PROJET PROGRAMMATION SYSTEME

Rapport de conception

- Type : Application informatique
- Date de livraison souhaitée : Mercredi 12 décembre 2018

Juliette PORTE – Erwan COURTOIS – Charly PEYSSARD – Hadrien DEVEVEY

eXia.CESI A3 2018-2019

cesi ÉCOLE D'INGÉNIEURS

TABLE DES MATIERES

TABLE DES FIGURES

I. Rappel du besoin

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…).

II. Equipe

Pour ce projet, notre équipe est composée de PORTE Juliette en tant que chef de projet ainsi que de Erwan COURTOIS, Charly PEYSSARD et Hadrien DEVEVEY.

Erwan COURTOIS

Charly PEYSSARD

Hadrien DEVEVEY

Juliette PORTE

Chef de projet

Figure 1 : Présentation de l'équipe

III. Méthodologie du projet

Afin d'aborder au mieux ce sujet et satisfaire en priorité en client, nous avons choisi d'orienter notre méthodologie sur une méthode agile. Voici le déroulé de notre phase de conception :

- **Lecture et analyse du sujet (groupe)**

En effet, nous avons débuté par une première lecture en groupe, puis une seconde afin d'en retirer tous les objectifs, livrables et fonctionnalités attendues. Grâce à ce brainstorming, nous avons pu comprendre le fonctionnement global (et détaillé) du système attendu ainsi que cibler les tâches à accomplir lors de cette phase de conception.

- **Dispersion (binômes)**

Une fois les tâches définies, et un schéma global de notre système réalisé, nous nous sommes répartis par binômes pour commencer la phase de conception. Cette méthodologie nous a permis d'avoir, à chaque fois, plusieurs visions sur le même digramme, lorsqu'un sujet peut être compris de plus d'une dizaine de façon différente.

- **Briefing régulier (groupe)**

Une réunion de validation de groupe a été réalisé chaque soir, permettant à tous de visualiser le travail de chacun ainsi que de le compléter au besoin. Ces réunions nous ont aussi permis de mesurer l'avancement du projet au fur et à mesure.

- **Finalisation (groupe)**

Pour la finalisation de cette phase de conception, nous nous sommes de nouveau réunis en groupe afin de corriger les éventuelles erreurs ainsi que de relire le rapport de conception.

Pour respecter au mieux les principes de la méthode agile, nous avons réalisé un versionning de chacun de nos diagrammes, afin de garder le plus de sauvegardes possibles de ces derniers. Une conversation sur le support « Microsoft Teams » nous a permis d'entretenir un contact en permanence.

Afin d'encadrer au mieux notre projet ainsi que de respecter les dates limites de rendu du dossier, nous avons réalisé un planning initial. Il nous a aussi permis de visualiser les personnes assignées à chaque tâche ainsi que le chemin critique. Pour des soucis de taille, retrouver ce planning en annexe ; « Annexe 1 : Planning initial du projet ».

Durée du projet : Lundi 03 décembre au Mardi 12 décembre 2018 (9 jours)

Durée de la période de conception : Lundi 03 décembre au Mercredi 05 décembre 2018 (3 jours)

IV. Spécificités techniques et contraintes du projet

Voici le cahier des charges fonctionnel de notre projet :

- Diagrammes de cas d'utilisation et d'activité de chaque poste de travail.
- Simulation graphique du fonctionnement du restaurant en temps réel. Pour les besoins de la démonstration (et des tests), nous vous demandons d'avoir un mode simulation en temps accéléré à l'échelle (1' = 1'').
- L'application doit permettre de visualiser l'état de chaque personne (salariés ou clients) et de chaque objet modélisé ainsi que les situations limites, et mettre des alertes sur le manque de telle ou telle ressource (presque plus d'assiettes ou plus de verres du tout, …). Pour cela vous devez avoir une fenêtre de contrôle (vous pouvez vous inspirer des pages de contrôle des superviseurs comme Nagios ou Centreon avec lesquels vous avez travaillé dans l'UE précédente).
- Prévoir le mode « **PAUSE** » de votre application pour pouvoir stopper tous les processus et analyser la situation en cours.
- Modélisation/implémentation de plusieurs catégories de clients avec des comportements différents.
- Modélisation/implémentation de tous les postes décrits dans la partie « Description de postes ».
- Modélisation/implémentation de tout le matériel décrit (et nécessaire pour la réalisation des recettes)
- Préconisations sur le dimensionnement du restaurant et des ressources : les chiffres donnés sont corrects, surdimensionnés ou sous-dimensionnés. Dans ce cas-là, vous devez donner les postes à pourvoir ou à supprimer et le matériel à garder, à supprimer ou à racheter. Également, on attend de vous des préconisations sur le processus global de la gestion du restaurant.
- 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.

Ce projet sera donc réalisé en C# / .NET et utilisera une base de données SQL Server pour stocker nos informations.

Lors de la rédaction du sujet et du cahier des charges, nous avons identifié un certain nombre de contraintes :

- Utilisation d'au moins un IPC par type
- Utilisation de threads et pools
- Utilisation des sockets (échanges salle-cuisine)
- Utilisation de Design Pattern (au moins 5 au choix en plus du MVC qui est obligatoire)
- Utilisation de GIT pour la gestion de versions décentralisé du code de notre application

- Utilisation de test en TDD

V. Analyse et conception

Le cycle de vie du logiciel que nous allons développer comprend à minima les activités suivantes :

- **Définition des objectifs**
- **Analyse des besoins**
- **Conception**, consistant à définir précisément chaque sous-ensemble du logiciel.
- **Codage** (Implémentation ou programmation), soit la traduction dans un langage de programmation des fonctionnalités définies lors de phases de conception.
- **Tests unitaires**, permettant de vérifier individuellement que chaque sous-ensemble du logiciel est implémenté conformément aux spécifications.
- **Intégration**, dont l'objectif est de s'assurer de l'interfaçage des différents éléments (modules) du logiciel. Elle fait l'objet de tests d'intégration consignés dans un document.
- **Qualification** (ou recette), c'est-à-dire la vérification de la conformité du logiciel aux spécifications initiales.
- **Documentation**, visant à produire les informations nécessaires pour l'utilisation du logiciel et pour des développements ultérieurs.
- **Mise en production**
- **Maintenance**, comprenant toutes les actions correctives (maintenance corrective) et évolutives (maintenance évolutive) sur le logiciel

Dans ce rapport, nous allons nous pencher sur la conception de notre projet.

Par soucis de normalisation, et pour s'adapter à tous les langages, nos diagrammes comportant du code seront en anglais, de même pour ceux en rapport avec la base de données.

a) Diagramme de cas d'utilisation

En UML, on établit des Diagrammes de Cas d'Utilisation pour savoir à quoi va servir notre logiciel. Les principaux éléments sont :

- Les acteurs (les bonhommes)
- Les cas d'utilisation (les ellipses)
- Le système qui englobe tous les cas d'utilisation

Nous avons réalisé deux diagrammes de cas d'utilisation : un pour la cuisine et un autre pour la salle. A chaque acteur du restaurant sont relié les tâches qu'il peut réaliser.

