

Configuração Completa de Integração Social: Instagram, Twitter e Azure

1. Credenciais da API do Meta (Instagram)

1.1 Criação do Aplicativo no Meta Developer Dashboard

1.1.1 Acesso e Configuração Inicial O processo de criação de um aplicativo no **Meta Developer Dashboard** representa o fundamento essencial para qualquer integração com as APIs do Instagram e Facebook. Para iniciar, é necessário possuir uma **conta Facebook válida** e acessar o portal oficial em developers.facebook.com. Após o login, o desenvolvedor deve navegar até a seção **“Meus Aplicativos” (My Apps)**, localizada no menu superior do dashboard, onde encontrará a opção para criar um novo aplicativo.

A seleção do tipo de aplicativo constitui uma decisão crítica que determinará as capacidades e permissões disponíveis. Para integrações de negócios que envolvem **publicação de conteúdo no Instagram**, a opção recomendada é **“Outro” (Other)**, seguida da subcategoria **“Negócios” (Business)**. Esta configuração habilita o acesso ao **Instagram Graph API**, que é a interface oficial para interação programática com contas Instagram Business e Creator. A escolha inadequada do tipo de aplicativo pode resultar em limitações de permissões que impedirão completamente a funcionalidade desejada, particularmente o acesso à permissão `instagram_content_publish` que requer verificação de negócios subsequente.

Durante o fluxo de criação, o sistema solicita informações básicas incluindo **nome do aplicativo**, **email de contato** e, opcionalmente, **associação a Business Manager**. O nome do aplicativo deve ser descritivo e alinhado à marca do produto final, pois será visível durante fluxos de autenticação OAuth. O email de contato deve ser monitorado regularmente para receber notificações sobre revisões de aplicativo e alterações de política. Após a criação, o aplicativo é inicializado em **“Modo de Desenvolvimento” (Development Mode)**, que restringe o acesso a contas de teste explicitamente configuradas, proporcionando ambiente seguro para desenvolvimento.

1.1.2 Configuração de Produtos e Permissões Com o aplicativo criado, o desenvolvedor deve adicionar os produtos relevantes que habilitarão as funcionalidades de integração. No menu lateral do dashboard, a seção **“Adicionar Produto” (Add Product)** permite a seleção de **“Instagram”** e **“Webhooks”**. A adição do produto Instagram expõe automaticamente as configurações para integração com a plataforma, incluindo opções para tokens de acesso e permissões de API.

As permissões fundamentais para publicação de conteúdo incluem:

Permissão	Nível de Acesso	Descrição
<code>instagram_basic</code>	Standard	Acesso a informações básicas da conta e mídia
<code>instagram_content_publish</code>	Advanced	Publicação de fotos, vídeos, carrosséis e stories
<code>instagram_manage_insights</code>	Standard	Acesso a métricas de desempenho

Permissão	Nível de Acesso	Descrição
instagram_manage_comments	Standard	Moderação de comentários

É crucial compreender que `instagram_content_publish` opera inicialmente em **modo de desenvolvimento com acesso limitado**, e requer **acesso avançado (Advanced Access)** para operação em produção. Este acesso avançado só é concedido após conclusão do processo de **Business Verification** e **App Review**, detalhados na seção 4.

O produto **Webhooks** complementa a integração ao permitir recebimento de notificações em tempo real sobre eventos como publicações concluídas, menções em comentários e atualizações de stories. A configuração deste produto demanda preparação prévia de endpoint HTTPS, conforme explorado na seção 3.

1.2 Obtenção de META_APP_ID e META_APP_SECRET

1.2.1 Localização das Credenciais As credenciais de autenticação do aplicativo — **META_APP_ID** e **META_APP_SECRET** — constituem os elementos mais sensíveis da integração. Para localizá-las, o desenvolvedor deve acessar “**Configurações**” (**Settings**) → “**Básico**” (**Basic**) no menu lateral do dashboard. Nesta página, o “**ID do Aplicativo**” (**App ID**) é exibido prominentemente no topo como uma sequência numérica de aproximadamente 15 dígitos. Este identificador é **público por natureza** e utilizado em URLs de autenticação OAuth.

