Tutorial Técnico: Integração de Login e Insights do TikTok na Plataforma KAIRÓS

Autor: Manus AI

Este tutorial detalhado destina-se a desenvolvedores que desejam integrar a funcionalidade de "Login com TikTok" e a exibição de dados de Insights na plataforma SaaS KAIRÓS. A arquitetura proposta utiliza React/Next.js para o frontend e n8n.io para o backend/automação, comunicando-se exclusivamente com a API oficial do TikTok for Developers.

Introdução

A plataforma KAIRÓS visa fornecer aos seus clientes uma visão abrangente do seu desempenho no TikTok. A funcionalidade permitirá que os utilizadores que conectarem as suas contas do TikTok visualizem métricas cruciais de perfil e de vídeos, como visualizações, curtidas, comentários e compartilhamentos. Para garantir a segurança, a escalabilidade e a conformidade com as políticas do TikTok, este guia descreve um fluxo de integração que adere estritamente às melhores práticas de autenticação (OAuth 2.0) e recuperação de dados (TikTok for Developers API).

É fundamental compreender que a única fonte de dados aceitável para insights do TikTok é a API oficial do TikTok for Developers. Soluções alternativas, como o uso de iframes ou web scraping, não são viáveis nem seguras para este propósito. O uso de iframes para exibir conteúdo de terceiros pode introduzir vulnerabilidades de segurança e não oferece um método padronizado para autenticação e acesso a dados protegidos. O web scraping, por sua vez, é inerentemente frágil, suscetível a quebras com pequenas alterações na estrutura do site do TikTok, e viola os termos de serviço da maioria das plataformas, incluindo o TikTok, podendo levar ao bloqueio da aplicação ou da conta do utilizador. Além disso, o scraping não fornece acesso direto a métricas de insights detalhadas que são exclusivas da API.

Parte 1: Configuração no Portal TikTok for Developers

Esta secção aborda os passos iniciais necessários para configurar a sua aplicação no portal TikTok for Developers, um pré-requisito para interagir com a API do TikTok e implementar o fluxo de "Login com TikTok".

Passo 1: Como registrar uma conta e criar um novo App no portal de desenvolvedores do TikTok

Para começar, aceda ao portal TikTok for Developers em https://developers.tiktok.com/. Se ainda não tiver uma conta de desenvolvedor, precisará registar-se. Uma vez autenticado, siga os seguintes passos:

1. No painel de controle, procure a opção para criar uma nova aplicação (geralmente "Create New App" ou similar).
2. Será solicitado que forneça informações básicas sobre a sua aplicação:
    - **Nome da Aplicação (App Name):** Escolha um nome que represente a sua plataforma, como "KAIRÓS".
    - **Descrição da Aplicação (App Description):** Uma breve descrição do que a sua aplicação faz.
    - **Ícone da Aplicação (App Icon):** Faça o upload de um ícone para a sua aplicação.
    - **URL da Política de Privacidade (Privacy Policy URL):** Forneça a URL da política de privacidade da sua plataforma.
    - **URL dos Termos de Serviço (Terms of Service URL):** Forneça a URL dos termos de serviço da sua plataforma.
3. Conclua o processo de criação da aplicação. Após a criação, será redirecionado para a página de detalhes da sua aplicação.

Passo 2: Onde encontrar o Client Key e o Client Secret do aplicativo

O Client Key (também conhecido como App Key ou Client ID) e o Client Secret são credenciais exclusivas da sua aplicação, essenciais para a comunicação segura com a API do TikTok. O Client Key é público e usado para identificar a sua aplicação nas solicitações de

autenticação, enquanto o Client Secret é uma chave confidencial que nunca deve ser exposta no frontend.

Para encontrá-los:

1. Na página de detalhes da sua aplicação no portal TikTok for Developers, procure a secção "Credenciais" (Credentials) ou "Informações da Aplicação" (App Info).
2. Nesta secção, encontrará o **Client Key** e o **Client Secret**. O Client Secret estará oculto por padrão; pode ser necessário clicar em "Mostrar" (Show) ou "Gerar" (Generate) para revelá-lo.

ATENÇÃO: O Client Secret é uma credencial sensível. Nunca o inclua no código do seu frontend, em repositórios públicos ou em qualquer local acessível ao cliente. Ele deve ser usado exclusivamente no seu backend (n8n, neste caso) para realizar operações seguras, como a troca de códigos de autorização por tokens de acesso.

