

# L'architecture MVC

# Contenu

- **L'architecture MVC**
- **Implémentation avec l'application Liste de contacts**

# L'architecture MVC

# Introduction

- **Au dernier cours, nous avons vu un premier exemple d'application utilisant une base de données**
- **Nous avons un script PHP pour chaque page de l'application**
- **Chacun de ces scripts PHP gère :**
  - Les requêtes à la base de données
  - La logique
  - L'affichage

# Séparation des responsabilités

- **Dans une application professionnelle, on veut séparer l'accès aux données du traitement et de l'affichage**
- **Le code est ainsi plus facile à comprendre et à maintenir**

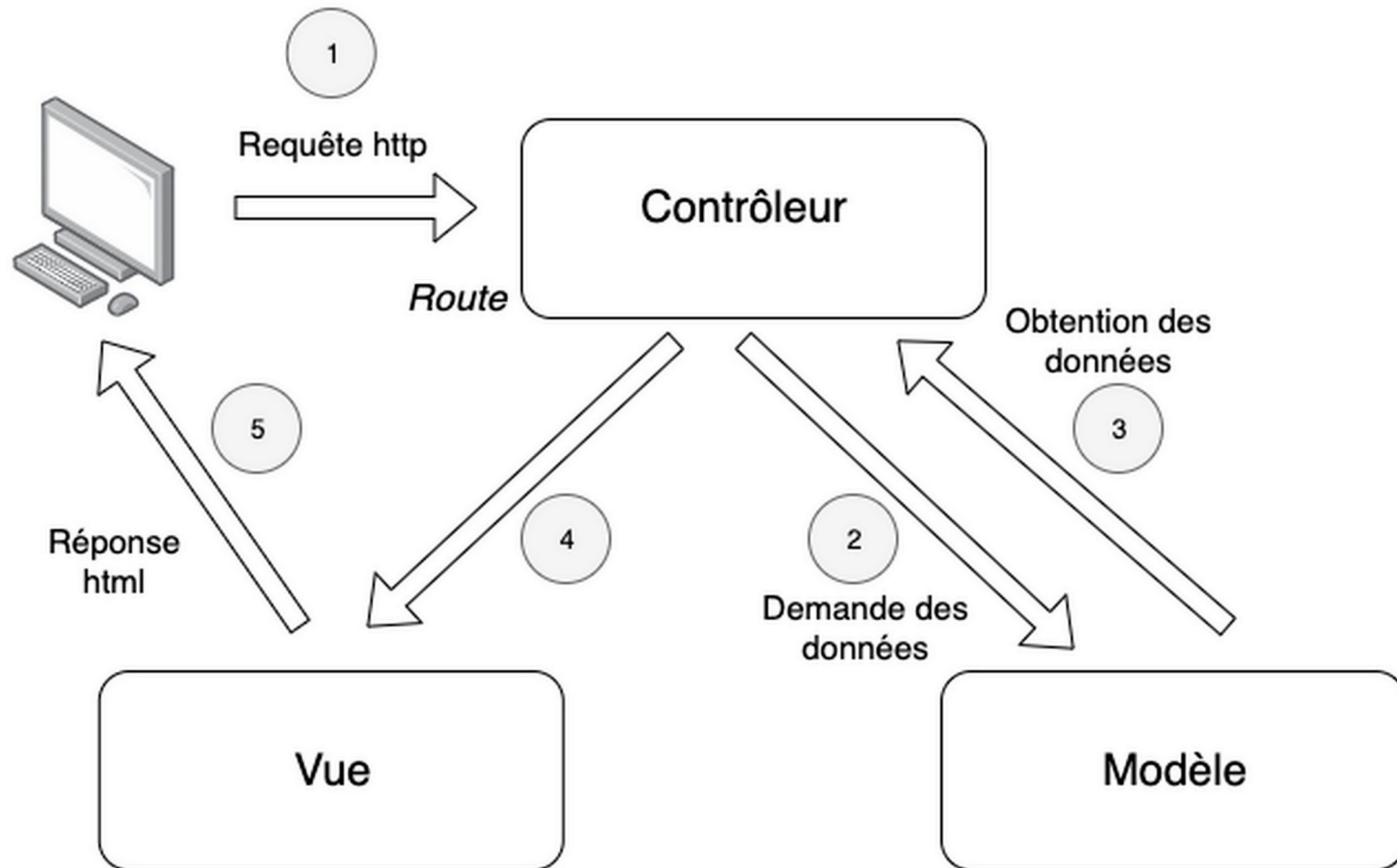
# MVC

- L'architecture **MVC** est une façon fréquente d'organiser le code d'une application pour atteindre cet objectif
  - Signifie « Modèle-vue-contrôleur »

# MVC

- **Consiste à diviser le code pour une page en trois parties distinctes:**
  - Le **modèle**, qui gère les requêtes à la base de données
  - La **vue**, qui gère l'affichage
  - Le **contrôleur**, qui fait le lien entre la vue et le modèle et effectue les traitements
    - Ex: valider les données reçues en GET ou POST, effectuer des calculs, etc.