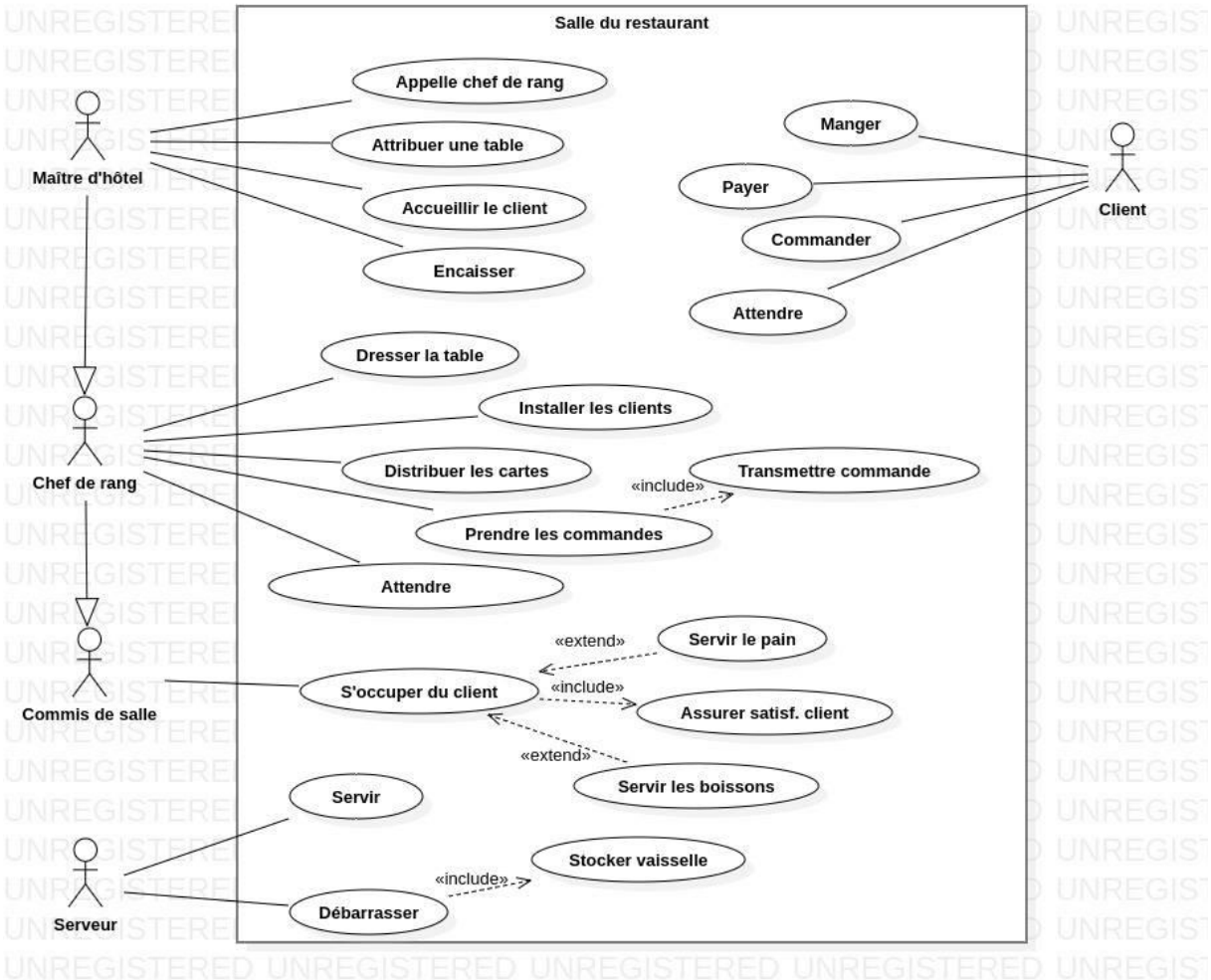


Figure 2: Diagramme de cas d'utilisation de la salle du restaurant

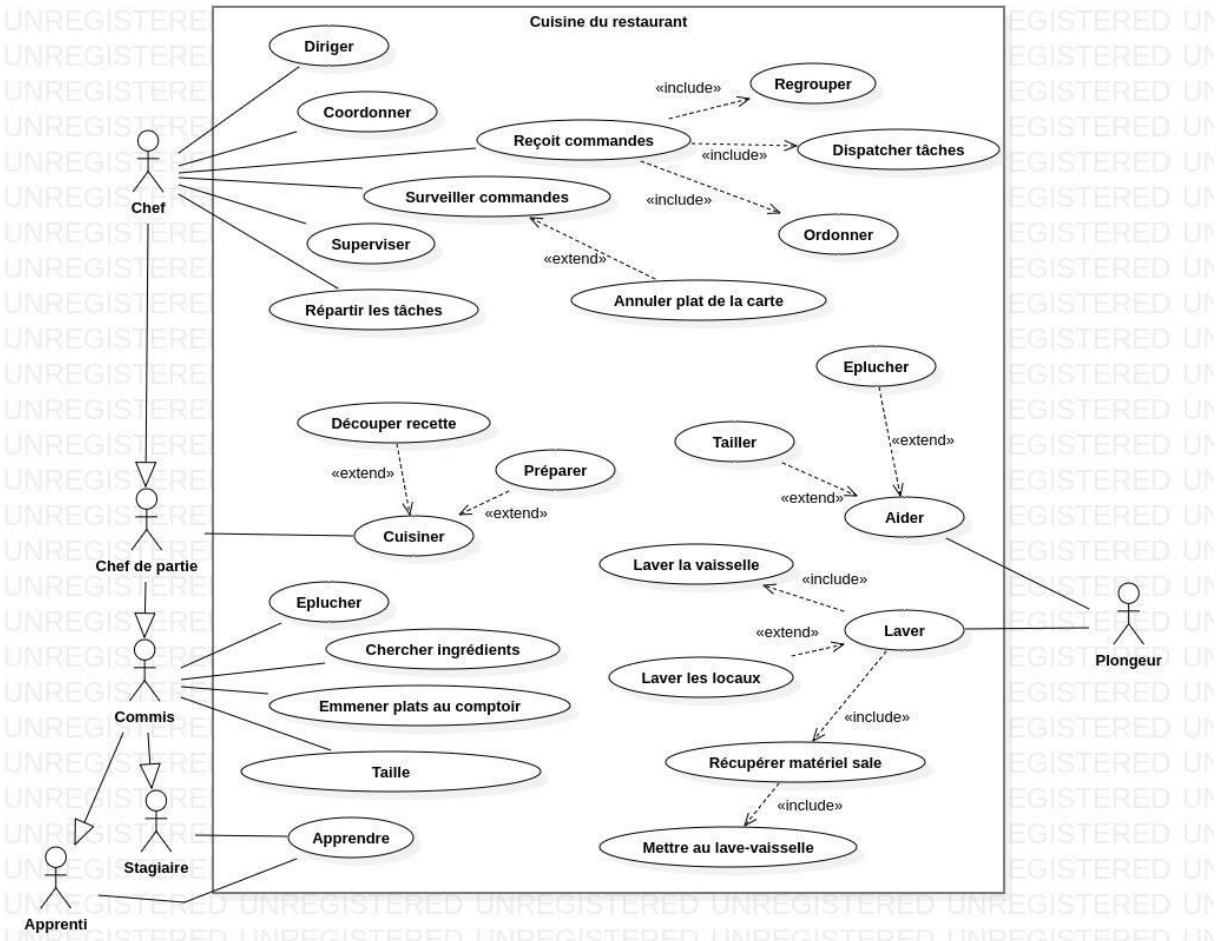


Figure 3 : Diagramme de cas d'utilisation de la cuisine du restaurant

b) Diagramme d'activité

Afin de définir au mieux les différents comportements de chacun des employés du restaurant, nous vous avons réalisé un diagramme pour chacun.

1. Maître d'hôtel

Le maitre d'hôtel, qui est immobile à l'accueil, doit d'abord attendre la sollicitation d'un client.

Si le client vient d'arriver, le maitre d'hôtel lui demande s'il fait partie d'un groupe et de combien de personnes il est composé (le cas échéant). Le maitre d'hôtel sélectionne ensuite une table disponible avant de notifier le chef de rang de la table concernée. Si le client sollicite le maitre d'hôtel pour payer, ce dernier va encaisser le payement avant de notifier le chef de rang de la table libérée. Une fois la tâche réalisée, le maitre d'hôtel vérifie si son service est fini et arrête ses actions si plus aucun client n'est présent, ou recommence le processus en attendant la sollicitation d'un client.

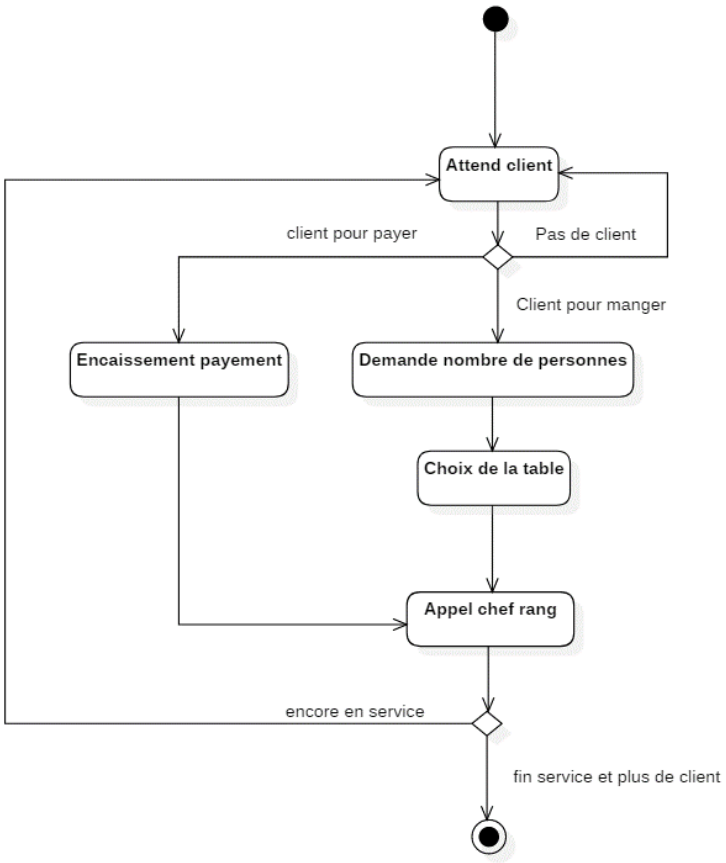


Figure 4 : Diagramme d'activité du maître d'hôtel

2. Chef de rang

Chaque chef de rang va attendre des requêtes pendant qu'il est stationné à un endroit précis de la salle.

Si il un client est prêt à commander, alors le chef de rang rejoint sa table et prend sa commande, avant de transmettre la commande au chef de cuisine. S'il est sollicité directement par le maitre d'hôtel, il existe trois cas de figure :

- Un client part, le chef de rang va alors dresser la table pour le prochain client.
- Un changement d'affectation de carré.
- Un ou des clients doivent être placés, le chef de rang va alors les amener à leur table et leur distribuer des cartes.

Une fois la tâche réalisée, le chef de rang vérifie si son service est fini et arrête ses actions si plus aucun client n'est présent, ou ne recommence le processus en attendant la sollicitation d'un client ou du maître d'hôtel.

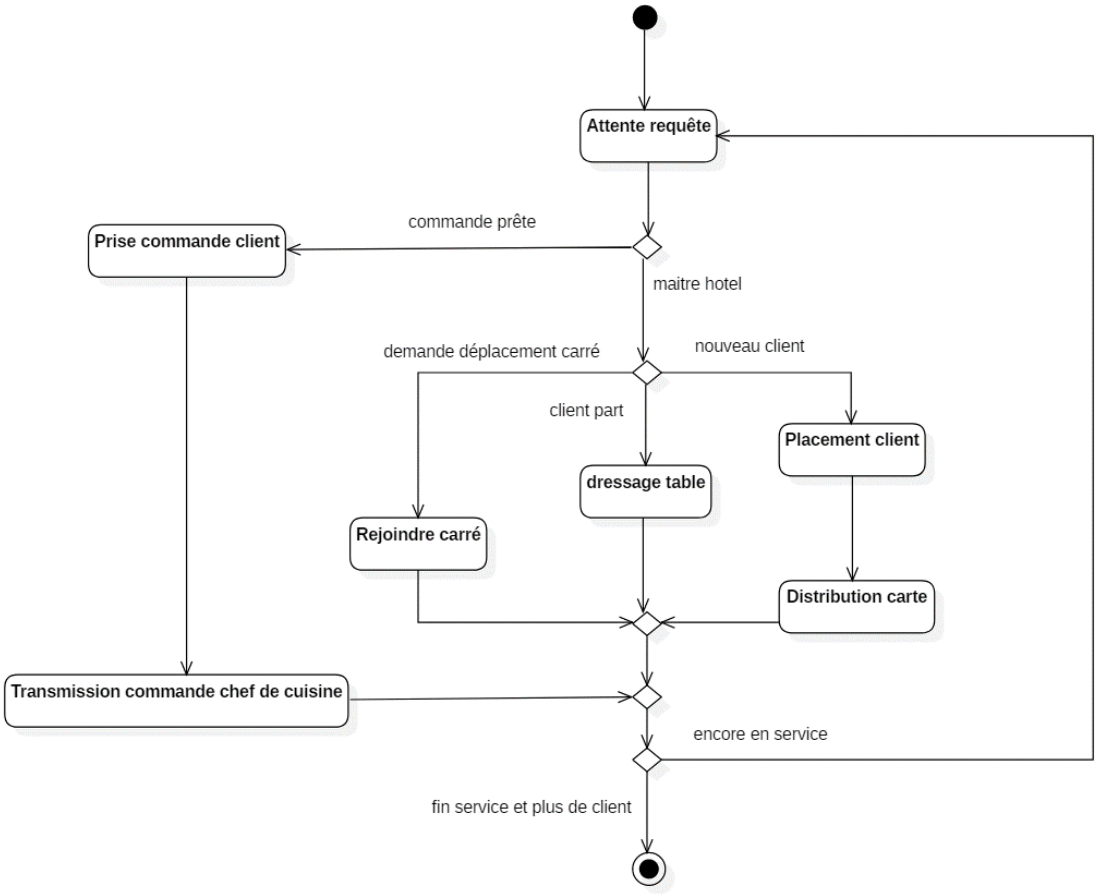


Figure 5 : Diagramme d'activité du(/des) chef(s) de rang

3. Chef de cuisine

Le chef de cuisine se charge d'organiser la cuisine, et doit donc attendre une commande avant d'agir.

Une fois qu'il reçoit une commande, le chef de cuisine vérifie si tous les ingrédients sont disponibles. S'il en manque, le chef de cuisine annule les plats de la carte qui contiennent cet ingrédient. Si tous les ingrédients sont en quantité suffisante, le chef de cuisine regroupe les commandes avant de les répartir entre les différents cuisiniers.

Une fois la tâche réalisée, le chef de cuisine vérifie si son service est fini et arrête ses actions si plus aucun client n'est présent, ou recommence le processus en en attendant une nouvelle commande.

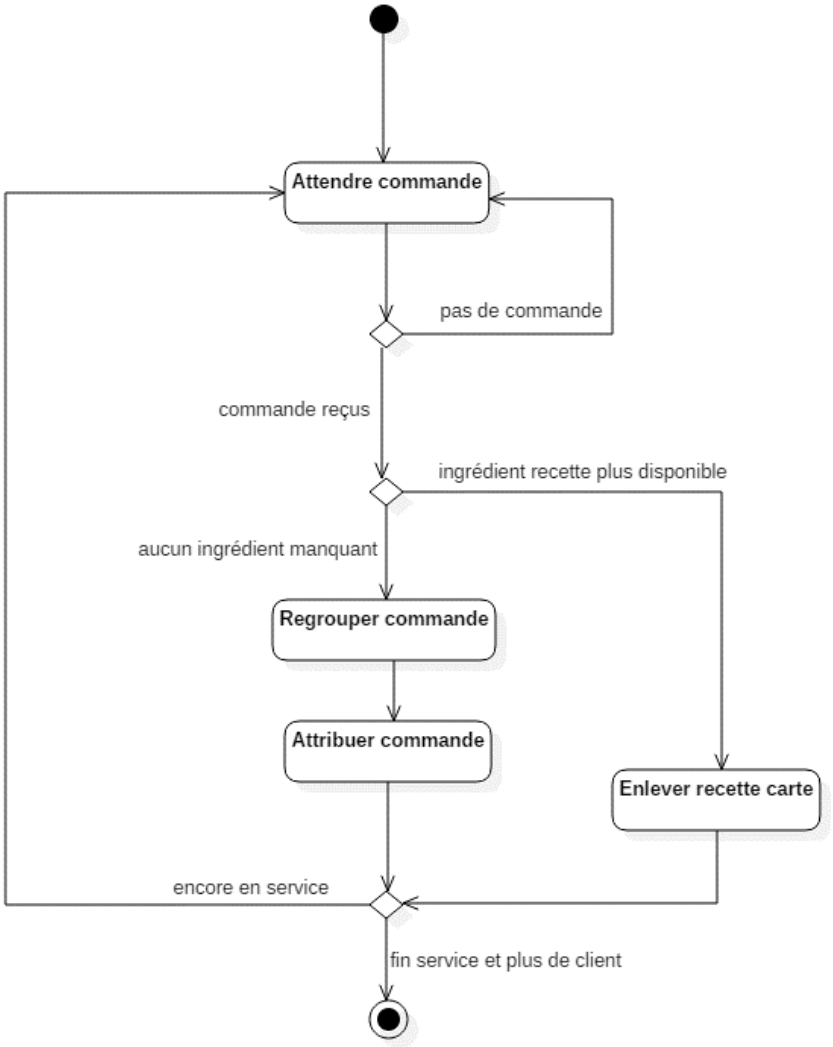


Figure 6 : Diagramme d'activité du chef de cuisine

4. Commis de salle

Les commis sont sous les ordres des chefs de partis, et attendent donc leurs ordres avant d'agir.