O “**Chave Secreta do Aplicativo**” (**App Secret**) é apresentado de forma **mascarada por padrão**, exigindo clique no botão “**Mostrar**” (**Show**) para revelação — momento em que o sistema pode solicitar **reautenticação por segurança**. O App Secret é uma **string alfanumérica de 32 caracteres** que funciona como senha mestra do aplicativo, sendo utilizado para:

- Assinar requisições de API server-to-server
- Validar autenticidade de webhooks recebidos
- Trocar códigos de autorização por tokens de acesso

A **exposição acidental do App Secret representa incidente de segurança grave**, permitindo que terceiros assumam a identidade do aplicativo. A Meta fornece mecanismo de **regeneração** que invalida imediatamente o valor anterior, devendo ser utilizado em caso de qualquer suspeita de comprometimento.

1.2.2 Configuração de Segurança Adicional Além das credenciais primárias, configurações complementares fortalecem a segurança da integração:

Configuração	Localização	Propósito
URLs de Redirecionamento OAuth Válidos	Configurações → Básico	Prevenir ataques de redirecionamento aberto
Domínios Permitidos para o Aplicativo	Configurações → Básico	Restringir origens de chamadas SDK

Configuração	Localização	Propósito
Modo de Desenvolvimento	Toggle no topo do dashboard	Limitar acesso a contas de teste

Para a aplicação Azure, o domínio `kling-video-generator.azurewebsites.net` deve ser explicitamente adicionado em ambas as configurações. A Meta realiza **validação rigorosa de correspondência exata** de URLs de callback, rejeitando qualquer variação de protocolo, porta ou path.

1.3 Configuração no Azure

1.3.1 Armazenamento em Azure Key Vault O Azure Key Vault representa o serviço de gerenciamento de secrets mais robusto para ambientes Microsoft Azure, oferecendo **criptografia em repouso com chaves gerenciadas pela Microsoft ou pelo cliente (BYOK)**, controle de acesso baseado em Azure Active Directory, e logging completo de operações para auditoria.

O processo de configuração inicia-se com a criação ou acesso a um Key Vault na **mesma região da aplicação**. Dentro do Key Vault, dois secrets são criados na seção **“Secrets”**:

Nome do Secret	Valor	Observação
META-APP-ID	[App ID do dashboard]	Identificador público do aplicativo
META-APP-SECRET	[App Secret do dashboard]	Credencial confidencial — máxima proteção

Cada secret suporta **versionamento automático**, permitindo rotação sem interrupção de serviço. Recomenda-se configurar **datas de expiração** como lembrete para renovação periódica. As **políticas de acesso** devem conceder permissões **“Get”** e **“List”** à **identidade gerenciada (Managed Identity)** do Azure App Service, eliminando necessidade de armazenar credenciais de serviço no código.

1.3.2 Integração com Azure App Service A integração entre Key Vault e App Service utiliza **referências diretas de secrets em variáveis de ambiente**, com sintaxe especial que o App Service resolve em runtime. No menu **“Configuração” (Configuration) → “Configurações de Aplicativo” (Application settings)**, criam-se:

Nome da Configuração	Valor
META_APP_ID	@Microsoft.KeyVault(SecretUri=https://<vault-name>.vault.azure.net/secrets/META-APP-ID/)

Nome da Configuração	Valor
META_APP_SECRET	@Microsoft.KeyVault(SecretUri=https://<vault-name>.vault.azure.net/secrets/META-APP-SECRET/)

A **omissão da versão** no URI faz com que o App Service sempre utilize a versão mais recente do secret, facilitando rotações sem reimplantação. A validação de funcionamento é realizada através de **logs de inicialização da aplicação** e testes de conectividade com endpoints de debug da Meta Graph API.

2. Credenciais da API do Twitter (X)

2.1 Criação de Conta e Aplicação no Twitter Developer Portal

2.1.1 Acesso ao Portal de Desenvolvedores O **Twitter Developer Portal**, acessível em developer.twitter.com, passou por transformações significativas desde 2022, incluindo reestruturação de planos de acesso e revisão de políticas. O processo inicia com **login utilizando conta Twitter existente**, seguido de inscrição no programa de desenvolvedores que pode incluir **verificação de identidade e justificativa do caso de uso**.

