

# Bachelier en Informatique de Gestion

## Web : principes de base Projet de Développement Web

Enseignement supérieur économique de type court

Code FWB : 7534 29 U32 D1, 7534 30 U32 D3

Code ISFCE : 4IWPB, 4IPW3



# Table des matières

## Généralités

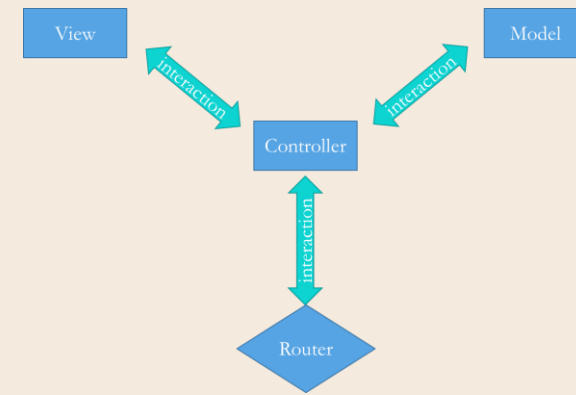
- 01. Introduction au web
- 03. Outils
- 05. Format XML
- 06. Format JSON

## Front-End

- 12. Structure HTML
- 13. Formulaire HTML
- 14. Mise en forme CSS
- 15. Adaptabilité
- 17. Javascript
- 18. Bibliothèque jQuery
- 19. Composant Vue.js

## Back-End

- 21. Middleware PHP
- 22. Traitement du formulaire
- 23. [Architecture MVC](#)
- 24. Données SQL
- 25. Données NoSQL
- 27. Requête asynchrone



