DÉVELOPPEMENT FULLSTACK

Fullstack React + Symfony : application web complète

Développer une application web complète avec React en frontend et Symfony / API Platform en backend. Un parcours intensif pour devenir développeur fullstack opérationnel.

DURÉE	TARIF HT	NIVEAU	LANGUE	GROUPE	FORMAT
5j (35h)	**1990.00 €**	**Avancé**	**Français**	**3-12**	**Formation**

1 PUBLIC VISÉ

- Développeurs maîtrisant React et Symfony souhaitant assembler leurs compétences pour créer une application web complète et découplée.
- Personnes ayant suivi les formations React : développer des interfaces modernes et Symfony 7 : initiation au framework PHP professionnel ou disposant d'une expérience équivalente.
- Développeurs front-end ou back-end souhaitant élargir leur profil vers le fullstack en conditions réelles.
- Personnes visant un poste de développeur fullstack JavaScript/PHP en entreprise ou en freelance.

2 PRÉREQUIS

- Maîtrise de React : composants, hooks, React Router, appels API. Maîtrise de Symfony : routing, Doctrine ORM, formulaires, Security Bundle.
- Les formations React : développer des interfaces modernes et Symfony 7 : initiation au framework PHP professionnel sont requises ou une expérience équivalente.
- Notions de base sur HTTP, REST et les tokens JWT.

3 OBJECTIFS PÉDAGOGIQUES

- Concevoir et architecturer une application découplée avec React en frontend et Symfony en backend
- Créer une API REST avec API Platform : ressources, filtres, pagination et sérialisation
- Implémenter une authentification JWT de bout en bout avec LexikJWTBundle
- Développer le frontend React consommant l'API : routing, state management et gestion des tokens
- Gérer les rôles et les permissions côté API et côté client React
- Gérer l'upload de fichiers entre le frontend et l'API Symfony
- Déployer l'application complète sur un VPS avec Nginx et Docker Compose

4 PROGRAMME DÉTAILLÉ

Jour 1 (7h) – Architecture du projet et API Platform

Module 1 – Architecture d'une application fullstack découplée (2h)

Concevoir l'architecture (1h)

- Monorepo vs multi-repo : avantages, inconvénients, choix pour ce projet
- Le flux de communication frontend/backend : requêtes HTTP, JSON, CORS
- Configurer CORS dans Symfony pour autoriser les requêtes depuis React
- Structure du projet : dossier backend/ Symfony et dossier frontend/ React
- Les outils du développeur fullstack : Postman, React DevTools, Symfony Profiler

Mise en place de l'environnement (1h)

- Initialiser le projet Symfony avec les dépendances API Platform
- Initialiser le projet React avec Vite et configurer le proxy vers l'API
- Configurer la base de données PostgreSQL et les variables d'environnement
- Cas pratique : monorepo opérationnel avec les deux projets qui communiquent

Module 2 – API Platform : créer une API REST (5h)

Premiers pas avec API Platform (2h)

- Ce qu'API Platform apporte par rapport à Symfony seul : génération automatique, documentation
- Créer une ressource API avec l'attribut #[ApiResource]
- Les opérations : GetCollection, Get, Post, Put, Patch, Delete
- La documentation Swagger UI générée automatiquement
- Tester les endpoints depuis Swagger UI et Postman
- Cas pratique : ressource Article avec CRUD complet documenté

Sérialisation et groupes (1h30)

- Les groupes de sérialisation : contrôler les champs exposés par opération
- normalization_context et denormalization_context
- Exposer des champs calculés avec des getters
- Gérer les relations imbriquées : IRI vs objet embarqué
- Cas pratique : ressource Article avec groupes de lecture et d'écriture distincts

Filtres et pagination (1h30)

- SearchFilter : filtrer par valeur exacte ou partielle
- RangeFilter et DateFilter : filtrer par intervalle
- OrderFilter : trier les résultats depuis l'URL
- La pagination API Platform : configurer le nombre d'éléments par page
- Cas pratique : endpoint de recherche d'articles avec filtres et pagination

