Ce dossier présente l'architecture logiciel du projet. Il présente les différents digrammes UML, les patrons de conceptions suivis et le modèle conceptuel de la BDD.

Projet Programmation Système

Dossier Architecture (logiciel)

Par Paul B, Geoffroy dF, Antoine G, Martin R & Thibault S

SOMMAIRE

Table des matières

[I – Introduction 2](#_Toc531793577)

[II – Diagrammes Fonctionnels 2](#_Toc531793578)

[A – Diagramme de cas d’utilisation 2](#_Toc531793579)

[B – Diagramme d’activité de chaque poste de travail 3](#_Toc531793580)

[1 – Client 4](#_Toc531793581)

[2 – Maître d’hôtel 5](#_Toc531793582)

[3 – Chef de rang 6](#_Toc531793583)

[4 – Serveur 7](#_Toc531793584)

[5 – Commis de salle 8](#_Toc531793585)

[6 – Chef 9](#_Toc531793586)

[7 – Chef salé 10](#_Toc531793587)

[8 – Chef pâtissier 10](#_Toc531793588)

[9 – Commis de cuisine 11](#_Toc531793589)

[10 – Plongeur 11](#_Toc531793590)

[III – Diagrammes Techniques 12](#_Toc531793591)

[A – Diagramme de composants 12](#_Toc531793592)

[B – Diagramme de classes 13](#_Toc531793593)

[C – Diagramme de séquences 15](#_Toc531793594)

[IV – Les Design Pattern utilisés 16](#_Toc531793595)

[A – Singleton 16](#_Toc531793596)

[B – Observer 16](#_Toc531793597)

[C – Strategy 16](#_Toc531793598)

[D – Builder 16](#_Toc531793599)

[E – Factory 16](#_Toc531793600)

[F – MVC 16](#_Toc531793601)

[V – Modélisation 17](#_Toc531793602)

[A – MCD 17](#_Toc531793603)

[B – Script de création de la BDD 18](#_Toc531793604)

[VI – Conclusion 20](#_Toc531793605)

# I – Introduction

Pour ce projet, nous devons réaliser une application permettant de gérer le fonctionnement d’un restaurant afin de satisfaire au mieux les clients.

Ce premier rapport présente toutes nos architectures logicielles que l’on va utiliser. Il est possible d’y retrouver tous les diagrammes UML techniques (composants, classes, séquence) ou encore fonctionnels (cas d’utilisation, activité). Nous effectuerons également une présentation de tous les design pattern que nous utiliserons ainsi qu’une modélisation de notre architecture.

# II – Diagrammes Fonctionnels

## A – Diagramme de cas d’utilisation

Voici notre diagramme de cas d’utilisation. Celui-ci décrit précisément les fonctionnalités, les tâches qui seront représentés dans notre code, de manière à se rapprocher le plus de la réalité.

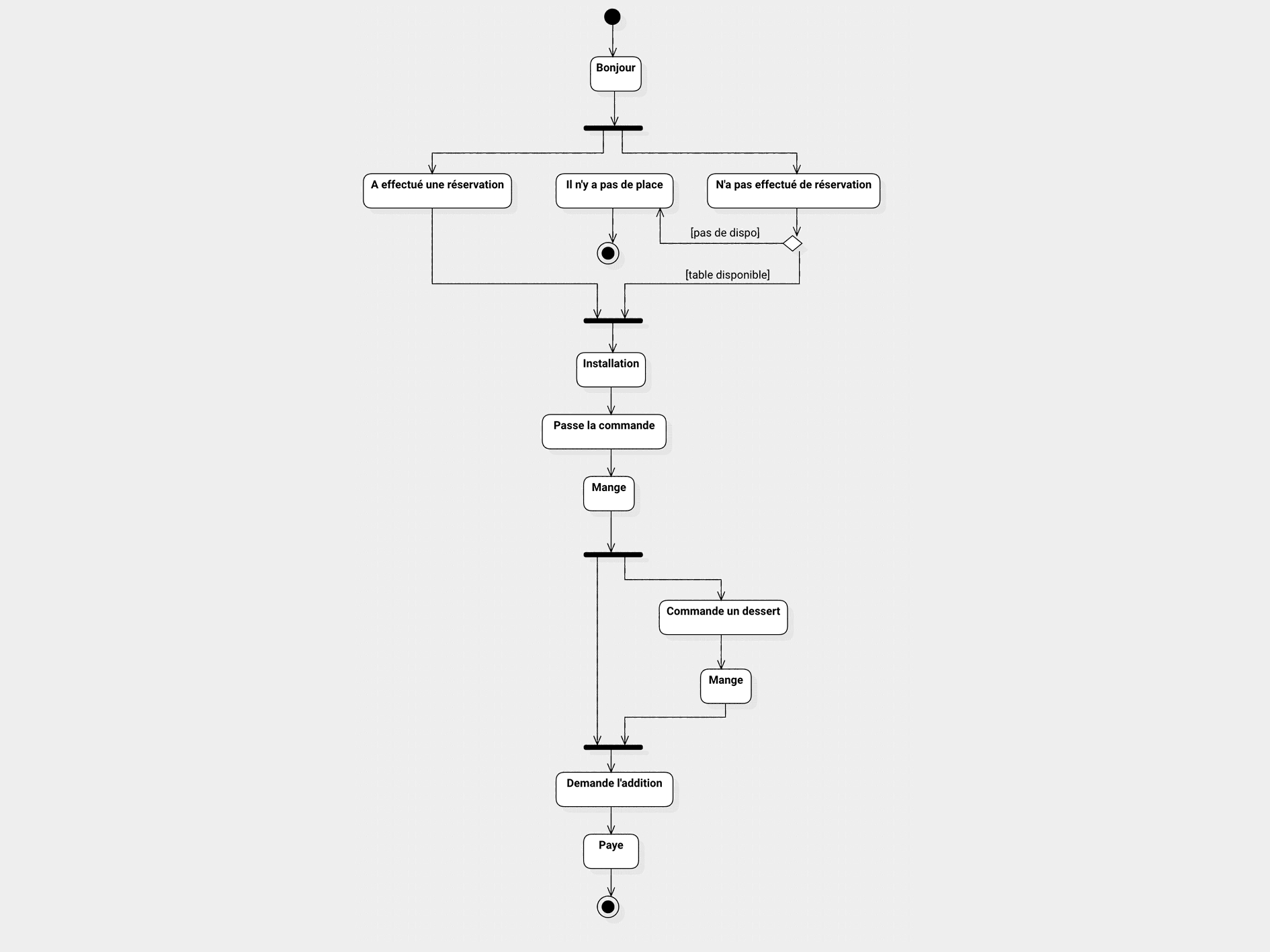
Ici, Le client il est accueilli et derrière il consomme son repas. L’accueil sollicite le chef de rang et le maître d’hôtel qui vont faire le lien avec la cuisine. Les différents chefs et le commis de cuisine vont préparer la commande. Une fois que cette dernière est prête, le client consomme son repas. Le serveur et le commis vont s’occuper du service/des tâches qui en découlent et le plongeur fait la plonge.

