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

Autor: Manus AI

Este tutorial detalhado destina-se a desenvolvedores que desejam integrar a funcionalidade de "Login com Instagram" 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 Graph oficial da Meta.

Introdução

A plataforma KAIRÓS visa fornecer aos seus clientes na área de comunicação política uma visão abrangente do seu desempenho no Instagram. A página "Meu Desempenho" permitirá que os utilizadores que conectarem as suas contas do Instagram visualizem métricas cruciais como alcance, engajamento e crescimento de seguidores. Para garantir a segurança, a escalabilidade e a conformidade com as políticas da Meta, este guia descreve um fluxo de integração que adere estritamente às melhores práticas de autenticação (OAuth 2.0) e recuperação de dados (API Graph).

É fundamental compreender que a única fonte de dados aceitável para insights do Instagram é a API Graph oficial da Meta. 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, como clickjacking, 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 Instagram, e viola os termos de serviço da maioria das plataformas, incluindo o Instagram, 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 Inicial na Plataforma Meta for Developers

Esta secção aborda os passos iniciais necessários para configurar a sua aplicação na plataforma Meta for Developers, um pré-requisito para interagir com a API Graph do Instagram e implementar o fluxo de "Login com Instagram".

Passo 1: Como criar um novo App na plataforma de desenvolvedores da Meta

Para começar, aceda à plataforma Meta for Developers em developers.facebook.com. Certifique-se de que está autenticado com a sua conta do Facebook. Uma vez na plataforma, siga os seguintes passos:

1. No canto superior direito, clique em "Meus Aplicativos" (My Apps).
2. Clique no botão "Criar Aplicativo" (Create App).
3. Será solicitado que escolha um tipo de aplicativo. Para a integração do Instagram, selecione "Consumidor" (Consumer) ou "Empresa" (Business), dependendo do escopo e dos requisitos futuros da sua aplicação. Para a maioria das integrações de login e insights, "Consumidor" é um bom ponto de partida, mas "Empresa" pode ser mais apropriado se a KAIRÓS tiver outras integrações de negócios com a Meta no futuro. Selecione o tipo e clique em "Continuar" (Continue).
4. Preencha os detalhes do seu aplicativo:
    - **Nome de Exibição do Aplicativo (App Display Name):** Este é o nome que os utilizadores verão quando a sua aplicação solicitar permissões. Escolha um nome que represente a sua plataforma, como "KAIRÓS".
    - **Email de Contato do Aplicativo (App Contact Email):** Um endereço de email para contato.

- **Finalidade do Aplicativo (App Purpose):** Selecione a opção que melhor descreve o uso do seu aplicativo, por exemplo, "Para você mesmo ou seu próprio negócio" (For yourself or your own business).

5. Clique em "Criar Aplicativo" (Create App). Poderá ser solicitado que insira a sua palavra-passe do Facebook para verificação de segurança.

Após a criação, será redirecionado para o Painel do seu novo aplicativo. Este é o centro de controlo onde gerirá todas as configurações e produtos relacionados com a sua aplicação.

Passo 2: Como configurar o produto "Login com Facebook"

Embora o objetivo seja o "Login com Instagram", a autenticação para a API Graph do Instagram é gerida através do produto "Login com Facebook" na plataforma Meta for Developers. Siga estes passos para adicioná-lo e configurá-lo:

1. No Painel do seu aplicativo, no menu lateral esquerdo, procure a secção "Produtos" (Products).
2. Clique no botão "Adicionar Produto" (Add Product).
3. Na lista de produtos disponíveis, localize "Login com Facebook" (Facebook Login) e clique em "Configurar" (Set up).
4. Após adicionar o produto, ele aparecerá no menu lateral esquerdo, abaixo de "Produtos". Clique em "Login com Facebook" e, em seguida, em "Configurações" (Settings).
5. Nesta página de configurações, certifique-se de que a opção "Login de Cliente OAuth" (Client OAuth Login) está ativada. Esta é uma configuração crucial para permitir o fluxo de autenticação OAuth 2.0.
6. Para a integração com o Instagram, também precisará adicionar o produto "Login com Instagram" (Instagram Login) se ainda não o fez. No Painel do seu aplicativo, no menu lateral esquerdo, em "Produtos", clique em "Adicionar Produto" (Add Product). Localize "Login com Instagram" (Instagram Login) e clique em "Configurar" (Set up). Depois de adicionado, clique em "Login com Instagram" no menu lateral e, em seguida, em "Básico" (Basic Display). Ative a opção "API de Exibição Básica do Instagram" (Instagram Basic Display API) se for apresentada.

