WealCome
Expertise en conception logicielle
BDD / TDD / DDD / Clean Architecture

PROGRAMME DE FORMATION

Document généré le 08/11/2026

TDD, DDD et Clean Architecture dans le monde Java

Type d'action : **Action de formation** Durée totale : **14h**

Informations de session

Lieu : Visio par Zoom

Formateur : Michaël AZERHAD - Président de S.A.S.U WealCome et architecte / expert technique depuis plus de 20 ans

Dates et horaires

Date	Début	Fin	Durée
07/12/2026	09:15	12:15	3h
07/12/2026	13:00	17:00	4h
08/12/2026	09:15	12:15	3h
08/12/2026	13:00	17:00	4h

Description

Les projets logiciels — backend comme frontend — souffrent souvent d'une complexité accidentelle qui apparaît quelques mois après leur démarrage.

Cette complexité non anticipée entraîne :

- un ralentissement massif du développement
- une peur de modifier le code existant
- des corrections par workarounds
- une perte de testabilité
- une perte d'intention métier dans le code
- une architecture rigide et difficile à faire évoluer

Cette formation montre comment prévenir cette dérive grâce à trois pratiques essentielles :

- Test-Driven Development (TDD)
- Domain-Driven Design (DDD)
- Clean / Hexagonal Architecture

Réalisation complète d'un backend Java réaliste, en live coding avancé, basé sur un cas métier digne de ce que l'on rencontre en entreprise — loin des katas simplistes.

Le groupe peut choisir le taux d'utilisation des technologies IA comme Claude Code : 0% - 20% - 50% - 100%

Technologies :

- Java 25+
- Spring-Boot / Rest APIs
- Hibernate/JPA
- PostgreSQL
- JUnit 5 / AssertJ
- TestContainers (Docker).

Objectifs de la formation

- Expliquer les principes fondamentaux du Test-Driven Development (TDD), du Domain-Driven Design (DDD) et de la Clean / Hexagonal Architecture dans un environnement Java / Spring-Boot
- Identifier les causes principales de la complexité accidentelle dans un projet logiciel et mettre en œuvre des pratiques permettant de la prévenir.
- Appliquer les patterns stratégiques et tactiques clés du Domain-Driven Design (DDD), voire de CQRS pour modéliser un domaine métier réaliste.
- Concevoir et développer un backend Java from scratch en suivant une méthodologie professionnelle basée sur le TDD Outside-In.
- Structurer une application en Clean Architecture avec séparation claire entre cas d'usage, modèles métier, ports et adaptateurs.
- Mettre en place une stratégie de tests complète combinant tests unitaires, d'acceptation, d'intégration et end-to-end.
- Analyser et comparer les différents types de tests afin de comprendre leur rôle dans la conception logicielle et la robustesse du système.
- Refactorer du code existant pour améliorer sa lisibilité, sa testabilité et son expressivité métier.
- Remettre en question les croyances courantes liées au TDD, au DDD et à la Clean Architecture à travers des exercices et échanges interactifs.
- Être capable de transformer un backend legacy en architecture hexagonale testable.
- Utiliser un assistant IA (Claude Code) pour produire du code conforme aux principes du Software Craftsmanship (TDD, DDD, Clean Architecture) et vérifier sa qualité par les tests.

Prérequis

- Bonne maîtrise de Java ou d'un langage similaire comme C#
- Notions aux frameworks, Spring-Boot / Spring-IOC / Hibernate précisément sont un plus
- Bonnes connaissances en programmation orientée objet
- Capacité à écrire un simple test unitaire avec JUnit

Public visé

Particuliers et professionnels :

- CTO / Technical Leader
- Développeur Backend
- Développeur Full Stack
- Architecte technique

Méthodes pédagogiques

- Des apports théoriques sur le processus
- Des exemples concrets
- Des démonstrations complètes par le formateur en live coding
- Exercices réalisés en live par les participants afin de s'exercer sur cette application d'entreprise.
- Challenges proposés quant au TDD, Clean Archi et au refactoring de code (modification de structure du code)

La formation inclut une démonstration guidée de Claude Code, reposant sur des agents et skills conçus par le formateur, afin d'illustrer une pratique assistée par IA alignée avec TDD, DDD et Clean/Hexagonal Architecture.

Les participants observent puis reproduisent les étapes (prompting, itérations, validation par tests, refactoring), sur un cas métier réaliste.

Capacités développées

- Concevoir un use case métier en Java en appliquant une démarche TDD Outside-In.
- Écrire des tests exprimant clairement des règles de gestion métier.
- Utiliser le refactoring pour faire émerger une architecture propre et lisible.
- Identifier les causes de la complexité accidentelle et appliquer des pratiques pour la prévenir.
- Déterminer des Bounded Contexts plus facilement.
- Modéliser un Bounded Context avec les concepts clés du Domain-Driven Design (Aggregates, Value Objects, Domain Events (Outbox Pattern), CQRS lorsque pertinent).
- Structurer une application en Clean / Hexagonal Architecture avec séparation claire entre domaine, cas d'usage, ports et adaptateurs.
- Développer un backend Java / Spring-Boot réaliste intégrant API REST, persistance et logique métier dans une architecture testable.
- Mettre en place une stratégie de tests combinant tests unitaires/ d'acceptation, d'intégration et end-to-end avec JUnit et TestContainers.
- Refactorer du code existant pour améliorer sa lisibilité, sa testabilité et son expressivité métier.
- Refactorer les tests de sorte à améliorer la lisibilité, la concision et donc la productivité.
- Analyser la qualité d'une architecture existante, code possiblement legacy, et proposer des améliorations concrètes.
- Argumenter des choix de conception logicielle auprès d'une équipe technique.
- Identifier et remettre en question les fausses croyances courantes liées au TDD, au DDD et à la Clean Architecture.

