Formation : Campus Atlas - Kotlin Multiplatform, développer des applications multiplateformes

Cours pratique - 3j - 21h00 - Réf. LKD
Prix : 1650 € H.T.

NEW

À l'issue de cette formation, les participants seront en mesure de réduire significativement les coûts de développement et de maintenance, d'accélérer le time-to-market de leurs applications, et de dialoguer efficacement avec les équipes natives lors de l'intégration des modules KMP dans les projets existants. Ce programme de formation est destiné aux salariés des branches professionnelles relevant de l'OPCO Atlas.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Connaître les concepts de Kotlin Multiplateform
- Mettre en place un environnement de développement
- Concevoir l'architecture d'une application mobile multiplateforme
- Construire une interface utilisateur fluide et performante
- Réaliser des tests pour assurer la qualité et la fiabilité des applications

Public concerné

Pour les adhérents à l'OPCO Atlas : développeurs, architectes.

Prérequis

Avoir des connaissances en langage de programmation (Java, C#, C++, JavaScript, Python, etc.).

Méthodes et moyens pédagogiques

Méthodes pédagogiques

Pour optimiser le parcours d'apprentissage, des modules e-learning peuvent être fournis avant et après la session présentielle ou la classe virtuelle, sur simple demande du participant.

PARTICIPANTS

Pour les adhérents à l'OPCO Atlas : développeurs, architectes.

PRÉREQUIS

Avoir des connaissances en langage de programmation (Java, C#, C++, JavaScript, Python, etc.).

COMPÉTENCES DU FORMATEUR

Les experts qui animent la formation sont des spécialistes des matières abordées. Ils ont été validés par nos équipes pédagogiques tant sur le plan des connaissances métiers que sur celui de la pédagogie, et ce pour chaque cours qu'ils enseignent. Ils ont au minimum cinq à dix années d'expérience dans leur domaine et occupent ou ont occupé des postes à responsabilité en entreprise.

MODALITÉS D'ÉVALUATION

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques...
Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

Modalités d'évaluation

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques…

Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

Programme de la formation

1 Algorithmique, raisonner avant de concevoir - Contenu digital learning préformation

- Introduction à l'algorithmique.
- Les instructions de base en pseudo-code.

Activités digitales

Cette formation en ligne apprend à raisonner avant de concevoir un programme en découvrant les bases de l'algorithmique. Les participants étudieront notamment les instructions fondamentales en pseudo-code.

2 Introduction à Kotlin Multiplatform

- Problématiques du développement multiplateforme traditionnel.
- Évolution de KMM vers KMP : historique et vision JetBrains.
- Comparaison avec les autres solutions (React Native, Flutter, Xamarin).
- Cas d'usage et success stories (Netflix, VMware, Philips).

Travaux pratiques

Brainstorming collectif : "Quels sont les défis actuels en développement mobile ?". Analyse comparative des solutions multiplateformes en sous-groupes.

3 Architecture et concepts fondamentaux

- Architecture en couches : Common, Android, iOS, web.
- Mécanisme expect/actual : déclarations et implémentations.
- Stratégies de partage de code : what to share versus what not to share.

Travaux pratiques

Création de premières déclarations expect/actual simples.

4 Configuration de l'environnement

- Prérequis système par plateforme (Windows, macOS, Linux).
- Installation des outils : Android Studio, Xcode, plugins KMP.
- Configuration Gradle et structure de projet.

Travaux pratiques

Installation guidée sur les postes de travail avec vérification collective.

MOYENS PÉDAGOGIQUES ET TECHNIQUES

• Les moyens pédagogiques et les méthodes d'enseignement utilisés sont principalement : aides audiovisuelles, documentation et support de cours, exercices pratiques d'application et corrigés des exercices pour les formations pratiques, études de cas ou présentation de cas réels pour les séminaires de formation.
• À l'issue de chaque formation ou séminaire, ORSYS fournit aux participants un questionnaire d'évaluation du cours qui est ensuite analysé par nos équipes pédagogiques.
• Une feuille d'émargement par demi-journée de présence est fournie en fin de formation ainsi qu'une attestation de fin de formation si le participant a bien assisté à la totalité de la session.

MODALITÉS ET DÉLAIS D'ACCÈS

L'inscription doit être finalisée 24 heures avant le début de la formation.

ACCESSIBILITÉ AUX PERSONNES HANDICAPÉES

Pour toute question ou besoin relatif à l'accessibilité, vous pouvez joindre notre équipe PSH par e-mail à l'adresse psh-accueil@orsys.fr.

5 Premier Projet KMP

- Création d'un projet KMP avec l'assistant IntelliJ.
- Structure des dossiers : commonMain, androidMain, iosMain.
- Première classe partagée et ses implémentations spécifiques.
- Build et exécution sur les différentes plateformes.

Travaux pratiques
Développement en binômes : création d'une app "Hello KMP" avec affichage de la plateforme courante. Présentation des réalisations et retours d'expérience.

6 Gestion des dépendances

- Dépendances communes versus spécifiques.
- Écosystème des librairies KMP : kotlinx, Ktor, SQLDelight.
- Configuration Gradle avancée pour les sourceSets.