Passo 3: Onde encontrar o App ID e o App Secret

O App ID e o App Secret são credenciais exclusivas da sua aplicação, essenciais para a comunicação segura com a API da Meta. O App ID é público e usado para identificar a sua aplicação nas solicitações de autenticação, enquanto o App Secret é uma chave confidencial que nunca deve ser exposta no frontend.

Para encontrá-los:

1. No Painel do seu aplicativo, no menu lateral esquerdo, clique em "Configurações" (Settings) e, em seguida, em "Básico" (Basic).
2. Nesta página, encontrará o **ID do Aplicativo (App ID)** e a **Chave Secreta do Aplicativo (App Secret)**. O App Secret estará oculto por padrão; clique em "Mostrar" (Show) e insira a sua palavra-passe do Facebook para revelá-lo.

ATENÇÃO: A Chave Secreta do Aplicativo é uma credencial sensível. Nunca a inclua no código do seu frontend, em repositórios públicos ou em qualquer local acessível ao cliente. Ela deve ser usada exclusivamente no seu backend (n8n, neste caso) para realizar operações seguras, como a troca de códigos de autorização por tokens de acesso.

Passo 4: Como configurar as URIs de Redirecionamento do OAuth

As URIs de Redirecionamento do OAuth (OAuth Redirect URIs) são um componente crítico de segurança no fluxo OAuth 2.0. Elas informam à Meta 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, a Meta recusará a solicitação, protegendo a sua aplicação contra ataques de phishing e redirecionamentos maliciosos.

Para configurar as URIs de Redirecionamento:

1. No Painel do seu aplicativo, no menu lateral esquerdo, navegue até "Login com Facebook" (Facebook Login) e clique em "Configurações" (Settings).
2. Na secção "URIs de Redirecionamento OAuth Válidos" (Valid OAuth Redirect URIs), adicione todas as URLs para as quais a Meta 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. Além disso, para a API de Exibição Básica do Instagram, também precisará configurar as URIs de redirecionamento. No menu lateral esquerdo, em "Login com Instagram" (Instagram Login), clique em "Básico" (Basic Display). Na secção "URIs de Redirecionamento OAuth Válidos" (Valid OAuth Redirect URIs), adicione as mesmas URLs que adicionou para o Login com Facebook.
4. Clique em "Salvar Alterações" (Save Changes) na parte inferior da página.

Requisitos e Restrições Técnicas Adicionais

Tipo de Conta do Instagram

É crucial notar que a API Graph da Meta para Insights do Instagram funciona exclusivamente para **Contas de Instagram Business ou Creator**. Contas pessoais não têm acesso a estas métricas através da API. O seu tutorial deve incluir uma nota clara sobre como o utilizador pode verificar ou converter a sua conta:

- **Verificar Tipo de Conta:** O utilizador pode verificar o tipo de conta nas configurações do seu perfil no aplicativo do Instagram, em "Configurações e privacidade" > "Ferramentas e controles para criadores de conteúdo" (ou "Para empresas"). Se vir opções como "Mudar para conta pessoal" ou "Mudar para conta de criador de conteúdo/negócio", isso indica o tipo atual.
- **Converter Conta:** Se a conta for pessoal, o utilizador pode convertê-la para uma conta Business ou Creator diretamente nas configurações do aplicativo do Instagram. Este processo é simples e geralmente envolve a vinculação a uma Página do Facebook.

Permissões (Scopes) Necessárias

Durante o fluxo OAuth, a sua aplicação precisará solicitar permissões específicas (scopes) para aceder aos dados do perfil e, mais importante, aos insights do Instagram. As permissões são cruciais para definir o que a sua aplicação pode fazer com os dados do utilizador. As permissões que provavelmente precisará solicitar são:

- `instagram_basic` : Permite que a sua aplicação leia o perfil básico do Instagram do utilizador, incluindo nome de utilizador, ID e tipo de conta.
- `instagram_manage_insights` : **Esta é a permissão mais importante para o seu caso de uso**, pois permite que a sua aplicação aceda aos dados de insights de mídia e perfil do Instagram Business ou Creator.
- `pages_show_list` : Permite que a sua aplicação veja uma lista das Páginas do Facebook que o utilizador gere. O Instagram Business/Creator está vinculado a uma Página do Facebook, e esta permissão é necessária para identificar a Página correta.

