UML
Diagramme de séquence MVC
Diagramme des Classes des Modèles

SOMMAIRE

Edition : juin 2025

MVC

Problématique générale : quels fichiers, quels dossiers pour mes projets

- Le MVC s'intéresse à la problématique de l'organisation des fichiers et des dossiers du code.

- A noter particulièrement les problèmes suivants :
  - Quels dossiers dois-je créer ?
  - Comment dois-je organiser mes fichiers ?
  - Comment passe-t-on d'un fichier à un autre ?
  - Quels sont les inclusions à envisager ?

Organisation non-MVC

Principes d'organisation non-MVC

- On travaille page par page.
- On découpe comme on peut.
- On factorise comme on peut.

Défauts de l'organisation non-MVC

- Tout est mélangé : le SQL, le code serveur et le HTML.
- Les pages peuvent devenir très grosses.
- La maintenance n'est pas facile.
- Travail à plusieurs est rendu difficile.

Architecture MVC

MVC : Modèle - Vue - Contrôleur.

- L'architecture MVC sépare la logique du code en trois parties, trois ensembles de fichiers :
  - le **modèle** : s'occupe des requête SQL et de l'ORM (le passage de l'objet de la POO au données de la BD.
  - la **vue** : s'occupe de l'affichage, en général en HTML
  - le **contrôleur** : s'occupe des traitements. Il correspond au programme principal dans le code serveur et fait le lien entre la vue et le modèle.

Avantages

- Cela rend le code plus facile à mettre à jour et permet d'organiser le travail en 3 parties et donc de travailler en parallèle.

- C'est adapté à la méthode Agile.

- C'est couramment utilisé dans les framework (Laravel, Django, Spring Boot, Angular, etc.)

- La connaissance de l'architecture MVC rend capable de créer un site web de qualité et facile à maintenir.

  - L'architecture MVC est une bonne pratique de programmation.

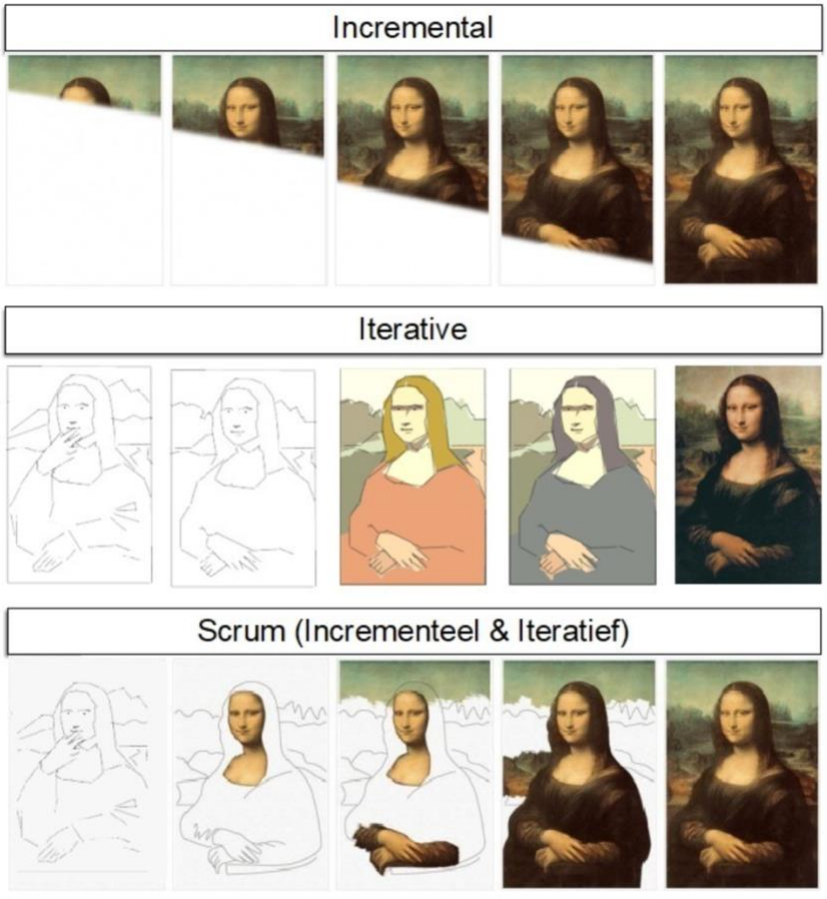
Le Modèle (SQL)

- Le modèle fournit les outils pour accéder aux données du site :
  - Les données de la BD.
  - Les éventuels
  - Les éventuels données et constantes prédéfinies.
- Ces outils proposent des fonctions pour faire des Insert, des Update, des Delete, des Select.
  - Ces fonctions peuvent renvoyer des tableaux de données ou des tableaux d'objets.
  - Les fonctions seront appelées par le contrôleur qui exploitera les résultats.

