MFE-IT

Référence : of-DEV-FS26

# Formation Du C# classique au full-stack .NET / Vue.js : reprendre et faire évoluer une application en équipe

*Passer du client lourd C# au développement web moderne .NET / Vue.js*

**Durée : 5 jours | Heures : 35 h | Tarif : 2 450 € HT**

*En présentiel ou à distance · Groupe de 1 à 3 · Session garantie dès 1 inscrit · ~80 % de pratique*

## DESCRIPTION

Cette formation accompagne un développeur déjà aguerri — typiquement issu du client lourd (Windows Forms, WPF, C# « classique ») — dans son passage vers une architecture web moderne et découplée. Elle ne réapprend pas à programmer : elle réoriente une expertise existante vers l'écosystème .NET actuel et le front-end Vue.js, en partant toujours de l'outil et de la mise en pratique plutôt que de la théorie. Au fil de cinq journées à dominante atelier (environ 80 % de travaux pratiques sur un projet fil rouge), le participant apprend à récupérer le code d'une application, à suivre le trajet complet d'une donnée de l'écran jusqu'à la base, à faire évoluer une fonctionnalité, à la tester, à la déboguer et à la soumettre à la revue de son équipe.

## OBJECTIFS PÉDAGOGIQUES

À l'issue de cette formation, le participant sera capable de :

- Récupérer, modifier et soumettre du code via Git et Azure DevOps en respectant le flux de travail d'une équipe.
- Explorer et comprendre une base de code existante que l'on n'a pas écrite.
- Déboguer une anomalie simultanément côté front-end et côté back-end.
- Écrire un test unitaire pour sécuriser une modification.
- Implémenter une évolution de bout en bout, de l'interface jusqu'à la base de données.
- Interroger les données avec Entity Framework Core et LINQ plutôt qu'en SQL écrit à la main.
- Diagnostiquer un incident d'authentification ou de conteneurisation.
- Concevoir un traitement planifié autonome de surveillance.

## PRÉREQUIS

- Bonne maîtrise du langage C# (classes, interfaces, generics ; la notion d'async/await est un plus).
- Connaissance du SQL et des bases de données relationnelles.
- Simple lecture de HTML / CSS / JavaScript (aucune pratique avancée requise).
- Aucune expérience préalable de Git, des API REST, de Vue.js ou de Docker n'est nécessaire.

## PUBLIC VISÉ

- Développeur expérimenté venant du client lourd et rejoignant une équipe travaillant sur une architecture web découplée.
- Développeur ou responsable technique disposant d'une solide culture C# / SQL mais peu familier des outils collaboratifs modernes (Git, Azure DevOps, revue de code).
- Profil autonome devant s'intégrer à un flux de travail d'équipe encadré.
- Développeur amené à reprendre et faire évoluer une application dont il n'est pas l'auteur.

## PROGRAMME DÉTAILLÉ

La formation est à dominante atelier (environ 80 % de pratique) et s'appuie sur un dépôt de projet dédié servant de fil rouge tout au long des cinq journées.

### Module 1 — Travailler en équipe avec Git et Azure DevOps

Le socle de tout le parcours : sans une pratique solide du versioning collaboratif, aucun des modules suivants n'est exploitable en équipe. On construit d'abord un modèle mental clair de Git (instantanés, zone de préparation, historique local et distant), puis on installe le cycle quotidien : cloner, observer l'état, valider, publier, récupérer. On aborde les branches et la stratégie de l'équipe, la résolution de conflits de fusion, et les moyens d'annuler proprement un travail. Côté plateforme, on découvre les dépôts Azure DevOps, les pull requests, la revue de code et les politiques de branche, ainsi que l'intégration de Git dans Visual Studio et la rédaction de commits lisibles.

**Atelier pratique :** *cloner le dépôt réel, développer une modification sur une branche dédiée, la publier, provoquer puis résoudre un conflit de fusion, et mener une pull request jusqu'à sa validation.*

### Module 2 — Lire, naviguer et déboguer une base de code sous Visual Studio

Comment prendre en main un code que l'on n'a pas écrit, puis le modifier sans rien casser. On apprend à circuler dans une solution inconnue (recherche de symboles, hiérarchie d'appels, recherche de toutes les références, accès à l'implémentation), à exploiter pleinement le débogueur (points d'arrêt conditionnels, fenêtres de surveillance, pile d'appels, exécution immédiate, débogage multi-projets) et à lire l'historique d'un fichier (comparaison de versions, annotation).

**Atelier pratique :** *retrouver le chemin complet d'une fonctionnalité dans le projet réel, puis localiser deux anomalies volontairement introduites à l'aide du débogueur.*

### Module 3 — Fiabiliser ses évolutions par les tests unitaires (xUnit)

Le test unitaire vu comme un filet de sécurité pour modifier sereinement. On clarifie ce que l'on teste et ce que l'on ne teste pas, on écrit des tests avec xUnit selon le schéma préparer / agir / vérifier, et on introduit l'isolation des dépendances via l'injection de dépendances et les objets simulés (notions).

