

Projet Programmation Système A3

*« Master Chef Info »*

Table des matières

[Suivi de modifications 6](#_Toc531773252)

[Diagramme 7](#_Toc531773253)

[Cas d’utilisation 7](#_Toc531773254)

[Application 7](#_Toc531773255)

[Restaurant 8](#_Toc531773256)

[Cuisine 9](#_Toc531773257)

[Séquence 10](#_Toc531773258)

[Application 10](#_Toc531773259)

[Restaurant 11](#_Toc531773260)

[Cuisine 12](#_Toc531773261)

[Activité 13](#_Toc531773262)

[Restauration 13](#_Toc531773263)

[Cuisine 17](#_Toc531773264)

[Classe 21](#_Toc531773265)

[Cuisine 22](#_Toc531773266)

[Restaurant 22](#_Toc531773267)

[Composant 23](#_Toc531773268)

[MCD (Modèle Conceptuel de Données) 24](#_Toc531773269)

[Design Pattern (patron de conception) 25](#_Toc531773270)

[MVC 25](#_Toc531773271)

[Qu’est-ce que le MVC ? 25](#_Toc531773272)

[Comment l’avons-nous utilisé ? 25](#_Toc531773273)

[Observer 26](#_Toc531773274)

[Qu’est-ce que l’observer ? 26](#_Toc531773275)

[Comment nous l’utilisons et pourquoi ? 26](#_Toc531773276)

[Factory 26](#_Toc531773277)

[Qu’est-ce que Factory ? 26](#_Toc531773278)

[Comment et pourquoi l’avons-nous utilisé ? 26](#_Toc531773279)

[Decorator 26](#_Toc531773280)

[Comment et pourquoi l’avons-nous utilisé ? 26](#_Toc531773281)

[Singleton 27](#_Toc531773282)

[Qu’est-ce que Singleton ? 27](#_Toc531773283)

[Comment et pourquoi nous l’avons utilisé ? 27](#_Toc531773284)

[Iterator 27](#_Toc531773285)

[Qu’est-ce que Iterator ? 27](#_Toc531773286)

[Comment et pourquoi nous l’avons utilisé ? 27](#_Toc531773287)

# Suivi de modifications

|  |  |  |
| --- | --- | --- |
| Date de la modification | Auteur de la modification | Objet de la modification |
| 2018/12/04 | Remy Van camp | Création du document |

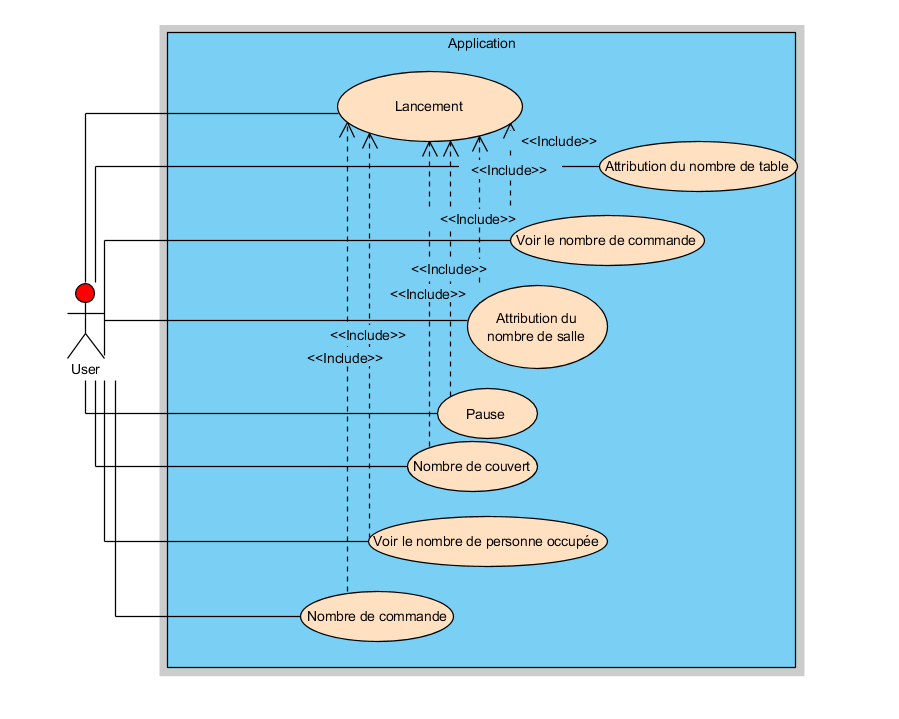
# Diagramme

## Cas d’utilisation

Les diagrammes de cas d’utilisation sont des diagrammes UML utilisés pour donner une vision globale du comportement fonctionnel d'un système logiciel. Ils sont utiles pour des présentations auprès de la direction ou des acteurs d'un projet. Dans un diagramme de cas d'utilisation, les utilisateurs sont appelés acteurs, ils interagissent avec les cas d'utilisation.

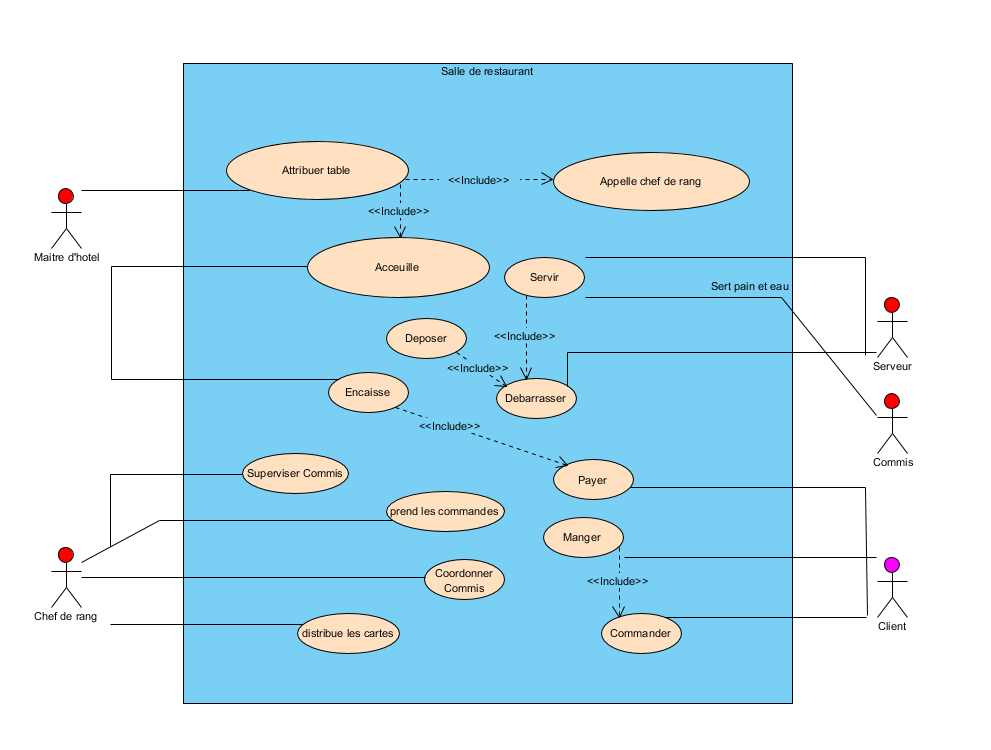
Source : *https://fr.wikipedia.org/wiki/Diagramme\_des\_cas\_d%27utilisation*