- L'idée générale est que dans un SI, la base de données est centrale.
  - **Si la BD est bien conçue, l'application sera facile à maintenir et à faire évoluer.**
  - **Si la BD est mal conçue, l'application sera complexe à maintenir.**

Modèle et méthode Agile-Scrum

- On peut faire évoluer la BD au fur et à mesure des besoins : c'est la logique de la programmation « agile » purement incrémental



- Pour le modèle, mieux vaut utiliser de l'agile-scrum (incrémental et itératif).

- Notre méthode consiste à traiter d'abord la quasi-totalité de la BD : on réalise d'abord tout ce qu'on peut réaliser rapidement de la BD avec le DCM et le MPD.

- Ensuite, on adapte en incrémental et itératif avec l'évolution du cahier des charges et de la réalisation.

La Vue (HTML)

- La vue affiche la page HTML qui commence par un **DOCTYPE.**
- Elle est appelée par le Contrôleur.
- Elle récupère des variables du Contrôleur et les affiche.
- La vue fait des calculs très simples : des boucles d'affichage et parfois des tests simples.

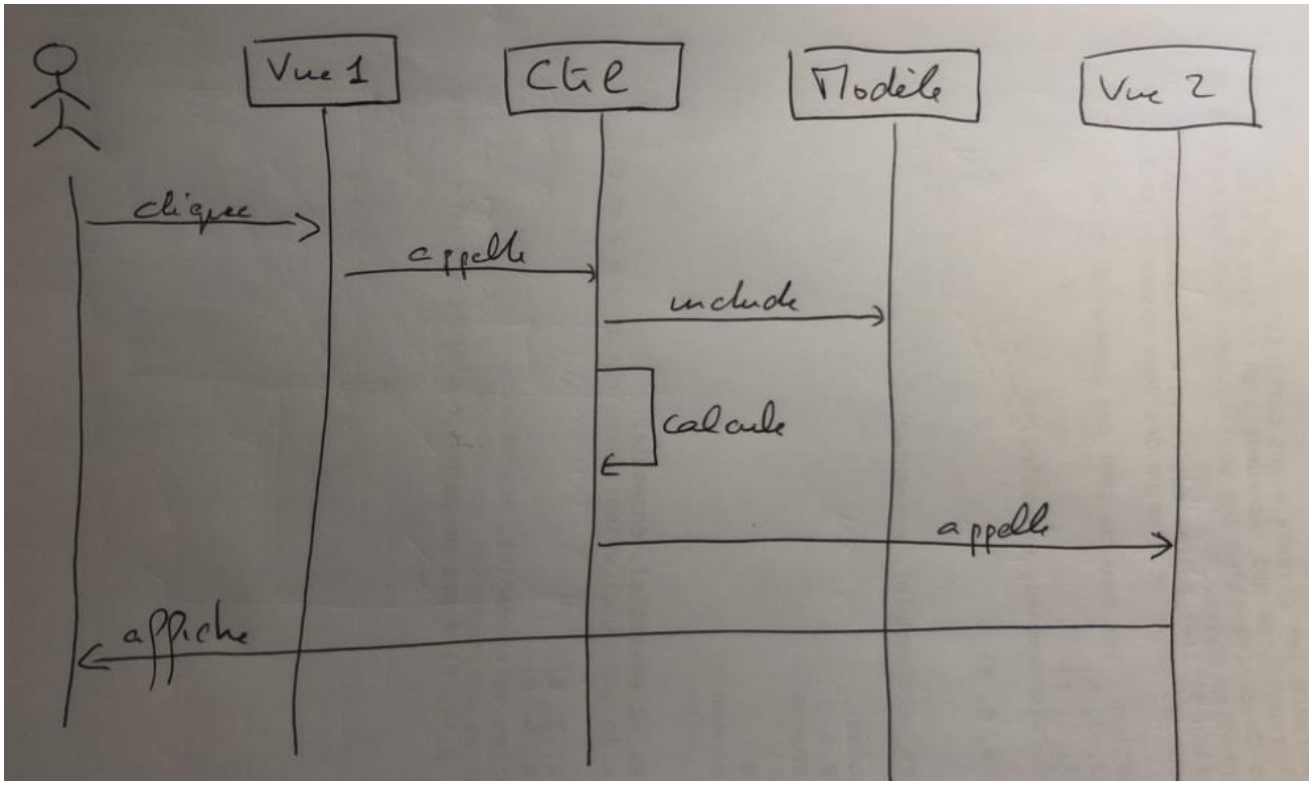
Le Contrôleur (PHP)