## B – Diagramme d’activité de chaque poste de travail

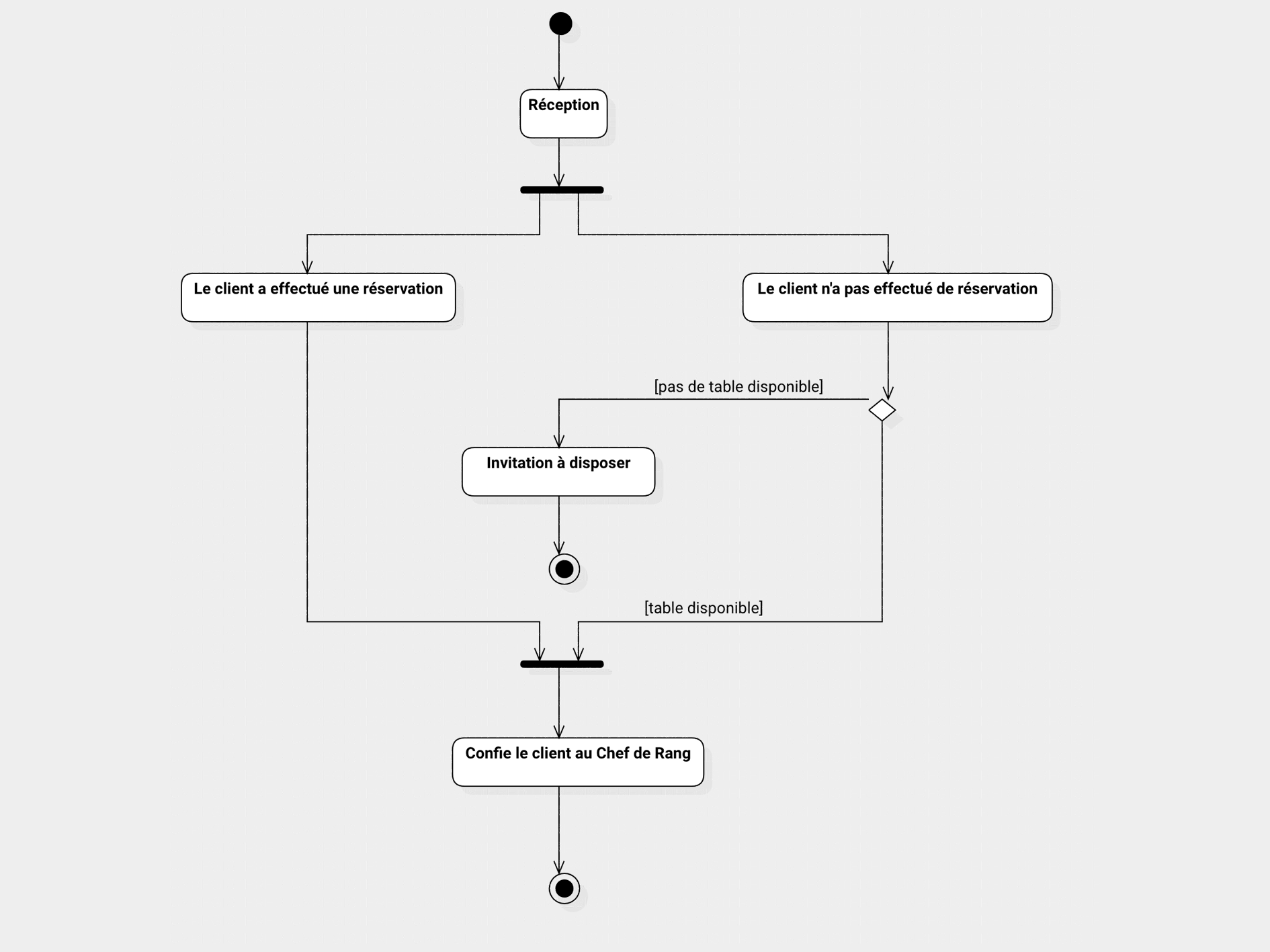
Les diagrammes d’activités représentent les activités de chaque poste de travail. On y retrouve un enchainement logique des actions qui sont réalisés par chaque personne dans notre programme.

Chaque diagramme présent à la suite représente un rôle dans le restaurant (du client au plongeur).