### Application



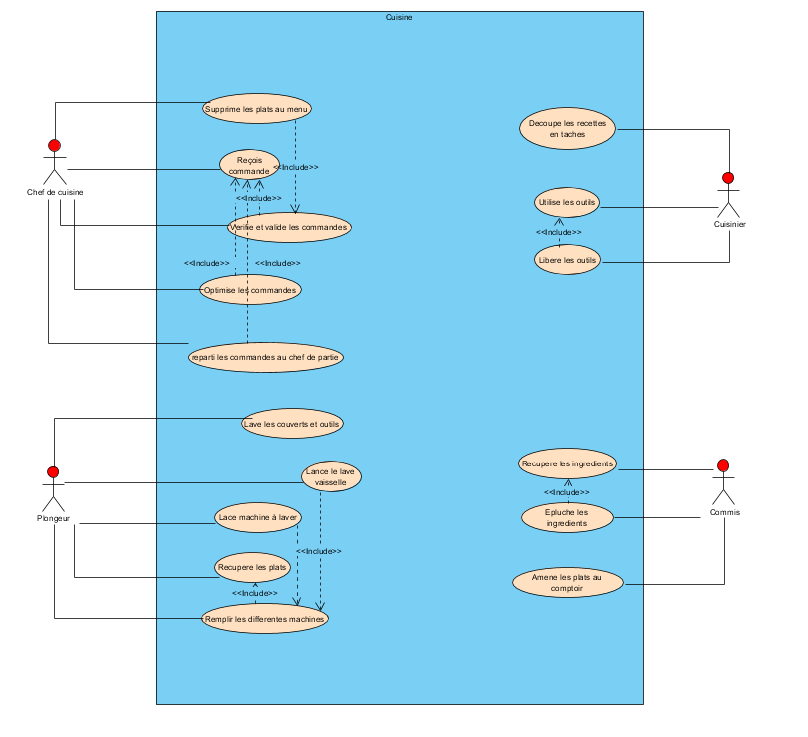
**Explication**: L’Utilisateur va pouvoir dans l’interface qui lui est mise à disposition, entrer les attributs pour émuler le restaurant. Ceci est possible que si l’application est lancée. Nous pouvons voir également que la visualisation des métriques ne sont disponibles que lorsque l’application fonctionne.

### Restaurant



**Explication** : Dans ce diagramme nous pouvons voir plusieurs points bloquants, qui requièrent un certain agencement entre les cas d’utilisation. Prenons le cas du maitre d’hôtel il ne peut attribuer une table que s’il a accueilli des clients et de ce fait il appelle le chef de rang que s’il a attribué une table. Maintenant intéressons-nous au client, il ne peut manger sans avoir commander et on peut voir qu’il y a la notion de paiement celle-ci est bloquante à la situation du maitre d’hôtel car il doit encaisser que si le client paye.

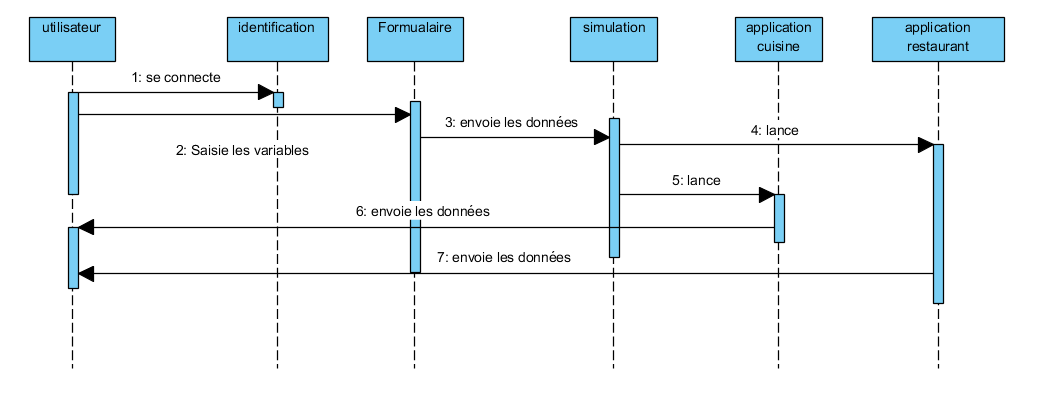
### Cuisine



**Explication** : En ce qui concerne la cuisine, nous avons 4 acteurs représentés, nous avons le chef de cuisine, le plongeur, le cuisinier et le commis. Tous ces acteurs sont associés à des cas d’utilisation, ceux-ci n’ont parfois pas de cas de blocage, en conséquent les acteurs peuvent réaliser leurs actions sans que cela ne bloque les actions des autres. Cependant il y a la présence de cas bloquant comme par exemple le chef de cuisine dans notre cas, va pouvoir supprimer un plat du menu seulement s’il refuse la commande par manque de stock par exemple. Pour le plongeur il y a différentes parties bloquantes notamment en ce qui concerne les lancements du lave-vaisselle et de la machine à laver car le remplissage de celles-ci est obligatoire. Les commis, pour éplucher les ingrédients, doivent récupérer et donc chercher ces ingrédients.

