Module 05 - Pattern MVC, architecture du framework Symfony et premiers pas

Pattern MVC

- Le Pattern Modèle Vue Contrôleur (*Model View Controller*) ;
- Pattern d'architecture logicielle qui existe depuis un moment, notamment utilisé dans les applications desktop SmallTalk dès les années 80.

Fonctionnement par défaut du web

On se souvient que le web est fondé sur 3 technologies :

- Le protocole HTTP
- Les adresses web / URL : une chaîne de caractères décrivant let nom et l'emplacement d'une ressource (adresse d'un document sur le web: protocole, machine, nom de la ressource, etc.), possibilité de créer de la navigation entre les ressources ;
- Le standard HTML: les documents structurés téléchargés par les clients pour accéder à du contenu ;

Le web, à l'origine, était *public* et en lecture seule pour le client : une URL permet à un client d'accéder à une ressource sur une machine distante via le protocole HTTP.

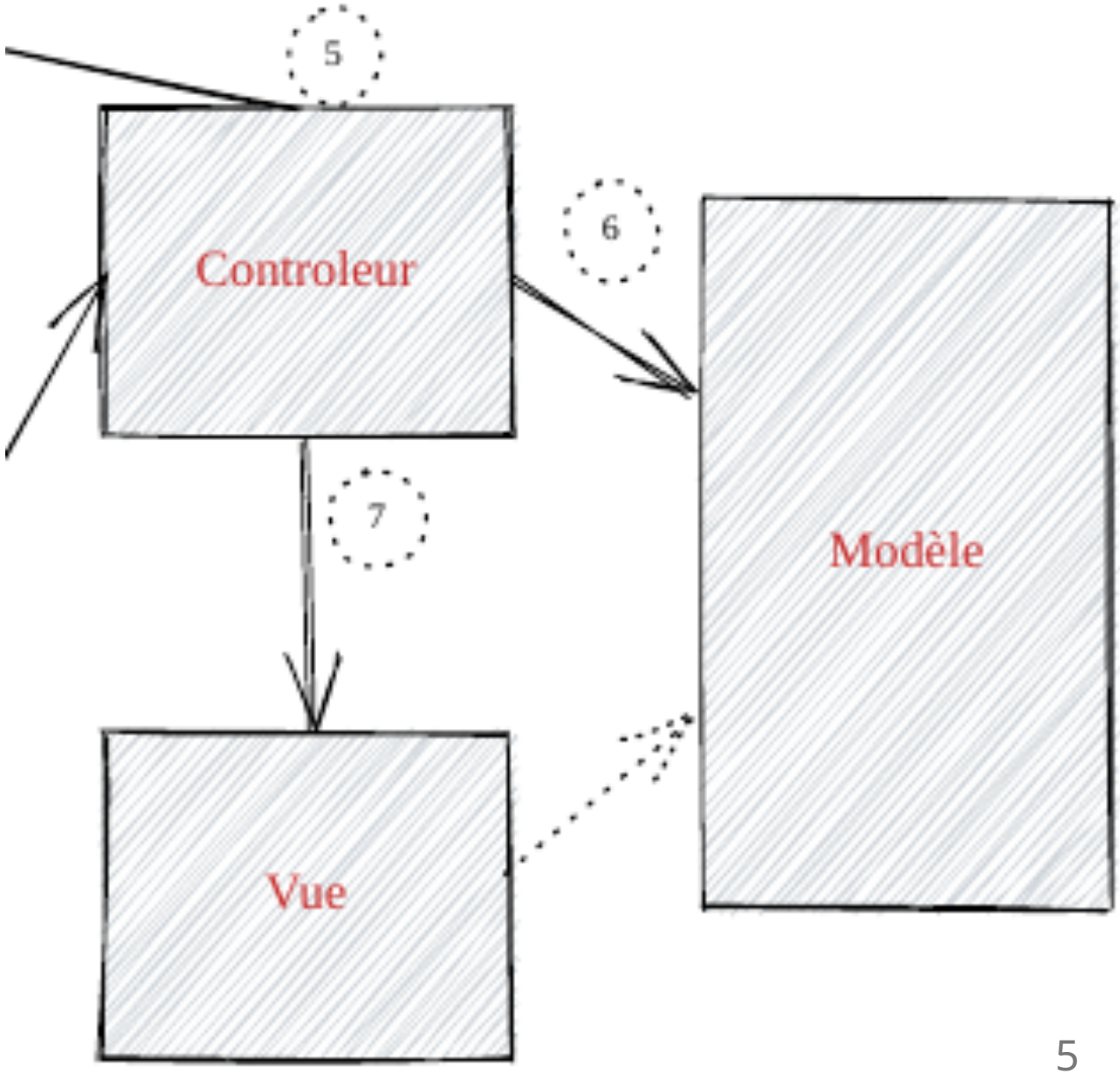
Contrôler l'expérience utilisateur·ice

Aujourd'hui, la majorité des sites web disposent d'une couche applicative (une requête déclenche l'exécution d'un programme pour générer une réponse): on parle de programmation *Common Gateway Interface* (CGI) ou de site web dynamique. Les sites ont besoin d'authentification, d'autorisations, d'appliquer des règles métier, etc. Les sites web sont devenus de réelles applications (web).

