**Design Patterns**

* Strategy

Le strategy permet de définir une famille d'algorithmes, de placer chacun d'eux dans une classe distincte et de rendre leurs objets interchangeables.

Application :

Le strategy s’utilisera dans le menu, puisqu’il y a plusieurs des recettes pour les entrées, les plats et les desserts, le strategy permettra choisir les recettes en fonctionne des ingrédients et d’autres facteurs.

* Observer

L’observer permet de définir un mécanisme de souscription pour notifier à plusieurs objets tout événement survenant dans l'objet qu'ils observent.

Application :

L’observer s’utilisera dans la partie « vue » dans le MVC et chaque fois qu’une modification est présentée sera notifiée et une mise à jour sera effectuée dans le système.

* Singleton

Le singleton permet de vous assurer qu'une classe n'a qu'une seule instance, tout en fournissant un point d'accès global à cette instance.

Application :

Le singleton s’utilisera dans la partie « Model » dans le MVC puisqu’une seule instance est nécessaire pour les donnes à afficher.

* Builder

Le builder permet de produire différents types et représentations d'un objet en utilisant le même code de construction.

Application :

Le builder s’utilisera pour prendre les commandes, puisque les commandes sont un process de construction qui peut être représentée en différents façons, c’est-à-dire tous les commandes sont conformées pour une entrée, un plat et un dessert ; n’importe pas les variations qui peuvent exister, toujours c’est le même process de construction.

* Façade

La façade fournit une interface unifiée à un ensemble d'interfaces d'un sous-système. Façade définit une interface de niveau supérieur facilitant l'utilisation du sous-système.

Application :

La façade s’utilisera pour interagir d’une façon moins complexe avec la classe de la salle de restauration et de la cuisine et simplifier l’utilisation de leurs méthodes.