Passo 3: Como configurar as "Redirect URIs" permitidas para o fluxo OAuth

As Redirect URIs (URIs de Redirecionamento) são um componente crítico de segurança no fluxo OAuth 2.0. Elas informam ao TikTok para onde redirecionar o utilizador após a autenticação bem-sucedida e garantem que o código de autorização seja enviado apenas para um local confiável e pré-aprovado. Se a URI de redirecionamento na solicitação de autenticação não corresponder exatamente a uma das URIs configuradas, o TikTok recusará a solicitação, protegendo a sua aplicação contra ataques de phishing e redirecionamentos maliciosos.

Para configurar as Redirect URIs:

1. Na página de detalhes da sua aplicação no portal TikTok for Developers, procure a secção "Configurações de OAuth" (OAuth Settings) ou "Configurações de Segurança" (Security Settings).
2. Na secção "Redirect URIs" (URIs de Redirecionamento), adicione todas as URLs para as quais o TikTok deve redirecionar os utilizadores após a autenticação. Para a sua arquitetura, isto incluirá a URL do seu frontend (Next.js/Vercel) onde o código de autorização será capturado. Por exemplo:

- `https://sua-plataforma-kairós.vercel.app/auth/callback`
- `http://localhost:3000/auth/callback` (para desenvolvimento local)

3. Salve as alterações.

Passo 4: Como solicitar acesso aos produtos e escopos necessários (especialmente user.insights)

Para aceder aos dados do TikTok, a sua aplicação precisará solicitar acesso a produtos e escopos específicos. Os escopos definem as permissões que a sua aplicação terá sobre os dados do utilizador. Para os insights, o escopo `user.insights` é crucial.

1. No portal TikTok for Developers, na página de detalhes da sua aplicação, procure a secção "Produtos" (Products) ou "Permissões" (Permissions).
2. Adicione os produtos e escopos necessários. Para este tutorial, os principais escopos são:
   - `user.info.basic` : Permite que a sua aplicação aceda a informações básicas do perfil do utilizador, como nome de utilizador e ID.
   - `video.list` : Permite que a sua aplicação aceda à lista de vídeos do utilizador.
   - `user.insights` : **Este é o escopo mais importante para o seu caso de uso**, pois permite que a sua aplicação aceda aos dados de análise do perfil e dos vídeos do utilizador.
3. **Acesso à API:** Para aceder ao escopo `user.insights` , a sua aplicação precisará ser aprovada no processo de revisão do TikTok. Este processo é semelhante ao App Review da Meta e garante que a sua aplicação utiliza os dados do utilizador de forma responsável e de acordo com as políticas do TikTok. Durante o desenvolvimento, pode testar a sua aplicação com estes escopos usando contas de teste ou contas de desenvolvedor. No entanto, para que a sua aplicação seja utilizada por utilizadores em produção, com acesso a dados reais, a aprovação no processo de revisão é obrigatória.

Parte 2: O Fluxo de Login no Frontend (React/Next.js)

Esta secção detalha como implementar o fluxo de autenticação OAuth 2.0 no seu frontend React/Next.js, desde a criação do botão de login até o envio do código de autorização para o backend.

Passo 1: Como criar um componente de botão "Conectar com TikTok"

No seu aplicativo React/Next.js, crie um componente simples que servirá como o ponto de entrada para o fluxo de autenticação. Este componente será um botão que, ao ser clicado, iniciará o processo de redirecionamento para a página de autorização do TikTok.

Exemplo de componente React (arquivo `components/TikTokConnectButton.js`):

JSX

```
import React from 'react';

const TikTokConnectButton = () => {
  const handleConnect = () => {
    // A URL de autorização será construída aqui no próximo passo
    const tiktokAuthUrl = 'https://www.tiktok.com/auth/authorize?
client_key=SEU_CLIENT_KEY&redirect_uri=SUA_REDIRECT_URI&scope=user.info.basic
,video.list,user.insights&response_type=code';
    window.location.href = tiktokAuthUrl;
  };

  return (
    <button
      onClick={handleConnect}
      style={{
        backgroundColor: '#000000',
        color: 'white',
        padding: '10px 20px',
        borderRadius: '5px',
        border: 'none',
        cursor: 'pointer',
        fontSize: '16px',
        fontWeight: 'bold',
      }}
    >
      Conectar com TikTok
    </button>
  );
```

```
};

export default TikTokConnectButton;
```

Passo 2: Como construir a URL de autorização OAuth do TikTok

A URL de autorização é o ponto de partida do fluxo OAuth. Ela direciona o utilizador para a página de login e autorização do TikTok, onde ele concederá permissões à sua aplicação. A URL deve incluir vários parâmetros essenciais:

- `client_key`: O Client Key do seu aplicativo TikTok que obteve no Passo 2 da Parte 1.
- `redirect_uri`: A URI de redirecionamento que configurou no Passo 3 da Parte 1. Esta é a URL para onde o TikTok enviará o utilizador de volta após a autorização, juntamente com o código de autorização.
- `scope`: Uma lista separada por vírgulas dos escopos que a sua aplicação está a solicitar. Conforme discutido, inclua `user.info.basic`, `video.list` e `user.insights`.
- `response_type=code`: Indica que você espera um código de autorização como resposta. Este código será trocado por um Access Token no backend.