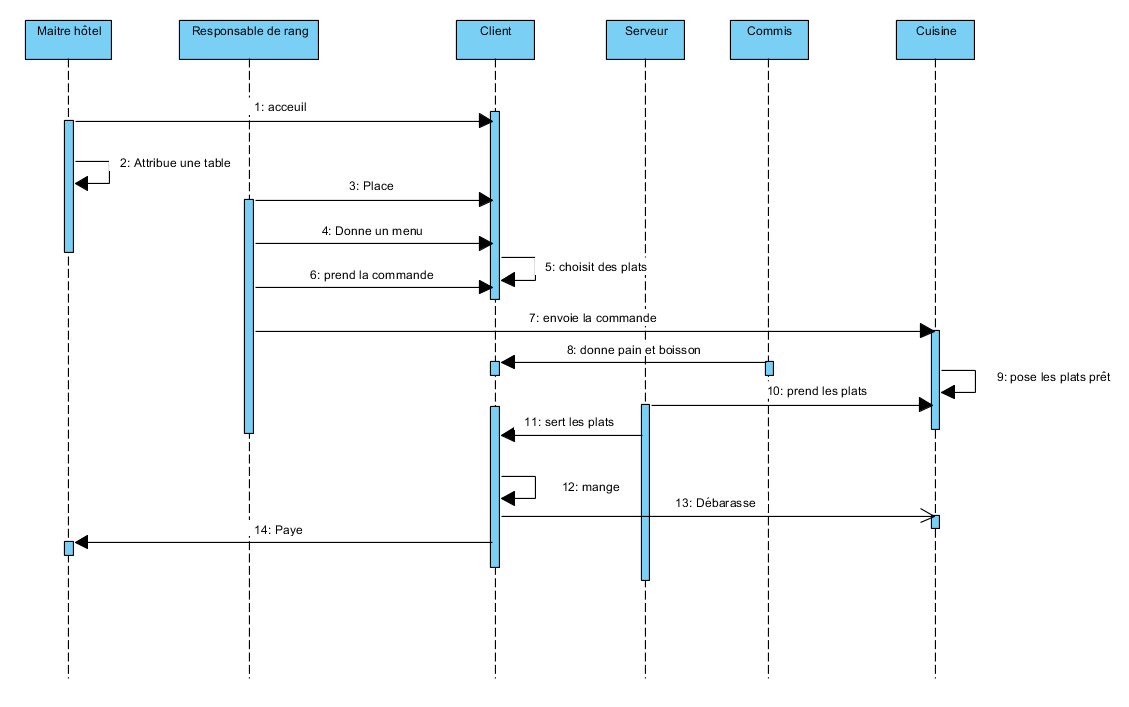
## Séquence

### Application



Lorsque l’utilisateur lance l’application dans un premier temps il faut qu’il s’authentifie ensuite un formulaire va s’afficher et l’utilisateur doit saisir les valeurs nécessaires à la simulation, la simulation se lance ce qui va mettre en route les applications cuisine et restaurant. Une fois la simulation finie les deux applications renvoie les données à l’utilisateur dans l’application principale.

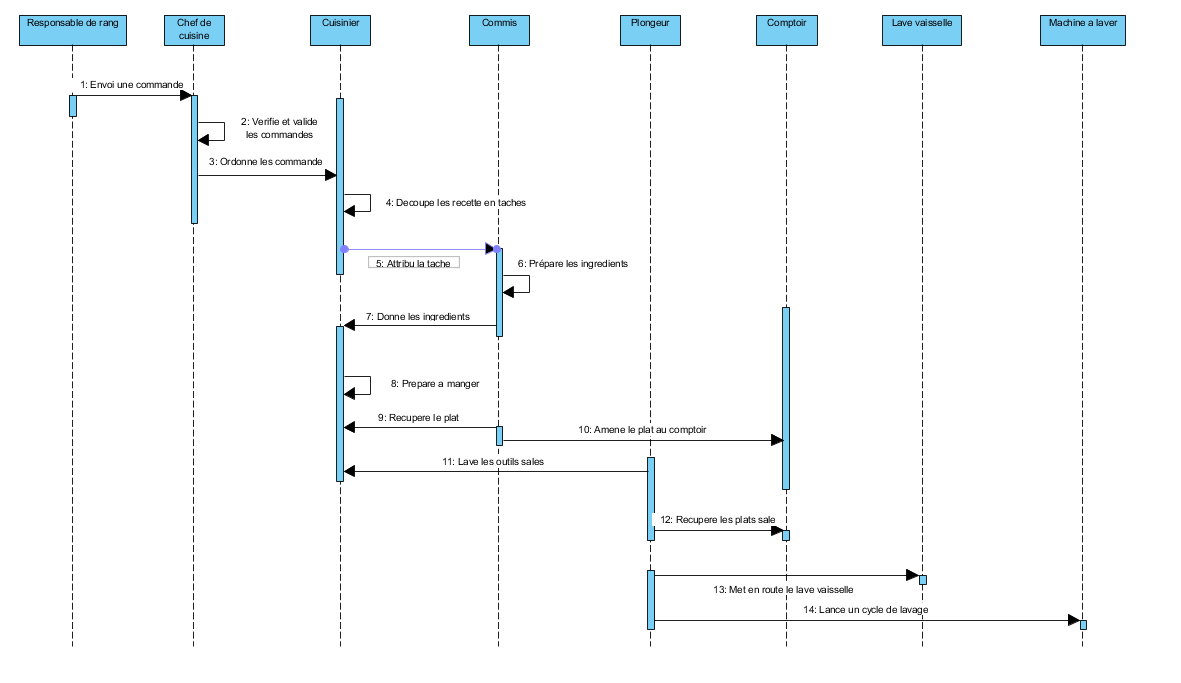
### Restaurant



Lorsqu’un client arrive, le maitre d’hôtel l’accueil puis leur choisit une table. Ensuite le maitre de rang accompagne le client jusqu’à sa table, leur distribue des menus. Une fois que les clients ont choisi leur plat le serveur vient prendre leur commande puis la transmet à la cuisine. Le commis leur apporte du pain et leur boisson.

Quand le plat est prêt le serveur va le chercher et le sert au client. Une fois que le client a fini de manger le serveur débarrasse la table et le client va payer au maitre d’hôtel.

### Cuisine



Lorsqu’un chef de cuisine reçoit une commande d’un responsable de rang. Il va tout d’abord vérifier et valider cette commande suite à cela il va dispatcher les commandes entre les différents cuisiniers.

Le cuisinier va alors découper les recettes en tâches puis attribuer des tâches aux commis qui dépendent de lui. Le commis va alors se charger de préparer les ingrédients puis les donner à son cuisinier. Suite a cela le cuisinier prépare le plat. Dès que le plat est terminé le commis va se charger d’amener les plats au comptoir. A ce moment-ci le plongeur va s’occuper de laver les outils sale utiliser par le cuisinier et le commis. Puis va s’occuper récupérer les plats sur le comptoir puis de les nettoyer.

Le plongeur aura comme autre rôle de lancer indépendamment de ses autres actions le lave-vaisselle ainsi que la machine à laver

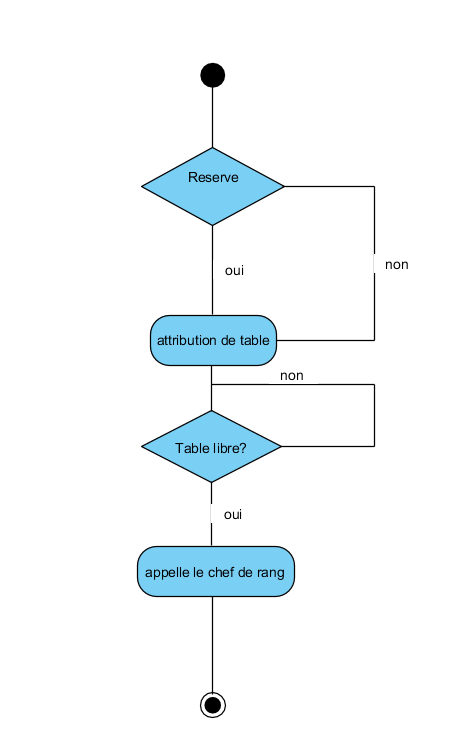
## Activité

Le diagramme d'activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements parallélisables.