- `pages_read_engagement` : Permite que a sua aplicação leia o conteúdo e as informações básicas de uma Página do Facebook, o que é necessário para aceder aos dados do Instagram através da Página vinculada.

Estas permissões devem ser incluídas na URL de autorização OAuth que o seu frontend construirá.

App Review da Meta

Para que a sua aplicação possa solicitar permissões avançadas, como `instagram_manage_insights` , a Meta exige que a aplicação passe por um processo de "App Review". Este processo é uma revisão de segurança e conformidade realizada pela Meta para garantir que a sua aplicação utiliza os dados do utilizador de forma responsável e de acordo com as suas políticas. Durante o desenvolvimento, pode testar a sua aplicação com estas permissões 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, o App Review é obrigatório.

O processo de App Review geralmente envolve:

1. **Submissão de Detalhes:** Fornecer informações detalhadas sobre como a sua aplicação usa cada permissão solicitada.
2. **Demonstração em Vídeo:** Gravar um vídeo que demonstre o fluxo de utilizador na sua aplicação e como as permissões são utilizadas.
3. **Testes:** A Meta pode realizar testes na sua aplicação para verificar a conformidade.

É um processo que requer planeamento e atenção aos detalhes, e pode levar algum tempo para ser aprovado. Comece o processo de App Review assim que tiver uma versão funcional da sua aplicação que utilize as permissões necessárias.

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 Instagram"

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 da Meta.

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

JSX

```
import React from 'react';

const InstagramConnectButton = () => {
  const handleConnect = () => {
    // A URL de autorização será construída aqui no próximo passo
    const metaAuthUrl = 'https://www.facebook.com/v19.0/dialog/oauth?
client_id=SEU_APP_ID&redirect_uri=SUA_REDIRECT_URI&scope=instagram_basic,instagram_manage_insights,pages_show
_list,pages_read_engagement&response_type=code';
    window.location.href = metaAuthUrl;
  };

  return (
    <button
      onClick={handleConnect}
```

```
      style={{
        backgroundColor: '#E1306C',
        color: 'white',
        padding: '10px 20px',
        borderRadius: '5px',
        border: 'none',
        cursor: 'pointer',
        fontSize: '16px',
        fontWeight: 'bold',
      }}
    >
      Conectar com Instagram
    </button>
  );
};

export default InstagramConnectButton;
```

Passo 2: Como construir a URL de autorização OAuth da Meta

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

- `client_id`: O ID do seu aplicativo Meta (App ID) que obteve no Passo 3 da Parte 1.
- `redirect_uri`: A URI de redirecionamento que configurou no Passo 4 da Parte 1. Esta é a URL para onde a Meta enviará o utilizador de volta após a autorização, juntamente com o código de autorização.
- `scope`: Uma lista separada por vírgulas das permissões que a sua aplicação está a solicitar. Conforme discutido, inclua `instagram_basic`, `instagram_manage_insights`, `pages_show_list` e `pages_read_engagement`.
- `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 `InstagramConnectButton` (ou onde preferir iniciar o fluxo), a URL seria construída da seguinte forma:

JavaScript

```
const APP_ID = process.env.NEXT_PUBLIC_META_APP_ID; // Use variáveis de ambiente
const REDIRECT_URI = process.env.NEXT_PUBLIC_META_REDIRECT_URI;
const SCOPES = 'instagram_basic,instagram_manage_insights,pages_show_list,pages_read_engagement';

const metaAuthUrl = `https://www.facebook.com/v19.0/dialog/oauth?
client_id=${APP_ID}&redirect_uri=${REDIRECT_URI}&scope=${SCOPES}&response_type=code`;

window.location.href = metaAuthUrl;
```

