PWS - MVC

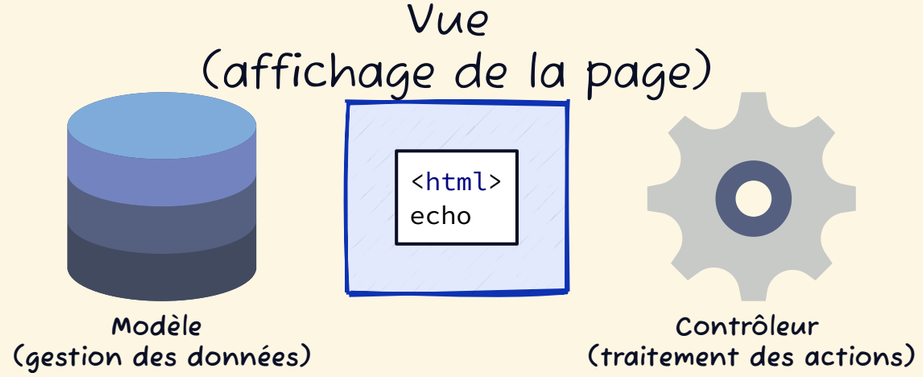
-

Modèle Vue Contrôleur

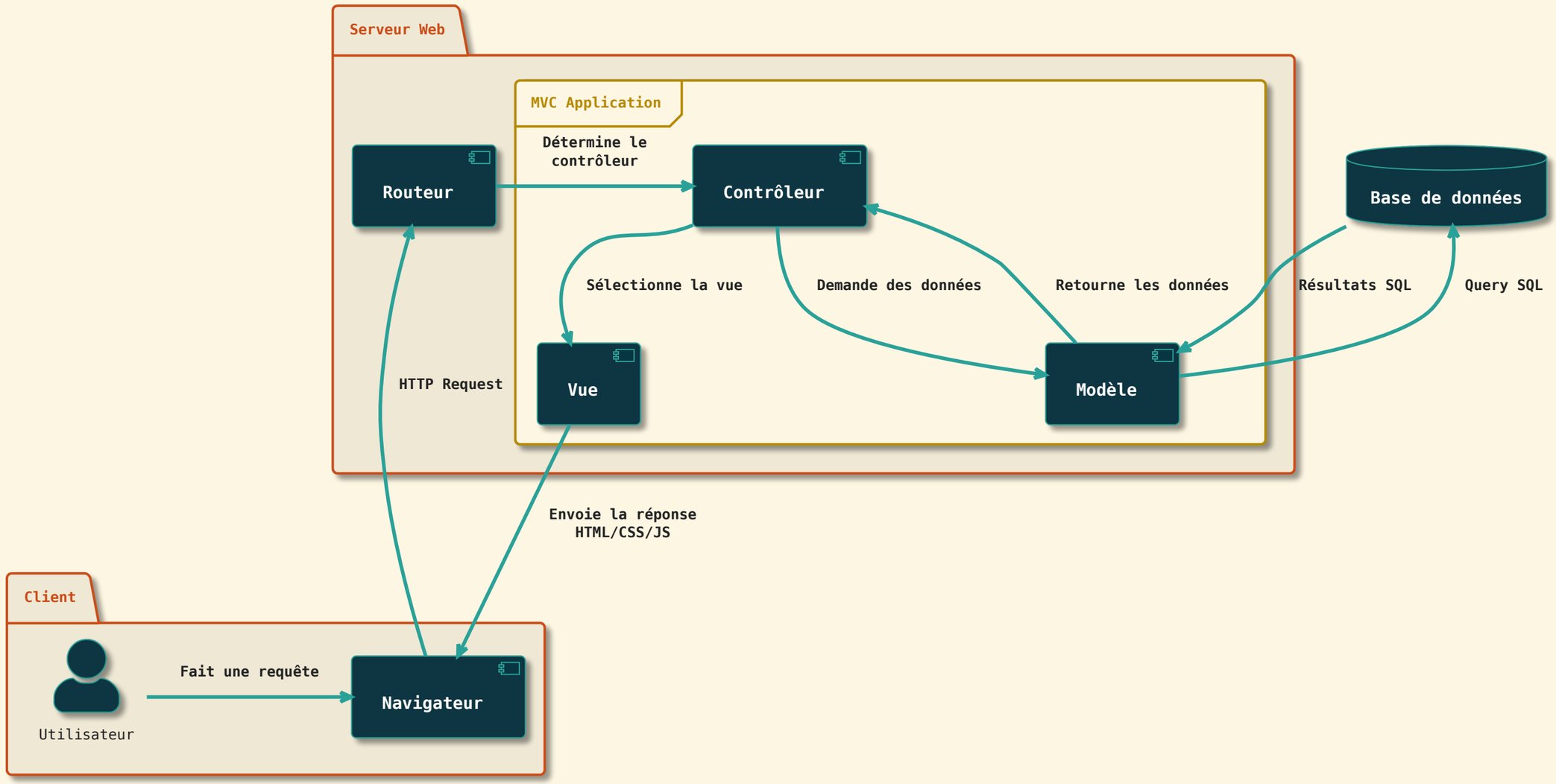
Pierre Blarre - Florian Rodriguez

Le design pattern MVC (Model-View-Controller)

- C'est l'organisation du code d'une application permettant de séparer:
  - la logique métier: **le modèle** (la gestion des données)
  - la présentation: **la vue** (l'affichage et sa gestion)
  - le contrôle: **le contrôleur** (le traitement des actions de l'utilisateur)
- Cela permet de rendre le code plus lisible, plus maintenable et plus évolutif.
- Cette architecture implique aussi la création d'un **routeur** qui va rediriger les requêtes HTTP vers les bonnes actions du contrôleur.



- Le modèle est indépendant des autres modules.
- La vue est dépendante du modèle. Elle interroge celui-ci pour en afficher une représentation.
- Le contrôleur dépend de la vue et du modèle : la vue comporte des éléments visuels que l'utilisateur peut actionner.
- Le contrôleur répond aux actions effectuées sur la vue et modifie les données du modèle.



- Le motif MVC a été créé dans le but de mettre en œuvre des interfaces utilisateur.
- Le cycle action → mise à jour → affichage induit par ce patron est bien adapté aux applications web.
- Le patron impose la séparation des sujets, et les balises HTML sont ainsi confinées aux vues, ce qui améliore la maintenabilité de l'application.

Flux de traitement

En résumé, lorsqu'un client envoie une requête à l'application :

- la requête envoyée depuis la vue est analysée par le contrôleur.
- le contrôleur demande au modèle approprié d'effectuer les traitements et notifie à la vue que la requête est traitée.
- la vue notifiée fait une requête au modèle pour se mettre à jour (par exemple affiche le résultat du traitement via le modèle).

Exemple de structure de fichiers d'une application mvc

- L’application appelle **toujours** `index.php`
- `index.php` analyse la requête HTTP (l’url demandée) et “appelle” le contrôleur et la méthode concernés.
- Le contrôleur, en fonction de la méthode demandée, va instancier des modèles puis appeler une vue qui sera renvoyée au client.
- Le fichier `bootstrap.php` peut contenir des informations de configuration, comme par exemple les identifiants de base de données, mais aussi la fonction de chargement automatique des classes (`spl_autoload`).
- Pour les vues, en général on va créer une template principale contenant les éléments qui apparaissent sur toutes les pages du site (ex: en-tête du site avec logo et menu, pied de page, sidebar, etc.)
- Dans le répertoire des modèles, on aura nos modèles de données et nos classes “repository” qui permettent de récupérer les données depuis une base de données.