Projet programmation système

Master Chef info

A3
05/12/2018

Equipe :

- Dorian CLETON
- Nicolas HURTEVENT
- Robin MAISANO
- Mehdi SANSAL (Chef de projet)

Table des matières

Table des Figures :

I. 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.

II. Rappel de la demande

Dans le cadre du projet "Master Chef info" confié par le restaurant, notre équipe est chargée de réaliser un simulateur de restaurant simulant entièrement le fonctionnement d'un véritable restaurant dans les moindres détails :

Gestion de la salle :

- Accueil et redirection des clients vers leurs tables
- Prises des commandes
- Gestion de l'eau et du pain de la table
- Gestion du matériel tel que les couverts, nappes et serviettes
- Réception des plats au comptoir par le serveur qui les amène à leur table
- Encaissement

Gestion des cuisines :

- Réception des commandes depuis la salle
- Répartition des plats au chefs de parties par le chef de cuisine
- Délégation de certaines tâches aux commis par les chefs de partie
- Livraison du plat fini au comptoir par un commis
- Gestion des ustensiles de cuisine tel que les casseroles, feux de cuisson ou couteaux

Gestion des stocks :

- Réception du stock
- Consommation des produits en faisant les recettes

III. Gestion de projet :

Pour mener à bien tout projet, il est primordial de commencer par la partie gestion de projets qui consiste à découper le projet en plusieurs parties puis détailler ces dernières en tâches et enfin attribuer des ressources à chaque tâche.

Ce travail laborieux peut paraître comme une perte de temps mais en réalité cela permet de cerner les objectifs à atteindre et gérer au mieux chaque ressource ce qui se traduit par une efficacité accrue et de meilleurs délais.

1. PBS

Le PBS consiste en une décomposition du projet par livrable. Il permet donc le découpage du projet ce qui facilite la compréhension des attendus.
Nous avons défini notre PBS en fonction des différents livrables ci-dessous

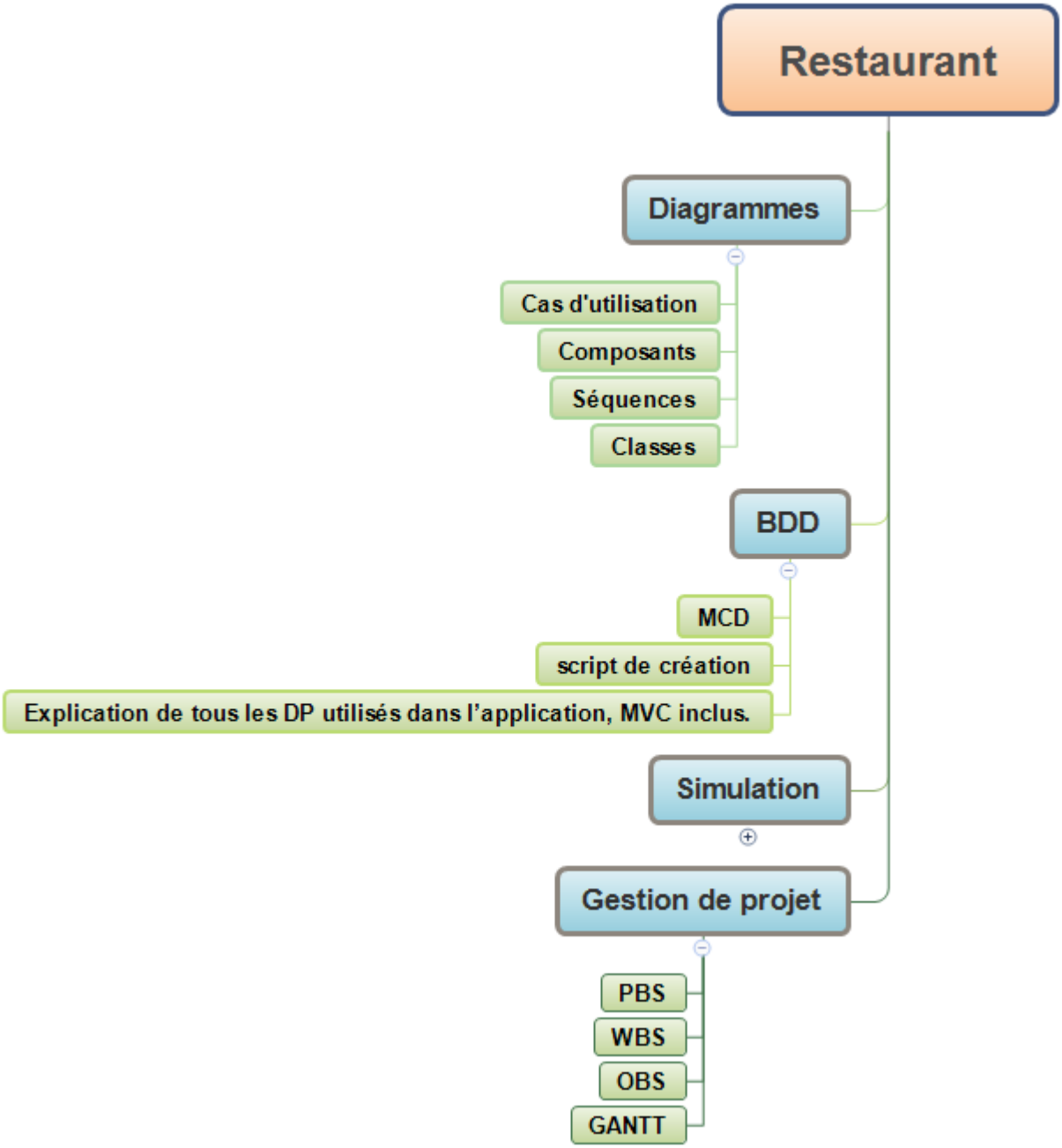


Figure 1: PBS

2. WBS

Il s'agit d'identifier toutes les tâches nécessaires à la réalisation du produit et à la conduite du projet.
Il est construit à partir du PBS, ce WBS est le travail à accomplir pour réaliser notre planning qui sera un outil essentiel dans le suivi et dans l'ordonnancement des tâches accomplir.
Nous avons réalisé notre WBS en découpant le projet en tâches :

