

Projet DÉveloppement systÈme .NET

Rapport d’Architecture



05 décembre 2018

vINCENT vallet & Matthis Moyon & Flavien Spataro & Camille AUBRET & Loïck LEGAY

EXIA CESI



Table des matières

[Contexte : 2](#_Toc531793194)

[Présentation de l’équipe : 2](#_Toc531793195)

[Organisation de l’équipe : 3](#_Toc531793196)

[Cahier des Charges : 3](#_Toc531793197)

[Architecture de notre application 3](#_Toc531793198)

[Diagramme de cas d’utilisation 4](#_Toc531793199)

[Diagramme d’activité 5](#_Toc531793200)

[Diagramme de classe 9](#_Toc531793201)

[Diagramme de composant 11](#_Toc531793202)

[Diagramme de séquence 12](#_Toc531793203)

[Les différents Design Pattern 14](#_Toc531793204)

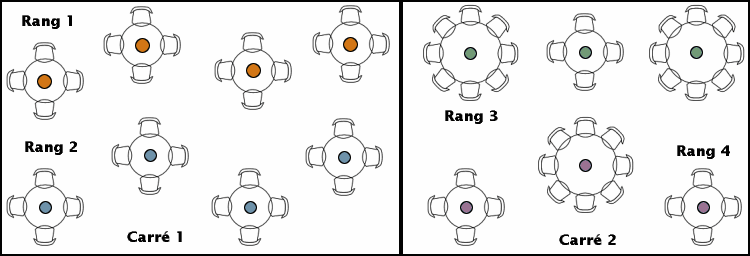
[MCD 16](#_Toc531793205)

[Script SQL 16](#_Toc531793206)

## Contexte :

Une grande chaîne internationale de restaurants souhaite s’équiper d’une nouvelle application informatique pour améliorer l’accueil du public, le remplissage des salles, la gestion des réservations et l’organisation du travail en cuisine. Depuis quelque temps, dans un restaurant, les incidents regrettables se multiplient : file d’attente trop importante, désorganisation du service entre la salle et la cuisine, manque de matériel pour servir ou cuisiner entrainant de très longues attentes en salle pour les clients… En résumé, les clients partaient très insatisfaits. Aujourd’hui on sait qu’un client insatisfait prend une des applications pour noter le restaurant et saisit un avis négatif, ce qui a pour conséquence de baisser la réputation de la maison.

Le directeur du restaurant a décidé d’investir dans le développement d’une application de gestion et supervision du fonctionnement de son restaurant (salle de restauration et cuisine). Il fait appel à vous, étudiants ingénieurs CESI en A3, pour l’aider sur ce projet. Vous devez analyser les résultats de la simulation et faire des propositions d’amélioration. Vos propositions seront dans l’obligation de reposer sur des éléments mesurables pour mettre en évidence, sans ambiguïté, les gains que vos propositions suggèrent et donc doivent être exprimées en unité de mesure (temps, pourcentage, argent, etc…).



## Présentation de l’équipe :

Vincent VALLET

Chef de projet

Flavien SPATARO

Développeur C#

Loïck LEGAY

Développeur C#

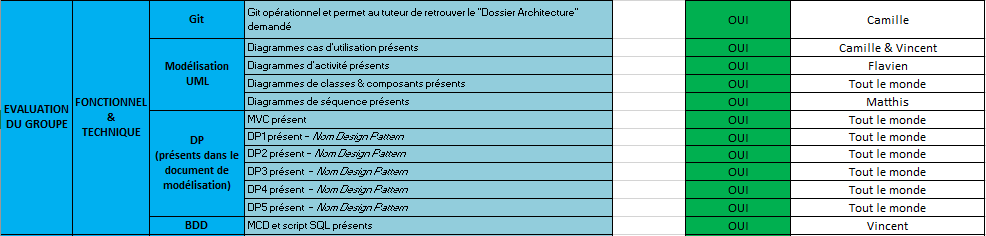
Camille AUBRET

Développeuse C#

Matthis MOYON

Développeur C#

## Organisation de l’équipe :



## Cahier des Charges :

Pour ce projet nous devons mettre en place une application en C# de simulation d’une journée de restauration avec différentes contraintes comme :

* La présence de différents diagrammes : Diagrammes de cas d’utilisation, Diagrammes d’activité de chaque poste de travail, Diagrammes de composants et classes de l’application (ou applications), Diagramme(s) de séquences.
* La présence de différents Design Pattern au sein de notre programme.
* La présence d’une base de données, donc une modélisation MVC ainsi que du script.
* Une simulation graphique du fonctionnement du restaurant en temps réel.
* L’application doit permettre de visualiser l’état de chaque personne et de chaque objet modélisé ainsi que les situations limites, et mettre des alertes sur le manque de telle ou telle ressources.
* Il doit y avoir un mode « **PAUSE** » de notre application pour pouvoir stopper tous les processus et analyser la situation en cours.
* Les temps de chaque tâche, les quantités d’objets ou des postes, le nombre de clients par type, le temps en mode accéléré, … tout doit être paramétrable dans l’application.