Importante: Utilize variáveis de ambiente (`process.env.NEXT_PUBLIC_...`) para o `APP_ID` 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 na Meta, 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/callback.js`):

JSX

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

const AuthCallbackPage = () => {
  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 Meta:', 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...</p>
      {/* Adicione um spinner ou mensagem de carregamento */}
    </div>
  );
};

export default AuthCallbackPage;
```

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 `AuthCallbackPage`:

JSX

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

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

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

      if (code) {
```

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

  const sendCodeToBackend = async (authCode) => {
    try {
      const response = await fetch(process.env.NEXT_PUBLIC_N8N_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:', data);
        // Redirecionar o utilizador para a página de sucesso ou dashboard
        router.push('/dashboard/meu-desempenho');
      } else {
        console.error('Erro ao enviar código para o backend:', response.statusText);
        router.push('/auth/error');
      }
    } catch (error) {
      console.error('Erro de rede ao enviar código para o backend:', error);
      router.push('/auth/error');
    }
  };

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

export default AuthCallbackPage;
```

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_WEBHOOK_URL, {
//   code: authCode,
// });
```

Certifique-se de que `process.env.NEXT_PUBLIC_N8N_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 Instagram. É aqui que a segurança é primordial, pois o App Secret e os tokens de acesso de longa duração 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, `/instagram-auth-callback`. O URL completo do webhook será algo como `https://your-n8n-instance.com/webhook/instagram-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 a API da Meta para trocar o código de autorização por um Access Token de curta duração

Após receber o código de autorização, o n8n precisará trocá-lo por um Access Token de curta duração com a API da Meta. 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 da Meta:
   - **Método (Method):** `POST`
   - **URL:** `https://graph.facebook.com/v19.0/oauth/access_token` (Verifique a versão mais recente da API Graph da Meta).
   - **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_id`: O seu App ID da Meta (obtido na Parte 1, Passo 3).
     - `client_secret`: O seu App Secret da Meta (obtido na Parte 1, Passo 3). **Este é o único lugar onde o App Secret deve ser usado.**
     - `grant_type`: `authorization_code`
     - `redirect_uri`: A mesma URI de redirecionamento que usou no frontend e configurou na Meta.
     - `code`: O código de autorização recebido do nó Webhook. Use uma expressão como `{{ $json.body.code }}`.

- `client_id` : `SEU_APP_ID`
- `client_secret` : `SEU_APP_SECRET`
- `grant_type` : `authorization_code`
- `redirect_uri` : `SUA_REDIRECT_URI`
- `code` : `{{ $json.body.code }}`

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

Passo 3: (Etapa avançada e crucial) Usando outro nó de HTTP Request, como trocar o token de curta duração por um Access Token de longa duração

Os Access Tokens de curta duração geralmente expiram em cerca de uma hora. Para evitar que os utilizadores tenham que se autenticar novamente com frequência, é essencial trocá-los por Access Tokens de longa duração, que podem durar até 60 dias. Este é um passo crucial para uma boa experiência do utilizador.

1. Adicione outro nó "HTTP Request" (HTTP Request) após o nó que obteve o token de curta duração.
2. Configure este novo nó HTTP Request:
   - **Método (Method):** `GET`
   - **URL:** `https://graph.facebook.com/v19.0/oauth/access_token` (Verifique a versão mais recente da API Graph da Meta).
   - **Parâmetros de Consulta (Query Parameters):** Adicione os seguintes parâmetros:
     - `grant_type` : `fb_exchange_token`
     - `client_id` : O seu App ID da Meta.
     - `client_secret` : O seu App Secret da Meta.
     - `fb_exchange_token` : O Access Token de curta duração obtido no passo anterior. Use uma expressão como `{{ $json.access_token }}` (assumindo que o nó anterior produziu `access_token` na sua saída).
   - `grant_type` : `fb_exchange_token`
   - `client_id` : `SEU_APP_ID`
   - `client_secret` : `SEU_APP_SECRET`
   - `fb_exchange_token` : `{{ $json.access_token }}`
3. A resposta desta requisição conterá o novo `access_token` (de longa duração) e um novo `expires_in`. Este token de longa duração é o que deve ser armazenado no seu banco de dados.

