DÉVELOPPEMENT FULLSTACK

Fullstack avec Next.js : API Routes et base de données

Exploiter Next.js comme framework fullstack : Server Actions, API Routes, connexion à une base de données avec Prisma et déploiement production.

DURÉE	TARIF HT	NIVEAU	LANGUE	GROUPE	FORMAT
5j (35h)	**1790.00 €**	**Avancé**	**Français**	**3-12**	**Formation**

1 PUBLIC VISÉ

- Développeurs Next.js maîtrisant l'App Router souhaitant exploiter le framework comme solution fullstack complète.
- Personnes ayant suivi la formation Next.js : construire des applications web modernes ou disposant d'une expérience Next.js équivalente en production.
- Développeurs React souhaitant gérer le backend et le frontend dans un seul projet sans framework séparé.
- Personnes visant un poste de développeur fullstack JavaScript en entreprise ou en freelance.

2 PRÉREQUIS

Maîtrise de Next.js App Router : Server Components, Client Components, Server Actions, routing. La formation Next.js : construire des applications web modernes est recommandée. Notions de SQL et de base de données relationnelle. Savoir utiliser le terminal et VS Code.

3 OBJECTIFS PÉDAGOGIQUES

- Comprendre les différentes architectures fullstack possibles avec Next.js et choisir la plus adaptée
- Modéliser et gérer une base de données relationnelle avec Prisma ORM
- Maîtriser les API Routes et les Server Actions pour les mutations de données
- Implémenter une authentification complète avec NextAuth.js v5
- Gérer les sessions, les cookies et la protection des routes
- Gérer l'upload et le stockage de fichiers dans une application Next.js
- Sécuriser les variables d'environnement et les secrets en production
- Déployer une application fullstack Next.js sur Vercel avec base de données cloud

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) – Architecture fullstack et Prisma ORM

Module 1 – Architectures fullstack avec Next.js (2h)

Choisir son architecture (1h)

- Next.js fullstack vs Next.js + API séparée : avantages et compromis de chaque approche
- Quand Next.js seul suffit et quand un backend dédié s'impose
- Le modèle mental fullstack Next.js : Server Components, Server Actions, API Routes
- Vue d'ensemble du projet fil rouge : plateforme de contenu avec espace public et back-office

Mise en place de l'environnement (1h)

- Créer le projet Next.js avec TypeScript, Tailwind et les dépendances nécessaires
- Configurer les variables d'environnement : .env.local, .env.production
- Organiser les dossiers : app/, lib/, components/, actions/, types/
- Cas pratique : projet opérationnel avec structure professionnelle

Module 2 – Prisma ORM : modélisation et requêtes (5h)

Installer et configurer Prisma (1h)

- Installer Prisma et initialiser le schéma
- Configurer la connexion à PostgreSQL en local avec Docker
- Le client Prisma singleton : éviter les connexions multiples en développement Next.js
- Prisma Studio : explorer et modifier les données visuellement

Modéliser le schéma (2h)

- Les types de données Prisma : String, Int, Boolean, DateTime, Enum
- Les relations : @relation, one-to-many, many-to-many avec table de jointure implicite
- Les attributs de champ : @id, @unique, @default, @updatedAt
- Générer et exécuter les migrations avec prisma migrate dev
- Seeder la base de données avec prisma db seed
- Cas pratique : schéma complet User, Post, Category, Tag avec toutes les relations

Requêtes Prisma dans Next.js (2h)

- CRUD avec Prisma : create, findMany, findUnique, update, delete
- Filtres : where, contains, startsWith, in, gte, lte
- Relations imbriquées : include et select pour charger les données liées
- Pagination avec take et skip
- Transactions Prisma : garantir l'atomicité des opérations multiples
- Cas pratique : page de liste et de détail avec données Prisma depuis un Server Component

Jour 2 (7h) – API Routes, Server Actions et mutations

Module 3 – API Routes avec App Router (3h)

Créer des API Routes (1h30)

- Structure d'une API Route dans App Router : route.ts
- Handlers HTTP : GET, POST, PUT, PATCH, DELETE
- Lire les paramètres d'URL, les query params et le body JSON
- NextResponse : retourner du JSON avec le bon statut HTTP

Middleware Next.js : intercepter les requêtes avant les routes

- Cas pratique : API REST complète pour une ressource avec pagination

Quand utiliser les API Routes (1h30)