## 23. Architecture "Model View Controller"

Architecture Web

Modèle

Vue

Contrôleur

Router

Avantages

Inconvénients

Historique

Implantation web

Architecture Desktop



= BUROTIX 0

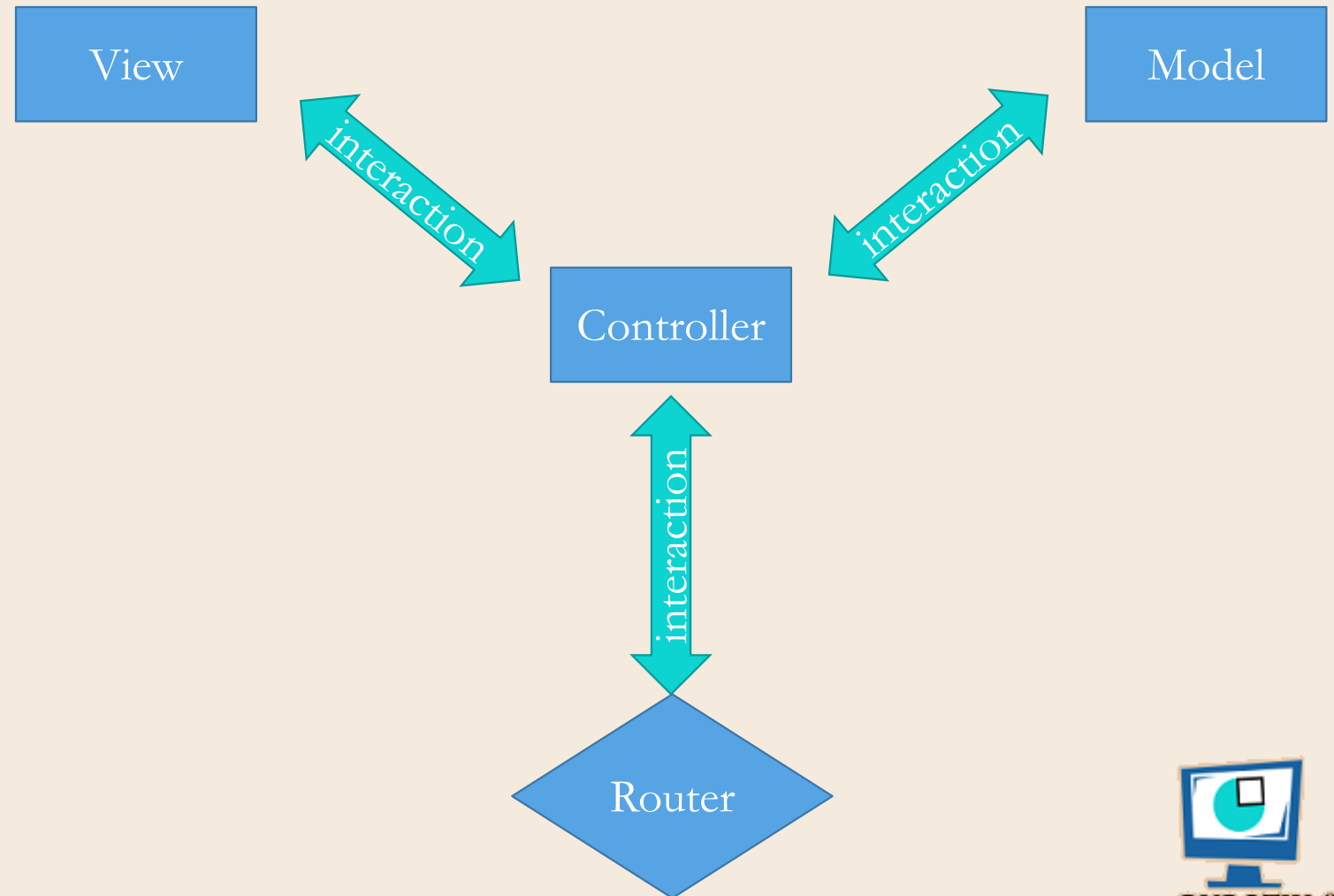
# Architecture MVC WEB



= BUROTIX ()

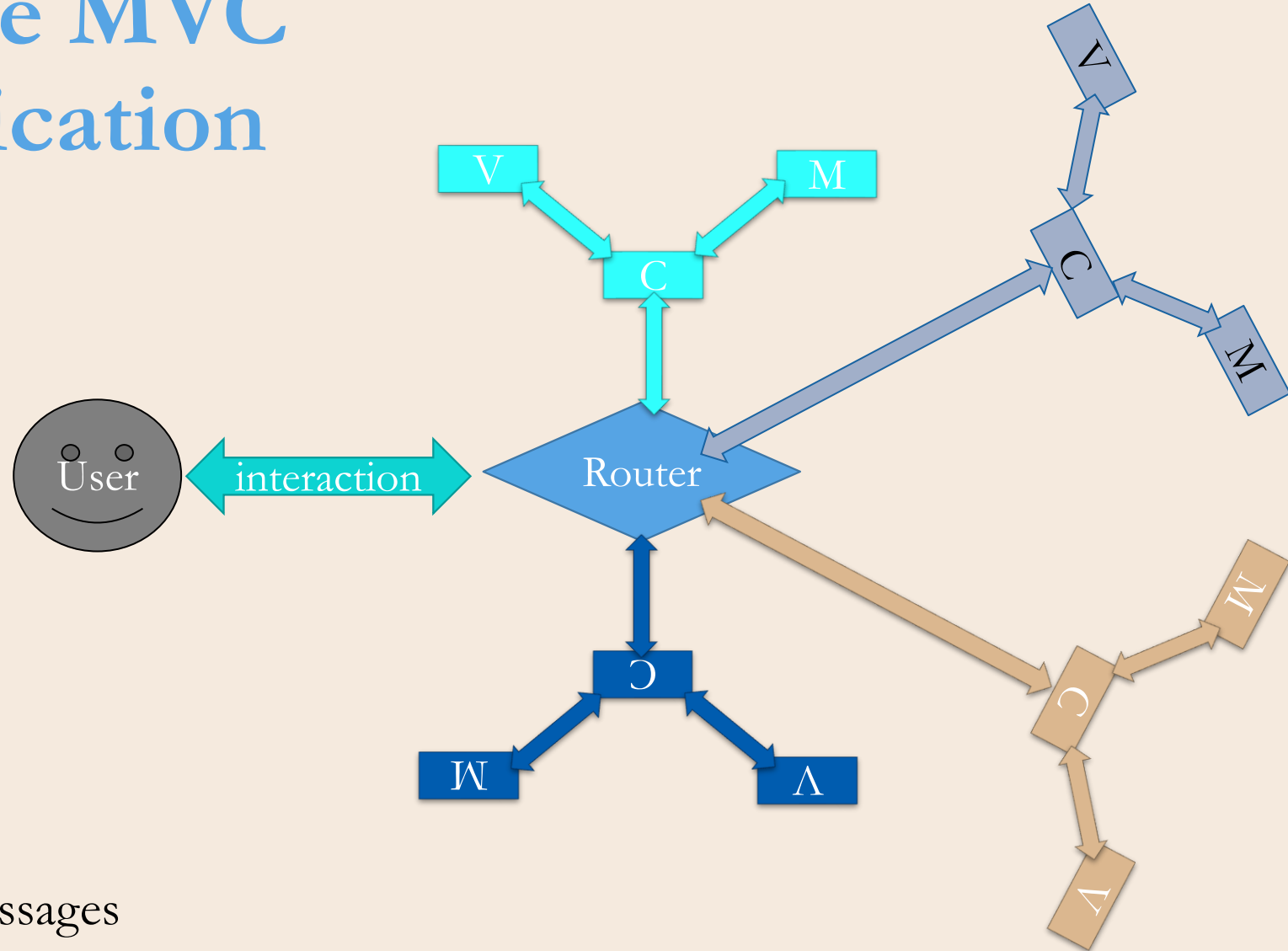
# Architecture MVC pour un composant de l'application