No componente `TikTokConnectButton` (ou onde preferir iniciar o fluxo), a URL seria construída da seguinte forma:

JavaScript

```
const CLIENT_KEY = process.env.NEXT_PUBLIC_TIKTOK_CLIENT_KEY; // Use
variáveis de ambiente
const REDIRECT_URI = process.env.NEXT_PUBLIC_TIKTOK_REDIRECT_URI;
const SCOPES = 'user.info.basic,video.list,user.insights';

const tiktokAuthUrl = `https://www.tiktok.com/auth/authorize?
client_key=${CLIENT_KEY}&redirect_uri=${REDIRECT_URI}&scope=${SCOPES}&respons
e_type=code`;

window.location.href = tiktokAuthUrl;
```

Importante: Utilize variáveis de ambiente (`process.env.NEXT_PUBLIC_...`) para o `CLIENT_KEY` e `REDIRECT_URI` no Next.js. Isso garante que essas informações não sejam codificadas

diretamente no seu código-fonte e possam ser facilmente geridas para diferentes ambientes (desenvolvimento, produção).

Passo 3: Como lidar com o redirecionamento do utilizador de volta para o nosso site

Após o utilizador autorizar a sua aplicação no TikTok, ele será redirecionado de volta para a `redirect_uri` que especificou. O código de autorização será anexado a esta URL como um parâmetro de consulta (`code`). O seu frontend precisará capturar este código.

Crie uma nova página no seu aplicativo Next.js que corresponda à sua `redirect_uri` (por exemplo, `pages/auth/tiktok/callback.js`):

JSX

```
import React, { useEffect } from 'react';
import { useRouter } from 'next/router';

const TikTokAuthCallbackPage = () => {
  const router = useRouter();

  useEffect(() => {
    if (router.isReady) {
      const { code, error } = router.query;

      if (code) {
        console.log('Código de autorização recebido:', code);
        // Enviar este código para o backend (n8n) no próximo passo
        sendCodeToBackend(code);
      } else if (error) {
        console.error('Erro na autenticação TikTok:', error);
        // Lidar com o erro, talvez redirecionar para uma página de erro
        router.push('/auth/error');
      }
    }
  }, [router.isReady, router.query]);

  const sendCodeToBackend = async (authCode) => {
    // Implementação no próximo passo
    console.log('Enviando código para o backend...');
  };

  return (
    <div>
```

```
      <p>Processando autenticação TikTok...</p>
      {/* Adicione um spinner ou mensagem de carregamento */}
    </div>
  );
};

export default TikTokAuthCallbackPage;
```

O `useEffect` com `router.isReady` garante que os parâmetros da URL estão disponíveis antes de tentar acedê-los. O `code` é o que precisamos; o `error` indica que o utilizador recusou a autorização ou ocorreu um problema.

Passo 4: Como enviar este código de autorização de forma segura do frontend para um webhook no nosso backend (n8n)

O código de autorização é de curta duração e deve ser trocado por um Access Token no backend. É crucial que esta comunicação seja segura, pois o código de autorização é a chave para obter o token de acesso. O frontend enviará este código para um webhook específico no seu n8n.

Continuando o exemplo da `TikTokAuthCallbackPage`:

JSX

```
import React, { useEffect } from 'react';
import { useRouter } from 'next/router';

const TikTokAuthCallbackPage = () => {
  const router = useRouter();

  useEffect(() => {
    if (router.isReady) {
      const { code, error } = router.query;

      if (code) {
        sendCodeToBackend(code);
      } else if (error) {
        console.error('Erro na autenticação TikTok:', error);
        router.push('/auth/error');
      }
    }
  }, [router.isReady, router.query]);
```

```
  const sendCodeToBackend = async (authCode) => {
    try {
      const response = await
fetch(process.env.NEXT_PUBLIC_N8N_TIKTOK_WEBHOOK_URL, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ code: authCode }),
      });

      if (response.ok) {
        const data = await response.json();
        console.log('Resposta do backend TikTok:', data);
        // Redirecionar o utilizador para a página de sucesso ou dashboard
        router.push('/dashboard/meu-desempenho-tiktok');
      } else {
        console.error('Erro ao enviar código para o backend TikTok:',
response.statusText);
        router.push('/auth/error');
      }
    } catch (error) {
      console.error('Erro de rede ao enviar código para o backend TikTok:',
error);
      router.push('/auth/error');
    }
  };

  return (
    <div>
      <p>Processando autenticação TikTok...</p>
    </div>
  );
};

export default TikTokAuthCallbackPage;
```

Exemplo de código usando `fetch` (ou `axios`):

Utilize `fetch` (nativo do navegador) ou uma biblioteca como `axios` para fazer uma requisição POST para o seu webhook do n8n. O corpo da requisição deve ser um JSON contendo o `code` de autorização.

