Formation **React Native Avancé**

Plongez au cœur de la nouvelle architecture de React Native ! En deux jours intensifs, cette formation avancée vous fait passer du simple développement mobile à la maîtrise des performances et des patterns professionnels : Fabric, Turbo Modules, JSI et Hermes n'auront plus de secrets pour vous. Vous découvrirez le bridging natif, mettrez en place des tests E2E robustes avec Maestro et MSW, optimiserez vos listes avec FlashList et saurez profiler vos apps comme un·e pro grâce à Flipper, Xcode Instruments et Android Profiler. Parfait pour les développeurs déjà aguerris à React Native ou Expo qui veulent livrer des applications plus rapides, plus fiables et taillées pour la production.

Durée
2 jours

Objectifs pédagogiques

- Approfondir la nouvelle architecture de React Native : Fabric, Turbo Modules, JSI, Hermes
- Comprendre le bridging natif (avec exemples C++), sans exiger une maîtrise totale du langage, mais en montrant l'architecture et les bonnes pratiques
- Mettre en place des tests avancés, y compris E2E (Maestro), tests mutateurs et mocking réseau (MSW)
- Découvrir une gestion de données avancée (ex. TanStack Query) et des patterns architecturaux (Data layer, Domain layer, etc.)
- Maîtriser les outils de performance et de profiling (Flipper, Xcode Instruments, Android Profiler) et les optimisations (FlashList, images, animations)

Public
Développeurs

Prérequis
Bases solides en JavaScript/TypeScript. Expérience réelle en React Native (ou Expo), navigation, Hooks, etc.

Programme de formation

Phase d'inclusion

Accueil des participants, présentation des objectifs et contextes professionnels de chacun.

Nouvelle architecture avancée (Fabric, Turbo Modules, JSI, Hermes)

Hermes & JSI :
Pourquoi Hermes ? Gains en perf & taille d'app.
Comment vérifier et activer Hermes sur un projet Expo ou bare RN.
Intro à la JSI et différence avec le Bridge classique.
Turbo Modules & Fabric :
Qu'est-ce que Fabric ? Comment ça impacte le rendu ?
Comment activer la nouvelle architecture concrètement et vérifier le bon fonctionnement.

Bridging natif & exemple React Native Vision Camera

Pourquoi le bridging ? : gains de performances, accès aux APIs iOS/Android, réutilisation de code existant (C/C++).
Exemple concret :
Aperçu de react-native-vision-camera (qui inclut du code C++).
Architecture côté iOS (Swift/ObjC) et Android (Kotlin/Java/C++).
Comment le code C++ est relié au JavaScript via la JSI / Turbo Module.
Démonstration (sans exiger de coder du C++ pur) :
Survol du code natif, discussion sur les patterns de bridging.
Tips pour déboguer un module natif.

Data Layer avancé : TanStack Query & architecture

TanStack Query (React Query) :
Gestion avancée du cache, invalidations, requêtes parallèles, pagination.
Stratégies offline-first : persister le cache en local.
Architecture modulaire :
Séparation en Data layer / Domain layer / UI layer.
Comparaison rapide avec Redux, Zustand (selon préférences).

Tests avancés : unitaires, composants, E2E, mutation testing

React Native Testing Library
Configuration Jest avancée, mocking des modules natifs.
Tests de composants complexes (navigation, hooks personnalisés).
Mock Service Worker (MSW) :
Principe et avantages pour mocker les requêtes réseau.
Mise en place sur un projet React Native (ou Expo) – plus simple côté web, mais possible via proxy / Node.
Tests E2E :
Introduction à Maestro : installation, scripts, interactions sur l'UI.
Comparaison avec Detox / Appium.
Intégration en CI (GitHub Actions, Bitrise, CircleCI…).
Mutation Testing (tests mutateurs) :
Concepts (Stryker, etc.).
Comment ça aide à valider la robustesse de la suite de tests.

Debugging & Profiling avancés

Outils : Flipper, React Native Debugger, Chrome Dev Tools.
Profiling :
Xcode Instruments (Time Profiler, mémoire), Android Profiler (CPU, mémoire).
Analyser les blocages JS thread vs. UI thread.
Stratégies de correction (memo, useCallback, scinder des composants).

Optimisations réelles : listes, images, animations

FlashList :
Avantages (cell recycling), différence vs. FlatList, Benchmarks.
Migration / usage concret, gestion des images dans la liste.
Gestion des images :
Cache, lazy loading, librairies alternatives (Fast Image).
Redimensionnement, réduction du poids, format WebP, etc.
Animations avancées (si le public le demande) :
Reanimated 2/3, le concept de thread UI, synchronisation JS/UI.

Déploiement avancé et Expo Modules

Expo Modules :
Config plugins, personnalisation iOS/Android.
Quand et comment passer en bare workflow si on a besoin de plus de contrôle.
Déploiement :
EAS Build : build config, Over-The-Air updates contrôlés.
Hors Expo : signature APK/IPA, distribution TestFlight / Play Store.
Optimisations finales (Hermes bundle, minification, suppression logs).

Moyens et méthodes pédagogiques

- La formation alterne entre présentations des concepts théoriques et mises en application à travers d'ateliers et exercices pratiques (hors formation de type séminaire).
- Les participants bénéficient des retours d'expérience terrains du formateur ou de la formatrice
- Un support de cours numérique est fourni aux stagiaires

Modalités d'évaluation

- **En amont de la session de formation**, un questionnaire d'auto-positionnement est remis aux participants, afin qu'ils situent leurs connaissances et compétences déjà acquises par rapport au thème de la formation.

- **En cours de formation**, l'évaluation se fait sous forme d'ateliers, exercices et travaux pratiques de validation, de retour d'observation et/ou de partage d'expérience, en cohérence avec les objectifs pédagogiques visés.

- **En fin de session**, le formateur évalue les compétences et connaissances acquises par les apprenants grâce à un questionnaire reprenant les mêmes éléments que l'auto-positionnement, permettant ainsi une analyse détaillée de leur progression.