DEZVOLTARE WEB

Ghid Next.js Performanță 2026

SSG, SSR, Core Web Vitals, caching și deploy VPS — cum să alegi stack-ul potrivit pentru afacerea ta

~30 pagini · 22 min citire · PDF A4

Next.js este framework-ul preferat pentru site-uri rapide, optimizate SEO și experiențe moderne. Dar nu e soluția magică pentru orice proiect — un blog simplu sau un magazin cu sute de SKU-uri poate fi mai ieftin pe WordPress. Acest ghid te ajută să înțelegi când Next.js merită investiția, cum obții scoruri PageSpeed excelente, cum deploy-ezi pe VPS și când e momentul să angajezi o agenție.

valcode.dev/resurse
Descarcă gratuit · Consultanță la cerere

Cuprins

1 De ce Next.js în 2026

Next.js 14/15 aduce App Router stabil, Server Components, streaming SSR și optimizări de imagini out-of-the-box. Pentru site-uri de prezentare, landing pages, SaaS și platforme cu conținut dinamic, oferă viteză și SEO pe care WordPress cu 25 de plugin-uri nu le atinge fără efort disproporționat.

- React ecosystem: componente reutilizabile, hiring mai ușor, librării moderne
- Rendering flexibil: static, server, incremental — alegi per pagină
- Image Optimization built-in: WebP, lazy load, sizes responsive automat
- SEO nativ: metadata API, sitemap, robots.txt, Open Graph fără plugin-uri
- API Routes / Server Actions: backend integrat fără server PHP separat
- Deploy flexibil: Vercel (zero config), VPS (control cost), Docker (portabilitate)

2 SSG vs SSR vs ISR — ce alegi per pagină

Modul de rendering decide viteza, costul serverului și freshness-ul conținutului. Alegerea greșită — SSR pentru tot — generează costuri Vercel mari și TTFB lent fără beneficiu real.

1. Inventariază paginile: care se schimbă zilnic vs lunar vs niciodată.
2. Homepage, servicii, contact → SSG sau ISR cu revalidate 86400.
3. Dashboard user, coș personalizat → SSR sau CSR cu autentificare.
4. Blog articole → ISR revalidate 3600 sau on-demand revalidation la publish.

- SSG (Static Site Generation): pagini build-uite la deploy — homepage, about, servicii. Cel mai rapid, CDN-friendly
- SSR (Server-Side Rendering): HTML generat la request — pagini personalizate, date live. Cost CPU per vizită
- ISR (Incremental Static Regeneration): static cu revalidare (ex. la 3600s) — blog, cataloage medii
- CSR (Client-Side Rendering): evită pentru conținut SEO-critical — Google indexează mai greu
- App Router: export const dynamic = 'force-static' sau revalidate = 3600 per layout/page
- Regula practică: 80% static/ISR, 20% SSR doar unde e necesar

3 Core Web Vitals — ținte și măsurare

LCP (Largest Contentful Paint), INP (Interaction to Next Paint) și CLS (Cumulative Layout Shift) sunt metrici Google folosite la ranking. Next.js oferă unelte excelente, dar implementarea greșită — fonturi neoptimizate, imagini mari, JS excessiv — produce scoruri slabe la fel ca orice alt framework.

- LCP < 2,5s: optimizează hero image, folosește next/image cu priority pe above-the-fold
- INP < 200ms: reduce JavaScript client-side, preferă Server Components
- CLS < 0,1: rezervă spațiu pentru imagini (width/height), evită injectare dinamică above-fold
- Măsoară cu PageSpeed Insights (lab) + Search Console CWV (field data real)
- Vercel Speed Insights sau web-vitals library pentru monitoring continuu
- Testează pe mobil 4G throttled — 70% din trafic RO e mobil

4 Optimizare imagini în Next.js

Imaginile reprezintă 50–70% din greutatea paginii tipice. next/image face resize automat, servește WebP/AVIF, lazy load și previne CLS — dar trebuie folosit corect, nu ca pe un img HTML obișnuit.