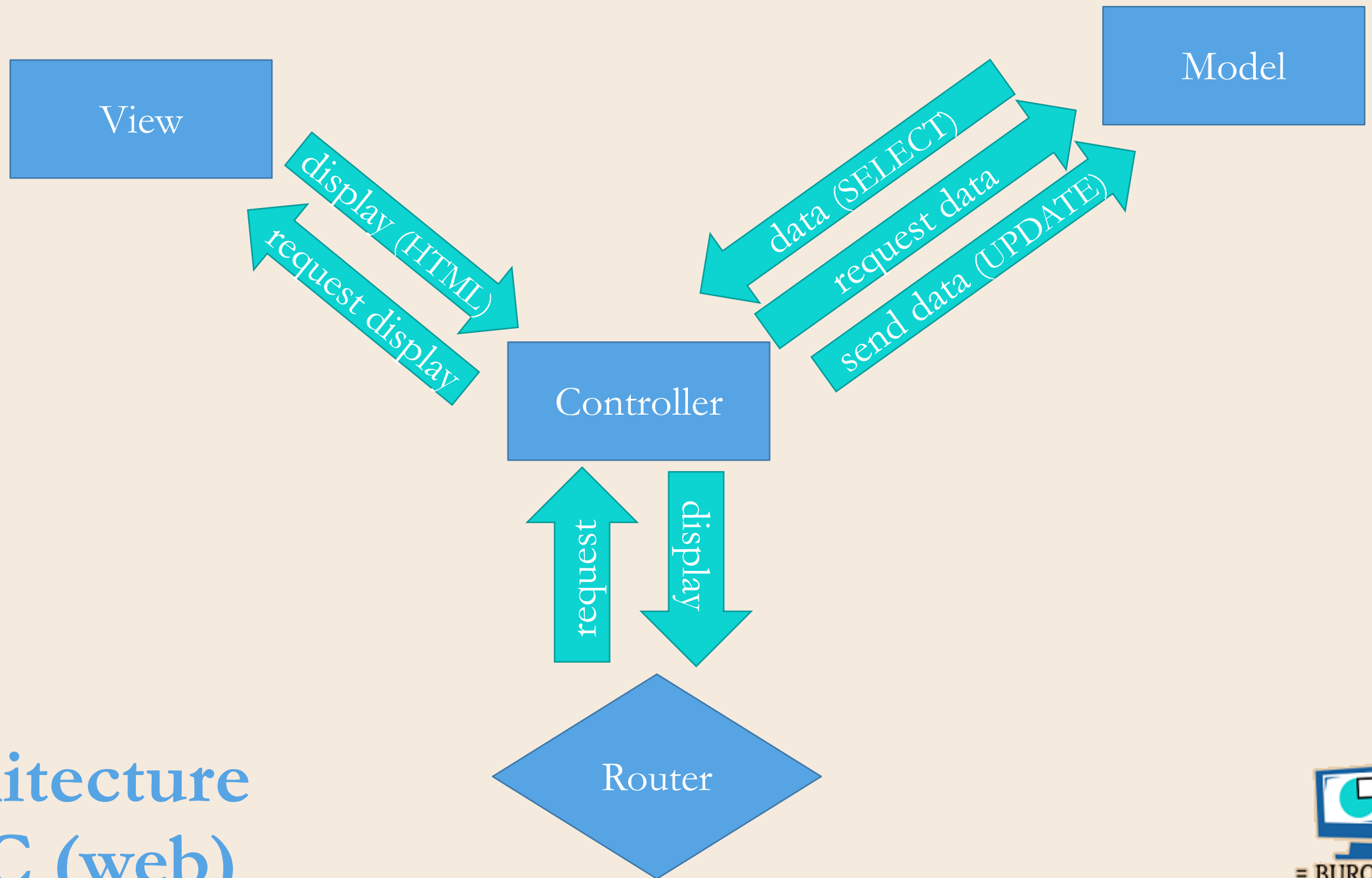
- Model
  - Données
  - Accès aux données
  - Logiques des données
- View
  - Présentation des données
  - Saisie des données
- Controller
  - Interaction avec Model
  - Interaction avec View
  - Interaction avec Router



