Style architecturaux :

Client/serveur :

C’est une structure d’application distribuée qui divise les taches ou les charges de travail entre les fournisseurs d’une ressource ou d’un service et les consommateurs du service

• Serveurs: • Attendent à gérer des connections

• Fournissent des ressources

• Clients: • Demandent des services

• Initient des connexions

• Peut être déployé sur du matériel séparé ou pas

→messagerie

→ serveur de fichier

→ serveur base de donnée

Plugin :

Un composant qui rajoute une fonctionnalité spécifique à un programme existant  
Raisons: • Permettre la création des fonctionnalités nouvelles par des développeurs tiers

• Supporter le rajout facile de nouvelles fonctionnalités

• Réduire la taille d’un programme

• Séparer le code source d’un programme à cause de licences incompatibles

Avantages :

Alléger le noyau de fonctionnalités secondaires, ce qui en allège la maintenance et diminue la surface de vulnérabilité aux attaques de sécurité.

* Modulariser le développement des fonctionnalités annexes en équipes indépendantes.
* Donner accès, pour une même fonction, à différentes solutions (chacune dans un plugin différent) : différentes implémentations ou algorithmes, différentes ressources mises en oeuvre, différents publics, différents degrés de complexité, différents coûts (gratuit ou payant), etc.
* Avoir un aperçu concret d'une fonctionnalité qui sera ensuite éventuellement intégrée au logiciel principal dans une version améliorée.

MVC (modèle-Vue-Controlleur) [vue en programmation mobile]

Sépare la couche Interface Graphique des autres parties du système => la couplage est réduit entre l’interface graphique et le reste du système  
Bon moyen d’obtenir la cohésion des couches  
Les couches du patron MVC:

• Modèle • gère les données et les objets métier

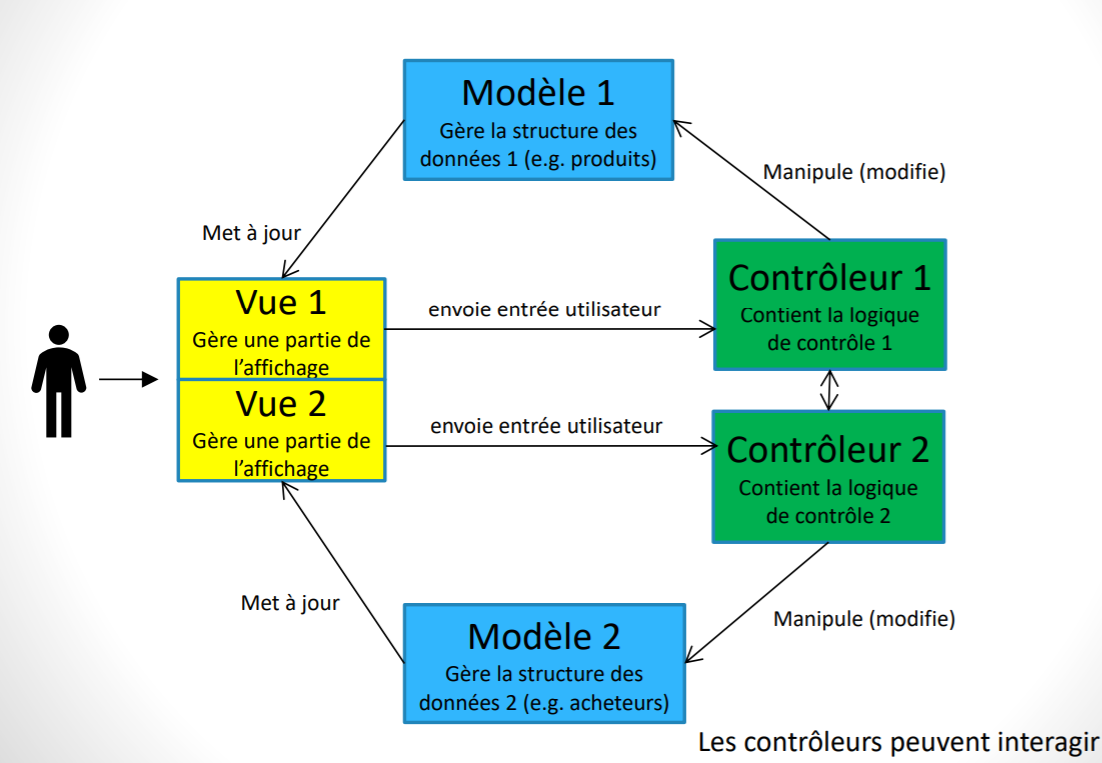
• les classes sont usuellement spécialisées pour l’application

• Vue • gère la mise en forme et l’affichage (apparence, boutons, interaction utilisateur)

• Contrôleur • achemine les commandes vers le modèle et vers la vue

• exécute la logique métier et répond aux évènements de la Vue

Le patron MVC peut être intégré dans la couche présentation d’une architecture N-tier



Avantage :Un avantage apporté par ce modèle est la clarté de l'architecture qu'il impose. Cela simplifie la tâche du développeur qui tenterait d'effectuer une maintenance ou une amélioration sur le projet. En effet, la modification des traitements ne change en rien la vue. Par exemple on peut passer d'une base de données de type SQL à XML en changeant simplement les traitements d'interaction avec la base, et les vues ne s'en trouvent pas affectées. Le MVC montre ses limites dans le cadre des applications utilisant les technologies du web, bâties à partir de serveurs d'applications.Des couches supplémentaires sont alors introduites ainsi que les mécanismes d’inversion de contrôle et d’injection de dépendance

Architecture en couche/ multicouche:

Architecture en couches

Chaque couche communique UNIQUEMENT avec la couche inférieure  
Les couches inférieures ne connaissent pas les couches supérieures  
La couche supérieure voit les couches inférieures comme un set de services  
Cette notion est fondamentale pour une bonne conception  
Les couches sont remplaçables  
Pas d’impact sur les autres couches et dépendances si les interfaces restent inchangées  
Exemple: Couche interface utilisateur (bureau, web, mobile)

On sépare bien les responsabilité  
La cohésion est très bonne  
Les couches regroupent les responsabilités de manière solidaire

3 couches :

Fonctionne un peu de la m^me manière que le mvc présentation=vue +modèle , traitement = controleur, base de donnée= serveur sql

→ sauvegarde des données sur serveur avec les architectures : mvc, multicouche, client/serveur et plugin (en fonction de la manière dont l’app été développée)

→ sauvegrade des données (possible) en cache : plugin, mvc

→ messagerie instantané (réactualisation non nécessaire) → plugin, client/serveur, (multicouche)

(requête généré par le serveur directement) (mvc donnée non trouvée)