraping e modelos de IA podem gerar custos significativos, especialmente em larga escala. É vital implementar estratégias para monitorar e controlar esses gastos para evitar surpresas na fatura.

- **Monitoramento de Uso:** A maioria dos provedores de API oferece painéis de controle onde você pode monitorar o uso das suas chaves. Verifique esses painéis regularmente. No n8n, você pode adicionar nós `Log` ou `Webhook` para enviar métricas de uso para um sistema de monitoramento externo (ex: Prometheus, Grafana, ou um simples Google Sheet) após cada chamada de API.
- **Limites de Requisição:** Configure limites de requisição (rate limits) nos seus nós `HTTP Request` no n8n, se a API de scraping tiver limites por minuto/hora. Isso evita que você exceda os limites e incorra em custos adicionais ou bloqueios.
- **Cache:** Para dados que não mudam com frequência, considere implementar um mecanismo de cache. Antes de fazer uma chamada a uma API paga, verifique se os dados já estão disponíveis no seu banco de dados (Supabase) e se ainda são válidos. Isso pode reduzir drasticamente o número de requisições pagas.
- **Otimização de Prompts de IA:** Para os nós de IA, otimize os seus prompts para serem o mais concisos e eficientes possível. Modelos de IA são geralmente cobrados por tokens de entrada e saída. Um prompt bem elaborado pode reduzir o custo por inferência.
- **Alertas de Orçamento:** Configure alertas de orçamento nos provedores de API e nos serviços de nuvem (se aplicável) para ser notificado quando o uso se aproximar de um determinado limite financeiro.

Resiliência e Logging: Garantindo a Manutenção Proativa do Sistema

Um pipeline de dados que depende de fontes externas e não oficiais precisa ser resiliente e ter um sistema de logging robusto para identificar e resolver problemas rapidamente.

- **Nós Error Trigger:** O n8n possui nós `Error Trigger` que podem ser configurados para capturar falhas em qualquer parte do workflow. Conecte esses nós a um sistema de notificação.

- **Notificações de Falha:** Em caso de erro, envie uma notificação imediata para a sua equipe (ex: por e-mail via nó `Email Send`, por Slack via nó `Slack`, ou por Telegram via nó `Telegram`). A mensagem deve incluir detalhes sobre o erro, o nó que falhou e, se possível, os dados que causaram a falha.
- **Logging Detalhado:** Utilize nós `Log` no n8n para registrar informações importantes em pontos chave do workflow. Isso inclui:
  - Início e fim da execução do workflow.
  - Sucesso ou falha de cada chamada de API de scraping.
  - Volume de dados processados.
  - Erros específicos e mensagens de exceção.
  - Resultados da transformação e enriquecimento de IA.
    Esses logs podem ser enviados para um serviço de logging centralizado (ex: Datadog, ELK Stack) ou para um simples arquivo no seu servidor, dependendo da sua infraestrutura.
- **Retentativas (Retries):** Para falhas temporárias de rede ou de API, configure retentativas automáticas nos nós `HTTP Request`. O n8n permite configurar o número de retentativas e o intervalo entre elas.
- **Monitoramento de Saúde:** Além dos alertas de falha, implemente um monitoramento de saúde para o seu workflow. Isso pode ser um simples ping para um endpoint do n8n que verifica se o workflow está ativo, ou uma verificação mais complexa que valida se os dados estão sendo atualizados no Supabase regularmente.

Ao enfatizar estas áreas, o tutorial não apenas fornece instruções técnicas, mas também orienta o desenvolvedor sobre como construir um sistema de tendências que seja não só funcional, mas também sustentável e fácil de manter a longo prazo, apesar da natureza volátil das fontes de dados de scraping.

Conclusão

Este tutorial forneceu um guia abrangente e estratégico para construir um pipeline de dados para o "Radar de Tendências" na sua plataforma KAIRÓS. Ao combinar a orquestração poderosa do n8n.io com APIs de scraping para dados públicos, inteligência artificial para enriquecimento e a capacidade vetorial do Supabase para personalização, você pode criar uma funcionalidade única e valiosa para os seus clientes.

Lembre-se da natureza dinâmica das fontes de dados de scraping e da importância de implementar resiliência, monitoramento e controle de custos. Com uma abordagem modular e atenção às melhores práticas, o seu "Radar de Tendências" será uma ferramenta poderosa para os profissionais de comunicação política.

Referências

[1] SerpApi. *Google Trends API*. Disponível em: https://serpapi.com/google-trends-api

[2] Bright Data. *Google Trends Scraper*. Disponível em: https://brightdata.com/products/serp-api/google-search/trends

[3] GitHub. *Pytrends*. Disponível em: https://github.com/GeneralMills/pytrends

[4] Scrapingdog. *X (Twitter) Scraping API*. Disponível em: https://www.scrapingdog.com/blog/best-twitter-scraper/

[5] Apify. *Twitter Scraper*. Disponível em: https://apify.com/scrapers/twitter

[6] Bright Data. *Twitter Scraper*. Disponível em: https://brightdata.com/products/web-scraper/twitter

[7] Data365. *TikTok Trends API*. Disponível em: https://data365.co/blog/tiktok-trends-api

[8] Apify. *TikTok Scraper*. Disponível em: https://apify.com/clockworks/tiktok-scraper

[9] Bright Data. *TikTok Scraper*. Disponível em: https://brightdata.com/products/web-scraper/tiktok

[10] Scrapingdog. *Instagram Scraping API*. Disponível em: https://www.scrapingdog.com/blog/instagram-scraper/

[11] Apify. *Instagram Scraper*. Disponível em: https://apify.com/apify/instagram-scraper

[12] Bright Data. *Instagram Scraper*. Disponível em: https://brightdata.com/products/web-scraper/instagram