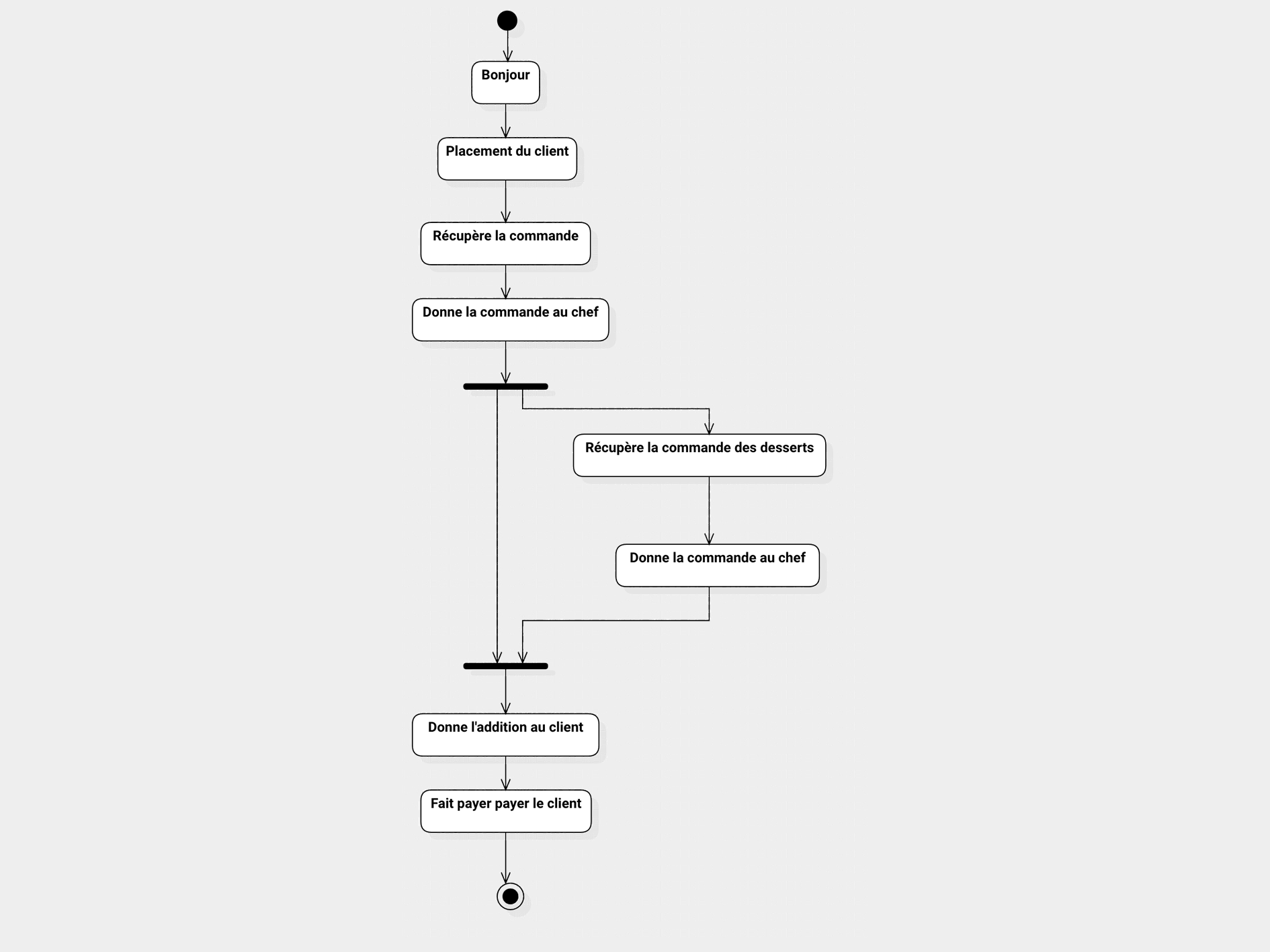
### 1 – Client



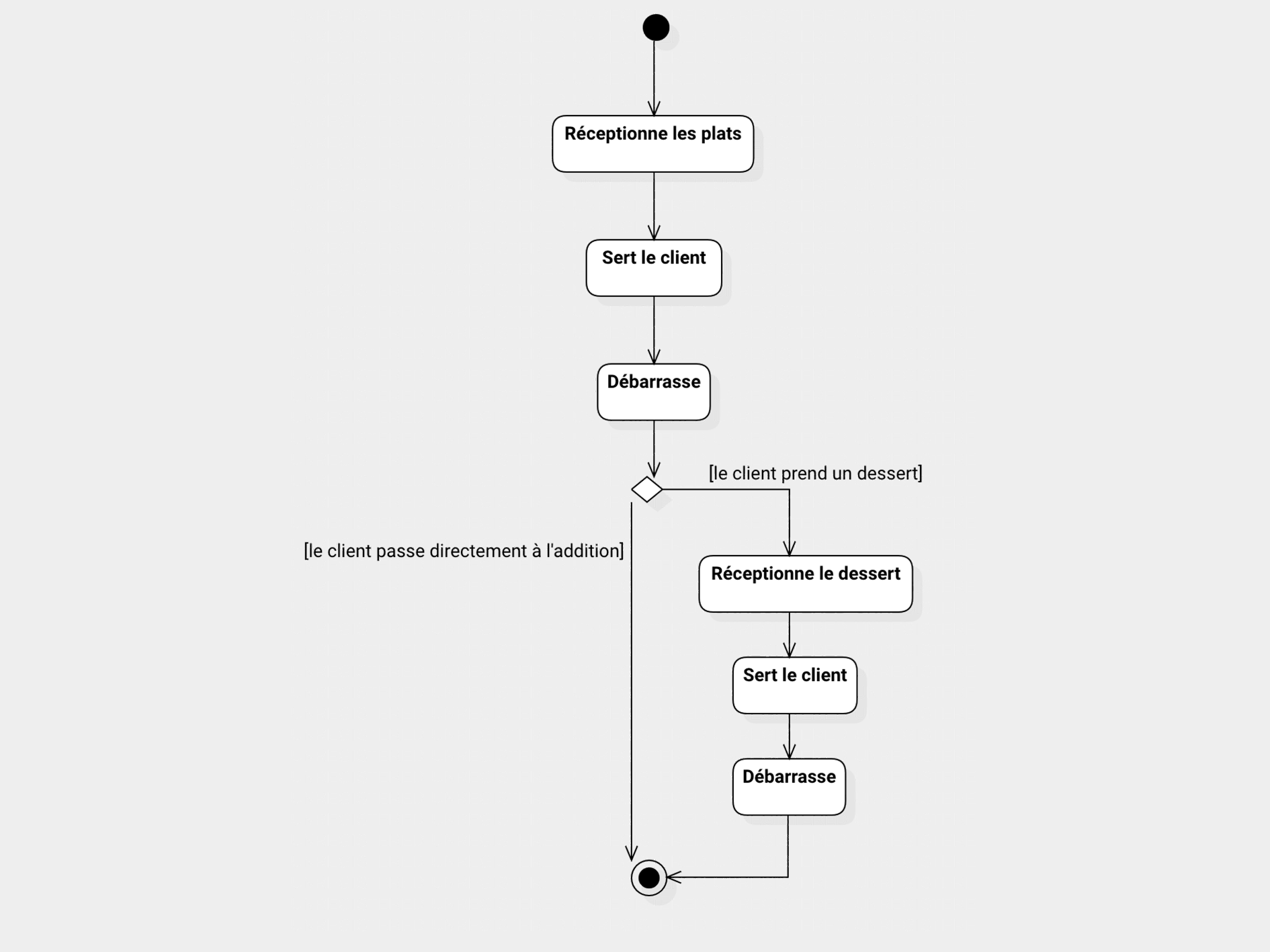
### 2 – Maître d’hôtel



### 3 – Chef de rang



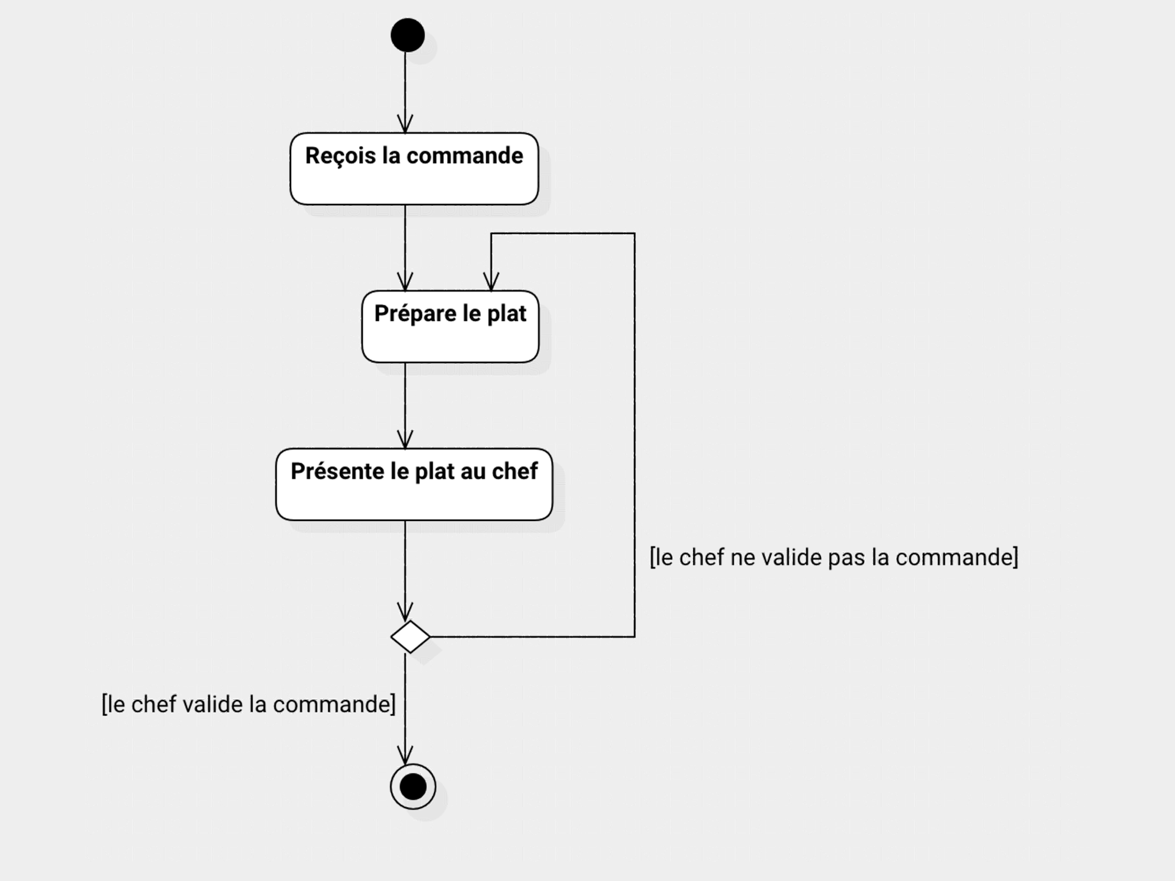
### 4 – Serveur



### 5 – Commis de salle

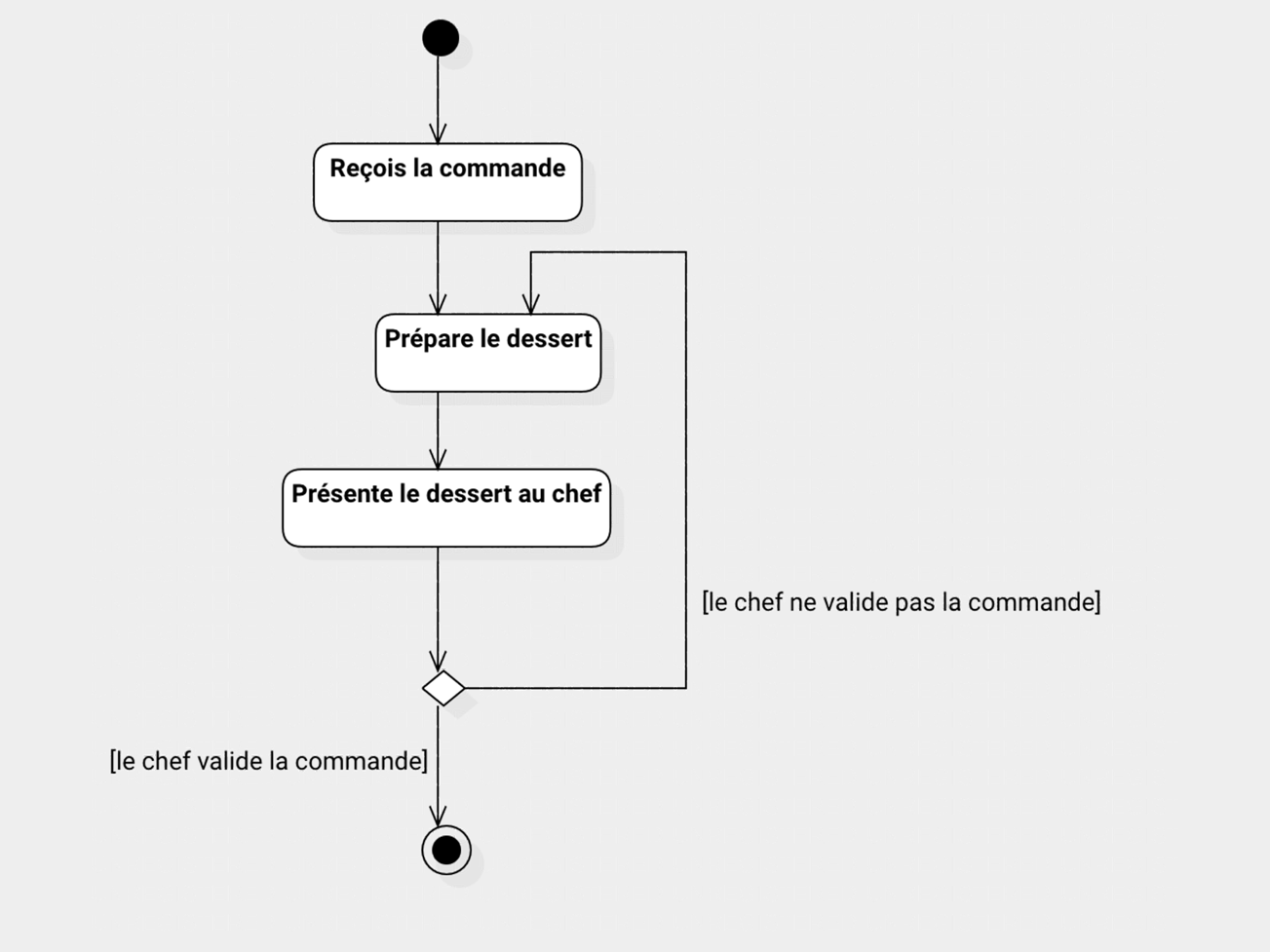
### 6 – Chef

### 7 – Chef salé

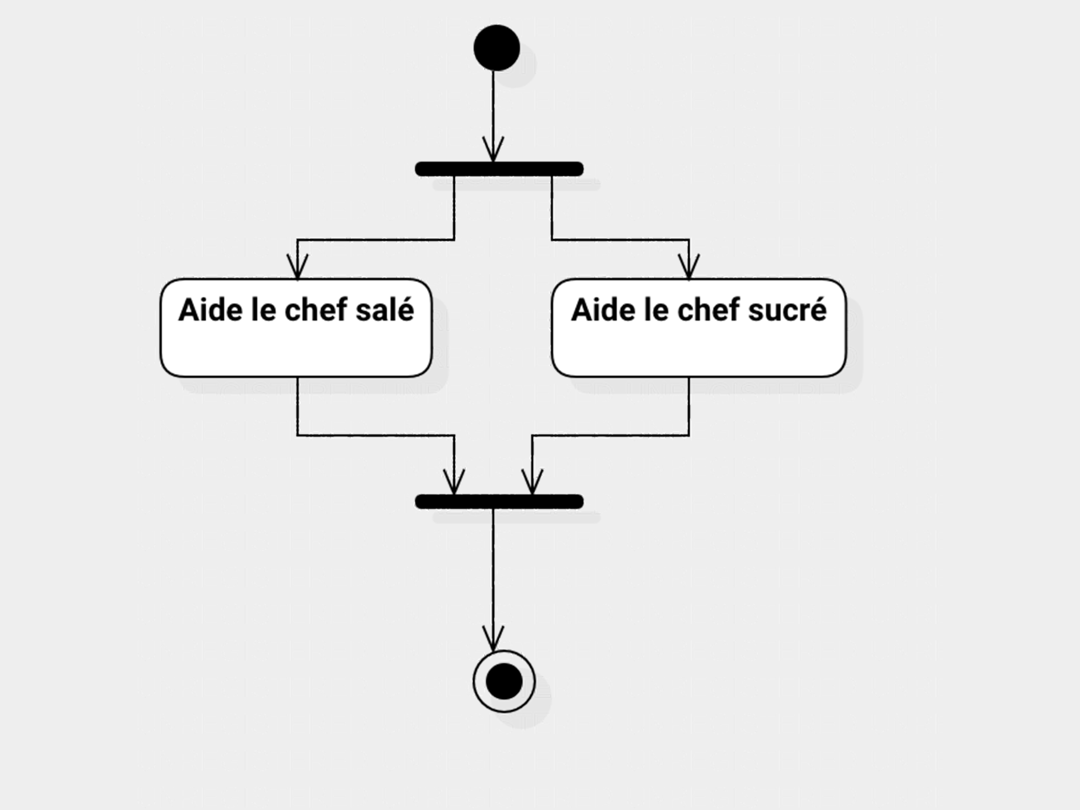