On souhaite contrôler les accès aux ressources et aux actions possibles. D'où la nécessité de mettre en place de l'*architecture logicielle* pour organiser cette couche applicative. **Une possibilité est le pattern MVC**.

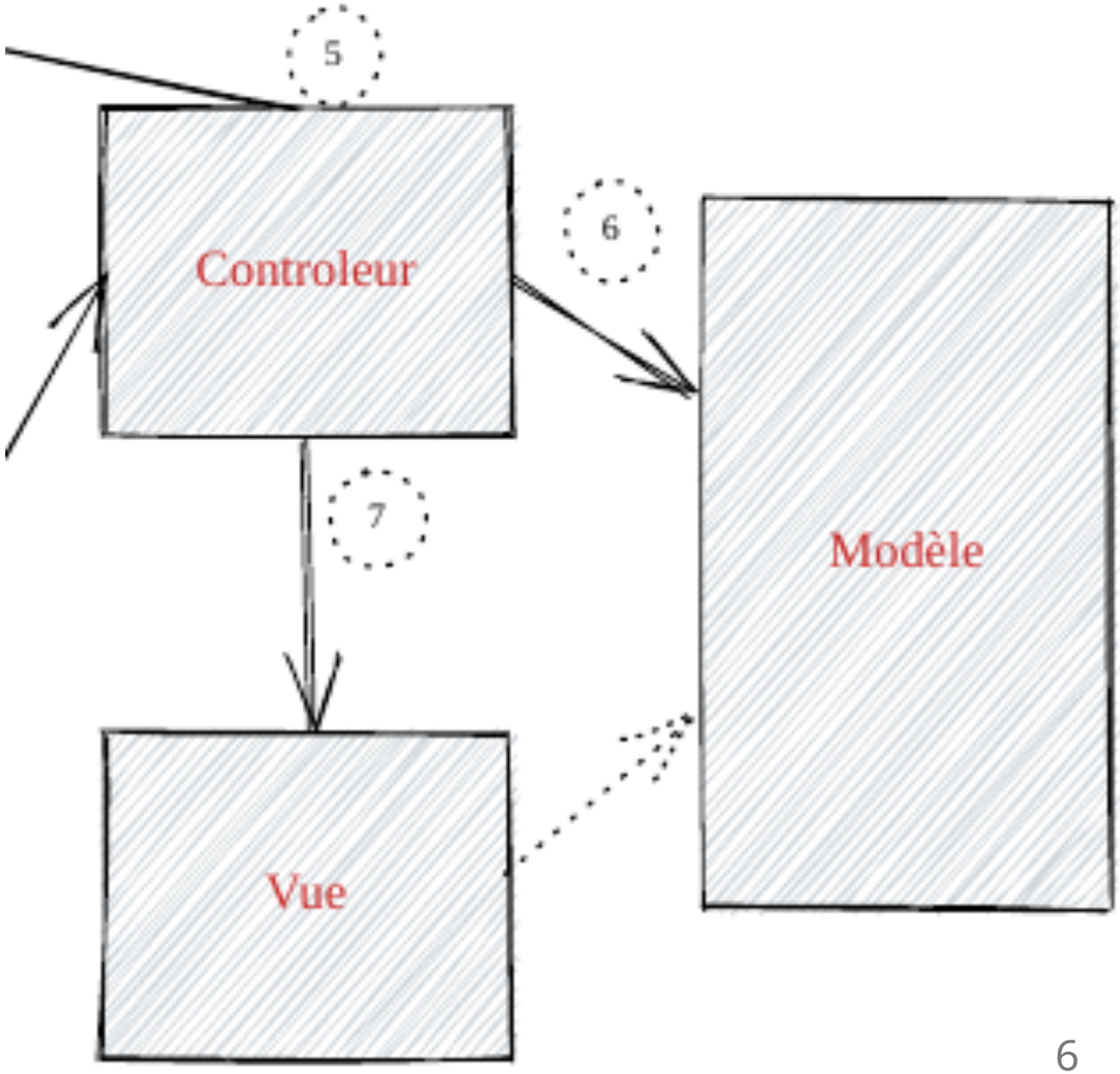
Le Pattern MVC appliqué au web

- Un module, un **rôle**:
  - **Controller** : gère requête du client pour une ressource (URL) et retourne la réponse (input/ output) ;
  - **Model** : Logique métier, use cases et données métiers ;
  - **View** : La réponse retournée au client (web: document HTML + assets) ;



