Engineer

Next.js

N marco de React para crear aplicaciones web

Qué es

Next.js es un marco de trabajo que envuelve React, la biblioteca para construir interfaces de usuario con componentes, y aporta lo que React deja a cada aplicación: enrutamiento, renderizado en el servidor, obtención de datos y una compilación ya configurada. Tanto el paquete como su orden se llaman `next`, como en `next dev` y `next build`.

Se publicó por primera vez en 2016 para, según sus autores, «ofrecer una forma sencilla de renderizar aplicaciones React en el servidor». El diseño inicial se basaba en unos pocos principios: ninguna configuración inicial; el sistema de archivos como API, de modo que un archivo en `pages/` se convertía en una ruta; renderizado en servidor y división del código automáticos; y obtención de datos a cargo del desarrollador. En 2023, las notas de versión ya reconocían a más de 2.600 desarrolladores junto al equipo principal de Vercel.

Next.js 13 añadió el 25 de octubre de 2022 un directorio `app` en beta, basado en React Server Components, con diseños anidados y transmisión progresiva. Next.js 13.4 declaró estable App Router el 4 de mayo de 2023; desde entonces, la documentación describe dos enrutadores. Pages Router es «el enrutador original, aún mantenido y en mejora». Difieren incluso en la versión de React que ejecutan: App Router incorpora versiones canary de React, mientras Pages Router utiliza la versión que indique el `package.json` del proyecto. En App Router, las páginas y los diseños se renderizan en el servidor, salvo que un archivo elija ejecutarse en el navegador con la directiva `"use client"`.

Next.js 16 llegó el 21 de octubre de 2025. Convirtió Turbopack en el empaquetador predeterminado, cambió el nombre de `middleware.ts` a `proxy.ts` e introdujo Cache Components: un modelo donde la directiva `"use cache"` marca lo que puede almacenarse en caché y todo lo demás se renderiza al recibir la

solicitud. Next.js 16.4, del 6 de octubre de 2026, recomienda Cache Components para todas las aplicaciones, activa ese modelo en proyectos nuevos creados con `create-next-app` y anuncia que «será la opción predeterminada en Next.js 17».

Utilice el concepto en una decisión

Aplique «¿Qué es Next.js?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Interfaces que cargan rápido, funcionan en el teléfono y se comportan igual en los navegadores que utilizan sus clientes.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Dirección del desarrollo integral de mapas GIS — cartografía integral, de la base de datos al navegador

Dirección del desarrollo integral de mapas GIS Póngase en contacto →

Por qué importa

La decisión que resuelve Next.js es si dejar que un marco de trabajo controle el enrutamiento, el renderizado y el servidor de una aplicación React o montar esas piezas por su cuenta y mantener el control sobre esas decisiones.

Ofrece convenciones en lugar de configuración. Las rutas nacen de archivos, los empaquetadores y compiladores se configuran automáticamente y cada ruta puede construirse de antemano, renderizarse en el servidor por solicitud o renderizarse en el navegador. Los Server Components mantienen el acceso a los datos y los secretos en el servidor y envían menos JavaScript al navegador. Para una aplicación que combina páginas públicas, una zona con sesión iniciada y su propio backend, esa elección por ruta es el principal atractivo.

El primer coste es un servidor con Node.js 20.9 o posterior para Next.js 16. El ajuste `output: 'export'` escribe un archivo HTML por ruta para cualquier servidor web estático; la guía lo presenta como una forma de empezar con contenido estático y añadir funciones de servidor después. Sin embargo, la lista de lo que exige un servidor Node.js abarca buena parte de lo que distingue a Next.js: Proxy, Server Actions, cookies, reescrituras, redirecciones, cabeceras, Incremental Static Regeneration y optimización de imágenes con el cargador predeterminado.

El segundo coste es el ritmo. En un año hubo cuatro versiones funcionales después de 16.0, y el modelo de caché está a punto de volver a cambiar su opción predeterminada. Las notas de 16.4 dicen que el modelo Cache Components

sustituye «los comportamientos implícitos de caché de versiones anteriores de App Router»: el código escrito para Next.js 13 a 15 utiliza otras reglas de caché. Los tutoriales de Internet también se reparten entre ambos enrutadores.

El tercer coste es que actualizar también es trabajo de seguridad. El 3 de diciembre de 2025, el proyecto publicó un aviso sobre CVE-2025-66478, una vulnerabilidad crítica del protocolo React Server Components, e indicó que se actualizara inmediatamente a versiones corregidas. El blog publicó otros tres artículos de seguridad entre el 30 de septiembre y el 14 de octubre de 2026. Adoptar Next.js obliga al equipo a seguir sus versiones de cerca. Un equipo que publica documentos iguales para todos los visitantes paga por un modelo de servidor que apenas utiliza.