- Le contrôleur est le « main » c'est le « chef d'orchestre » :
- Il contrôle la gestion des données, les traitements et l'affichage.
- Il fonctionne en 4 étapes :
  1. Il charge si nécessaire :
     a. les fonctions du Modèle (include en PHP) pour manipuler les données de la BD,
     b. les fonctions d'accès aux fichiers,
     c. les fonctions d'accès constantes hors BD de l'application
     d. les fonctions pour traiter les données passées en paramètre de la page (GET, POST et SESSION)
  2. Il gère les droits d'accès : il détermine par exemple si le visiteur a le droit de voir la page ou non
  3. Il traite les données nécessaires avec ces outils pour arriver aux résultats attendus.
  4. En fonction de ses calculs, il appelle la vue correspondante en lui fournissant les données dont elle a besoin.

Principe de mise en oeuvre

- Les architectures MVC mises en œuvre s'appuient sur de la théorie.
- En pratique, chacun adapte la théorie.
- Il y a donc plusieurs façons de mettre en œuvre le MVC.

Diagramme de séquence MVC :



1. Un utilisateur clique sur un élément cliquable d'une vue.
2. Cet élément fait appel à un contrôleur (par un href ou par un formulaire).
3. Le contrôleur « include » un modèle
4. Ensuite, il utilise des fonctions du modèle pour charger des données et faire des calculs.
5. Selon les résultats, il include une nouvelle vue en lui fournissant les données qu'il a récupérées et calculées.
6. La nouvelle vue est affichée à l'utilisateur.

L'utilisateur peut alors recommencer le processus.