### 

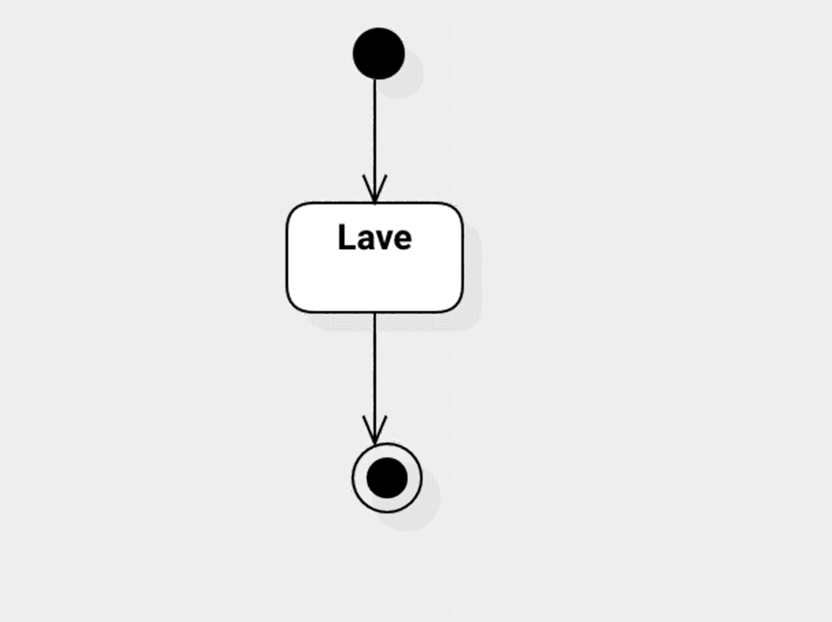
### 8 – Chef pâtissier



### 9 – Commis de cuisine

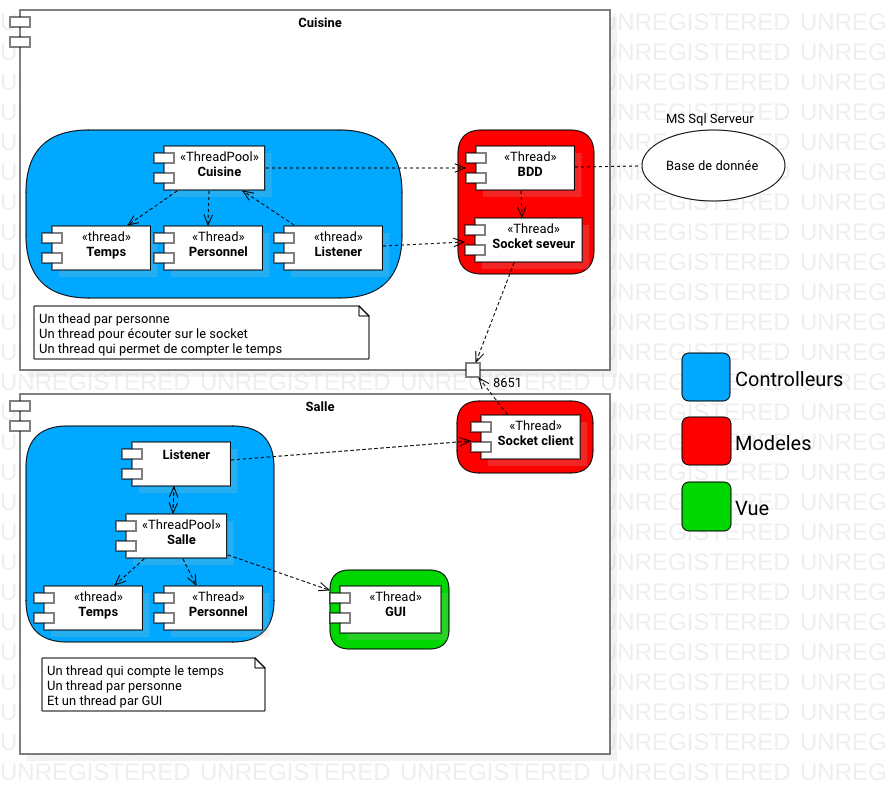


### 10 – Plongeur



# III – Diagrammes Techniques

## A – Diagramme de composants



Permet de visualisez nos deux programmes :

* Dans notre première application, pour gérer la Salle on peut retrouver le modèle MVC.

Notre vue va s’occupez de la salle, mais aussi de la cuisine.

* Dans notre deuxième application, qui elle gère la cuisine nous n’avons pas de vue, mais nous utilisons quand même les Contrôleurs et Modèles

Le but est d’utiliser un socket sur le port 8651 pour effectuer la communication entre nos deux applications

## 

## B – Diagramme de classes

Le diagramme représente les différents élément composant notre système et leurs relations.

On retrouve les deux programmes. D'un côté nous avons la classe Salle et la classe Cuisine.

La communication entre les deux s'effectue avec les classes CommunicationSalleCuisine et CommunicationCuisineSalle, basé sur un patron 'observer' avec les classes PrésentoireSalle et PassePlat.