« <https://fr.wikipedia.org/wiki/Diagramme_d%27activité>  »

### Restauration

#### Maitre d’hôtel



Le maitre d’hôtel accueil les clients, vérifie s’ils ont une réservation ou non. Décide de la table ou ils se mettront. Puis appelle le chef de rang pour qu’il puisse les accompagner.

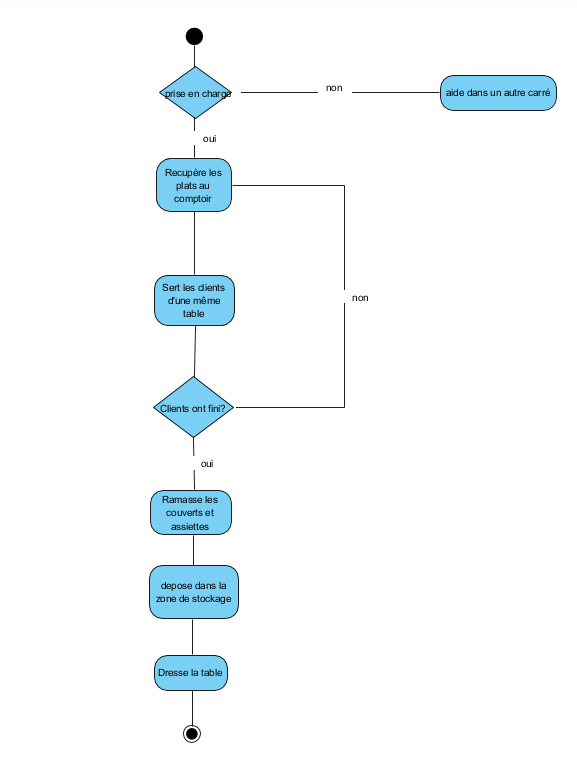
#### Chef de rang

#### 

Le chef de rang quand il n’est pas occupé peut aller aider ses collègues en salle.

En temps normal il accompagne les clients de l’entrée à leur table. Puis il donne les cartes au client et prend leur commande.

#### Serveur

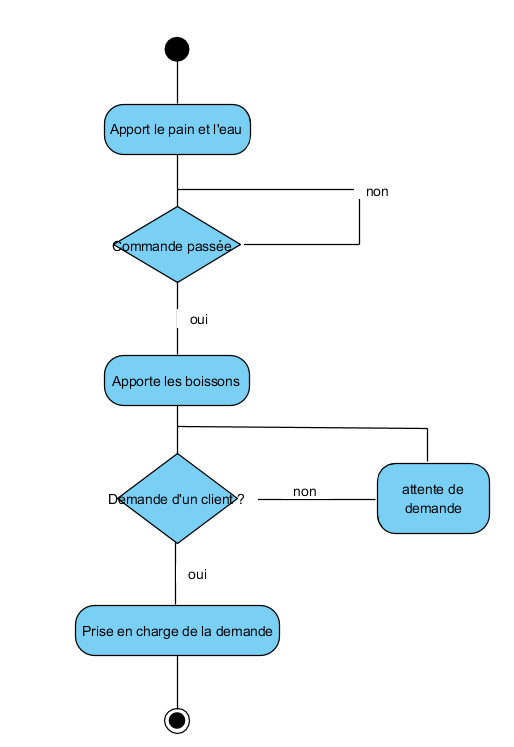


Si le serveur n’est pas occupé il peut aider dans un autre carré.

Il récupère les plats au comptoir puis les sert. Il réitère cette opération jusqu’à ce que des clients aient finis. Quand ils ont fini il ramasse les assiettes et les couverts, il les dépose dans la zone de stockage. Puis va redresser la table.

#### 

#### Commis

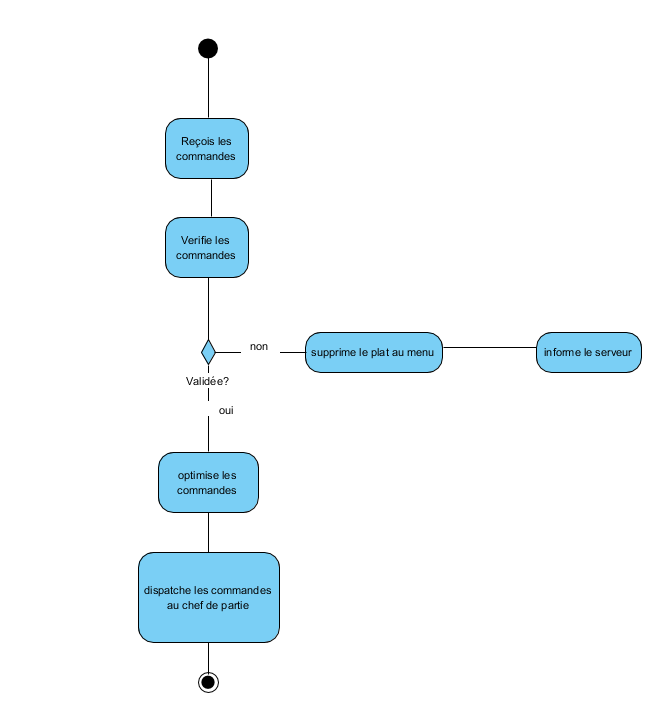


Le commis apporte le pain et l’eau. Une fois qu’une commande est passée par un client, il ramène les boissons.

Il se tient à la disposition des clients s’ils ont une demande.

### Cuisine

#### Chef de cuisine

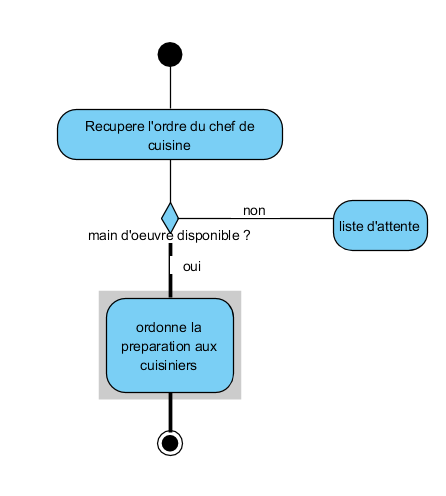


Le chef de cuisine va recevoir les commandes puis les vérifier.

Si jamais ces commandes sont validées et qu’elles sont réalisables on dispatche de façon optimal les commandes entres les différents chefs de partie.

