EcomLabs

GUÍA PASO A PASO

hreflang en Next.js para un sitio bilingüe

La ayuda que genera las etiquetas hreflang recíprocas en Next.js, el sitemap con las versiones por idioma, qué código usar (es o es-MX) y un script en Node que valida el HTML de tus páginas. Con la checklist.

PARA	Sitios en Next.js (App Router) con una versión en inglés y otra en español
PROBADO CON	Node 22.22.3 · resolvers de metadata de Next.js 15 y tsc --strict · HTML público guardado el 12 de noviembre de 2026
ACTUALIZADA	12 de noviembre de 2026
DOCUMENTACIÓN	developers.google.com/search/docs/specialty/international/localized-versions
ARTÍCULO	ecomlabs.dev/g/hreflang
AUTOR	Luis Morales · @luismoralesdev

Índice

Antes de empezar	Las reglas de Google
Paso 01	La ayuda y la página
Paso 02	El sitemap
Paso 03	es o es-MX
Paso 04	El script de validación
Al terminar	Checklist

ANTES DE EMPEZAR

Las reglas de Google

Según Google Search Central, estas reglas son las que hacen que hreflang funcione:

- Cada versión se lista a sí misma y a todas las demás. Si dos páginas no se apuntan una a la otra, las etiquetas se ignoran.
- Las URLs son completas, con `https://`.
- El código es el idioma (ISO 639-1) y, opcional, la región (ISO 3166-1 alfa-2): `es` y `es-MX` sí; una región sola y `es-419` no.
- `x-default` es el valor para cuando ninguna otra versión coincide.
- HTML, cabecera HTTP y sitemap son equivalentes; usar los tres a la vez no aporta nada en Search.

PASO 01

La ayuda y la página

Una sola función arma la lista, así las dos versiones usan la misma y la reciprocidad queda garantizada. Validada con `tsc --strict` contra los tipos de Next.js 15:

```
// lib/hreflang.ts
import type { Metadata } from "next";

export type Locale = "en" | "es";

// English lives at the root and Spanish under /es. Each version's whole path is
// passed in, because the slug can differ between languages.
export function pageAlternates(paths: Record<Locale, string>, current: Locale): Metadata["alternates"] {
  return {
    canonical: paths[current],
    languages: { en: paths.en, es: paths.es, "x-default": paths.en },
  };
}

// app/blog/[slug]/page.tsx (the English page; the Spanish page passes "es")
export async function generateMetadata({ params }: Props): Promise<Metadata> {
  const { slug } = await params;
  const post = await getPost(slug); // { slug: { en, es } } from your data layer
  return {
    title: post.title,
    alternates: pageAlternates(
      { en: `/blog/${post.slug.en}`, es: `/es/blog/${post.slug.es}` },
      "en",
    ),
  };
}
```

```
// app/layout.tsx: lets the relative paths above resolve to full URLs
export const metadata: Metadata = { metadataBase: new URL("https://example.com") };
```

Cambia `https://example.com` por tu dominio. Sin `metadataBase`, una ruta relativa rompe el build. Las páginas estáticas (portada, servicios) usan la misma ayuda con sus dos rutas.

PASO 02

El sitemap

Desde Next.js 14.2.0, cada entrada del sitemap acepta `alternates.languages`. El ejemplo de la documentación no lista la propia versión; Google la pide, así que aquí sí está:

```
// app/sitemap.ts
import type { MetadataRoute } from "next";

const BASE = "https://example.com";

// One entry per page, with the path of each language version.
const pages = [
  { en: "/", es: "/es" },
  { en: "/services", es: "/es/servicios" },
];

export default function sitemap(): MetadataRoute.Sitemap {
  return pages.flatMap((p) => {
    const languages = { en: BASE + p.en, es: BASE + p.es, "x-default": BASE + p.en };
    // Both versions get a <url>, and both carry the same full list.
    return [
      { url: languages.en, alternates: { languages } },
      { url: languages.es, alternates: { languages } },
    ];
  });
}
```

PASO 03

es o es-MX

CÓDIGO	¿SE ADMITE?	CUÁNDO
`es`	Sí	Todo el español, sin región
`es-MX`	Sí	Una versión específica para México
`es-419`	No	No es ISO 639-1 más ISO 3166-1
`MX`	No	La región no puede ir sola
`x-default`	Sí (reservado)	Cuando ninguna otra versión coincide

Si tienes una versión para México y otra para el resto, lista `es-MX` y `es`, como el ejemplo de Google con `en-GB`, `en-US` y `en`. Los tipos de Next.js aceptan `es-419` sin quejarse: el error aparece en Google, no en tu editor.