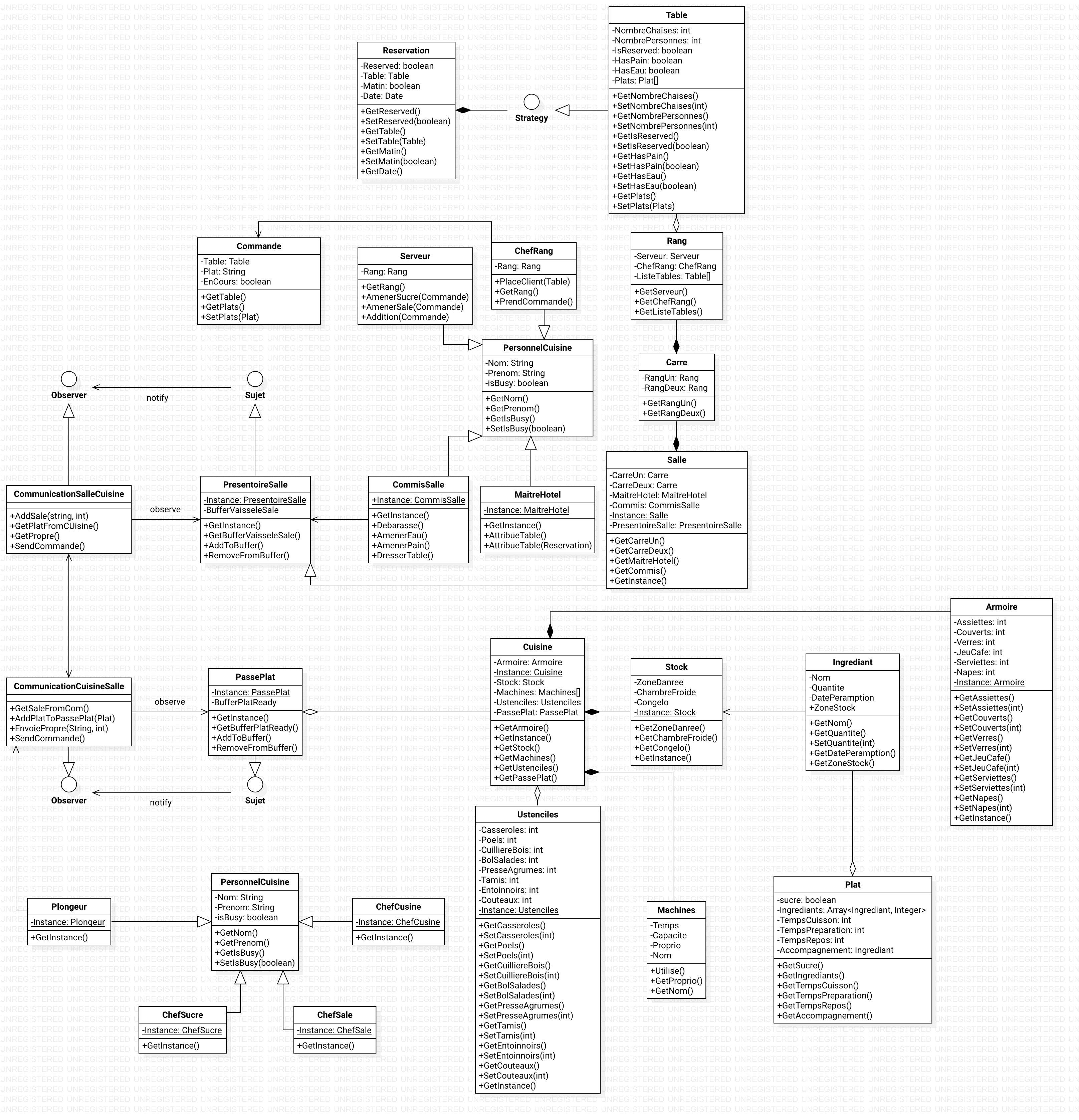
Dans les grandes lignes, le diagramme définit les classes suivantes :

Du coté Salle, nous retrouvons les classes Carre et Rang agrégeant de la classe Salle et la classe Table, composant la classe Rang.

Nous avons une classe PersonnelSalle de laquelle hérite des classes modélisant les différents membres du personnel de salle.

Du côté Cuisine, nous avons modélisé les classes Armoire, Stock et Machines, Aggrégant de la classe cuisine.

Nous avons une classe PersonnelCuisine de laquelle hérite des classes modélisant les différents membres du personnel de cuisine.



## C – Diagramme de séquences

Grâce à ce diagramme de séquence on peut facilement voir les interactions entre le client et les différents employés du restaurant.

On peut donc aussi voir quelles seront les interactions qui serons transmise par le socket entre la cuisine et la salle.

# IV – Les Design Pattern utilisés

## A – Singleton

Ce premier DP est utilisé pour tout ce qui est unique (exemple : le maître d’hôtel, la salle, la cuisine, etc.). L’objectif de ce DP est de restreindre l’instanciation d’une classe à un seul objet.

## B – Observer

Le rôle de ce DP, c’est qu’en cas de notification, les observateurs effectuent une action en fonction des informations qui leurs parviennent. Ici, c’est la communication entre la salle et la cuisine (c’est notre serveur client)

Exemple : dès qu’un plat est prêt, on le pose sur le passe-plat et la salle va être notifiée. Ensuite, le serveur amène le plat au client.

## C – Strategy

On utilise ce DP sur notre table et sur les réservations. Notre classe table possède une réservation qui sera amenée à changer régulièrement.

## D – Builder

On l’utilise avec les interactions avec la Base de Données (BdD). Tous les obj qui sont stockés dans la BdD utiliseront un Builder pour être utilisés dans l’application (exemple avec les ingrédients).

## E – Factory

On utilise ce DP pour les classes Cuisine Salle. Ce sont les deux classes qui vont fabriquer toutes celles qui en découlent.

Exemple : la salle va créer les carrés, les carrés vont créer les rangs, qui vont créer les tables.