- API Routes vs Server Actions : la règle de décision
- Exposer une API consommée par des clients externes : mobile, tiers
- Webhooks : recevoir et traiter des événements depuis des services externes
- Rate limiting sur les API Routes avec des headers personnalisés
- Cas pratique : webhook de paiement Stripe simulé

Module 4 – Server Actions : mutations sans API (4h)

Créer et appeler des Server Actions (2h)

- Déclarer une Server Action avec use server dans un fichier dédié actions/
- Appeler une Server Action depuis un formulaire : action={serverAction}
- Appeler une Server Action depuis un Client Component avec un handler
- Revalider le cache après une mutation : revalidatePath et revalidateTag
- Redirections depuis une Server Action avec redirect()
- Cas pratique : création et suppression d'articles via Server Actions

Validation et gestion des erreurs (2h)

- Valider les données côté serveur avec Zod avant toute persistance
- Retourner des erreurs de validation au formulaire avec useFormState
- useFormStatus : afficher l'état pending du formulaire
- useOptimistic : mettre à jour l'UI avant la confirmation serveur
- Try/catch dans les Server Actions : gérer les erreurs Prisma
- Cas pratique : formulaire de création avec validation Zod et feedback d'erreur par champ

Jour 3 (7h) – Authentification avec NextAuth.js v5

Module 5 – Mise en place de NextAuth.js (3h)

Configurer NextAuth.js v5 (1h30)

- Installer et configurer NextAuth.js v5 avec App Router
- Le fichier auth.ts : configuration centralisée des providers et callbacks
- Prisma Adapter : stocker les sessions et les comptes en base de données
- Provider Credentials : authentification par email et mot de passe avec bcrypt
- Providers OAuth : Google et GitHub en quelques lignes
- Cas pratique : système de connexion fonctionnel avec Credentials et Google

Sessions et callbacks (1h30)

- Strategy JWT vs database : choisir selon le besoin
- Les callbacks NextAuth : authorize, jwt, session
- Enrichir le token JWT avec des données personnalisées : rôle, id utilisateur
- Accéder à la session côté serveur avec auth() et côté client avec useSession
- Cas pratique : session enrichie avec le rôle de l'utilisateur

Module 6 – Protection des routes et autorisation (4h)

Middleware et routes protégées (2h)

Protéger des routes avec le middleware Next.js et auth()

- Matcher : définir précisément les routes protégées
- Redirection vers la page de connexion pour les utilisateurs non authentifiés
- Page de connexion personnalisée avec le composant SignIn
- Gestion de l'inscription : formulaire, Server Action et création de compte Prisma
- Cas pratique : back-office entièrement protégé avec redirection automatique

Autorisation par rôle (2h)

- Implémenter les rôles avec un enum Prisma : USER, ADMIN, EDITOR
- Vérifier le rôle dans une Server Action avant toute mutation
- Vérifier le rôle dans un Server Component pour le rendu conditionnel
- Vérifier le rôle dans le middleware pour bloquer les routes d'administration
- Cas pratique : système de rôles complet avec trois niveaux d'accès

Jour 4 (7h) – Fonctionnalités avancées et optimisation

Module 7 – Upload et stockage de fichiers (2h)

Gérer l'upload dans Next.js (1h)

- Uploader un fichier depuis un formulaire vers une API Route
- Valider le type MIME et la taille du fichier côté serveur
- Stocker localement dans public/ pour le développement

Stockage cloud (1h)

- Uploader vers Cloudinary ou AWS S3 depuis une Server Action
- Stocker l'URL du fichier dans Prisma après l'upload
- Supprimer un fichier du stockage cloud lors de la suppression en base
- Cas pratique : upload d'avatar utilisateur avec stockage Cloudinary

Module 8 – Sécurité de l'application (2h)

Sécuriser une application Next.js fullstack (1h)

- Headers de sécurité HTTP dans next.config.js : CSP, X-Frame-Options, HSTS
- Protection CSRF dans les Server Actions : vérification de l'origine
- Valider toutes les entrées avec Zod : ne jamais faire confiance aux données client
- Éviter l'exposition des erreurs Prisma en production

Gestion des secrets et des variables d'environnement (1h)