Jour 2 (7h) – Authentification JWT de bout en bout

Module 3 – Authentification JWT côté Symfony (3h)

Mise en place de LexikJWTBundle (1h30)

- Installer et configurer LexikJWTAuthenticationBundle
- Générer les clés RSA pour signer les tokens
- Configurer le Security Bundle : firewall API stateless avec JWT

L'endpoint /api/login : recevoir les identifiants et retourner un JWT

- Structure d'un JWT : header, payload, signature
- Cas pratique : obtenir un token JWT depuis Postman

Protéger les ressources API (1h30)

- Restreindre les opérations API Platform selon l'authentification
- Les Security Voters dans API Platform : contrôle d'accès par objet
- Exposer l'utilisateur courant via un endpoint /api/me
- Rafraîchir un token expiré avec gesdinet/jwt-refresh-token-bundle
- Cas pratique : endpoint CRUD articles réservé aux utilisateurs authentifiés

Module 4 – Authentification JWT côté React (4h)

Gérer les tokens dans React (2h)

- Stocker le JWT : localStorage vs httpOnly cookie — sécurité et compromis
- Créer un contexte AuthContext pour partager l'état d'authentification
- Le hook useAuth : accéder à l'utilisateur et aux méthodes login/logout
- Intercepteur Axios : injecter automatiquement le token dans les requêtes
- Gérer l'expiration du token : intercepter les 401 et rediriger vers la connexion
- Cas pratique : connexion avec persistance du token et déconnexion

Routes protégées et gestion des rôles (2h)

- Composant PrivateRoute : rediriger si non authentifié
- Décoder le JWT côté client pour lire les rôles sans appel API
- Affichage conditionnel selon le rôle : masquer les actions non autorisées
- Synchroniser l'état d'authentification entre les onglets du navigateur
- Cas pratique : interface avec sections publiques et sections réservées aux admins

Jour 3 (7h) – Frontend React et consommation de l'API

Module 5 – Architecture React pour une application fullstack (3h)

Structure et organisation du projet React (1h)

- Organisation des dossiers : pages/, components/, hooks/, services/, contexts/
- La couche service : encapsuler les appels API dans des fonctions dédiées
- Typer les réponses API avec PropTypes ou TypeScript
- Variables d'environnement Vite : VITE_API_URL pour pointer vers le backend

State management pour une application fullstack (2h)

- Quand useState suffit et quand passer à un state global
- Context API pour les données partagées : utilisateur, panier, notifications
- Le pattern optimistic update : mettre à jour l'UI avant la confirmation API
- Gestion des états asynchrones : loading, error, data dans les hooks personnalisés
- Cas pratique : hook useArticles avec pagination, filtres et gestion des erreurs

Module 6 – Fonctionnalités métier de l'application (4h)

CRUD complet depuis React (2h)

- Lister les ressources avec pagination côté API
- Formulaire de création avec validation côté client et gestion des erreurs 422 de l'API

Formulaire d'édition avec pré-remplissage des données existantes

- Suppression avec confirmation et mise à jour optimiste de la liste
- Cas pratique : interface de gestion d'articles complète

Upload de fichiers (2h)

- Configurer VichUploaderBundle côté Symfony pour gérer les fichiers
- Créer un endpoint d'upload dédié dans API Platform
- Envoyer un fichier depuis React avec FormData et Axios
- Afficher un aperçu du fichier avant l'envoi
- Cas pratique : upload d'image de couverture pour un article

Jour 4 (7h) – Fonctionnalités avancées et qualité

Module 7 – Gestion avancée des erreurs et UX (2h)

Erreurs API et feedback utilisateur (1h)

- Mapper les erreurs de validation Symfony (RFC 7807) vers les champs du formulaire React
- Notifications globales : système de toast pour les succès et les erreurs
- Error boundaries React : capturer les erreurs de rendu
- Page 404 et redirections côté React Router

Performance et expérience utilisateur (1h)