1. Audit: grep -r '<img' src/ — zero rezultate e ținta.
2. Configurează remotePatterns pentru domeniul CMS/CDN.
3. Hero: Image cu priority, width={1200}, quality={80}, placeholder="blur".
4. Re-testează LCP în PageSpeed după fiecare batch de optimizări.

- Înlocuiește toate <img> cu <Image> din next/image
- Prop priority={true} doar pe hero/LCP — altfel lazy load implicit
- Specifică width + height sau fill cu container sized — previne CLS
- sizes prop corect: (max-width: 768px) 100vw, 50vw — evită descărcarea imaginii 4K pe mobil
- remotePatterns în next.config.js pentru imagini externe (CDN, CMS)
- Calitate 75–80 e sweet spot; 100 e rar necesar și încetinește

5 Caching — straturi și strategie

Caching-ul transformă un site „rapid” într-unul instant. Next.js cache-uiește la multiple niveluri: build static, Data Cache, Full Route Cache, CDN edge. Înțelegerea straturilor te ajută să eviți conținut stale sau, invers, server overload.

- Static pages: cache indefinite la CDN până la next build/deploy
- fetch() cu cache: 'force-cache' (default static) vs cache: 'no-store' (dinamic)
- revalidate: 3600 — ISR, regenerează pagina max o dată/oră
- Cache-Control headers via next.config.js headers() pentru assets statice
- CDN (Cloudflare/Vercel Edge): cache HTML static, bypass pentru cookies autentificare
- On-demand revalidation: revalidatePath('/blog') la publish din CMS webhook

6 SEO în Next.js — metadata, sitemap, structură

Next.js App Router are API de metadata declarativă — fără react-helmet, fără plugin SEO. Totuși, SEO nu e automat: trebuie title/description unice, structură URL logică, schema markup și internal linking planificat.

1. Definire metadata template în layout.tsx root (title.default, title.template).
2. Creează sitemap.js cu toate rutele statice + dinamice din CMS.
3. Submit sitemap în Google Search Console.
4. Verifică rich results cu Google Rich Results Test.

- export const metadata = { title, description, openGraph, alternates } per page/layout
- generateMetadata() pentru pagini dinamice (blog/[slug]) cu date din CMS
- app/sitemap.js — generează sitemap.xml dinamic din conținut
- app/robots.js — allow/disallow, link sitemap
- JSON-LD via <script type="application/ld+json"> — LocalBusiness, Article, FAQ
- URL-uri în română fără diacritice, structură flat: /servicii/seo-bucuresti

7 Deploy pe VPS — producție Next.js

Vercel e cel mai simplu, dar costul crește cu traficul. Deploy pe VPS (Hetzner + nginx + PM2) costă fix 5–20€/lună indiferent de vizite — ideal pentru site-uri cu trafic mare sau control strict al datelor.

1. next.config.js: output: 'standalone'.
2. Pe VPS: git clone, npm ci, npm run build.
3. PM2 start + nginx config + certbot.
4. Testează: curl -I https://domeniu.ro → 200, verifică header x-nextjs-cache.

- Build: npm run build → output standalone (next.config.js: output: 'standalone')
- Transfer: rsync sau git pull pe VPS în /var/www/app
- PM2: pm2 start .next/standalone/server.js --name nextapp
- nginx reverse proxy: proxy_pass http://127.0.0.1:3000 cu headere X-Forwarded-For/Proto
- SSL via Certbot, UFW activ, fail2ban configurat
- CI/CD opțional: GitHub Actions → SSH deploy script la push pe main

8 Next.js vs WordPress — cum decizi

Nu există câștigător universal. WordPress câștigă la conținut editabil de non-tehnicieni, ecosistem WooCommerce matur și cost inițial mic. Next.js câștigă la performanță, design custom, scalabilitate frontend și experiențe interactive.