Si jamais elles ne sont pas validées, on supprime le plat du menu puis ont en informe le serveur

#### Chef de partie

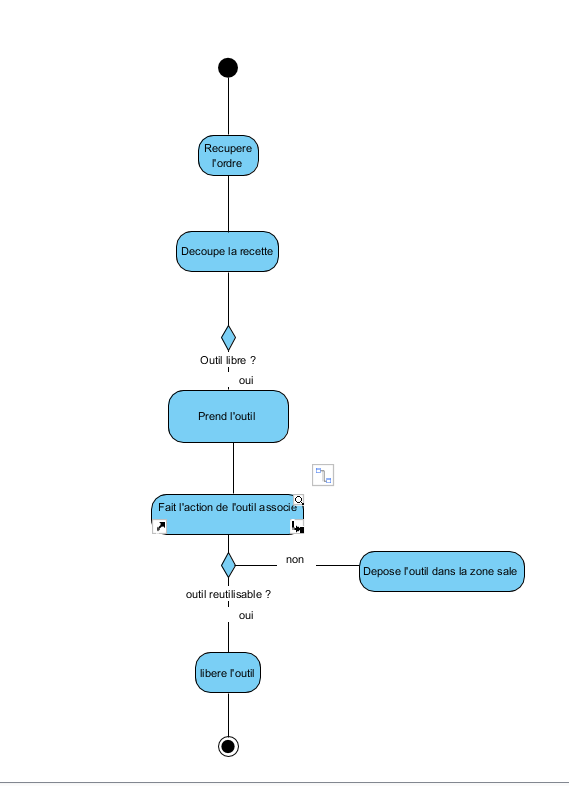


#### Le chef de partie a pour rôle de récupérer l’ordre donné précédemment par le chef de cuisine.

Si la main d’œuvre nécessaire est disponible on va alors ordonner la préparation du plat aux cuisiniers.

Si jamais la main d’œuvre n’est pas disponible on laisse en file d’attente les ordres du chef

#### Cuisinier



Le cuisinier a pour rôle de récupérer les ordres du chef de partie.

Il va ensuite découper la recette en différentes tâches.

Si jamais ces outils sont disponibles il va alors réaliser l’action associé à cet outil.

À la suite de cet action deux cas sont possibles les objets sont réutilisables ou non.

Si jamais ces outils ne sont pas réutilisables on les dépose dans une zone dites « sale »

#### 

#### Commis cuisine

#### 

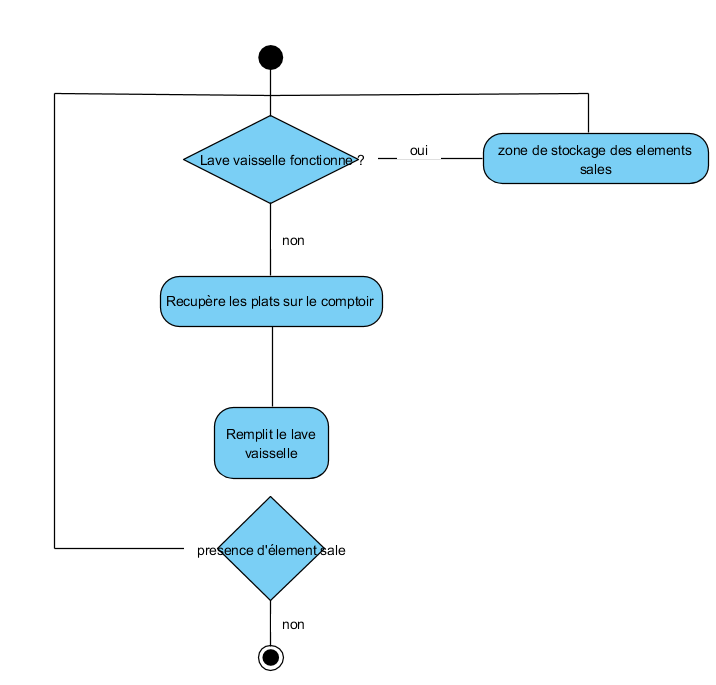
#### Le commis de cuisine va récupérer un ordre du cuisinier

Si l’outil dont il a besoin est libre il va donc commencer la tache dont il est chargé.

#### Si son outil n’est pas réutilisable il va le déposer dans la zone du « sale ».

Si cet outil est assez propre pour être utiliser de nouveau il va donc le libérer suite a cela le commis de cuisine se charge d’amener les plats sur le comptoir.

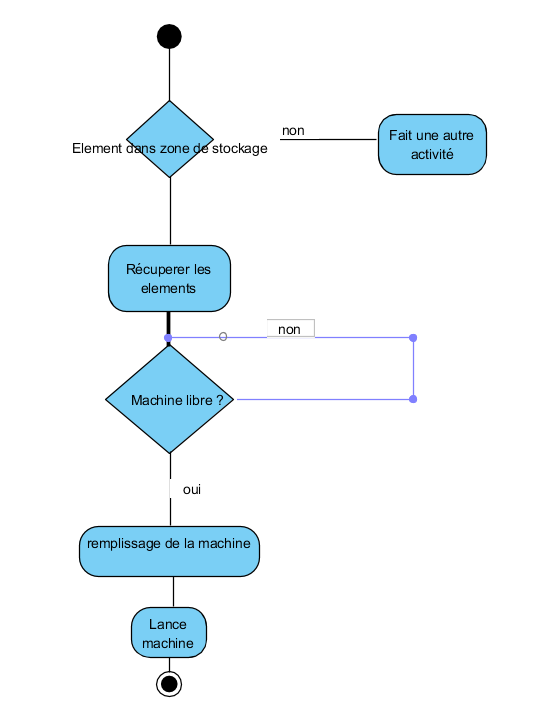
#### Plongeur



Le plongeur va tout d’abord devoir s’assurer de l’état du lave-vaisselle

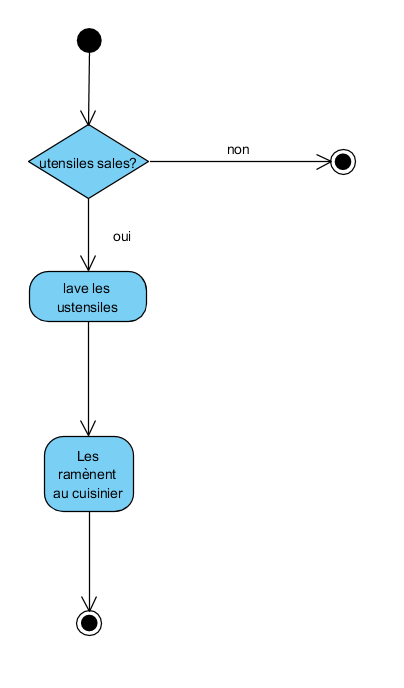
Si le lave-vaisselle est en marche il va amener les éléments sales à la zone de stockage.

Si celui-ci est fonctionnel il va récupérer les plats sur le comptoir et remplir le lave-vaisselle avec eux.