PASO 04

El script de validación

Guarda `check-hreflang.mjs`. Lee HTML guardado, sin dependencias:

```
// Usage: node check-hreflang.mjs page-en.html page-es.html [more.html]
// Each file is the saved HTML of one URL. The pages you pass form the set
// that gets checked for return links.
import { readFileSync } from "node:fs";

const CODE = /^(x-default|[a-z]{2}(-[A-Z]{2})?)$/; // ISO 639-1, optional ISO 3166-1 alpha-2
const attr = (tag, name) => tag.match(new RegExp(`\\s${name}\\s*=\\s*"([^"]*)"`, "i"))?.[1];
const norm = (u) => (u ?? "").replace(/\/$/, "");

function parse(file) {
  const tags = readFileSync(file, "utf8").match(/<link\b[^>]*>/gi) ?? [];
  const canonical = tags.filter((t) => attr(t, "rel") === "canonical").map((t) => attr(t, "href"))[0];
  const alternates = tags
    .filter((t) => attr(t, "rel") === "alternate" && attr(t, "hreflang"))
    .map((t) => ({ lang: attr(t, "hreflang"), href: attr(t, "href") }));
  return { file, canonical, alternates };
}

const pages = process.argv.slice(2).map(parse);
const byUrl = new Map(pages.map((p) => [norm(p.canonical), p]));
let failed = false;

for (const p of pages) {
  const errors = [];
  const warnings = [];
  if (!p.canonical) errors.push("no rel=canonical (used here as the page's own URL)");
```

```
  if (p.alternates.length === 0) errors.push("no hreflang tags");
  for (const { lang, href } of p.alternates) {
    if (!CODE.test(lang)) errors.push(`unsupported code "${lang}"`);
    if (!/^https?:\/\//.test(href ?? "")) errors.push(`href is not fully qualified: "${href}"`);
  }
  if (p.alternates.length && !p.alternates.some((a) => a.lang === "x-default")) warnings.push("no x-default");
  if (p.alternates.length && !p.alternates.some((a) => norm(a.href) === norm(p.canonical))) {
    errors.push("the page does not list itself");
  }
  for (const { lang, href } of p.alternates) {
    const other = byUrl.get(norm(href));
    if (other && other !== p && !other.alternates.some((a) => norm(a.href) === norm(p.canonical))) {
      errors.push(`${lang} -> ${href}: that page does not link back`);
    }
  }
  const lines = [...errors.map((e) => `\n     - ${e}`), ...warnings.map((w) => `\n     ! ${w}`)];
  console.log(`${errors.length ? "FAIL" : "OK  "} ${p.file}${lines.join("")}`);
  if (errors.length) failed = true;
}
process.exit(failed ? 1 : 0);
```

Guarda el HTML de cada par con `curl -sS -o home-en.html https://tu-dominio.com/` y `curl -sS -o home-es.html https://tu-dominio.com/es`, y pasa los dos archivos juntos: `node check-hreflang.mjs home-en.html home-es.html`. Sale con código 1 si hay errores. Un `x-default` ausente avisa, no falla. Revisa un par de cada tipo de página: portada, producto, artículo.

> **CUIDADO**
>
> El script lee el HTML que le pasas, no el sitemap. Revisa el sitemap aparte: cada `<url>` debe llevar un `xhtml:link` por cada versión, incluida la suya. Lo probamos con HTML público guardado de dos pares de páginas de un sitio real (pasan) y con una página rota escrita a mano (falla con cuatro hallazgos).

AL TERMINAR

Checklist

- [ ] Cada página lista todas sus versiones y a sí misma, con URLs completas.
- [ ] La lista sale de una sola función, y las dos versiones usan la misma.
- [ ] Cada idioma tiene su propio `canonical`.
- [ ] `x-default` apunta a la versión por defecto.
- [ ] Usas `es`, o `es-MX` solo si hay una versión para México.
- [ ] Los slugs de cada idioma se guardan, no se calculan con un prefijo.
- [ ] `metadataBase` está en el layout raíz y el sitemap lleva la lista completa.
- [ ] El script pasa en una página de cada tipo.