JavaScript

```
// Exemplo com axios (se preferir)
// import axios from 'axios';

// const response = await
axios.post(process.env.NEXT_PUBLIC_N8N_TIKTOK_WEBHOOK_URL, {
//   code: authCode,
// });
```

Certifique-se de que `process.env.NEXT_PUBLIC_N8N_TIKTOK_WEBHOOK_URL` aponta para o URL do webhook que irá configurar no n8n. Este URL também deve ser uma variável de ambiente para flexibilidade e segurança.

Parte 3: Gestão de Tokens no Backend (n8n)

Esta secção aborda a lógica do backend no n8n para gerir o código de autorização e os tokens de acesso do TikTok. É aqui que a segurança é primordial, pois o Client Secret e os tokens de acesso serão manuseados.

Passo 1: Como criar um nó de Webhook no n8n para receber o código de autorização do frontend

No n8n, um nó de Webhook é o ponto de entrada para receber dados de fontes externas, como o seu frontend. Crie um novo workflow e adicione um nó de Webhook:

1. No seu painel do n8n, clique em "Novo Workflow" (New Workflow).
2. Pesquise e adicione o nó "Webhook" (Webhook).
3. Configure o nó Webhook:
    - **Método HTTP (HTTP Method):** Selecione `POST`, pois o seu frontend enviará o código via POST.
    - **Caminho do Webhook (Webhook Path):** Defina um caminho único e seguro, por exemplo, `/tiktok-auth-callback`. O URL completo do webhook será algo como `https://your-n8n-instance.com/webhook/tiktok-auth-callback`.

- **Modo de Resposta (Response Mode):** Pode começar com "Normal" (Normal) para depuração, mas para produção, considere "Sem Resposta" (No Response) ou "Resposta Personalizada" (Custom Response) para ter mais controlo.

4. Ative o workflow para que o webhook esteja ativo e pronto para receber requisições.

Quando o frontend enviar o código de autorização, ele chegará a este nó de Webhook. O código estará disponível na saída do nó, geralmente em `{{ $json.body.code }}`.

Passo 2: Usando um nó de HTTP Request no n8n, como fazer uma chamada para o endpoint oauth/token/ da API do TikTok para trocar o código de autorização por um Access Token

Após receber o código de autorização, o n8n precisará trocá-lo por um Access Token com a API do TikTok. Este é o primeiro passo na obtenção do token.

1. Adicione um nó "HTTP Request" (HTTP Request) após o nó Webhook.
2. Configure o nó HTTP Request para fazer uma requisição POST para o endpoint de token do TikTok:
   - **Método (Method):** `POST`
   - **URL:** `https://open.tiktokapis.com/oauth/access_token/`
   - **Parâmetros de Consulta (Query Parameters):** Não são necessários para esta requisição.
   - **Corpo da Requisição (Body Parameters):** Selecione "Form URL-Encoded" (Form URL-Encoded) ou "x-www-form-urlencoded" e adicione os seguintes parâmetros:
     - `client_key`: O seu Client Key do TikTok (obtido na Parte 1, Passo 2).
     - `client_secret`: O seu Client Secret do TikTok (obtido na Parte 1, Passo 2). **Este é o único lugar onde o Client Secret deve ser usado.**
     - `grant_type`: `authorization_code`
     - `redirect_uri`: A mesma URI de redirecionamento que usou no frontend e configurou no TikTok.

    - `code`: O código de autorização recebido do nó Webhook. Use uma expressão como `{{ $json.body.code }}`.
- `client_key`: `SEU_CLIENT_KEY`
- `client_secret`: `SEU_CLIENT_SECRET`
- `grant_type`: `authorization_code`
- `redirect_uri`: `SUA_REDIRECT_URI`
- `code`: `{{ $json.body.code }}`

3. A resposta desta requisição conterá o `access_token`, `refresh_token`, `expires_in` (tempo de expiração do access token em segundos) e `refresh_expires_in` (tempo de expiração do refresh token em segundos). O token estará disponível na saída do nó HTTP Request, por exemplo, `{{ $json.access_token }}`.