## Architecture de notre application

Nous allons vous présenter notre architecture pour la mise en place de notre simulateur de Restaurant. Nous devions mettre en place différents diagrammes comme :

### Diagramme de cas d’utilisation

Les diagrammes de cas d’utilisations 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, mais pour le développement, les cas d'utilisation sont plus appropriés. Dans notre cas nous avons représenté la vision globale du comportement après l’adaptation de notre outil de gestion et de supervision :

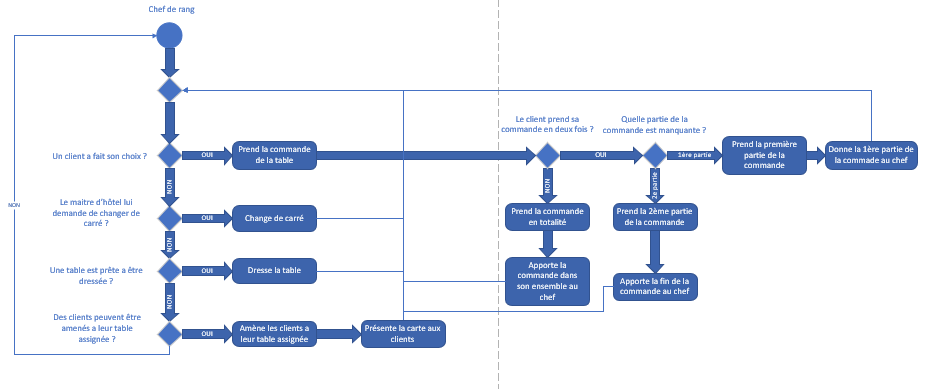


### Diagramme d’activité

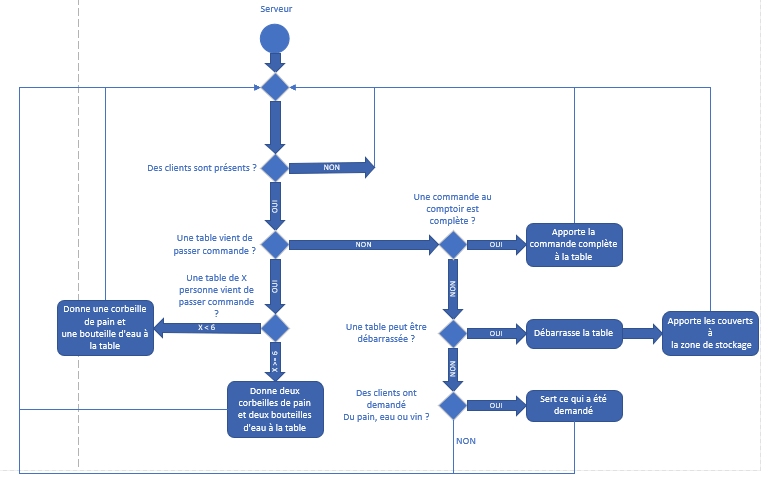
Le diagramme d’activité est un diagramme comportemental d'UML, permettant de représenter le déclenchement d'événements en fonction des états du système et de modéliser des comportements et pour décrire un flux de travail (workflow). Nous avons décidé de modéliser le comportement de chaque acteur c’est-à-dire les différents employés et les clients.

