Model-View-ViewModel

Introduction

Le Model-View-ViewModel (MVVM) est un modèle architectural destiné à structurer les applications en séparant la logique métier, la présentation et la gestion des états de l'interface utilisateur.
Conçu pour les applications WPF (Windows Presentation Foundation) de Microsoft, MVVM s'est ensuite largement imposé dans le développement desktop, mobile (ex : Android, SwiftUI) et Web (ex : Angular).
Objectif : faciliter la liaison automatique des données entre interface et modèle (data binding) et simplifier le test unitaire et la maintenance en séparant strictement responsabilités et comportements.

Concepts

- Model (Modèle) : Contient la logique métier et les données. Exemples : objets métier, gestion de persistance.
- View (Vue) : Composant graphique (UI) qui affiche les données. Déclare les liaisons aux propriétés exposées par le ViewModel.
- ViewModel : Sert de pont entre la vue et le modèle. Contient les propriétés et commandes que la vue peut lier et utiliser. Expose l'état de la vue mais ne contient aucun code spécifique à l'interface.

Architecture

- La Vue s'abonne aux propriétés et commandes du ViewModel via un système de liaison de données (data binding).
- Le ViewModel récupère ou modifie les données via le Modèle.
- Les changements dans le Modèle (ex : base de données mise à jour) se propagent automatiquement à la Vue grâce au binding.

Utilisation

- Faible couplage entre l'interface utilisateur et la logique métier.
- Facilité de test du ViewModel sans UI.
- Réactivité de l'interface grâce à la liaison de données automatique.
- Réutilisation accrue des ViewModels pour différentes vues.
- Complexité accrue pour les très petites applications.
- Dépendance forte à des frameworks de binding (ex: WPF, Angular).
- Surcharge possible dans le ViewModel si mal structuré.

Conclusion

Le MVVM est particulièrement adapté aux interfaces interactives modernes, notamment lorsqu'il est nécessaire de synchroniser dynamiquement l'état de l'interface avec les données.
En séparant clairement les responsabilités et en facilitant le test unitaire, MVVM aide à développer des applications plus maintenables, modulaires et scalables.
Il est recommandé pour les projets WPF, Xamarin, SwiftUI, Angular ou tout environnement proposant un système de binding performant.