GUÍA PASO A PASO

Google no indexa tu página nueva: diagnóstico en orden

Siete pasos en el orden en que Google procesa una página: inspección de URLs, robots.txt, noindex, respuesta y canónica, sitemap y enlaces, un script en Node que revisa todo de una vez y cómo pedir la indexación. Con la checklist.

PARA	Cualquier sitio con una propiedad en Search Console y páginas nuevas que no aparecen
PROBADO CON	Node 22.22.3 · sitio de prueba local con diez casos · páginas públicas leídas el 19 de noviembre de 2026
ACTUALIZADA	19 de noviembre de 2026
DOCUMENTACIÓN	developers.google.com/search/docs/crawling-indexing
ARTÍCULO	ecomlabs.dev/g/indexar
AUTOR	Luis Morales · @luismoralesdev

Índice

Paso 01	Lee la inspección de URLs
Paso 02	robots.txt
Paso 03	noindex, respuesta y canónica
Paso 04	Sitemap y enlaces internos
Paso 05	Pide la indexación una vez
Anexo	El script
Al terminar	Checklist

PASO 01

Lee la inspección de URLs

Solo el propietario o un usuario con permisos completos puede solicitar la indexación. Pega la URL completa en la barra de Search Console y lee los campos en este orden; el primero que falle te dice a qué paso ir:

CAMPO	SI VES ESTO	VE A
Sitemaps · Página de referencia	Los dos vacíos	Paso 04
¿Se permite el rastreo?	No	Paso 02
Obtención de página	Error, redirección o 404	Paso 03
¿Se permite la indexación?	No, con noindex	Paso 03
Canónica declarada y seleccionada por Google	Distintas	Paso 03

"Descubierta: actualmente sin indexar": Google aplazó el rastreo (paso 04). "Rastreada: actualmente sin indexar": entró y no la indexó por ahora; Google dice que es habitual.

PASO 02

robots.txt

Si hay un grupo `User-agent: Googlebot`, Google usa ese y no el de `*`. Dentro del grupo gana la regla más larga que coincide; si empatan, la menos restrictiva. Las rutas distinguen mayúsculas, `*` es cualquier secuencia y `$` marca el final.

```
User-agent: *
Disallow: /admin
Disallow: /*?sort=
Allow: /drafts/public$
Disallow: /drafts/
Sitemap: https://tu-dominio.com/sitemap.xml
```

Aquí `/drafts/public` se puede rastrear (su `Allow` es más largo), `/drafts/launch` no, y ninguna URL con `?sort=`. Tras cambiar el archivo, pide un nuevo rastreo en el informe de robots.txt.

> **CUIDADO**
> Una página bloqueada en robots.txt no puede mostrarle su noindex a Google. Para sacarla de Google: desbloquéala y ponle noindex.

PASO 03

noindex, respuesta y canónica

```
URL=https://tu-dominio.com/blog/nuevo
curl -sI "$URL" | grep -i -E "^HTTP|^location|^x-robots-tag"
curl -s "$URL" | grep -o -i -E '<meta[^>]*name="(robots|googlebot)"[^>]*>'
curl -s "$URL" | grep -o -i -E '<link[^>]*rel="canonical"[^>]*>'
```

1. `curl -sI` pide solo cabeceras: quieres `200`, sin `location` y sin `X-Robots-Tag: noindex`.
2. El primer `grep` del HTML saca las etiquetas `meta` de robots o googlebot: ninguna debe decir `noindex` ni `none`.
3. El segundo saca la canónica: debe ser la propia URL, absoluta, con la misma barra final.

PASO 04

Sitemap y enlaces internos

Comprueba que la URL completa está en el sitemap (el script del anexo lo hace). Después, enlaza la página con un `<a href>` desde una ya indexada: portada, categoría, índice del blog o artículos relacionados. Un `onclick` o un `span` con `href` no son enlaces rastreables.

PASO 05

Pide la indexación una vez

1. Corre el script del anexo y pulsa **Probar URL publicada**. Si aún ve el problema, el arreglo no está en producción.
2. Pulsa **Solicitar indexación** una sola vez: hay un límite diario y repetirla no acelera el rastreo. Con muchas páginas, envía el sitemap con `lastmod`.
3. Si arreglaste un estado entero del informe, usa **Validar corrección**: según Google, suele tardar unas dos semanas.

Según Google, la indexación suele tardar "aproximadamente un día" y puede tardar "una o dos semanas", sin garantía.