Le plongeur va vérifier la présence d’élément dans la zone de stockage si jamais des éléments sont présent il va les récupérer.

Il vérifie ensuite la disponibilité de la machine si celle-ci est disponible il va devoir remplir la machine puis la lancer



Le plongeur vérifie si les ustensiles sont sales, si oui ils les lavent puis les ramènent.

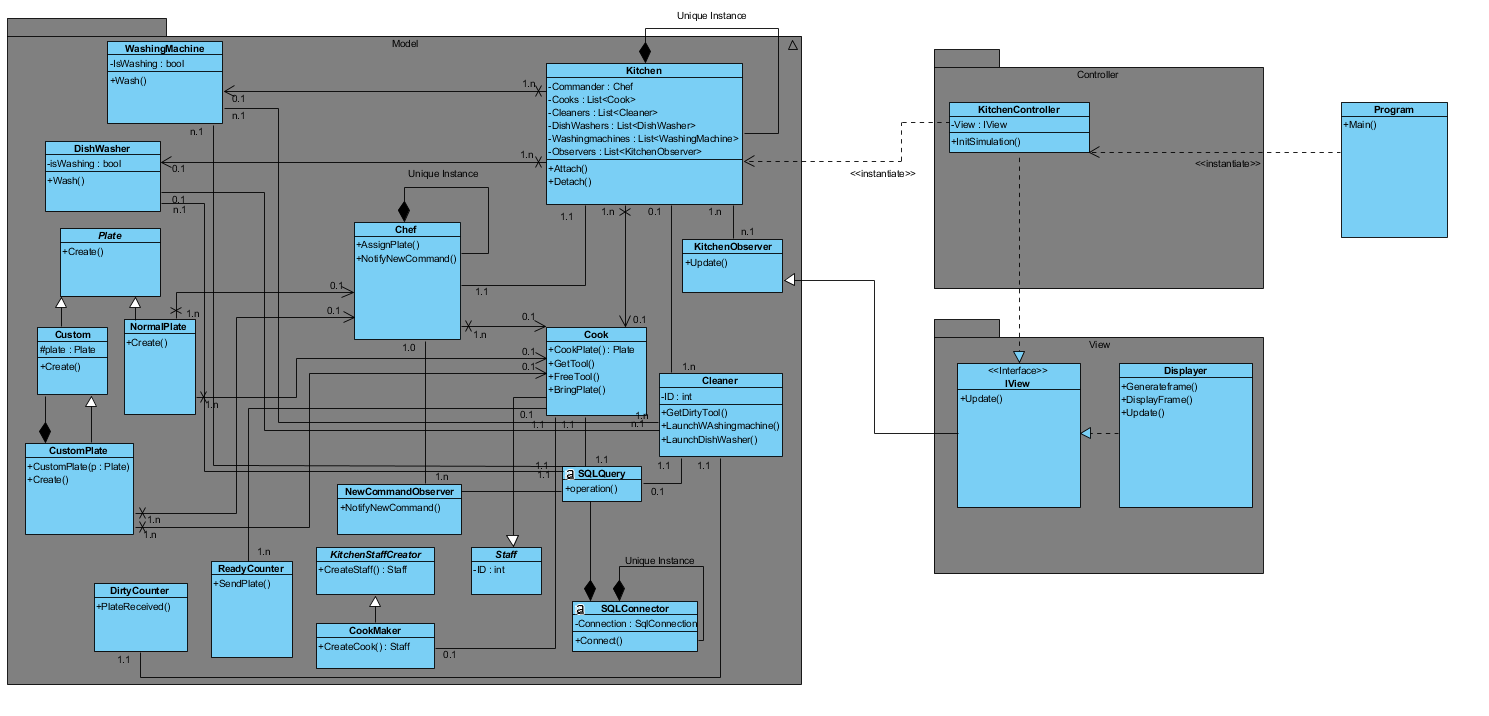
## Classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.

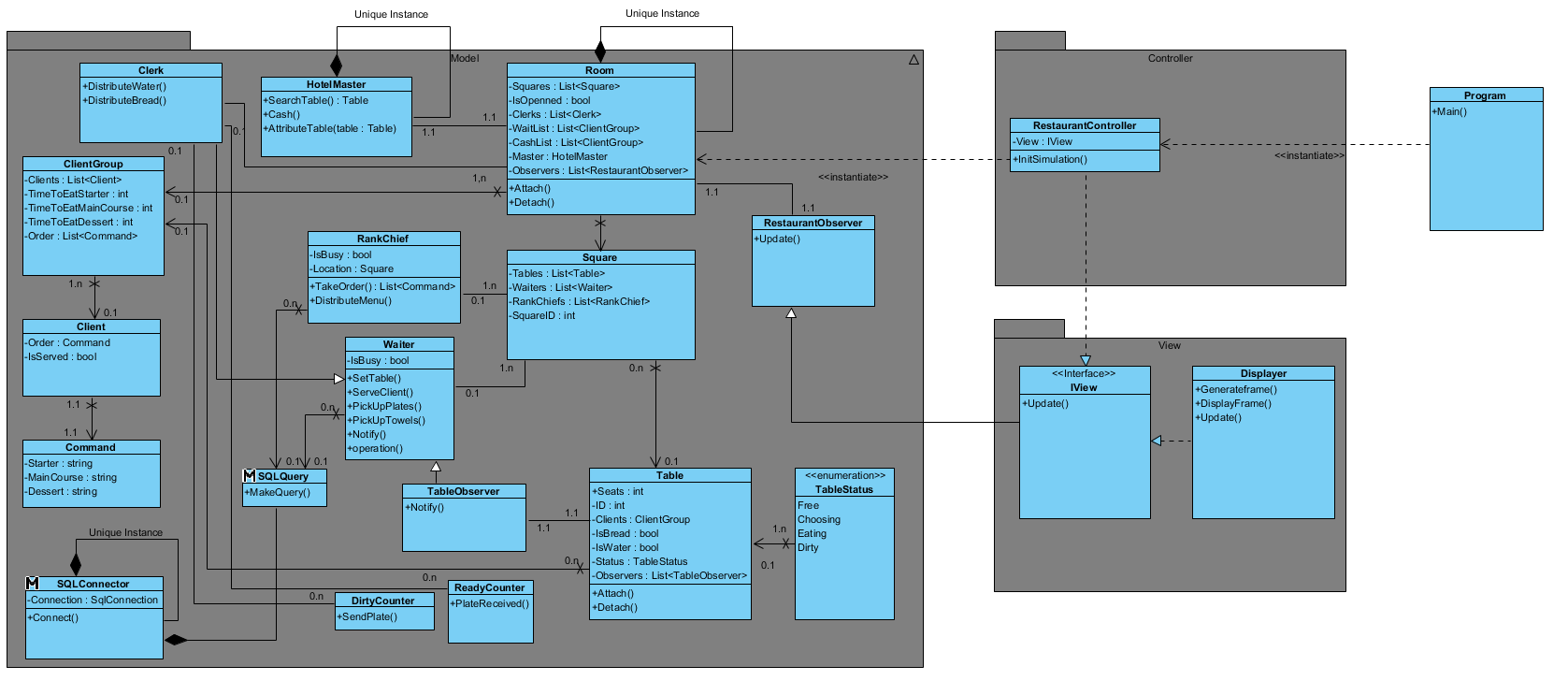
Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

