Qualité des applications

3 j (21 heures) Ref : DEQL

Public

Architectes, développeurs, analystes, chefs de projets…

Pré-requis

Niveau : Disposer d'une première expérience de développement et maîtriser un langage de programmation (C#, Java ou C++)
Techniques (formations en classe virtuelle) : Vous devez disposer d'un ordinateur connecté à internet, d'un micro et d'une caméra

Moyens pédagogiques

Modalité : Formation présentielle ou Formation distancielle (classe virtuelle) - Inter / Intra - Groupes de 4 à 12 stagiaires
Méthodes : Présentation des concepts, discussion technique, démonstrations, exercices et TP
Matériel :
Présentiel : Un poste informatique par stagiaire connecté à internet, à une imprimante en réseau et au réseau informatique,
Les salles sont équipées d'un tableau interactif ou d'un vidéoprojecteur et d'un paperboard
Distanciel : Aelion met à disposition de chaque stagiaire
- Un PC équipé des outils et logiciels nécessaires à la formation auquel vous accédez via un outil de prise en main à distance
- Un accès à un outil de classe virtuelle (Meet)
Support de formation : Un support de formation sera remis à chaque stagiaire en fin de formation : plateforme collaborative intégrant le code source des exercices réalisés en formation, webographie, mémos

Modalités de suivi et d'évaluation

Questionnaire d'évaluation des pré-requis, suivi des connaissances tout au long de la formation, évaluation des acquis en fin de formation
Questionnaire d'évaluation de la satisfaction en fin de stage, feuille de présence émargée par demi-journée par les stagiaires et le formateur, attestation de fin de formation

Appliquée au développement, la qualité logicielle vise à assurer un niveau optimal en termes de stabilité, de maintenabilité, de scalabilité ou encore à fournir des métriques pertinentes durant toutes les phases du projet. Avoir connaissance des bonnes pratiques de développement et des anti-patterns est un minimum mais ne garantit pas que ces guidelines ont bien été appliquées par toutes et tous. La mise en place d'une stratégie d'intégration continue, l'automatisation des tests, l'analyse statique du code et la couverture fonctionnelle sont autant de méthodes qui permettent d'apporter au processus de développement une couche d'agilité, et induisent des cycles de livraison plus court et mieux maîtrisés. Cette formation permet d'introduire les bonnes pratiques à appliquer sur toutes les phases d'un projet informatique, mais aussi de sensibiliser les développeurs aux problématiques de maintenance et de montée en charge qui sortent habituellement de leur périmètre d'intervention, mais qui sont pourtant largement impactées par leurs activités de génie logiciel. A l'issue de la formation, vous serez capable de développer une application informatique de qualité en utilisant les méthodes et outils de bonnes pratiques.

Action collective OPCO ATLAS - Inscription CampusAtlas

Objectifs

Décrire les bonnes pratiques d’écriture d’un code incluant la maintenance de l’application
Utiliser GIT pour gérer les codes sources
Identifier les outils nécessaires à la fabrique logicielle pour produire des livrables de qualité
Identifier l'offre des outils de tests de performance et de charge
Utiliser les méthodes puis outils pour les tests
Appréhender les outils et phases de mise en oeuvre d’une intégration continue

Programme détaillé

DECRIRE LES BONNES PRATIQUES D’ECRITURE D’UN CODE INCLUANT LA MAINTENANCE DE L’APPLICATION

Cycle de vie complet d'une application (ALM et principes DEVOPS avec CI/CD)
Bonnes pratiques pour l’écriture et la structure d’un code source
Méthodes et outils de Linting
Méthodes et outils de documentation
Méthodes et outils d’organisation du code (structurer ses dossiers, espaces de noms)
Techniques de mutualisation de code
Utiliser et créer des librairies
Gestion des dépendances
Origine des défauts logiciels
Sensibilisation au coût d'un programme non testé Indicateur

UTILISER GIT POUR GERER LES CODES SOURCES

GIT pour gérer les codes sources
Installation et configuration
Principes de GIT
Approche décentralisée
Branches et commit
Les états Clear/stage/commit
Guide pratique de git
Récupérer un projet
Créer un commit
Gestion des branches
Branches locales et remotes
Merge et rebase
Flux de travail collaboratif avec Git (exemple du Gitflow workflow).

IDENTIFIER ET UTILISER LES OUTILS NECESSAIRES A LA FABRIQUE LOGICIELLE POUR PRODUIRE DES LIVRABLES DE QUALITE

Les outils des forges logicielles: ISSUES, WIKI, ETC.

Flux de travail collaboratif avec Git (exemple du Gitflow worklow)

IDENTIFIER L'OFFRE DES OUTILS DE TESTS DE PERFORMANCE ET DE CHARGE

Les tests en agile ou cycle en V (TDD, BDD, ATDD)

Niveaux de tests : composants, intégration, système

Cible des tests : fonctionnels / non fonctionnels, architecture logicielle, non régression

UTILISER LES METHODES PUIS OUTILS POUR LES TESTS UNITAIRE, LES TESTS QUALITE, LES TESTS DE PERFORMANCE ET DE CHARGE

Ecriture de tests unitaires dans un langage de programmation

Présentation d’une librairie de tests

Ecriture d’un ou plusieurs tests unitaires

Utilisation de doublures (Mocks, Stubs, Spy, Fakes...)

Exécution des tests et génération de rapports

Tests qualités avec Sonar

Présentation de Sonar (fonctionnement, règles qualités et profils qualités)

Branchement d’un projet sur Sonar

Exécution des tests Sonar

Tests de performance et de charge avec Gatling

Présentation de Gatling

Branchement d’un projet sur Gatling

Exécution des tests Gatling

APPREHENDER LES OUTILS ET PHASES DE MISE EN œUVRE D’UNE INTEGRATION CONTINUE

Introduction a la CI avec GITLAB CI

Rappels sur les principes de CI/CD et Devops

Intégration Continue avec Gitlab CI

Configuration Gitlab CI/CD

Pipelines de CI/CD

Gestion des runners

Déploiement continu et Gitlab Runner

Mise en pratique des outils Gitlab ci

Définition d’un Pipeline Gitlab CI

Les tests dans la pipeline (Ajout des tests automatisés dans la pipeline)

Outils externes qui peuvent compléter Gitlab (Tests qualité avec Sonar, etc.)

Rappels sur les principes de CI/CD et Devops

Déploiement Continu avec GitLab CI

Dépôt de packages (Mise en place registre docker, déploiement d’image dans la pipeline)

Déploiement d’une MR dans un environnement de recette, déploiement en staging

Mise en pratique des outils gitlab ci

Définition d’un Pipeline GitLab CI/CD

Les déploiements dans la pipeline (Ajout d’un déploiement)

Outils externes qui peuvent compléter GitLab (Aperçu de Ansible)