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

O Access Token de longa duração é a chave para aceder aos dados do Instagram do utilizador. Ele deve ser armazenado de forma segura e associado 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 da Meta sejam projetados para serem usados diretamente, se o seu banco de dados não tiver segurança de nível empresarial, considere criptografar o token antes de armazená-lo. 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 o token está claramente associado ao ID único do cliente KAIRÓS que o autorizou. Isso permitirá que você recupere o token correto quando precisar buscar insights para um cliente específico.
3. **Armazenamento de Metadados:** Além do token, armazene também o `expires_in` (data de expiração) e o `user_id` do Instagram (que pode ser obtido com o token de longa duração através de uma chamada à API `/me`). Isso é útil para gerir a expiração do token e para identificar o utilizador do Instagram.
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 o token de longa duração.
   - 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)
     - `instagram_access_token` : `{{ $json.access_token }}` (do nó HTTP Request anterior)
     - `instagram_token_expires_at` : Calcule a data de expiração com base em `expires_in` e a data/hora atual.
     - `instagram_user_id` : (Obtenha este ID fazendo uma chamada simples à API Graph `/me?fields=id` com o token de longa duração antes de armazenar).

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

Com o Access Token de longa duração armazenado, o n8n pode agora ser configurado para coletar periodicamente os dados de insights do Instagram 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

O primeiro passo no workflow agendado é recuperar os Access Tokens de longa duração 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 ativos e os IDs de utilizador do Instagram associados aos seus clientes KAIRÓS.
3. A saída deste nó será uma lista de itens, onde cada item representa um cliente com o seu token e ID de utilizador do Instagram. O n8n processará cada um desses itens sequencialmente ou em paralelo, dependendo da configuração do workflow.

Passo 3: Usando o token, como fazer as chamadas à API Graph para buscar os dados de Insights

Para cada cliente, você usará o Access Token e o ID do utilizador do Instagram para fazer chamadas à API Graph 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 de banco de dados (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):** `GET`
   - **URL:** Construa a URL usando o ID do utilizador do Instagram e os parâmetros de métrica e período. Use expressões para referenciar os dados do nó anterior.
     - Exemplo de URL: `https://graph.facebook.com/v19.0/{{ $json.instagram_user_id }}/insights`
   - **Parâmetros de Consulta (Query Parameters):**
     - `metric` : `impressions,reach,follower_count,engagement` (adicione outras métricas conforme necessário. Verifique a documentação da Meta para todas as métricas disponíveis).
     - `period` : `day` (ou `week` , `days_28` para períodos maiores. A disponibilidade depende da métrica).
     - `access_token` : `{{ $json.instagram_access_token }}` (o token do cliente).
   - `metric` : `impressions,reach,follower_count,profile_views,website_clicks,get_directions_clicks,phone_call_clicks,text_message_clicks,email_contacts`
   - `period` : `day`
   - `access_token` : `{{ $json.instagram_access_token }}`
   - **Insights de Perfil:** `GET /{ig-user-id}/insights?metric={metric}&period={period}`
     - Métricas comuns: `reach` , `impressions` , `profile_views` , `follower_count` (para crescimento, use `period=day` e calcule a diferença).
   - **Insights de Mídia (Postagens):** `GET /{ig-media-id}/insights?metric={metric}`
     - Métricas comuns: `engagement` , `impressions` , `reach` , `saves` , `comments` , `likes` .

- Para obter `ig-media-id`, primeiro liste as mídias do utilizador: `GET /{ig-user-id}/media`.

3. A resposta da API Graph 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 períodos.

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

Os dados brutos da API Graph 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` e à data dos insights.
   - `client_id_kairós`
   - `insight_date` (data do insight)
   - `impressions`
   - `reach`
   - `follower_count`
   - `engagement_rate` (calculado)
   - ... 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 (e não diretamente à API do Instagram)

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

1. **Segurança:** O Access Token do Instagram nunca é exposto 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 da Meta, implementando lógicas de cache e controlo de taxa para evitar exceder os limites da API.