Un commis peut être demandé à chercher un ingrédient dans le stock. Si jamais un ingrédient vient à manquer, le commis prévient le chef de cuisine. Si un plat est prêt, le commis doit l'amener au comptoir afin que les serveurs puissent le prendre. S'il est demandé au commis d'éplucher ou de tailler des légumes, il va tout d'abord vérifier si un outil est disponible, et l'attendre si ce n'est pas le cas. Une fois l'outil disponible, il épluche ou taille le légume, puis libère l'outil et le place à côté de l'évier.

Une fois sa tâche réalisée, le commis vérifie si son service est fini et arrête ses actions si plus aucun client n'est présent, ou recommence le processus en en attendant une nouvelle requête de la part d'un cuisinier.

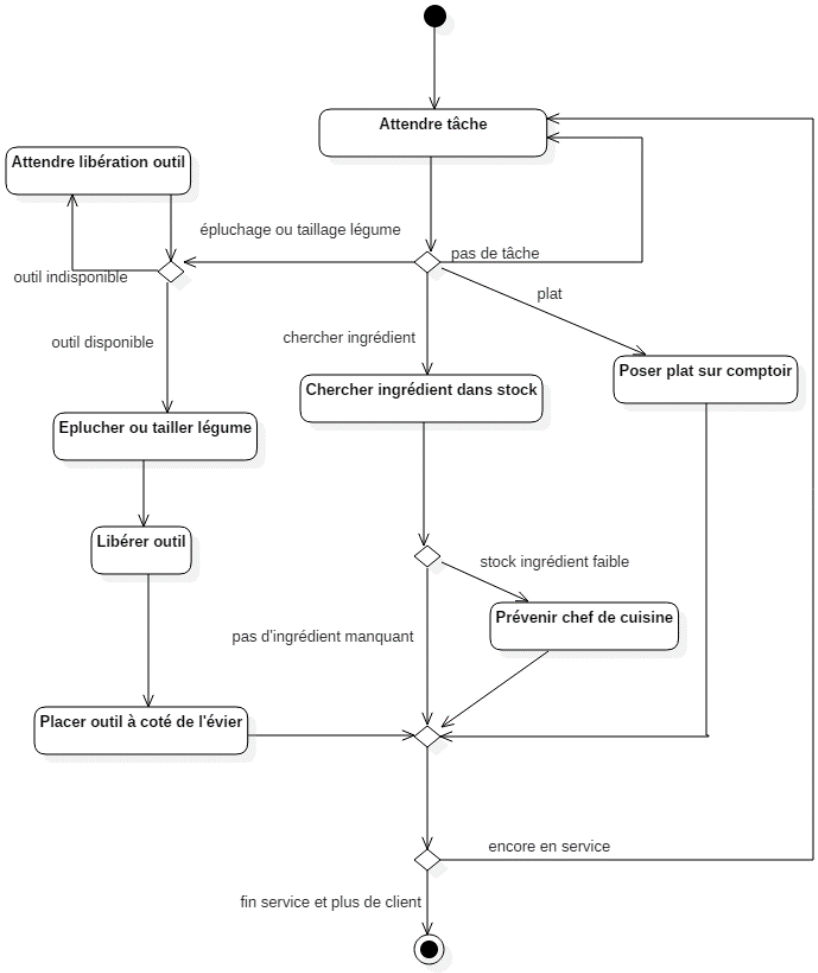


Figure 7 : Diagramme d'activité du(/des) commis de salle

5. Serveur

Chaque serveur doit attendre une requête au comptoir avant d'agir.

Si un client a fini une commande, alors le serveur doit mettre le pain et l'eau sur la table du client concerné.

Si tous les plats d'une table sont prêts, le serveur doit prendre les plats au comptoir puis servir la table concernée.

Si un client a fini son plat, le serveur doit alors débarrasser ledit plat et amener les couverts sales dans la zone de stockage sale. De plus, si la table a fini le repas, le serveur doit la débarrasser et amener la nappe et les serviettes dans la lingerie, puis lancer la machine à laver s'il y a plus de 10 nappes et serviettes dans cette zone de stockage.

Une fois la tâche réalisée, le serveur vérifie si son service est fini et arrête ses actions si plus aucun client n'est présent, ou recommence le processus en en attendant une nouvelle sollicitation.

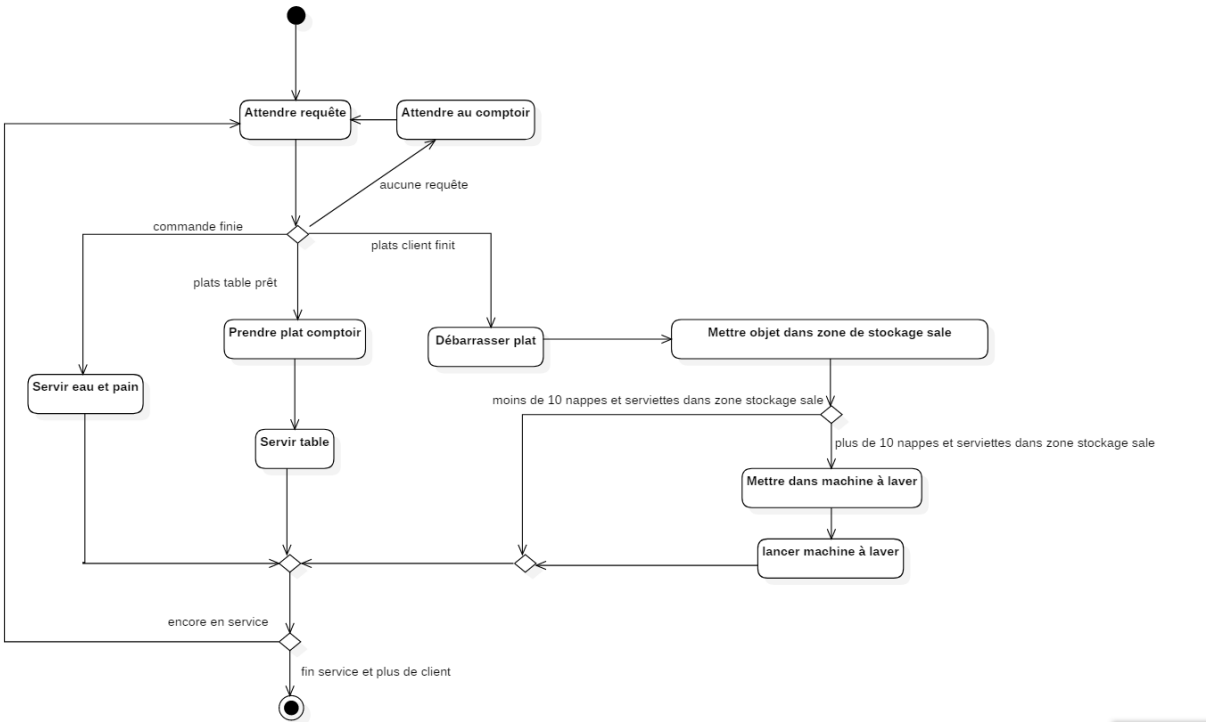


Figure 8 : Diagramme d'activité du(/des) serveur(s)

6. Chef de partie

Les cuisiniers sont sous les ordres du chef de cuisine, et attendent donc une requête de sa part avant d'agir.

Une fois l'ordre reçu, le cuisinier regarde si l'outil pour sa prochaine étape est disponible. Si ce n'est pas le cas, il attend la libération de l'outil. Si l'outil est disponible, il réalise l'étape de sa recette, et libère l'outil avant de le poser à côté de l'évier (pour nettoyage) s'il n'en a plus besoin. Le cuisinier vérifie ensuite si sa recette est finie. Si c'est le cas, il libère tous les outils et les places à côté de l'évier.

Une fois la tâche réalisée, le cuisinier vérifie si son service est fini et arrête ses actions si plus aucun client n'est présent, ou recommence le processus en en attendant une nouvelle requête de la part du chef de cuisine.

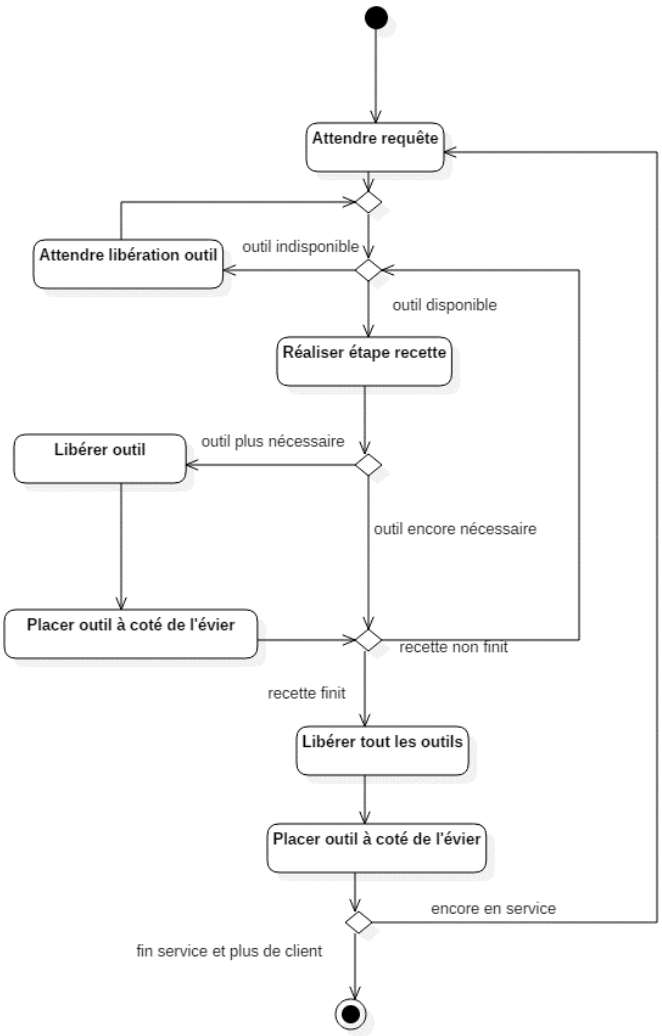


Figure 9 : Diagramme d'activité du(/des) chef(s) de partie

7. Le plongeur

Le plongeur doit nettoyer le matériel utilisé en cuisine et le matériel utilisé en salle. Il attend donc un objet à nettoyer ou une demande pour taller ou éplucher les légumes.

S'il est demandé au plongeur d'éplucher ou de tailler des légumes, il va tout d'abord vérifier si un outil est disponible, et l'attendre si ce n'est pas le cas. Une fois l'outil disponible, il épluche ou taille le légume, puis libère l'outil et le place à côté de l'évier.

S'il doit nettoyer un objet, il existe deux cas de figure :

- Si c'est un objet de cuisine, qui est donc à côté de l'évier, le plongeur le nettoie à la main avant de le rendre disponible.
- Si c'est un plat sale, le plongeur doit aller le chercher au comptoir dans le stock des plats sale. SI c'est un autre objet, le plongeur va le chercher dans le stock approprié. Il va ensuite le placer dans la machine associée (machine à laver ou lave-vaisselle) ou si la machine est en cours ou qu'elle n'a plus de place il place l'objet dans la zone de stockage d'objet sale.

Si une machine est finie, le plongeur vide la machine et range les objets, puis recharge la machine et la relance. Il finit par libérer les objets qui ont été nettoyés. Si une machine n'a pas été lancée depuis 10 minutes et qu'elle n'est pas vide, le plongeur la démarre.

Une fois sa tâche réalisée, le commis vérifie si son service est fini et arrête ses actions si plus aucun objet n'est encore sale, ou recommence le processus en en attendant une nouvelle requête de la part d'un cuisinier.

