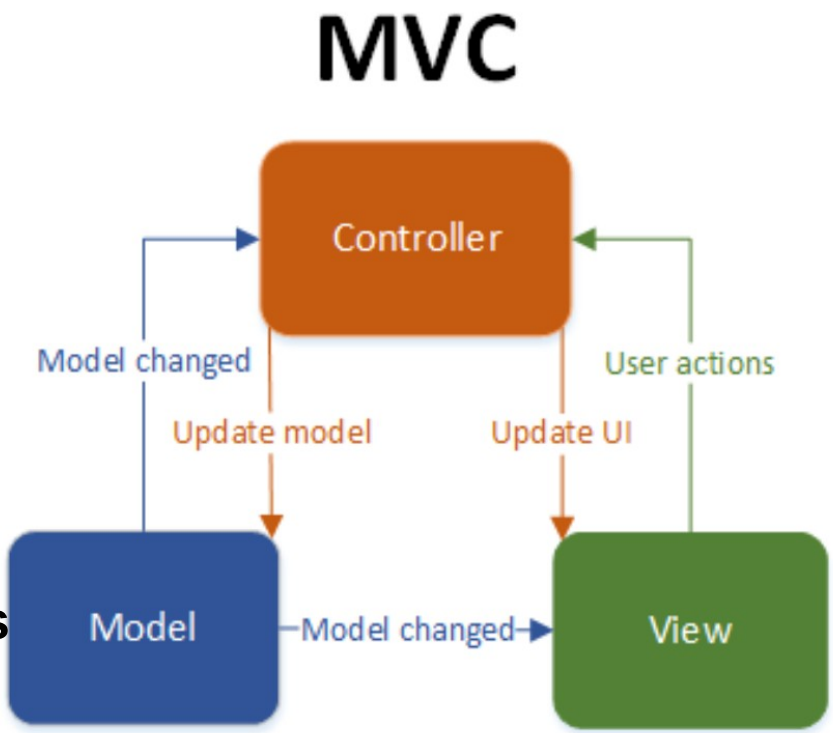
UNIVERSITÉ CÔTE D'AZUR

UTILISATION DU PATTERN MVC SUR ANDROID

@.fR Frédéric RALLO

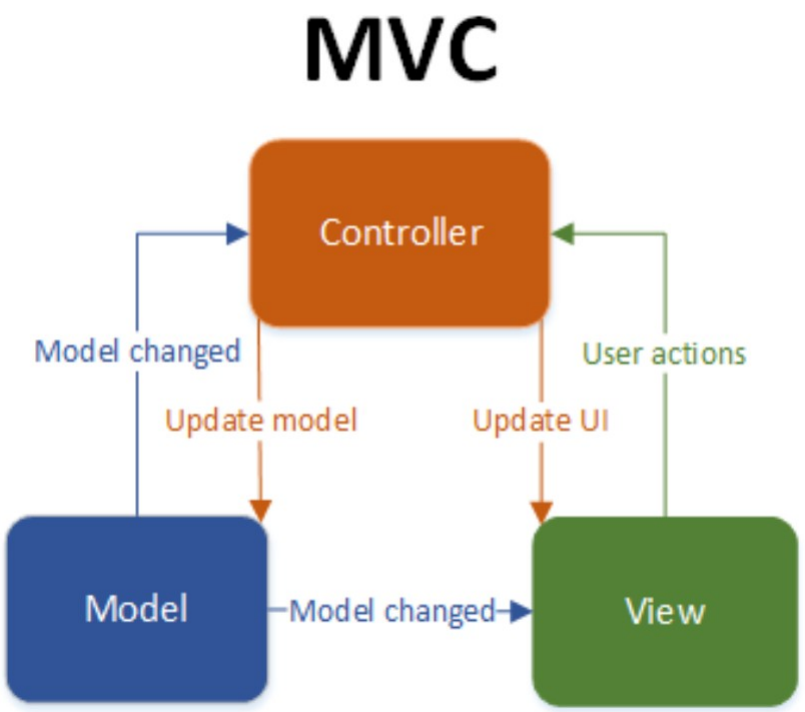
UN PATRON QUI RÈGNE EN MAÎTRE !

- Motif d'architecture logicielle
- Destiné aux interfaces graphiques
- Lancé en 1978
- composé de trois types de modules
- Ayant trois responsabilités différentes
- MVC utilise le design pattern Observer/observable



DANS QUEL CAS L'UTILISER ?