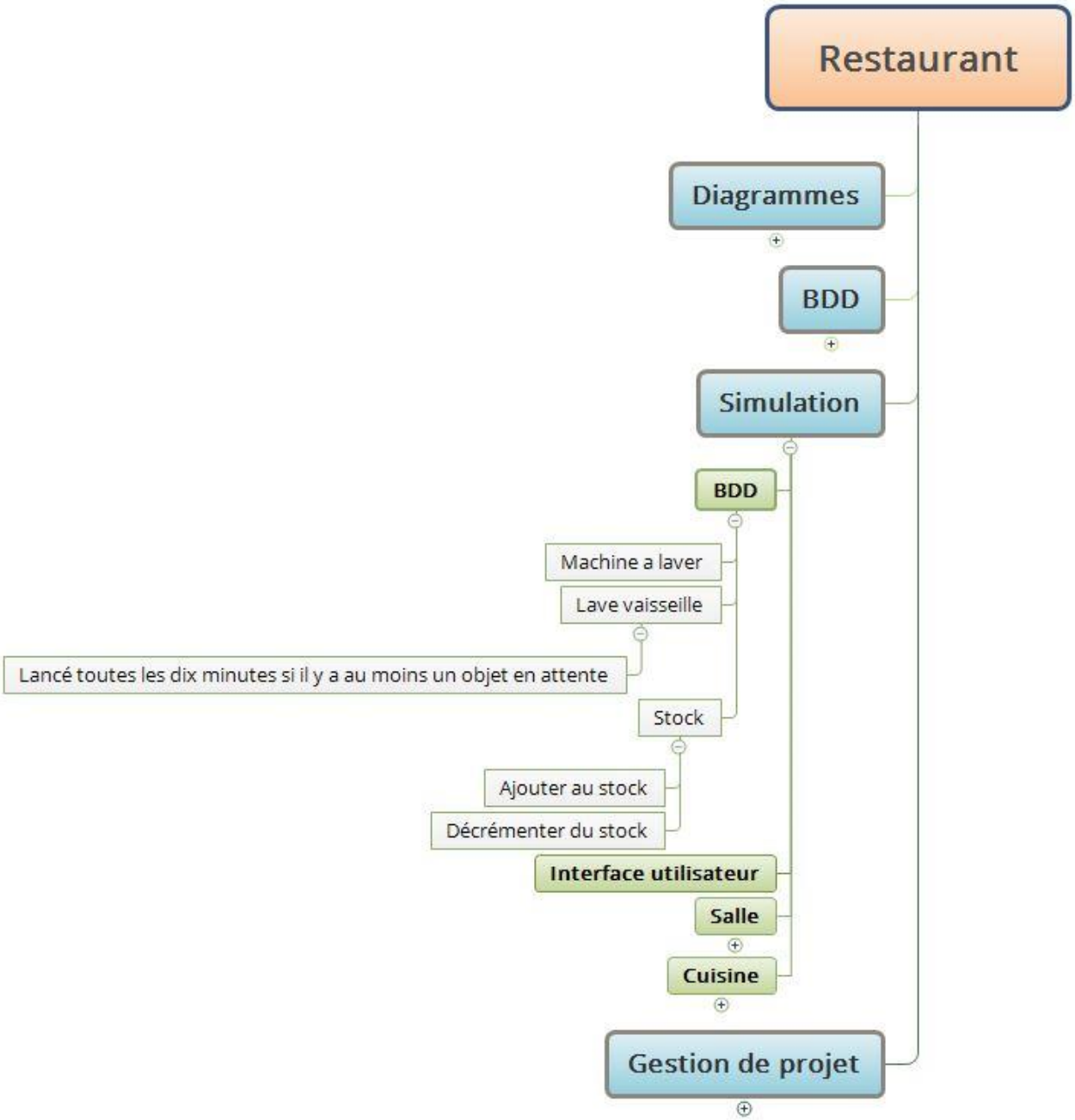


Figure 2: WBS général

Figure 3: WBS salle

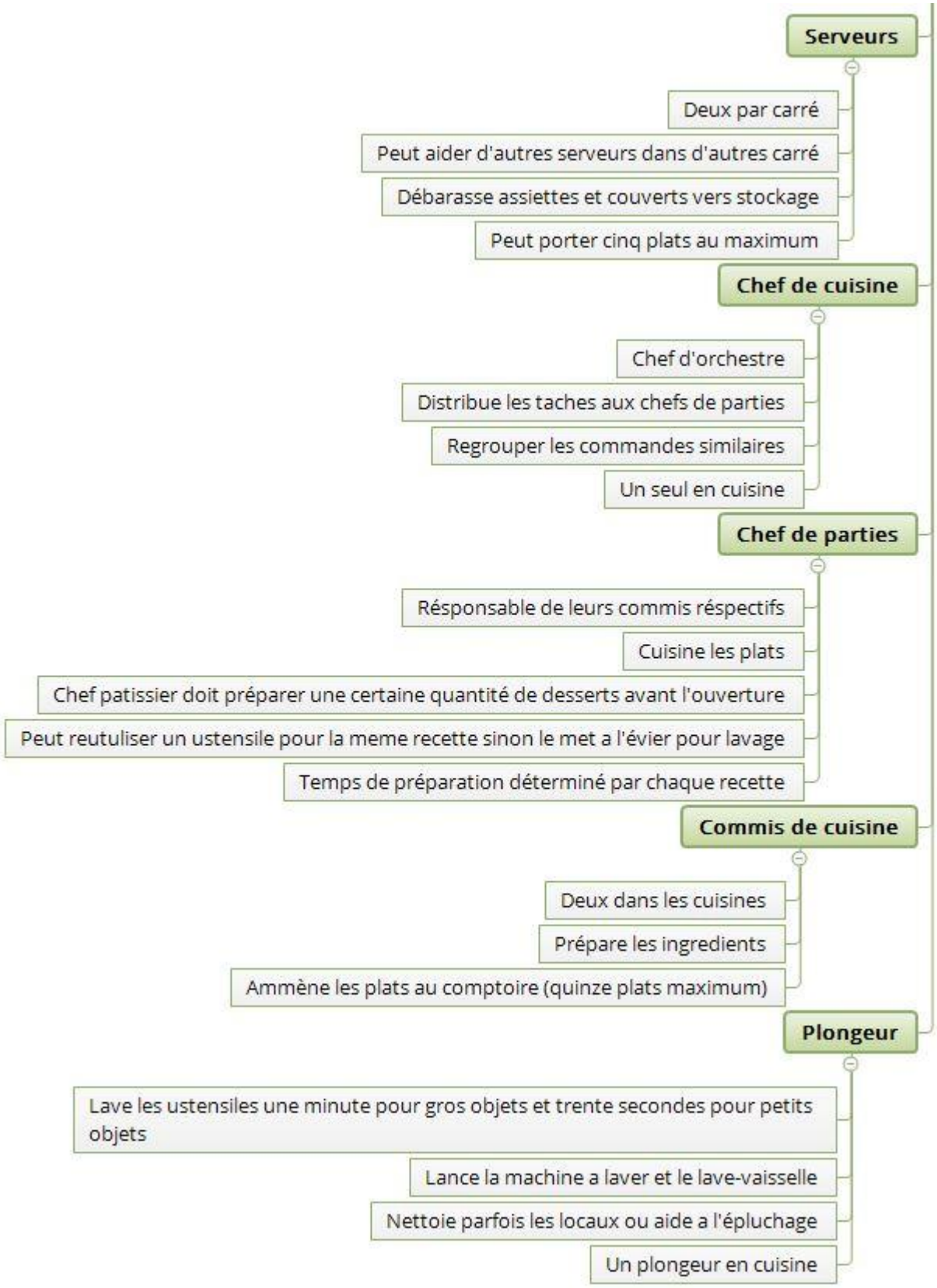


Figure 4: WBS cuisine

3. OBS

L'OBS sert à déterminer le rôle et de l'autorité de chaque membre du groupe, à chaque élément est attribué un responsable, avec les tâches et les objectifs correspondants.

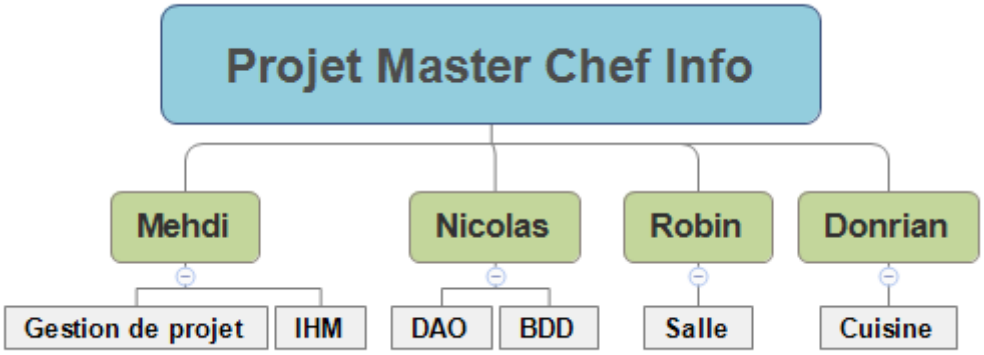
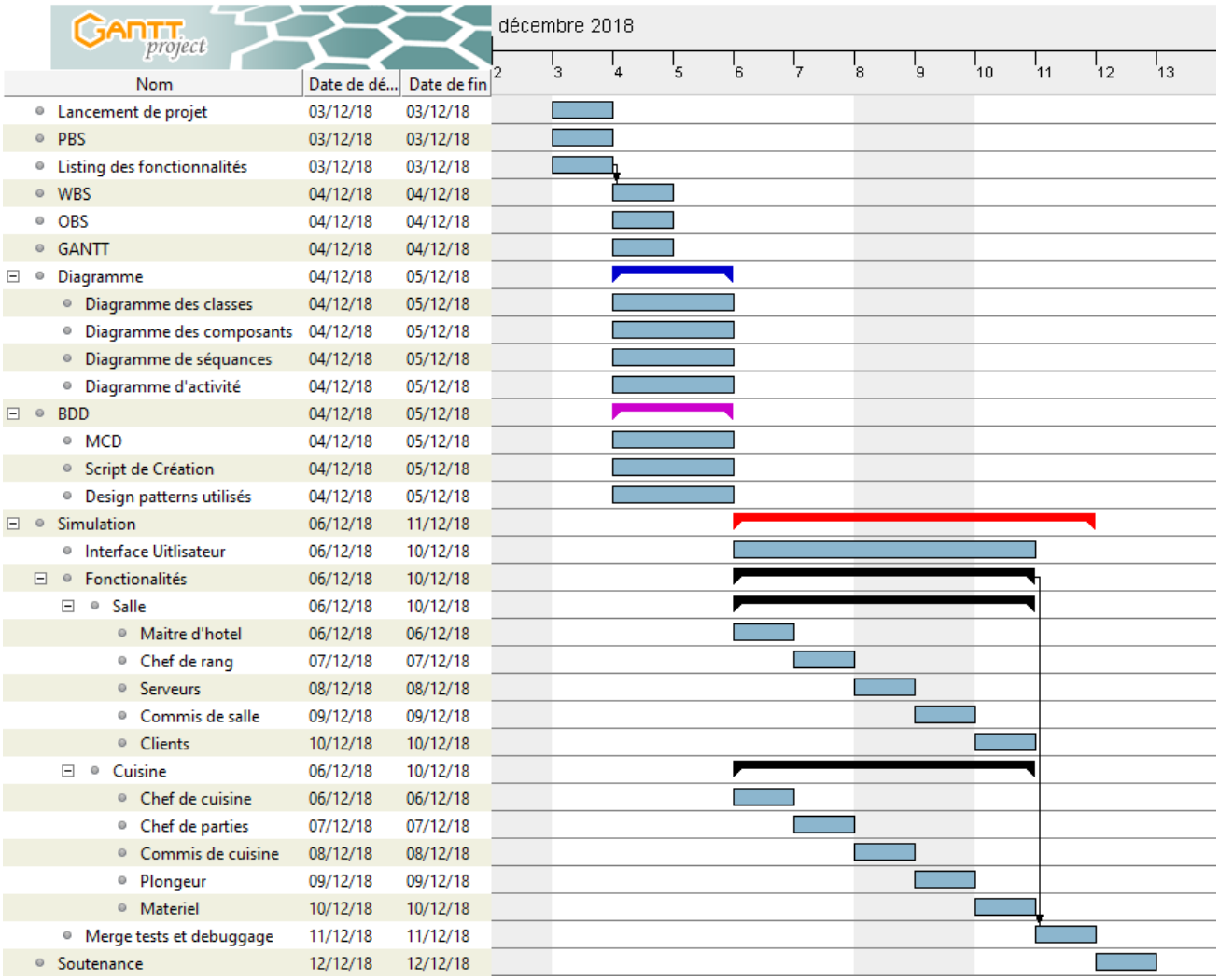


Figure 5: OBS

4. GANTT

Le diagramme de Gantt est très couramment utilisé en gestion de projet et est l'un des outils les plus efficaces pour représenter visuellement l'état d'avancement des différentes tâches qui constituent un projet.

Grace au diagramme de GANTT nous avons une visibilité détaillée de chaque tâche à accomplir, sa durée ainsi que les ressources qui lui sont allouées.