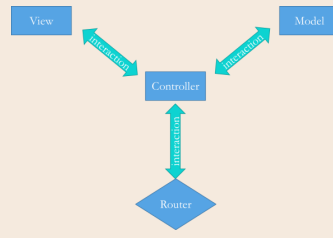
# Architecture MVC pour l'application

- plusieurs Models
- plusieurs Views
- plusieurs Controllers
- un Router
  - Aiguillage des messages





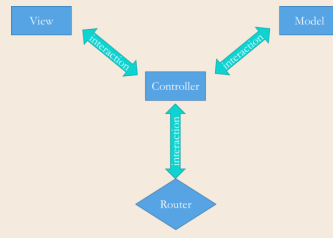
# Architecture MVC (web)



# Model

- Indépendant des autres modules
- Gérer toutes les données
- Gérer la logique des données
  - Validation
  - Lecture
  - Enregistrement
- Interagir avec **Controller**
  - **Controller** paramétrise les requêtes pour SELECT
  - **Controller** fournit les données pour UPDATE, INSERT, ..
- Exemple : application bancaire
  - Fichier des clients
  - Liste des dépôts
  - Vérification : les retraits ne dépassent pas la limite de crédit

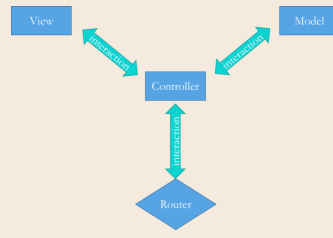




# View

- Indépendant des autres modules
- Présenter les données via des éléments visuels
  - Texte, Table, Graphique, ...
  - Balises HTML, Javascript, CSS
  - Tkinter, Canvas
- Préparer la mise à jour des données
- Interagir avec **Controller**
  - **Controller** envoie les données à **View**
  - **View** met en forme les données reçues

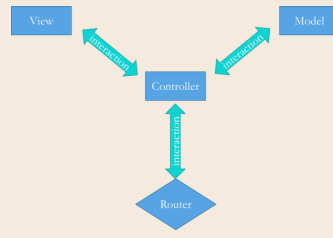




# Contrôleur / Controller

- Dépendant de Model et View
- Traiter les actions de Router
  - Callbacks
  - Validation des données d'un formulaire
- Modifier Model
  - Mise à jour (UPDATE)
  - Ajout (INSERT)
  - Suppression (DELETE)
- Modifier View
  - Suite aux modifications de Model
  - Suite aux actions de Router





# Routeur / Router

- Dépendant des différents Controller
- Traiter les actions de User
  - Évènements souris et clavier
- Sélection du composant à appeler
- Interagir avec Controller



= BUROTIX ()



# Architecture MVC en pratique

## Comment écrire une application web ?

framework AWebWiz

source d'inspiration : CodeIgniter, simplifié pour les débutants

auteur : Alain Wafflard

# MVC en pratique

## AWebWiz

### structure

- └ **/** : racine de l'application

- └ **/app** : code (PHP) de l'application

- └ **/model**

- └ **/view**

- └ **/controller**

- └ autres : **/helper**, **/config**, ...