A estrutura organizacional adota hierarquia de **Projetos (Projects)** contendo **Aplicativos (Apps)**, permitindo agrupamento lógico por produto ou ambiente. Para o cenário descrito, recomenda-se:

Elemento	Nome Sugerido	Propósito
Projeto	“Kling Video Generator”	Container de alto nível
App — Desenvolvimento	klings-video-generator-dev	Testes locais e integração
App — Produção	klings-video-generator-prod	Carga real de usuários

2.1.2 Configuração do Projeto e Aplicativo Dentro de um projeto, a criação de aplicativo solicita **nome único global, descrição e ambiente de execução**. A seleção de ambiente influencia configurações padrão de callback URLs e limites de taxa. Para desenvolvimento inicial, **“Development”** é apropriado; antes da promoção para produção, alterar para **“Production”** com atualização das URLs de callback.

A configuração de **“User authentication settings”** habilita fluxos OAuth, com seleção de **“Read and Write” permissions** para capacidade de publicação. O campo **“Type of App”** determina o fluxo de autorização: **“Web App, Automated App or Bot”** para aplicações servidor-servidor com autenticação de usuário.

2.2 Obtenção de `TWITTER_CLIENT_ID` e `TWITTER_CLIENT_SECRET`

2.2.1 Geração de Credenciais de Autenticação As credenciais são obtidas na seção “**Keys and Tokens**” do dashboard de aplicativos. O Twitter implementa **dois modelos de autenticação**:

Modelo	Credenciais	Uso Recomendado
OAuth 2.0	Client ID, Client Secret	Novas integrações, maior segurança
OAuth 1.0a	API Key, API Secret Key, Access Token, Access Token Secret	Compatibilidade legada, algumas operações de mídia

Para **OAuth 2.0**, as credenciais principais são:

- **`TWITTER_CLIENT_ID`** (Client ID): identificador público do aplicativo
- **`TWITTER_CLIENT_SECRET`** (Client Secret): **credencial confidencial para autenticação server-to-server**

Além destas, o fluxo de publicação em nome de usuário requer **Access Token** obtido através de autorização OAuth completa, ou gerado diretamente no portal para conta própria em modo de desenvolvimento.

2.2.2 Configuração de Permissões de API Os **escopos (scopes)** definem operações permitidas:

Escopo	Descrição	Necessário para
<code>tweet.read</code>	Leitura de tweets do usuário autenticado	Verificação de estado
<code>tweet.write</code>	Criação de novos tweets	Publicação de conteúdo
<code>users.read</code>	Acesso a informações de perfil	Identificação de conta
<code>media.write</code>	Upload de mídia	Anexos em tweets

A configuração de **callback URLs** em “**Authentication**” → “**Callback URLs**” deve incluir endpoints exatos como `https://kling-video-generator.azurewebsites.net/api/auth/callback/twitter`. O **Twitter não suporta wildcards**, exigindo registro explícito de cada variação.

2.3 Configuração no Azure

2.3.1 Armazenamento Seguro O procedimento no Azure Key Vault segue padrão idêntico ao das credenciais Meta:

Nome do Secret	Valor	Tags Recomendadas
TWITTER-CLIENT-ID	[Client ID do portal]	Environment:Production, Platform:Twitter
TWITTER-CLIENT-SECRET	[Client Secret do portal]	RotationSchedule:Quarterly
TWITTER-ACCESS-TOKEN	[Token de usuário]	AccountType:Service, Expires:[data]

Para **tokens de acesso de usuário**, considerar armazenamento em **Azure Redis Cache com persistência criptografada** ou **banco de dados com criptografia em campo**, dado seu tempo de vida limitado e necessidade de refresh.

2.3.2 Integração com Aplicação As referências de Key Vault em configurações de aplicativo:

Nome da Configuração	Valor de Referência
TWITTER_CLIENT_ID	@Microsoft.KeyVault(SecretUri=https://<vault-name>.vault.azure.net/secrets/TWITTER-CLIENT-ID/)
TWITTER_CLIENT_SECRET	@Microsoft.KeyVault(SecretUri=https://<vault-name>.vault.azure.net/secrets/TWITTER-CLIENT-SECRET/)

O **teste de conectividade** deve validar: (a) leitura correta de credenciais, (b) capacidade de obter token de acesso válido, e (c) execução de publicação de teste em conta dedicada. O **Azure Application Insights** deve rastrear métricas de latência, taxa de sucesso e consumo de quota de rate limit.