Nom	Date de dé...	Date de fin
Lancement de projet	03/12/18	03/12/18
PBS	03/12/18	03/12/18
Listing des fonctionnalités	03/12/18	03/12/18
WBS	04/12/18	04/12/18
OBS	04/12/18	04/12/18
GANTT	04/12/18	04/12/18
Diagramme	04/12/18	05/12/18
Diagramme des classes	04/12/18	05/12/18
Diagramme des composants	04/12/18	05/12/18
Diagramme de séquances	04/12/18	05/12/18
Diagramme d'activité	04/12/18	05/12/18
BDD	04/12/18	05/12/18
MCD	04/12/18	05/12/18
Script de Création	04/12/18	05/12/18
Design patterns utilisés	04/12/18	05/12/18
Simulation	06/12/18	11/12/18
Interface Uitlisateur	06/12/18	10/12/18
Fonctionalités	06/12/18	10/12/18
Salle	06/12/18	10/12/18
Maitre d'hotel	06/12/18	06/12/18
Chef de rang	07/12/18	07/12/18
Serveurs	08/12/18	08/12/18
Commis de salle	09/12/18	09/12/18
Clients	10/12/18	10/12/18
Cuisine	06/12/18	10/12/18
Chef de cuisine	06/12/18	06/12/18
Chef de parties	07/12/18	07/12/18
Commis de cuisine	08/12/18	08/12/18
Plongeur	09/12/18	09/12/18
Materiel	10/12/18	10/12/18
Merge tests et debuggage	11/12/18	11/12/18
Soutenance	12/12/18	12/12/18

Figure 6: Diagramme GANTT

IV. Architecture

1) Diagrammes :

a) Diagramme de cas d'utilisation

Le diagramme de cas d'utilisations permet de décrire l'interaction entre l'acteur et le système. L'idée est de dire que l'utilisateur d'un système logiciel a un objectif quand il utilise le système. Le cas d'utilisation est une description des interactions qui vont permettre à l'acteur d'atteindre son objectif en utilisant ce système.

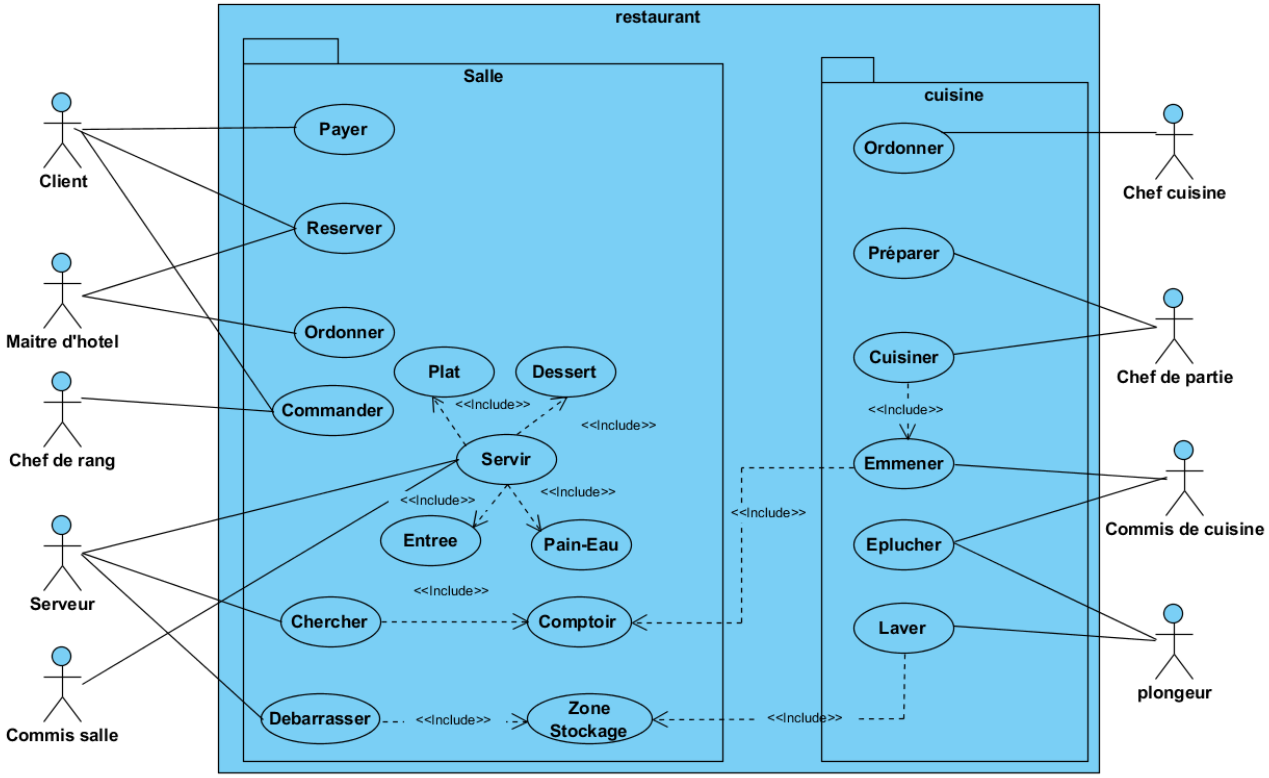


Figure 7: Diagramme de cas d'utilisation

b) Diagramme de séquences

Un diagramme de séquence est un type de diagramme d'interaction, il décrit comment et dans quel ordre plusieurs objets fonctionnent ensemble.

Les principales informations qu'ils contient sont les messages échangés entre les lignes de vie, présentés dans un ordre chronologique. Le temps y est représenté explicitement par une dimension (la verticale) et s'écoule de haut en bas.