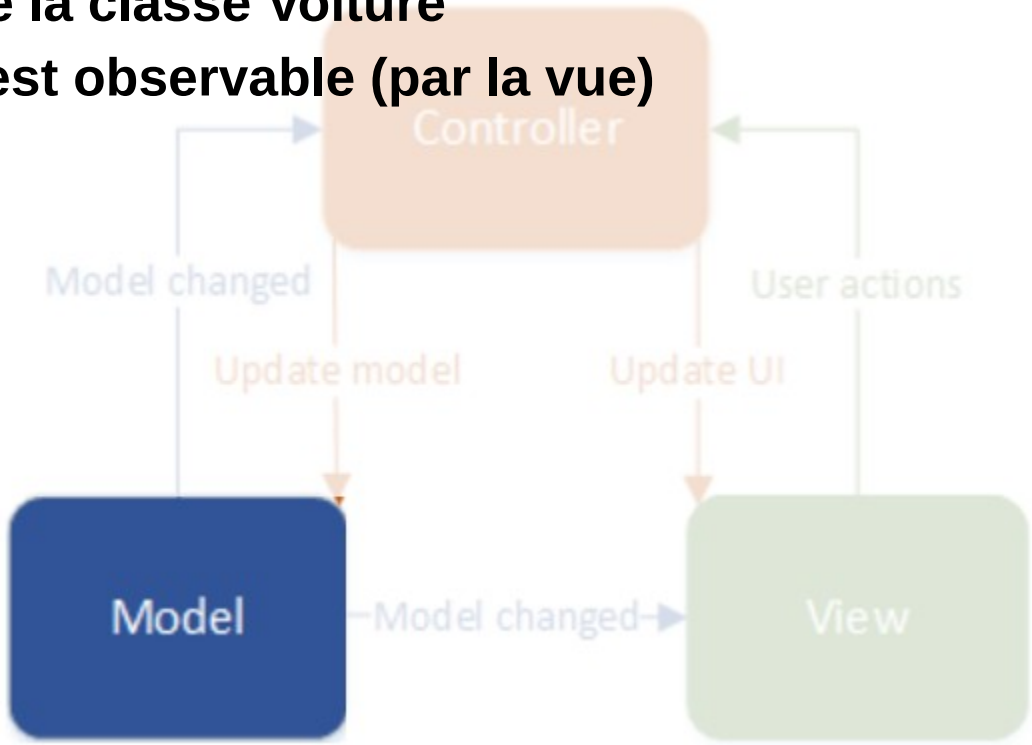
- **MVC maintient la logique métier dans le modèle.**
- **Prend en charge les techniques asynchrones.**



MODÈLE

Définition

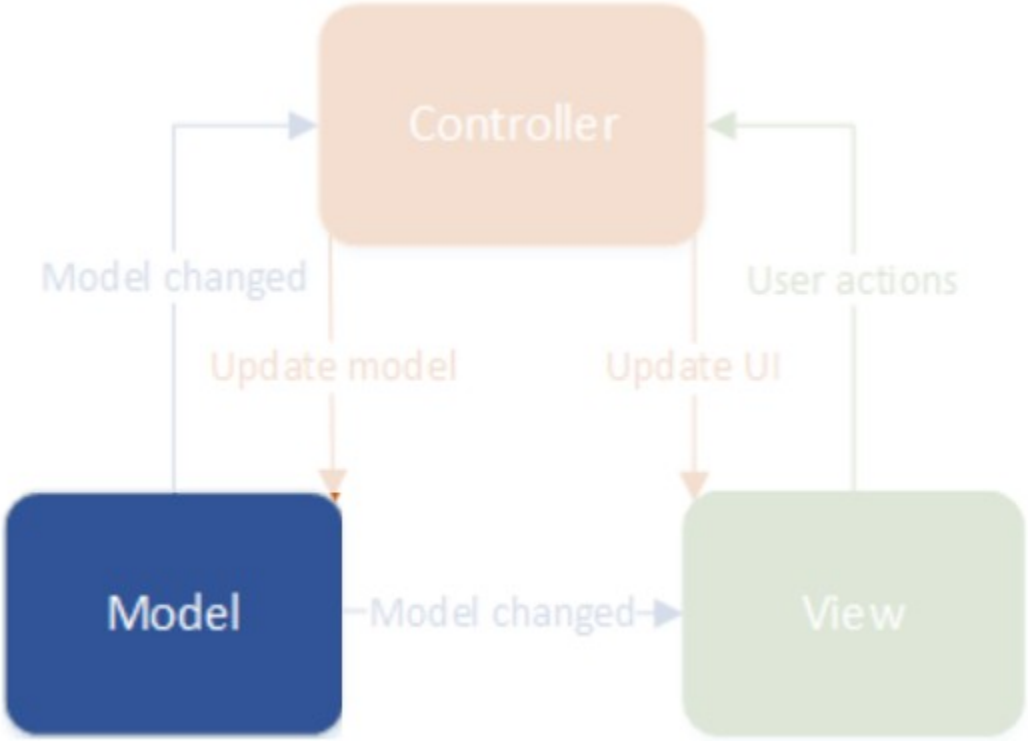
- **Il modélise un objet (au sens java) que l'on veut manipuler**
- **C'est donc une classe avec ses attributs et ses méthodes**
- **par exemple la classe Voiture**
- **Le modèle est observable (par la vue)**



MODÈLE

Quelles relation a-t-il avec les autres modules ?

