# Développement de smart contrat niveau avancé

2 j (14 heures) Ref : IABC006

**Public**

développeurs ayant déjà une expérience de base en programmation de smart contracts et souhaitant approfondir leurs compétences

**Pré-requis**

Connaissance de base de la programmation en Solidity
Expérience pratique dans le développement et le déploiement de smart contracts simples
Familiarité avec les concepts de la blockchain et de la technologie des registres distribués

**Moyens pédagogiques**

Formation réalisée en présentiel ou à distance selon la formule retenue
Nombreux exercices pratiques et mises en situation, échanges basés sur la pratique professionnelle des participants et du formateur, formation progressive en mode participatif. Vidéoprojecteur, support de cours fourni à chaque stagiaire

**Modalités de suivi et d'évaluation**

Feuille de présence émargée par demi-journée par les stagiaires et le formateur
Exercices de mise en pratique ou quiz de connaissances tout au long de la formation permettant de mesurer la progression des stagiaires
Questionnaire d'évaluation de la satisfaction en fin de stage
Auto-évaluation des acquis de la formation par les stagiaires
Attestation de fin de formation

La formation "Développement de Smart Contracts niveau Avancé" est une immersion intensive de deux jours destinée aux développeurs ayant déjà une expérience de base en programmation de smart contracts et souhaitant approfondir leurs compétences. Cette formation explore des concepts avancés de la programmation en Solidity, l'optimisation et la sécurité des smart contracts, ainsi que les meilleures pratiques pour le développement d'applications décentralisées (dApps) robustes et efficaces. Les participants apprendront à créer des smart contracts complexes, à interagir avec différents protocoles et à utiliser des outils avancés pour le développement et le déploiement.

Cette formation avancée de deux jours est conçue pour transformer des développeurs ayant une connaissance de base en Solidity en experts capables de créer, optimiser et sécuriser des smart contracts complexes. Les participants repartiront avec une compréhension approfondie des concepts avancés et des outils nécessaires pour exceller dans le développement d'applications blockchain.

## Objectifs

Maîtriser les concepts avancés de la programmation en Solidity

Optimiser la performance et la sécurité des smart contracts

Développer et déployer des applications décentralisées (dApps) complexes

Intégrer des protocoles DeFi et d'autres services blockchain dans des smart contracts
Utiliser des outils avancés pour le test, le débogage et la gestion des smart contracts
Appliquer les meilleures pratiques pour la sécurité et l'optimisation des gas fees

# Programme détaillé

## PROGRAMMATION AVANCEE EN SOLIDITY

Structures de données complexes : mappings, structs, arrays dynamiques :
Solidity permet la gestion avancée des données avec des structures comme les mappings, structs pour regrouper des données de types différents, et les arrays dynamiques pour stocker des collections de données dont la taille peut varier.
Modifiants, événements, et gestion des erreurs :
Les modifiants sont des fonctionnalités permettant de modifier le comportement des fonctions dans Solidity, utiles pour la gestion des autorisations et des conditions préalables. Les événements permettent de notifier les utilisateurs des changements d'état importants dans les contrats. La gestion des erreurs est cruciale pour assurer la robustesse des smart contracts.
Inheritance et design patterns en Solidity :
L'héritage en Solidity permet la réutilisation du code et la structuration des smart contracts en utilisant des contrats parents et des contrats enfants. Les design patterns comme Singleton, Factory, et Proxy sont des approches avancées pour organiser et optimiser la logique des contrats.

## OPTIMISATION DES SMART CONTRACTS

Techniques d'optimisation pour réduire les gas fees :
Les gas fees sont des frais de traitement sur Ethereum. L'optimisation inclut la réduction de la complexité des opérations, l'utilisation de types de données moins coûteux en gas, et l'évitement de boucles coûteuses pour minimiser les frais de transaction.
Utilisation de la bibliothèque OpenZeppelin pour des implémentations sécurisées et optimisées :
OpenZeppelin offre des bibliothèques sécurisées pour le développement de smart contracts, prévenant des vulnérabilités courantes et facilitant l'implémentation de standards comme ERC-20 et ERC-721 avec des contrats robustes et testés.

## SECURITE DES SMART CONTRACTS

Analyse des vulnérabilités courantes : reentrancy, integer overflow/underflow, etc :
Les vulnérabilités comme la reentrancy (exploitée dans le hack de The DAO), et les erreurs d'overflow/underflow peuvent causer des failles critiques. Comprendre et prévenir ces risques est essentiel pour assurer la sécurité des smart contracts.
Pratiques de codage sécurisé et utilisation d'outils de sécurité (MythX, Slither) :
MythX et Slither sont des outils d'analyse statique qui identifient automatiquement les vulnérabilités dans le code des smart contracts, permettant aux développeurs de corriger les problèmes de sécurité avant le déploiement.
Audit de smart contracts et étude de cas de hacks célèbres :

Les audits de sécurité par des experts sont recommandés pour identifier et corriger les failles potentielles. L'étude des hacks célèbres comme celui de Parity ou de KuCoin enrichit la compréhension des risques et des bonnes pratiques en matière de sécurité.

ATELIER :

Implémentation de Smart Contracts Sécurisés

Les participants développeront, testeront et auditeront des smart contracts complexes en utilisant les meilleures pratiques de sécurité. L'utilisation d'outils comme MythX et la rédaction de tests rigoureux garantiront la robustesse des contrats.

## DEVELOPPEMENT ET INTEGRATION DE DAPPS

## DEVELOPPEMENT DE DAPPS AVANCEES

Les frameworks comme Truffle et Hardhat simplifient le développement d'interfaces utilisateur (UI) interactives qui interagissent avec les smart contracts via Web3.js ou Ethers.js. Cela permet aux développeurs de créer des applications décentralisées (dApps) complexes avec une interface utilisateur conviviale.

## INTEGRATION DES PROTOCOLES DEFI

Les protocoles DeFi comme Uniswap, Compound et Aave sont intégrés aux smart contracts pour offrir des fonctionnalités financières avancées telles que le prêt, le trading et le yield farming. Les développeurs apprendront à intégrer ces protocoles dans leurs dApps pour fournir des services financiers décentralisés.

## TEST ET DEPLOIEMENT AVANCE

Utilisation de Ganache pour des tests locaux avancés :

Ganache permet aux développeurs de simuler des réseaux Ethereum locaux pour tester et déboguer leurs smart contracts dans des environnements contrôlés avant le déploiement sur le réseau principal.

Stratégies de débogage et gestion des migrations de smart contracts :

Les développeurs apprendront des techniques de débogage avancées pour identifier et résoudre les erreurs dans leurs smart contracts. La gestion des migrations assure la mise à jour et la gestion efficace des versions des contrats déployés.

ATELIER :

Projet Capstone : Développement d'une dApp Complexe

Les participants concevront, développeront et déploieront une dApp intégrant des fonctionnalités avancées et des protocoles DeFi. La présentation et la revue des projets par les pairs permettront de consolider les compétences acquises et de partager les meilleures pratiques.

## SESSION DE CLOTURE : SYNTHESE ET Q&R

Récapitulatif des concepts avancés en Solidity et des meilleures pratiques de développement de dApps. Les formateurs guideront une discussion sur les prochaines étapes pour devenir un expert en développement blockchain. La session de questions-réponses offrira aux participants l'occasion d'approfondir leurs connaissances et de discuter des tendances émergentes dans l'écosystème

blockchain.