+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Mis à jour le 28/04/2025

S'inscrire

Formation Architecture Hexagonale et TDD

3 jours (21 heures)

Présentation

L'Hexagonal Architecture n'est pas une architecture à proprement parler, mais des principes d'architectures mis en avant par Alistair en 2005 mais aussi affinés par tout un tas d'autres auteurs dont un principal : Robert C. Martin alias Uncle Bob en 2012.

Ce dernier a intitulé sa variante de l'hexagonale nommée « Clean Architecture » en apportant pas mal de guidelines et de prévention.

Ces principes d'architecture permettent tout un tas de bénéfices :

- Découplage fort entre la partie métier de l'application et la partie infrastructure/technologique, rendant la conception du cerveau de l'application bien plus facile
- Testabilité de la partie métier grandement améliorée ; le réel TDD rendu possible !
- Capacité à remettre à plus tard les choix technologiques d'infrastructure.
- Capacité à changer les technologies sans effort, évitant alors des refontes chronophages et fastidieuses

À l'issue de cette formation Architecture Hexagonale, vous apprendrez à maîtriser les principes essentiels, produire des logiciels organisés et créer des modèles de domaines performants.

Objectifs

- Maîtriser les principes de l'Hexa/Clean Architecture, tels que l'inversion de dépendances
- Savoir démarrer un projet from scratch pratiquant TDD et l'Hexa Architecture avec l'esprit d'"émergence de design du code"
- Sensibilisation aux décisions cruciales d'architecture technique globale
- Savoir manier un TDD orienté comportement dans une Hexagonal Architecture pour une productivité accrue et sans faille
- Savoir intégrer des composants d'infrastructure tels qu'une base de donnée PostgreSQL et des API tierces partenaires sans toucher au coeur de l'appli
- Savoir dissocier la logique métier de l'application du framework Spring-Boot

- Connaître tous les pièges à éviter dans une Hexa/Clean Architecture
- Être pleinement conscient de la différence majeure entre TDD et le simple fait d'écrire des tests
- Savoir ordonner les actions à réaliser lors de l'élaboration d'une Hexagonal Architecture
- Toutes les questions répondues et quiproquos/déformations populaires au sujet des pratiques révélés

Public visé

- Technical Leaders
- Développeurs Backend
- Développeurs Full Stack
- Architectes techniques

Pré-requis

- Maîtrise de Java ou tout autre langage orienté objet
- Notions des concepts principaux de la POO : Interfaces / Classes abstraites / Polymorphisme
- Notion d'écriture de tests avec JUnit 5 et AssertJ

Technologies utilisées

- Java 20
- Maven 3
- Spring-Boot / Rest APIs
- Hibernate/JPA
- PostgreSQL
- JUnit 5 / AssertJ
- TestContainers (Docker)

Programme de notre Formation Architecture hexagonale et TDD

Jour 1 : Les Fondations d'une Architecture Robuste

Les Fondements Architecturaux

- Pourquoi l'Architecture Logicielle est Essentielle : Souligner les enjeux des systèmes complexes et le rôle crucial d'une architecture bien pensée
- Les Qualités d'une Architecture Efficace : Présenter les caractéristiques d'une bonne architecture (maintenabilité, évolutivité, testabilité, lisibilité, etc)
- Les Bénéfices Concrets : Mettre en évidence les avantages tangibles pour le projet et l'équipe (réduction des coûts, time-to-market, qualité, collaboration)

Clean Architecture : Un Modèle de Conception Clair

- SOLID : Les Cinq Principes Clés : Expliquer en détail chaque principe SOLID (Responsabilité Unique, Ouvert/Fermé, Substitution de Liskov, Ségrégation des Interfaces, Inversion des Dépendances) avec des exemples concrets
- Modularisation, Organiser le Code pour la Clarté : Présenter les stratégies de modularisation et leur importance pour la gestion de la complexité et la réutilisation
- Anatomie de la Clean Architecture : Décrire chaque couche (Entities, Use Cases, Interface Adapters, Frameworks & Drivers), leur rôle et les règles de dépendance

Introduction Pratique au Test-Driven Development (TDD)

- Les Tests Unitaires : Votre Filet de Sécurité : Expliquer l'objectif, les caractéristiques et les bonnes pratiques des tests unitaires
- Découpage vertical : Du Fonctionnel au Technique : Introduire l'idée de découper les fonctionnalités en tranches pour faciliter le développement et les tests
- Le Cycle Immuable du TDD : Red, Green, Refactor : Détailler chaque étape du cycle TDD et son importance pour guider la conception
- Développer le Métier Pas à Pas avec TDD : Mettre en pratique le TDD pour implémenter de la logique métier simple, en insistant sur la conception émergente et la qualité du code

Jour 2 : Maîtriser l'Isolation avec l'Architecture Hexagonale

L'Architecture Hexagonale

- Points communs et différences avec la Clean Architecture : Établir les similitudes (séparation des préoccupations, domaine au centre) et les différences (ports/adapters vs couches) entre les deux architectures
- Ports et Adapters : Expliquer en détail le rôle des Ports (contrats) et des Adapters (implémentations pour interagir avec le monde extérieur) Distinguer les Adapters primaires (Driving) et secondaires (Driven)
- Structure d'un Projet Hexagonal : Présenter une structure de projet typique mettant en évidence l'isolation du domaine