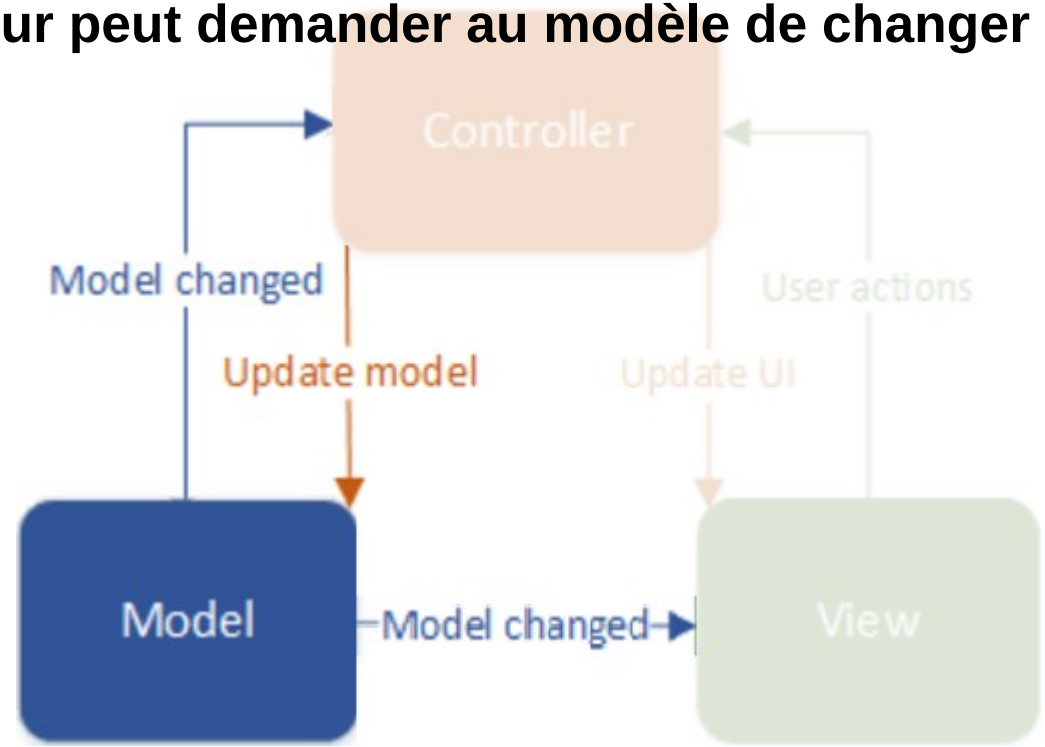
- **La référence du modèle est connue du Contrôleur (association)**
- **Le Modèle connaît la référence du Contrôleur (association)**



MODÈLE

Quels liens a-t-il avec les autres modules ?

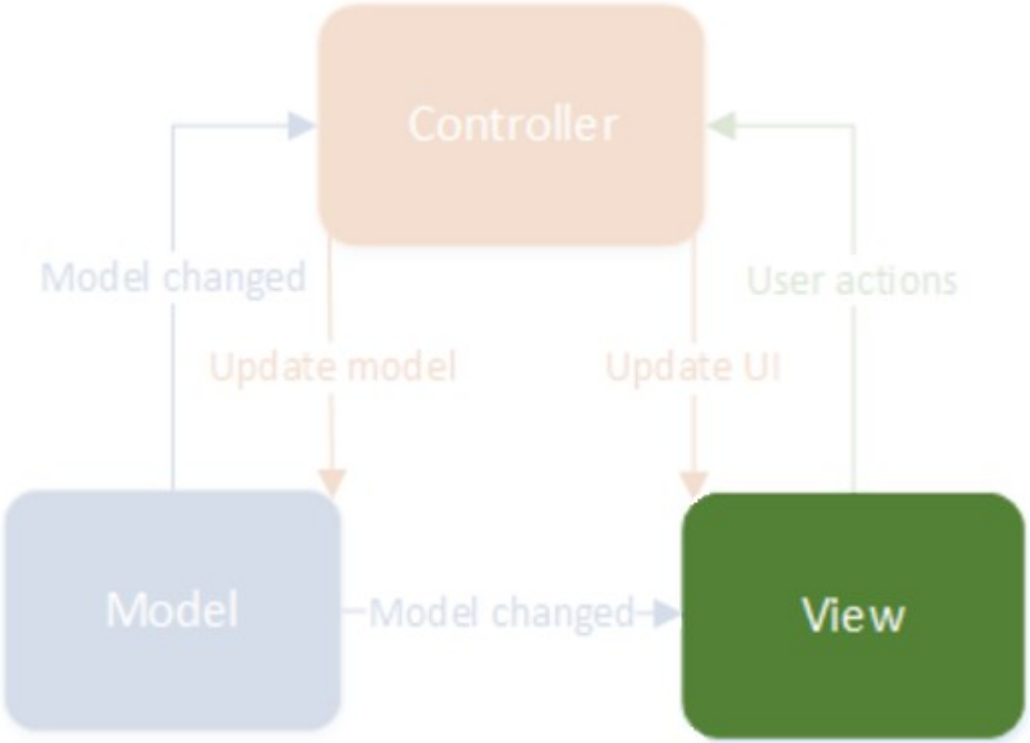
- **Lorsque le modèle change, il informe le contrôleur**
- **Lorsque le modèle change, la vue est informée (elle l'observe)**
- **Le contrôleur peut demander au modèle de changer**



