

Mis à jour le 06/11/2025

S'inscrire

Formation Clean Architecture .NET Core

3 jours (21 heures)

Présentation

Créez une application faiblement couplée, à dépendance inversée !

La Clean Architecture est une architecture logicielle destinée à garder le code sous contrôle, sans qu'il soit nécessaire d'y mettre de l'ordre pour éviter que quiconque ne touche à un code après sa publication. Le concept principal de l'architecture propre est que le code/logique de l'application doit être écrit sans aucune dépendance directe.

Modifiez-vous la base de données ou l'interface utilisateur, le cœur du système (règles métier/domaine) ne doit pas être modifié. Cela signifie que les dépendances externes sont complètement remplaçables.

La clean architecture permet de rendre une application indépendante de tout framework, base de données, interface utilisateur. Elle est testable et bien organisée. Cette architecture se compose de 4 catégories distinctes : Domain, Application, Infrastructure et Présentation. Cette architecture est souvent appelée Onion architecture ou architecture hexagonale ou même port & adapter. Le mot clé clean architecture vient d'[Uncle Bob](#).

Grâce à notre formation Clean Architecture en .NET Core, vous serez capable de créer une architecture, afin de développer des APIs testable et robuste qui pourront être facilement compris et reprises par d'autres développeurs.

Dans cette formation, comme dans toutes nos formations que nous vous proposons nous utiliserons la [dernière version stable](#) (.NET 9 avec le nouveau [langage C# 13](#)).

Objectifs

- Vous aider à garder votre application facile à développer, à comprendre et à maintenir
- Structure d'un projet Clean Architecture
- Utilisation de CQRS (Command Query Responsibility Segregation)

- Implémentation des tests unitaires et d'intégration

Public visé

- Développeurs Web et applicatif
- Architectes

Pré-requis

- Connaissance en programmation orientée objet
- Connaissance d'environnement .NET
- [Tester Mes Connaissances](#)

Pré-requis techniques

- Visual Studio 2022 (Community à minima)
- Docker Desktop afin de pouvoir lancer des images (facultatif)

Programme de notre Formation Architecture Clean avec .NET Core

[Jour 1 - Matin]

Introduction à l'architecture : En quoi l'architecture impacte votre activité ?

- Définition de l'architecture logicielle et rôle stratégique dans un projet
- Impact direct sur la maintenabilité, la qualité et la rapidité de livraison
- Coûts cachés d'une mauvaise architecture (dette technique, instabilité, refontes coûteuses)
- Relation entre architecture, agilité et capacité à faire évoluer le produit
- Mini-échange interactif : tour de table des difficultés rencontrées par les participants dans leurs projets

Analyse de dépôts de code "Legacy" en MVC et en N-Tier

- Rappel des principes des architectures MVC et N-Tier
- Identification des points de douleur fréquents :
 - Couplage fort entre couches
 - Multiplication de la logique métier dans les contrôleurs ou dans la base de données
 - Difficultés de tests unitaires et d'évolution

- Lecture critique d'extraits de code "legacy" en .NET (MVC/N-Tier)
- Mise en évidence des symptômes d'une dette technique avancée (code dupliqué, dépendances circulaires, complexité croissante)
- Comparaison avec les objectifs d'une architecture moderne (indépendance, testabilité, évolutivité)
- Atelier pratique : exploration en groupe d'un dépôt simplifié "legacy" et discussion sur ce qui pose problème

[Jour 1 - Après-midi]

Définition du projet "fil rouge", et début du code

- Présentation du projet fil rouge qui servira de support tout au long de la formation
- Clarification des besoins fonctionnels et des règles métier de départ
- Mise en place de la solution .NET (squelettes de projets, dépendances de base)
- Atelier pratique : création collective du squelette du projet et premières entités métier.

Principes de base des tests automatisés

- Pourquoi tester ? Différence entre tests manuels et automatisés
- Les types de tests : unitaires, d'intégration, end-to-end (panorama rapide)
- Les bénéfices sur la qualité, la maintenabilité et la confiance dans le code
- Introduction à xUnit dans l'écosystème .NET
- Règles d'écriture de tests efficaces : AAA (Arrange, Act, Assert), lisibilité, indépendance
- Atelier pratique : écriture des premiers tests unitaires sur le code du projet fil rouge, avec la méthode TDD.