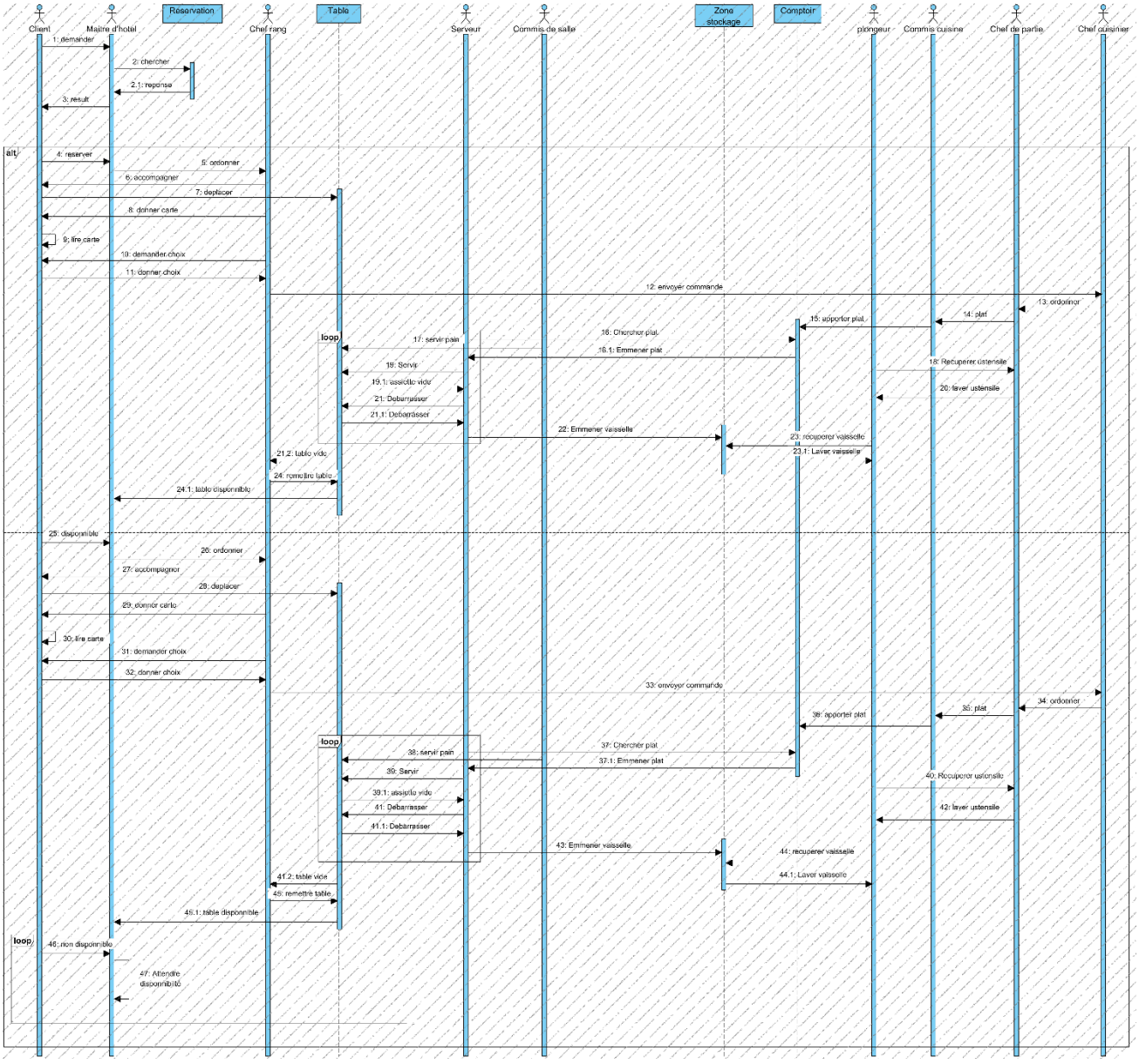


Figure 8: Vue globale du diagramme de séquences

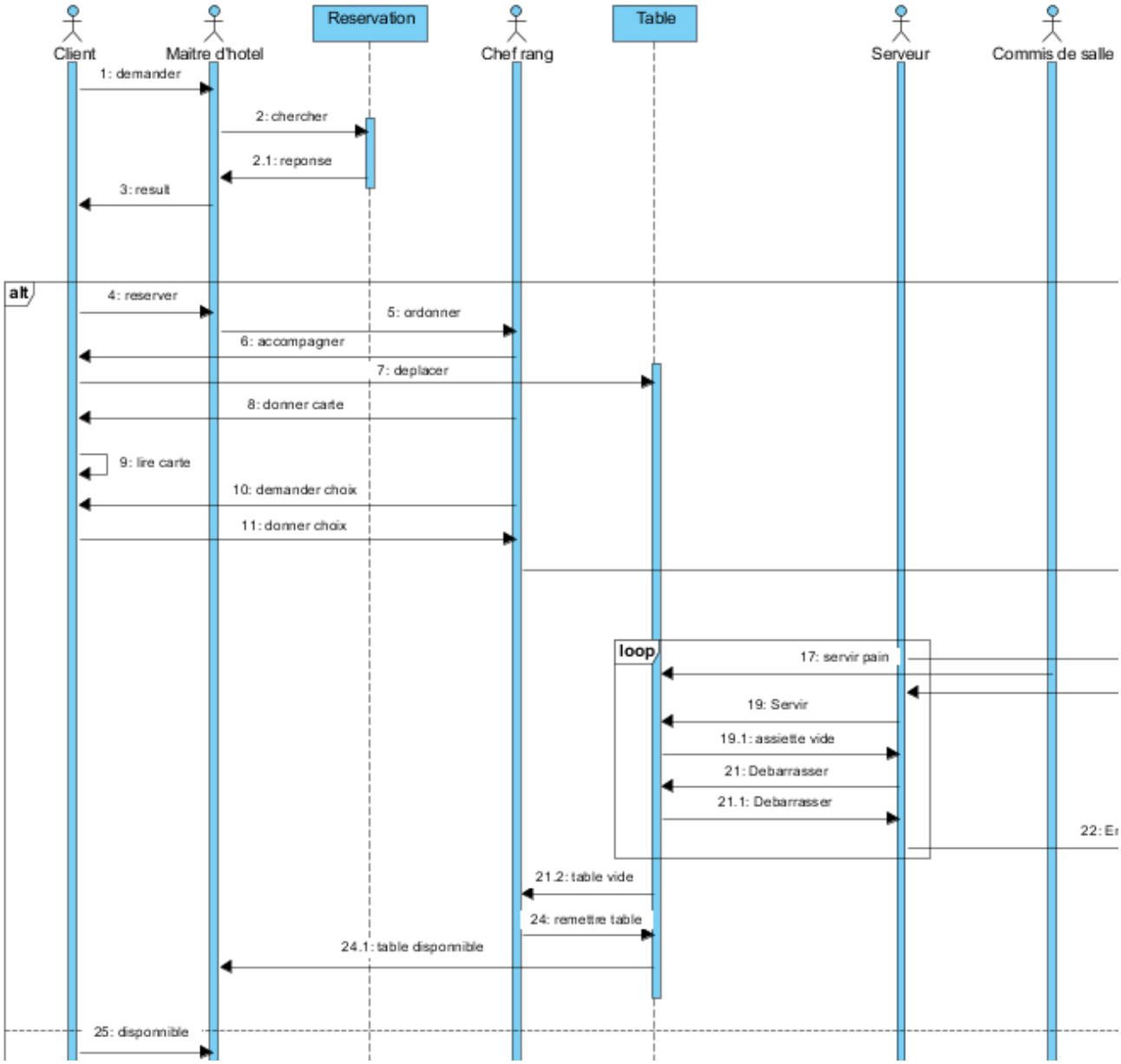


Figure 9: Partie haute gauche du diagramme de séquences

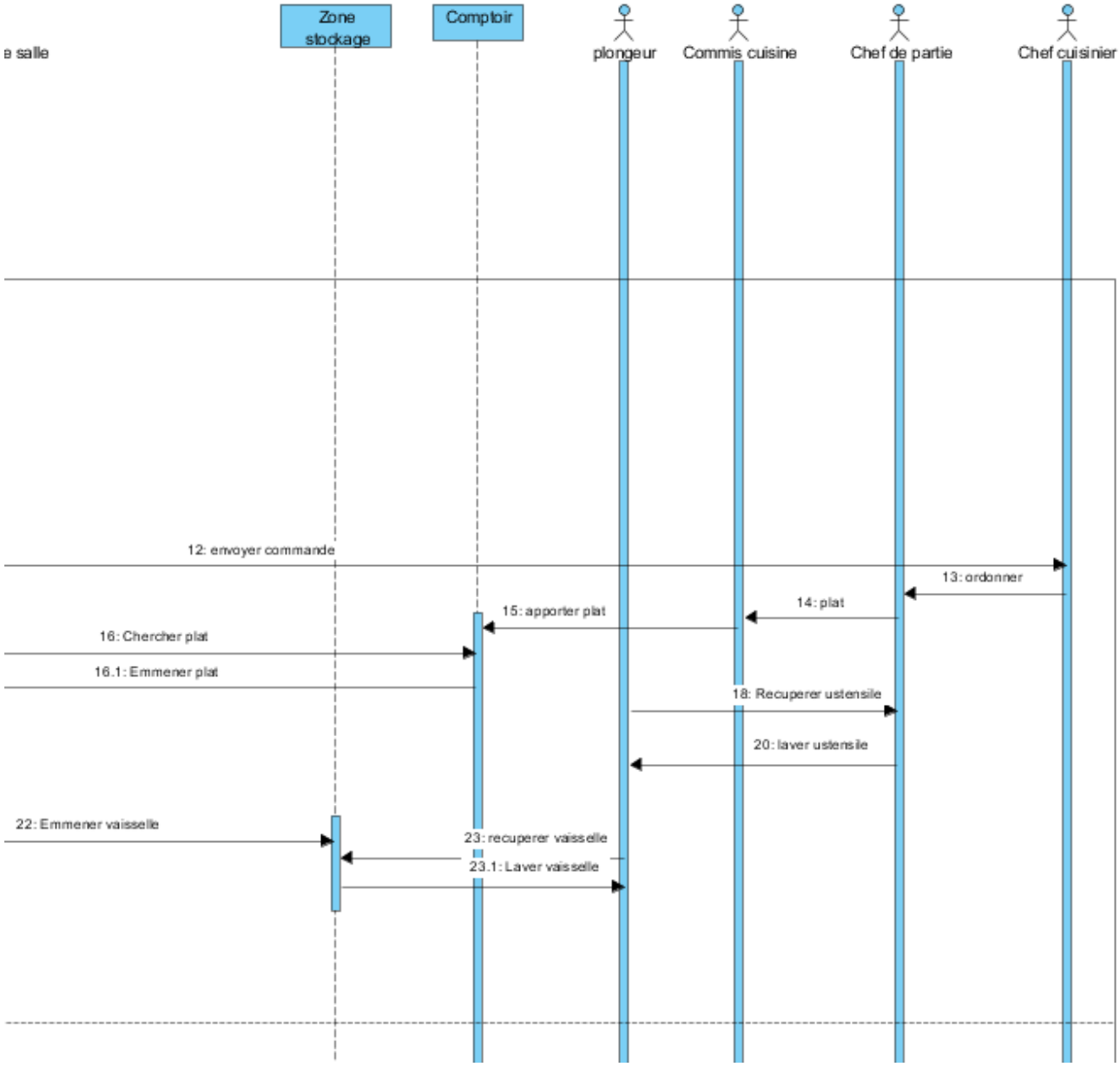


Figure 10: Partie haute droite du diagramme de séquences

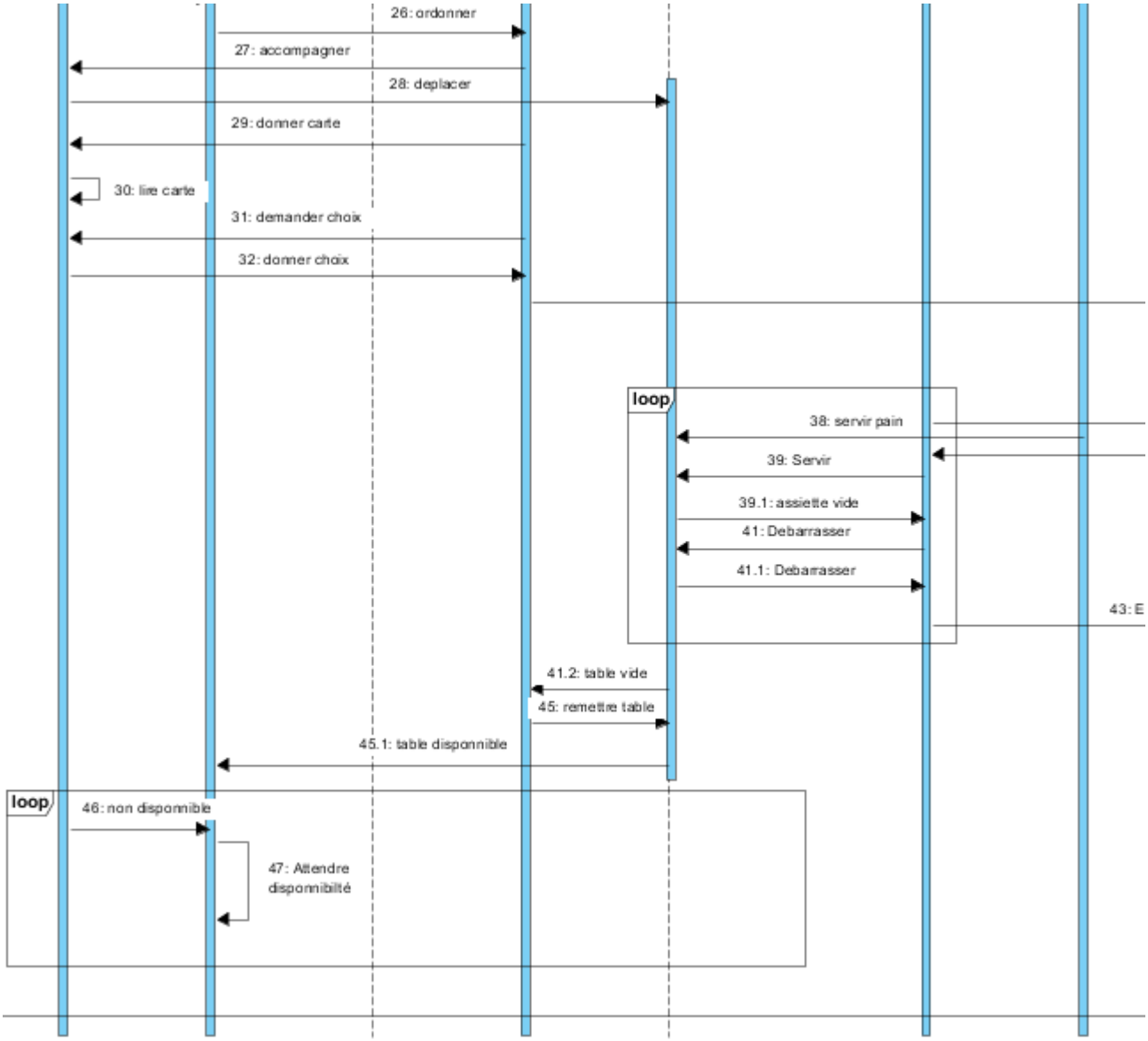


Figure 11: Partie basse gauche du diagramme de séquences

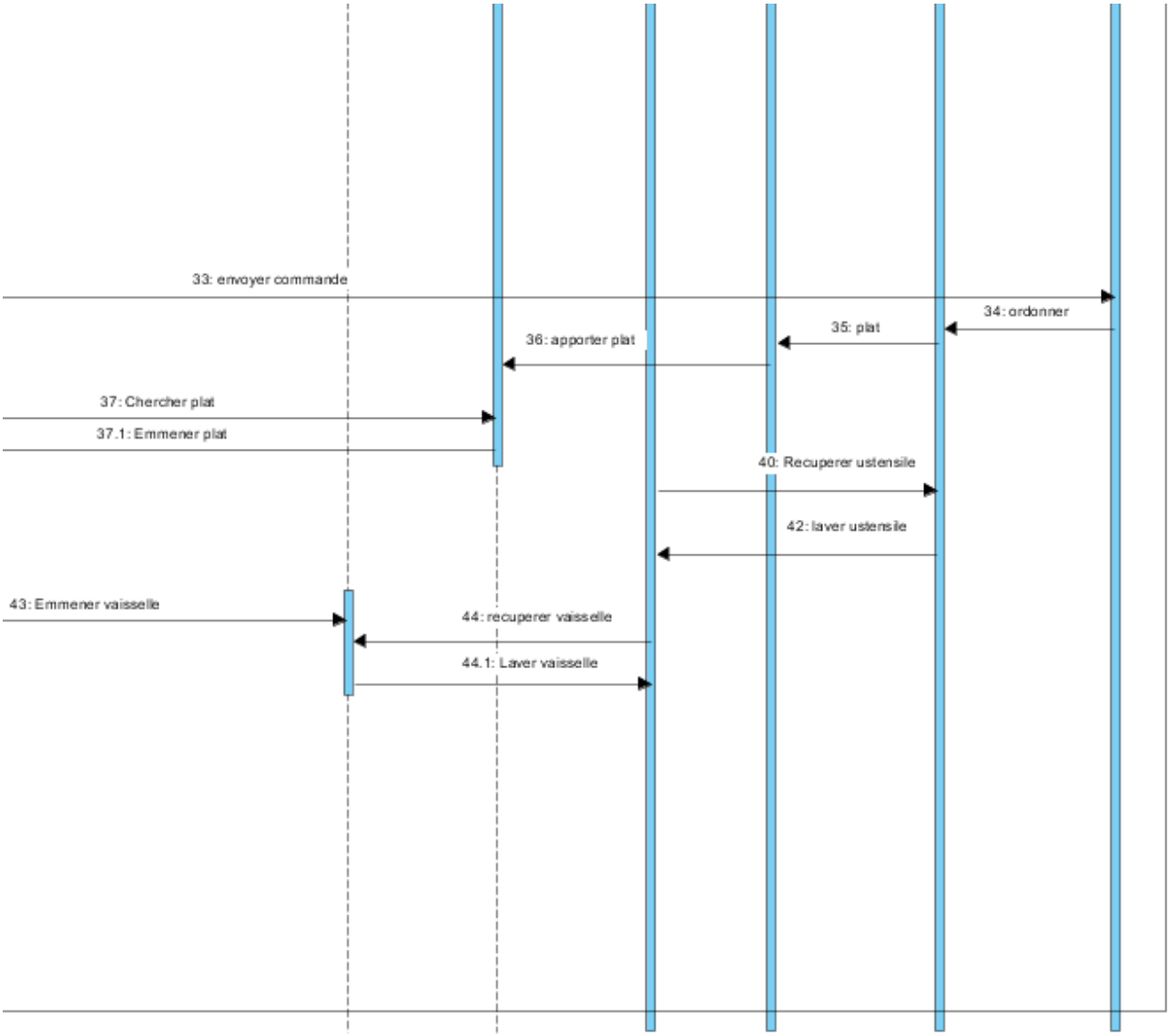


Figure 12: Partie basse droite du diagramme de séquences

c) Diagramme de composants