VUE

Définition

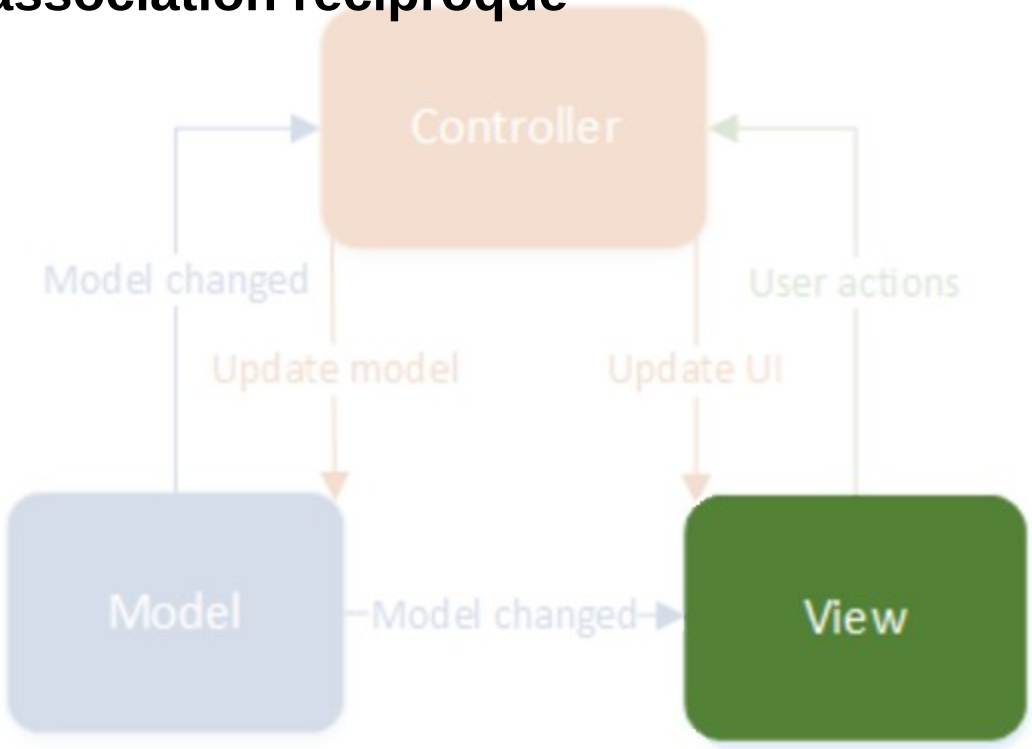
- **Contient les éléments de l'interface graphique (les Widgets)**
- **La vue est un observateur du modèle**



VUE

Quelles relation a-t-il avec les autres modules ?

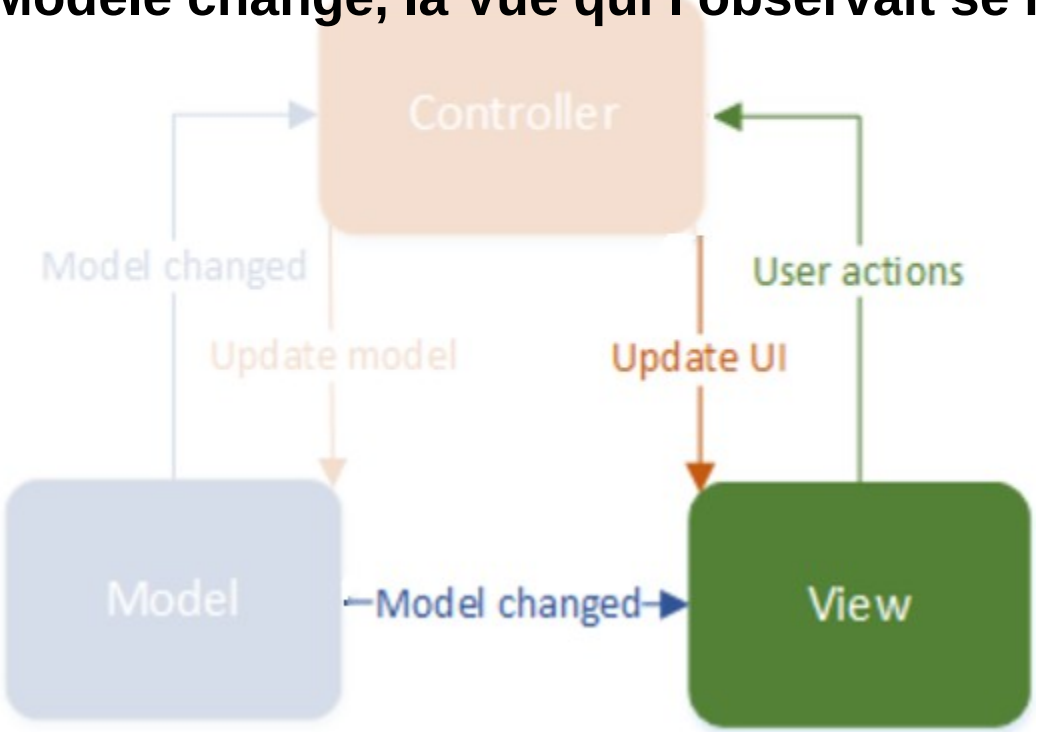
- **La référence de la Vue est connue du Contrôleur**
- **La Vue connaît la référence du Contrôleur**
- **On parle d'association réciproque**



VUE

Quels liens a-t-il avec les autres modules ?