## F – MVC

Modèle : Le Modèle va gérer tous ce qui se trouve en dehors de notre application, par exemple le modèle est en charge de la connexion a la base de données.  
Comme certaine de nos ressources sont stocker dans l’autre application, le modèle gérer aussi à connexion socket vers l’autre application.

Vue : La vue sera uniquement dans l’application « Salle » car notre serveur ne dispose pas d’interface graphique. Cette vue représentera la salle et la cuisine, ainsi que les différents employés et clients

Contrôleur : Le contrôleur va nous permettre de faire le lien entre la vue et le model. Le contrôleur répond aux actions effectuées sur la vue et modifie les données du modèle. Le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue.

# V – Modélisation

## A – MCD

## B – Script de création de la BDD

CREATE TABLE ZONE\_CONSERVATION(

    id INT IDENTITY (1,1) NOT NULL ,

    nom VARCHAR (50) NOT NULL ,

    quantite\_max INT NOT NULL ,

    tps\_conservation INT NOT NULL ,

    CONSTRAINT ZONE\_CONSERVATION\_PK PRIMARY KEY (id)

);

CREATE TABLE INGREDIENT(

    id INT IDENTITY (1,1) NOT NULL ,

    nom VARCHAR (50) NOT NULL ,

    quantite INT NOT NULL ,

    date\_peremption DATETIME NOT NULL ,

    id\_ZONE\_CONSERVATION INT NOT NULL ,

    CONSTRAINT INGREDIENT\_PK PRIMARY KEY (id)

    ,CONSTRAINT INGREDIENT\_ZONE\_CONSERVATION\_FK FOREIGN KEY (id\_ZONE\_CONSERVATION) REFERENCES ZONE\_CONSERVATION(id)

);

CREATE TABLE TYPE\_RECETTE(

    id INT IDENTITY (1,1) NOT NULL ,

    nom VARCHAR (50) NOT NULL ,

    CONSTRAINT TYPE\_RECETTE\_PK PRIMARY KEY (id)

);

CREATE TABLE TABLE(

    id INT IDENTITY (1,1) NOT NULL ,

    nb\_chaise INT NOT NULL ,

    pain bit NOT NULL ,

    bouteille bit NOT NULL ,

    CONSTRAINT TABLE\_PK PRIMARY KEY (id)

);

CREATE TABLE COUVERTS(

    id INT IDENTITY (1,1) NOT NULL ,

    nom VARCHAR (50) NOT NULL ,

    nb\_total INT NOT NULL ,

    nb\_propre INT NOT NULL ,

    nb\_sale INT NOT NULL ,

    type bit NOT NULL ,

    id\_TABLE INT ,

    CONSTRAINT COUVERTS\_PK PRIMARY KEY (id)

    ,CONSTRAINT COUVERTS\_TABLE\_FK FOREIGN KEY (id\_TABLE) REFERENCES TABLE(id)

);

CREATE TABLE USTENSILES(

    id INT IDENTITY (1,1) NOT NULL ,

    nom VARCHAR (50) NOT NULL ,

    nb\_total INT NOT NULL ,

    nb\_sale INT NOT NULL ,

    nb\_propre INT NOT NULL ,

    CONSTRAINT USTENSILES\_PK PRIMARY KEY (id)

);

CREATE TABLE RECETTE(

    id INT IDENTITY (1,1) NOT NULL ,

    nom VARCHAR (50) NOT NULL ,

    tps\_cuisson INT NOT NULL ,

    tps\_preparation INT NOT NULL ,

    tps\_repos INT NOT NULL ,

    accompagnement VARCHAR (20) NOT NULL ,

    id\_TYPE\_RECETTE INT NOT NULL ,

    id\_USTENSILES INT NOT NULL ,

    CONSTRAINT RECETTE\_PK PRIMARY KEY (id)

    ,CONSTRAINT RECETTE\_TYPE\_RECETTE\_FK FOREIGN KEY (id\_TYPE\_RECETTE) REFERENCES TYPE\_RECETTE(id)

    ,CONSTRAINT RECETTE\_USTENSILES0\_FK FOREIGN KEY (id\_USTENSILES) REFERENCES USTENSILES(id)

);

CREATE TABLE Composer(

    id INT NOT NULL ,

    id\_RECETTE INT NOT NULL ,

    quantite INT NOT NULL ,

    CONSTRAINT Composer\_PK PRIMARY KEY (id,id\_RECETTE)

    ,CONSTRAINT Composer\_INGREDIENT\_FK FOREIGN KEY (id) REFERENCES INGREDIENT(id)

    ,CONSTRAINT Composer\_RECETTE0\_FK FOREIGN KEY (id\_RECETTE) REFERENCES RECETTE(id)

);

CREATE TABLE COMMANDER(

    id INT NOT NULL ,

    id\_TABLE INT NOT NULL ,

    id\_INGREDIENT INT NOT NULL ,

    CONSTRAINT COMMANDER\_PK PRIMARY KEY (id,id\_TABLE,id\_INGREDIENT)

    ,CONSTRAINT COMMANDER\_RECETTE\_FK FOREIGN KEY (id) REFERENCES RECETTE(id)

    ,CONSTRAINT COMMANDER\_TABLE0\_FK FOREIGN KEY (id\_TABLE) REFERENCES TABLE(id)

    ,CONSTRAINT COMMANDER\_INGREDIENT1\_FK FOREIGN KEY (id\_INGREDIENT) REFERENCES INGREDIENT(id)

);

# VI – Conclusion

En conclusion, l'établissement de ce dossier d'architecture logiciel nous a permis de nous approprier le sujet. Cela nous a permis de réfléchir à la conception, en amont de la réalisation, en nous basant sur des patrons de conceptions précis. Le diagramme de classe et Le MCD nous ont d'ailleurs généré une partie du code et du script de celle-ci. Nous commençons donc la réalisation en étant désormais précisément guidé.