Passo 3: Como gerenciar o Refresh Token

O TikTok fornece um `refresh_token` que deve ser usado para obter novos `access_tokens` quando eles expirarem. Isso evita que os utilizadores tenham que se autenticar novamente com frequência. O fluxo de renovação é crucial para manter o acesso contínuo aos dados do utilizador.

Quando o `access_token` expirar, você usará o `refresh_token` para obter um novo. Isso geralmente é feito num workflow separado ou como parte do workflow de coleta de insights, antes de fazer as chamadas à API de insights.

1. Adicione um nó "HTTP Request" (HTTP Request) para renovar o token.
2. Configure este nó HTTP Request:
    - **Método (Method):** `POST`
    - **URL:** `https://open.tiktokapis.com/oauth/access_token/`
    - **Corpo da Requisição (Body Parameters):** Selecione "Form URL-Encoded" e adicione os seguintes parâmetros:

- `client_key`: O seu Client Key do TikTok.
- `client_secret`: O seu Client Secret do TikTok.
- `grant_type`: `refresh_token`
- `refresh_token`: O Refresh Token armazenado no seu banco de dados. Use uma expressão como `{{ $json.refresh_token }}` (assumindo que você o recuperou do banco de dados).

- `client_key`: `SEU_CLIENT_KEY`
- `client_secret`: `SEU_CLIENT_SECRET`
- `grant_type`: `refresh_token`
- `refresh_token`: `{{ $json.refresh_token_do_banco }}`

3. A resposta desta requisição conterá um novo `access_token` e, possivelmente, um novo `refresh_token`. Você deve atualizar o `access_token` e o `refresh_token` no seu banco de dados com os novos valores.

Passo 4: A melhor prática para armazenar de forma segura o Access Token e o Refresh Token em nosso banco de dados, associados ao ID do cliente KAIRÓS

O Access Token e o Refresh Token são as chaves para aceder aos dados do TikTok do utilizador. Eles devem ser armazenados de forma segura e associados ao ID do cliente KAIRÓS no seu banco de dados. A forma exata de armazenamento dependerá do seu sistema de banco de dados (SQL, NoSQL, etc.), mas as melhores práticas de segurança são universais.

1. **Criptografia (se aplicável):** Embora os tokens de acesso do TikTok sejam projetados para serem usados diretamente, se o seu banco de dados não tiver segurança de nível empresarial, considere criptografar os tokens antes de armazená-los. No entanto, para a maioria dos casos, a segurança do banco de dados e o acesso restrito são suficientes.
2. **Associação ao Cliente:** Certifique-se de que os tokens estão claramente associados ao ID único do cliente KAIRÓS que os autorizou. Isso permitirá que você recupere os tokens

corretos quando precisar buscar insights para um cliente específico.

3. **Armazenamento de Metadados:** Além dos tokens, armazene também o `expires_in` (data de expiração do access token) e o `refresh_expires_in` (data de expiração do refresh token). Isso é útil para gerir a renovação do token e para identificar o utilizador do TikTok.

4. **Nó de Banco de Dados no n8n:** O n8n oferece nós para interagir com vários bancos de dados (PostgreSQL, MySQL, MongoDB, etc.). Adicione o nó de banco de dados apropriado após o nó HTTP Request que obteve os tokens.

    - Configure o nó para inserir ou atualizar um registo na sua tabela de utilizadores ou numa tabela dedicada a integrações.
    - Mapeie os campos:
        - `client_id_kairós` : (Obtido de alguma forma, talvez do contexto do webhook ou de um nó anterior que identifica o utilizador KAIRÓS)
        - `tiktok_access_token` : `{{ $json.access_token }}` (do nó HTTP Request anterior)
        - `tiktok_refresh_token` : `{{ $json.refresh_token }}` (do nó HTTP Request anterior)
        - `tiktok_token_expires_at` : Calcule a data de expiração com base em `expires_in` e a data/hora atual.
        - `tiktok_refresh_expires_at` : Calcule a data de expiração com base em `refresh_expires_in` e a data/hora atual.
        - `tiktok_user_id` : (Obtenha este ID fazendo uma chamada simples à API de informações do utilizador com o access token antes de armazenar).

Parte 4: Coleta Periódica de Insights no Backend (n8n)

Com o Access Token e Refresh Token armazenados, o n8n pode agora ser configurado para coletar periodicamente os dados de insights do TikTok para os seus clientes. Esta secção descreve como automatizar este processo.

Passo 1: Como criar um novo workflow no n8n que roda periodicamente

Para coletar insights regularmente, você precisará de um workflow no n8n que seja acionado por um agendador. Isso garante que os dados sejam atualizados automaticamente sem intervenção manual.