#### Maître d’hôtel

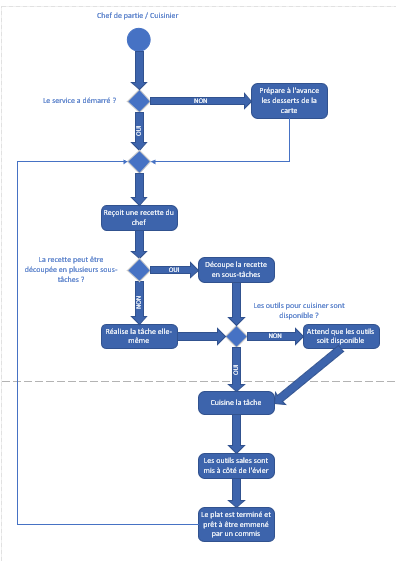
#### Chef de rang

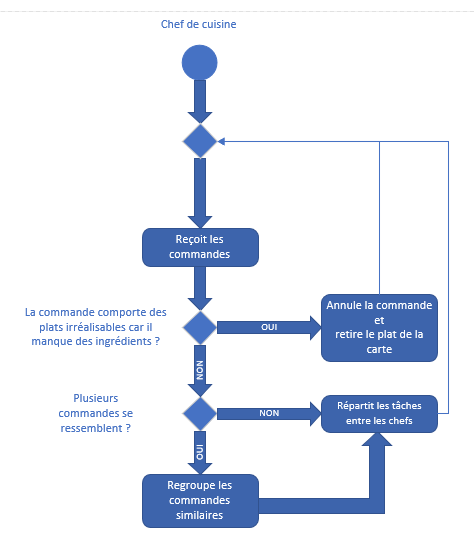


#### Serveur

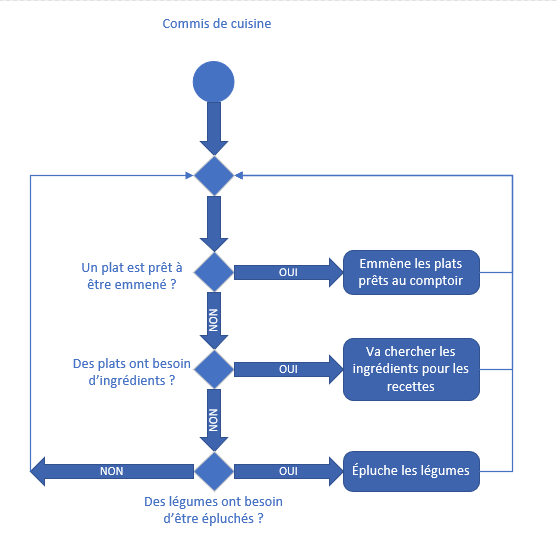


#### Chef de cuisine Chef de partie / Cuisinier

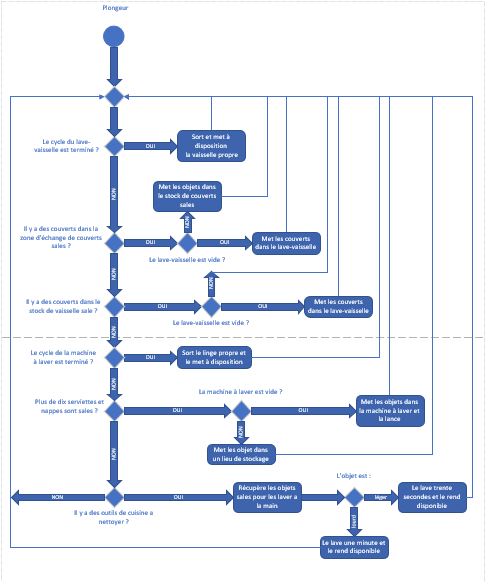