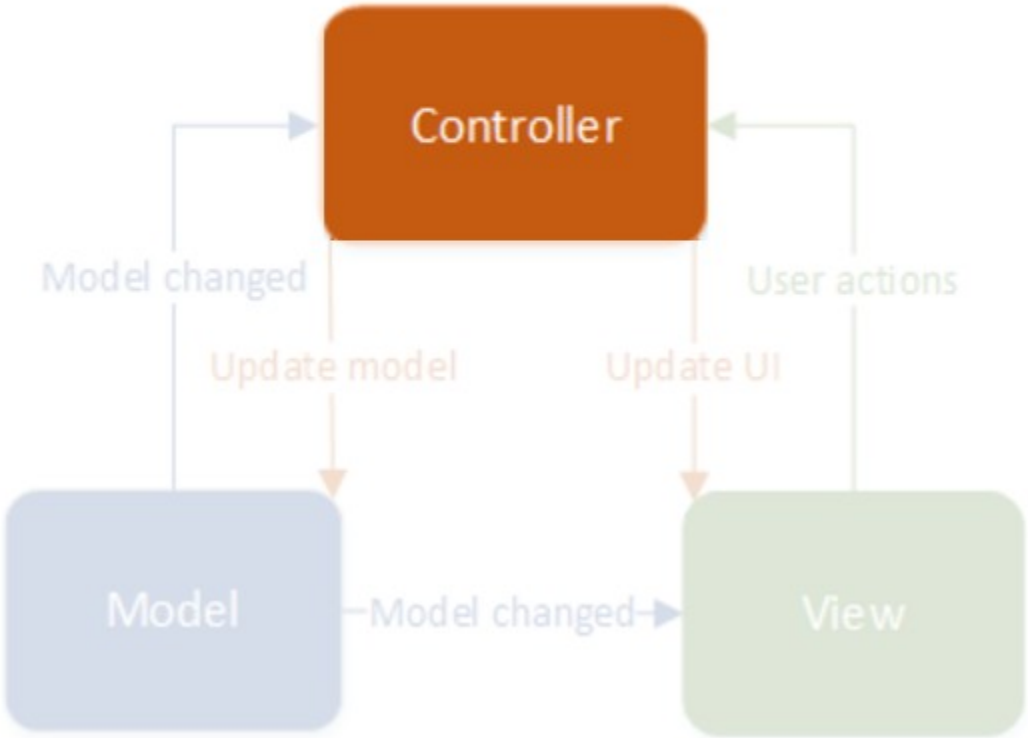
- **Lorsque l'utilisateur agit sur la Vue, celle-ci informe le Contrôleur**
- **Le Contrôleur peut demander de mettre à jour la Vue (masquer…)**
- **Lorsque le Modèle change, la Vue qui l'observait se met à jour**



CONTRÔLEUR

Définition

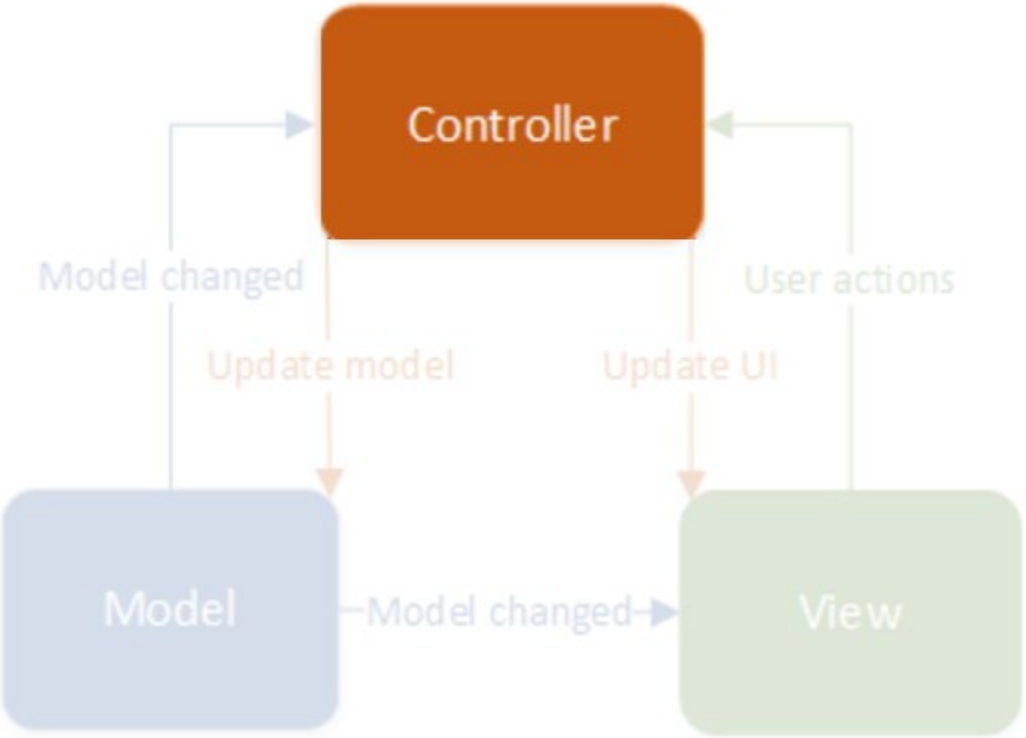
- **Le Contrôleur agit en fonction des actions de l'utilisateur.**
- **Le Contrôleur peut mettre à jour le modèle**
- **Le Contrôleur peut mettre à jour la vue**



CONTRÔLEUR

Quelles relation a-t-il avec les autres modules ?