3. Webhook do Instagram para Notificações

3.1 Preparação do Endpoint na Aplicação Azure

3.1.1 Implementação do Handler de Verificação O protocolo de webhooks do Instagram implementa **mecanismo de verificação desafio-resposta (challenge-response)** para confirmar propriedade do endpoint. Quando configurado no dashboard, o Meta envia **requisição HTTP GET** contendo:

Parâmetro	Valor Esperado	Descrição
hub.mode	"subscribe"	Indica operação de subscrição

Parâmetro	Valor Esperado	Descrição
hub.verify_token	ixnwgqkadhpfokjmxzhhrbxq	Token configurado pelo desenvolvedor
hub.challenge	[string aleatória]	Valor que deve ser retornado

O endpoint `https://klings-video-generator.azurewebsites.net/api/social/webhooks/instagram` deve implementar lógica que **valide o verify_token** e **retorne o challenge** em caso de correspondência. Falha em qualquer aspecto resulta em **rejeição da subscrição**.

3.1.2 Estrutura do Código de Verificação Implementação exemplar em Python com FastAPI:

```
from fastapi import FastAPI, Request, HTTPException

app = FastAPI()
VERIFY_TOKEN = "ixnwgqkadhpfokjmxzhhrbxq" # Carregar de variável de ambiente/Key Vault

@app.get("/api/social/webhooks/instagram")
async def verify_webhook(request: Request):
    mode = request.query_params.get("hub.mode")
    token = request.query_params.get("hub.verify_token")
    challenge = request.query_params.get("hub.challenge")

    if mode == "subscribe" and token == VERIFY_TOKEN:
        return int(challenge) # Meta espera retorno como número inteiro
        raise HTTPException(status_code=403, detail="Verification failed")

@app.post("/api/social/webhooks/instagram")
async def receive_webhook(request: Request):
    # Validação de assinatura com APP_SECRET
    signature = request.headers.get("X-Hub-Signature-256")
    payload = await request.body()

    if not verify_signature(payload, signature, APP_SECRET):
        raise HTTPException(status_code=403, detail="Invalid signature")

    # Processamento assíncrono do evento
    event_data = await request.json()
    await process_event_async(event_data) # Enfileirar para worker

    return {"status": "received"} # Confirmação rápida ao Meta
```

Elementos críticos de segurança:

- O **verify token** nunca deve ser **hardcoded** — carregar de Azure Key Vault ou variável de ambiente

- Comparação de tokens deve ser **case-sensitive e constant-time** (usar `hmac.compare_digest` quando disponível)
- Resposta ao challenge deve ser **puramente o valor numérico**, sem markup HTML ou JSON

3.1.3 Processamento de Eventos de Webhook Após verificação, o Meta envia **notificações via HTTP POST** com payload JSON estruturado:

```
{
  "object": "instagram",
  "entry": [{
    "id": "17841405793187218",
    "time": 1620000000,
    "changes": [{
      "field": "mentions",
      "value": {
        "media_id": "17918195224117851",
        "comment_id": "17892250648466172"
      }
    }]
  }]
}
```

O processamento deve seguir **padrão assíncrono e resiliente**:

Etapa	Ação	Objetivo
1. Recebimento	Validar assinatura, responder HTTP 200	Evitar timeouts e retries do Meta
2. Enfileiramento	Publicar em Azure Service Bus/Queue Storage	Desacoplamento e durabilidade
3. Processamento	Worker consome e executa lógica de negócio	Escalabilidade independente

O Meta implementa **retry com backoff exponencial** para falhas, mas **notificações duplicadas são possíveis** — a aplicação deve garantir **idempotência** no processamento.

3.2 Registro do Webhook no Meta Developer Dashboard

3.2.1 Acesso à Configuração de Webhooks No **Meta Developer Dashboard**, navegar para “**Webhooks**” no menu lateral esquerdo. A interface apresenta objetos disponíveis para assinatura; selecionar “**Instagram**” como objeto de interesse. O botão “**Assinar a este objeto**” (**Subscribe to this object**) inicia o fluxo de configuração.