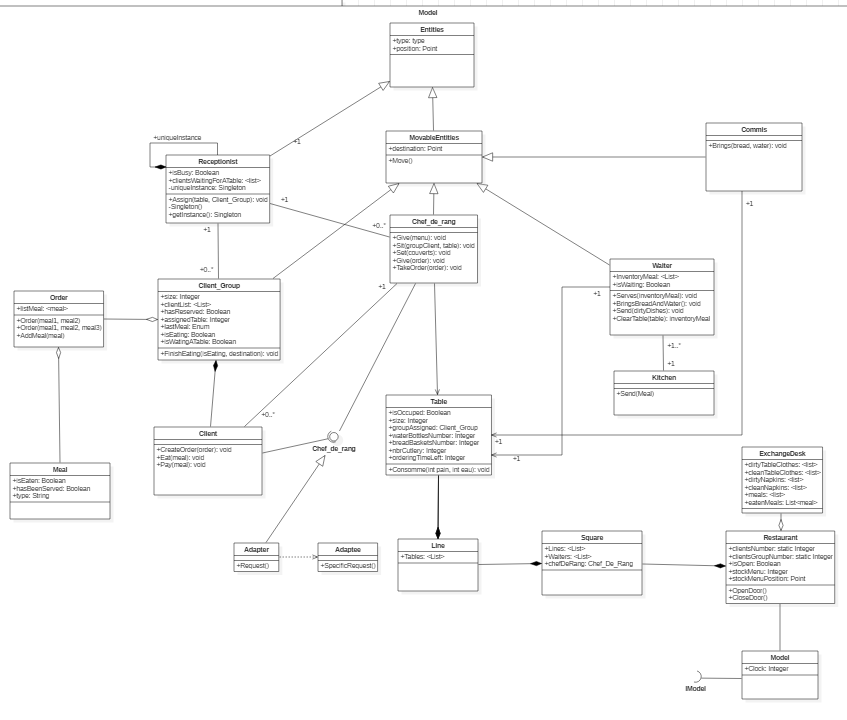
#### Commis de cuisine

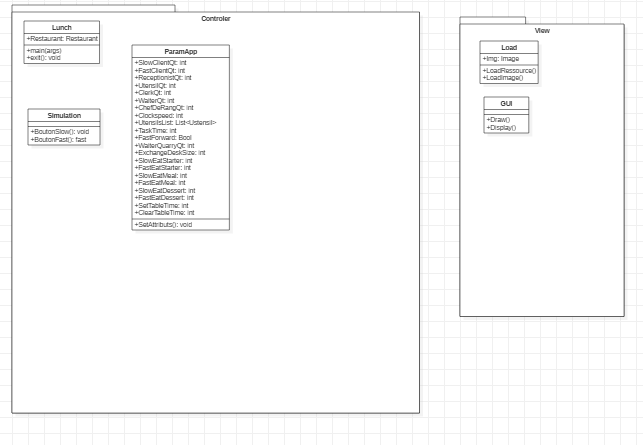


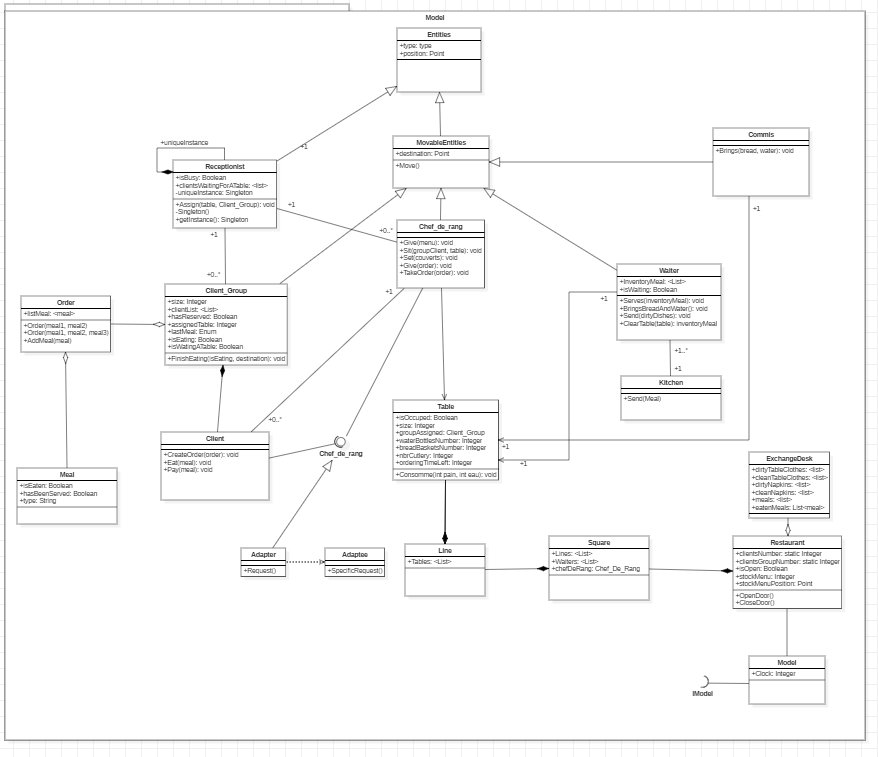
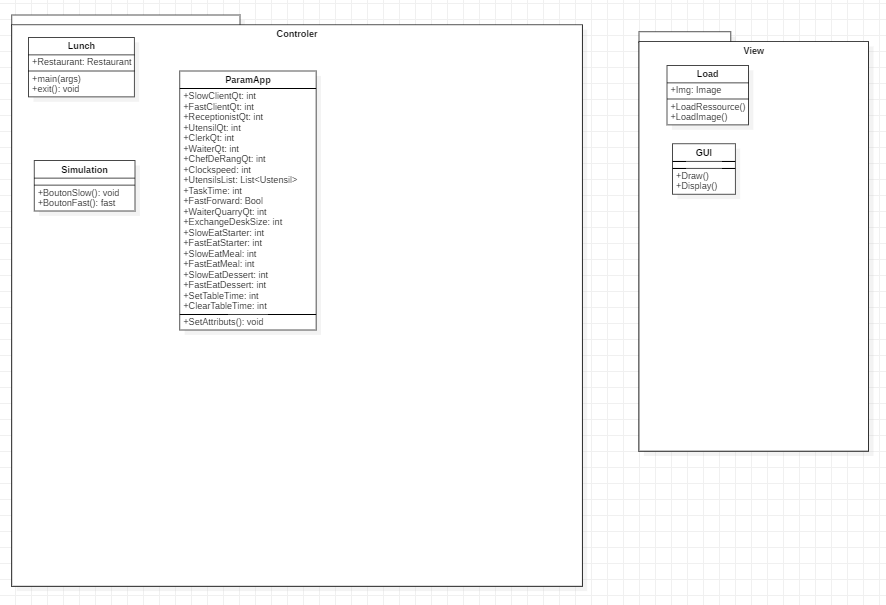
#### Plongeur



