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

PROGRAMME DE FORMATION

Document généré le 03/11/2026

TDD et Clean Architecture dans le monde Swift

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
25/11/2026	09:15	12:15	3h
25/11/2026	13:00	17:00	4h
26/11/2026	09:15	12:15	3h
26/11/2026	13:00	17:00	4h

Description

Les applications iOS, de l'interface aux couches métier, 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, avec une logique métier enfouie dans les Views et des ViewModels obèses
- une perte d'intention métier dans le code
- une architecture rigide, couplée aux frameworks Apple et difficile à faire évoluer

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

- Test-Driven Development (TDD)
- Clean / Hexagonal Architecture

Réalisation complète d'une application iOS réaliste, en live coding avancé, basée 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 :

- Swift 6 (async/await, strict concurrency)
- SwiftUI
- Swift Testing (tests core et présentation)
- XCUITest (tests UI end-to-end)
- Swift Package Manager (modularisation par couche et par fonctionnalité)
- URLSession / API REST
- SwiftData ou Core Data
- Xcode
- Claude Code

Objectifs de la formation

Objectifs pédagogiques

- Expliquer les principes fondamentaux du Test-Driven Development (TDD) et de la Clean / Hexagonal Architecture dans un environnement Swift / iOS.
- Identifier les causes principales de la complexité accidentelle dans un projet logiciel et mettre en œuvre des pratiques permettant de la prévenir.
- Concevoir et développer une application iOS en Swift 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, ports, adaptateurs et couche de présentation SwiftUI.
- Exprimer clairement l'intention métier dans le code à travers les cas d'usage, le nommage et les modèles de présentation.
- Mettre en place une stratégie de tests complète combinant tests unitaires, d'acceptation, d'intégration et tests UI end-to-end avec Swift Testing et XCUITest.
- 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 et à la Clean Architecture à travers des exercices et échanges interactifs.
- Être capable de transformer une application iOS legacy, où la logique métier est enfouie dans les Views et les ViewModels, en architecture hexagonale testable.
- Utiliser un assistant IA (Claude Code) pour produire du code conforme aux principes du Software Craftsmanship (TDD, Clean Architecture), assurer la qualité de la conception par un jeu itératif de tests rapides.

Prérequis

- Bonne maîtrise de Swift
- Maîtrise de SwiftUI ou UIKit est un plus
- Bonnes connaissances en programmation orientée objet et en programmation orientée protocoles
- Capacité à écrire des tests avec XCTest ou Swift Testing

Public visé

Particuliers et professionnels :

- CTO / Technical Leader
- Développeur iOS / Swift
- Développeur mobile
- Architecte mobile

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 et Clean/Hexagonal Architecture.

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

Capacités développées

Capacités développées

- Concevoir un cas d'usage en Swift 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.
- Isoler les dépendances techniques (réseau, persistance, horloge, frameworks Apple) derrière des ports pour garder le cœur de l'application testable.
- Modéliser l'état des écrans et la logique de présentation dans des ViewModels testables, découplés de SwiftUI.
- Structurer une application en Clean / Hexagonal Architecture avec séparation claire entre modèles, cas d'usage, ports, adaptateurs et présentation.
- Développer une application iOS réaliste intégrant API REST, persistance locale et logique métier dans une architecture testable.
- Mettre en place une stratégie de tests combinant tests unitaires / d'acceptation, d'intégration et UI end-to-end avec Swift Testing et XCUITest.
- 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 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 le TDD et la Clean Architecture afin d'ajuster la direction pédagogique au niveau réel du groupe
- Présentation du projet fil rouge : application iOS Swift réaliste inspirée d'un cas métier rencontré en entreprise ambitieuse (pas de kata simpliste)
- Atelier BDD (Behaviour-Driven Development) et/ou Example Mapping pour construire une vision commune des attentes fonctionnelles
- Identification des règles de gestion, des scénarios principaux et des comportements attendus de l'interface
- 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 Muter 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
- Modélisation de l'état des écrans et de la logique de présentation dans des ViewModels testables, découplés de SwiftUI
- Isolation des dépendances techniques (réseau, persistance, horloge, frameworks Apple) derrière des ports
- 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 (logique enfouie dans les Views et les ViewModels) vers des architectures robustes et évolutives
- Gestion de la concurrence avec Swift Concurrency (async/await, actors, @MainActor) dans une architecture testable
- 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

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 (Views SwiftUI) et adaptateurs secondaires (client HTTP avec URLSession, persistance avec SwiftData ou Core Data)
- Démonstration concrète de l'inversion de dépendances avec une composition root et l'injection de dépendances dans SwiftUI, intégrées dans une architecture hexagonale
- Continuité du live coding avec de nouvelles fonctionnalités
- Mise en place de tests UI end-to-end avec XCUITest pour valider un parcours utilisateur complet, avec des adaptateurs in-memory pour garantir des tests fiables et rapides
- Mise en évidence de la séparation entre modèles de l'application, DTOs réseau et modèles de persistance, afin d'éviter que le schéma de l'API ou de la base ne contamine la logique de l'application
- Illustration en live coding du mapping entre ces modèles en conservant les règles de gestion
- 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 d'intégration et UI end-to-end, 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.