

Projet Programmation Système A3

*« Master Chef Info »*

Table des matières

[UML 4](#_Toc531856080)

[Diagrammes de cas d’utilisations 5](#_Toc531856081)

[Diagrammes d’activités 7](#_Toc531856083)

[Diagrammes de séquences 15](#_Toc531856084)

[Diagrammes de classes 15](#_Toc531856086)

[Diagramme de composants 17](#_Toc531856087)

[Base de données 18](#_Toc531856088)

[MCD 18](#_Toc531856089)

[Script SQL Base de données 18](#_Toc531856090)

[Design Pattern 19](#_Toc531856091)

[Factory 19](#_Toc531856092)

[Singleton 19](#_Toc531856093)

[Observer 19](#_Toc531856094)

[Strategy 19](#_Toc531856095)

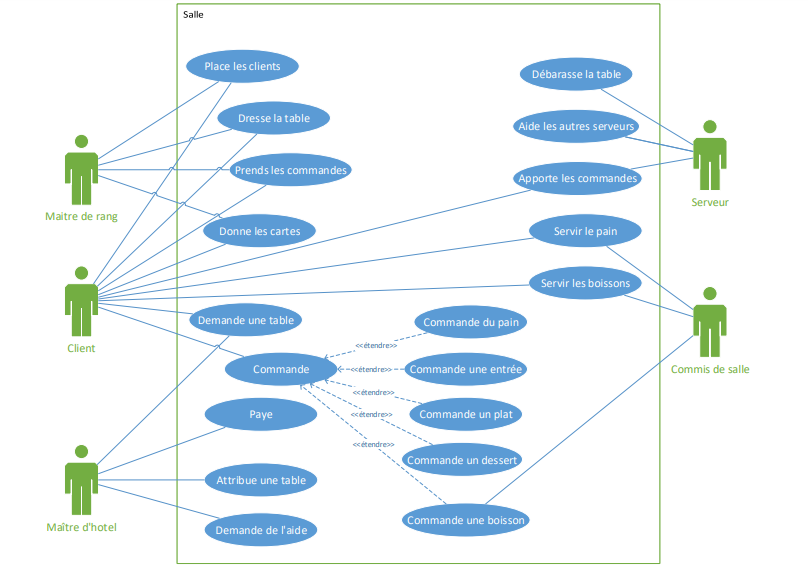
[MVC 19](#_Toc531856096)