# MVC



© Benoît Prieur / Wikimedia Commons / CC BY-SA 4.0



# MVC

- L'architecture MVC est souvent implémentée en utilisant **la programmation orientée objet**
- Il s'agit d'ailleurs d'un **patron de conception** (***design pattern***)
  - C'est-à-dire une façon courante d'utiliser des objets en réponse à un problème courant
  - [https://fr.wikipedia.org/wiki/Patron\\_de\\_conception](https://fr.wikipedia.org/wiki/Patron_de_conception)





























# Implémentation avec l'application Liste de contacts

# Implémentation

- Dans l'exemple du cours d'aujourd'hui, nous allons revoir l'application du dernier cours, adaptée en MVC
- Autre nouveauté: nous allons désormais utiliser des noms en anglais (noms de variables, classes, fonctions, etc) afin de produire un code plus professionnel

## Liste de contacts

[Ajouter un contact](#)

Nom	Prénom	Actions
Delisle	Gaétan	 
Delisle	Huguette	 
Paquette	Serge	 
Paquette	Micheline	 
Croteau	René	 
Rondeau	Denis	 
Bouchard	Patrick	 
Arcand	Marc	 
Cantin	Julie	 
Talbot	Louise	 
Boissonneau	Jean-Guy	 
Brodeur	Claudio	 
Michaud	Caroline	 
Laverdière	Anne-Sophie	 

# Modèle

- **Nous aurons un seul modèle, appelé ContactModel**
  - Défini dans « models/contacts.php »
- **Classe offrant les méthodes suivantes:**
  - `getAll()`: Retourne tous les contacts de la base de données
  - `get($id)`: Retourne un contact spécifique
  - `insert($firstName, $lastName, $phoneNumbers, $addresses, $emailAddresses)`: Insère un contact dans la base de données
  - `update($id, $firstName, $lastName, $phoneNumbers, $addresses, $emailAddresses)`: Met à jour un contact dans la base de données
  - `delete($id)`: Supprime un contact sur la base de données

# Vues

- **Nous aurons une vue pour chaque page**
  - `views/list.php`: Liste des contacts
  - `views/display.php`: Afficher un contact
  - `views/add.php`: Ajouter un contact
  - `views/edit.php`: Modifier un contact
- **Chaque vue sera un fichier PHP contenant directement du HTML dans lesquelles les données fournies par le contrôleur seront insérées**

# Contrôleurs

- **Il y aura un contrôleur pour chaque page, ainsi que pour la suppression**
  - ListController
  - DisplayController
  - AddController
  - EditController
  - DeleteController

# Contrôleurs

- **Chaque contrôleur possèdera 2 méthodes**
  - `handle`: Valide les paramètres GET, récupère les données voulues depuis le modèle, puis affiche la vue
  - `handlePost` (*facultatif*): Valide les données soumises depuis un formulaire, puis appelle les méthodes appropriées du modèle
- **Puisque les différents contrôleurs possèderont les mêmes méthodes, nous les ferons hériter d'une classe parente *Controller***

# index.php

- Le fichier *index.php* vérifiera la valeur du paramètre GET « action » (`$_GET['action']`)
- La valeur du paramètre déterminera le contrôleur à instancier (et donc la page à afficher)
- Actions possibles:
  - list
  - display
  - add
  - update
  - delete
- Par défaut, l'action *list* sera assumée



# index.php

- **Exemples d'URLs:**

- `index.php?action=list`
- `index.php?action=add`
- `index.php?action=display&id=27`
- `index.php?action=edit&id=27`
- `index.php?action=delete&id=27`

# Fin de la présentation

Des questions?

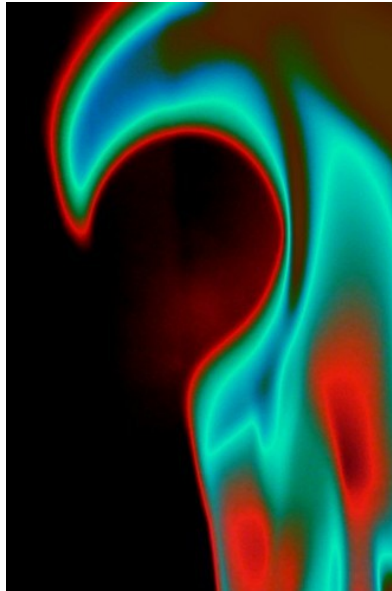


Image par limecools (CC BY-NC-SA 2.0)