.NET : BONNES PRATIQUES ET DESIGN PATTERNS

Nombre de jours -> 4

Participants -> Programmeurs, architectes système et tous ceux qui développent des applications .NET.

Pré-requis -> Une expérience de la programmation

Objectifs -> Ce cours est conçu pour ceux qui souhaitent développer leurs connaissances de base du langage .NET. Les développeurs apprennent à identifier et à résoudre des problèmes courants de conception et d'architecture en appliquant les bonnes pratiques .NET de développement d'applications.

Moyens pédagogiques, techniques et d'encadrement ->

- 1 poste de travail complet par personne
- De nombreux exercices d'application
- Mise en place d'ateliers pratiques
- Remise d'un support de cours
- Passage de certification(s) dans le cadre du **CPF**
- Mise en place de la Charte contrôle et qualité OPCA
- **Notre plateforme d'évaluation** :
  - Evaluation des besoins et objectifs en pré et post formation
  - Evaluation technique des connaissances en pré et post formation
  - Evaluation générale du stage

---

1 - Interfaces simples pour algorithmes complexes

- Unification des interfaces de sous-systèmes pour en faciliter l'emploi et la réutilisation, mise en œuvre du pattern Façade

2 - Variation des fonctionnalités en programmant avec des interfaces

- Créer des abstractions pour optimiser l'adaptabilité et la flexibilité de l'application
- Utilisation du pattern Strategy

3 - Extension dynamique du comportement d'un objet

- Augmenter des fonctionnalités sans impact sur le code
- Composer des objets avec le pattern Decorator

4 - Réutilisation et flexibilité

- Éliminer la duplication de code en factorisant un algorithme
- Utilisation du pattern Template Method

5 - Interfaçage de classes incompatibles

- Transformation d'une interface pour ajouter de la valeur à du code existant, exploitation du pattern Adapter

6 - Automatisation des tests unitaires

- Réduire les cycles de développement via les tests automatisés
- Amélioration de la qualité grâce aux tests
- Supprimer les erreurs via les tests de régression réutilisables

7 - Intégration des tests et de l'écriture du code

- Ecriture des tests en premier
- Génération immédiate de notifications (vert-rouge) pour une meilleure qualité du code et des cycles de développement plus courts
- Organiser, coordonner et effectuer des cas de tests
- Isolation d'un environnement de test de classes avec le Pattern Mock-Object pour des tests fiables et renouvelables
- Refactorisation rythmée par les tests pour une validation immédiate

8 - Refactorisation du code pour améliorer la conception

- Amélioration de la conception grâce à une refactorisation préservant le comportement
- Éliminer la duplication de code en refactorisant avec des patterns

9 - Architecture d'une application multicouches

- Architectures en couches pour faciliter la réutilisation, l'évolutivité et la longévité
- Accès aux données à partir de la couche métier
- Découpler la création des objets grâce au pattern Factory
- Conserver l'identité des objets avec le pattern Identity Map

10 - Programmation des niveaux de l'application

- Isoler l'interface les couches d'UI avec les patterns MVC et MVVM
- Gestion des états d'une application avec le pattern State
- Restructurer des tables d'une BdB sans affecter le code

11 - Modélisation de la couche métier

- Relâchement du lien entre les modèles objets métier et la base de donnés avec le pattern Domain Model
- Mappage entre des objets métier riches et des tables de bases de données grâce au pattern Data Mapper
- Réduction des hiérarchies d'objets métier avec le pattern Inheritance Mapper
- Exploiter la programmation déclarative dans l'Entity Framework de Microsoft pour implémenter des classes correspondantes aux données

12 - Organisation et implémentation de la logique métier

- Gérer les types d'entité ignorant la persistance avec l'Entity Framework
- Manipulation de groupes d'entités avec des classes métier

13 - Appliquer les bons principes de conception des classes

- Ouvert/fermé
- Responsabilité unique
- Substitution de Liskov
- Séparation des interfaces
- L'inversion des dépendances

14 - Simplification du code d'accès aux données

- Diminution du code d'accès aux données grâce aux propriétés de navigation des entités
- Suppression du code de mise à jour de la base avec le suivi des modifications des entités
- Automatisation de la revue de code avec l'outil FxCop
- Réduire le code de l'IU à travers l'exploitation du data binding