4. **Flexibilidade:** Se a API da Meta 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/instagram-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 MyPerformancePage = () => {
  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(`${process.env.NEXT_PUBLIC_N8N_API_URL}/api/instagram-insights?
clientId=${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...</p>;
  if (error) return <p>Erro ao carregar insights: {error}</p>;
  if (!insights) return <p>Nenhum insight disponível.</p>;

  // ... renderizar os dados no próximo passo
  return (
    <div>
```

```
      <h1>Meu Desempenho no Instagram</h1>
      {/* Exibir dados aqui */}
      <pre>{JSON.stringify(insights, null, 2)}</pre>
    </div>
  );
};

export default MyPerformancePage;
```

Passo 2: Um exemplo simples de como exibir esses dados 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 `MyPerformancePage`:

JSX

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

const MyPerformancePage = () => {
  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(`${process.env.NEXT_PUBLIC_N8N_API_URL}/api/instagram-insights?
clientId=${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...</p>;
  if (error) return <p>Erro ao carregar insights: {error}</p>;
  if (!insights) return <p>Nenhum insight disponível.</p>;

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

      <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fit, minmax(250px, 1fr))', gap: '20px'
}}>
        <div style={cardStyle}>
          <h2>Alcance</h2>
          <p style={metricValueStyle}>{insights.reach || 'N/A'}</p>
          <p style={metricDescriptionStyle}>Número de contas únicas que viram qualquer uma das suas
publicações.</p>
        </div>

        <div style={cardStyle}>
          <h2>Impressões</h2>
```

```
          <p style={metricValueStyle}>{insights.impressions || 'N/A'}</p>
          <p style={metricDescriptionStyle}>Número total de vezes que as suas publicações foram vistas.</p>
        </div>

        <div style={cardStyle}>
          <h2>Seguidores</h2>
          <p style={metricValueStyle}>{insights.follower_count || 'N/A'}</p>
          <p style={metricDescriptionStyle}>Número atual de seguidores.</p>
        </div>

        <div style={cardStyle}>
          <h2>Engajamento</h2>
          <p style={metricValueStyle}>{insights.engagement || 'N/A'}</p>
          <p style={metricDescriptionStyle}>Soma de likes, comentários, salvamentos e cliques no perfil.</p>
        </div>

        {/* Adicione mais métricas conforme necessário */}
        {insights.profile_views && (
          <div style={cardStyle}>
            <h2>Visualizações de Perfil</h2>
            <p style={metricValueStyle}>{insights.profile_views}</p>
            <p style={metricDescriptionStyle}>Número de vezes que o seu perfil foi visualizado.</p>
          </div>
        )}

        {insights.website_clicks && (
          <div style={cardStyle}>
            <h2>Cliques no Site</h2>
            <p style={metricValueStyle}>{insights.website_clicks}</p>
            <p style={metricDescriptionStyle}>Cliques no link do seu site no perfil.</p>
          </div>
        )}
      </div>

      {/* Exemplo de como exibir dados diários se você os processar */}
      {insights.impressions_daily && insights.impressions_daily.length > 0 && (
        <div style={{ marginTop: '40px' }}>
          <h2>Impressões Diárias</h2>
          <ul>
            {insights.impressions_daily.map((data, index) => (
              <li key={index}>Data: {new Date(data.end_time).toLocaleDateString()}, Impressões: {data.value}
</li>
            ))}
          </ul>
        </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 MyPerformancePage;
```

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 Instagram" 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 Graph da Meta. Ao seguir estas diretrizes, você pode garantir uma integração segura, eficiente e escalável, proporcionando aos seus clientes da área de comunicação política as métricas de desempenho do Instagram de que necessitam.

Lembre-se de que a plataforma Meta for Developers e a API Graph estão em constante evolução. Consulte sempre a documentação oficial da Meta para as informações mais recentes sobre versões da API, permissões e processos de App Review.

Referências

[1] Meta for Developers. *Documentação da API Graph*. Disponível em: https://developers.facebook.com/docs/graph-api

[2] Meta for Developers. *Login com Facebook*. Disponível em: https://developers.facebook.com/docs/facebook-login

[3] Meta for Developers. *API de Exibição Básica do Instagram*. Disponível em: https://developers.facebook.com/docs/instagram-basic-display-api

[4] Meta for Developers. *API Graph do Instagram*. Disponível em: https://developers.facebook.com/docs/instagram-api

[5] Meta for Developers. *Insights do Instagram*. Disponível em: https://developers.facebook.com/docs/instagram-api/guides/insights

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

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

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