Le Pattern MVC appliqué au web

- **Dépendances** entre les modules :
  - **Controller** : dépend de la View et du Model ;
  - **Model** : **indépendant** ! (également du framework, logique métier !) ;
  - **View** : dépend (souvent, mais pas nécessairement) du Model.



Principes du pattern MVC

- **Le Modèle reste indépendant des autres modules**. C'est votre code métier, votre valeur ajoutée. Il est idéalement indépendant de tout framework ou de toute architecture ;
- La gestion des entrées/sorties est déléguée au Contrôleur (input) ;
- La Vue a en charge la production de la réponse (document à retourner: HTML, JSON, etc.)
- Un même Modèle peut être utilisé par plusieurs Vues ou par plusieurs Contrôleurs;

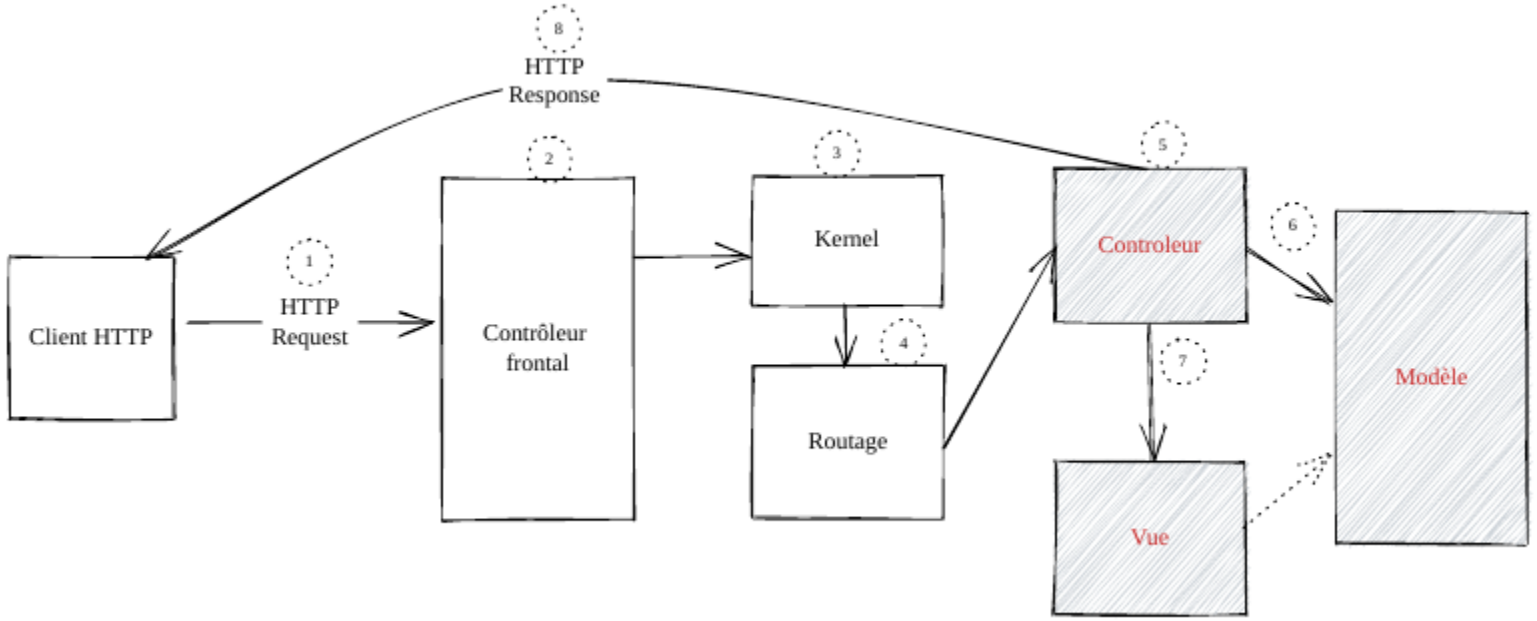
> Il existe d'autres architectures qui sont principalement des variations pour réduire davantage les couplages entre composants, et sont généralement plus sophistiquées, même si la philosophie reste globalement la même. Voir le pattern Model View ViewModel (MVVM), l'architecture hexagonale, la clean architecture (ou architecture en "peaux d'oignon"), etc.