3.2.2 Configuração de URL e Token de Verificação

No formulário de subscrição, inserir:

Campo	Valor
Callback URL	https:// kling-video-generator.azurewebsites.net/ api/social/webhooks/instagram
Verify Token	ixnwgqkadhpfokjmxzhhrbxq

Requisitos técnicos críticos:

- **HTTPS obrigatório** — certificado válido, não autoassinado
- **Endpoint acessível publicamente** — sem firewall ou restrições de IP
- **Tempo de resposta < 5 segundos** para requisição de verificação

Após clique em “**Verificar e Salvar**” (**Verify and Save**), o Meta emite imediatamente requisição GET de teste. **Sucesso** exibe indicador verde; **falha** apresenta mensagem diagnóstica como “URL não acessível” ou “Token de verificação incorreto”.

3.2.3 Validação e Testes

Tipo de Teste	Método	Objetivo
Verificação básica	curl/Postman com parâmetros simulados	Confirmar lógica de challenge-response
Teste integrado	Botão “Test” no dashboard	Validar recebimento de payload de exemplo
Evento real	Ação manual em conta Instagram de teste	Confirmar processamento end-to-end

O **Azure Application Insights** deve ser configurado para rastrear: requisições recebidas, latência de processamento, taxas de erro por tipo de evento, e ausência de heartbeats (indicativo de desativação do webhook pelo Meta).

3.3 Assinatura de Campos de Webhook

3.3.1 Seleção de Campos Relevantes

Após subscrição base, selecionar campos específicos de evento:

Campo	Evento Disparado	Caso de Uso Principal
mentions	Conta é @mencionada	Engajamento, resposta automática

Campo	Evento Disparado	Caso de Uso Principal
<code>story_insights</code>	Métricas de story disponíveis	Análise de efetividade de conteúdo
<code>media_publish</code>	Mídia publicada com sucesso	Confirmação de operação, atualização de status
<code>comments</code>	Novo comentário em publicação	Moderação, análise de sentimento

A seleção deve ser **deliberada e minimalista** — campos desnecessários aumentam volume de requisições e custo de infraestrutura sem valor agregado.

3.3.2 Teste de Integridade do Webhook Para **desenvolvimento local**, ferramentas como **ngrok** criam túnel seguro expondo servidor local à internet:

```
ngrok http 8000 # Gera URL pública temporária como https://abc123.ngrok.io
```

Esta URL pode ser registrada temporariamente no dashboard da Meta para iteração rápida, **eliminando necessidade de deploy contínuo para Azure durante desenvolvimento**. A versão paga do ngrok oferece **domínios customizados persistentes**, úteis para ambientes de staging.

A **validação de assinatura de payload** com **APP_SECRET** é obrigatória em produção:

```
import hmac
import hashlib

def verify_signature(payload: bytes, signature: str, app_secret: str) -> bool:
    """Verifica HMAC-SHA256 do payload contra assinatura recebida."""
    expected = hmac.new(
        app_secret.encode('utf-8'),
        payload,
        hashlib.sha256
    ).hexdigest()
    # Header format: "sha256=<hex_digest>"
    return hmac.compare_digest(f"sha256={expected}", signature)
```

O uso de `hmac.compare_digest` em vez de comparação de strings direta é **essencial para prevenir ataques de timing** que poderiam revelar informações sobre o secret correto.

4. Verificação de Negócios do Meta para Permissão `instagram_content_publish`

4.1 Pré-requisitos para Business Verification

4.1.1 Estrutura Organizacional Necessária A verificação de negócios (Business Verification) é **processo obrigatório** estabelecido pela Meta para confirmar identidade legal de organizações que desejam acesso a permissões sensíveis de API. Desde **1º de fevereiro de 2023**, a permissão `instagram_content_publish` em nível de produção **exige vinculação a Business Manager verificado**.

Os pré-requisitos estruturais incluem:

Elemento	Requisito	Verificação
Meta Business Manager	Conta ativa em business.facebook.com	Email corporativo verificado
Vínculo do aplicativo	App do Developer Dashboard associado ao BM	Transferência de propriedade para entidade legal
Administrador verificado	Pessoa com identidade confirmada na Meta	Documento oficial validado

A vinculação do aplicativo ao Business Manager é realizada em “**Configurações**” → “**Avançado**” → “**Business Manager**” no dashboard do aplicativo. Esta operação é **irreversível** e estabelece que a entidade corporativa, não o indivíduo desenvolvedor, é responsável legal pelas ações do aplicativo.