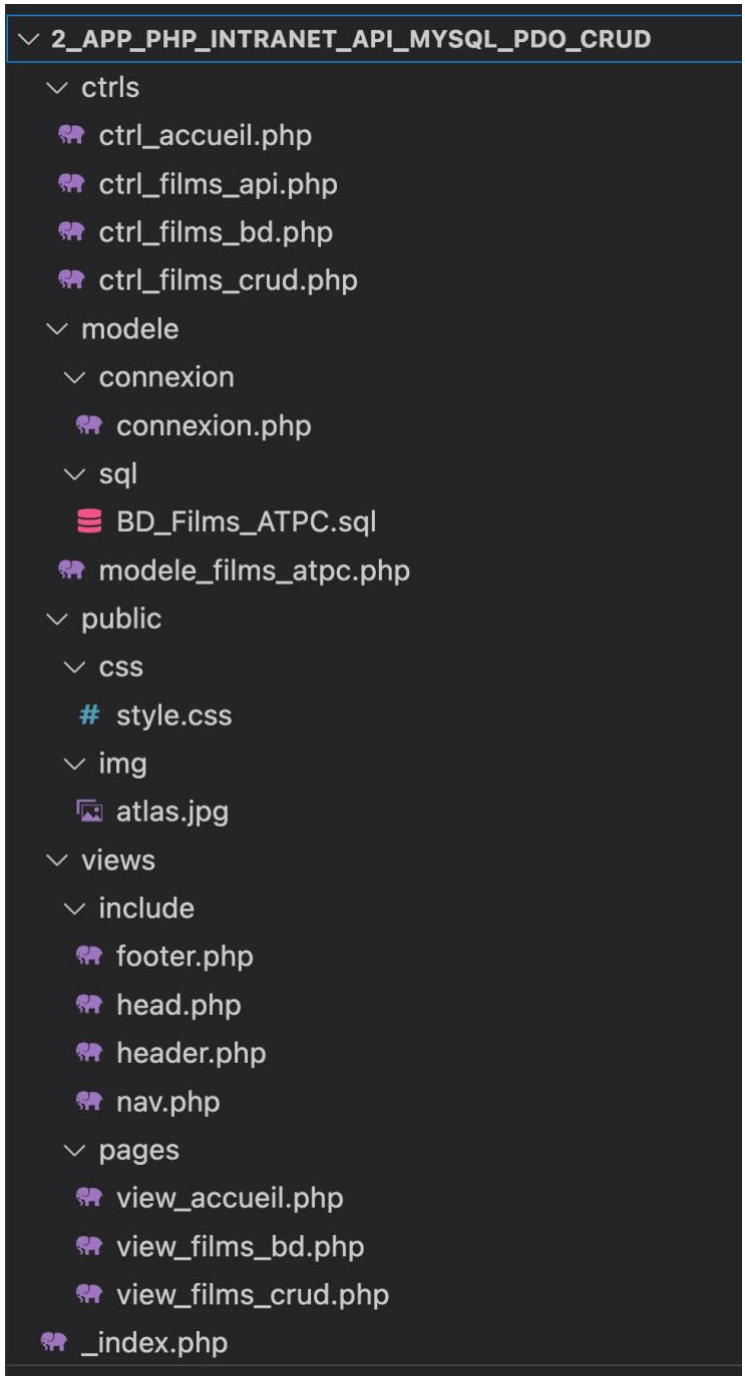
Organisation des fichiers dans le MVC

Organisation de répertoires MVC

- À la racine du site, on crée 4 dossiers et un fichier : index.html
- Cette architecture est minimaliste.
  - Elle fonctionne sans « routeur ».
  - Le contrôleur contient tous les « mains » des différentes pages.