Pour des soucis de taille, retrouver ce diagramme en annexe ; « Annexe 2 : Diagramme d'activité du plongeur ».

c) Diagramme de classe

Ce digramme va nous permettre de représenter de manière statique les éléments (ainsi que leur relations) qui composent notre système. Pour des soucis de taille, veuillez le retrouver uniquement en pièce jointe de ce rapport. Pour des soucis d'optimisation, il est de bonne pratique d'implémenter dans notre code des patrons de conception (plus souvent appelé « design pattern »). Voici ceux que nous avons choisi d'utiliser :

1. Le design pattern Modèle-Vue-Contrôleur (MVC)

Observons comment nous avons organisé l'application en Modèle vue contrôleur (MVC). C'est une façon d'appliquer le principe de séparation des responsabilités, en l'occurrence celles du traitement de l'information et de sa mise en forme.

Le **modèle** manipule la donnée. Dans un site Web, le modèle est souvent le code qui permet de faire de requêtes à la base de données. La **vue**, c'est de la présentation. C'est la manière avec laquelle on veut que la donnée soit présentée à l'utilisateur. Ça peut être le code qui produit du HTML ou du CSV, ou fait configurer des boutons dans une interface graphique. Le **contrôleur**, c'est tout le reste. Essayer de définir le contrôleur est généralement voué à l'échec, tant sa nature change d'une application à l'autre. Certains disent que c'est le code glue qui permet de lier le modèle et la vue. D'autres qu'il contient la logique de flux du programme. Le contrôleur est par ailleurs le point d'entrée d'un programme. Et ce sera essentiellement ça, le contrôleur de notre programme : un point d'entrée. Indication : le schéma ci-dessous permet uniquement de visualiser la forme du MVC.

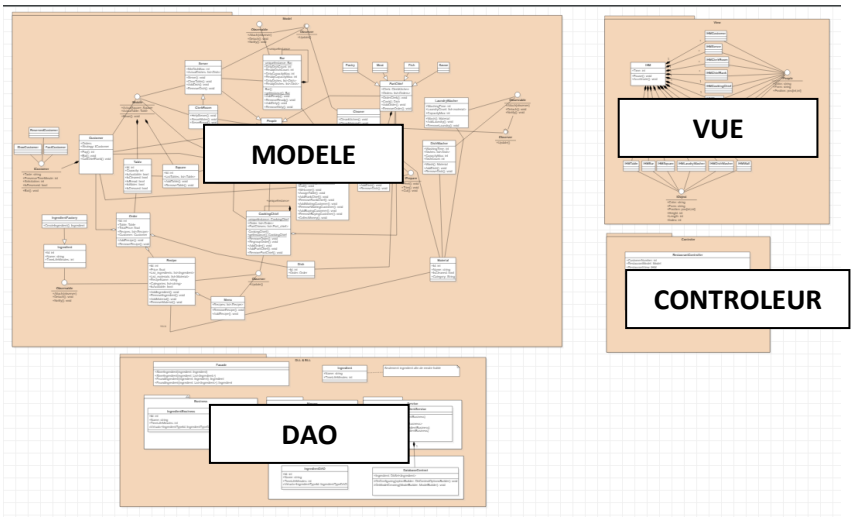


Figure 10 : Vue globale du MVC de l'application

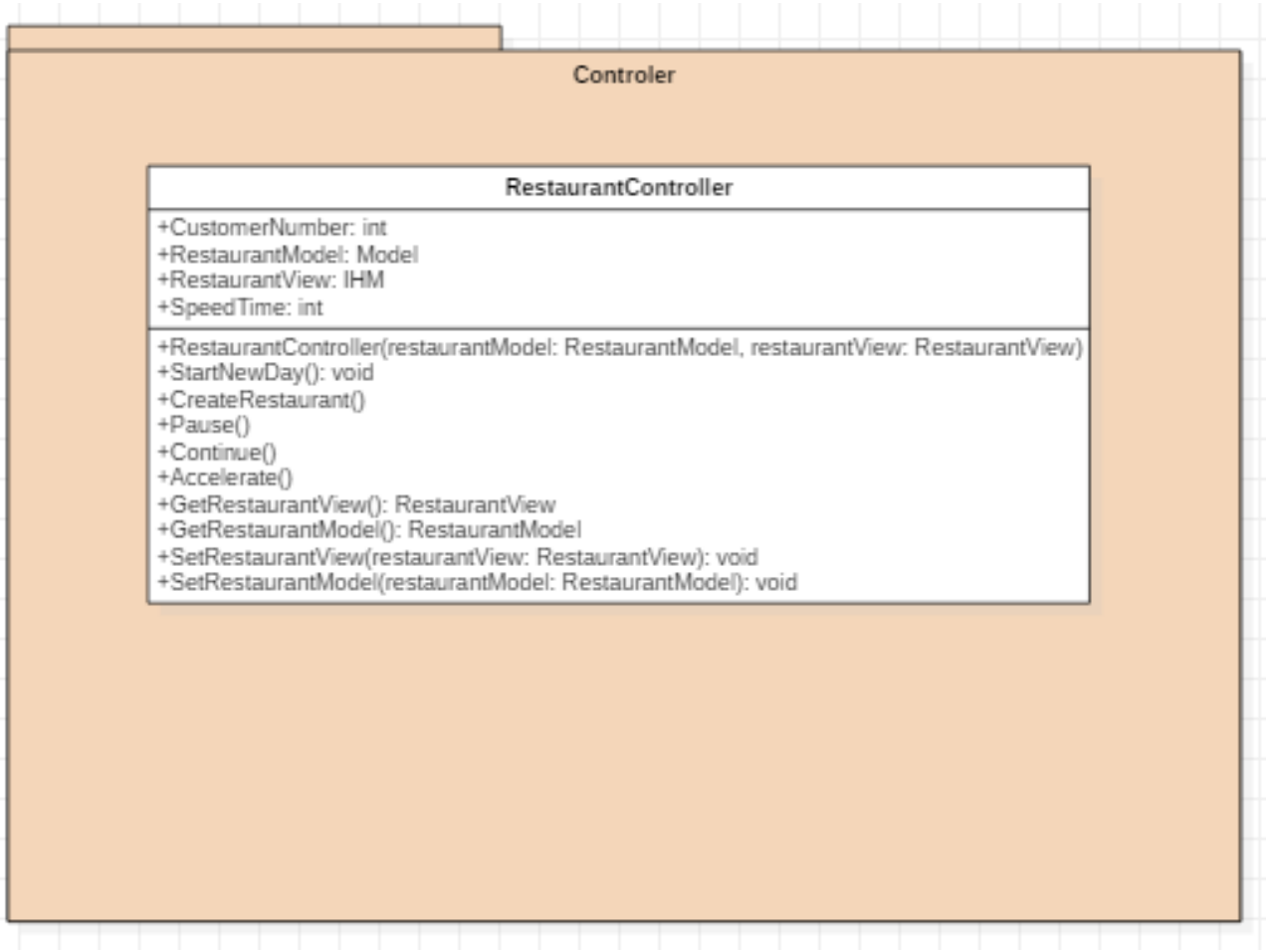


Figure 11 : Zoom sur le contrôleur de notre application

Pour des soucis de taille, retrouver le diagramme du Modèle en annexe ; « Annexe 3 : Zoom sur le modèle (MVC) de notre application » ou joint à ce rapport.

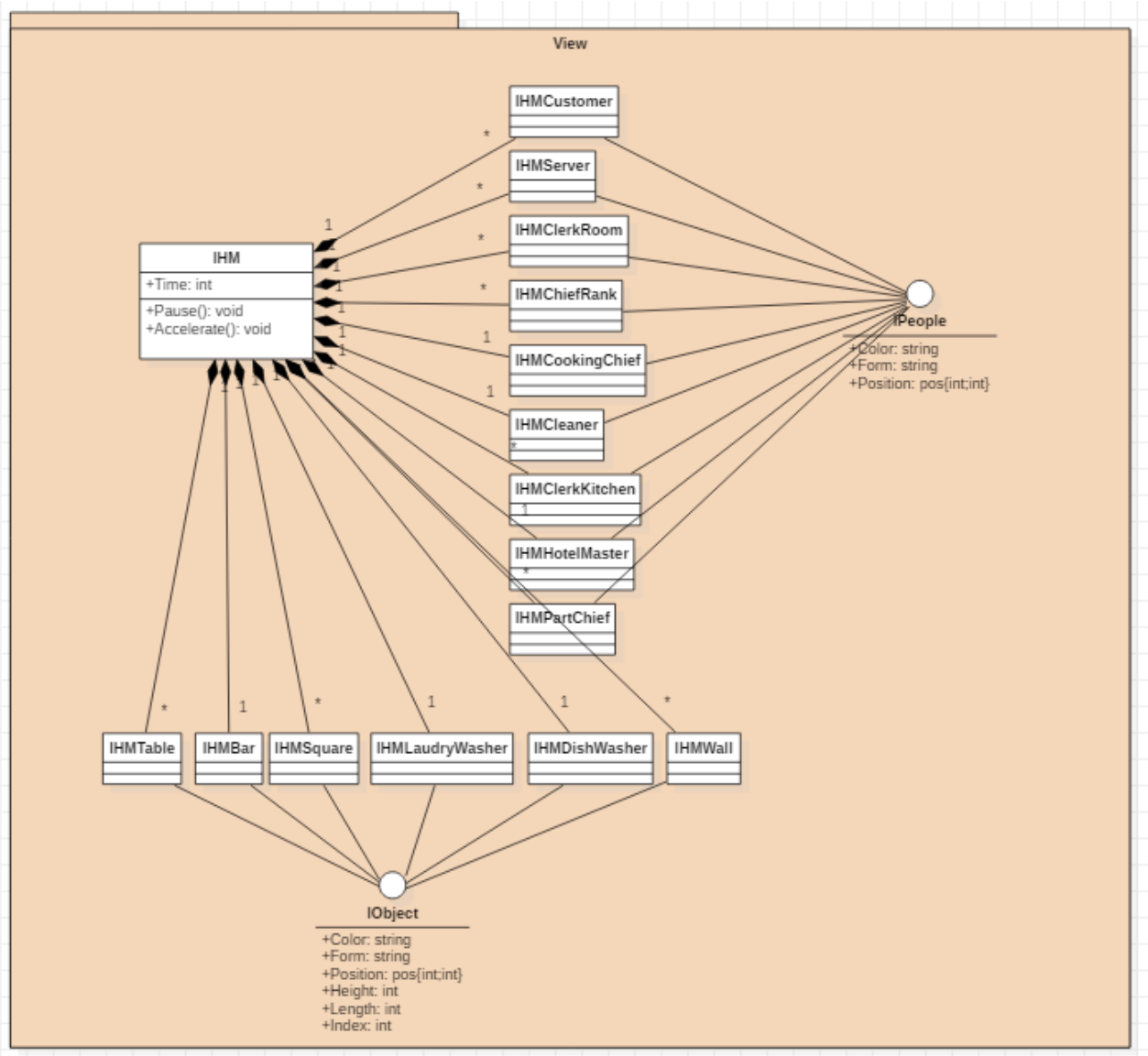


Figure 12 : Zoom sur la vue de notre application

Le package DLL&BLL (**voir ci-dessous**) permet d'utiliser le design pattern DAO (voir V.c)6.). Comme celui-ci doit être appliqué à chacune des classes, cela permet d'éviter la répétition.