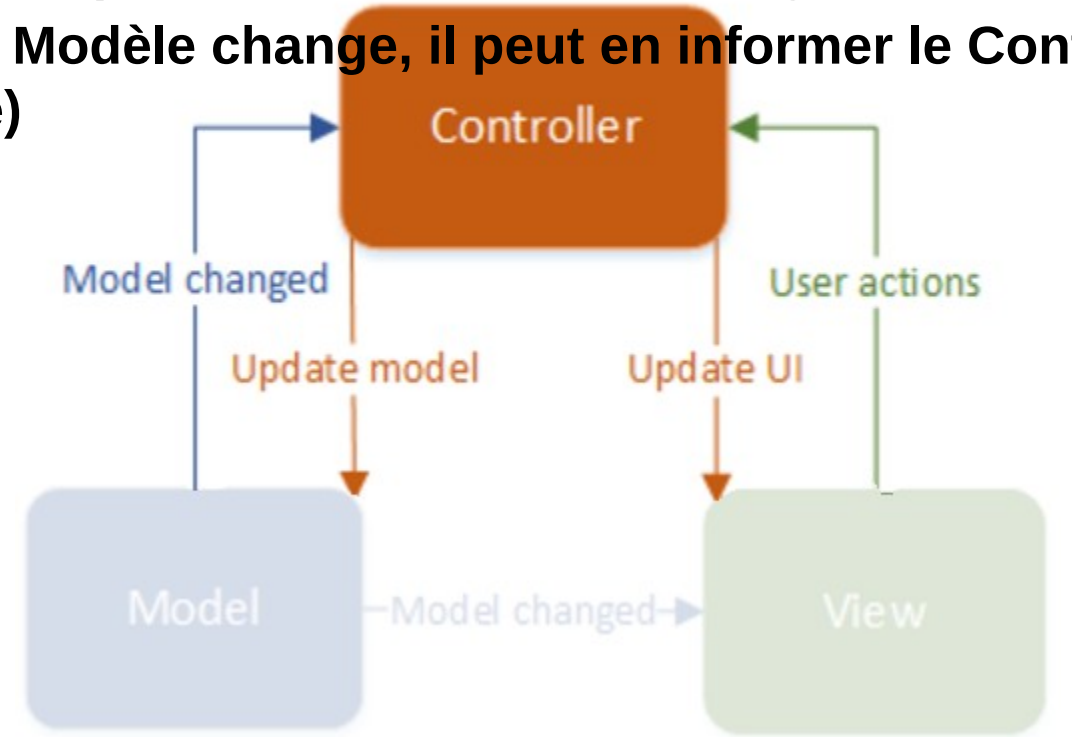
- **Le Contrôleur connaît la référence du Modèle (association)**
- **Le Contrôleur connaît la référence de la Vue (association)**



CONTRÔLEUR

Quels liens a-t-il avec les autres modules ?

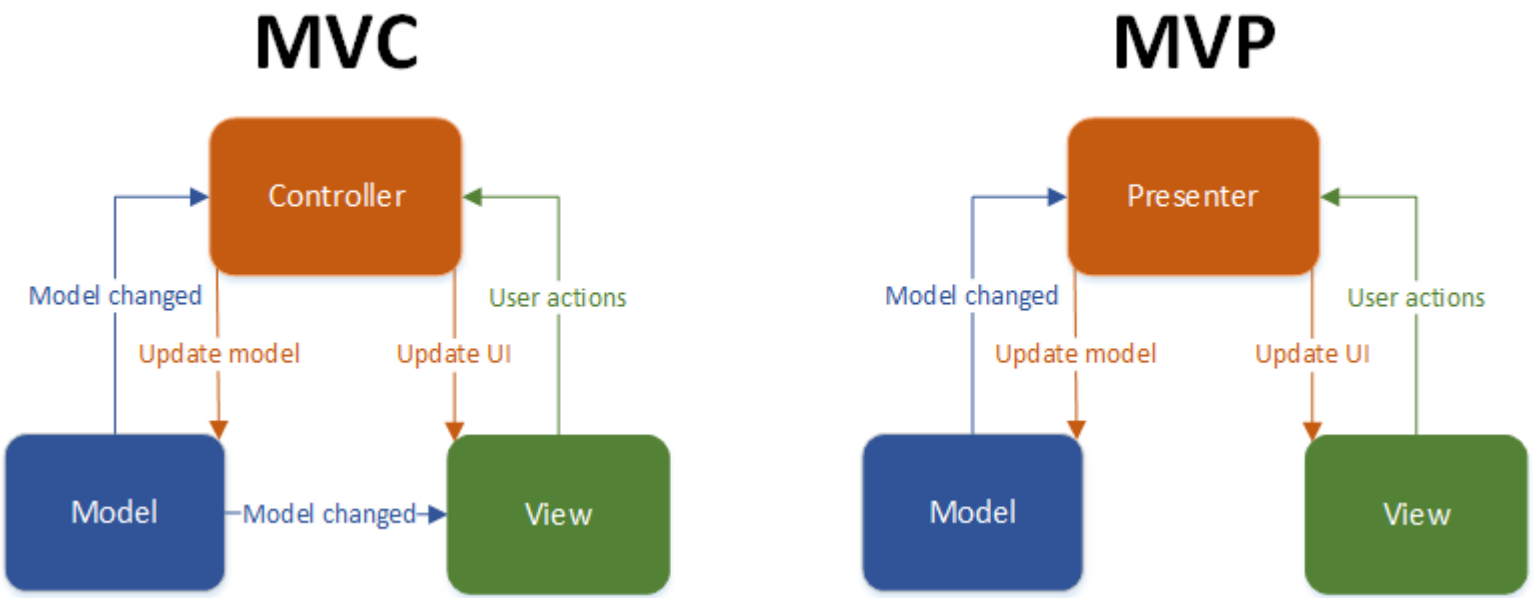
- **La Vue signale au Contrôleur le comportement de l'utilisateur**
- **Le Contrôleur peut demander de mettre à jour la Vue (masquer…)**
- **Le Contrôleur peut demander de mettre à jour le Modèle**
- **Lorsque le Modèle change, il peut en informer le Contrôleur (si nécessaire)**