Tests d'Intégration et End-to-End

- La Pyramide des Tests : Présenter la pyramide des tests (unitaires, intégration, end-to-end) et l'importance de chaque niveau
- Les Enjeux Cruciaux des Tests d'Intégration et End-to-End : Souligner leur rôle pour valider les interactions entre les composants et le comportement global du système

Gérer la Persistance des Données dans un Contexte Hexagonal

- Les propriétés ACID : Garantir l'Intégrité des Données : Expliquer les concepts d'Atomicité, Cohérence, Isolation et Durabilité dans la gestion des transactions

- Transactions Spring : Orchestrer les opérations : Présenter la gestion des transactions avec Spring et son intégration dans une architecture hexagonale
- Bien comprendre le fonctionnement de base d'Hibernate : Introduire les patterns couramment utilisés avec Hibernate (ORM) pour interagir avec la base de données sans coupler le domaine

Mise en Place d'un Environnement de Développement Isolé

- Gestion des Migrations avec Liquibase : Faire évoluer le Schéma : Atelier pratique sur l'utilisation de Liquibase pour gérer les changements de schéma de base de données de manière collaborative et versionnée
- Simulation d'APIs Externes avec Wiremock : Atelier pratique sur la création de mocks d'APIs externes pour faciliter les tests d'intégration et end-to-end sans dépendre des systèmes réels
- Atelier Pratique : Tests d'Intégration : Mise en œuvre de tests qui vérifient l'interaction entre les différentes parties de l'application (par exemple, les Use Cases et les Adapters de persistance)
- Atelier Pratique : Tests End-to-End : Introduction et mise en œuvre de tests qui simulent un flux utilisateur complet

Jour 3 : Adopter une Architecture Adaptative et Centrée sur le Métier

La Screaming Architecture : Le Code au Service du Métier

- Les Avantages d'une Arborescence Métier : Souligner comment une structure de code qui reflète le domaine facilite la compréhension par les non-développeurs et l'évolution du système
- Les Pièges à Éviter : Mettre en garde contre les excès et les mauvaises interprétations de la Screaming Architecture
- Exemples d'Arborescence : Présenter des exemples concrets de structures de projets orientées métier

Sensibilisation au Domain-Driven Design (DDD)

- DDD : Dompter la Complexité du Métier : Expliquer comment le DDD aide à comprendre et à modéliser un domaine complexe
- Bounded Contexts : Délimiter les responsabilités : Présenter le concept de Bounded Context et son rôle pour isoler les modèles de domaine et gérer les règles métier de manière cohérente
- Améliorer la Gestion des Règles Métier : Discuter de l'impact du DDD sur la clarté et la maintenabilité des règles métier

Architectures Émergentes : L'Évolution Organique

- Le Principe YAGNI : Ne pas spéculer sur l'Avenir : Expliquer l'importance de ne construire que ce qui est nécessaire maintenant
- Laisser l'Architecture Émerger : Discuter de la manière dont une architecture peut évoluer en réponse aux besoins réels et aux retours d'expérience

Construire Ensemble (Mob Programming)

- Mise en Œuvre Pratique : Application concrète de toutes les pratiques, principes et outils vus pendant la formation pour construire une API de gestion de bibliothèque
- Collaboration et Échanges : Encourager le travail en équipe, le partage des connaissances et la résolution collective des défis

Pour aller plus loin

Sociétés concernées

Cette formation s’adresse à la fois aux particuliers ainsi qu’aux entreprises, petites ou grandes, souhaitant former ses équipes à une nouvelle technologie informatique avancée ou bien à acquérir des connaissances métiers spécifiques ou des méthodes modernes.

Positionnement à l'entrée en formation

Le positionnement à l'entrée en formation respecte les critères qualité Qualiopi. Dès son inscription définitive, l'apprenant reçoit un questionnaire d’auto-évaluation nous permettant d’apprécier son niveau estimé sur différents types de technologies, ses attentes et objectifs personnels quant à la formation à venir, dans les limites imposées par le format sélectionné. Ce questionnaire nous permet également d’anticiper certaines difficultés de connexion ou de sécurité interne en entreprise (intraentreprise ou classe virtuelle) qui pourraient être problématiques pour le suivi et le bon déroulement de la session de formation.

Méthodes pédagogiques

Stage Pratique : 60% Pratique, 40% Théorie. Support de la formation distribué au format numérique à tous les participants.

Organisation

Le cours alterne les apports théoriques du formateur soutenus par des exemples et des séances de réflexions, et de travail en groupe.

Validation

À la fin de la session, un questionnaire à choix multiples permet de vérifier l’acquisition correcte des compétences.

Sanction

Une attestation sera remise à chaque stagiaire qui aura suivi la totalité de la formation.