4.1.2 Documentação Legal Requerida O processo exige **documentação comprobatória de existência e operação legal**:

Documento	Finalidade	Requisitos de Qualidade
Certificado de registro de empresa	Prova de constituição legal	Alta resolução (300+ DPI), bordas visíveis, texto legível
Identificação do representante legal	Confirmação de autoridade	Documento com foto, não expirado
Comprovante de endereço comercial	Validação de localização física	Data recente (< 3 meses), em nome da empresa
Contrato social/ estatuto	Definição de estrutura de governança	Registro em cartório ou equivalente

Documentos em **idiomas não-romanos** podem requerer **tradução juramentada**. A Meta utiliza **serviços de verificação de terceiros e inteligência artificial** para validar autenticidade; submissões de baixa qualidade ou suspeitas de adulteração resultam em **rejeição permanente com limitada possibilidade de reapelação**.

4.2 Processo de Verificação de Negócios

4.2.1 Início da Verificação no Business Manager O ponto de entrada é “**Configurações de Segurança**” → “**Centro de Segurança**” no Business Manager. A disponibilidade do botão “**Iniciar Verificação**” depende de fatores como idade da conta, histórico de atividade e elegibilidade regional. Algumas contas podem **não ter acesso imediato**, requerendo contato com suporte para ativação manual.

O formulário de verificação coleta:

Campo	Orientação
Nome legal completo	Exata correspondência com documentação oficial
Endereço comercial	Completo: rua, número, complemento, cidade, estado, CEP, país
País de operação principal	Determina requisitos regulatórios específicos
Telefone comercial	Linha atendida por sistema ou pessoa autorizada

A Meta realiza **verificação cruzada automática** com bases de dados comerciais; inconsistências detectadas prolongam significativamente o processo.

4.2.2 Submissão de Informações Comerciais Além de informações básicas, o sistema pode solicitar **detalhes operacionais**: número de funcionários, receita anual estimada, presença web em diretórios de negócios, e histórico de publicidade na plataforma Meta. Estes dados constroem **perfil de confiabilidade** e determinam nível de risco associado à concessão de acesso a APIs sensíveis.

A **verificação de telefone** é realizada via **chamada automatizada ou SMS** com código de confirmação. Falha nesta etapa **bloqueia progressão do processo**.

4.2.3 Verificação de Domínio e Email A confirmação de propriedade de domínio estabelece **controle técnico sobre presença online**:

Método	Implementação	Preferência
Registro DNS TXT	Adicionar entrada específica no DNS do domínio	Recomendado — mais seguro, persistente
Meta tag HTML	Incluir tag em <head> da página inicial	Conveniente, mas visível no código-fonte
Upload de arquivo	Arquivo HTML específico na raiz do domínio	Alternativa quando DNS não é acessível

A **verificação de email corporativo** exige que administradores possuam endereço no domínio verificado (ex: admin@empresa.com). A **autenticação de dois fatores (2FA)** é **obrigatória** para conclusão, representando camada adicional de segurança.

4.3 Revisão de Aplicativo (App Review) para `instagram_content_publish`

4.3.1 Solicitação de Acesso Avançado Com verificação de negócio concluída, o desenvolvedor torna-se elegível para **solicitar acesso avançado**. No Developer Dashboard: “**Revisão do Aplicativo**” → “**Permissões e Recursos**”, localizar `instagram_content_publish` e clicar “**Obter Acesso Avançado**” (Get Advanced Access).

O wizard de submissão solicita **descrição detalhada do caso de uso**, incluindo:

- O que a aplicação faz e **por que precisa publicar no Instagram**
- Como usuários autenticam e autorizam o aplicação
- Fluxo completo de criação e publicação de conteúdo
- Medidas de proteção de dados e privacidade implementadas

Respostas genéricas ou copiadas de templates são frequentemente rejeitadas — o revisor busca evidência de compreensão completa das implicações de acesso à API.