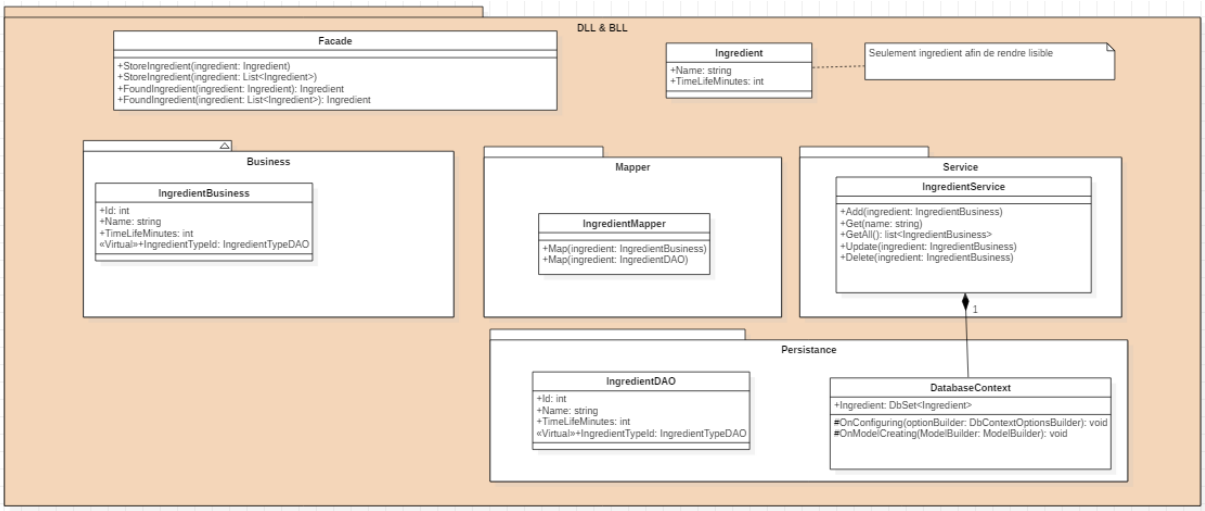


Figure 13 : Zoom sur le package BLL & DLL de notre application

2. Le design pattern Singleton

Le design pattern Singleton va nous permettre de bloquer une classe, c'est-à-dire qu'on aura au maximum un objet d'implémenté pour celle-ci. On l'a donc implémenté pour notre classe qui correspond à notre classe « CookingChief » (chef de cuisine), « Bar » (comptoir), ainsi que notre « HotelMaster » (maitre d'hôtel) qui doivent être unique.

Voici l'exemple de l'implémentation du design pattern singleton sur le « Bar » :

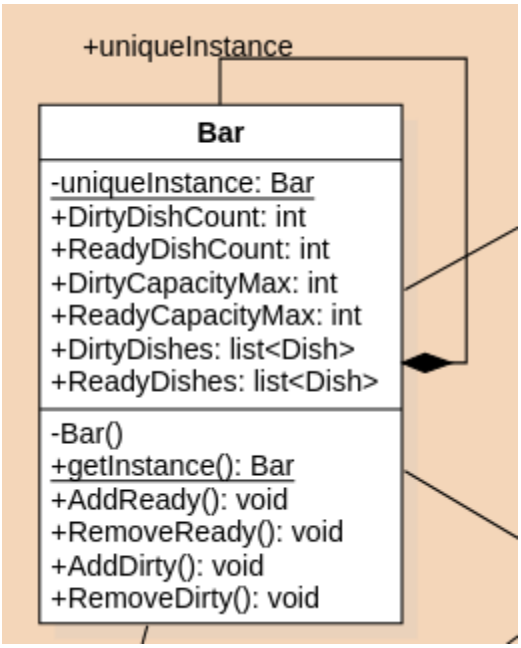


Figure 14: Design pattern Singleton appliqué au « Bar »

3. Le design pattern Fabrique

La fabrique 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. On l'implémente sur nos classes « Material » et « Ingredient » afin de pouvoir les créer de façon optimale dans notre code.

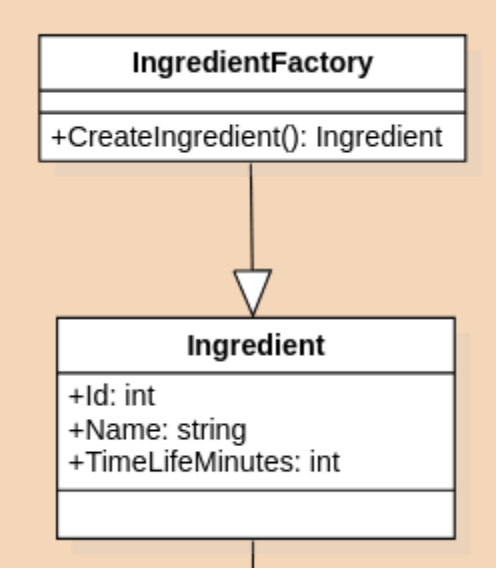


Figure 15 : Le design pattern Fabrique appliqué à « Ingredient »

4. Le design pattern Observeur

L'observateur permet d'envoyer un signal à des modules qui jouent 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). Cela va nous permettre simplifier notre gestion de l'observation des événements. C'est-à-dire qu'au niveau du comptoir, le serveur ainsi que le commis vont vérifier à tout moment s'il y a des plats prêts ou sales afin de les amener en salle pour être consommés ou au lave-vaisselle pour être lavé. Ou encore, au niveau de notre plongeur qui va vérifier à tout moment l'état de son lave-linge ou lave-vaisselle afin de savoir s'il doit le vider.

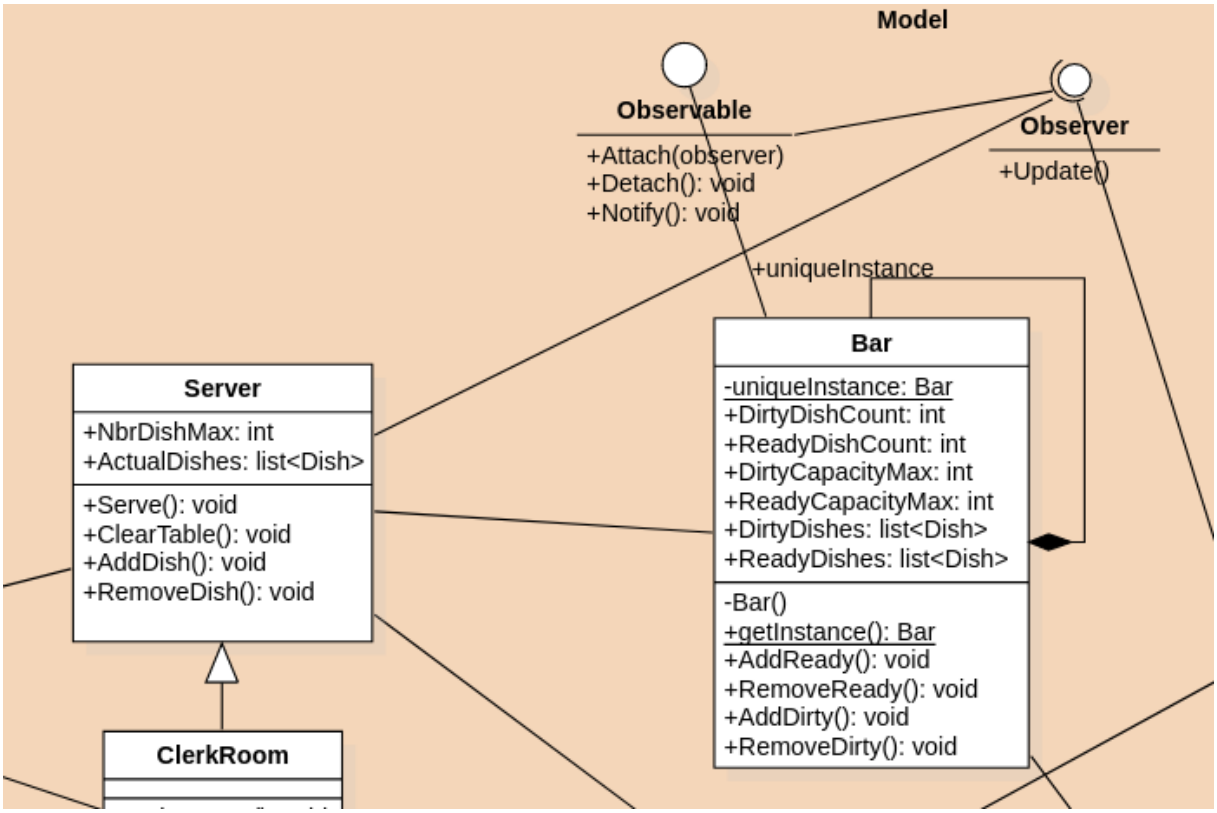


Figure 16 : Le design pattern Observeur appliqué au « Bar »

5. Le design pattern Stratégie

Ce design pattern crée une famille d'algorithme encapsulée sur le même objet et ainsi permet pour un même objet de choisir un algorithme différent. Au niveau de notre diagramme de classe, on implémente ce pattern pour notre client, car celui-ci peut être pressé, détendu et peut avoir réservé ce qui définit un algorithme différent pour un même client.

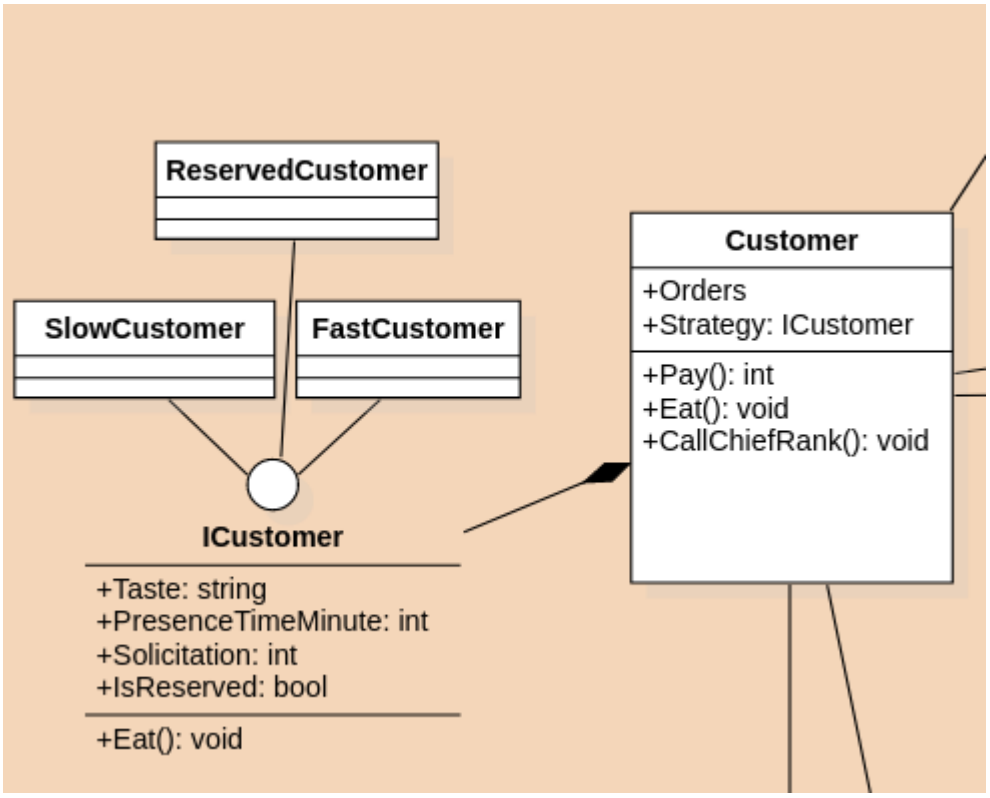


Figure 17 : Le design pattern Stratégie appliqué au « Customer »

6. Le design pattern Data Access Object

Le pattern DAO (Data Access Object) permet de faire le lien entre la couche métier et la couche persistante, ceci afin de centraliser les mécanismes de mapping entre notre système de stockage et nos objets. Ce sera donc par le biais de ces objets spécifiques de la DAO que nous pourrons récupérer des instances de nos objets métiers correspondants à des entrées dans notre base de données. Nous pouvons donc déterminer la façon dont nos objets vont travailler car nous connaissons les actions que ces objets devront faire.