- Skeleton loaders : afficher des squelettes pendant le chargement
- Infinite scroll vs pagination classique : implémentation avec l'API Platform
- Mémoïsation des appels API : éviter les requêtes redondantes
- Cas pratique : liste d'articles avec skeleton loader et pagination infinie

Module 8 – Tests de l'application fullstack (2h)

Tests de l'API Symfony (1h)

- Tester les endpoints API Platform avec WebTestCase
- Authentifier un utilisateur dans les tests avec un JWT de test
- Vérifier les codes HTTP, la structure JSON et les permissions
- Cas pratique : suite de tests pour les endpoints critiques de l'API

Tests du frontend React (1h)

- Tester un composant qui consomme une API avec Mock Service Worker
- Tester le flux d'authentification : connexion, redirection, déconnexion
- Cas pratique : tests du formulaire de connexion et d'une page protégée

Module 9 – Projet de synthèse : phase 1 (3h)

- Cahier des charges : plateforme de gestion de contenu avec espace public et back-office
- Mise en place de l'architecture monorepo et des deux projets
- Modélisation des entités Symfony et configuration API Platform
- Authentification JWT fonctionnelle des deux côtés
- Premières pages React connectées à l'API

Jour 5 (7h) – Déploiement et finalisation du projet

Module 10 – Déploiement sur VPS (3h)

Préparer l'application pour la production (1h)

Variables d'environnement de production : .env.local, secrets Symfony

- Build de production React : vite build et optimisation des assets
- Configuration Nginx : servir le frontend React et proxyer l'API Symfony
- HTTPS avec Let's Encrypt et Certbot

Docker Compose en production (2h)

- Dockerfile pour Symfony : PHP-FPM, extensions, Composer
- Dockerfile pour React : build multi-stage avec Nginx
- docker-compose.yml : orchestrer Symfony, React, PostgreSQL et Nginx
- Déployer sur un VPS : cloner, configurer, lancer
- Migrations en production : doctrine:migrations:migrate sans interruption
- Cas pratique : déploiement complet de l'application sur un VPS

Module 11 – Projet de synthèse : phase 2 et présentation (4h)

Finalisation du projet

- Compléter les fonctionnalités : CRUD complet, upload de fichiers, gestion des rôles
- Déployer l'application sur le VPS ou Docker Compose en local
- Revue de code collective : architecture, sécurité, cohérence API/frontend
- Présentation individuelle du projet : choix techniques, difficultés rencontrées, axes d'amélioration
- Retour formateur individualisé sur le projet rendu

5 COMPÉTENCES VISÉES

- Concevoir l'architecture d'une application fullstack découplée frontend/backend
- Développer et documenter une API REST avec API Platform
- Implémenter une authentification JWT sécurisée de A à Z
- Consommer une API REST depuis React avec gestion des états et des erreurs
- Gérer les permissions côté serveur et côté client de manière cohérente
- Déployer et maintenir une application fullstack en production sur un VPS

6 MODALITÉS PÉDAGOGIQUES

Formation délivrée en présentiel ou distanciel (visioconférence). Le formateur alterne entre méthode démonstrative (construction progressive d'une application fullstack complète tout au long de la semaine), méthode interrogative (analyse des choix d'architecture et des compromis frontend/backend) et méthode active (exercices pratiques et projet de synthèse fil rouge). L'accent est mis sur les pratiques professionnelles réelles : découplage, sécurité, 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 + Symfony CLI + Vite + Docker Desktop
- 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 et présentation d'une application fullstack complète déployée
- Questionnaire d'auto-évaluation des acquis en fin de parcours

9 CRITÈRES D'ÉVALUATION

- Architecture découplée fonctionnelle : frontend React et backend Symfony communiquant via l'API
- Authentification JWT opérationnelle de bout en bout avec gestion des rôles
- API Platform correctement configurée avec groupes de sérialisation et filtres
- Gestion cohérente des erreurs API côté React avec feedback utilisateur explicite
- Application déployée et accessible avec HTTPS sur un VPS ou via Docker Compose

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