ANEXO

El script

Revisa los pasos 02 a 04 de una vez: `node check-indexable.mjs https://tu-dominio.com/blog/nuevo`. Las líneas con `-` son fallos y salen con código 1; las que llevan `!`, avisos. No ejecuta JavaScript ni sabe qué canónica eligió Google: la inspección sigue siendo la referencia.

```
// Usage: node check-indexable.mjs https://example.com/new-page
// Reads what Google would read for one URL: robots.txt, the HTTP response, the
// robots meta tag, the canonical and the sitemaps. Node 18+, no dependencies.
const url = new URL(process.argv[2]);
const get = (u) => fetch(u, { redirect: "manual", headers: { "user-agent": "check-indexable" } });
const attr = (tag, name) => tag.match(new RegExp(`\\s${name}\\s*=\\s*["']([^"']*)["']`, "i"))?.[1] ?? "";
```

```
const fail = [], warn = [];

// robots.txt: the rules of the most specific group (googlebot, else *).
function rules(txt) {
  const groups = [];
  let open = false;
  for (const raw of txt.split(/\r?\n/)) {
    const m = raw.replace(/#.*/, "").match(/^\s*([a-z-]+)\s*:\s*(.*?)\s*$/i);
    if (!m) continue;
    const key = m[1].toLowerCase();
    if (key === "user-agent") {
      if (!open) groups.push({ agents: [], rules: [] });
      groups.at(-1).agents.push(m[2].toLowerCase());
      open = true;
      continue;
    }
    open = false;
    if ((key === "allow" || key === "disallow") && m[2] && groups.length) {
      groups.at(-1).rules.push({ allow: key === "allow", path: m[2] });
    }
  }
  const agent = groups.some((g) => g.agents.includes("googlebot")) ? "googlebot" : "*";
  return groups.filter((g) => g.agents.includes(agent)).flatMap((g) => g.rules);
}
const matches = (rule, path) => {
  const end = rule.endsWith("$");
  const body = (end ? rule.slice(0, -1) : rule).split("*").map((s) => s.replace(/[.+?^${}()|[\]\\]/g,
"\\$&"));
  return new RegExp(`^${body.join(".*")}${end ? "$" : ""}`).test(path);
};
// Longest matching rule wins; on a tie, allow wins (the least restrictive).
function blocked(txt, path) {
  let best;
  for (const r of rules(txt)) {
    if (!matches(r.path, path)) continue;
    if (!best || r.path.length > best.path.length || (r.path.length === best.path.length && r.allow)) best =
r;
  }
  return best ? !best.allow : false;
}

const robots = await get(new URL("/robots.txt", url));
const robotsTxt = robots.ok ? await robots.text() : "";
if (blocked(robotsTxt, url.pathname + url.search)) fail.push("robots.txt blocks it: Google cannot crawl it, so
it never sees a noindex");

const res = await get(url);
const html = res.status === 200 ? await res.text() : "";
if (res.status >= 300 && res.status < 400) fail.push(`redirects (${res.status}) to
${res.headers.get("location")}: only the target is indexed`);
else if (res.status !== 200) fail.push(`HTTP ${res.status}`);
```

```
const header = res.headers.get("x-robots-tag") ?? "";
if (/\b(noindex|none)\b/i.test(header)) fail.push(`X-Robots-Tag: ${header}`);
const metas = (html.match(/<meta\b[^>]*>/gi) ?? []).filter((t) => /^(robots|googlebot)$/i.test(attr(t,
"name")));
for (const t of metas) if (/\b(noindex|none)\b/i.test(attr(t, "content"))) fail.push(`meta ${attr(t, "name")}:
${attr(t, "content")}`);

const canonicalTag = (html.match(/<link\b[^>]*>/gi) ?? []).find((t) => /(^|\s)canonical(\s|$)/i.test(attr(t,
"rel")));
const canonical = canonicalTag ? new URL(attr(canonicalTag, "href"), url).href : "";
if (res.status === 200 && !canonical) warn.push("no rel=canonical: Google picks one");
if (canonical && canonical !== url.href) fail.push(`canonical points to ${canonical}`);

// Sitemaps from robots.txt, else /sitemap.xml; follows one level of sitemap index.
async function listed(sitemapUrl, depth = 0) {
  const r = await get(sitemapUrl);
  if (!r.ok) return false;
  const xml = await r.text();
  const locs = [...xml.matchAll(/<loc>\s*([^<]+?)\s*<\/loc>/g)].map((m) => m[1].replace(/&amp;/g, "&"));
  if (!/<sitemapindex/i.test(xml)) return locs.includes(url.href);
  if (depth === 0) for (const loc of locs) if (await listed(loc, 1)) return true;
  return false;
}
const sitemaps = [...robotsTxt.matchAll(/^\s*sitemap\s*:\s*(\S+)/gim)].map((m) => m[1]);
if (!sitemaps.length) sitemaps.push(new URL("/sitemap.xml", url).href);
let inSitemap = false;
for (const s of sitemaps) if (!inSitemap) inSitemap = await listed(s);
if (!inSitemap) warn.push("not in the sitemap");

console.log(`${fail.length ? "FAIL" : "OK  "} ${url.href}`);
for (const f of fail) console.log(`     - ${f}`);
for (const w of warn) console.log(`     ! ${w}`);
process.exit(fail.length ? 1 : 0);
```

AL TERMINAR

Checklist

- [ ] La inspección no muestra bloqueos de rastreo ni de indexación.
- [ ] Ninguna regla de robots.txt cubre la ruta y no hay noindex en la meta ni en `X-Robots-Tag`.
- [ ] Responde 200 sin redirección, la canónica es la propia URL y el script pasa.
- [ ] Está en el sitemap y una página indexada la enlaza.
- [ ] Pediste la indexación una vez y revisas de nuevo en una semana.