DÉVELOPPEMENT FRONTEND

Next.js : construire des applications web modernes

Aller au-delà de React avec Next.js : rendu serveur, App Router, optimisation des performances, SEO et déploiement production.

DURÉE	TARIF HT	NIVEAU	LANGUE	GROUPE	FORMAT
5j (35h)	**1390.00 €**	**Avancé**	**Français**	**2-12**	**Formation**

1 PUBLIC VISÉ

- Développeurs React maîtrisant les hooks et les composants fonctionnels souhaitant adopter un framework full-stack professionnel.
- Développeurs front-end voulant prendre en charge le rendu serveur, le SEO et les performances dans leurs projets React.
- Personnes ayant suivi la formation React : développer des interfaces modernes (FRT-REA) ou disposant d'une expérience React équivalente en production.

2 PRÉREQUIS

- Bonne maîtrise de React : composants fonctionnels, hooks (useState, useEffect, useContext), React Router.
- La formation React : développer des interfaces modernes est recommandée.
- Notions de base en HTTP et REST. Savoir utiliser le terminal et VS Code.

3 OBJECTIFS PÉDAGOGIQUES

- Comprendre l'architecture de Next.js et la différence entre App Router et Pages Router
- Maîtriser les stratégies de rendu : SSR, SSG, ISR et rendu hybride par page
- Créer des Server Components et des Client Components et savoir choisir entre les deux
- Mettre en place des layouts imbriqués, des états de chargement et des error boundaries
- Gérer des routes API et des Server Actions pour les mutations de données
- Implémenter une authentification complète avec NextAuth.js
- Optimiser les performances : images, polices, métadonnées et Core Web Vitals
- Déployer une application Next.js en production sur Vercel avec variables d'environnement

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) – Fondamentaux Next.js et App Router

Module 1 – Découverte de Next.js (2h)

Pourquoi Next.js ? (45min)

- Le problème que Next.js résout : SEO, performances et complexité d'architecture
- React seul vs Next.js : ce que le framework apporte concrètement
- Next.js dans l'écosystème : Vercel, Remix, Nuxt — positionnement
- App Router vs Pages Router : l'évolution majeure de Next.js 13+
- Cas d'usage : e-commerce, blog, SaaS, dashboard — quelle stratégie pour quel projet

Mise en place de l'environnement (45min)

- Créer un projet Next.js avec create-next-app
- Structure du projet App Router : app/, public/, next.config.js
- Les conventions de fichiers : page.tsx, layout.tsx, loading.tsx, error.tsx
- Lancer le serveur de développement et comprendre le hot reload

Routing basé sur le système de fichiers (30min)

- Créer des pages avec le dossier app/
- Routes statiques, dynamiques et catch-all
- Le composant Link et la navigation côté client
- Cas pratique : créer une structure de routes pour un blog

Module 2 – Layouts et organisation de l'interface (2h)

Layouts imbriqués (1h)

- Le fichier layout.tsx : rôle et fonctionnement
- Layouts imbriqués : layout global, layout de section, layout de page
- Partager un état entre layouts sans re-render
- Route Groups : organiser les routes sans affecter l'URL
- Cas pratique : layout dashboard avec sidebar et header persistants

Loading, Error et Not Found (1h)

- loading.tsx : afficher un squelette pendant le chargement des données
- error.tsx : capturer les erreurs et afficher un fallback
- not-found.tsx : page 404 personnalisée par segment
- Suspense boundaries : contrôler la granularité du chargement
- Cas pratique : page produit avec loading skeleton et gestion d'erreur

Module 3 – Server Components vs Client Components (3h)

Comprendre le modèle de rendu (1h30)

- Le nouveau modèle mental : tout est Server Component par défaut
- Ce que les Server Components permettent : accès BDD, secrets, réduction du bundle JS
- Quand utiliser 'use client' : état, événements, hooks, API navigateur
- La frontière client/serveur : comment composer les deux types de composants
- Les erreurs classiques : passer des fonctions non-sérialisables, importer du code serveur côté client

Fetch de données côté serveur (1h30)

Fetch natif étendu par Next.js : revalidation, cache, no-store

- Fetch en parallèle avec Promise.all dans un Server Component
- Waterfall de données : comment l'éviter avec Suspense
- Accéder directement à la base de données depuis un Server Component
- Cas pratique : page de liste et de détail avec données serveur

Jour 2 (7h) – Stratégies de rendu et mutations de données

Module 4 – SSR, SSG et ISR (3h)

Static Site Generation (1h)

- generateStaticParams : pré-générer les routes dynamiques au build
- Quand choisir SSG : blog, documentation, landing pages
- Les limites du SSG : données temps réel, authentification
- Cas pratique : blog statique avec pages générées au build

Server Side Rendering (1h)