### Diagramme de classe

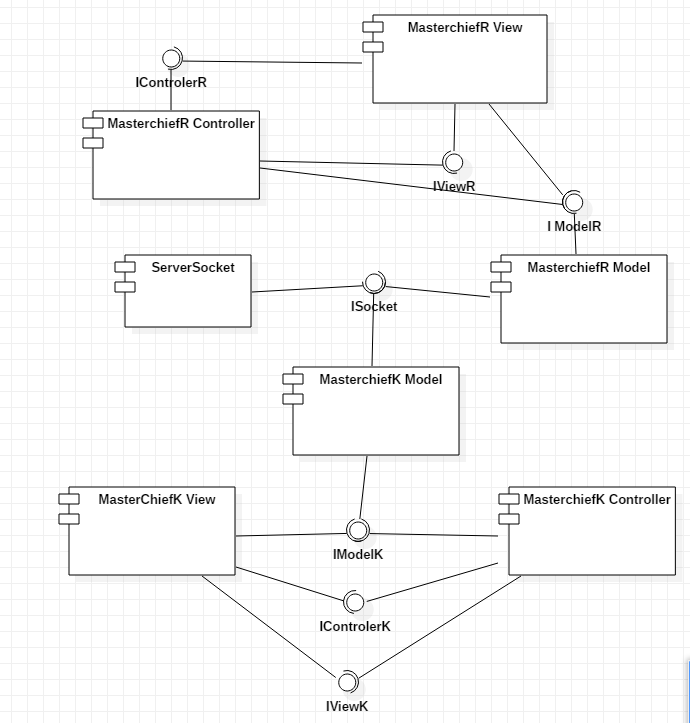
Le diagramme de classe représente les classes intervenant dans le système. Le diagramme de classe est une représentation statique des éléments qui composent un système et de leurs relations. Chaque application qui va mettre en œuvre le système sera une instance des différentes classes qui le compose. Nous avons décidé de mettre en place deux applications pour connecter les deux salles. Voici les deux diagrammes des deux MVC de nos deux applications





### Diagramme de composant

Le diagramme de composant décrit l'organisation du système du point de vue des éléments logiciels comme les modules (paquetages, fichiers sources, bibliothèques, exécutables), des données (fichiers, bases de données) ou encore d'éléments de configuration (paramètres, scripts, fichiers de commandes). Ce diagramme permet de mettre en évidence les dépendances entre les composants (*qui utilise quoi*).

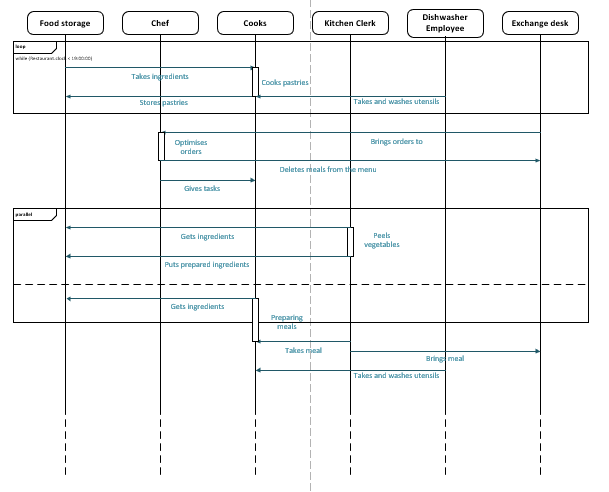


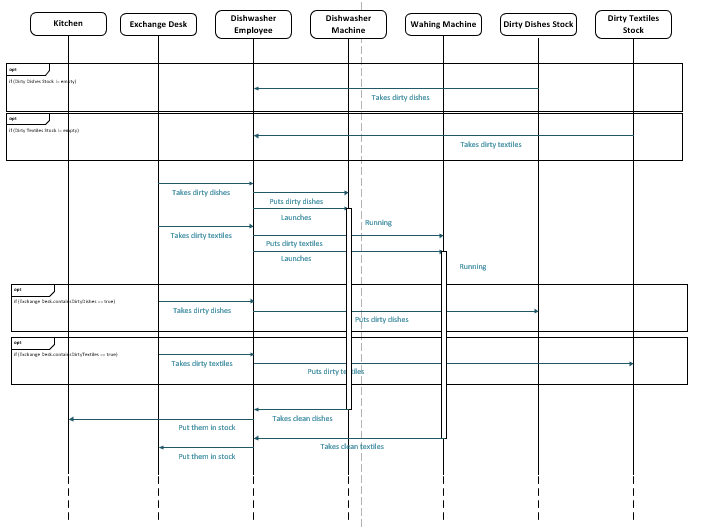
### Diagramme de séquence

Le diagramme de séquence est la représentation graphique des interactions entre les acteurs et le système selon un ordre chronologique dans la formulation UML. Ils vont représenter 2 cas dans notre projet le cas :