- Variables côté serveur vs variables exposées au client : NEXT_PUBLIC_
- Ne jamais exposer les secrets dans les Client Components ou les API Routes publiques
- Valider les variables d'environnement au démarrage avec Zod
- Cas pratique : audit de sécurité de l'application et correction des vulnérabilités

Module 9 – Performance et cache (3h)

Stratégies de cache Next.js (1h30)

- Le cache de fetch Next.js : force-cache, no-store, revalidate
- unstable_cache : mettre en cache les résultats de fonctions Prisma
- Les tags de cache : invalider précisément avec revalidateTag
- Full Route Cache : comprendre quand Next.js met une page en cache statique

Optimisation des performances (1h30)

- Éviter les waterfalls de données avec Promise.all dans les Server Components
- Streaming avec Suspense : afficher les données critiques en priorité
- Analyser le bundle avec @next/bundle-analyzer
- Cas pratique : optimiser une page lente avec cache et Suspense

Jour 5 (7h) – Déploiement et projet de synthèse

Module 10 – Déploiement sur Vercel avec base de données Neon (3h)

Configurer la base de données de production (1h)

- Créer une base de données PostgreSQL sur Neon : branchement et connection string
- Configurer Prisma pour le pooling de connexions en serverless avec Prisma Accelerate
- Exécuter les migrations en production : prisma migrate deploy
- Variables d'environnement sur Vercel : DATABASE_URL, NEXTAUTH_SECRET

Déployer sur Vercel (2h)

- Connecter le dépôt GitHub à Vercel
- Configurer les variables d'environnement de production dans Vercel
- Preview deployments : tester chaque branche avant la mise en production
- Domaine personnalisé et HTTPS automatique
- Vercel Analytics et Speed Insights : monitorer les performances en production
- Cas pratique : application complète déployée et accessible en production

Module 11 – Projet de synthèse (4h)

Réalisation d'une application fullstack Next.js complète

- Cahier des charges : plateforme de blog ou marketplace avec espace public et back-office
- Fonctionnalités : authentification NextAuth.js avec Credentials et OAuth, rôles USER et ADMIN, CRUD complet via Server Actions avec validation Zod, Prisma avec toutes les relations du schéma, upload de fichiers vers Cloudinary, cache optimisé avec revalidation, déploiement sur Vercel avec Neon
- Étape 1 : schéma Prisma et migrations
- Étape 2 : authentification et protection des routes
- Étape 3 : Server Actions CRUD avec validation et cache
- Étape 4 : déploiement sur Vercel et recette finale
- Revue de code collective : architecture, sécurité, performances
- Retour formateur individualisé sur le projet rendu

5 COMPÉTENCES VISÉES

- Concevoir et développer une application fullstack complète avec Next.js comme unique framework
- Modéliser un schéma de base de données avec Prisma et gérer les migrations
- Implémenter une authentification sécurisée avec NextAuth.js et protection des routes
- Gérer les mutations de données avec les Server Actions et la revalidation du cache
- Déployer et maintenir une application Next.js en production sur Vercel avec base de données Neon
- Sécuriser une application fullstack : validation, autorisation, secrets et headers HTTP

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou distanciel (visioconférence). Le formateur alterne entre méthode démonstrative (construction progressive d'une application fullstack tout au long de la semaine), méthode interrogative (analyse des choix d'architecture Next.js et des compromis cache/fraîcheur des données) et méthode active (exercices pratiques et projet de synthèse fil rouge). L'accent est mis sur les pratiques professionnelles réelles : sécurité, performances, déploiement et qualité du code.

7 MOYENS PÉDAGOGIQUES ET TECHNIQUES

- Support de cours numérique mis à disposition des apprenants
- Dépôt GitHub de démonstration avec le projet fil rouge versionné par étape
- Environnement de développement : VS Code + Node.js + Docker Desktop + Prisma Studio
- 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 corrigés à chaque module
- En fin de formation : réalisation d'une application fullstack Next.js complète déployée sur Vercel
- Questionnaire d'auto-évaluation des acquis en fin de parcours

9 CRITÈRES D'ÉVALUATION

- Schéma Prisma correctement modélisé avec relations et migrations fonctionnelles
- Authentification NextAuth.js opérationnelle avec protection des routes par rôle
- Server Actions validées avec Zod et cache correctement revalidé après chaque mutation
- Sécurité de l'application : secrets non exposés, validation des entrées, headers HTTP configurés
- Application déployée et accessible en production sur Vercel avec base de données Neon

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