Composants standards d'une application web

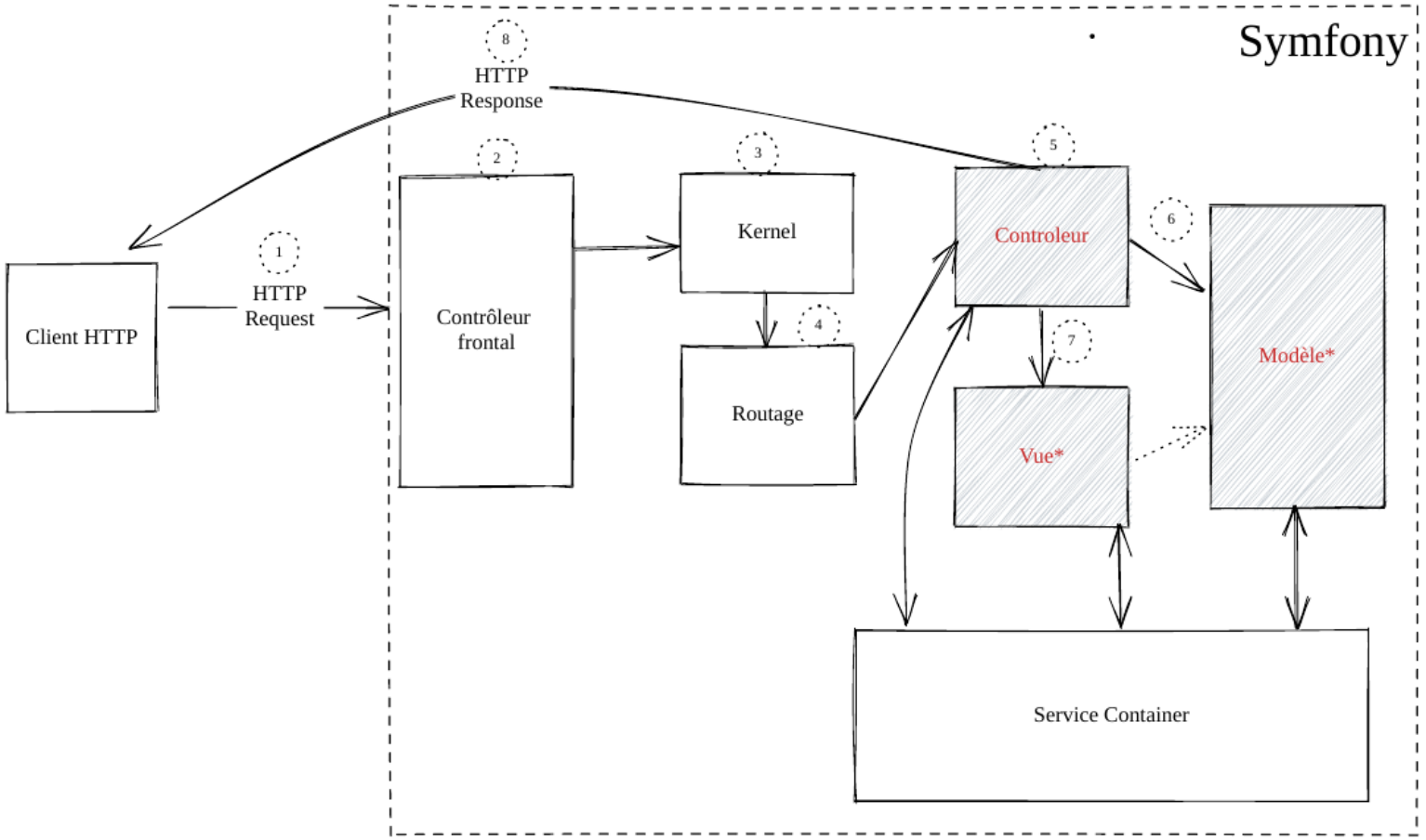
En général, dans toute architecture d'application MVC on trouvera des composants supplémentaires:

- **Contrôleur frontal**: point d'entrée de l'application. Contient le code d'amorçage. Permet d'appliquer des règles de
  gestion. Implémenté soit dans le code applicatif, soit dans la configuration du programme serveur (.htaccess avec Apache, configuration Nginx, etc.);
- **Routeur**: association entre une URL et une fonction (du code). Dans le pattern MVC orienté objet, on associe un contrôleur à
  une ou plusieurs URL ;
- **Kernel**: cœur de l'application web. Au minimum charge les dépendances, amorce le système.