Selon *https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes*

### Cuisine



### Restaurant

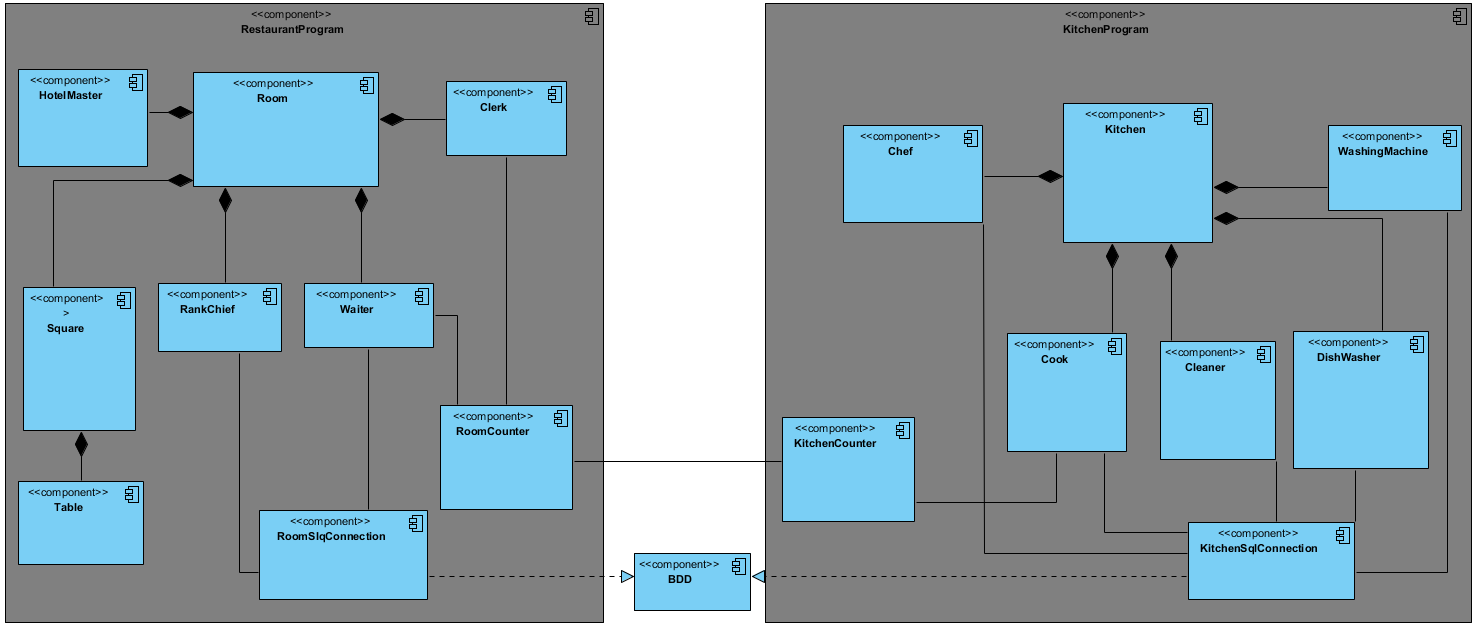


## Composant

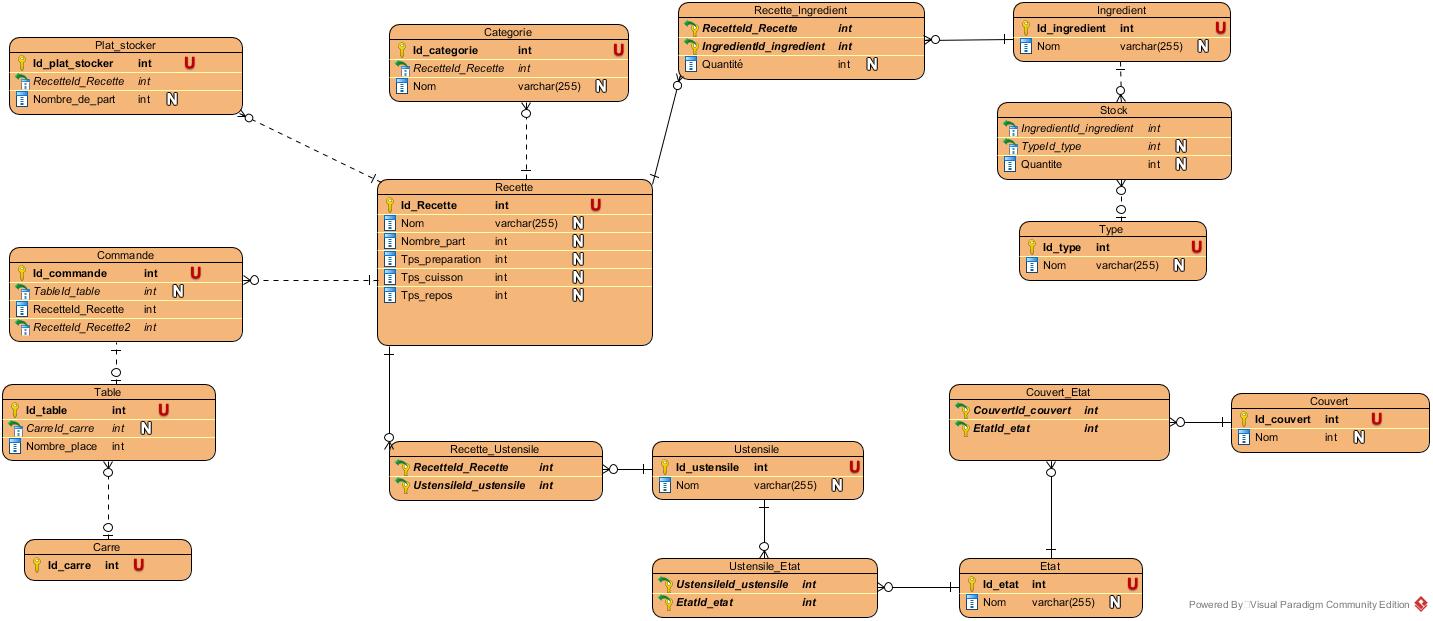
Le diagramme de composant va décrire, dans l’environnement de réalisation décrire les composants et leurs dépendances, il est composé de :

* Descriptions des implémentations du système qui sont les composants
* Groupes d’implémentations qui sont les modules
* Relations entre les diverses implémentations qui sont les dépendances

Le but est de décrire les choix d’implémentation et les dépendances de compilation et d’implémentation entre les composants du système.