1. Crie um novo workflow no n8n.
2. Adicione o nó "Schedule" (Agendador) como o nó inicial do workflow.
3. Configure o nó Schedule:
    - **Modo (Mode):** Selecione "Interval" (Intervalo) ou "Cron" (Cron) dependendo da frequência desejada.
        - **Intervalo:** Por exemplo, "Every 1 Hour" (A cada 1 hora) para atualizações por hora.
        - **Cron:** Para agendamentos mais complexos (ex: "0 0 * * *" para rodar à meia-noite todos os dias).
4. Ative o workflow.

Este workflow será o responsável por iterar sobre os seus clientes, buscar os seus tokens e coletar os insights.

Passo 2: Neste workflow, como buscar o Access Token do cliente do nosso banco de dados (e como usar o Refresh Token para obter um novo, se necessário)

O primeiro passo no workflow agendado é recuperar os Access Tokens e Refresh Tokens dos seus clientes do banco de dados. Você precisará de um nó de banco de dados para isso.

1. Após o nó Schedule, adicione um nó de banco de dados (por exemplo, "PostgreSQL", "MySQL", "MongoDB" ou um nó "Execute SQL" se estiver a usar um banco de dados relacional).
2. Configure o nó para selecionar todos os Access Tokens e Refresh Tokens ativos e os IDs de utilizador do TikTok associados aos seus clientes KAIRÓS.

3. Adicione um nó "Function" (Função) ou "Code" (Código) após o nó de banco de dados para verificar a validade do `access_token`. Se o `access_token` estiver prestes a expirar ou já tiver expirado, use o `refresh_token` para obter um novo (conforme descrito na Parte 3, Passo 3). Este nó deve atualizar o token no banco de dados antes de prosseguir.

4. A saída deste nó será uma lista de itens, onde cada item representa um cliente com o seu token (atualizado, se necessário) e ID de utilizador do TikTok. O n8n processará cada um desses itens sequencialmente ou em paralelo, dependendo da configuração do workflow.

Passo 3: Usando o token, como fazer chamadas ao endpoint user/insights/ da API. Forneça um exemplo de como solicitar métricas diárias para o perfil do utilizador nos últimos 30 dias. O corpo da requisição POST deve ser bem explicado.

Para cada cliente, você usará o Access Token e o ID do utilizador do TikTok para fazer chamadas à API do TikTok e obter os dados de insights. Você precisará de um nó "HTTP Request" dentro de um loop ou após um nó que itere sobre os resultados do banco de dados.

1. Adicione um nó "HTTP Request" após o nó que gerencia os tokens (ou dentro de um nó "Loop" se o seu banco de dados retornar vários itens).

2. Configure o nó HTTP Request para buscar os insights:

   - **Método (Method):** `POST`
   - **URL:** `https://open.tiktokapis.com/v2/user/insights/query/`
   - **Cabeçalhos (Headers):**
     - `Authorization` : `Bearer {{ $json.tiktok_access_token }}` (o token do cliente)
     - `Content-Type` : `application/json`
   - **Corpo da Requisição (Body Parameters):** Selecione "Raw" e "JSON" e forneça um JSON com as métricas e o período desejados. O corpo da requisição para insights do TikTok é um JSON, não parâmetros de consulta.

- `metrics`: Uma lista de métricas que você deseja obter. Consulte a documentação da API do TikTok for Developers para a lista completa de métricas disponíveis para `user.insights`.
- `dimensions`: Define a granularidade dos dados. `daily` para dados diários.
- `start_date` e `end_date`: O período para o qual você deseja os insights. Você precisará calcular essas datas dinamicamente no n8n (por exemplo, usando um nó "Date & Time" ou expressões JavaScript).

3. A resposta da API do TikTok conterá os dados de insights. Será um objeto JSON com uma estrutura que inclui `data`, onde cada item em `data` representa uma métrica com seus valores e dimensões.

Passo 4: Como processar a resposta da API e salvar os dados de insights de forma estruturada em nosso banco de dados para consumo do frontend

Os dados brutos da API do TikTok precisam ser processados e formatados antes de serem armazenados no seu banco de dados. Isso garante que os dados sejam consistentes e fáceis de consultar para exibição no frontend.

1. Adicione um nó "Function" (Função) ou "Code" (Código) após o nó HTTP Request que buscou os insights. Este nó permitirá que você manipule o JSON da resposta.
2. Dentro do nó Function/Code, escreva um script JavaScript para extrair as métricas relevantes e formatá-las. Por exemplo, você pode querer achatar a estrutura, renomear campos ou calcular valores derivados.
3. Após o nó Function/Code, adicione outro nó de banco de dados para salvar os dados processados. Configure-o para inserir ou atualizar os insights do cliente, associando-os ao `client_id_kairós`, `tiktok_user_id` e à data dos insights.
    - `client_id_kairós`
    - `tiktok_user_id`
    - `insight_date` (data do insight)

- `metric_name`
- `metric_value`
- ... e quaisquer outras métricas que você coletar.