4.3.2 Preparação de Materiais de Submissão O screencast em vídeo é elemento mais crítico e frequentemente decisivo para aprovação. Requisitos:

Aspecto	Especificação
Duração	Mínimo 2 minutos, idealmente 3-5
Conteúdo	Fluxo completo: autenticação → criação de conteúdo → publicação → resultado no Instagram
Qualidade	Resolução adequada, narração ou legendas explicativas
Áudio	Claro, ou legendas detalhadas se sem áudio

O vídeo deve demonstrar **explicitamente** a tela de consentimento de permissões da Meta com `instagram_content_publish` visível, provando que usuários são informados sobre o acesso concedido.

4.3.3 Fornecimento de Credenciais de Teste Para validação independente, a Meta requer:

Item	Descrição
Conta de teste Instagram Business/ Creator	Vinculada a página Facebook, com histórico mínimo de atividade
Token de acesso de teste	Gerado para conta de teste, com permissões necessárias
Documentação de reprodução	Passos claros para replicar funcionalidade demonstrada

Nota importante: Contas de teste do Facebook **não podem ser vinculadas a contas Instagram reais**, limitando a preparação de ambiente completo. Recomenda-se **explicitar esta limitação na submissão** e instruir o revisor sobre pré-requisitos necessários.

4.4 Acompanhamento e Aprovação

4.4.1 Ciclo de Revisão

Fase	Prazo Típico	Observações
Análise inicial	5-7 dias úteis	Pode variar com volume de submissões
Solicitação de informações adicionais	+3-5 dias	Comum para casos de uso inovadores ou complexos
Re-submissão após rejeição	+5-7 dias	Requer correções substantivas

A comunicação de resultado ocorre via **email para administradores do aplicativo** e notificação no dashboard. **Aprovação** ativa imediatamente a permissão; **rejeição** inclui motivo específico categorizado (ex: “experiência de usuário insuficiente”, “caso de uso não claro”, “violação de política”).

4.4.2 Pós-Aprovação Com `instagram_content_publish` ativa em produção, implementar:

Aspecto	Recomendação
Monitoramento de rate limits	Tracking de headers <code>X-RateLimit-Remaining</code> e <code>X-RateLimit-Reset</code>
Tratamento de erros	Códigos específicos: 403 (violação de política de conteúdo), 429 (limite excedido), 401 (autenticação falha)
Retry logic	Backoff exponencial respeitando <code>Retry-After</code>
Logging estruturado	Correlação de request IDs para troubleshooting com suporte Meta

A API do Instagram impõe **limites rigorosos**: tipicamente **25 publicações por dia** para contas em desenvolvimento, expandindo para limites maiores mas ainda restritivos para aplicações aprovadas. O excedente resulta em **throttling temporário** ou **suspensão de acesso em casos graves**.

5. Integração e Validação Final do Sistema

5.1 Testes End-to-End

5.1.1 Validação de Publicação no Instagram O fluxo completo de publicação via **Instagram Graph API** envolve etapas distintas:

Etapa	Endpoint	Descrição
1. Autenticação OAuth	<code>https://www.facebook.com/v18.0/dialog/oauth</code>	Usuário autoriza aplicação
2. Troca de código por token	<code>POST /oauth/access_token</code>	Obtém access token de curta duração
3. Extensão para token longo	<code>GET /oauth/access_token?grant_type=fb_exchange_token</code>	Token de 60 dias
4. Criação de container de mídia	<code>POST /{ig-user-id}/media</code>	Upload ou referência URL de mídia
5. Publicação do container	<code>POST /{ig-user-id}/media_publish</code>	Publicação efetiva no perfil

Para **imagens**, o container é criado com `image_url` e `caption`. Para **vídeos**, utiliza-se `video_url` com parâmetros adicionais de `thumbnail`. A confirmação final requer **verificação visual no perfil Instagram** — a API retorna `media_id` que pode ser consultado para status de processamento.