- 4 dossiers :
  - modeles
  - vues
  - contrleurs
  - public

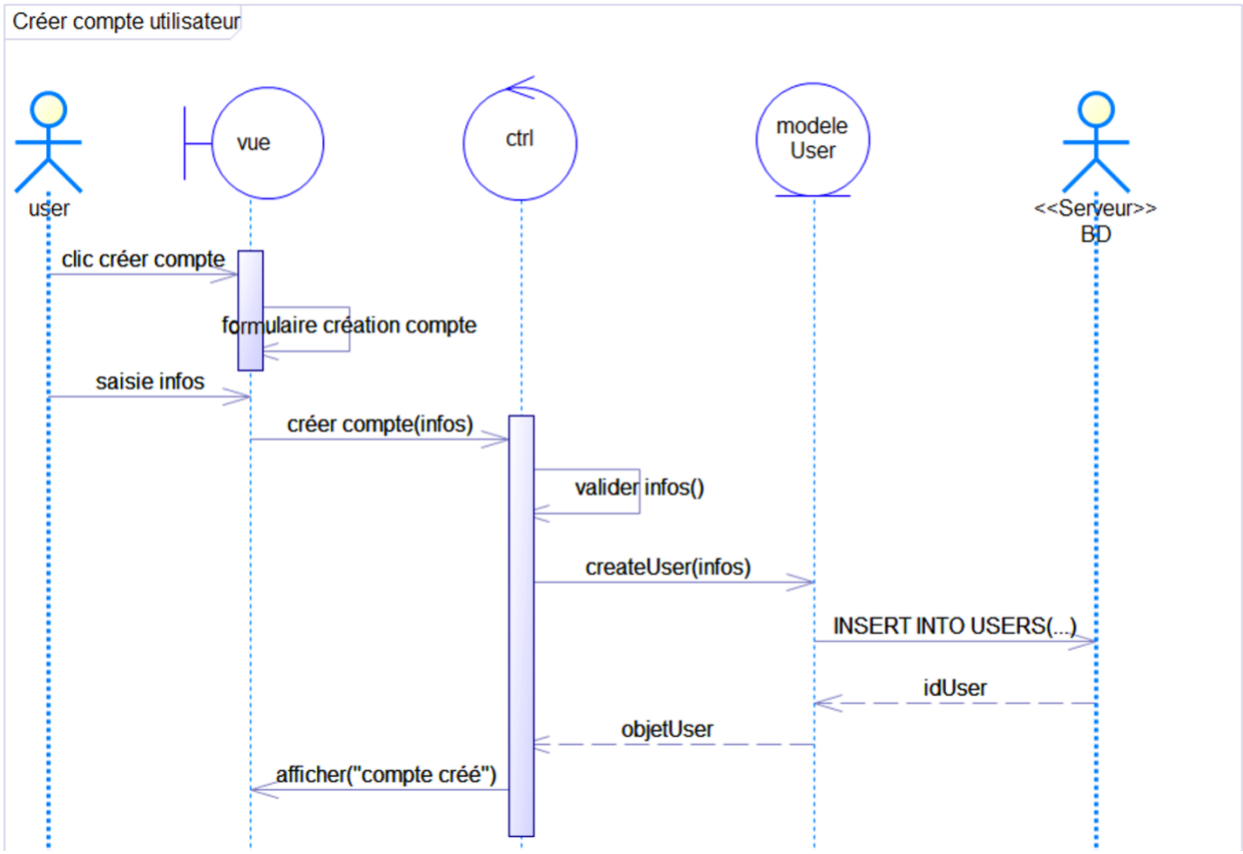
index.html, c'est l'entrée dans le site



MVC et UML : diagramme de séquence MVC d'un scénario nominal

DDS SNUC « Créer compte utilisateur »

Diagramme de séquence du scénario nominal du Use Case « Créer compte utilisateur »



Lecture du diagramme :

vue : n'importe quel objet de la classe Vue (pour la vue concerné).
ctrl : n'importe quel objet de la classe Controleur (pour le contrôleur concerné).
modèleUser : n'importe quel objet de la classe ModèleUser.

Pré-conditions :

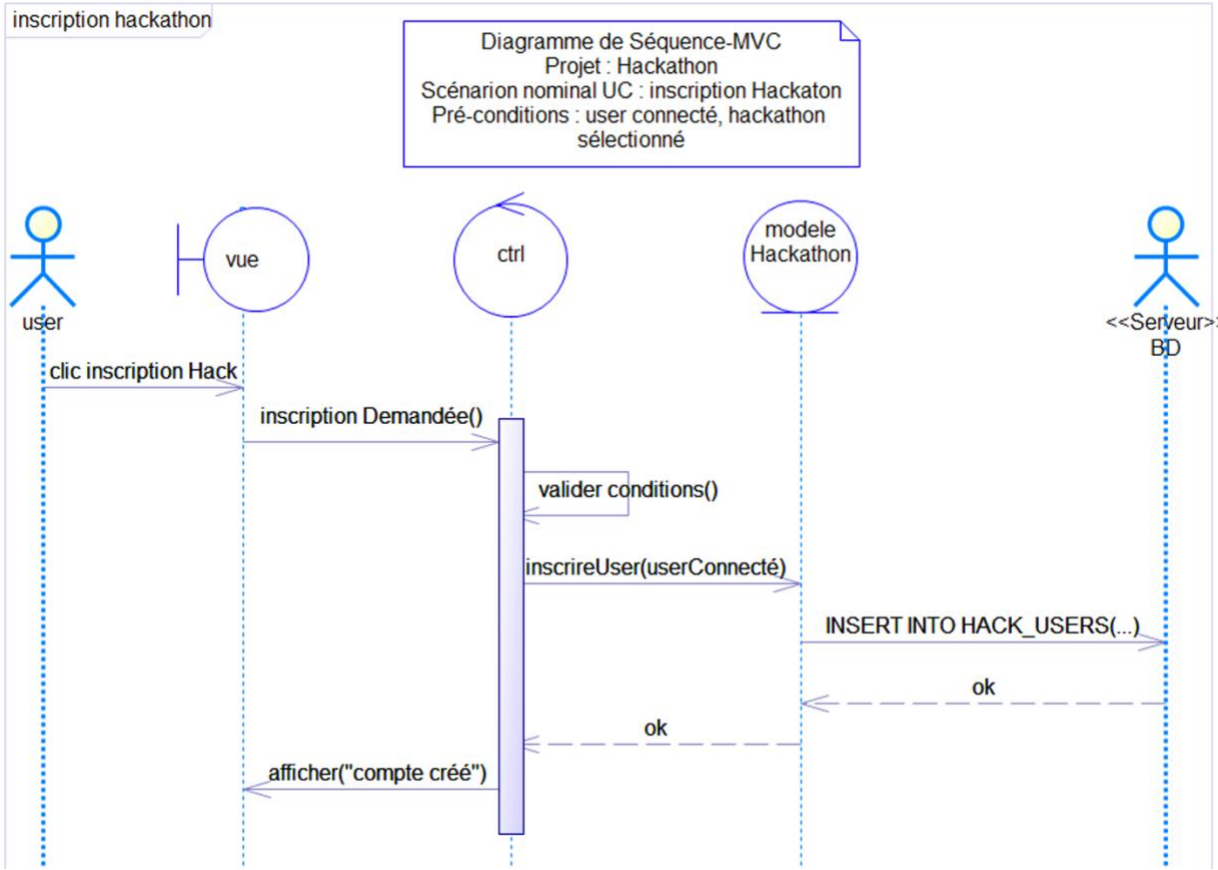
aucune (les données déjà présentes pour réaliser le use case)