- Alege WordPress dacă: echipa editează zilnic conținut, magazin WooCommerce complex, buget limitat initial
- Alege Next.js dacă: design premium custom, aplicație web (SaaS), viteză CWV critică, integrare API complexă
- Headless WordPress + Next.js frontend: combinație populară — CMS familiar, frontend rapid
- Migrare WP → Next.js: cost 3.000–15.000€+ în funcție de pagini, funcționalități, conținut
- Migrare Next.js → WP: rară, dar posibilă dacă clientul vrea autonomie editare
- Timeline: WordPress 2–4 săptămâni lansare; Next.js custom 4–12 săptămâni

9 Comparație costuri — 12 luni TCO

Costul real al unui site include dezvoltare, hosting, mentenanță, licențe și oportunitate pierdută din viteză/conversie slabă. Comparația de mai jos reflectă medii pentru SMB din România în 2026.

- WordPress site prezentare: 1.200€ dev + 15€/lună hosting + 50€/lună mentenanță = ~1.980€/an
- WordPress WooCommerce: 3.500€ dev + 40€/lună hosting + 80€/lună mentenanță = ~4.940€/an
- Next.js site prezentare (Vercel): 4.000€ dev + 0–20€/lună Vercel + 30€/lună mentenanță = ~4.360€/an
- Next.js pe VPS: 4.000€ dev + 10€/lună VPS + 50€/lună mentenanță = ~4.720€/an
- Next.js câștigă ROI la: trafic mare (Vercel Pro 20$/user), CWV ca avantaj competitiv, redesign premium
- WordPress câștigă ROI la: buget sub 2.000€, editare frecventă non-tehnic, magazin standard

10 Când să angajezi o agenție web

DIY cu template Next.js sau WordPress funcționează pentru MVP-uri. Când afacerea depinde de site pentru lead-uri, vânzări sau credibilitate, agenția aduce arhitectură, QA, SEO tehnic și suport post-lansare care justifică investiția.

- Angajează agenție când: buget dev > 3.000€, deadline fix, integrare ERP/CRM, conformitate GDPR complexă
- Semne că ai nevoie de ajutor: PageSpeed sub 50 mobil, site down lunar, zero leads din organic după 6 luni
- Ce să ceri: portofoliu cu CWV verificabile, contract mentenanță, SLA uptime, proprietate cod/domeniu
- Red flags: preț „site complet 500€”, fără staging, fără contract, hosting obligatoriu propriu scump
- Agenție locală RO vs remote: locală = comunicare, facturare RO, înțelegere piață; remote = specializare nișă
- Model recomandat: discovery call → propunere scop + timeline → milestone payments → 30 zile suport post-lansare

Întrebări frecvente

Next.js e gratuit? Ce costuri ascunse există?

Framework-ul e open-source și gratuit. Costurile vin din hosting (Vercel free tier limitat, VPS 5–20€/lună), dezvoltare (4.000–15.000€+ proiect custom), CMS headless opțional (Sanity, Contentful — free tier generos) și mentenanță lunară. Vercel Pro devine necesar la trafic > 100GB bandwidth/lună sau funcții serverless intensive.

Pot edita conținutul singur într-un site Next.js?

Da, cu un CMS headless: Sanity, Contentful, Strapi sau WordPress headless. Editorul e la fel de prietenos ca wp-admin pentru texte și imagini. Fără CMS, orice modificare de conținut necesită developer — de aceea CMS headless e aproape standard pe proiectele Next.js pentru clienți non-tehnici.

Cât durează dezvoltarea unui site Next.js pentru o firmă din România?

Site de prezentare (5–10 pagini, design custom): 4–8 săptămâni. Platformă cu funcționalități custom (calculatoare, dashboard, integrări API): 8–16 săptămâni. Depinde de complexitate design, conținut pregătit de client și feedback loop. La valcode.dev oferim timeline fix după discovery call de 30 minute.

Implementare completă

Site + SEO + Google Ads + Mentenanță
Audit gratuit · Demo site gratuit · Răspuns în 24h

valcode.dev/contact

+40 750 205 515 · contact@valcode.dev