Parte 5: Exibição dos Dados no Frontend (React/Next.js)

Finalmente, esta secção descreve como o frontend da KAIRÓS exibirá os dados de insights que foram coletados e armazenados no seu banco de dados pelo n8n.

Passo 1: Como o frontend deve buscar os dados de insights, que já foram processados e salvos no nosso banco, fazendo uma chamada a um webhook do n8n. Reforce por que esta arquitetura desacoplada é superior.

É crucial que o frontend não faça chamadas diretas à API do TikTok. A arquitetura proposta (Frontend -> n8n -> BD) é preferível por várias razões:

1. **Segurança:** O Access Token e o Refresh Token do TikTok nunca são expostos no frontend. Todas as operações sensíveis são encapsuladas no backend (n8n).
2. **Performance:** O frontend busca dados pré-processados e otimizados do seu próprio banco de dados, que é muito mais rápido e confiável do que fazer múltiplas chamadas à API externa para cada utilizador.
3. **Controlo de Taxa (Rate Limiting):** O backend (n8n) pode gerir e otimizar as chamadas à API do TikTok, implementando lógicas de cache e controlo de taxa para evitar exceder os limites da API.
4. **Flexibilidade:** Se a API do TikTok mudar, apenas o seu workflow do n8n precisa ser atualizado, não o seu frontend.
5. **Agregação e Transformação:** O n8n já processou e agregou os dados, apresentando-os ao frontend num formato pronto para consumo, reduzindo a carga de trabalho do frontend.

Para buscar os dados, crie um novo webhook no n8n que servirá como uma API para o seu frontend:

1. Crie um novo workflow no n8n.
2. Adicione um nó "Webhook" (Webhook) como o nó inicial.
3. Configure-o com um método `GET` (ou `POST` se precisar de enviar parâmetros complexos, como filtros de data) e um caminho seguro, por exemplo, `/api/tiktok-insights`.
4. Após o Webhook, adicione um nó de banco de dados para buscar os insights do cliente KAIRÓS que está a fazer a requisição. Você precisará de alguma forma de identificar o cliente (por exemplo, através de um token de sessão ou ID de utilizador enviado na requisição do frontend).
5. O nó "Respond to Webhook" (Responder ao Webhook) enviará os dados do banco de dados de volta para o frontend como JSON.

No seu frontend React/Next.js, você fará uma requisição a este webhook:

JSX

```
import React, { useEffect, useState } from 'react';

const MyTikTokPerformancePage = () => {
  const [insights, setInsights] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchInsights = async () => {
      try {
        // Assumindo que você tem uma forma de identificar o cliente KAIRÓS
        // Por exemplo, um ID de utilizador autenticado no frontend
        const clientId = 'ID_DO_CLIENTE_LOGADO'; // Substitua pela lógica
real

        const response = await
fetch(`${process.env.NEXT_PUBLIC_N8N_API_URL}/api/tiktok-insights?
clientId=${clientId}`);
        if (!response.ok) {
          throw new Error(`Erro HTTP: ${response.status}`);
```

```
        }
        const data = await response.json();
        setInsights(data);
      } catch (err) {
        setError(err.message);
      } finally {
        setLoading(false);
      }
    };

    fetchInsights();
  }, []);

  if (loading) return <p>Carregando insights do TikTok...</p>;
  if (error) return <p>Erro ao carregar insights do TikTok: {error}</p>;
  if (!insights) return <p>Nenhum insight do TikTok disponível.</p>;

  // ... renderizar os dados no próximo passo
  return (
    <div>
      <h1>Meu Desempenho no TikTok</h1>
      {/* Exibir dados aqui */}
      <pre>{JSON.stringify(insights, null, 2)}</pre>
    </div>
  );
};

export default MyTikTokPerformancePage;
```

Passo 2: Um exemplo simples de como exibir esses dados (ex: total de visualizações, total de seguidores) em um componente React

Com os dados de insights disponíveis no estado do seu componente React, você pode renderizá-los de forma significativa. Use componentes de UI para apresentar as métricas de forma clara e visualmente atraente.