5.1.2 Validação de Publicação no Twitter

Aspecto	API v2 (OAuth 2.0)	API v1.1 (OAuth 1.0a)
Autenticação	Client ID + Client Secret + Access Token	API Key + API Secret + Access Token + Token Secret
Endpoint de publicação	<code>POST /2/tweets</code>	<code>POST /1.1/statuses/update.json</code>
Limite de caracteres	280 (4.000 para Twitter Blue)	280
Upload de mídia	<code>POST /2/media/upload</code> (chunked para arquivos grandes)	<code>POST /1.1/media/upload.json</code>

O **upload de mídia no Twitter** para arquivos > 5MB requer **fluxo em múltiplas etapas**: INIT (inicialização), APPEND (chunks de dados), FINALIZE (confirmação) e opcionalmente STATUS (verificação de processamento). Esta complexidade é abstraída por bibliotecas cliente como **tweepy**, mas compreensão do fluxo é essencial para debugging.

5.1.3 Validação de Webhook

Cenário de Teste	Método	Critério de Sucesso
Verificação de endpoint	Requisição GET simulada	Retorno correto de challenge
Evento de menção	Comentário com @mention em conta de teste	Recebimento de payload no Azure
Evento de publicação	Publicação de mídia via API	Confirmação via <code>media_publish</code> webhook
Assinatura válida	Payload com header <code>X-Hub-Signature-256</code>	Validação bem-sucedida com <code>APP_SECRET</code>
Assinatura inválida	Payload com signature modificada	Rejeição com HTTP 403

O **Azure Application Insights** deve mostrar telemetria completa: requisições recebidas, tempo de processamento, resultados de validação de assinatura, e quaisquer exceções não tratadas.

5.2 Monitoramento e Operações

5.2.1 Logging e Observabilidade Configuração recomendada de **Azure Application Insights**:

Componente	Configuração	Finalidade
Request tracking	Automático para HTTP	Latência e taxa de sucesso de endpoints
Dependency tracking	Chamadas às APIs Meta e Twitter	Identificação de gargalos externos
Custom events	Publicação bem-sucedida/falha	Métricas de negócio
Exception tracking	Captura automática	Diagnóstico de falhas
Alerts	Taxa de erro > 5%, latência > 2s, ausência de heartbeats	Notificação proativa de problemas

Dashboards customizados devem correlacionar eventos de webhook com operações de API, fornecendo **visibilidade end-to-end** do fluxo de publicação.

5.2.2 Gerenciamento de Tokens

Tipo de Token	Tempo de Vida	Mecanismo de Renovação
Meta Access Token (curto)	1 hora	Troca por token longo (60 dias)
Meta Access Token (longo)	60 dias	Refresh automático antes da expiração
Twitter Access Token (OAuth 2.0)	2 horas	Refresh token (com escopo <code>offline.access</code>)
Twitter Access Token (OAuth 1.0a)	Indefinido (até revogação)	Não aplicável

A implementação de **refresh automático** deve: (a) monitorar tempo de expiração, (b) renovar com margem de segurança (ex: 24h antes para Meta, 10 min antes para Twitter), (c) persistir novo token de forma segura, e (d) implementar fallback para reautenticação manual em caso de falha de refresh.

5.3 Considerações de Segurança e Compliance

5.3.1 Proteção de Dados

Camada	Medida	Implementação
Em repouso	Criptografia AES-256	Azure Key Vault, Azure Storage Service Encryption
Em trânsito	TLS 1.2+	Configuração obrigatória em App Service
Acesso	Azure RBAC + Managed Identity	Princípio de menor privilégio
Auditoria	Azure Monitor logs	Retenção mínima 90 dias para compliance

A **conformidade com políticas de privacidade** das plataformas exige: consentimento explícito do usuário para cada permissão, capacidade de revogação de acesso, deleção de dados pessoais quando solicitado, e implementação de webhooks de deleção de dados da Meta para GDPR e regulamentações similares.

5.3.2 Resiliência e Recuperação

Padrão	Implementação	Cenário de Uso
Circuit Breaker	Biblioteca como <code>pybreaker</code> ou Polly	Falhas repetidas de API externa
Retry com backoff exponencial	Configuração em HTTP client	Throttling temporário, erros 5xx transitórios
Fallback degradado	Salvar rascunho local, notificar usuário	Indisponibilidade prolongada de serviço
Dead letter queue	Azure Service Bus DLQ	Eventos que falham processamento após retries

O **backup de configurações de integração** — incluindo IDs de aplicativo, URLs de webhook, e mapeamentos de permissões — deve ser realizado regularmente, com **documentação de procedimento de restauração** testada periodicamente em ambiente de disaster recovery.