DES PATRONS PROCHES

- **Difficile (mais pas impossible) à adapter avec Android**

Il ne faut pas confondre MVC et MVP et MVVC



DES PATRONS PROCHES

- **Quelques mots sur MVP (Model-View-Presenter)**

  Le modèle (Model) est identique à MVC

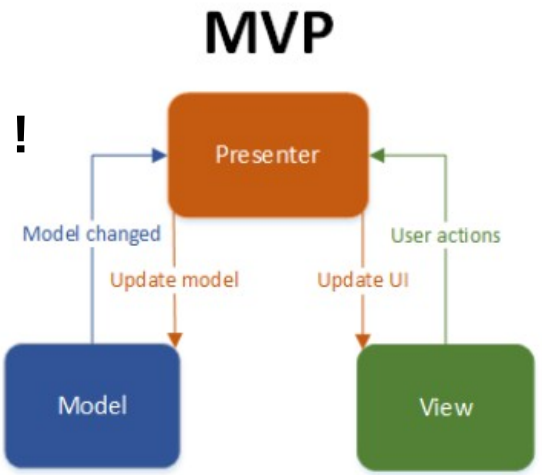
  La vue (View) affiche les éléments et communique avec Presenter

  Pour chaque vue, on affecte un Presenter

  Le présentateur est responsable de récupérer les données à partir du modèle, de les transformer en une forme appropriée pour la vue et de les envoyer à la vue pour affichage.

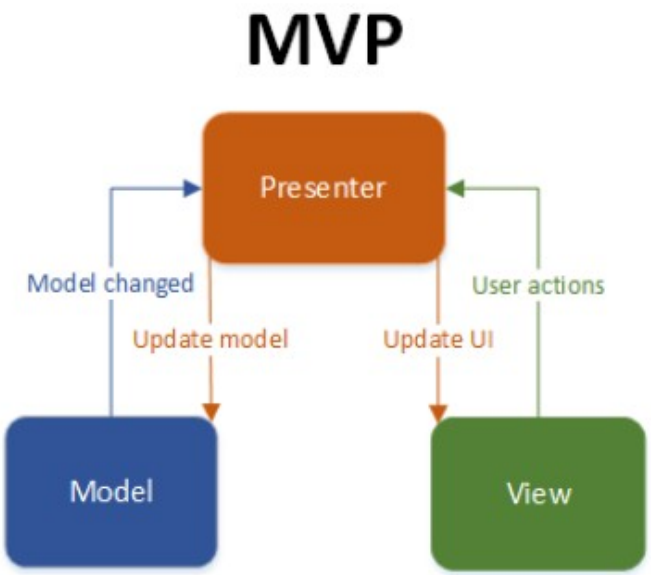
  Le présentateur gère également les actions de l'utilisateur, telles que les clics de souris et les saisies de clavier, et met à jour le modèle en conséquence.

