ARQUITECTURA Y CHECKLIST

Comentario a DM en Instagram: arquitectura y app de Meta

Las tres piezas de un sistema que responde comentarios de Instagram con un DM, la checklist de la app de Meta según su documentación, los permisos de cada configuración y el código mínimo probado para el webhook y el filtro.

PARA	Devs que conectan cuentas profesionales de Instagram a su app
PROBADO CON	Node 22.22.3 · node --test · Windows 11
ACTUALIZADA	3 de noviembre de 2026
DOCUMENTACIÓN	developers.facebook.com/documentation/instagram-platform
ARTÍCULO	ecomlabs.dev/g/comenta
AUTOR	Luis Morales · @luismoralesdev

Índice

Paso 01	La arquitectura en tres pasos
Paso 02	Checklist de la app de Meta
Paso 03	El código mínimo
Paso 04	Antes de salir a producción

PASO 01

La arquitectura en tres pasos

Un sistema de comentario a DM para Instagram hace siempre lo mismo: recibe un webhook por cada comentario, filtra los que traen tu palabra clave y contesta con una respuesta privada que usa el ID del comentario. El resto (base de datos, reintentos, panel) depende de tu producto.

PASO	QUÉ RECIBE	QUÉ HACE
1 · Webhook	`GET` de verificación y `POST` con `X-Hub-Signature-256`	Responde `hub.challenge` si el verify token coincide. Valida la firma HMAC-SHA256 del cuerpo crudo con el App Secret y responde con éxito pronto.
2 · Filtro	`entry[].changes[]` con `field: "comments"` y el texto del comentario	Compara palabras completas, sin mayúsculas ni acentos. Guarda el ID del comentario para no atenderlo dos veces: Meta reintenta durante 36 horas.
3 · Respuesta privada	El ID de la cuenta y el ID del comentario	`POST /<IG_ID>/messages` con `recipient.comment_id` y `message.text`.

Los límites de la respuesta privada, según la guía de Meta:

- Solo se puede enviar un mensaje a quien hizo el comentario.
- Dentro de los 7 días siguientes al comentario; en un Live, solo durante la transmisión.
- Seguimiento solo si la persona responde, dentro de las 24 horas siguientes.
- Llega a la bandeja de entrada si sigue la cuenta, o a Solicitudes si no.

PASO 02

Checklist de la app de Meta

En el orden de la documentación de Meta (consultada el 2 de noviembre de 2026). Meta recomienda Instagram API with Unified Login para integraciones nuevas.

- [ ] La cuenta que conectas es profesional (empresa o creador) y pública: las cuentas privadas no reciben notificaciones de comentarios.
- [ ] Creaste una app de Meta de tipo negocio y la conectaste a un negocio con la verificación completa (con Unified Login, la app es de un portafolio comercial).
- [ ] Agregaste el producto Instagram y elegiste una configuración: Unified Login, Instagram Login o Facebook Login.
- [ ] Pides solo los permisos de esa configuración (tabla abajo).
- [ ] Tu callback es HTTPS con un certificado válido (no autofirmado) y responde el `GET` de verificación.
- [ ] Guardaste la URL de callback y el verify token en el App Dashboard y te suscribiste al campo `comments`.
- [ ] Suscribiste cada cuenta conectada con un `POST` a `/<ID>/subscribed_apps`.

- [ ] Probaste con una cuenta profesional real invitada como Instagram Tester (no hay usuarios de prueba).
- [ ] La cuenta tiene encendido «Allow access to messages» en su app de Instagram; si no, la API de mensajes responde `(#200)`.
- [ ] App Review: app que se puede probar desde fuera, instrucciones para los revisores, un screencast por permiso, ícono de 1024×1024 y URL del aviso de privacidad.
- [ ] Tienes acceso avanzado a cada permiso, el negocio verificado y la app en modo Live.