[Jour 2 - Matin]

Présentation des principes SOLID

- Pourquoi SOLID reste une base essentielle pour les architectures modernes
- Détail de chaque principe avec exemples simples en C#
- Antipatterns typiques quand SOLID n'est pas respecté
- Comment détecter si un principe n'est pas respecté ?
- Exercice pratique : identifier les violations de SOLID dans les dépôts vus la veille

Présentation des concepts fondamentaux des architectures "modernes" (Ports & Adapters, Clean Architecture, Architecture Hexagonale)

- Constat des limites des approches MVC / N-Tier vues la veille
- Principes clés : séparation des préoccupations, inversion des dépendances, indépendance du domaine
- Ports & Adapters (Hexagonal Architecture) :
- Ports = interfaces métier
- Adapters = implémentations techniques (DB, API, UI)

- La Clean Architecture : organisation en couches concentriques (Domain, Application, Infrastructure, UI)
- Atelier pratique : schématisation des concepts sur le projet fil rouge

[Jour 2 - Après-midi]

Mise en place d'une architecture "Ports & Adapters" pour le projet fil rouge

- Création des contrats (ports) pour isoler le domaine
- Mise en place des premiers adapters (in-memory, API simulée)
- Intégration du code existant (fil rouge) dans cette nouvelle structure
- Avantages immédiats : testabilité, substitution des implémentations
- Atelier pratique : implémentation d'un use case avec Ports & Adapters dans le projet fil rouge

Évolution de l'architecture vers une "Clean Architecture"

- Structuration du projet en couches concentriques : Domain / Application / Infrastructure / UI
- Règles de dépendances : le domaine ne dépend de rien
- Adaptation progressive du code existant pour respecter la Clean Architecture
- Comparaison avec Ports & Adapters : ce qui change et ce qui reste
- Atelier pratique : migration du projet fil rouge vers une Clean Architecture complète

[Jour 3 - Matin]

Introduction à CQRS (Command Query Responsibility Segregation)

- Rappel du principe de séparation entre lecture et écriture
- Différences entre approche CRUD classique et CQRS
- Bénéfices : simplification des cas d'usage, meilleure évolutivité, alignement avec la Clean Architecture
- Mise en place de handlers (CommandHandler, QueryHandler) dans un projet .NET
- Atelier pratique : implémentation d'un cas d'usage du projet fil rouge avec CQRS

CQRS avancé et bonnes pratiques

- Gestion des DTOs spécifiques pour Command et Query
- Impact sur les tests : isolation et clarté
- Quand CQRS est pertinent... et quand il ne l'est pas
- Atelier pratique : ajout d'une Query et extension du projet fil rouge avec des tests unitaires dédiés

[Jour 2 - Après-midi]

Introduction au BDD (Behavior Driven Development)

- Différences entre TDD et BDD : focus sur le langage métier et la collaboration
- Rôle du BDD dans la validation des règles métiers
- Présentation de Gherkin : langage naturel structuré (Given / When / Then)
- Écriture de scénarios lisibles par les métiers et exécutables par les développeurs
- Outils de support dans l'écosystème .NET (SpecFlow, ReqNRoll, etc.)
- Atelier pratique : rédaction collaborative de scénarios Gherkin pour un use case du projet fil rouge et automatisation de tests d'acceptance avec Gherkin

Mise en œuvre pratique du BDD avec le projet fil rouge

- Transformation des scénarios en tests automatisés exécutables
- Démonstration : un scénario Gherkin qui pilote directement le code .NET
- Retour sur les avantages : communication métier, documentation vivante, qualité accrue
- Limites et pièges à éviter (scénarios trop techniques, duplication avec les tests unitaires)
- Atelier pratique : implémentation d'un scénario Gherkin end-to-end sur le projet fil rouge

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.