- SSR à la demande : quand et pourquoi
- Dynamic rendering : force-dynamic, cookies(), headers()
- Streaming SSR avec Suspense : envoyer le HTML progressivement
- Cas pratique : page de profil utilisateur rendue côté serveur

Incremental Static Regeneration (1h)

- ISR : régénérer une page statique à intervalle défini
- revalidate : configurer la durée de vie du cache
- On-demand revalidation : revalidatePath et revalidateTag
- Cas pratique : page de catalogue produit mise à jour toutes les 60 secondes

Module 5 – Routes API et Server Actions (4h)

Routes API avec App Router (1h30)

- Créer une route API avec route.ts : GET, POST, PUT, DELETE
- Lire les paramètres de requête et le body
- Retourner une réponse JSON avec NextResponse
- Middlewares : protéger des routes, rediriger, modifier les headers
- Cas pratique : API REST pour une ressource CRUD

Server Actions : mutations sans API (1h30)

- Qu'est-ce qu'une Server Action : définition et avantages
- Créer une Server Action avec 'use server'
- Appeler une Server Action depuis un formulaire ou un composant client
- Revalider le cache après une mutation : revalidatePath
- Gestion des erreurs et feedback utilisateur
- Cas pratique : formulaire de création avec Server Action et feedback immédiat

Optimistic Updates et useFormStatus (1h)

- useFormStatus : afficher l'état pending d'un formulaire
- useOptimistic : mettre à jour l'UI avant la confirmation serveur
- Cas pratique : liste de tâches avec suppression optimiste

Jour 3 (7h) – Authentification, base de données et optimisation

Module 6 – Authentification avec NextAuth.js (3h)

Mise en place de NextAuth.js (1h30)

- Installer et configurer NextAuth.js v5 avec App Router
- Providers : credentials, Google, GitHub
- Le fichier auth.ts : configuration centralisée
- Session strategy : JWT vs database sessions
- Cas pratique : authentification par email/mot de passe

Protéger les routes et accéder à la session (1h30)

- Middleware Next.js : rediriger les utilisateurs non authentifiés
- Accéder à la session côté serveur avec auth()
- Accéder à la session côté client avec useSession
- Afficher du contenu conditionnel selon le rôle utilisateur
- Cas pratique : dashboard protégé avec rôles admin et utilisateur

Module 7 – Accès base de données avec Prisma (2h)

Prisma avec Next.js (1h)

- Installer et configurer Prisma avec une base SQLite ou PostgreSQL
- Définir un schéma : models, relations, migrations
- Le client Prisma singleton dans un projet Next.js
- Cas pratique : schéma utilisateurs, articles, commentaires

Requêtes Prisma dans les Server Components (1h)

- Lire des données avec findMany, findUnique, include
- Écrire des données depuis une Server Action avec create, update, delete
- Paginer des résultats côté serveur
- Cas pratique : page de liste paginée avec filtres côté serveur

Module 8 – Optimisation des performances et SEO (2h)

Optimisation des assets (1h)

- next/image : redimensionnement automatique, lazy loading, formats WebP et AVIF
- next/font : chargement optimisé des polices sans layout shift
- next/script : charger des scripts tiers sans bloquer le rendu
- Analyser le bundle avec @next/bundle-analyzer

SEO et métadonnées (1h)

- L'API Metadata de Next.js : title, description, openGraph, twitter
- Métadonnées dynamiques avec generateMetadata
- Sitemap et robots.txt générés programmatiquement
- Structured data (JSON-LD) dans un Server Component
- Cas pratique : pages blog avec métadonnées dynamiques et Open Graph

Jour 4 (7h) – Patterns avancés et tests

Module 9 – Patterns avancés Next.js (3h)

Parallel Routes et Intercepting Routes (1h)

Parallel Routes : afficher plusieurs pages dans le même layout simultanément

- Intercepting Routes : ouvrir une modale sans quitter la page courante
- Cas pratique : galerie photo avec modale de détail et URL dédiée

Internationalisation (i18n) (1h)

- Configurer les locales dans Next.js avec middleware
- Détection automatique de la langue du navigateur
- Traductions côté serveur et côté client
- Cas pratique : application bilingue français/anglais

Sécurité et bonnes pratiques (1h)

- Headers de sécurité HTTP dans next.config.js : CSP, X-Frame-Options
- Valider les entrées des Server Actions avec Zod
- Variables d'environnement : ne jamais exposer les secrets côté client
- Cas pratique : audit de sécurité de l'application et correction des vulnérabilités

Module 10 – Tests et qualité de code (2h)

Tester une application Next.js (1h)

- Tester un Server Component avec Jest et React Testing Library
- Mocker les Server Actions dans les tests
- Tester les routes API avec des requêtes simulées
- Cas pratique : suite de tests pour les pages et Server Actions critiques

Qualité et outillage (1h)

- ESLint avec la configuration next/core-web-vitals
- Prettier : formater le code automatiquement
- Husky et lint-staged : vérifier le code avant chaque commit
- Cas pratique : configurer le pipeline de qualité complet sur le projet