CONFIGURACIÓN	HOST	PERMISOS PARA RESPUESTAS PRIVADAS
Unified Login	`graph.facebook.com`	`instagram_basic`, `instagram_manage_comments`, `instagram_manage_messages`
Instagram Login	`graph.instagram.com`	`instagram_business_basic`, `instagram_business_manage_comments`
Facebook Login	`graph.facebook.com`	`instagram_basic`, `instagram_manage_comments`, `pages_read_engagement`

> **CUIDADO**
>
> Dos páginas de Meta no coinciden: App Review dice que una app solo para tu propio negocio funciona con acceso estándar y sin revisión, pero la guía de webhooks pide acceso avanzado para las notificaciones `comments`. Si dependes de ese webhook, cuenta con App Review.

PASO 03

El código mínimo

Un ejemplo genérico que sigue la documentación pública de Meta, no el código de Comenta. Node 22, sin dependencias; 11 pruebas pasaron con `node --test` usando notificaciones falsas. No se llamó a la API real de Meta.

webhook.mjs: la verificación y la firma.

```
import { createHmac, timingSafeEqual } from "node:crypto";

// GET: Meta calls your callback URL with hub.* parameters when you set up webhooks.
export function verifyChallenge(params, verifyToken) {
  if (params.get("hub.mode") !== "subscribe" || params.get("hub.verify_token") !== verifyToken) return null;
  return params.get("hub.challenge");
}

// POST: X-Hub-Signature-256 is "sha256=" + the HMAC-SHA256 of the raw body, keyed with the app secret.
export function isSignedByMeta(rawBody, header, appSecret) {
  if (typeof header !== "string" || !header.startsWith("sha256=")) return false;
  const expected = createHmac("sha256", appSecret).update(rawBody).digest();
  const received = Buffer.from(header.slice("sha256=".length), "hex");
```

```
  return received.length === expected.length && timingSafeEqual(received, expected);
}
```

keyword.mjs: el filtro y la respuesta privada. `graph.instagram.com` es el host de Instagram Login; con Unified Login o Facebook Login usa `graph.facebook.com`.

```
// Lowercase and drop accents (é → e), but keep ñ: "año" and "ano" are different words.
const normalize = (s) => s.toLowerCase().normalize("NFD").replace(/(?!\u0303)\p{M}/gu, "").normalize("NFC");

// New comments whose text contains the keyword as a whole word.
export function matchingComments(payload, keyword, seen) {
  const word = normalize(keyword);
  const found = [];
  for (const entry of payload?.entry ?? []) {
    for (const change of entry.changes ?? []) {
      if (change.field !== "comments") continue;
      const id = change.value?.id ?? change.value?.comment_id;
      const words = normalize(change.value?.text ?? "").split(/[^\p{L}\p{N}]+/u);
      if (!id || seen.has(id) || !words.includes(word)) continue;
      seen.add(id);
      found.push({ accountId: entry.id, commentId: id });
    }
  }
  return found;
}

// Private reply: POST /<IG_ID>/messages with the comment ID as the recipient.
export async function sendPrivateReply({ accountId, commentId }, text, accessToken) {
  const res = await fetch(`https://graph.instagram.com/v26.0/${accountId}/messages`, {
    method: "POST",
    headers: { "content-type": "application/json", authorization: `Bearer ${accessToken}` },
    body: JSON.stringify({ recipient: { comment_id: commentId }, message: { text } }),
  });
  if (!res.ok) throw new Error(`private reply failed with ${res.status}`);
  return res.json();
}
```

Llama a `isSignedByMeta` con el cuerpo crudo antes de `JSON.parse`: según la guía de Messenger, Meta firma el payload con Unicode escapado, y volver a serializar el JSON cambia los bytes. En producción, guarda los IDs atendidos en tu base de datos con una restricción única en lugar del `Set`.

PASO 04

Antes de salir a producción

- [ ] La firma se compara con `timingSafeEqual`, nunca con `===`.
- [ ] Responder con éxito antes de empezar el trabajo lento.

- [ ] Una notificación duplicada no manda un segundo DM.
- [ ] El filtro no confunde *comentario* con *comenta*.
- [ ] Ningún log lleva tokens, el App Secret ni datos personales.
- [ ] Si la ley lo exige, avisas que el mensaje es automático.

Fuentes: developers.facebook.com/documentation/instagram-platform (private-replies, webhooks/setup, webhooks/examples, overview, app-review, instagram-api-with-unified-login) y la documentación de `crypto` de Node.js.