**Atelier pratique :** *sécuriser une modification en écrivant d'abord le test qui échoue, corriger le code, puis constater le passage au vert.*

### Module 4 — Architecture Web API : REST, HTTP, contrats et CQRS

Le véritable changement de paradigme : passer du monolithe à une architecture découplée. On explique pourquoi séparer front-end et back-end et les conséquences concrètes sur le développement, on revoit les fondamentaux HTTP (verbes, codes de statut, en-têtes, corps de requête), puis on décortique un contrôleur ASP.NET Core (routage, liaison de données, sérialisation JSON). On introduit les DTO et l'intérêt de ne jamais exposer directement ses entités, la lecture d'un contrat via Swagger / OpenAPI, et le patron CQRS (séparation des commandes et des requêtes).

**Atelier pratique :** *explorer l'API du projet via Swagger, tracer une requête de bout en bout au débogueur, créer un endpoint complet avec son DTO, puis ajouter une commande et une requête dans la structure existante.*

### Module 5 — Accès aux données avec Entity Framework Core & LINQ

Remplacer le réflexe de la requête SQL écrite à la main par l'approche ORM du projet. On pose le principe d'un ORM (ce que l'on gagne et ce que l'on perd), le contexte de données et le suivi des entités, la gestion des relations et le chargement des données associées. On travaille LINQ (projection, filtrage, jointures, agrégations, tri, pagination), la lecture du SQL généré et la correction des pièges de performance courants, ainsi que les migrations (créer, appliquer et relire).

**Atelier pratique :** *traduire en LINQ une série de requêtes SQL classiques, puis ajouter une propriété à une entité, générer la migration et inspecter le SQL produit.*

### Module 6 — Front-end Vue.js et débogage full-stack

Un périmètre volontairement ciblé : lire, comprendre et modifier le front, sans viser le métier de développeur front. On découvre la structure d'un projet Vue sous VS Code, l'anatomie d'un composant (gabarit, script, style), la liaison de données et les directives essentielles, les appels HTTP vers l'API et la gestion de l'asynchrone, ainsi que les outils du navigateur (onglet réseau, console, extension de développement Vue).

**Atelier pratique :** *ajouter un champ à un formulaire existant et le relier au modèle, puis suivre l'appel API dans l'onglet réseau, corrélé avec un point d'arrêt côté back-end.*

**Module 7 — Sécurité (OAuth2 / JWT), conteneurs Docker & traitements planifiés**

Un module à visée culture technique et diagnostic, pas administration. On distingue authentification et autorisation, on présente OAuth 2.0 / OpenID Connect et l'anatomie d'un jeton JWT (structure, revendications, rôles, expiration, propagation du front vers l'API). Côté conteneurs, on acquiert le vocabulaire indispensable (image, conteneur, volume) et la lecture d'un fichier de composition, le démarrage d'une pile et la consultation des journaux. Enfin, on aborde les traitements d'arrière-plan et l'ordonnancement : écrire un traitement récurrent, générer et envoyer un rapport automatisé, journaliser et superviser.

**Atelier pratique :** *décoder un jeton JWT et diagnostiquer une erreur d'accès, démarrer la pile applicative en conteneurs, puis écrire un traitement planifié de surveillance qui produit et envoie un rapport.*

## POINTS FORTS DE LA FORMATION

- Une transition guidée du client lourd C# vers le développement web moderne .NET / Vue.js.
- Environ 80 % de pratique sur un projet réel : de l'écran jusqu'à la base de données.
- Pensée pour un développeur expérimenté rejoignant une équipe ou reprenant une application.
- Petit groupe de 1 à 3 participants, session garantie dès 1 inscrit.

## MÉTHODES PÉDAGOGIQUES

### Format et déroulement

La formation est dispensée en présentiel ou à distance via une classe virtuelle interactive, avec un contenu adaptable aux besoins de votre projet. La répartition théorie/pratique est d'environ 20 %/80 %.

### Format ultra-personnalisé MFE-IT

Chaque session accueille de 1 à 3 participants, garantissant un accompagnement hautement individualisé. Un entretien préalable permet d'adapter le contenu au profil de chaque participant. Les sessions inter-entreprises sont garanties dès 1 inscrit (aucun risque de report sauf cas de force majeure).

### Évaluation des compétences

Tout au long de la formation, le formateur évalue la progression à travers des mises en situation et des travaux pratiques. À l'issue, une attestation de réussite est remise à chaque participant.

### Suivi post-formation

Pendant un mois après la formation, chaque participant peut contacter les formateurs MFE-IT pour toute question relative à la mise en œuvre des acquis. Une réponse est apportée sous 48 heures ouvrées.

### Accessibilité

MFE-IT s'engage à accueillir les personnes en situation de handicap. Contact : contact@mfe-it.com.

## INFORMATIONS PRATIQUES

### Ressources du formateur

- Démonstrations structurées alignées sur le programme détaillé
- Énoncés d'exercices et corrigés tout au long de la formation
- Un dépôt de projet et un environnement technique prêts à l'emploi
- Validation des acquis par le formateur à la fin de chaque atelier

### Certification et validation

À l'issue de la formation, une attestation est envoyée par e-mail précisant les objectifs, la nature, la durée et les résultats de l'évaluation. Un certificat de réalisation peut être fourni sur demande.