## UML

### Diagrammes de cas d’utilisations

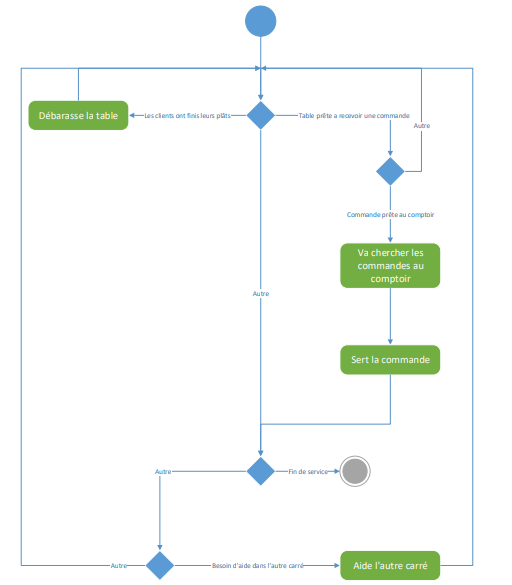
### 

Cuisine

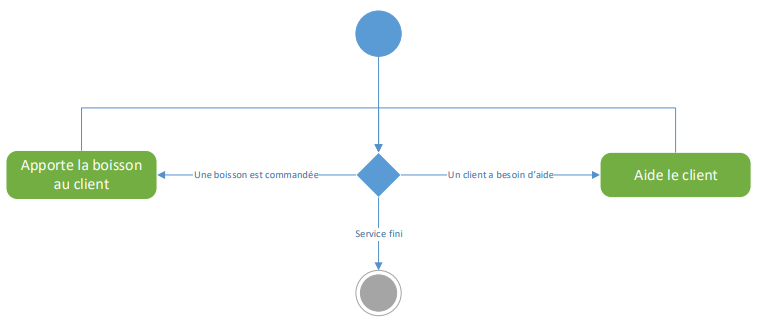


Salle

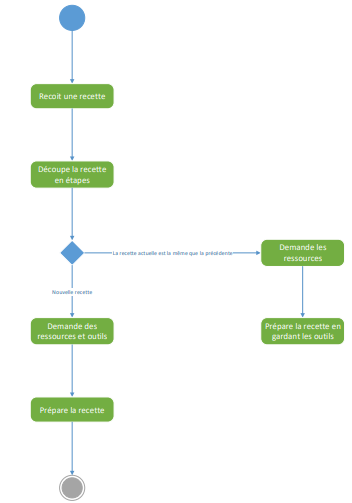
### Diagrammes d’activités



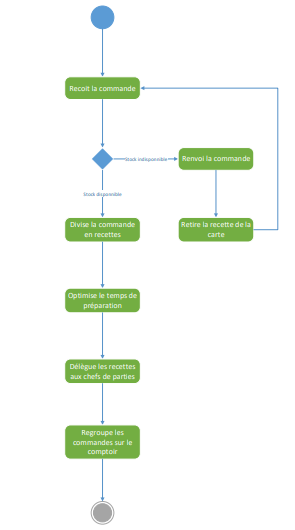
Serveur



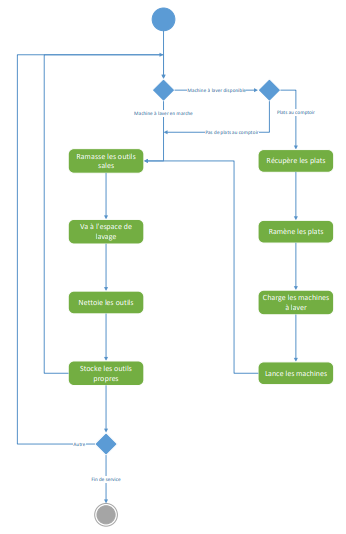
Commis de salle



