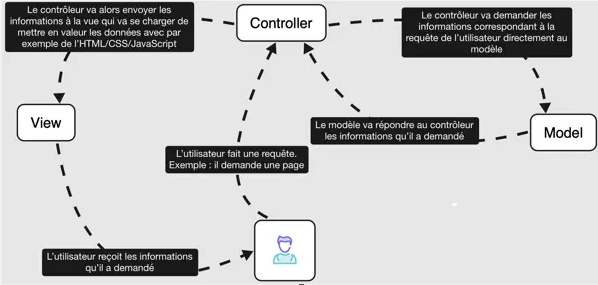
**THE DESIGN PATTERNS**

1. **MVC :**

Ici, le Pattern MVC va répondre à une problématique récurrente de tout développeur : structurer correctement et efficacement le code de son application.



Fonctionnement du Modèle-Vue-Contrôleur

* L’utilisateur fait une demande au contrôleur et il reçoit comme réponse une vue.
* Le contrôleur capte les requêtes des utilisateurs et envoie les informations qu’il récolte au travers d’une vue
  + Il est une sorte d’orchestre dans le modèle MVC
  + Il est dépourvu de logique
  + Il récupère les requêtes, demande des informations au modèle et les envois à la vue
* Le modèle contient toute la logique de l’application : la récupération des données, la modification des données, le traitement entre les données, les calculs, etc.
  + Il ne décide en aucun cas comment les informations sont appelées
* La vue contient les informations que le contrôleur lui a transmises
  + Il n’affiche que les informations

1. **Factory Method :**

Le factory pattern décrit une approche de programmation qui vous permet de créer des objets sans avoir à spécifier la classe exacte de ces objets. Cela permet d'échanger l’objet créé de manière souple et pratique. Pour la mise en œuvre, les développeurs ont recours au patron de conception fabrique, également appelé factory pattern, qui donne son nom au modèle. Cette méthode est soit spécifiée dans une interface et implémentée par une classe enfant, soit implémentée par une classe de base et éventuellement écrasée (par des classes dérivées).

Quel est l’objectif du factory pattern ?

Le factory pattern vise à résoudre un problème fondamental lors de l’instanciation, c’est-à-dire la création d’un objet concret d’une classe, dans la programmation orientée objet : créer un objet directement au sein de la classe, qui a besoin de cet objet ou devrait l’utiliser, est possible en principe, mais très rigide. Il lie la classe à cet objet particulier et rend impossible de modifier l'instanciation indépendamment de la classe. Le factory pattern évite un tel code en définissant d'abord une opération distincte pour la création de l'objet : la fabrique. Une fois appelée, elle génère l’objet, au lieu du constructeur de classe mentionné plus haut.

1. **Singleton :**

Le design pattern Singleton permet de s'assurer qu'une classe ne puisse produire qu'une seule et unique instance.

Quelles sont les propriétés du singleton pattern ?

Une fois que l’instance d’une classe a été créée à l’aide du patron de conception Singleton, il garantit qu’elle ne reste réellement qu’avec cette seule instance. Le singleton rend cette classe globalement accessible dans le logiciel. Il existe différentes méthodes pour cela dans les langages de programmation. Pour s’assurer qu’il reste avec une seule instance unique, il faut empêcher les utilisateurs de créer de nouvelles instances. Pour ce faire, le constructeur doit déclarer le patron comme « privé ». De cette façon, seul le code du Singleton peut instancier le Singleton lui-même. Cela garantit qu’un seul et même objet peut atteindre l’utilisateur. Si cette instance existe déjà, aucune nouvelle n’est créée.

1. **Bridge :**

Le Pont est un patron de conception structurel qui permet de séparer une grosse classe ou un ensemble de classes connexes en deux hiérarchies — abstraction et implémentation — qui peuvent évoluer indépendamment l’une de l’autre.

1. **Observer :**

Le Design Pattern observer nous permettra de faire interagir le package Model avec celui de la Vue. On pourra a chaque changement d’état d’un attribut dans le modèle, notifier la vue de ce changement et ainsi la vue pourra afficher les bonnes données aux bon moments.