

# PHP - P00



# Moviz

# Déroulé

- 5 lives d'une heure sur la mise en place d'un projet en php POO.
- Prérequis :
  - Avoir les connaissances de php et les notions de classe/objet et du modèle MVC.
- Questions/Réponse à la moitié (25min) et fin du live (55min).
- Sources du projet sur github :
  - [https://github.com/arirangz/moviz\\_poo](https://github.com/arirangz/moviz_poo)



# Programme des 5 lives

- Présentation du projet
- Conception de la base de données
- Création de la base de données
- Prise en main de la structure MVC
- Création des classes Entity
- Création des classes Repository
- Mise en place des contrôleurs
- Gestion des vues
- Gestion des formulaires




# Environnement technique

- PHP  $\geq$  8.1 et MySQL  $\geq$  5.7 (WAMP)
- Bootstrap 5.3
- VSCode



# Besoin

- La société Moviz veut proposer un site de notation et critique de films.
  - Elle souhaite que les utilisateurs puissent s'inscrire et noter un film et en même temps poster une critique (optionnel).
  - Elle souhaite pouvoir administrer les films (titre, synopsis, année de sortie, durée) qui seront catégorisés (thriller, action etc.) et pourront être associés à plusieurs réalisateurs (nom, prénom).
  - Elle souhaite également pouvoir administrer les critiques (validation requise) mais la note sera prise en compte automatiquement.
  - Eventuellement on souhaite pouvoir permettre aux visiteurs de filtrer les films (par catégorie, par réalisateur, recherche par mot clé).
  - Une charte graphique a été définie.
- 

# Charte graphique

- Police : **SFMono-Regular**

- Couleurs :



#09b468



#323b3c



#deb887

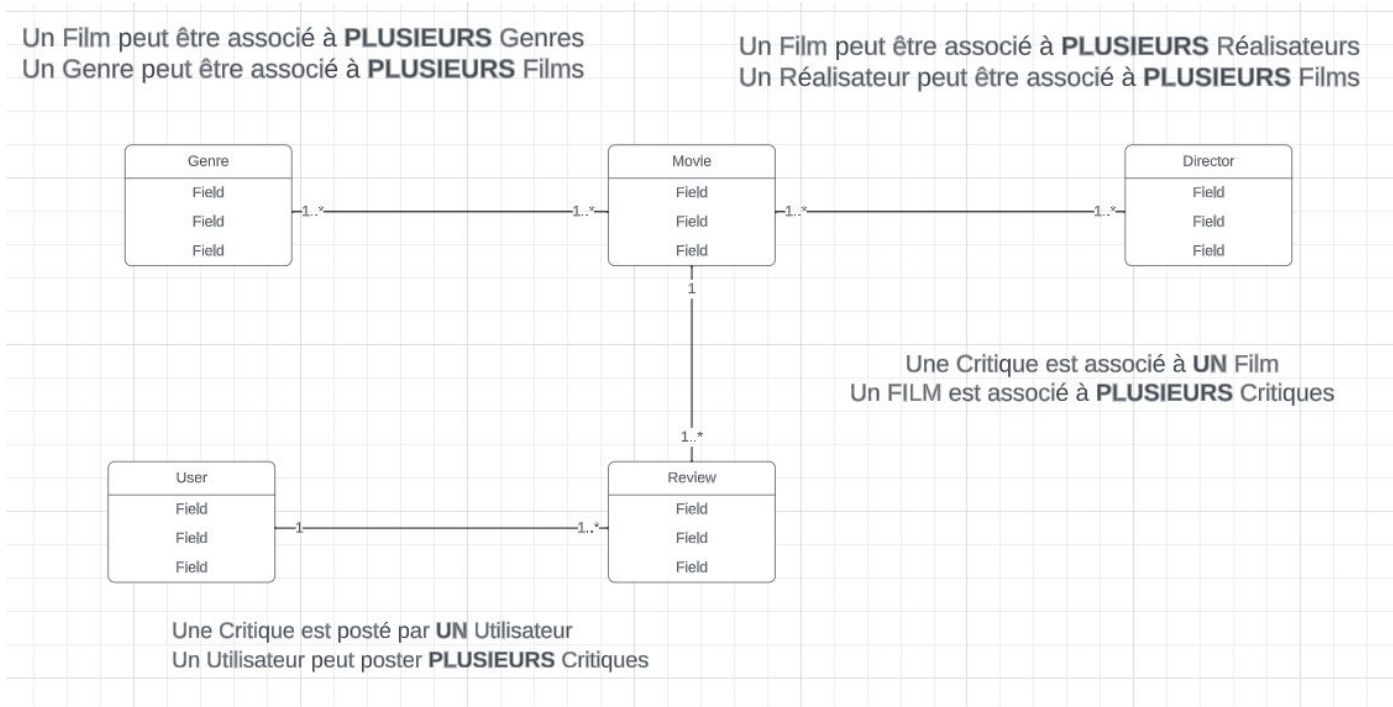
- Logo :