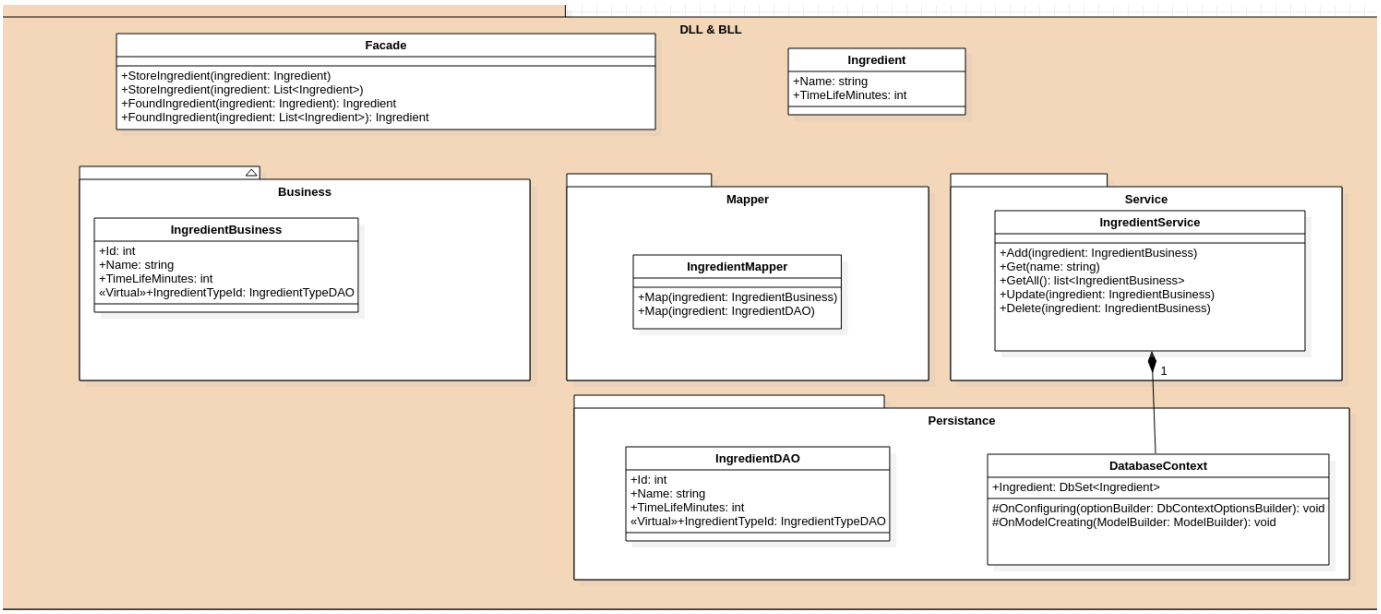


Figure 18 : Le design pattern DAO de l'application

d) Diagramme de séquences

Dans la continuité de notre conception UML, nous avons réalisé un diagramme de séquences pour représenter la séquence de messages entre les objets au cours d'une interaction. Afin de représenter au mieux le besoin posé, nous avons divisé ce diagramme en 4 parties ; les 4 grandes étapes de notre restaurant. C'est-à-dire « Accueillir » (sous-entendu le client), « Cuisiner », « Terminer le repas » et « Nettoyer ».

Notre premier diagramme (**voir ci-dessous**) représente l'accueil d'un client ainsi que comment celui-ci est servi. Lorsqu'un client arrive restaurant, le Maître d'hôtel l'accueille et lui assigne une table disponible. Ce dernier appel alors le Chef de rang qui l'amène à sa table en lui laissant une carte du restaurant puis attendre que le client ai choisi ce qu'il souhaite manger. C'est alors au commis de déposer la carafe d'eau et la corbeille de pain sur la table. Une fois que les clients ont passé commande auprès du Chef de rang, ce dernier la transmet au chef de cuisine. Une fois le plat prêt, le serveur s'occupe de l'apporter à la table du client.

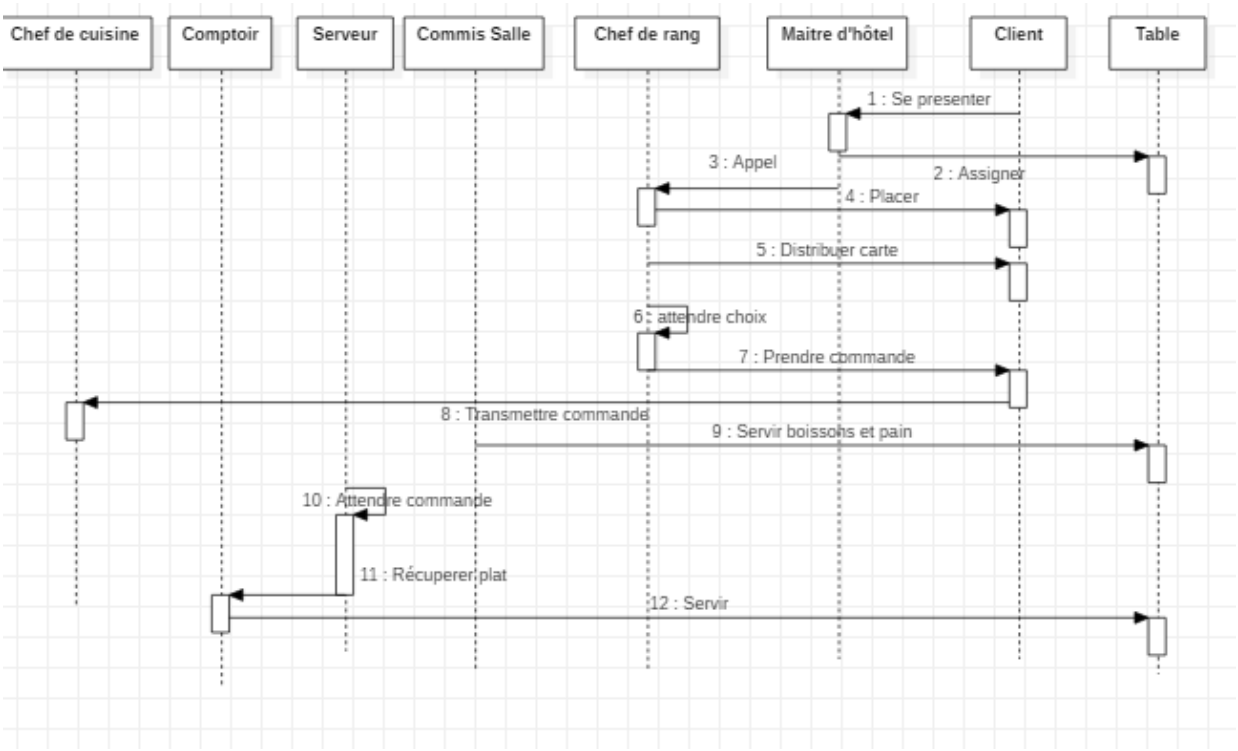


Figure 19: Digramme de séquence « Accueillir » (client)

Notre second diagramme (**voir ci-dessous**) se concentre sur le fonctionnement de la cuisine du restaurant. Comme on peut le voir, il débute lorsque le Chef de rang de la salle de restaurant transmet une commande au Chef de cuisine. Ce dernier réparti les tâches entre les différents cuisiniers (aussi nommé Maître de partie) qui, pour cuisiner, sollicite ses commis :

- Pour aller chercher les ingrédients en stock (chambre froide, congélateur, zone de denrée) et/ou pour les aider à préparer en épluchant, coupant ou encore taillant les ingrédients
- Puis pour emmener les plats prêts sur le comptoir pour que les serveurs en salle s'en occupe

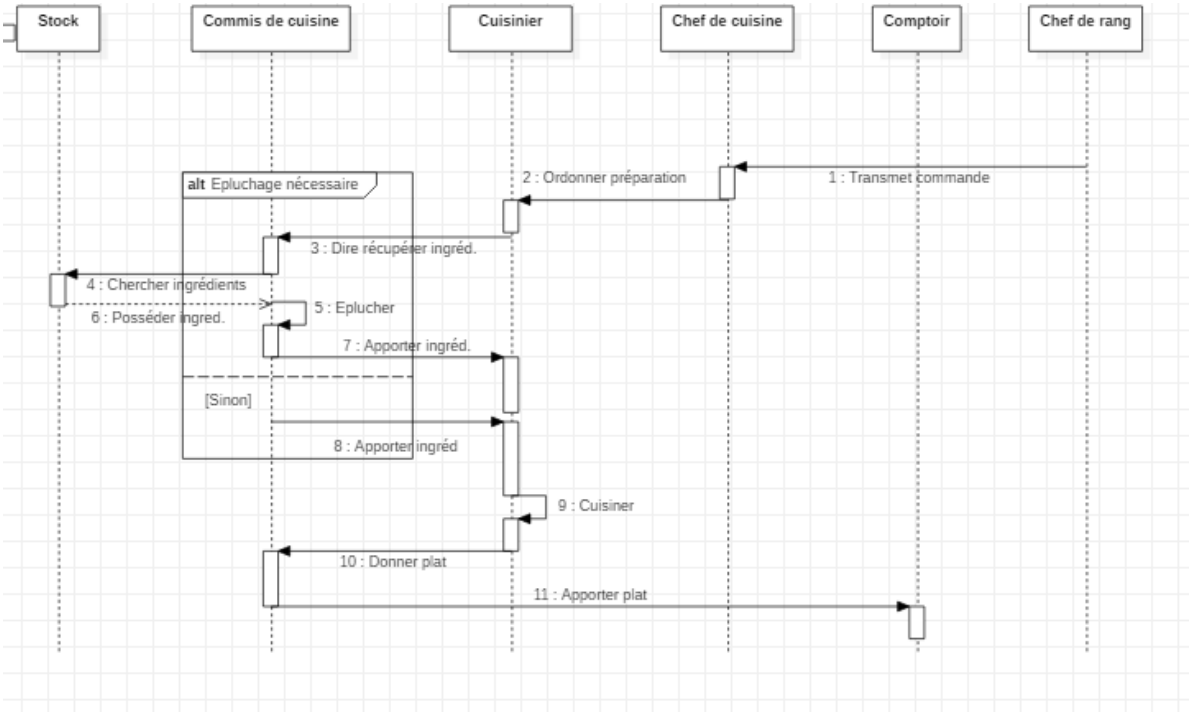


Figure 20: Diagramme de séquence "Cuisiner"

Notre troisième diagramme (**voir ci-dessous**) décrit comment se passe la fin du repas. En effet, le repas se termine lorsque le client a fini de manger. Il appelle ensuite le chef de rang qui va lui donner l'addition de son repas. Le client paie, part, puis le Maître d'hôtel demande au serveur de débarrasser la table. Une fois cette dernière tâche réalisée, le Chef de rang dresse la table pour les futurs clients.

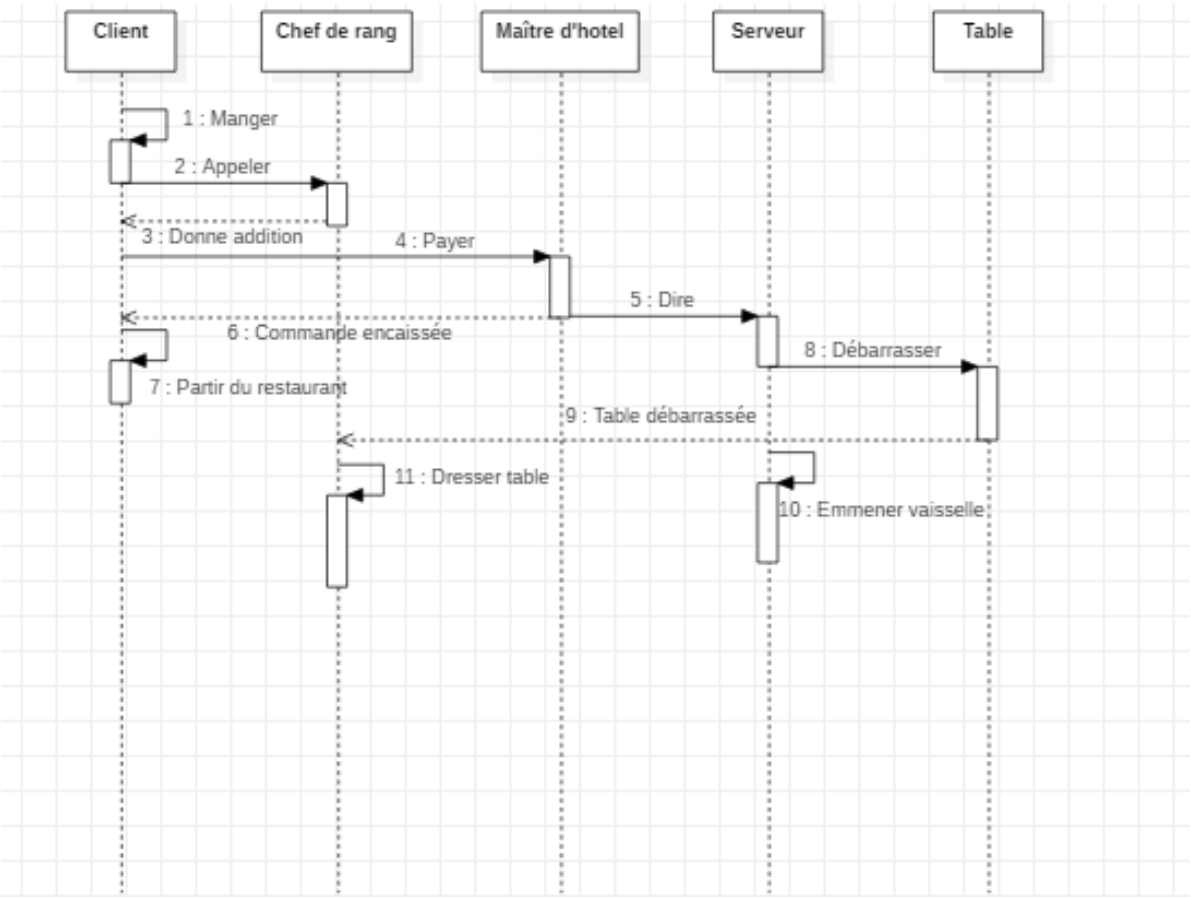


Figure 21: Diagramme de séquence « Terminer repas »

Notre quatrième et dernier diagramme (**voir ci-dessous**) permet de comprendre comment le plongeur agit. En effet, une fois que le serveur a déposé la vaisselle et les serviettes/nappes salles, le plongeur va remplir, grâce à ces dernier, la machine à laver et le lave-vaisselle. Une fois les cycles terminés, il vide ces derniers. Le plongeur peut aussi, comme le commis, aider le cuisinier.