#### Salle de Restauration

#### Cuisine



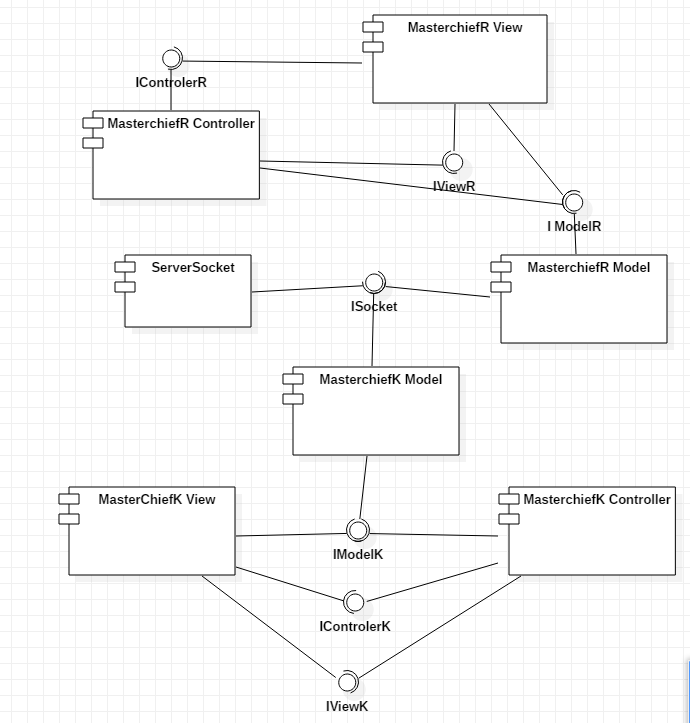


### Les différents Design Pattern

#### MVC

Le Design Pattern MVC est destiné aux interfaces graphiques. Le motif est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

* Un modèle (Model) contient les données à afficher, donc tous les éléments de notre restaurant (personnel, matériel, ustensile…).
* Une vue (View) contient la présentation de l'interface graphique, donc la représentation den la salle de restaurant et de la cuisine.
* Un contrôleur (Controller) contient la logique concernant les actions effectuées par l'utilisateur, donc uniquement le bouton pause et accéléré puisqu’il s’agit d’une simulation.

Dans notre simulateur nous avons modéliser 2 MVC, un pour la cuisine et un pour la salle de restauration pour pallier les problèmes de communication des sockets pour l’échange entre la cuisine et la salle.

1er MVC pour la salle de restauration

2ème MVC pour la cuisine

#### Singleton

Le Design Pattern Singleton a pour l'objectif de restreindre l'instanciation d'une classe à un seul objet (ou bien à quelques objets seulement). Il est utilisé lorsqu'on a besoin exactement d'un objet pour coordonner des opérations dans un système. Le modèle est parfois utilisé pour son efficacité, lorsque le système est plus rapide ou occupe moins de mémoire avec peu d'objets qu'avec beaucoup d'objets similaires. Nous l’utilisons pour avoir uniquement un employé (maître d’hôtel, cuisinier, chef de cuisine, serveur…)

#### Observer

Le Design Pattern Observer définit une relation entre objets de type un-à-plusieurs, de façon que, si un objet change d’état, tous ceux qui en dépendent en soient informés et mis à jour automatiquement. Nous l’utilisons pour

#### Strategy

Le Design Pattern Strategy est un DP de type comportemental grâce auquel des algorithmes peuvent être sélectionnés à la volée au cours du temps d'exécution selon certaines conditions. Il est utile pour des situations où il est nécessaire de permuter dynamiquement les algorithmes utilisés dans une application. Le patron stratégie est prévu pour fournir le moyen de définir une famille d'algorithmes, encapsuler chacun d'eux en tant qu'objet, et les rendre interchangeables. Ce patron laisse les algorithmes changer indépendamment des clients qui les emploient. Nous l’utilisons pour le chef de cuisine quand il doit ordonner la préparation des plats.

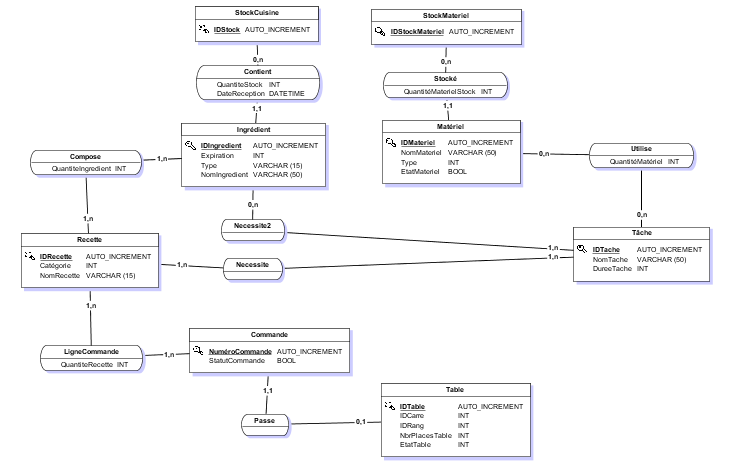
#### Factory

Le Design Pattern Factory définit une interface pour la création d'un objet en déléguant à ses sous-classes le choix des classes à instancier. Nous l’utilisons lors de la création des plats issues des recettes par les cuisiniers.