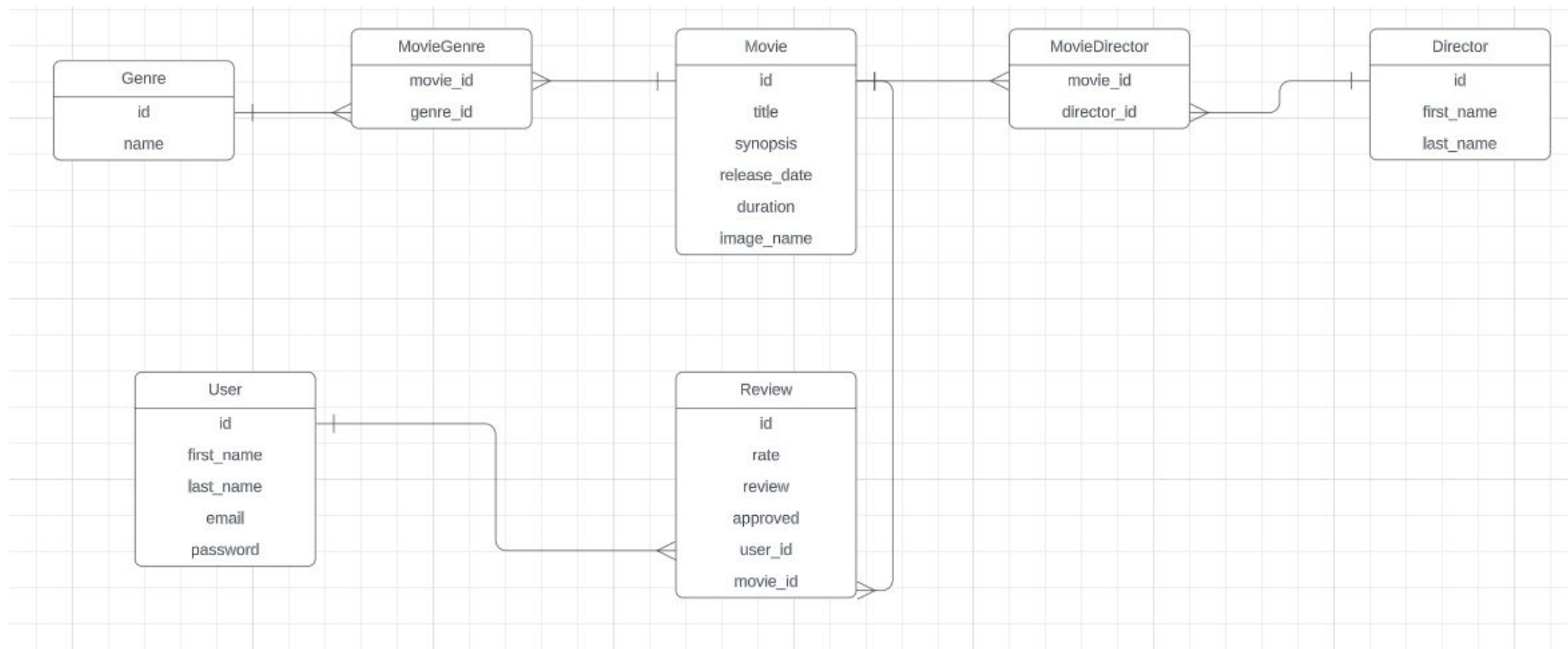
# Moviz

# Modèle Conceptuel de Données

<https://lucid.app/>



# Modèle Logique de Données







DOMAINE LOCAL

# Ajouter un domaine local : Etape 1

Sous windows :

Ouvrir le fichier C:\Windows\System32\drivers\etc\hosts

Ajouter une ligne vers votre domaine local.

ex:

127.0.0.1 monprojet.local

Sous mac, suivre les instructions :

<https://dev.to/crankysparrow/configuring-virtual-hosts-with-mamp-f3i>



# Ajouter un domaine local : Etape 2

Nous allons ensuite devoir indiquer à apache vers quel dossier le domaine doit pointer.

Pour cela il faut modifier le fichier httpd-vhosts.conf

- Sous **Wamp**  
C:\wamp64\bin\apache\apache2.4.51\conf\extra\httpd-vhosts.conf
- Sous **Xampp**  
C:\xampp\apache\conf\extra\httpd-vhosts.conf
- Sous **Mamp**  
/Applications/MAMP/conf/apache/extra/httpd-vhosts.conf

et ajouter la configuration en adaptant les informations

```
<VirtualHost *:80>  
    DocumentRoot "C:\wamp64\www\mon_projet"  
    ServerName monprojet.local  
</VirtualHost>
```