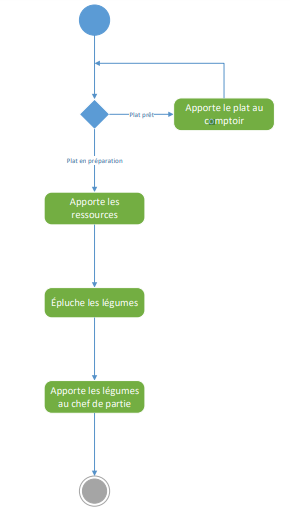
Chef de partie



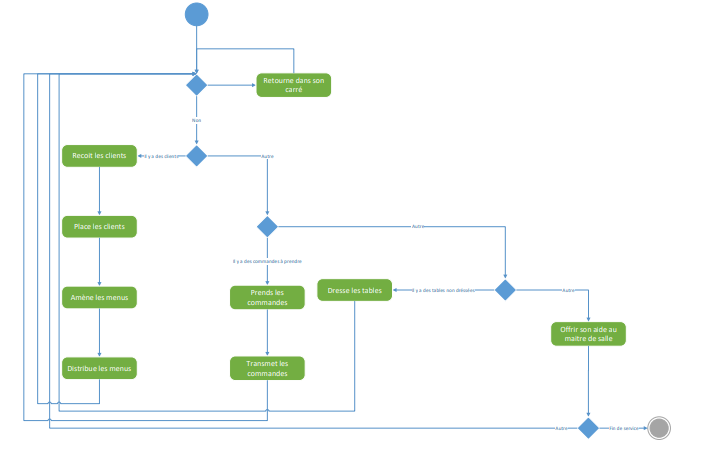
Chef de cuisine



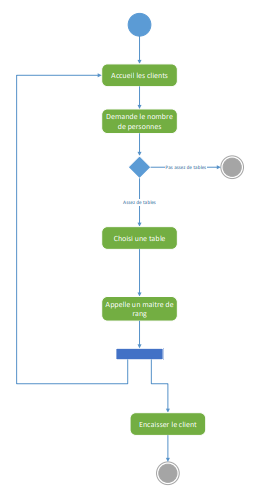
Plongeur



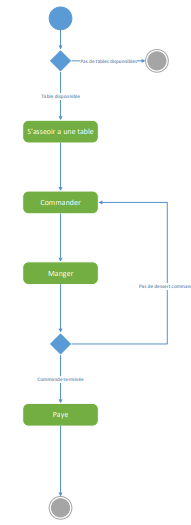
Commis de cuisine



Maitre de rang



Maitre de salle

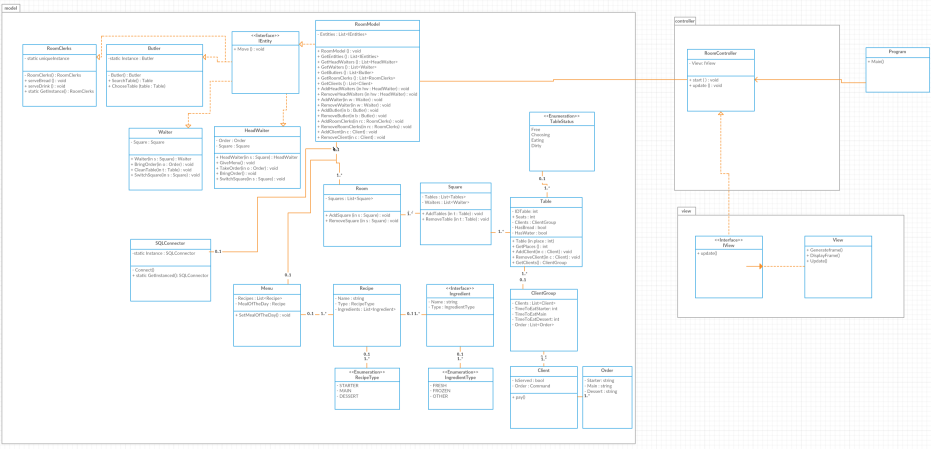


Client

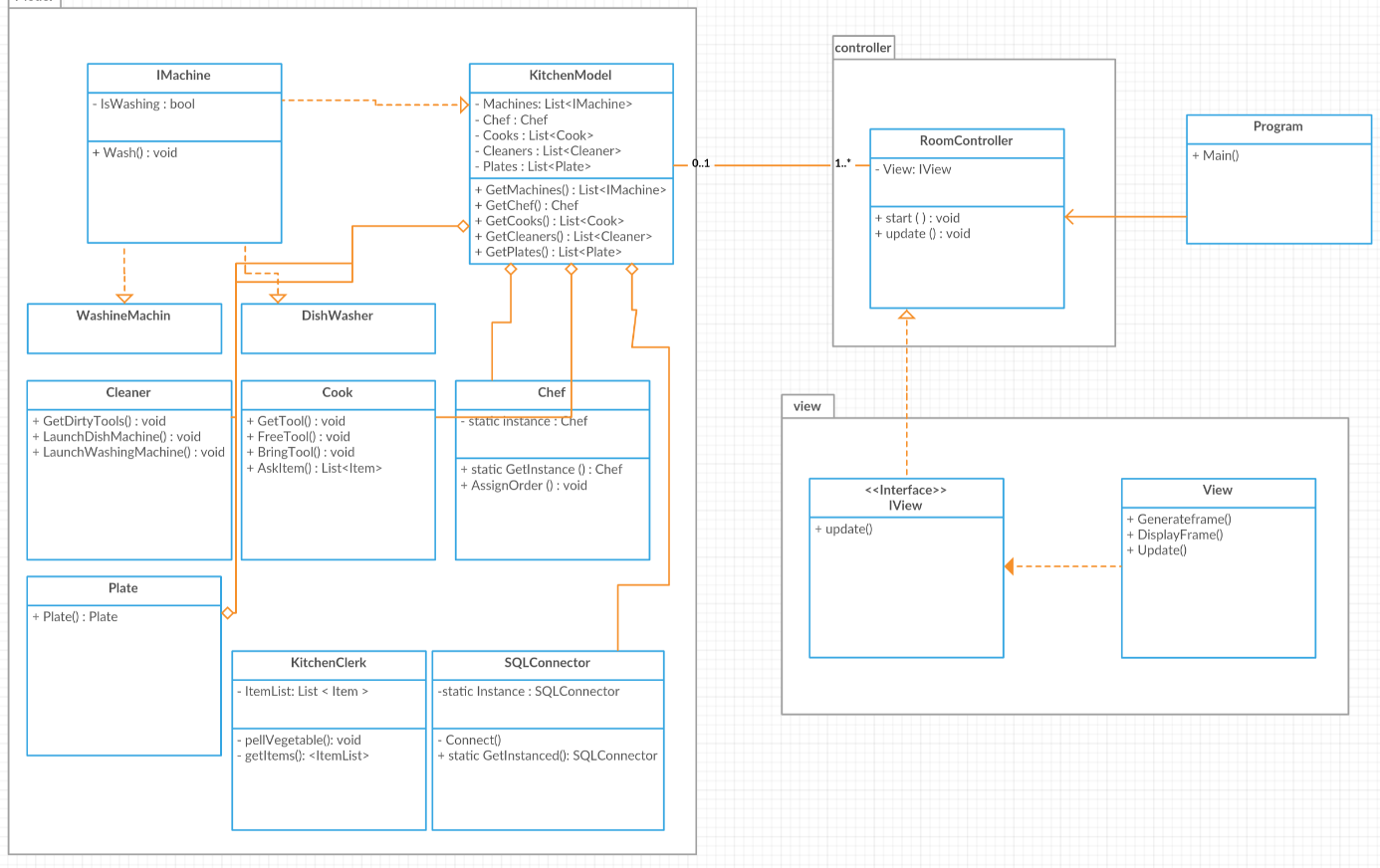