Travaux pratiques
Ajout et test de dépendances communes (kotlinx-coroutines, kotlinx-serialization).

7 Patterns et bonnes pratiques

- Organisation des packages et modules.
- Patterns recommandés : Repository, UseCase, ViewModel.
- Gestion d'erreurs multiplateforme avec classes scellées.

Travaux pratiques
Refactoring du projet créé précédemment selon les patterns présentés en sous-groupes

8 Architecture MVVM et Repository Design Pattern

- Clean Architecture adaptée au multiplateforme.
- Implémentation du pattern Repository avec interfaces communes.
- ViewModels partagés avec StateFlow et coroutines.
- Injection de dépendances avec Koin.

Travaux pratiques
Développement d'un gestionnaire d'utilisateurs avec Repository Design Pattern complet.

9 Gestion des données et API

- Client HTTP avec Ktor : configuration et utilisation.
- Sérialisation JSON avec kotlinx.serialization.
- Stockage local avec SQLDelight : setup et requêtes.
- Stratégies de cache et synchronisation.

Travaux pratiques
Création d'un client API REST pour récupérer des données météo. - Implémentation du cache local avec SQLDelight.

10 Gestion d'état et réactivité

- StateFlow et SharedFlow : concepts et différences.
- Gestion des états d'UI : Loading, Success, Error.
- Combinaison de flux avec combine et zip.

Travaux pratiques
Implémentation d'un gestionnaire d'état global pour l'application météo.

11 Interfaces utilisateur avec Compose Multiplatform

- Introduction à Compose Multiplatform : concepts de base.
- Composants UI : Text, Button, LazyColumn, Card.
- Gestion des thèmes et du Material Design.
- Navigation entre écrans avec Compose Navigation.
- Réactivité avec collectAsState.

Travaux pratiques
Création de l'interface de l'application météo avec Compose. Implémentation de la navigation entre écrans de liste et détail.

12 Intégration native

- Intégration Android : utilisation du module shared.
- Intégration iOS : framework et bridge Swift/Kotlin.
- Passage de données entre couches native et partagée.

Travaux pratiques
Intégration des ViewModels KMP dans les activités Android et ViewControllers iOS.

13 Gestion des plateformes spécifiques

- API spécifiques : géolocalisation, caméra, notifications.
- Stratégies d'abstraction des API système.
- Performance et optimisations par plateforme.

Travaux pratiques
Ajout de la géolocalisation à l'app météo avec implémentations spécifiques.

14 Stratégie de tests KMP

- Tests unitaires partagés : CommonTest et spécifiques.
- Mocking avec MockK dans un contexte multiplateforme.
- Tests de ViewModels et Repositories.
- Tests d'intégration avec bases de données et API.

Travaux pratiques
Écriture de tests unitaires pour les composants développés les jours précédents.

15 Tests d'interface utilisateur

- Tests Compose avec ComposeTestRule.
- Tests d'interaction utilisateur et vérification d'états.
- Tests de navigation et de flux utilisateur.
- Stratégies de tests pour les parties natives.

Travaux pratiques
Création de tests UI pour les écrans de l'application météo.

16 Déploiement et CI/CD

- Configuration GitHub Actions pour KMP.
- Build automatisé pour Android (APK/AAB) et iOS (IPA).
- Déploiement sur stores : Play Store, App Store.
- Gestion des certificats et signatures.

Travaux pratiques
Configuration d'un pipeline CI/CD basique avec GitHub Actions.

17 Projet Final - Application e-commerce

- Catalogue de produits avec recherche et filtres.
- Panier d'achat persistant.
- Authentification utilisateur.
- Interface responsive et navigation fluide.
- Tests unitaires et d'intégration.

Travaux pratiques
Architecture et planification du projet. Développement des fonctionnalités core. Tests et finalisations.

18 Perspectives et "aller plus loin"

- Écosystème KMP : nouvelles librairies et évolutions.
- Communauté et ressources : documentation, forums, conférences.
- Stratégies d'adoption en entreprise : migration progressive, formation équipes.
- Roadmap KMP : Compose Multiplatform Web, Desktop, Wasm.

Echanges
Élaboration d'un plan d'action personnalisé pour chaque participant.

19 Kotlin, les bases indispensables - Contenu digital learning post-formation

- Présentation de Kotlin.
- Les fondamentaux de Kotlin.
- Les fonctions.
- Les coroutines.
- La programmation orientée objet.
- Le développement Android.
- Le développement natif.
- Le développement JavaScript.
- Le développement côté serveur.

Activités digitales

Cette formation en ligne présente le langage Kotlin, ses fondamentaux, ses différentes fonctions, la notion de coroutine, la programmation orientée objet avec une démonstration du développement d'une application sous Android Studio, comment utiliser Kotlin pour du développement natif ou côté serveur et comment Kotlin peut générer du code JavaScript.

Dates et lieux

CLASSE À DISTANCE
2026 : 10 mars, 19 mai, 6 oct., 8 déc.

PARIS LA DÉFENSE
2026 : 5 mai, 29 sep., 1 déc.

METZ
2026 : 19 mai, 6 oct.

NANCY
2026 : 10 mars, 6 oct.