#### Adapter

Le Design Pattern Adapter convertit l'interface d'une classe en une autre interface exploitée par une application. Permet d'interconnecter des classes qui sans cela seraient incompatibles. Il est utilisé dans le cas où un programme se sert d'une bibliothèque de classe qui ne correspond plus à l'utilisation qui en est faite, à la suite d'une mise à jour de la bibliothèque dont l'interface a changé. Un objet *adapteur* expose alors l'ancienne interface en utilisant les fonctionnalités de la nouvelle. Nous l’utilisons pour que le client puisse spécifier une demande aux serveurs.

### MCD

Le MCD est un modèle de données ou diagramme pour des descriptions de haut niveau de modèles conceptuels de données. Il fournit une description graphique pour représenter de tels modèles de données sous la forme de diagrammes contenant des entités et des associations. De tels modèles sont utilisés dans les phases amont de conception des systèmes informatiques.

### Script SQL

Voici le script SQL de notre MCD :

#------------------------------------------------------------

# Script MySQL.

#------------------------------------------------------------

#------------------------------------------------------------

# Table: Recette

#------------------------------------------------------------

CREATE TABLE Recette(

IDRecette Int Auto\_increment NOT NULL ,

Categorie Int NOT NULL ,

NomRecette Varchar (15) NOT NULL

,CONSTRAINT Recette\_PK PRIMARY KEY (IDRecette)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: StockCuisine

#------------------------------------------------------------

CREATE TABLE StockCuisine(

IDStock Int Auto\_increment NOT NULL

,CONSTRAINT StockCuisine\_PK PRIMARY KEY (IDStock)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Ingrédient

#------------------------------------------------------------

CREATE TABLE Ingredient(

IDIngredient Int Auto\_increment NOT NULL ,

Expiration Int NOT NULL ,

Type Varchar (15) NOT NULL ,

NomIngredient Varchar (50) NOT NULL ,

QuantiteStock Int NOT NULL ,

DateReception Datetime NOT NULL ,

IDStock Int NOT NULL

,CONSTRAINT Ingredient\_PK PRIMARY KEY (IDIngredient)

,CONSTRAINT Ingredient\_StockCuisine\_FK FOREIGN KEY (IDStock) REFERENCES StockCuisine(IDStock)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Tâche

#------------------------------------------------------------

CREATE TABLE Tache(

IDTache Int Auto\_increment NOT NULL ,

NomTache Varchar (50) NOT NULL ,

DureeTache Int NOT NULL

,CONSTRAINT Tache\_PK PRIMARY KEY (IDTache)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: StockMateriel

#------------------------------------------------------------

CREATE TABLE StockMateriel(

IDStockMateriel Int Auto\_increment NOT NULL

,CONSTRAINT StockMateriel\_PK PRIMARY KEY (IDStockMateriel)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Matériel

#------------------------------------------------------------

CREATE TABLE Materiel(

IDMateriel Int Auto\_increment NOT NULL ,

NomMateriel Varchar (50) NOT NULL ,

Type Int NOT NULL ,

EtatMateriel Bool NOT NULL ,

QuantiteMaterielStock Int NOT NULL ,

IDStockMateriel Int NOT NULL

,CONSTRAINT Materiel\_PK PRIMARY KEY (IDMateriel)

,CONSTRAINT Materiel\_StockMateriel\_FK FOREIGN KEY (IDStockMateriel) REFERENCES StockMateriel(IDStockMateriel)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Table

#------------------------------------------------------------

CREATE TABLE Table(

IDTable Int Auto\_increment NOT NULL ,

IDCarre Int NOT NULL ,

IDRang Int NOT NULL ,

NbrPlacesTable Int NOT NULL ,

EtatTable Int NOT NULL

,CONSTRAINT Table\_PK PRIMARY KEY (IDTable)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Commande

#------------------------------------------------------------

CREATE TABLE Commande(

NumeroCommande Int Auto\_increment NOT NULL ,

StatutCommande Bool NOT NULL ,

IDTable Int NOT NULL

,CONSTRAINT Commande\_PK PRIMARY KEY (NumeroCommande)

,CONSTRAINT Commande\_Table\_FK FOREIGN KEY (IDTable) REFERENCES Table(IDTable)