Le diagramme de composants décrit le système modélisé sous forme de composants réutilisables et met en évidence leurs relations de dépendance

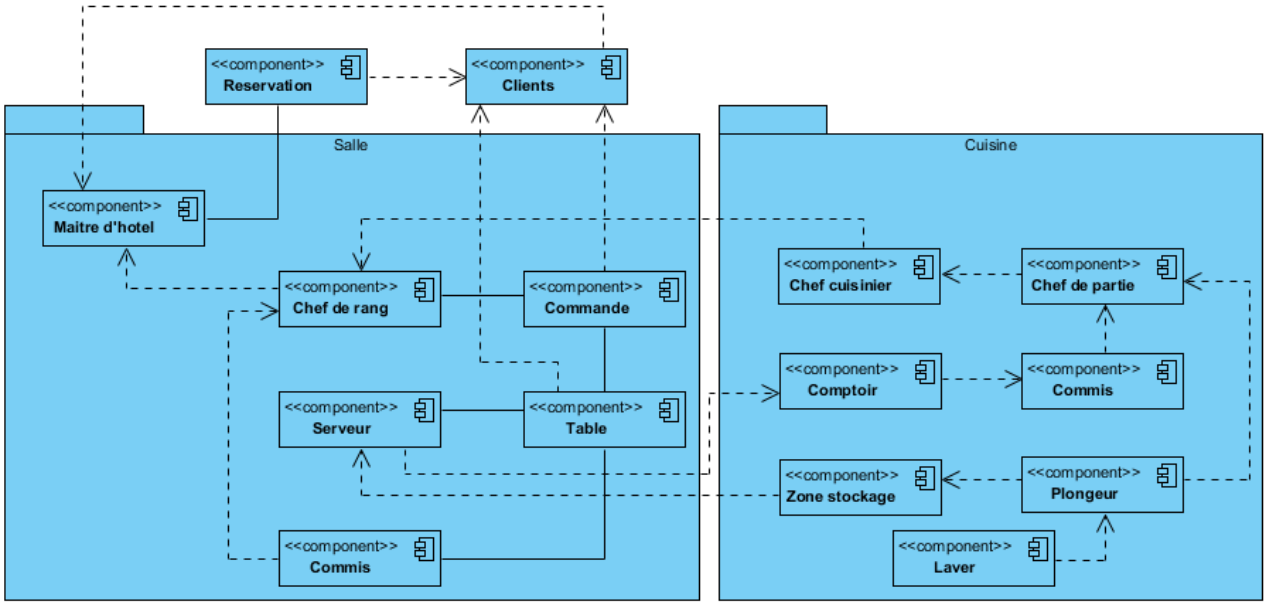


Figure 13: Diagramme de Composants

d) Diagramme de classe

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet. Il est le seul absolument obligatoire lors d'une telle modélisation. Il montre la structure interne du programme en représentant les classes et leurs relations en incluant leurs propriétés et leurs méthodes.

