EcomLabs

DECISION GUIDE

PWA vs native app for your store

A table to decide whether your store needs a native app or a PWA is enough, plus the minimum PWA checklist: manifest, service worker, HTTPS, icons, offline page and GA4 measurement. With the code we tested.

FOR	Online stores weighing an app
TESTED WITH	Chromium 151 (Playwright 1.62) · Node 22.22.3 · Windows 11
UPDATED	November 3, 2026
DOCS	web.dev/learn/pwa
ARTICLE	ecomlabs.dev/g/pwa-en
AUTHOR	Luis Morales · @luismoralesdev

Contents

Before you start	What this guide decides
Step 01	Decide: PWA or native app
Step 02	The manifest
Step 03	The service worker and its registration
Step 04	HTTPS, icons and the offline page
Step 05	Measure installs and launches
When you finish	Minimum PWA checklist

BEFORE YOU START

What this guide decides

Whether your store needs a native app or a PWA is enough, and what the minimum PWA has to include. A PWA is your same store plus three pieces: HTTPS, a manifest and a service worker. It installs from the browser, answers offline and, on iPhone, gets push once the shopper adds it to the Home Screen.

We tested the code in this guide on November 2, 2026 with Chromium 151 (Playwright 1.62) and Node 22.22.3 on Windows 11: the service worker controls the page after a reload, Chromium reads the manifest with no errors and no installability errors, and offline, a page never visited returns `offline.html`. We did not test a real install or an iPhone.

STEP 01

Decide: PWA or native app

Answer each row. If almost everything lands in the PWA column, start there and measure. This is our rule of thumb, not an Apple or Google rule.

CRITERION	A PWA IS ENOUGH IF	GO NATIVE IF
How they arrive	Google, ads, social	They open the app from its icon
Reorders	Every few months	Weekly or monthly, signed in
Push on iPhone	A bonus (Home Screen web apps only, iOS 16.4+)	Core to the business
Features	Catalog, cart, checkout	Live Activities (ActivityKit) or other native-only features
App store	Not your channel (a PWA can be listed too)	Your audience searches the App Store or Google Play
Team	The one already running the site	iOS, Android and each store's reviews
Fixed cost	The hosting you already pay	Apple: 99 USD per membership year · Google Play: US$25 once

STEP 02

The manifest

Save it as `manifest.webmanifest` at the root, serve it as `application/manifest+json` and link it on every page with `<link rel="manifest">`:

```
{
  "id": "/",
  "name": "Tienda Demo",
  "short_name": "Demo",
  "start_url": "/?source=pwa",
```

```
  "scope": "/",
  "display": "standalone",
  "background_color": "#ffffff",
  "theme_color": "#151412",
  "icons": [
    { "src": "/icons/icon-192.png", "sizes": "192x192", "type": "image/png" },
    { "src": "/icons/icon-512.png", "sizes": "512x512", "type": "image/png" }
  ]
}
```

- **What Chrome asks for** (web.dev): `short_name` or `name`, 192 px and 512 px icons, `start_url`, `display` set to `fullscreen`, `standalone`, `minimal-ui` or `window-controls-overlay`, and `prefer_related_applications` missing or `false`.
- `id`: without it, the identity is `start_url` (MDN), and changing that creates another app. If you already have an installed PWA with no `id`, set `id` to your current `start_url` before adding `?source=pwa`.
- `?source=pwa` on `start_url` marks launches from the icon so you can measure them.

STEP 03

The service worker and its registration

Save it as `/sw.js` at the domain root: its default scope is the folder it lives in. It caches only the shell; pages go to the network and, with no network, the offline page answers:

```
const CACHE = "shell-v1";
const SHELL = ["/offline.html", "/app.css", "/icons/icon-192.png"];

self.addEventListener("install", (event) => {
  event.waitUntil(caches.open(CACHE).then((cache) => cache.addAll(SHELL)));
});

self.addEventListener("activate", (event) => {
  event.waitUntil(
    caches.keys().then((keys) => Promise.all(keys.filter((k) => k !== CACHE).map((k) => caches.delete(k)))),
  );
});

self.addEventListener("fetch", (event) => {
  const { request } = event;
  const url = new URL(request.url);
  if (request.mode === "navigate") {
    event.respondWith(fetch(request).catch(() => caches.match("/offline.html")));
  } else if (request.method === "GET" && url.origin === location.origin && SHELL.includes(url.pathname)) {
    event.respondWith(caches.match(request).then((hit) => hit || fetch(request)));
  }
});
```

Register it at the top of `pwa.js`, loaded on every page. The first visit installs it; the page is controlled from the next load:

```
if ("serviceWorker" in navigator) {
  navigator.serviceWorker.register("/sw.js");
}
```

> **CAUTION**
>
> Do not cache product HTML, the cart or checkout: a shopper could see an old price, or a sold-out item shown as available. And when you change a shell file, bump the cache name (`shell-v2`) so `sw.js` changes and the browser installs the new version.

STEP 04

HTTPS, icons and the offline page

- **HTTPS** across the whole store. During development, `localhost` and `127.0.0.1` count as secure contexts.
- **Icons**: 192×192 and 512×512 PNGs at the paths in the manifest.
- `offline.html`: a short page with no prices or stock that only uses shell files. This is the one we tested, in Spanish; change the text and `lang`:

```
<!doctype html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Sin conexión</title>
<link rel="stylesheet" href="/app.css">
</head>
<body><h1>Sin conexión</h1><p>Revisa tu conexión y vuelve a intentarlo.</p></body>
</html>
```

STEP 05

Measure installs and launches

The rest of `pwa.js`. It uses the `gtag` from your GA4 tag:

```
window.addEventListener("appinstalled", () => {
  gtag("event", "pwa_install");
});

const standalone = window.matchMedia("(display-mode: standalone)").matches;
const fromIcon = new URLSearchParams(location.search).get("source") === "pwa";
if (standalone && fromIcon) {
  gtag("event", "pwa_launch");
}
```

- `pwa_install` only comes from Chromium browsers: Safari has no `appinstalled`.

- `pwa_launch` counts launches from the icon on any platform where the page opens with `display-mode: standalone`. We did not test it on an iPhone: confirm it in DebugView.
- In GA4, build a session segment in Explore of sessions that include `pwa_launch` and compare it with everyone else: purchases, revenue and how often they return.

WHEN YOU FINISH

Minimum PWA checklist

- [ ] The whole store is served over HTTPS.
- [ ] The manifest has `id`, `name`, `short_name`, a `start_url` with `?source=pwa`, `scope` and `display: "standalone"`.
- [ ] The 192 px and 512 px icons exist at the manifest's paths.
- [ ] Every page links the manifest and the server sends it as `application/manifest+json`.
- [ ] `/sw.js` lives at the root and is registered on every page.
- [ ] The cache holds only the shell and `offline.html`; HTML, cart and checkout go to the network.
- [ ] Offline, a page never visited shows `offline.html`.
- [ ] The cache name changes whenever the shell changes.
- [ ] GA4 receives `pwa_install` and `pwa_launch`, confirmed in DebugView.
- [ ] The native app decision is made with the `pwa_launch` segment, not before.