Module 11 – Déploiement et production (2h)

- Build de production : analyser la sortie next build
- Déployer sur Vercel : connexion GitHub, variables d'environnement, domaine custom
- Preview deployments : tester chaque branche avant la mise en production
- Monitoring : Vercel Analytics et Speed Insights
- Self-hosting : déployer sur un VPS avec Node.js ou Docker
- Cas pratique : déploiement complet de l'application sur Vercel

Jour 5 (7h) – Projet de synthèse

Module 12 – Projet de synthèse (7h)

Réalisation d'une application Next.js complète

- Cahier des charges : application de blog ou marketplace avec authentification
- Fonctionnalités : App Router avec layouts imbriqués, Server Components et Client Components, Server Actions pour les mutations avec validation Zod, authentification NextAuth.js avec rôles, accès base de données avec Prisma, métadonnées SEO dynamiques, optimisation des images et des polices, déploiement sur Vercel
- Étape 1 : mise en place de l'architecture et du schéma Prisma
- Étape 2 : authentification et protection des routes

Étape 3 : pages publiques et back-office avec Server Actions

- Étape 4 : optimisation SEO, performances et déploiement
- Revue de code collective : choix de rendu, architecture, sécurité
- Présentation individuelle du projet : choix techniques et axes d'amélioration
- Retour formateur individualisé sur le projet rendu

5 COMPÉTENCES VISÉES

- Concevoir et développer une application Next.js complète avec App Router
- Choisir la stratégie de rendu adaptée à chaque page selon le contexte (SEO, temps réel, données statiques)
- Gérer les appels de données côté serveur avec le système de cache de Next.js
- Sécuriser une application avec un système d'authentification par sessions ou JWT
- Optimiser les performances front-end mesurables : LCP, CLS, FID
- Déployer et maintenir une application Next.js en production sur Vercel

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou distanciel (visioconférence). Le formateur alterne entre méthode démonstrative (live coding commenté et progressif), méthode interrogative (analyse critique de choix d'architecture) et méthode active (exercices pratiques et mini-projets par module). L'accent est mis sur la compréhension du modèle mental Next.js — notamment la frontière serveur/client — plutôt que sur la mémorisation de l'API.

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec exercices et corrections par module
- Environnement de développement : VS Code + Node.js + Prisma + Vercel CLI
- Pour le distanciel : visioconférence (Zoom ou équivalent), partage d'écran, chat en direct
- Accès à la plateforme pédagogique LaPolaris (supports, ressources, émargement)

8 MODALITÉS D'ÉVALUATION

- En cours de formation : exercices pratiques et mini-projets corrigés à chaque module
- En fin de formation : réalisation et présentation d'une application Next.js complète avec authentification, base de données et déploiement
- Questionnaire d'auto-évaluation des acquis en fin de parcours

9 CRITÈRES D'ÉVALUATION

- Choix justifié de la stratégie de rendu selon le contexte de chaque page (SSG, SSR, ISR)
- Utilisation correcte des Server Components et Client Components sans violation de la frontière serveur/client
- Implémentation fonctionnelle de l'authentification avec protection des routes
- Mutations de données fiables via Server Actions avec validation Zod et revalidation du cache
- Qualité et lisibilité du code : organisation des fichiers, conventions Next.js, absence de logique dupliquée

10 MODALITÉS DE VALIDATION

Attestation de fin de formation délivrée à l'issue du parcours, conditionnée à une assiduité d'au moins 80 % et à la réalisation du projet de synthèse. L'attestation précise les objectifs atteints et les compétences acquises.

11 SUIVI ET ACCOMPAGNEMENT

- Feuilles d'émargement signées par demi-journée (présentiel) ou émargement numérique (distanciel)
- Traçabilité des activités pédagogiques réalisées
- Attestation d'assiduité délivrée en fin de formation
- Suivi individuel via les exercices corrigés et le projet de synthèse

12 CONDITIONS D'ACCÈS

Formation accessible sur inscription directe, sans prérequis administratif particulier. Le financement peut être pris en charge par l'employeur dans le cadre d'un plan de développement des compétences, ou en autofinancement avec possibilité de paiement en plusieurs fois.

13 DÉLAIS D'ACCÈS

Inscription possible jusqu'à **5 jours ouvrés** avant le début de la session. Pour toute demande urgente, nous contacter directement.

ACCESSIBILITÉ · HANDICAP

Nos formations sont accessibles aux personnes en situation de handicap. Pour toute situation nécessitant un aménagement (matériel, temporel ou pédagogique), nous vous invitons à nous contacter avant l'inscription afin d'étudier les adaptations possibles. Référent handicap : contact@lapolaris.fr

LaPolaris

TÉL. +33762584798

EMAIL contact@lapolaris.fr

WEB lapolaris.fr