- └ **/asset** : toute l'information privée

- └ **/database**

- └ **/static\_content**

- └ **/public** : racine du site web

- └ **/css**, **/js** : l'information nécessaire au navigateur

- └ **internal** : développement interne

- └ **bootstrap** : par exemple

- └ **/media** : matériel à destination de l'utilisateur (pdf, ...)

- └ **index.php** : router

- └ **/external** : package externes

# MVC en pratique : AWebWiz, structure

- Chaque composant de l'application est décomposé en sous-composants (parfois optionnels)
  - composant de nom *comapp*
  - */app/model/comapp.php* : fonctions gérant l'accès aux données
  - */app/view/comapp.php* : fonctions générant le code HTML
  - */app/controller/comapp.php* : fonctions traitant les input de l'utilisateur, et les output vers l'utilisateur
  - */app/asset/database/comapp.php* : toute l'information privée
  - etc.

# MVC en pratique : /app/model

- Dans le répertoire "**/app/model**", on trouve :
  - un fichier PHP par composant : *catalogue.php*, *login.php*
  - chacun contient une bibliothèque de fonctions PHP
  - ces fonctions gèrent l'accès et la manipulation de ces données
- Remarques :
  - tout le code PHP est encapsulé dans des fonctions;
  - on crée autant de fonctions que nécessaire;
  - interdit ici : langage HTML, superglobals PHP;
  - MODEL ne fait jamais appel ni à VIEW ni à CONTROLLER.



# MVC en pratique : /app/view

- Dans le répertoire **" /app/view "**, on trouve :
  - un fichier PHP par composant : *catalogue.php*, *login.php*
  - chacun contient une bibliothèque de fonctions PHP
  - ces fonctions créent le code HTML du composant
- Remarques :
  - tout le code PHP est encapsulé dans des fonctions;
  - on crée autant de scripts que nécessaire;
  - interdit ici : superglobals PHP, langage SQL, manipulation de fichiers ;
  - VIEW ne fait jamais appel ni à MODEL ni à CONTROLLER





# MVC en pratique : /app/controller

- Dans le répertoire "**/app/controller**", on trouve :
  - un fichier PHP par composant : *catalogue.php*, *login.php*
  - chacun contient une bibliothèque de fonctions PHP
  - chacun contient obligatoirement une fonction d'entrée
    - **main\_<composant>()** : *main\_Login()*, *main\_home()*, ...
  - ces fonctions gèrent la logique du composant
    - interaction utilisateur : superglobals
    - appel à MODEL : création ou lecture des données
    - appel à VIEW : création des contenus HTML
  - ces fonctions manipulent les superglobals :
    - **\$\_GET**, **\$\_POST**, **\$\_COOKIES**, **\$\_SESSION**, etc.



# MVC en pratique : /app/controller

- Remarques :
  - tout le code PHP est encapsulé dans des fonctions;
  - on crée autant de scripts que nécessaire;
  - interdit ici : langage SQL, langage HTML ;
  - CONTROLLER fait appel à VIEW et à MODEL ;



# MVC en pratique : /asset

- Dans le répertoire **" /asset"**, on trouve :
  - toute information privée, càd non partageable avec les utilisateurs
  - les bases de données : *catalogue.csv*, *login.csv*, etc.
  - les “*media*” contenant les images et autres documents relatifs aux produits du catalogue.
- Remarques :
  - on crée autant de répertoires ou de scripts que nécessaire;
  - **interdit ici** : script PHP



# MVC en pratique : /public

- **Racine du site web !**
- Dans le répertoire **"/public"**, on trouve :
  - toute information partageable avec les utilisateurs
  - **/media** : images et documents généraux du site
    - logo, fond de pages, illustrations, icônes, conditions générales, rapport annuel, ...
  - **/css** : feuilles de style
  - **/js** : code javascript
- Remarques :
  - on crée autant de répertoires ou de scripts que nécessaire;
  - **interdit ici** : script PHP