### Diagrammes de séquences

### 

### Diagrammes de classes



Salle



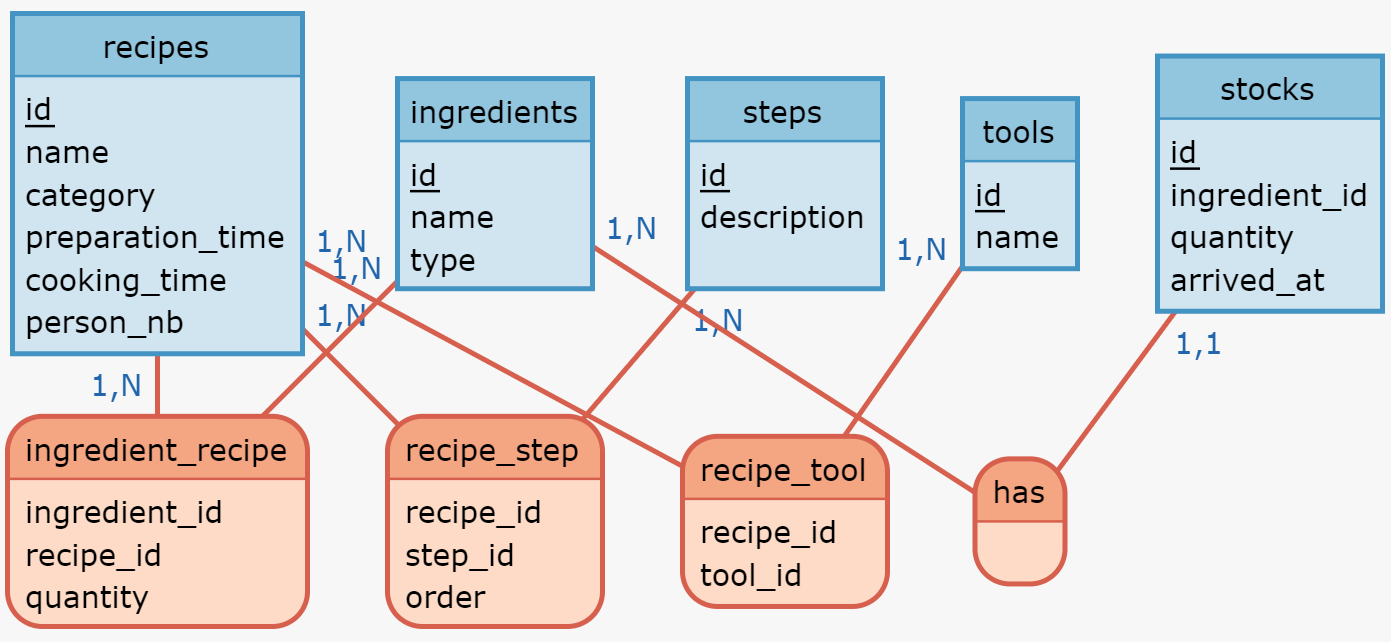
Cuisine

### Diagramme de composants

## https://puu.sh/Cdjhp/a9774a1022.png

## Base de données

### MCD



### Script SQL Base de données

CREATE DATABASE IF NOT EXISTS `master\_chief` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