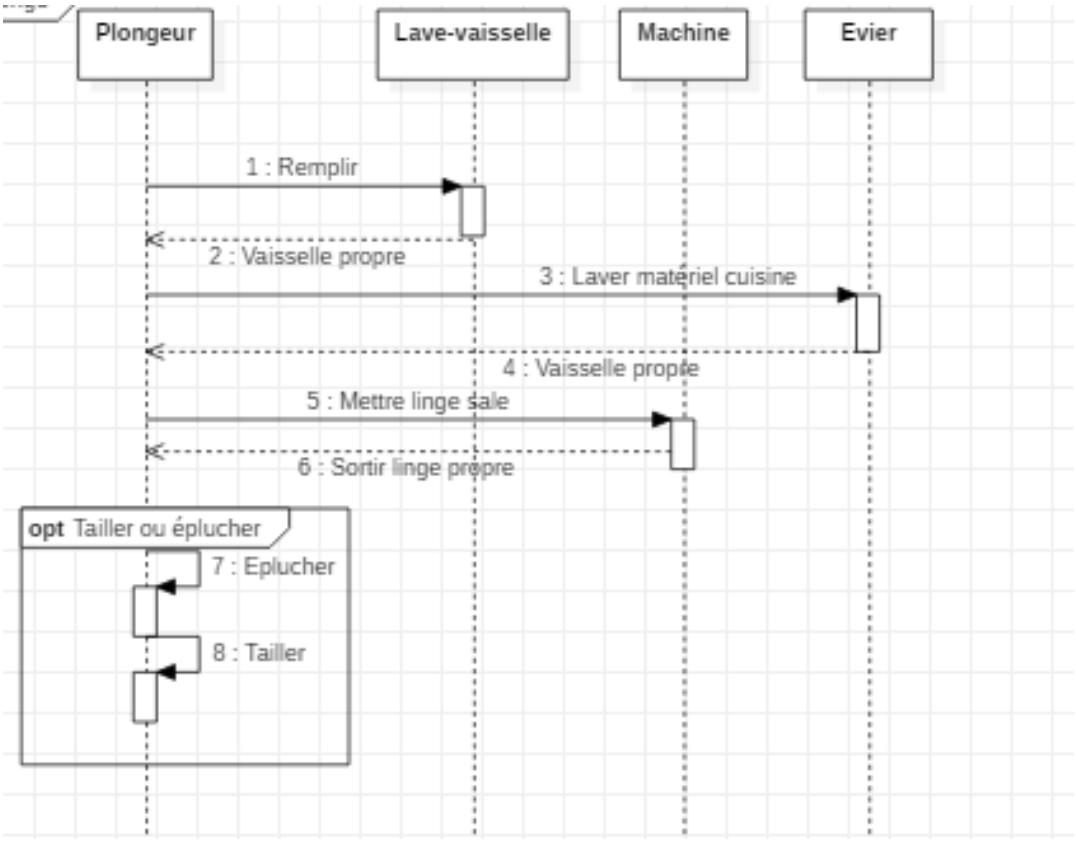


Figure 22: Diagramme de séquence « Nettoyer »

e) Diagramme de composants

Maintenant vient le diagramme de composants. Il permet d'illustrer les parties du logiciel, des contrôleurs intégrés, etc, qui feront partie d'un système

Le but est de proposer une interface facilitant la mise en œuvre d'un ensemble de classes généralement regroupées dans un ou plusieurs sous-systèmes. Le motif Façade permet d'offrir un niveau d'abstraction entre l'ensemble de classes et celles qui souhaitent les utiliser en proposant une interface de plus haut niveau pour utiliser les classes du sous-système.

Ici, comme nous avons appliqué un modèle MVC, nous pouvons observer les 3 packages correspondant ainsi que celui de la base de données reliée au « Model ».

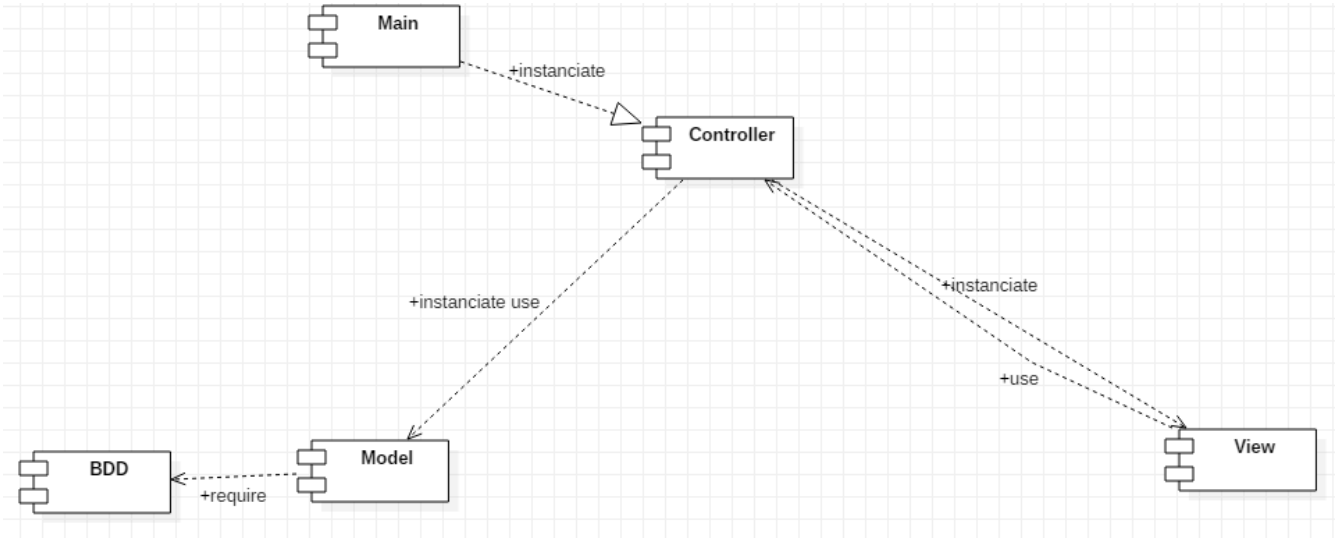


Figure 23 : Diagramme de composants de l'application

f) Modèle conceptuel de données (MCD)

Passons maintenant à la base de données et comment nous souhaitons l'organiser.

Lors de cette application nous sommes parties sur le principe que l'application serait modulable directement sur l'interface implémenter, pour cela nous avons tout répertorier en base de données pour ne pas avoir de valeur brute à rentrer dans le code.

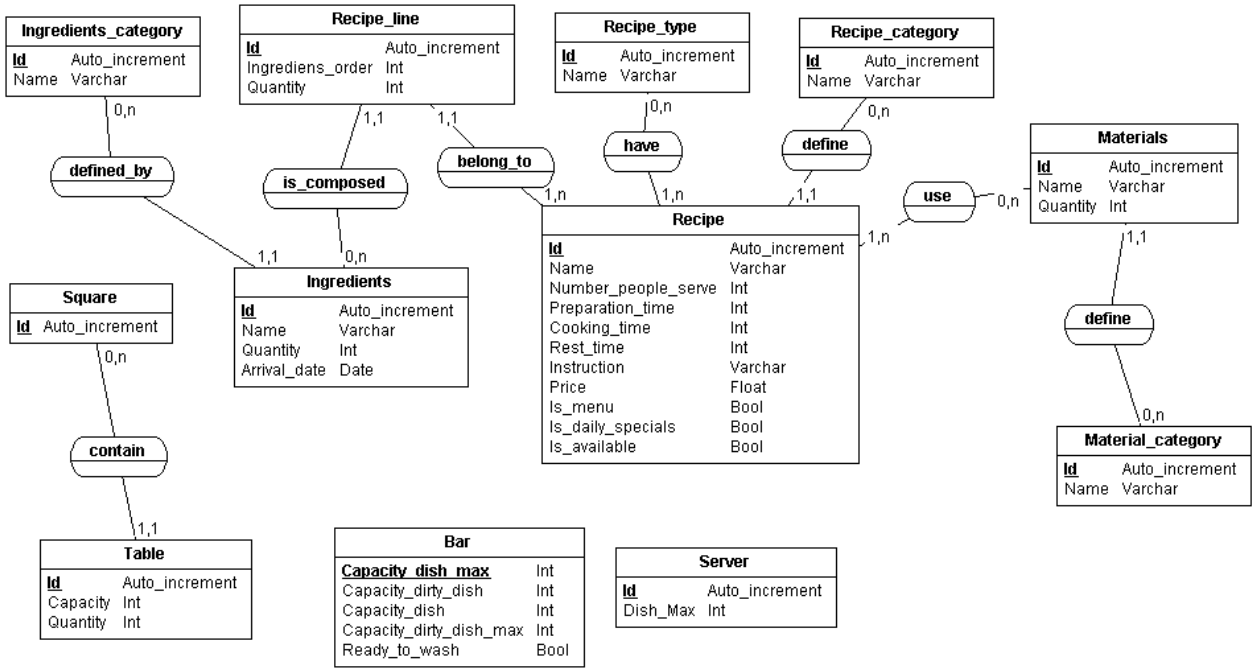


Figure 24 : vue globaleodèle conceptuel des données (MCD)

Dans ce MCD chaque relation correspond à une fonctionnalité nécessaire pour notre application ou pour le restaurant comme les différents postes reliés aux différents objets nécessaires à la réalisation de leurs travaux. Nous avons également différents types comme Recipe_type qui permet de connaître le type de recette car, comme souvent dans les restaurants, il y a un chef de partie par domaine d'activité.

Pour faciliter aux restaurants l'utilisation de l'application nous récupérons les lignes de recette dans une table nommé Recipe_line qui permet de stocker toutes les lignes des recettes pour la suivre correctement.

- **Le script SQL**

À la suite de ce MCD, nous avons pu extraire le code SQL de notre base de données que vous trouverez joint à ce rapport.

- **Le dictionnaire des données**

De notre MCD a pu découdre le dictionnaire des données. En effet, comme son nom l'indique, ce dictionnaire permet de recenser toutes les données que nous allons utiliser et manipuler dans notre application. Il précise leur désignation, leur code mnémonique, leur type et leur taille (maximum).