# MVC en pratique : le router "index.php"

- Dans le répertoire `"public"`, on trouve le script `router` de nom `"index.php"` dont le rôle est de :
  - Aiguiller les requêtes de l'utilisateur
    - clic sur un lien
    - envoi d'un formulaire
  - Sélectionner le composant concerné par cette requête
  - Appeler le Controller de ce composant
    - importance de la fonction Controller `main_<composant>()`
- Remarques :
  - il y a un seul router et son nom est imposé : `index.php` ;
  - ce fichier ne doit pas être modifié ;
  - toutes les pages du site sont supposées être appelables depuis `Router` ;
  - le code PHP du `Router` n'est pas encapsulé dans des fonctions

# Exos

# Exo 01 : log-in, log-out

- Démonstration de l'architecture générale d'une app en MVC
- Reprenez l'exercice "log-in, log-out"
- Fichier de départ : **exo01\_login\_start.php**
- A réécrire en MVC.
- Composants
  - login-logout form
  - page d'accueil (juste "hello world")
- Solution : **exo01\_login\_solution.zip**
  - à télécharger et à disséquer



# Exo 03 : catalogue VOO

- Fichier de départ :  
exo03\_catalogue\_start.nhn
- Réécrivez-le en MVC.
- Solution (TBC) :  
exo03\_solution.zip

**VOO : Catalogue des produits**

Enregistrer mon panier Voir mon panier Acheter

Votre prénom :  Votre nom :

Chercher dans le catalogue

<b>TOUDOU mobile</b> <ul style="list-style-type: none"><li>• Appels : 60 min</li><li>• SMS : illimité</li><li>• Internet : 300 Mo (4G)</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>	<b>TATOO mobile</b> <ul style="list-style-type: none"><li>• Appels : illimité</li><li>• SMS : illimité</li><li>• Internet : 600 Mo (4G)</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>	<b>TOODATA tablette</b> <ul style="list-style-type: none"><li>• Appels : inclus</li><li>• SMS : inclus</li><li>• Internet : 1 Go (4G)</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>	<b>Téléphone Blabla</b> <ul style="list-style-type: none"><li>• Appels : 0,10 euro/min</li><li>• SMS : inclus</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>
<b>Internet Un Peu</b> <ul style="list-style-type: none"><li>• Volume : 1000 Mo</li><li>• Vitesse : 30 Mbps</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>	<b>Internet Beaucoup</b> <ul style="list-style-type: none"><li>• Volume : 10000 Mo</li><li>• Vitesse : 40 Mbps</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>	<b>Internet Passionnément</b> <ul style="list-style-type: none"><li>• Volume : 100000 Mo</li><li>• Vitesse : 80 Mbps</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>	<b>Internet A la folie</b> <ul style="list-style-type: none"><li>• Volume : illimité</li><li>• Vitesse : 140 Mbps</li></ul> <a href="#">Plus d'info</a> <p>Ajouter au panier :</p> <p>À la pièce : 12 € <input type="text"/></p> <p>Groupe de 6 : 60 € <input type="text"/></p>



# Exo 05 : le menu

- Fichier de départ : un des précédents
- Ajoutez-y un menu élémentaire (`<ul>...`)
  - Les liens du menu pointent vers du contenu statique
- Écrivez la gestion de ce menu en mode MVC.
- Solution (TBC) :  
`exo05_solution.zip`



# Exo 06 : login obligatoire et catalogue

- Soit un site avec un catalogue-produit et des utilisateurs identifiés.
- Le catalogue ne s'affiche que si l'utilisateur est identifié.
- Écrivez ce site en MVC.
- Solution (TBC) :  
`exo06_solution.zip`

