EcomLabs

ARCHITECTURE AND CHECKLIST

Instagram comment to DM: architecture and Meta app setup

The three pieces of a system that answers Instagram comments with a DM, the Meta app checklist from Meta's documentation, the permissions for each setup and the minimal tested code for the webhook and the filter.

FOR	Developers connecting Instagram professional accounts to their app
TESTED WITH	Node 22.22.3 · node --test · Windows 11
UPDATED	November 3, 2026
DOCS	developers.facebook.com/documentation/instagram-platform
ARTICLE	ecomlabs.dev/g/comenta-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Step 01	The architecture in three steps
Step 02	Meta app setup checklist
Step 03	The minimal code
Step 04	Before you go live

STEP 01

The architecture in three steps

An Instagram comment-to-DM system always does the same thing: it receives a webhook for each comment, keeps the ones that contain your keyword, and answers with a private reply that uses the comment ID. Everything else (database, retries, dashboard) depends on your product.

STEP	WHAT IT RECEIVES	WHAT IT DOES
1 · Webhook	A verification `GET` and a `POST` with `X-Hub-Signature-256`	Answers `hub.challenge` when the verify token matches. Validates the HMAC-SHA256 of the raw body with the App Secret and returns a success response promptly.
2 · Filter	`entry[].changes[]` with `field: "comments"` and the comment text	Compares whole words, ignoring case and accents. Stores the comment ID so it is not handled twice: Meta retries for 36 hours.
3 · Private reply	The account ID and the comment ID	`POST /<IG_ID>/messages` with `recipient.comment_id` and `message.text`.

Private reply limits, per Meta's guide:

- Only one message can be sent to the commenter.
- Within 7 days of the comment; on a Live, only during the broadcast.
- Follow-ups only if the person responds, within 24 hours of that response.
- It lands in the Inbox if they follow the account, or in Requests if not.

STEP 02

Meta app setup checklist

In the order Meta's documentation gives it (checked on November 2, 2026). Meta recommends the Instagram API with Unified Login for new integrations.

- [ ] The account you connect is professional (business or creator) and public: private accounts do not receive comment notifications.
- [ ] You created a Business type Meta app and connected it to a business that completed Business Verification (with Unified Login, the app is owned by a business portfolio).
- [ ] You added the Instagram product and picked a setup: Unified Login, Instagram Login or Facebook Login.
- [ ] You request only that setup's permissions (table below).
- [ ] Your callback is HTTPS with a valid certificate (not self-signed) and answers the verification `GET`.
- [ ] You saved the callback URL and verify token in the App Dashboard and subscribed to the `comments` field.
- [ ] You subscribed every connected account with a `POST` to `/<ID>/subscribed_apps`.
- [ ] You tested with a real professional account invited as an Instagram Tester (test users are not supported).

- [ ] The account has "Allow access to messages" turned on in its Instagram app; otherwise the messaging API returns `(#200)`.
- [ ] App Review: an app that can be tested externally, reviewer instructions, a screencast per permission, a 1024×1024 icon and a privacy policy URL.
- [ ] You have Advanced Access to each permission, a verified business and the app in Live mode.

SETUP	HOST	PERMISSIONS FOR PRIVATE REPLIES
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`

> **CAUTION**
> Two Meta pages disagree: App Review says an app that only serves your own business works with Standard Access and no review, but the webhooks guide requires Advanced Access for `comments` notifications. If you depend on that webhook, plan on App Review.

STEP 03

The minimal code

A generic example that follows Meta's public documentation, not Comenta's code. Node 22, no dependencies; 11 tests passed with `node --test` using fake notifications. Meta's real API was not called.

webhook.mjs: verification and signature.

```
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: the filter and the private reply. `graph.instagram.com` is the Instagram Login host; with Unified Login or Facebook Login use `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();
}
```

Call `isSignedByMeta` with the raw body before `JSON.parse`: per Meta's Messenger guide, the payload is signed with escaped unicode, and re-serializing the JSON changes the bytes. In production, keep handled IDs in your database under a unique constraint instead of the `Set`.

STEP 04

Before you go live

- [ ] The signature is compared with `timingSafeEqual`, never with `===`.
- [ ] You return a success response before starting slow work.
- [ ] A duplicate notification does not send a second DM.
- [ ] The filter does not mistake *comentario* for *comenta*.

- [ ] No log carries tokens, the App Secret or personal data.
- [ ] Where the law requires it, you disclose that the message is automated.

Sources: developers.facebook.com/documentation/instagram-platform (private-replies, webhooks/setup, webhooks/examples, overview, app-review, instagram-api-with-unified-login) and the Node.js `crypto` documentation.