Programme de formation

Travailler sur du code legacy

Cette formation vise l'appropriation des techniques et outils nécessaires pour améliorer la qualité et la maintenabilité d'un code existant. En abordant les défis liés au maintien du code legacy, les causes de la complexité, les principes SOLID, les techniques de refactoring ou encore les design patterns, cette formation offre une vue d'ensemble complète de la gestion d'un code legacy. Les participants auront l'occasion de mettre en pratique les connaissances acquises grâce à des exercices et des mises en situation sur des exemples de code réels. Les échanges en groupe et les feedbacks permettront aux participants de s'entraider et de s'améliorer mutuellement.

Prérequis
Savoir écrire des tests unitaires et avoir des notions de Clean Code.

Durée
2 jours

Public
Architectes-techniques, developpeurs

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 (variable selon la formation suivie).
- 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 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.

Programme de formation

Introduction à la formation Code Legacy
Définitions : code legacy, dette technique, clean code...
Refactoring tactique vs refactoring stratégique
Les défis liés au maintien du code : complexité, risque de bugs, évolutivité...
Les conséquences sur l'équipe de développement et sur l'entreprise

Les causes de la complexité
Les causes de la complexité : code dupliqué, état partagé, boucles imbriquées, etc.
Les conséquences de la complexité sur la qualité et la maintenabilité du code
Les techniques pour gérer la complexité : décomposition en sous-problèmes, encapsulation, découpage en modules, etc.
Les outils pour mesurer la complexité : Cyclomatic complexity, Halstead complexity, etc.

Principes SOLID
Les principes SOLID : Single Responsibility - Open/Closed - Liskov Substitution - Interface Segregation - Dependency Inversion
Comment appliquer les principes SOLID pour améliorer la qualité et la maintenabilité du code

Analyse de code et techniques de refactoring tactique
Les différents types de "code smells" : Duplicate code, Long Method, Feature Envy...
Comment identifier les "code smells" dans le code : la complémentarité des outils et de l'oeil humain sur la détection des odeurs
Outils : SonarQube, PMD, FindBugs... Critères de sélection
Les techniques de refactoring : Extract Method, Rename Method, Replace Conditional with Polymorphism, etc.
Quand et comment utiliser chaque technique
Les bonnes pratiques pour refactorer efficacement
Tests de non-regression (acceptation, intégration...)

Entraînement pratique
Le refactoring stratégique
Les design patterns pertinents pour briser les dépendances (Adapter, Strategy, Command...)
Pratique en groupe avec un code Legacy réel
Discussion sur les défis rencontrés et les solutions trouvées

Conclusion et synthèse de la formation
Conclusion de la formation
Questions-réponses
Questionnaires de satisfaction