Justification des Design Pattern

**Modèle Vue Contrôleur**

Le MVC est utilisé pour scinder l’application en 3 parties distinctes, chacune ayant une tâche bien définie.

Dans le cadre de notre projet, la vue s’occupera d’afficher à l’écran la simulation, le contrôleur a pour rôle d’interpréter les actions de l’utilisateur et de les envoyer au modèle pour les interpréter. Et le model est le point central gérant toute la simulation (salle de restauration, cuisine et interaction avec la base de données).

**Singleton**

Le DP Singleton permet d’assurer qu’une classe n’est instancié qu’une seule fois et assure un accès global à cette instance.

Ce Design Pattern nous permettra de simplifier la communication avec la base de données et d’éviter le doublon de postes importants (ex : Maître d’Hôtel).

**State**

Le DP State permet de modifier le comportement d’un objet lorsque son état interne change.

Il sera notamment utilisé pour changer le comportement du Maître d’Hôtel à l’accueil de nouveaux clients en fonction de la disponibilité des tables.

**Observer**

Le DP Observer permet de définir une dépendance de 1 à n objets permettant de prévenir les n objets lors d’un changement d’état de l’objet observé.

Ici les différents changements d’état de la table (ex : « A choisit un repas », « A fini de manger », « Demande la carte des dessert ») alertera les employés pour leur permettre d’agir en fonctions des besoins.

**Iterator**

**Il fournit un moyen d’accéder aux éléments d’un objet agrégé de manière séquentielle sans exposer sa représentation sous-jacente.**

**Il permettra d’enchainer les commandes dans la cuisine.**

**Strategy**

Permet de permuter dynamiquement les algorithmes utilisés dans l’application, indépendamment du client qui les emploient.

Il permettra de partager des méthodes communes au sein de la cuisine.