Architecture complète



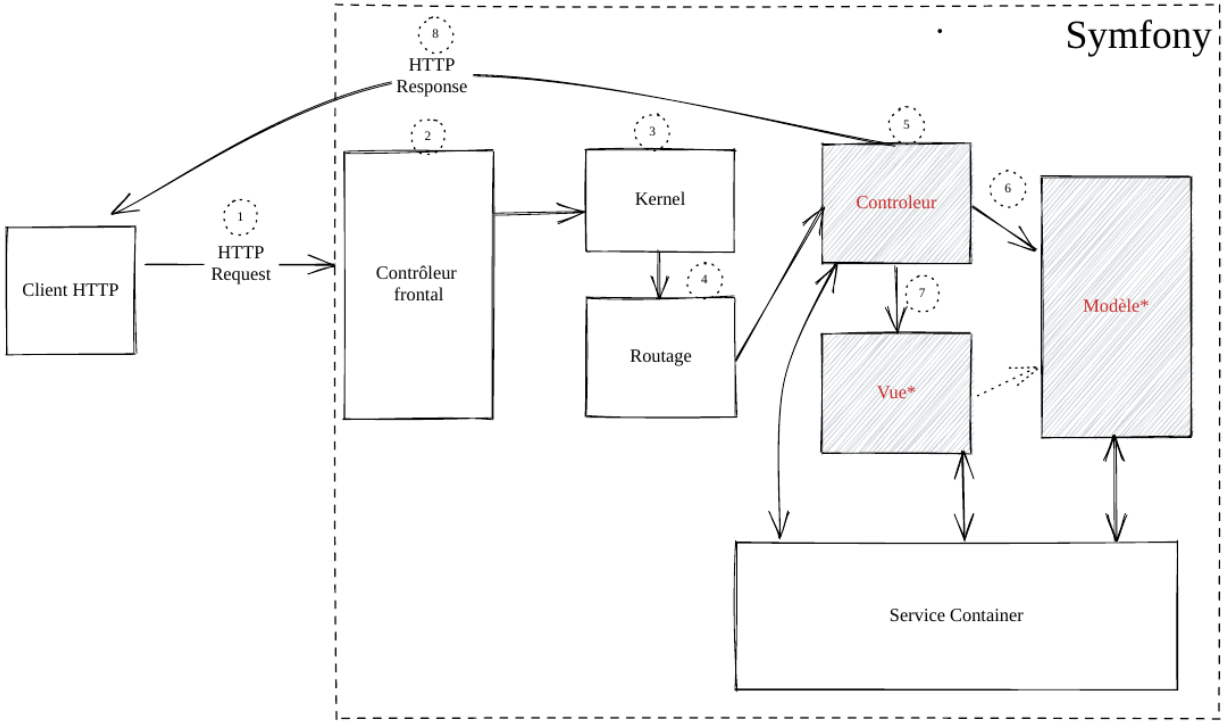
Architecture du framework Symfony



L'architecture de Symfony

- Symfony introduit un composant très important dans son architecture et son fonctionnement: **le Service Container**.
- Le **Service Container** contient des *Services* (des objets PHP), il est mis à disposition du Contrôleur pour l'aider à générer sa réponse HTTP. Il permet d'injecter dans n'importe quel Contrôleur des instances d'objets.

> On reviendra au *Service Container* dans le module 7 sur les services et



L'architecture de Symfony

Symfony ne vous force pas à utiliser le *pattern* MVC, **seule la couche Contrôleur est obligatoire dans le framework**.

> Symfony est également un très bon framework/suite de composants pour créer des applications CLI et des web API (via l'excellent framework API Platform, assemblage de composants Symfony pour construire des services web)

TP : Application MVC en vanilla PHP

Nous allons créer une application web suivant l'architecture MVC en *vanilla PHP*, sans framework. Nous allons mettre en place :

- Le contrôleur frontal ;
- Le routeur/routage
- Les contrôleurs;
- Le modèle
- Les vues.

Le site web expose des articles sur l'URL `/articles?title=titre`. Le site génère une page HTML avec le titre `h1` demandé.

> Une proposition vous est fournie.

TP : Application MVC en vanilla PHP

Objectifs :

- Comprendre concrètement le pattern et le rôle de chaque composant;
- Étudier le code produit: code *système* vs code *métier*, complexité (flot de contrôle, etc.)
- Illustrer l'intérêt d'utiliser un framework, à savoir disposer d'outils pour se concentrer sur la complexité du code métier et plus celle du code système.