Explication

L'utilisateur clique sur créer un compte
La vue affiche un formulaire de saisie
L'utilisateur saisie les infos et valide : on précisera les données à fournir
La vue appelle une méthode créer compte(infos)
Le contrôleur valide les infos saisies
Il appelle la méthode createUser(infos) du Modèle
Le modèle fait l'insert dans la BD
La BD retourne l'id du user créé
Le modèle retourne l'id du user créé
Le contrôleur affiche dans la vue un message pour dire que le compte est créé

DDS SNUC « s'inscrire au hackathon choisi »

Diagramme de séquence système du scénario nominal du Use Case « s'inscrire au hackathon choisi »



Lecture du diagramme :

vue : n'importe quel objet de la classe Vue (pour la vue concerné).
ctrl : n'importe quel objet de la classe Controleur (pour le contrôleur concerné).
modèleHackaton : n'importe quel objet de la classe ModèleHackathon.

inscrireUser(userConnecté) →hackathonChoisi veut dire :
hackathonChoisi. inscrireUser(userConnecté)

Pré-conditions :

- Un user est connecté : on a un objet userConnecté
- Un hackathon est choisi : on a un objet hackathonChoisi

Explication

- L'utilisateur clique sur s'inscrire
- La vue appelle une méthode incriptionDemandé()
- Le contrôleur valide qu'il a le userConnecté et le hackathonChoisi
- Il appelle la méthode inscrireUser(userConnecté) du Modèle du Hackaton
- Le modèle fait l'insert dans la BD
- La BD retourne ok
- Le modèle retourne ok
- Le contrôleur affiche dans la vue un message de validation de l'inscription

DC-Modèles

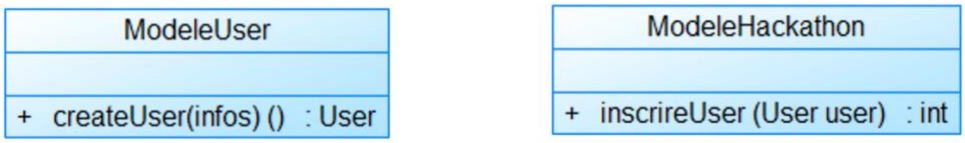
Principes

- Avec le MVC, on va créer un DC-Modèles qui contient les classes des modèles.
  - Il y a une classe du modèle par classe métier.
  - Ce sont des classes sans attribut.
  - Par contre, elles ont des méthodes qu'on met au jour avec l'analyse MVC

- On pourrait mettre les méthodes directement dans les classes métiers.
  - Dans ce cas, on intérêt à les regroupes à la fin.

- Les méthodes des modèles contiennent le code SQL

DC-Modèles



SQL des méthodes

ModeleUser.createUser(info) :

- INSERT INTO USER values
- (on précise les values qu'on passe)

ModeleHackathon. inscrireUser(user) :

- INSERT INTO HACKATHONS_USERS values
- (on précise les values qu'on passe)