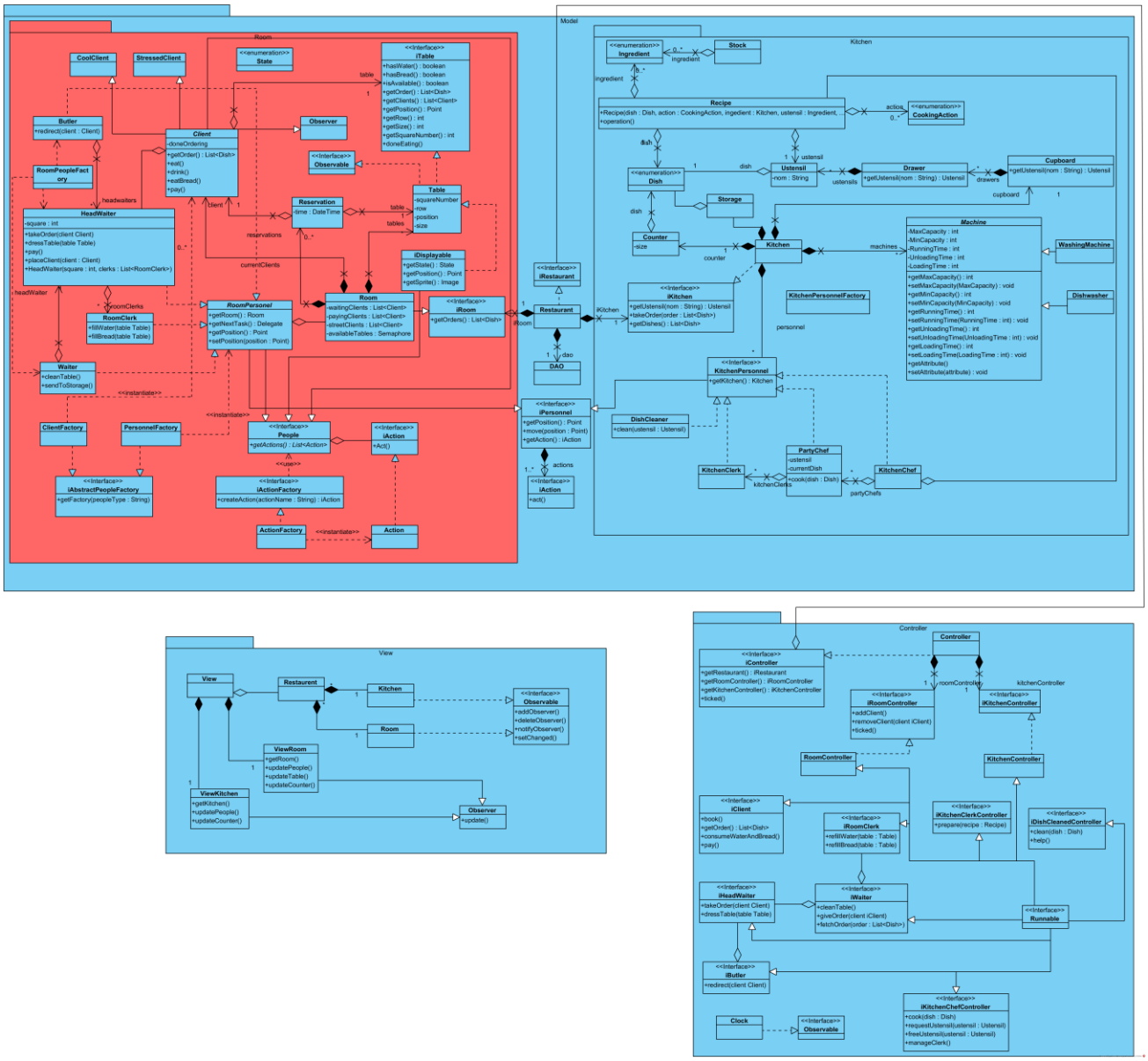


Figure 14: Vue d'ensemble du diagramme de classe

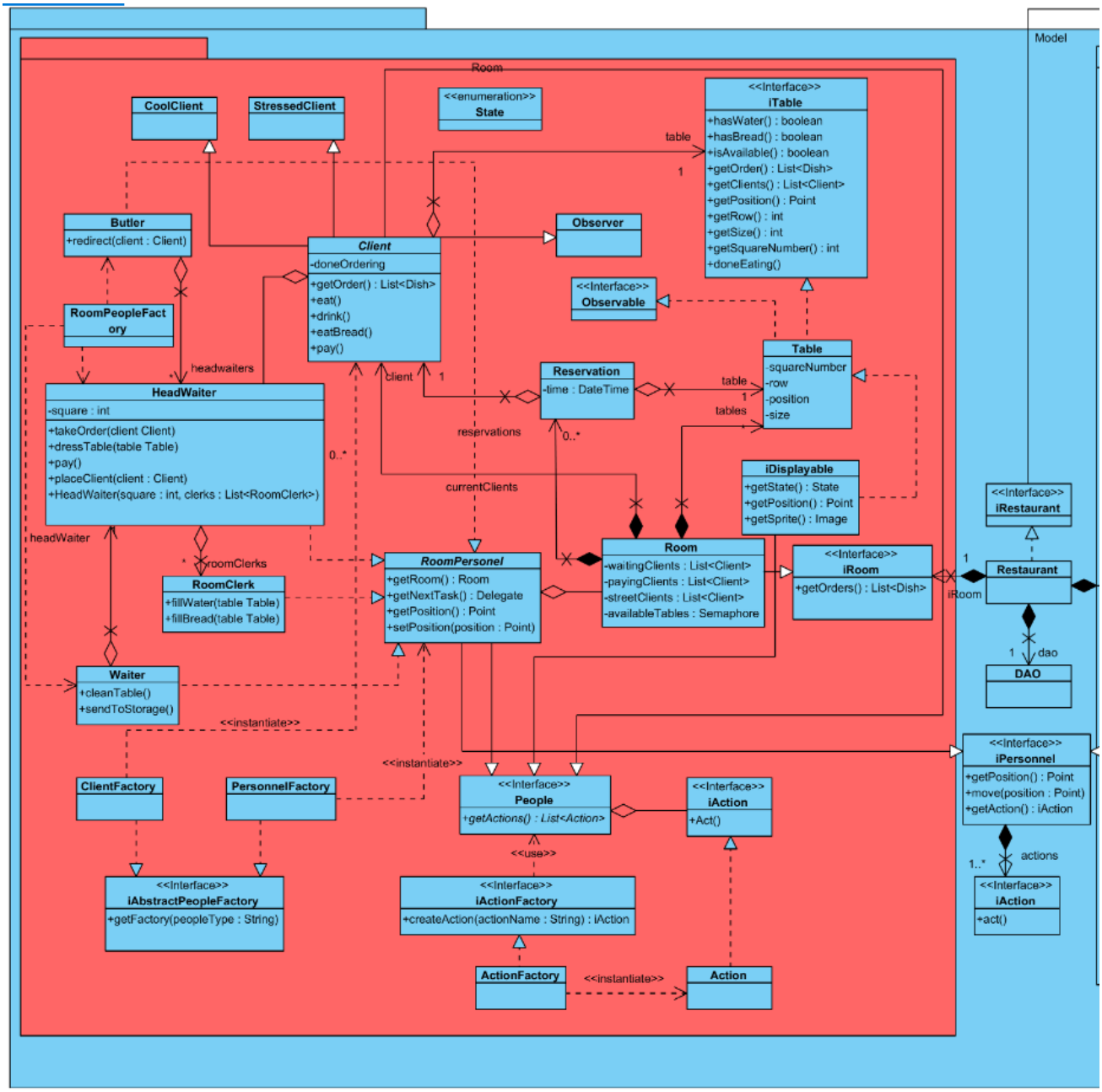


Figure 15: Partie de la salle du diagramme de classe

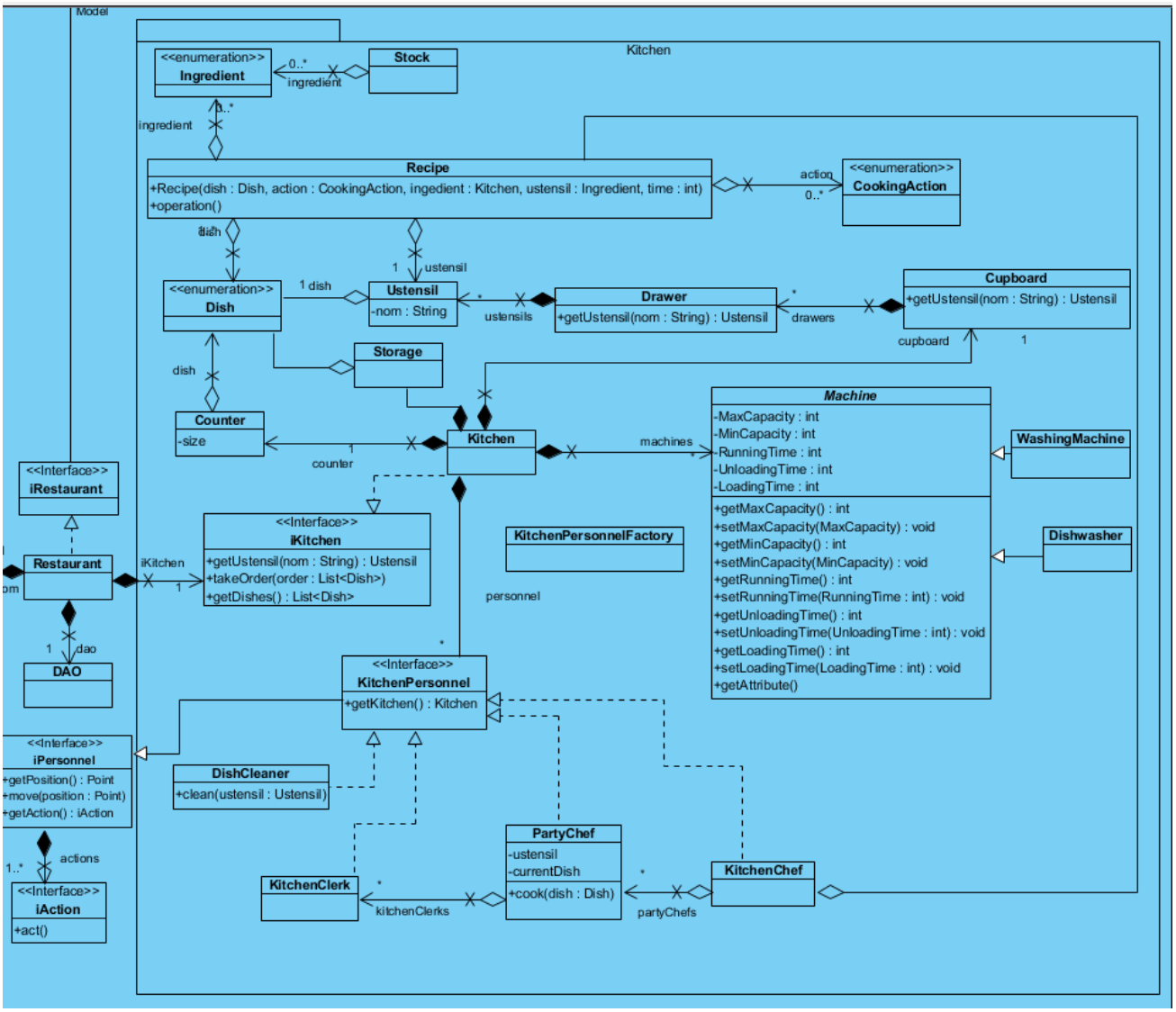


Figure 16: Partie cuisine du diagramme de classe

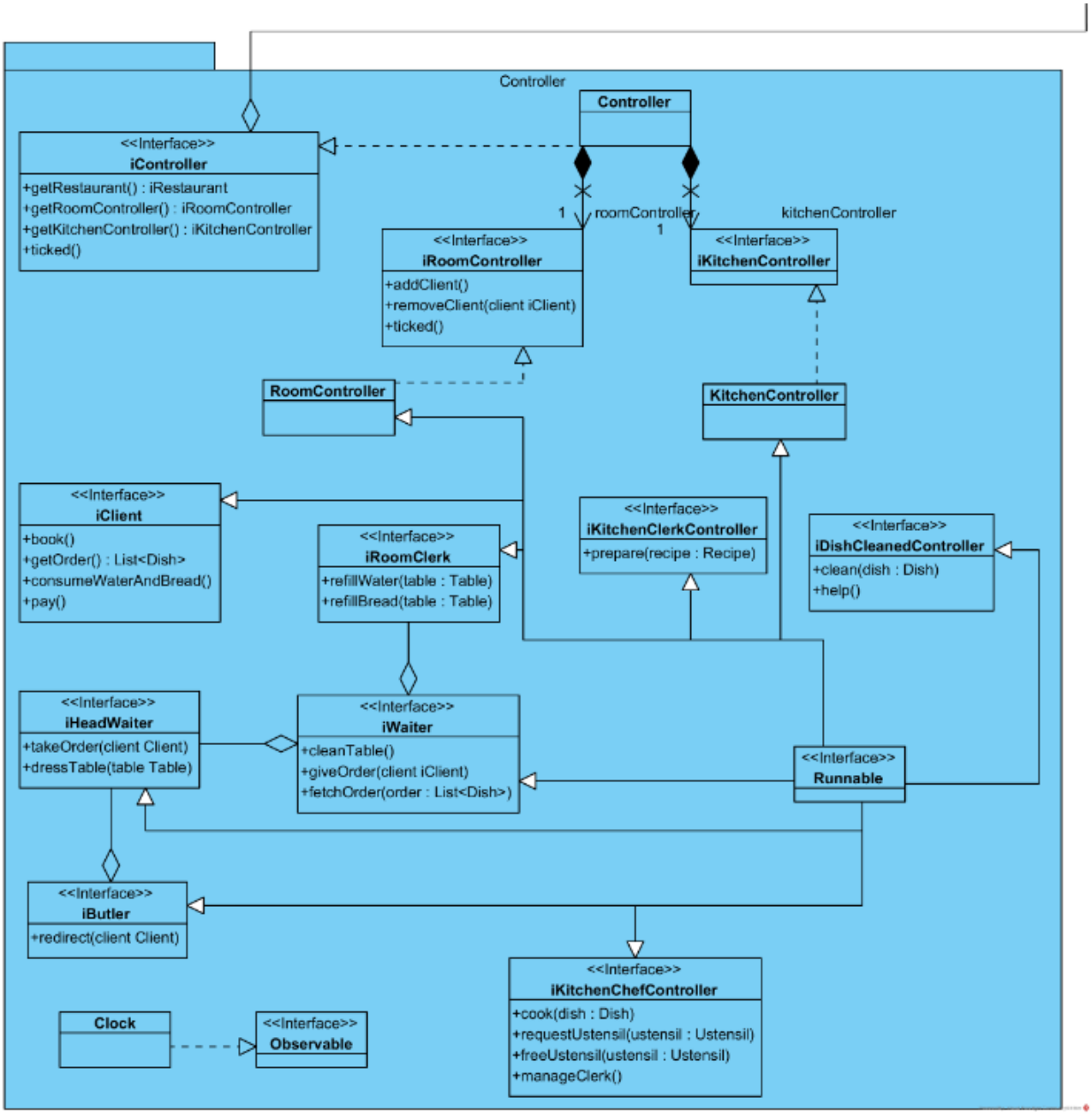


Figure 17: Partie contrôleur du diagramme de classe

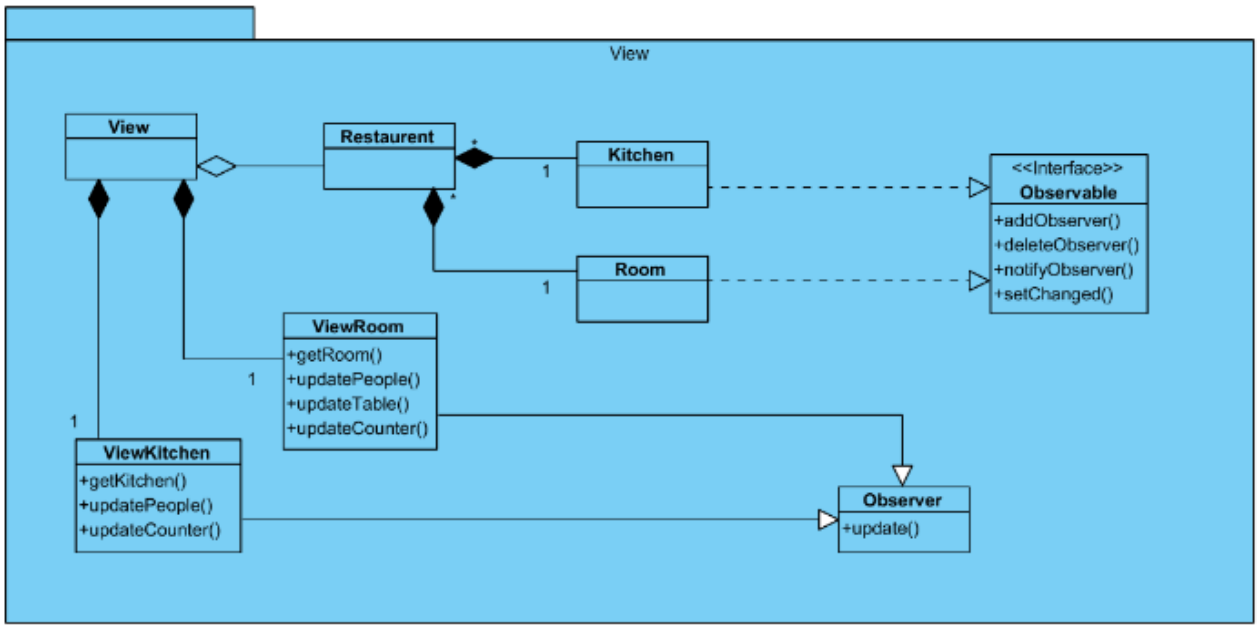


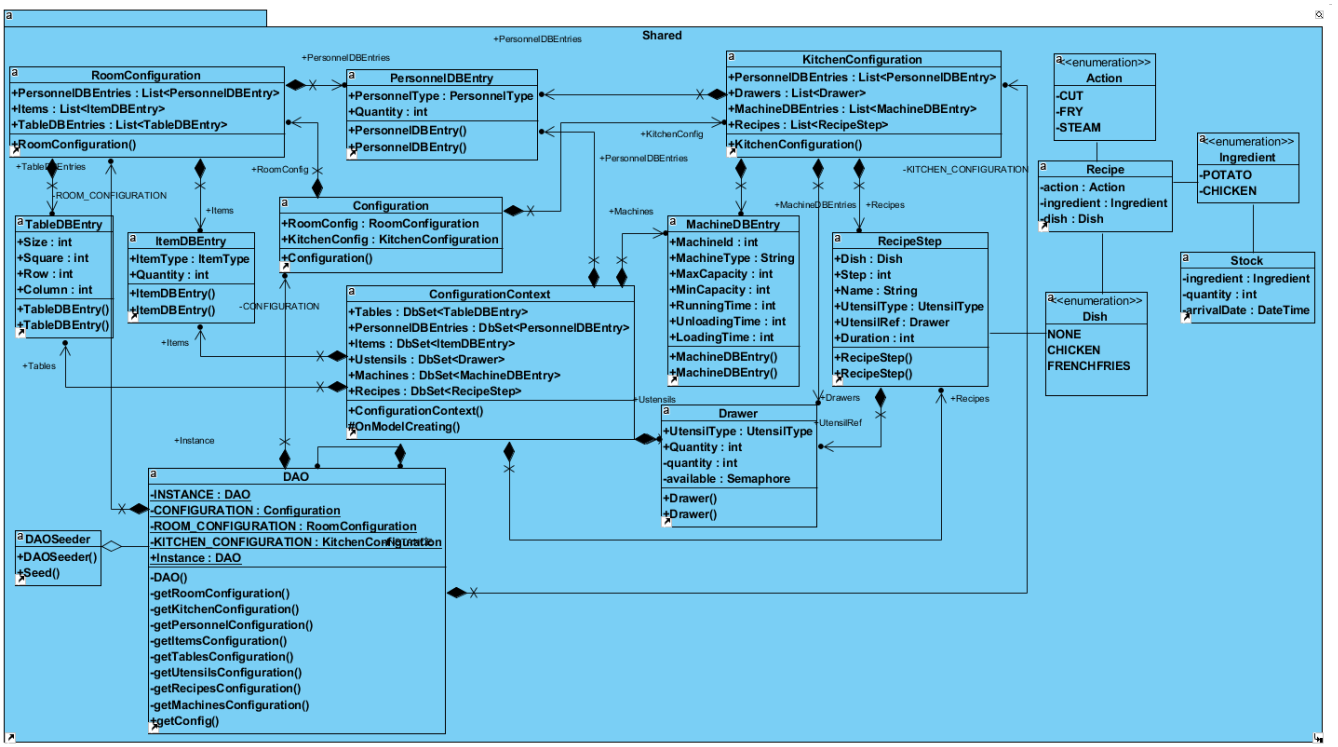
Figure 18: Partie vue du diagramme de classe

2) Base de données :

1) MCD

Le modèle conceptuel des données (**MCD**) a pour but d'écrire de façon formelle les données qui seront utilisées par le système d'information. Il s'agit donc d'une représentation des données, facilement compréhensible, permettant de décrire le système d'information à l'aide d'entités.

Dans le cadre de notre application, nous avons décidés d'utiliser un ORM (Object Relational Mapper). Il s'agit d'un outil permettant de retranscrire des objets (composante logiciel) en tables d'une base de données relationnelle (données). Cet outil nous permet de faire abstraction de la base de données relationnelle pour se focaliser sur les objets. Il est donc plus pertinent de présenter le diagramme de classe du modèle de l'application plutôt que le MCD de la base de données.



–

Figure 19: Diagramme de classe de la DAO

V. Justification d'utilisation des designs patterns.

Un Design Pattern est une solution à un problème récurrent dans la conception d'applications orientées objet. Il est considéré comme la bonne pratique à adopter pour résoudre un problème d'architecture de logiciel donné.

1) MVC

Il consiste à découper l'application en trois parties :

- **Le modèle** qui contient toutes les données, états ainsi que la logique applicative. Il n'a pas conscience de la vue ni du contrôleur, même s'il fournit une interface pour accéder à son état et le manipuler et peut informer les observateurs des changements d'état.
- **La vue** qui fournit une présentation graphique du modèle. En général, la vue reçoit directement du modèle l'état et les données qu'elle doit afficher.
- **Le contrôleur** qui accepte les entrées des utilisateurs et détermine ce qu'elles signifient pour le modèle.