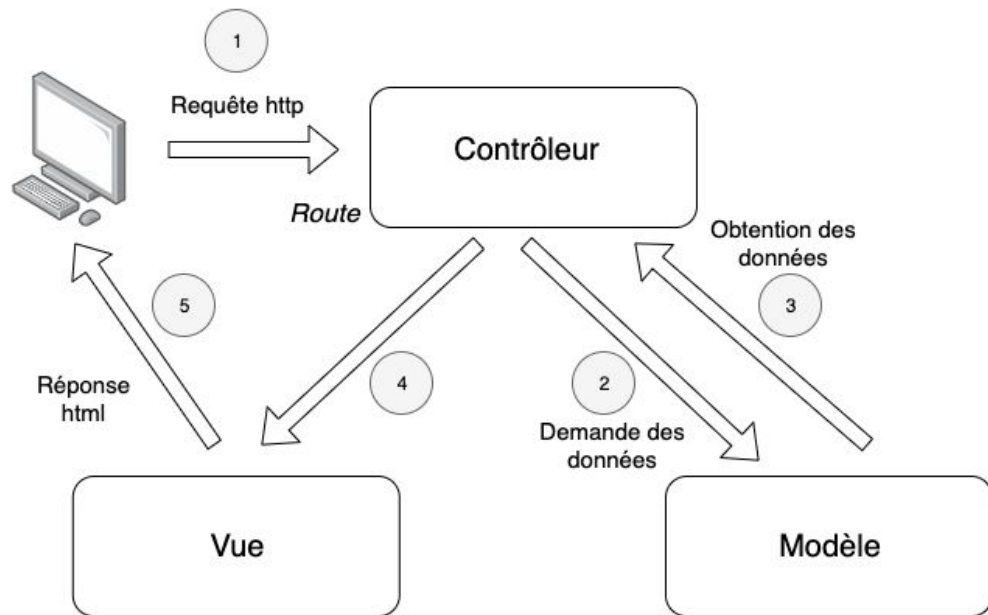
Il faut ensuite redémarrer le serveur





# MVC

# MVC



# MVC : Définition

Le modèle MVC (Modèle-Vue-Contrôleur) est un modèle de conception pour le développement de logiciels qui permet de **séparer les responsabilités** des différentes parties d'une application en trois composantes distinctes : **le modèle, la vue et le contrôleur**.



# MVC

- **Modèle** : Le modèle **représente les données** de l'application et les règles métier associées. Il contient les méthodes qui permettent de manipuler ces données, ainsi que les fonctions de validation, de transformation et de persistance.
- **Vue** : La vue représente l'**interface utilisateur de l'application**, c'est-à-dire la manière dont les données sont présentées à l'utilisateur final. Elle est responsable de l'affichage des données, de leur mise en forme et de leur organisation.
- **Contrôleur** : Le contrôleur agit comme l'**intermédiaire** entre le modèle et la vue. Il gère les interactions utilisateur (requêtes), récupère les données du modèle et les transmet à la vue.



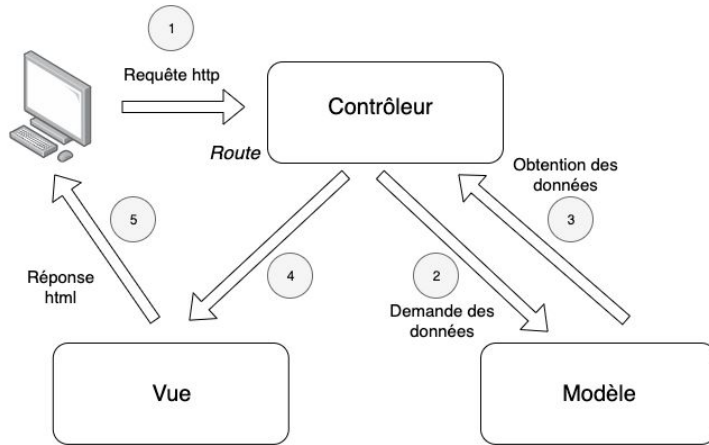
# Importance de la séparation des responsabilités

- La séparation des responsabilités entre le modèle, la vue et le contrôleur est une pratique importante dans le développement web.
- Elle permet de **faciliter la conception, la maintenance et l'évolution** des applications, en favorisant la réutilisation et la modularité du code.
- Elle permet également d'**améliorer la lisibilité et la compréhensibilité du code**, en le découpant en composantes indépendantes et cohérentes.
- Enfin, elle **facilite la collaboration entre les développeurs**, en leur permettant de travailler sur des parties distinctes de l'application sans se marcher sur les pieds.





# MVC



## Exemple pour une page article

- 1. Un utilisateur demande un article
- 2. Le contrôleur `ArticleController.php` récupère les informations de la requête (ex: `id_article`) et fait la demande au modèle `Article.php` pour récupérer les données
- 3. `Article.php` retourne les données à `ArticleController.php`
- 4. `ArticleController.php` retourne les données à la vue et charge la vue `article.php` qui contient le html
- 5. La vue complète est retournée à l'utilisateur