,CONSTRAINT Commande\_Table\_AK UNIQUE (IDTable)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: LigneCommande

#------------------------------------------------------------

CREATE TABLE LigneCommande(

IDRecette Int NOT NULL ,

NumeroCommande Int NOT NULL ,

QuantiteRecette Int NOT NULL

,CONSTRAINT LigneCommande\_PK PRIMARY KEY (IDRecette,NumeroCommande)

,CONSTRAINT LigneCommande\_Recette\_FK FOREIGN KEY (IDRecette) REFERENCES Recette(IDRecette)

,CONSTRAINT LigneCommande\_Commande0\_FK FOREIGN KEY (NumeroCommande) REFERENCES Commande(NumeroCommande)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Compose

#------------------------------------------------------------

CREATE TABLE Compose(

IDIngredient Int NOT NULL ,

IDRecette Int NOT NULL ,

QuantiteIngredient Int NOT NULL

,CONSTRAINT Compose\_PK PRIMARY KEY (IDIngredient,IDRecette)

,CONSTRAINT Compose\_Ingredient\_FK FOREIGN KEY (IDIngredient) REFERENCES Ingredient(IDIngredient)

,CONSTRAINT Compose\_Recette0\_FK FOREIGN KEY (IDRecette) REFERENCES Recette(IDRecette)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Necessite

#------------------------------------------------------------

CREATE TABLE Necessite(

IDRecette Int NOT NULL ,

IDTache Int NOT NULL

,CONSTRAINT Necessite\_PK PRIMARY KEY (IDRecette,IDTache)

,CONSTRAINT Necessite\_Recette\_FK FOREIGN KEY (IDRecette) REFERENCES Recette(IDRecette)

,CONSTRAINT Necessite\_Tache0\_FK FOREIGN KEY (IDTache) REFERENCES Tache(IDTache)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Utilise

#------------------------------------------------------------

CREATE TABLE Utilise(

IDMateriel Int NOT NULL ,

IDTache Int NOT NULL ,

QuantiteMateriel Int NOT NULL

,CONSTRAINT Necessite\_PK PRIMARY KEY (IDMateriel, IDTache)

,CONSTRAINT Necessite\_Recette\_FK FOREIGN KEY (IDMateriel) REFERENCES Recette(IDMateriel)

,CONSTRAINT Necessite\_Tache0\_FK FOREIGN KEY (IDTache) REFERENCES Tache(IDTache)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Necessaire2

#------------------------------------------------------------

CREATE TABLE Necessaire2(

IDIngredient Int NOT NULL ,

IDTache Int NOT NULL

,CONSTRAINT Necessite\_PK PRIMARY KEY (IDIngredient, IDTache)

,CONSTRAINT Necessite\_Recette\_FK FOREIGN KEY (IDIngredient) REFERENCES Recette(IDIngredient)

,CONSTRAINT Necessite\_Tache0\_FK FOREIGN KEY (IDTache) REFERENCES Tache(IDTache)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Passe

#------------------------------------------------------------

CREATE TABLE Passe(

NumeroCommande Int NOT NULL ,

IDTable Int NOT NULL

,CONSTRAINT Necessite\_PK PRIMARY KEY (NumeroCommande, IDTable)

,CONSTRAINT Necessite\_Recette\_FK FOREIGN KEY (NumeroCommande) REFERENCES Recette(NumeroCommande)

,CONSTRAINT Necessite\_Tache0\_FK FOREIGN KEY (IDTable) REFERENCES Tache(IDTable)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Contient

#------------------------------------------------------------

CREATE TABLE Contient(

IDStock Int NOT NULL ,

IDIngredient Int NOT NULL

,CONSTRAINT Necessite\_PK PRIMARY KEY (IDS IDIngredient tock, IDTable)

,CONSTRAINT Necessite\_Recette\_FK FOREIGN KEY (IDStock) REFERENCES Recette(IDStock)

,CONSTRAINT Necessite\_Tache0\_FK FOREIGN KEY (IDIngredient) REFERENCES Tache(IDIngredient)

)ENGINE=InnoDB;

#------------------------------------------------------------

# Table: Stocke

#------------------------------------------------------------

CREATE TABLE Stocke (

IDStockMateriel Int NOT NULL ,

IDMateriel Int NOT NULL

QuantiteMaterielStock Int NOT NULL

,CONSTRAINT Necessite\_PK PRIMARY KEY (IDS IDStockMateriel, IDMateriel)

,CONSTRAINT Necessite\_Recette\_FK FOREIGN KEY (IDStockMateriel) REFERENCES Recette(IDStockMateriel)

,CONSTRAINT Necessite\_Tache0\_FK FOREIGN KEY (IDMateriel) REFERENCES Tache(IDMateriel)

)ENGINE=InnoDB;