Cómo se aplica en la práctica

En el trabajo de la fundadora en NextMariner, la interfaz Next.js 16 renderiza cada parte de la forma que le corresponde: es la elección por ruta llevada a la práctica. Las páginas públicas y de marketing se generan estáticamente; las partes realmente interactivas siguen renderizándose en el cliente. En el panel autenticado y los grandes listados de directorio, una factoría, `createPrefetchedServerPage`, obtiene los datos de la página en el servidor y los entrega al cliente ya cargados. Las páginas pasan a la factoría de una en una; el caso reconoce que el resto de la aplicación autenticada sigue renderizándose en el cliente, pendiente de migración: «una migración con un mecanismo que funciona y la siguiente página por migrar claramente identificada, en lugar de una conversión completa». App Router se diseñó para ese tipo de traslado; las notas de 13.4 describen su adopción incremental.

Un conjunto de pruebas por capas da una capa propia a la interfaz Next.js: 543 especificaciones Vitest y Playwright, incluidos 48 archivos de extremo a extremo y 30 especificaciones de navegador, junto a capas separadas para la API Go y la base de datos. Cada capa se comprueba donde reside su comportamiento, en lugar de probarlo todo desde un navegador. Importa en un marco cuyo árbol de componentes se ejecuta en parte en el servidor y en parte en el navegador.

Cuando la fundadora, trabajando en NextMariner, reforzó la seguridad de una aplicación, las protecciones orientadas al navegador se incorporaron a `proxy.ts`, el middleware renombrado. Establece una Content-Security-Policy con un nonce por solicitud y `strict-dynamic`, además de HSTS y cookies SameSite. La autorización reside en otro lugar: los derechos de acceso, que abarcan nivel, rol,

organización y alcance por buque, vuelven a comprobarse en el servidor en cada solicitud; los controles de la interfaz solo cuentan como experiencia de usuario. Esa división coincide con la referencia de Proxy, que considera la función un último recurso e indica que se verifique la autorización «dentro de cada Server Function, en lugar de depender solo de Proxy».

Dónde surgen los problemas

Tras añadir una redirección de acceso, la página de inicio de sesión carga sin estilos. Sin un `matcher`, Proxy se ejecuta en cada solicitud, incluidos los archivos compilados de `_next/static`, las imágenes optimizadas de `_next/image` y todo lo que haya en `public/`. Una redirección destinada a páginas alcanza también el CSS, JavaScript y las imágenes que necesita la página de acceso. Asigne a Proxy un patrón que omita esas rutas:

```
// proxy.ts
export const config = {
    matcher: ["/((?!_next/static|_next/image|favicon.ico).*)"],
};
```

Los archivos de `public/` no comparten prefijo: añada también sus rutas.

Una acción de formulario funciona sin sesión en una ruta que parece protegida. Las Server Functions no tienen rutas propias: llegan como solicitudes POST a la página donde se utilizan. Si un patrón omite una ruta, también omite sus funciones. La referencia de Proxy advierte de que cambiar el patrón o mover una función a otra ruta al refactorizar «puede eliminar silenciosamente la cobertura de Proxy». Compruebe la sesión y los permisos del usuario dentro de cada Server Function; trate Proxy como una capa exterior, nunca como la única.

Un usuario guarda un cambio y la página sigue mostrando el valor anterior. En Next.js 16, `revalidateTag(tag, 'max')` aplica a las entradas etiquetadas el comportamiento stale-while-revalidate: sigue sirviéndose la copia en caché mientras se obtiene otra nueva. Es adecuado para contenido que tolera consistencia eventual, pero sorprende a quien acaba de pulsar Guardar. Llame a `updateTag(tag)` desde la Server Action que realizó el cambio: caduca la entrada y lee datos actualizados dentro de la misma solicitud. La antigua llamada `revalidateTag(tag)` con un solo argumento está obsoleta y ahora provoca un error de TypeScript.

Termina el codemod de actualización, pero la compilación sigue fallando en `params` o `cookies()`. Next.js 15 convirtió las API de solicitud en asíncronas, con compatibilidad síncrona temporal; Next.js 16 la elimina. El codemod `upgrade` no ejecuta todas las migraciones: ejecute también `npx @next/codemod@canary next-async-request-api .` y después `npx next typegen` para generar los auxiliares `PageProps`, `LayoutProps` y `RouteContext`. La misma actualización elimina `next lint` y hace que `next build` deje de ejecutar el análisis estático; añada ESLint o Biome como paso independiente.

Lecturas adicionales

- Documentación de Next.js — la documentación oficial, con un selector entre las versiones App Router y Pages Router de cada página.
- Next.js 16 — la versión que introdujo Cache Components, `proxy.ts` y Turbopack como empaquetador predeterminado, con sus cambios incompatibles.
- Cómo actualizar a la versión 16 — los codemods, los requisitos de versión y todas las API eliminadas o renombradas.
- proxy.js — los patrones de selección, la posición de Proxy en el orden de las solicitudes y por qué se renombró middleware.
- Next.js 16.4 — los argumentos a favor de Cache Components y el plan para convertirlo en opción predeterminada en Next.js 17.
- Cómo crear una exportación estática — qué funciones necesitan servidor y qué escribe `output: 'export'`.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Next.js?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/nextjs/