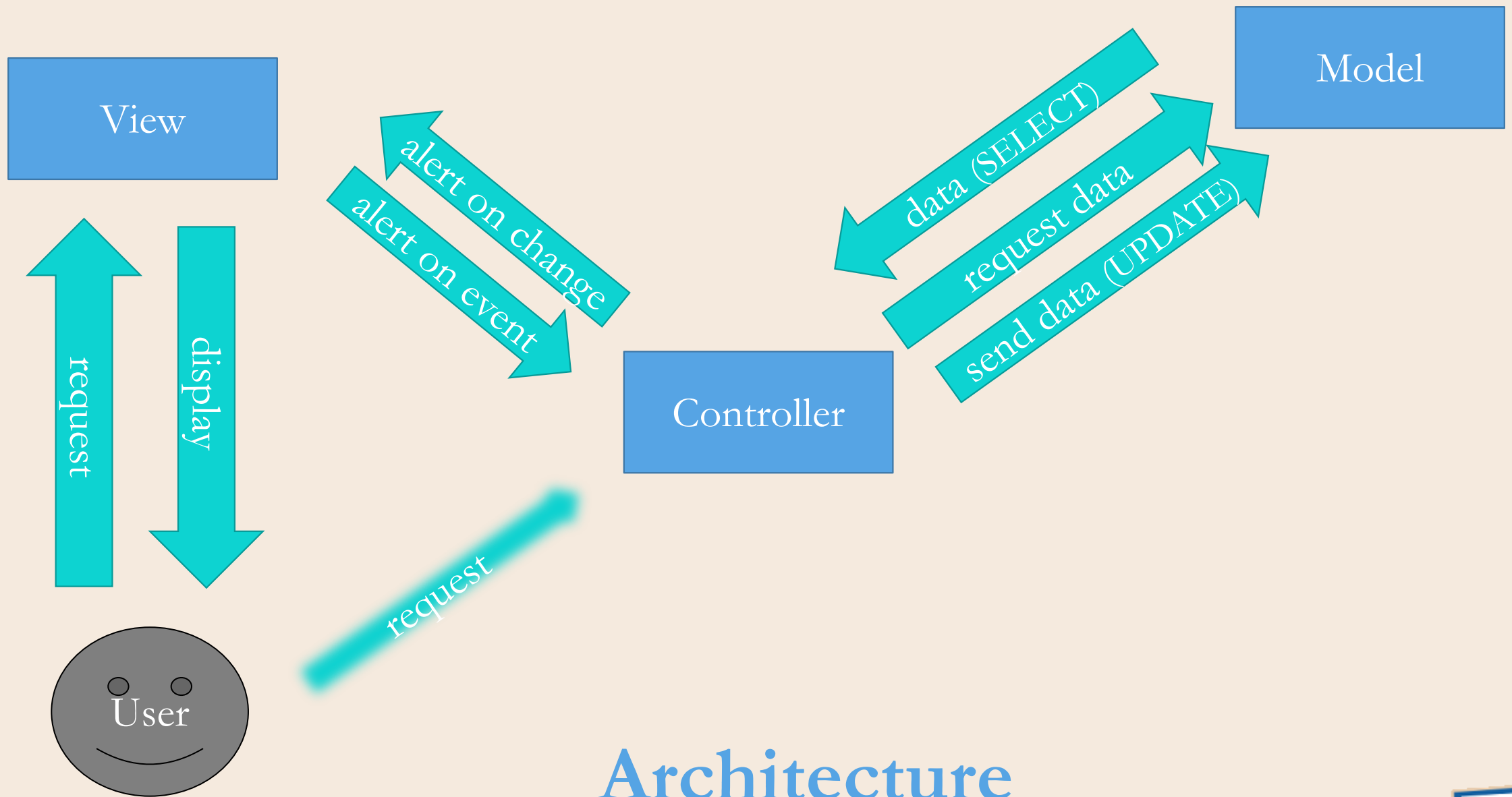


# Architecture MVC Desktop

par ex. en Python Tkinter



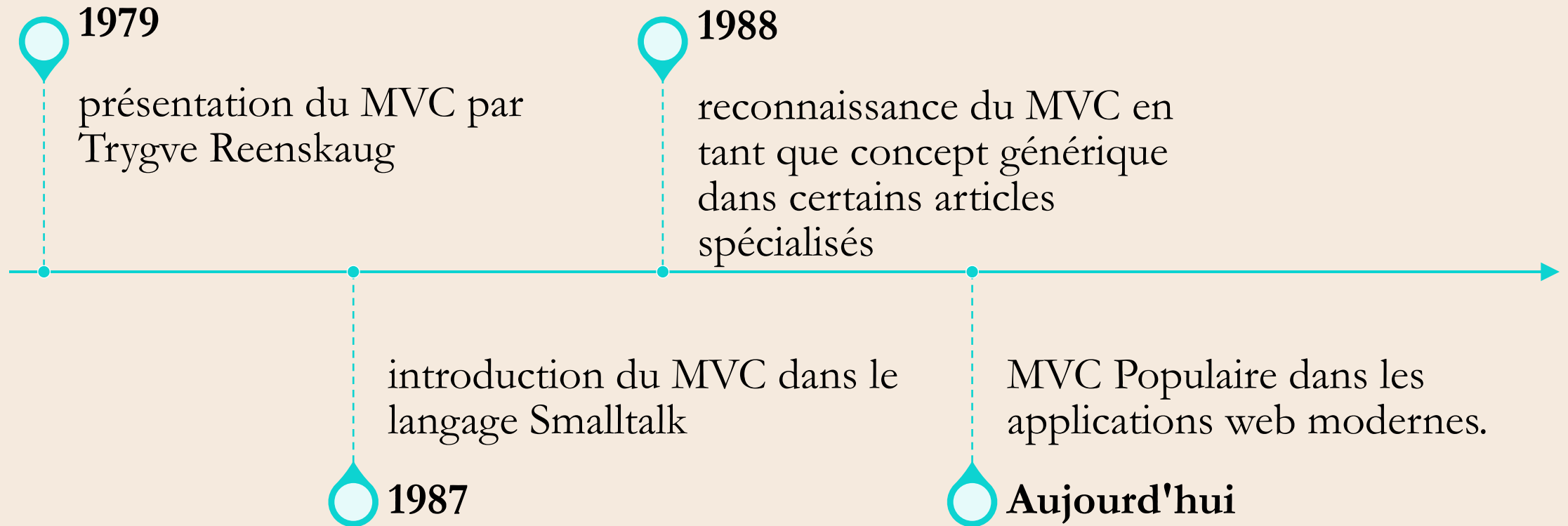
= BUROTIX ()



# Architecture MVC (desktop)

# Conclusions

# Historique



# Des noms !

- Ruby on Rails
- Django
- CakePHP
- Yii
- CherryPy
- Spring MVC
- Catalyst
- Rails
- Zend Framework
- CodeIgniter
- Laravel
- Fuel PHP
- Symphony



# Avantages

- **Maintenance aisée du code**
  - code plus facile à étendre et à développer
  - composant Model testable séparément de l'utilisateur
  - prise en charge facilitée de nouveaux types de web clients
  - différents composants développables en parallèle.
  - migration de base de données facilitée
- **Réduction de la complexité**
  - division de l'application (en modèle, vue et contrôleur)
  - router unique traitant les requêtes de l'utilisateur
  - séparation de business logic et UI logic
- **Meilleur support pour le test-driven development (TDD)**
  - Modularité et indépendance des classes et des objets, donc testables séparément.
- **Bien adapté aux applications web complexes**
  - développement par de grandes équipes de concepteurs et de développeurs
  - facilitation de Search Engine Optimization (SEO)
  - exploitation et amélioration des fonctionnalités proposées par ASP.NET, JSP, Django, etc.



# Inconvénients

- Code difficile à lire, à modifier, à tester unitairement et à réutiliser par un développeur extérieur.
  - augmentation du nombre de lignes de code
- Navigation dans le framework parfois complexe.
  - introduction de plusieurs couches d'abstraction obligeant les utilisateurs à s'adapter à l'architecture du MVC.
  - appropriation de l'application
- ralentie à cause de l'apprentissage
- Difficulté d'appliquer MVC dans un UI moderne
  - architecture non adaptée aux SPA
  - MVC orienté serveur, non orienté client
- Difficulté pour plusieurs programmeurs de mener une programmation parallèle.
- Connaissance de plusieurs technologies nécessaire.