Modalités d'évaluation

- QCM en fin de formation
- Exercices de code avec Pull Requests
- Échanges et discussions

Documents et supports pour les apprenants

- Supports de cours en markdown très complets
- Claude Code : Agents et Skills personnels du formateur optimisés pour du TDD efficace en autonomie
- Projet Git du live coding avec X branches
- Partage de ressources (articles/blogs/discussions de forums Tech)
- Discussion privées sur le canal Slack dédié au mini-groupe post-formation

Déroulé de la formation

Jour 1

- Tour de table initial pour partager les attentes, objectifs personnels et niveaux des participants
- Questions ciblées sur TDD, Domain-Driven Design et Clean Architecture afin d'ajuster la direction pédagogique au niveau réel du groupe
- Présentation du projet fil rouge : backend Java / Spring-Boot réaliste inspiré d'un cas métier rencontré en entreprise ambitieuse (pas de kata simpliste)
- Atelier BDD (Behaviour-Driven Development) et/ou Event Storming pour construire une vision commune du domaine métier et des attentes fonctionnelles
- Identification des règles de gestion, scénarios principaux et invariants métier
- Introduction au NoEstimates et à l'importance de travailler en itérations très courtes (« minimum learnable »)
- Démarrage du live coding en Test-Driven Development en approche outside-in sur l'architecture hexagonale, en partant des ports primaires, avec un style majoritairement classicist adapté aux cas métier réels
- Explication de la bonne utilisation des mocks et distinction entre solitary unit tests et sociable unit tests afin de comprendre leur rôle dans la conception logicielle et la testabilité
- Démonstration de Mutation Testing avec PiTest pour assurer la pertinence du TDD
- Illustration en live coding de l'impact des choix de tests sur l'émergence du design et la qualité de l'architecture
- Introduction progressive des notions de Domain-Driven Design : Aggregates, Value Objects, Bounded Contexts, invariants métier, et approche CQRS lorsque pertinent
- Mise en évidence de l'émergence naturelle de la Clean / Hexagonal Architecture par la pratique rigoureuse du TDD et l'inversion de dépendances
- Approfondissement du projet avec des cas métier plus complexes et refactoring d'architectures naïves ou legacy vers des architectures robustes et évolutives
- Introduction aux Domain Events et à la gestion de la complexité accidentelle dans les projets backend et frontend
- Démonstration de Claude Code avec agents et skills conçus par Michaël AZERHAD afin d'utiliser l'IA tout en respectant les principes du Software Craftsmanship
- Phases de mob programming, questions ouvertes et revue de code pour ancrer les apprentissages
- Évaluation des acquis par échanges, revue de code et auto-évaluation des capacités développées

Jour 2

- Récapitulatif du travail accompli la veille pour garantir la compréhension de tous les stagiaires
- Mise en place de l'infrastructure autour du projet : adaptateurs primaires et secondaires avec Spring-Boot, Hibernate (JPA), et éventuellement message broker avec Outbox Pattern si le niveau du groupe le permet
- Démonstration concrète de l'inversion de dépendances avec Spring-Boot et intégration dans une architecture hexagonale
- Création d'un second Bounded Context avec un Use Case drivé en TDD et communication avec le premier par Domain Events
- Mise en place d'un journey test pour valider la boucle complète inter-bounded contexts
- Mise en évidence de la séparation entre Domain Model et Persistent Model grâce au Snapshot Pattern, afin d'éviter les dépendances accidentelles entre logique métier et infrastructure
- Illustration en live coding de la transformation d'un aggregate en snapshot persistant et reconstruction du modèle métier en conservant les invariants
- Continuité des exercices avec refactoring de code legacy simulé ou amélioration d'un code existant
- Démonstration de Claude Code avec agents et skills conçus par Michaël AZERHAD afin d'utiliser l'IA pour l'infrastructure et les tests end-to-end, d'intégration et de journey tout en respectant les principes du Software Craftsmanship
- Phases de mob programming, questions ouvertes et revue de code pour ancrer les apprentissages
- Évaluation des acquis par échanges, revue de code et auto-évaluation des capacités développées
- Mise en place du canal Slack privé WealCome pour assurer un suivi post-formation de chaque stagiaire et accompagner la mise en pratique sur des projets réels

Modalité d'accès

Contactez-nous par mail : contact@wealcomecompany.com

Délais d'accès

Nous pouvons programmer les formations en fonction de vos contraintes et de nos disponibilités.

Moyens d'accompagnement et d'assistance (Formation à distance)

En cas de difficulté d'accès à la plateforme de visioconférence ou pour toute question technique ou pédagogique pendant la session, le stagiaire dispose d'un point de contact unique pour obtenir une assistance immédiate :

Contact d'assistance technique et pédagogique : WealCome

E-mail : contact@wealcomecompany.com

Accessibilité handicap

Nos formations sont accessibles aux personnes en situation de handicap.

Les aspects, l'accessibilité et le type de handicap au regard des modalités pédagogiques sont à évoquer impérativement au cours de l'entretien préalable à toute contractualisation afin de pouvoir orienter ou accompagner au mieux les personnes en situation de handicap.