SQLI INSTITUT

400 FORMATIONS
POUR FAIRE LA DIFFÉRENCE

INGÉNIERIE LOGICIELLE AGILE

CODE STAGE : SC006

OBJECTIFS

Donner un éclairage sur les bonnes pratiques de développement logiciel Faire prendre conscience au collaborateur qu'il est responsable de sa partie de production et qu'il doit mettre en oeuvre tous les moyens pour la réaliser. Favoriser une équipe auto-or

DURÉE

4 jours

PUBLIC

Développeurs
Architectes
Testeurs
Futurs Managers Agiles (Futurs Scrum masters)
Responsables qualité/méthodes

PRÉ-REQUIS

Avoir suivi le cours P-AGI ou avoir une culture agile. Connaître le mode projet et les difficultés rencontrées. Connaître un langage de programmation (notamment pour la mise en oeuvre du Coding Dojo).

PROGRAMME

Jour 1 : Scrum

Module 1 : Transparence, introspection, adaptation et leadership
La science de Scrum
Atelier : Optimiser sa production
Scrum dans la pratique
Atelier : ScrumGame
Création et gestion du « Product Backlog »
Atelier : Le Product Backlog

Jour 2 : Ecrire les User Stories et leurs tests de recette. Pratiques collaboratives

Module 2 : Mise en situation pratique
Atelier : Création d'un projet
Atelier : Remanier les User Stories

Module 3 : Collaboration
Travailler ensemble comme une seule équipe
Inclure le client (Product Owner) dans le processus
Atelier : Réunion quotidienne d'enfer
Définition of « Done »
Rétrospective
Principe du Pair Programming
Autre mode de collaboration

Jour 3 : Immersion

Cette partie est réalisée dans un mode en immersion. Tout le contenu est réalisé en travaillant sur ordinateur.

Module 4 : Architecture et Conception
Principes d'architecture dans un environnement Agile
Pratique de Conception dans une équipe Agile
Principes qui permettent d'amplifier facilement la testabilité et le Refactoring

Module 5 : Test Driven Development (TDD)
Etude du développement à base de « Test-First » incluant les concepts suivants :
Définition et Principes
Théorie et xUnit
Les 3A
Gérer les exceptions

Jour 4 : Refactoring

Module 6 : Qu'est-ce que le Refactoring ?
Conception émergeante
Test Driven Design
Quand faut-il refactorer ?
Comment éviter les grandes dettes techniques

Refactoring pour la maintenance

Module 7 : Les meilleures pratiques de Développement Agile
Conditions Limites
TDD et Gestion de base de données SGBDR
TDD et Gestion des données liées aux fichiers et aux repository (SVN, Git, Sourcesafe)
TDD et Gestion des IHM

Module 8 : Techniques avancées avec le TDD
Corriger des anomalies
Gérer la montée en charge et la sécurité des produits NTIC
Gestion de la sécurité logicielle
Gestion de la performance
Stress tests

Module 9 : Les objects Mock
Mock, Stub et Fake
Application de la théorie sans utiliser de bibliothèque
Découverte des bibliothèques du marché

Module 10 : Self-Test et Outils collaboratifs
Intégration Continue (SVN, Git, Sourcesafe)
Intégrateur Continu (Hudson, Cruisecontrol)
Qualimétrie (Sonar)
Test Driven Requirement avec Fitnesse
Behaviour Driven Development (Cucumber)

Module 11 : Conclusion