Continuando o exemplo de `MyTikTokPerformancePage`:

JSX

```
import React, { useEffect, useState } from 'react';

const MyTikTokPerformancePage = () => {
  const [insights, setInsights] = useState(null);
  const [loading, setLoading] = useState(true);
```

```
  const [error, setError] = useState(null);

  useEffect(() => {
    const fetchInsights = async () => {
      try {
        const clientId = 'ID_DO_CLIENTE_LOGADO'; // Substitua pela lógica
real
        const response = await
fetch(`${process.env.NEXT_PUBLIC_N8N_API_URL}/api/tiktok-insights?
clientId=${clientId}`);
        if (!response.ok) {
          throw new Error(`Erro HTTP: ${response.status}`);
        }
        const data = await response.json();
        setInsights(data);
      } catch (err) {
        setError(err.message);
      } finally {
        setLoading(false);
      }
    };

    fetchInsights();
  }, []);

  if (loading) return <p>Carregando insights do TikTok...</p>;
  if (error) return <p>Erro ao carregar insights do TikTok: {error}</p>;
  if (!insights) return <p>Nenhum insight do TikTok disponível.</p>;

  // Exemplo de como acessar e exibir as métricas
  // Assumindo que 'insights' é um array de objetos como processado no n8n
  const totalFollowers = insights.find(item => item.metric ===
'total_followers' && item.dimensions === 'daily')?.value; // Exemplo para uma
métrica diária
  const totalViews = insights.find(item => item.metric === 'total_views' &&
item.dimensions === 'daily')?.value;

  return (
    <div style={{ fontFamily: 'Arial, sans-serif', padding: '20px' }}>
      <h1>Meu Desempenho no TikTok</h1>

      <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fit,
minmax(250px, 1fr))', gap: '20px' }}>
        <div style={cardStyle}>
          <h2>Total de Seguidores</h2>
          <p style={metricValueStyle}>{totalFollowers || 'N/A'}</p>
          <p style={metricDescriptionStyle}>Número total de seguidores na
data mais recente.</p>
```

```
        </div>

        <div style={cardStyle}>
          <h2>Total de Visualizações</h2>
          <p style={metricValueStyle}>{totalViews || 'N/A'}</p>
          <p style={metricDescriptionStyle}>Número total de visualizações de
vídeo na data mais recente.</p>
        </div>

        {/* Adicione mais métricas conforme necessário */}
        {insights.map((item, index) => {
          if (item.metric !== 'total_followers' && item.metric !==
'total_views') {
            return (
              <div key={index} style={cardStyle}>
                <h2>{item.metric.replace(/_/g, ' ').replace(/total/g,
'Total')}</h2>
                <p style={metricValueStyle}>{item.value || 'N/A'}</p>
                <p style={metricDescriptionStyle}>Métrica {item.metric} para
a data {item.date}.</p>
              </div>
            );
          }
          return null;
        })}
      </div>

    </div>
  );
};

const cardStyle = {
  backgroundColor: '#f9f9f9',
  border: '1px solid #ddd',
  borderRadius: '8px',
  padding: '20px',
  boxShadow: '0 2px 4px rgba(0,0,0,0.1)',
  textAlign: 'center',
};

const metricValueStyle = {
  fontSize: '2.5em',
  fontWeight: 'bold',
  color: '#333',
  margin: '10px 0',
};

const metricDescriptionStyle = {
```

```
  fontSize: '0.9em',
  color: '#666',
};

export default MyTikTokPerformancePage;
```

Este exemplo usa estilos inline para demonstração, mas em uma aplicação real, você usaria CSS Modules, Styled Components ou Tailwind CSS para gerenciar os estilos de forma mais organizada. A chave é mapear os dados recebidos do backend para elementos de UI que os apresentem de forma clara e intuitiva para o utilizador final.

Conclusão

Este tutorial forneceu um guia abrangente e passo a passo para integrar o "Login com TikTok" e a exibição de insights na sua plataforma KAIRÓS, utilizando uma arquitetura robusta com React/Next.js e n8n.io, e aderindo estritamente às melhores práticas de segurança e às políticas da API oficial do TikTok for Developers. Ao seguir estas diretrizes, você pode garantir uma integração segura, eficiente e escalável, proporcionando aos seus clientes as métricas de desempenho do TikTok de que necessitam.

Lembre-se de que o portal TikTok for Developers e a sua API estão em constante evolução. Consulte sempre a documentação oficial do TikTok para as informações mais recentes sobre versões da API, escopos e processos de revisão de aplicações.

Referências

[1] TikTok for Developers. *Documentação da API*. Disponível em: https://developers.tiktok.com/

[2] TikTok for Developers. *Guia de Autenticação OAuth 2.0*. Disponível em: https://developers.tiktok.com/doc/oauth-2-0-overview

[3] TikTok for Developers. *API de Insights do Utilizador*. Disponível em: https://developers.tiktok.com/doc/user-insights-api

[4] n8n.io. *Documentação*. Disponível em: https://docs.n8n.io

[5] Next.js. *Documentação*. Disponível em: https://nextjs.org/docs

[6] React. *Documentação*. Disponível em: https://react.dev/