## MCD (Modèle Conceptuel de Données)



**Explication** : La table centrale de notre MCD est celle des recettes. Celle-ci dispose de colonnes qui décrivent les différents temps utiles à la recette et le nombre de part. Pour définir si une recette est une entrée, un plat ou un dessert, nous avons créé une table Catégorie, une clef étrangère des catégories se trouve dans la table Recette.

Les recettes nécessitent des ustensiles. Ceux si sont définis dans la table ustensile, pour savoir s’ils sont prêts à l’emploie il a fallu leur attribuer un état. En effet il y a une table état qui décrit les états possibles des ustensiles ainsi que des couverts. Il y a une table intermédiaire entre celle des ustensiles et d’Etat pour décrire l’Etat de chacun d’eux et de la même manière pour les couverts.

Pour les parts qui doivent être stockées nous avons créé un table Stocker, celle si comprend l’identifiant des recettes ainsi que la quantité de part disponible.

Pour la gestion des aliments nous avons créé une table Ingrédient celle-ci liste les ingrédients utilisés par le restaurant. Pour les stocker nous avons créé un table Stock, et une table type. La table type définit les trois types de stockage où peuvent être les aliments. Donc dans la table Stock nous avons l’identifiant d’un aliment, avec l’identifiant du type de stockage ainsi que sa quantité. De plus pour connaitre les ingrédients utilisés par un recette il y a une table intermédiaire entre celle des ingrédients et celle des recettes. Celle si contient l’identifiant des recettes, celui des aliments ainsi que la quantité nécessaire à la recette.

Les commandes sont reliées aux tables qui sont elles-mêmes reliées aux carrés. Une commande comprend l’identifiant de la table et ceux des recettes commandées.

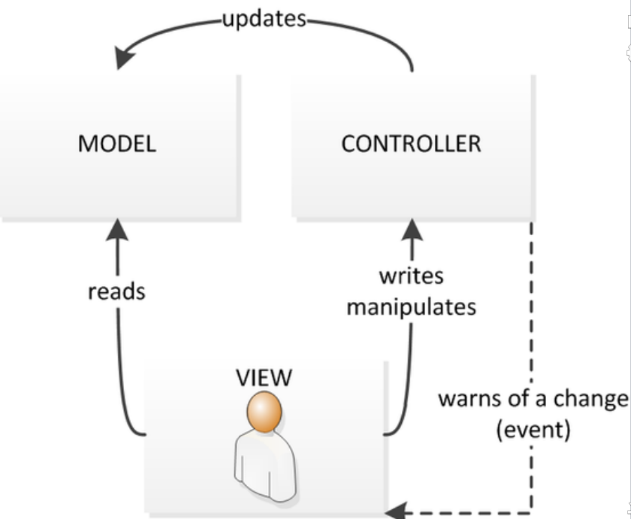
# Design Pattern (patron de conception)

## MVC

### Qu’est-ce que le MVC ?

Le MVC est l’acronyme de Model-View-Controller, c’est un patron de conception qui est destiné aux interfaces graphiques. Il est composé de trois types de modules ayant trois responsabilités différentes :

* Un modèle (Model) contient les données à afficher.
* Une vue (View) contient la présentation de l'interface graphique.
* Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur.



### Comment l’avons-nous utilisé ?

Le Design pattern MVC est utilisé dans le cadre structurel de l’application, il permet de séparer les différents composants pour l’évolutivité du programme.

## Observer

### Qu’est-ce que l’observer ?

L’observer est un patron de concepteur ayant pour but d’envoyer un signal à des modules jouant le rôle d’observateurs. En cas de notification, les observateurs effectuent alors l'action adéquate en fonction des informations qui parviennent depuis les modules qu'ils observent (les observables).

### Comment nous l’utilisons et pourquoi ?

L’observer est utilisé dans la surveillance et la supervision des éléments dans le programme, par exemple la gestion de l’arrivée d’un nouveau client est gérée par un observer.

## Factory

### Qu’est-ce que Factory ?

La fabrique est un patron de conception. Elle permet d'instancier des objets dont le type est dérivé d'un type abstrait. La classe exacte de l'objet n'est donc pas connue par l'appelant.

Plusieurs fabriques peuvent être regroupées en une fabrique abstraite permettant d'instancier des objets dérivant de plusieurs types abstraits différents.

Les fabriques étant en général uniques dans un programme, on utilise souvent le patron de conception singleton pour les implémenter.

Selon : <https://fr.wikipedia.org/wiki/Fabrique_(patron_de_conception)>

### Comment et pourquoi l’avons-nous utilisé ?

La Factory est utilisée pour la création des différents parti multiple existant dans la cuisine.

## Decorator

Le design Decorator a pour but d’attacher dynamiquement des responsabilités supplémentaires à un objet. Il fournit une alternative souple à l’héritage, pour étendre des fonctionnalités.

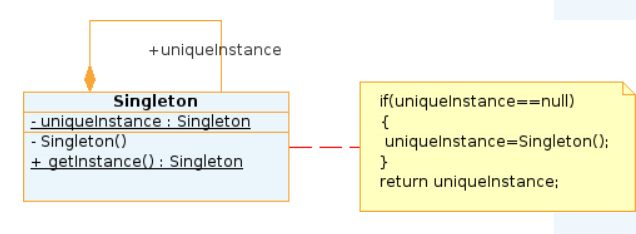
### Comment et pourquoi l’avons-nous utilisé ?

Le Decorator est utilisé dans le cadre de la création des plats, une spécialisation d’un plat se fait via le Decorator, par exemple une crêpe a la chantilly est une responsabilité supplémentaire de la crêpe normale.

## Singleton

### Qu’est-ce que Singleton ?

Le design pattern Singleton garantit qu’une classe n’a qu’une seule instance et fournit un point d’accès global à cette instance.



### Comment et pourquoi nous l’avons utilisé ?

Le Singleton est utilisé dans les cas où il n’y a qu’une seule instance de l’objet qui doit exister au moment de l’exécution, comme par exemple la salle du restaurant.

## Iterator

### Qu’est-ce que Iterator ?

Le design pattern Iterator est le plus commun des modèles de comportement. L'idée étant de limiter la vision d'une collection par un utilisateur. Typiquement une collection contient un ensemble d'objets stocké par différentes méthodes (un tableau, un vecteur...), l'exploitant qui accède au contenu de la collection ne souhaite pas être concerné par cette manière de gérer les objets. La collection offre donc un point d'accès unique sous la forme d'une interface Iterator.

### Comment et pourquoi nous l’avons utilisé ?

L’Iterator est utilisé par le biais des objets List<>, celle-ci utilisent l’iterator pour la gestion des elements.