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

PROGRAMME DE FORMATION

Document généré le 27/01/2026

TDD, DDD et Clean Architecture dans le monde Python

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

Informations de session

Lieu : Visio par Zoom

Formateur : Michaël AZERHAD

Dates et horaires

Date	Début	Fin	Durée
09/02/2026	09:15	12:15	3h
09/02/2026	13:00	17:00	4h
10/02/2026	09:15	12:15	3h
10/02/2026	13:00	17:00	4h

Objectifs de la formation

- Sensibilisation à la discipline TDD et à la Clean Architecture dans le monde Python afin de prévenir la complexité accidentelle, y compris l'introduction aux notions principales du DDD (Domain-Driven Design) et de CQRS.
- Montée en compétences au TDD et à la Clean Architecture dans le monde Python à travers un live coding très professionnel et approfondi, à vocation de démonstration et de mises en situation pour les exercices.
- Il s'agira de réaliser from scratch une application digne de ce qu'on attend en entreprise.
- Compréhension et démonstration des différents types de tests (unitaire / acceptation / end-to-end / intégration).
- Clarification de chacun des concepts maîtres et annexes à travers des séances de questions-réponses et autres démonstrations concrètes.

Description

Les backends comme les frontends souffrent d'une complexité accidentelle dans la plupart des projets. Une complexité accidentelle est une complexité additionnelle malvenue qui aurait pu être évitée. Quand s'observe-t-elle réellement ? Quelques mois après le début du projet. Pourquoi donc le terme "accidentelle" et non le terme "additionnelle" ? Car bien souvent, elle survient par surprise, sans anticipation aucune. Surtout lorsque le développeur n'est pas aguerri sur les concepts théoriques relatifs à la conception logicielle. Quelles en sont ses conséquences :

- Un temps de développement considérablement augmenté pour la moindre fonctionnalité normalement simple.
- Une peur extrême de changer le code existant, de peur d'y engendrer des impacts non perçus ni maîtrisés.
- Par cette peur de changer/casser, chaque bug remonté est corrigé avec des contournements (workarounds) ; autrement dit des « pansements » indirects proches de la supercherie.
- Une expressivité du code réduite à néant, par à-coups de ces contournements et du stress engendré par la soumission face au code existant.
- Un code qui devient de moins en moins testable, par des raccourcis contraires aux bonnes pratiques d'architecture et de conception logicielle, souvent passés inaperçus, sans prise de conscience.
- Des sessions fatigantes de débogage à outrance devant ce code jugé farfelu, non expressif et souvent bancal.
- Une application devenue très rapidement inflexible, dont même la moindre évolution technologique, telle une mise à jour de frameworks, devient le signe d'un besoin de refonte globale …

Est-ce une fatalité ? Devant le nombre de projets dans ce cas-là, il faut croire que oui. Mais il existe heureusement des pratiques logicielles qui redonnent de l'espoir et qui atténuent drastiquement cette complexité accidentelle, si bien comprises et bien menées. Parmi ces pratiques, je mettrai l'accent sur deux d'entre-elles qui me paraissent totalement cruciales et qui ont changé mon quotidien dès 2011 :

- Le Test-Driven Development (TDD)
- La Clean Architecture (cf. Architecture Hexagonale ou Ports/Adapters).

Dans cette formation, je vous démontrerai comment réaliser un backend from scratch avec ces pratiques dans un live coding avancé et fluide, sans manquer de vous exposer une bonne stratégie de tests combinant tests d'acceptation, tests unitaires, tests d'intégration et tests end-to-end. Mon premier souhait : que vous compreniez bien que le TDD n'est pas une technique de test mais de codage permettant une amélioration notable de vos algorithmes et de vos designs. Le sujet sera un projet Python avec FastAPI, bien plus proche de ce qu'on attend de nous en entreprise qu'un simple kata de tri de nombres. Technologies :

- Python
- FastAPI (REST APIs)
- SQLAlchemy (ORM)
- PostgreSQL
- Pytest (tests unitaires, intégration)
- TestContainers (Docker pour tests)

Ce sera interactif avec des exercices sur le chemin, des échanges de questions/réponses au tac au tac, et surtout une bonne ambiance, à la fois professionnelle et détendue. Ayant l'habitude d'enseigner sur les sujets Craft, j'ai acquis une pédagogie qui vous plaira et qui ne laissera personne sur le carreau. Aussi, j'assurerai un suivi sous forme de réponses à vos nouvelles

questions post-formation de sorte à ce que chacun d'entre vous évolue et progresse sans blocage et dans la bonne direction au quotidien. Pour finir, vous constaterez pour beaucoup que quasiment tout ce que vous pensiez au sujet de ces pratiques est en réalité une fausse croyance ... Je n'ai plus qu'à vous souhaiter la bienvenue dans ce noble monde du développement logiciel professionnel. Les sessions regroupent entre 3 et 11 personnes, afin de garder une haute qualité d'interaction.

Prérequis

- Bonne maîtrise de Python
- Notions du framework FastAPI et du concept d'injection de dépendances
- Bonnes connaissances en Programmation orientée objet
- Capacité à écrire un simple test unitaire avec Pytest

Public visé

Particuliers et professionnels :

- 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 sur une application type entreprise
- Challenges proposés quant au TDD, DDD, la Clean Architecture et refactoring de code

Déroulé de la formation

Jour 1

- Rapide tour de table, présentation de chacun et exposition des attentes
- Introduction et cours théorique sur le TDD, cassant les énormes quiproquos à son sujet
- Si le groupe est vraiment novice en TDD, kata éventuel et judicieusement choisi de mises en pratique avec Python et Pytest
- Introduction et cours théorique sur la Clean Architecture
- Démarrage d'écriture d'une application "from scratch" digne d'un cas réel d'entreprise en TDD (User Story avec plusieurs règles de gestion) tout en respectant la Clean Architecture dans le monde Python.
- Approche agile avec le mindset NoEstimates initiée par un mini atelier BDD.
- otions DDD principales expliquées (Aggregates, Value Objects, Repositories, Factories (au sens DDD), Bounded Contexts, Domain Events).
- Approche CQRS (séparation lecture et écriture).
- Séances de questions/réponses tout au long

Jour 2

- Suite du live coding de l'application "from scratch" digne d'un cas réel d'entreprise
- Séances de refactoring continu du code ET des tests de l'application, rendues plaisantes et sans crainte grâce au TDD
- Utilisation de Mutation Testing avec Mutmut pour prouver que le code coverage à lui tout seul est dangereux, et pouvant guider à merveille le refactoring d'un code Legacy.
- Clarification de concepts subtils relatifs au TDD et à la Clean Architecture
- Liaison à une base de données PostgreSQL avec SQLAlchemy - démonstration de tests d'intégration
- Utilisation de TestContainers pour assurer un environnement de test reproductible
- Exposition des services réalisés sous forme d'API REST avec FastAPI et tests end-to-end
- Séance de questions/réponses tout au long

WealCome

192 avenue de la Division Leclerc, 95160 Montmorency, France