USE `master\_chief`;

CREATE TABLE `RECIPES` (

`id` INT(11) NOT NULL,

`name` VARCHAR(63),

`category` VARCHAR(63),

`preparation\_time` INT,

`cooking\_time` INT,

`person\_nb` INT,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `INGREDIENTS` (

`id` INT(11) NOT NULL,

`name` VARCHAR(63),

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `STEPS` (

`id` INT(11) NOT NULL,

`description` LONGTEXT,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `TOOLS` (

`id` INT(11) NOT NULL,

`name` VARCHAR(63),

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `STOCKS` (

`id` INT(11) NOT NULL,

`ingredient\_id` INT NOT NULL,

`quantity` INT,

`arrived\_at` DATE,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `INGREDIENT\_RECIPE` (

`id` INT(11) NOT NULL,

`ingredient\_id` INT NOT NULL,

`recipe\_id` INT NOT NULL,

`quantity` INT,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `RECIPE\_STEP` (

`id` INT(11) NOT NULL,

`recipe\_id` INT NOT NULL,

`step\_id` INT NOT NULL,

`order` INT,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `RECIPE\_TOOL` (

`id` INT(11) NOT NULL,

`recipe\_id` VARCHAR(42),

`tool\_id` VARCHAR(42),

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `STOCKS` ADD FOREIGN KEY (`ingredient\_id`) REFERENCES `INGREDIENTS` (`id`);

ALTER TABLE `INGREDIENT\_RECIPE` ADD FOREIGN KEY (`ingredient\_id`) REFERENCES `INGREDIENTS` (`id`);

ALTER TABLE `INGREDIENT\_RECIPE` ADD FOREIGN KEY (`recipe\_id`) REFERENCES `RECIPES` (`id`);

ALTER TABLE `RECIPE\_STEP` ADD FOREIGN KEY (`step\_id`) REFERENCES `STEPS` (`id`);

ALTER TABLE `RECIPE\_STEP` ADD FOREIGN KEY (`recipe\_id`) REFERENCES `RECIPES` (`id`);

ALTER TABLE `RECIPE\_TOOL` ADD FOREIGN KEY (`tool\_id`) REFERENCES `TOOLS` (`id`);

ALTER TABLE `RECIPE\_TOOL` ADD FOREIGN KEY (`recipe\_id`) REFERENCES `RECIPES` (`id`);

## Design Pattern

Factory : Nous utilisons le design pattern factory pour instancier des objets dérivé d’un type abstrait.

Ici notre Factory : EntityFactory permet d’instancier des entités (Client, Commit, Chef de rang, etc.) qui sont dérivées du type Entity.

Singleton : Sert à restreindre l’instanciation d’une classe à un seul objet.

Nous utilisons un Singleton pour les postes n’ayant qu’un employé, par exemple (il y a un seul maitre d’hôtel, un seul commit). Nous limitons donc l’instanciation de la classe commit à une seule instanciation.

Observer : Il sert à envoyer un signal à des modules qui jouent le rôle d’observateurs. En cas de notification, les observateurs effectuent l’action adéquate en fonction des informations qui parviennent depuis les modules qu’ils observent (les observables).

Ici, le maitre d’hôtel indique au maitre de rang qu’une table a été attribué à des clients et qu’il doit les accompagner à celle-ci

Strategy :

Permet de sélectionner des algorithmes à la volée au cours du temps d’exécution selon certaines conditions.

Permet de permuter dynamiquement les algorithmes utilisés dans une application

MVC : Modèle d’architecture logicielle destiné à séparer les interfaces graphiques de la partie back-end.

Elle est composée de 3 parties :

-Modèle (Données à afficher)

-Vue : Interface Graphique

-Contrôleur : Contient la logique des actions de l’utilisateur