Introduction

Le design pattern MVC (Modèle-Vue-Contrôleur) est une architecture logicielle largement utilisée pour organiser et structurer le code dans une application. Ce pattern sépare les préoccupations de l'application en trois composants principaux :

- Modèle (Model) : Gère la logique métier et les données.

- Vue (View) : Affiche l'interface utilisateur et reflète l'état du modèle.

- Contrôleur (Controller) : Interagit avec les utilisateurs et orchestre les actions entre le modèle et la vue.

Pourquoi utiliser MVC ?

1. Séparation des responsabilités : Chaque composant est responsable d'un aspect spécifique, facilitant ainsi la maintenance et l'évolution du code.

2. Réutilisabilité : Les modèles et les vues peuvent être réutilisés dans différentes parties de l'application.

3. Testabilité : La séparation logique permet de tester les composants de manière indépendante.

Description de l’implémentation MVC dans notre projet

1. Le Modèle (Model)

- Responsabilité :

- Gère les données de l'application.

- Définit la logique métier et les règles de gestion.

2. La Vue (View)

- Responsabilité :

- Affiche les informations au client (utilisateur).

- Reçoit des données du contrôleur pour les afficher sous forme compréhensible.

3. Le Contrôleur (Controller)

- Responsabilité :

- Sert d'intermédiaire entre le modèle et la vue.

- Traite les requêtes des utilisateurs, exécute la logique métier via le modèle, et retourne les résultats à la vue.

Flux de données dans MVC

1. Requête utilisateur : L'utilisateur effectue une action (par exemple, en cliquant sur un bouton).

2. Contrôleur :

- Reçoit la requête.

- Exécute la logique nécessaire via le modèle.

- Passe les données récupérées à la vue.

3. Vue: Affiche les données sous un format compréhensible.

Avantages :

- Amélioration de l'organisation du code.

- Collaboration facilitée entre développeurs (front-end et back-end).

- Facilité de test et de débogage.

Conclusion

Le design pattern MVC va être essentiel dans notre projet pour structurer efficacement le code et faciliter la collaboration entre les développeurs. Cette architecture garantit une application modulaire, maintenable et scalable.