- **MVP est plus simple à adapter à Android !**

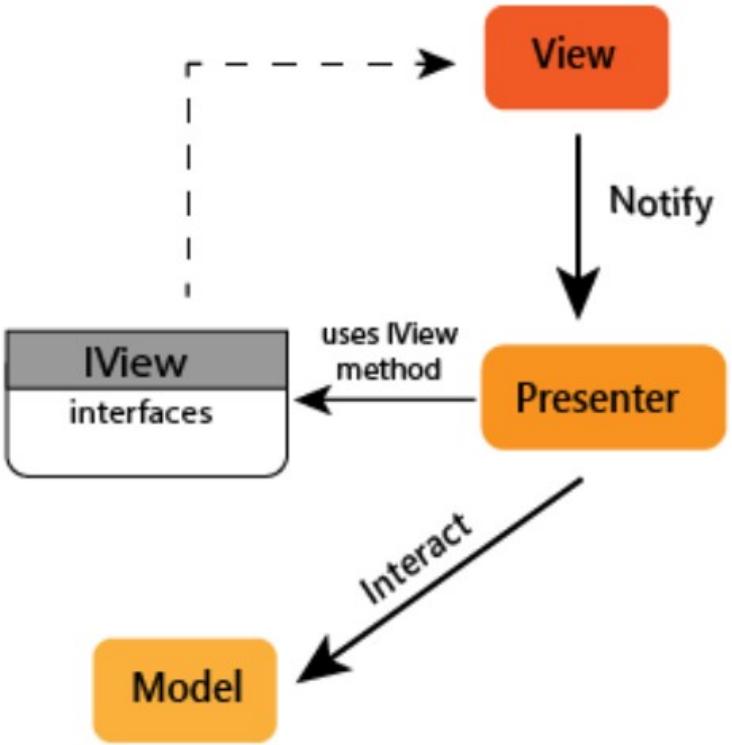


DES PATRONS PROCHES

- **MVP est plus proche d'Android !**



dans Android



POUR ALLER PLUS LOIN

- **Le pattern MVVM**

- **C’est une combinaison de MVC et MVP.**

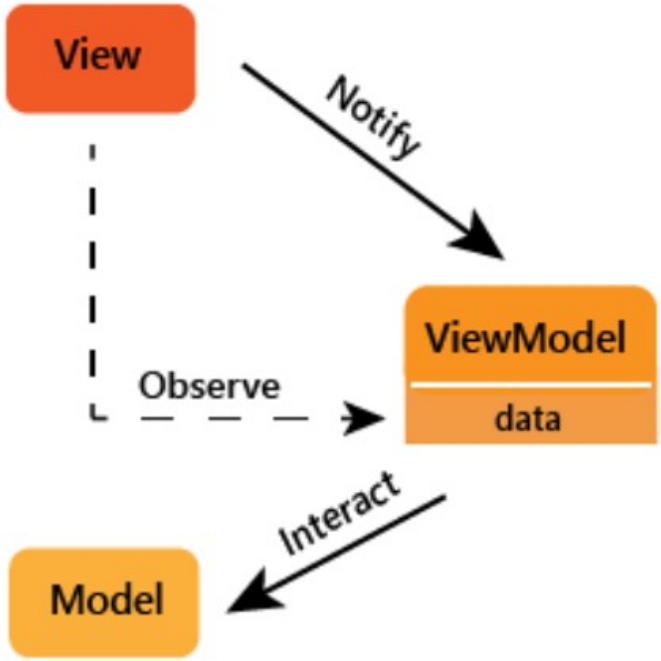
  À pour but de convertir les objets de données du modèle de manière à ce qu'ils puissent être facilement gérés et présentés.

  La vue (View) et le modèle (Model) sont identiques à MVC

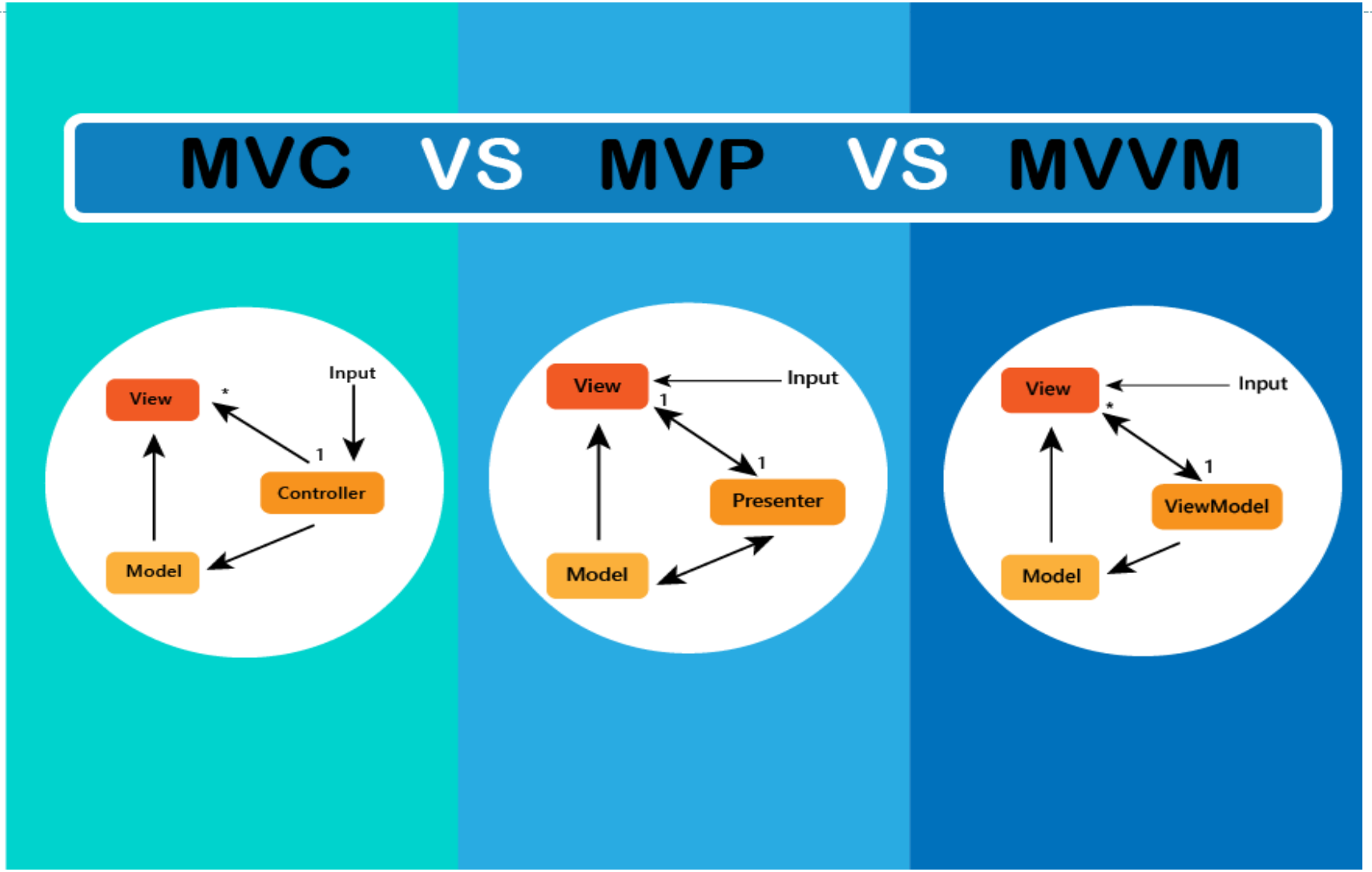
  Le ViewModel est une abstraction de la vue qui fournit en plus un wrapper de modèle de données auquel se lier. Le ViewModel contient les commandes que la vue peut utiliser pour influencer le modèle.

  La principale différence entre MVVM et MVP est que ModelView ne connaît pas la View contrairement au Presenter dans MVP

  On crée 1 seul ViewModel qui gère l’ensemble des View



CONCLUSION


MVC VS MVP VS MVVM
View
Input
*
1
Controller
Model
View
Input
1
1
Presenter
Model
View
Input
*
1
ViewModel
Model