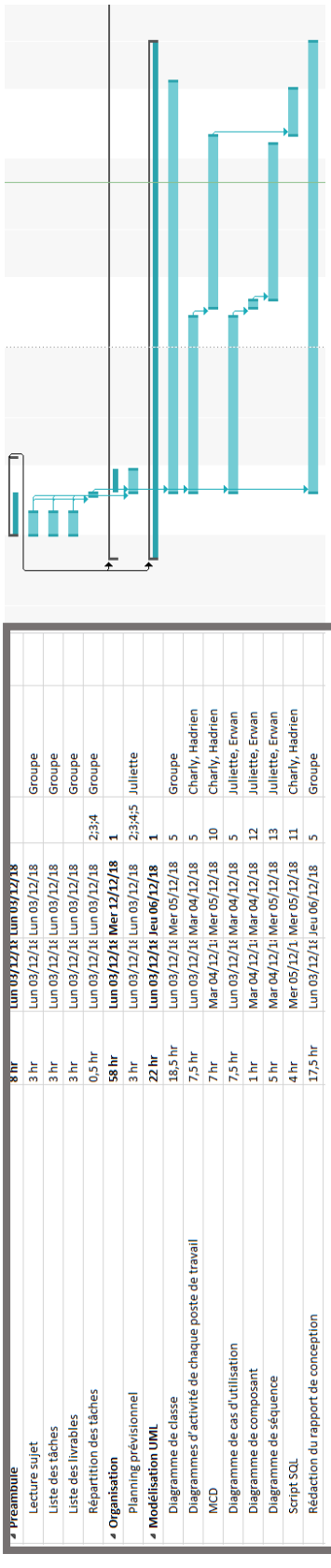
Voici notre dictionnaire des données :

Numéro	Code mnémonique	Désignation	Type de données	Taille
1	ID	ID des objets (à trouver dans plusieurs tables)	AUTO-INCREMENT	7
2	Name	Nom des objets (à trouver dans plusieurs tables)	Varchar	255
3	Number_people_serve	Nombre personne pour une recette	INT	7
4	Preparation_time	Temps de préparation	INT	7
5	Cooking_time	Temps de cuisson	INT	7
6	Rest_time	Temps de repos	INT	7
7	Instruction	Instruction de la recette	Varchar	255
8	Price	Prix du plat sur la carte	Float	255
9	Quantity	Quantité de l'objet (table, matériel ou ingrédient en stock)	INT	7
10	Arrival_Date	Date de l'arrivé de l'ingrédient	Date	
11	Capacity_dish_max_bar	Capacité max de plat propre sur le bar	INT	7
12	Capacity_dirty_dish_max	Capacité max de plat sale sur le bar	INT	7
13	Capacity_dish	Capacité de plat prêt sur le bar	INT	7
14	Capacity_dirty_dish	Capacité de plat actuel sale sur le bar	INT	7
15	Ready_to_wash	Plat prêt pour savoir s'il y a des plats à laver	INT	7
16	Capacity	Capacité de la table	INT	7

Figure 25: Dictionnaire des données de l'application

ANNEXES

Annexe 1 : Planning initial du projet (GANTT)



◢ **Préambule**	**8 hr**	**Lun 03/12/18**	**Lun 03/12/18**		
Lecture sujet	3 hr	Lun 03/12/18	Lun 03/12/18		Groupe
Liste des tâches	3 hr	Lun 03/12/18	Lun 03/12/18		Groupe
Liste des livrables	3 hr	Lun 03/12/18	Lun 03/12/18		Groupe
Répartition des tâches	0,5 hr	Lun 03/12/18	Lun 03/12/18	2;3;4	Groupe
◢ **Organisation**	**58 hr**	**Lun 03/12/18**	**Mer 12/12/18**	**1**	
Planning prévisionnel	3 hr	Lun 03/12/18	Lun 03/12/18	2;3;4;5	Juliette
◢ **Modélisation UML**	**22 hr**	**Lun 03/12/18**	**Jeu 06/12/18**	**1**	
Diagramme de classe	18,5 hr	Lun 03/12/18	Mer 05/12/18	5	Groupe
Diagrammes d'activité de chaque poste de travail	7,5 hr	Lun 03/12/18	Mar 04/12/18	5	Charly, Hadrien
MCD	7 hr	Mar 04/12/18	Mer 05/12/18	10	Charly, Hadrien
Diagramme de cas d'utilisation	7,5 hr	Lun 03/12/18	Mar 04/12/18	5	Juliette, Erwan
Diagramme de composant	1 hr	Mar 04/12/18	Mar 04/12/18	12	Juliette, Erwan
Diagramme de séquence	5 hr	Mar 04/12/18	Mer 05/12/18	13	Juliette, Erwan
Script SQL	4 hr	Mer 05/12/18	Mer 05/12/18	11	Charly, Hadrien
Rédaction du rapport de conception	17,5 hr	Lun 03/12/18	Jeu 06/12/18	5	Groupe

Figure 26 : Planning initial du projet

Annexe 2 : Diagramme d'activité du plongeur

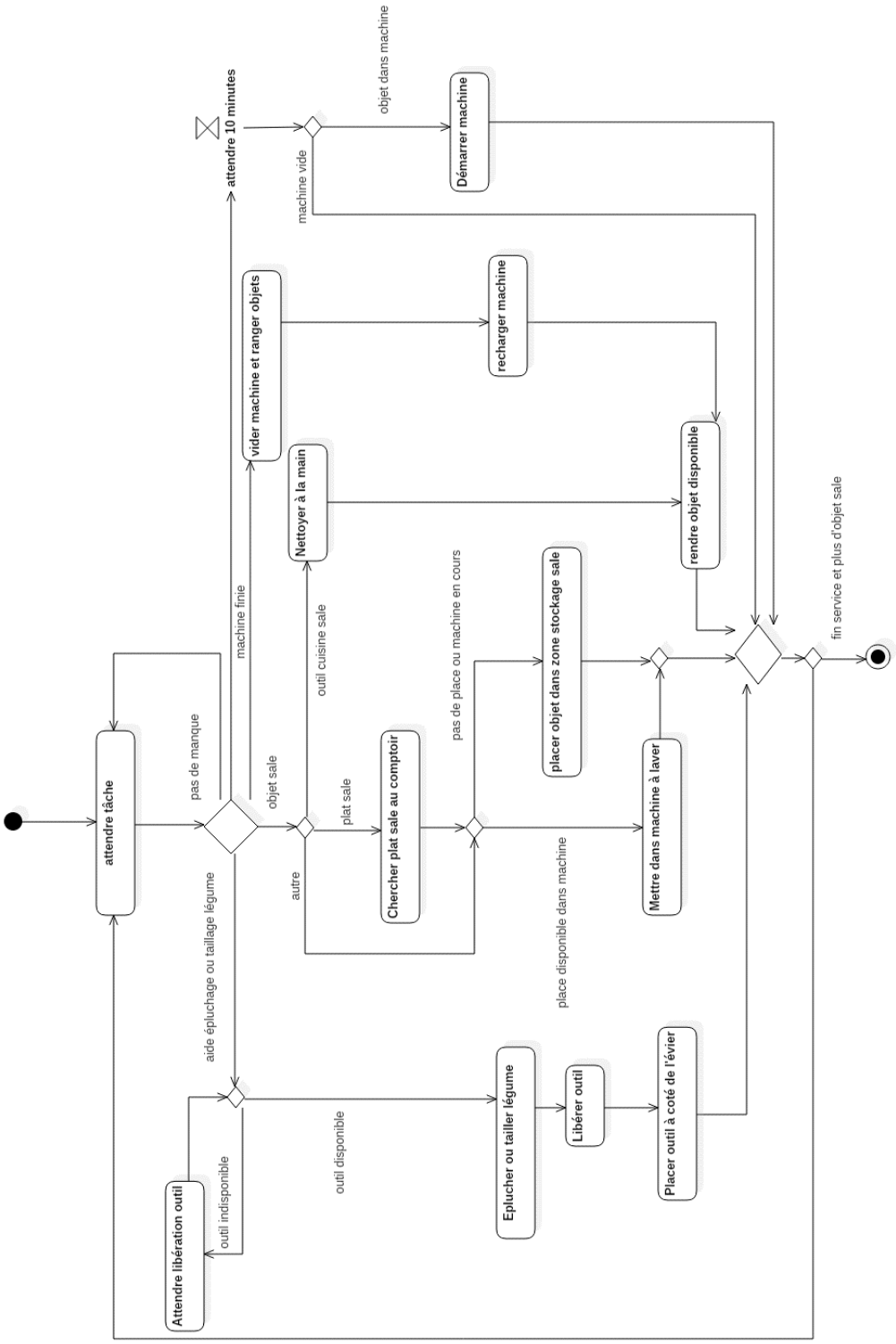


Figure 27: Diagramme d'activité du(/des) plongeur(s)

Annexe 3 : Zoom sur le modèle (MVC) de notre application

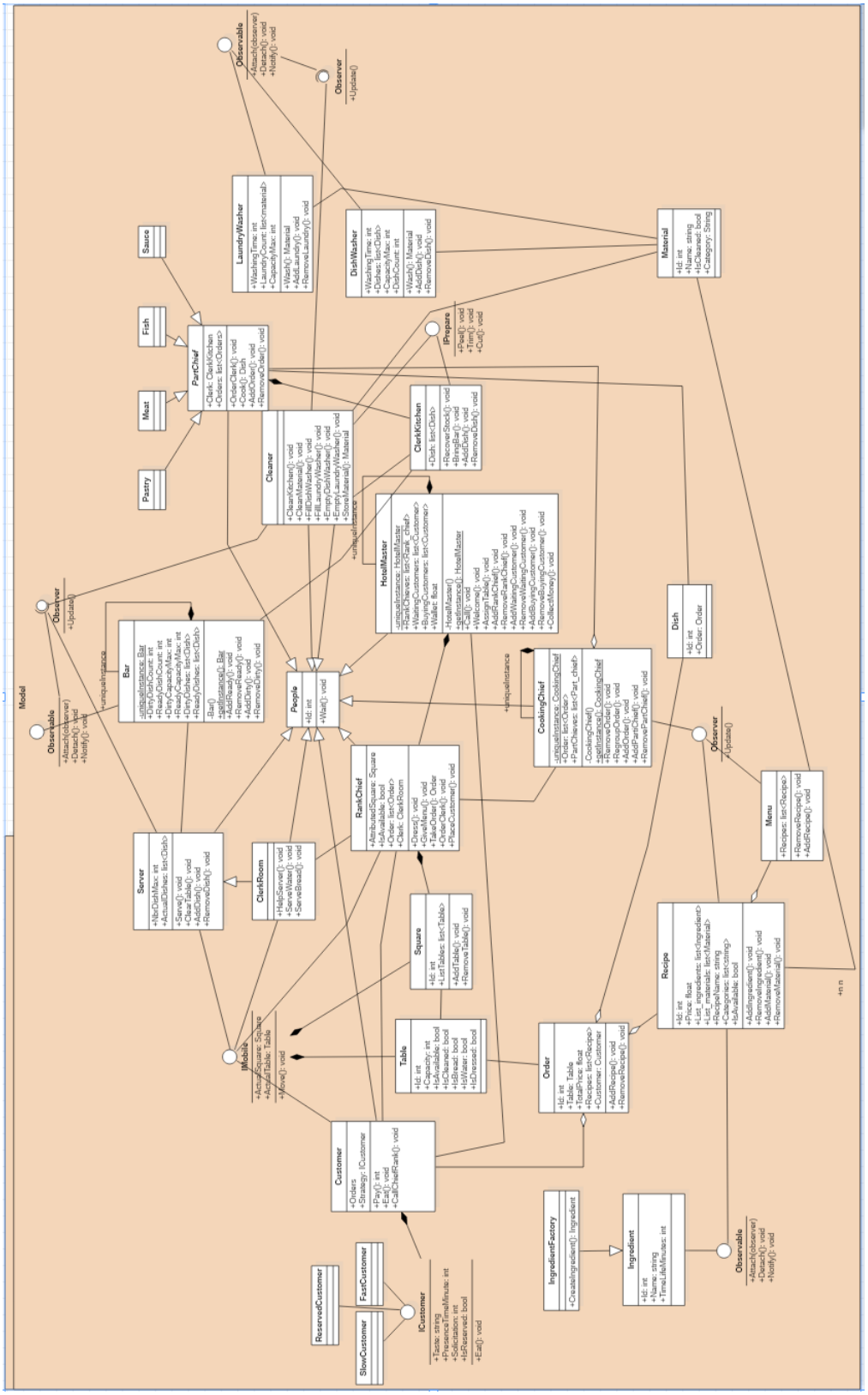


Figure 28 : Zoom sur le modèle de notre application