Pour ce projet nous avons utilisé le pattern MVC, car c'est une architecture qui sépare les différentes parties de l'application en fonction de leurs fonctions, elle permet donc de faciliter la maintenance et l'amélioration de l'application par différents acteurs.

2) Strategy

Le DP Strategy consiste à attribuer à chaque classe une interface qui va contenir les fonctions et propriétés qui seront utilisées par d'autres objets mais qui n'auront pas à connaitre le type de l'objet, uniquement son interface.

Dans le cadre de notre projet, cela nous permet de séparer les différentes parties du MVC et de regrouper les interfaces dans un projet commun, connu de tous. Ainsi, même si nous changeons le fonctionnement de l'une des parties, le "contrat" qu'est l'interface, nous permet de nous assurer que les autres parties pourrons continuer de fonctionner sans aucun souci.

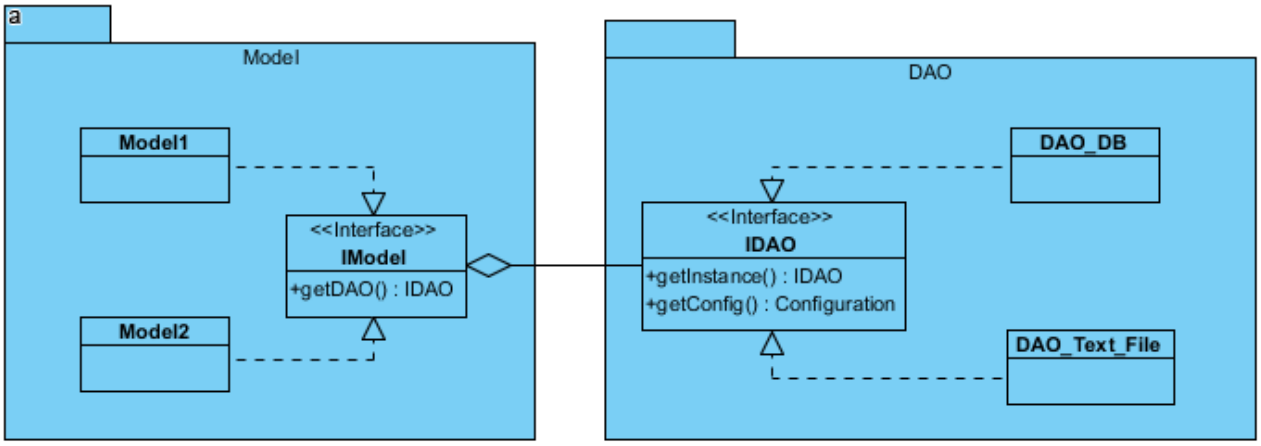


Figure 20: Utilisation du DP Strategy

Dans cet exemple simplifié on peut remplacer à tout moment le modèle par Model1 ou Model2 ou DAO_DB par DAO_Text_File sans affecter le bon fonctionnement du programme.

3) Observer

Le DP Observer permet de générer des signaux à transmettre d'un observable vers un observateur. Il sert donc à effectuer une écoute passive où l'on notifie l'observateur lors d'un changement plutôt qu'une écoute active demandant à intervalle régulier si un changement a eu lieu.

Dans le cadre de notre projet, cela nous permet de transmettre des informations à plusieurs objets, sans que l'objet observable n'ai à connaitre tous ses observateurs.

Exemple d'utilisation : dans notre programme le client va observer la table qui lui est attribuée et celle-ci va le prévenir lorsque des plats seront servis. Ce DP permet également de prévenir le commis chargé de remettre du pain sur la table lorsque celle-ci viendra à manquer

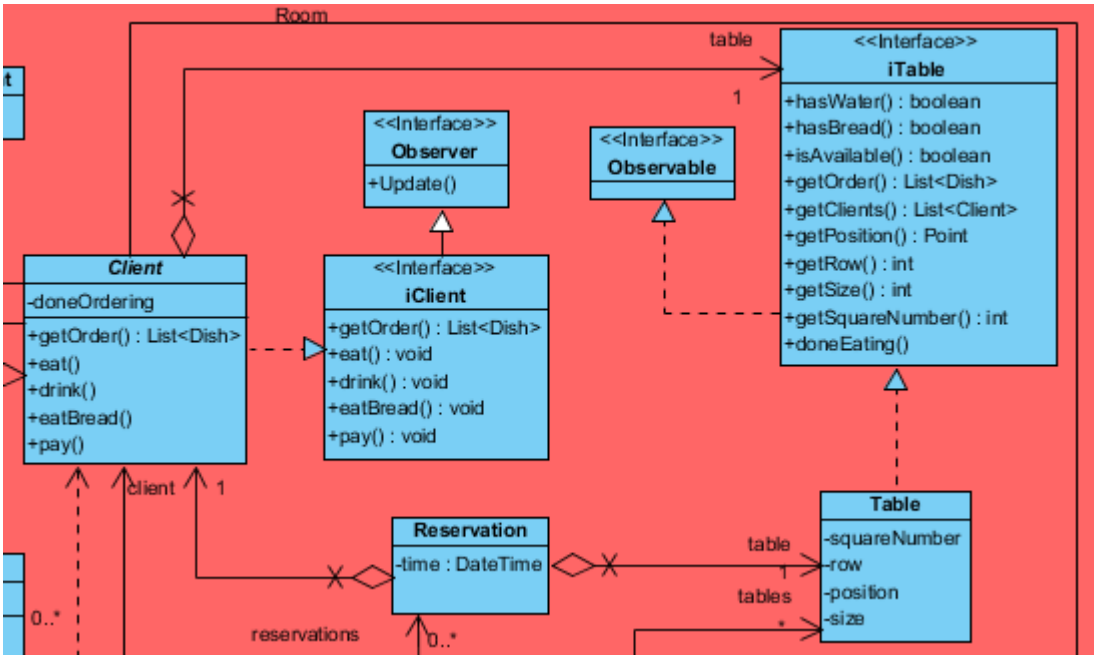


Figure 21: Utilisation du DP Observer

4) Singleton

Un singleton permet de s'assurer qu'une classe donnée ne possède qu'une seule instance. On s'en sert principalement pour s'assurer qu'un objet effectuant une connexion, et donc ladite connexion, est unique. Dans notre cas il est utilisé pour s'assurer que la connexion à la base de données soit unique.

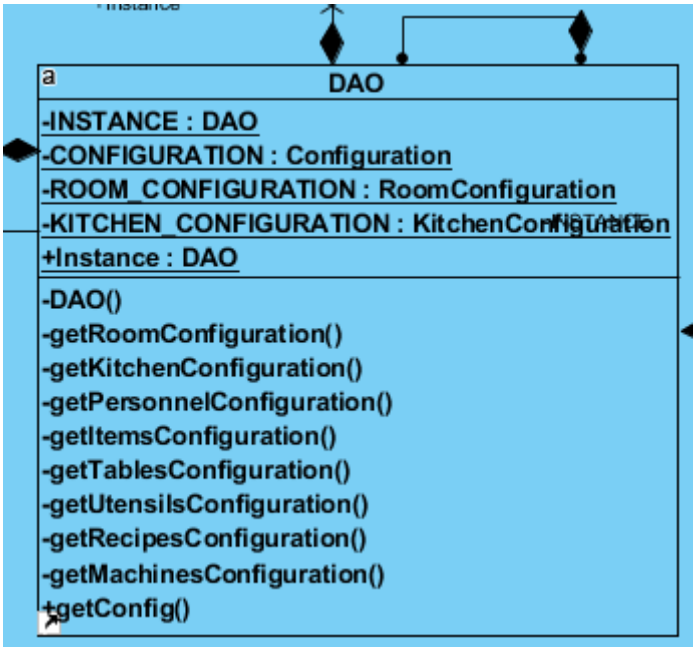


Figure 22 : Utilisation du DP Singleton

5) Factory

La factory est un pattern de création qui permet d'instancier des objets dont le type est dérivé d'une superclasse.

Nous avons utilisé ce design pattern pour plusieurs composants de notre application :

- Pour créer notre personnel (clients, serveurs, chef cuisinier, etc…) à partir d'une énumération contenant chaque type de personnel
- Pour créer nos ustensiles (casseroles, couteaux, …) à partir d'une énumération également
- Pour créer les actions des différents acteurs

Exemple d'utilisation :

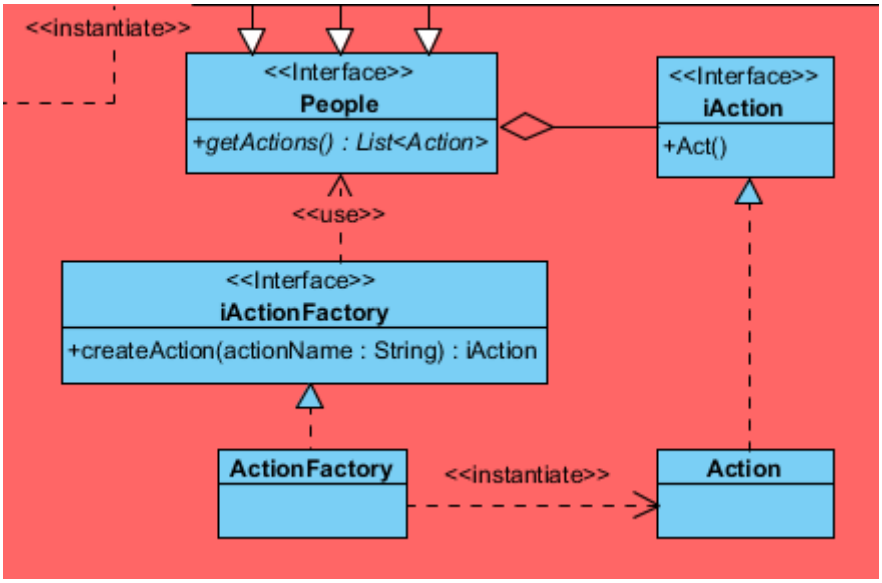


Figure 23: Utilisation du DP Factory