Maréchal Laure-Anne, Séré Peyrigain Vincent, Duthieuw Marine, Dupont Alexis

Rapport de conception

UML et BDD

Table des matières

I. L'UML:

a. Diagramme de classes

Le diagramme de classes va nous permettre, comme son nom l'indique, de représenter les différentes classes qui vont constituer notre programme. Il s'agit d'un diagramme qui va montrer la structure interne de notre système.

Nous travaillons donc avec une architecture Modèle-Vue-Contrôleur. Notre application suive une architecture en trois couches (Présentation, Traitement et Données). Nous avons ainsi réalisé un premier diagramme de classe qui comprend notre modèle (Couche traitement) puis un second avec le contrôleur et la vue (Couche présentation).

Dans le diagramme de classe du modèle, nous avons détailler les différentes classes pour qui vont manipuler les informations ainsi que les classes qui vont nous permettre de nous connecter à la base de données. Ainsi, nous avons le package DAL and BLL (pour Data Access Layer and Business Logic Layer) qui va contenir nos classes qui vont nous servir pour la récupération et l'envois d'information dans la base de données. Nous n'avons pas modélisé la totalité des classes qui seront présentes dans ce package. Nous avons fait le choix de ne représenter qu'un exemple de classe qu'il y aura pour garder notre diagramme de classe lisible.

Nous avons une totale de sept Design Patterns :

- Observer
- Decorator
- Singleton
- Strategy
- Factory
- Facade
- MVC

Nous les avons tous détailler dans une partie qui leur est dédiée.

b. Diagramme de séquence

Le diagramme de séquence est un outil de représentation des interactions entre les différents éléments du système et les acteurs.

Après lecture du sujet, nous avons convenu la création de plusieurs diagrammes de séquences, afin de modéliser notre système le plus clairement possible.

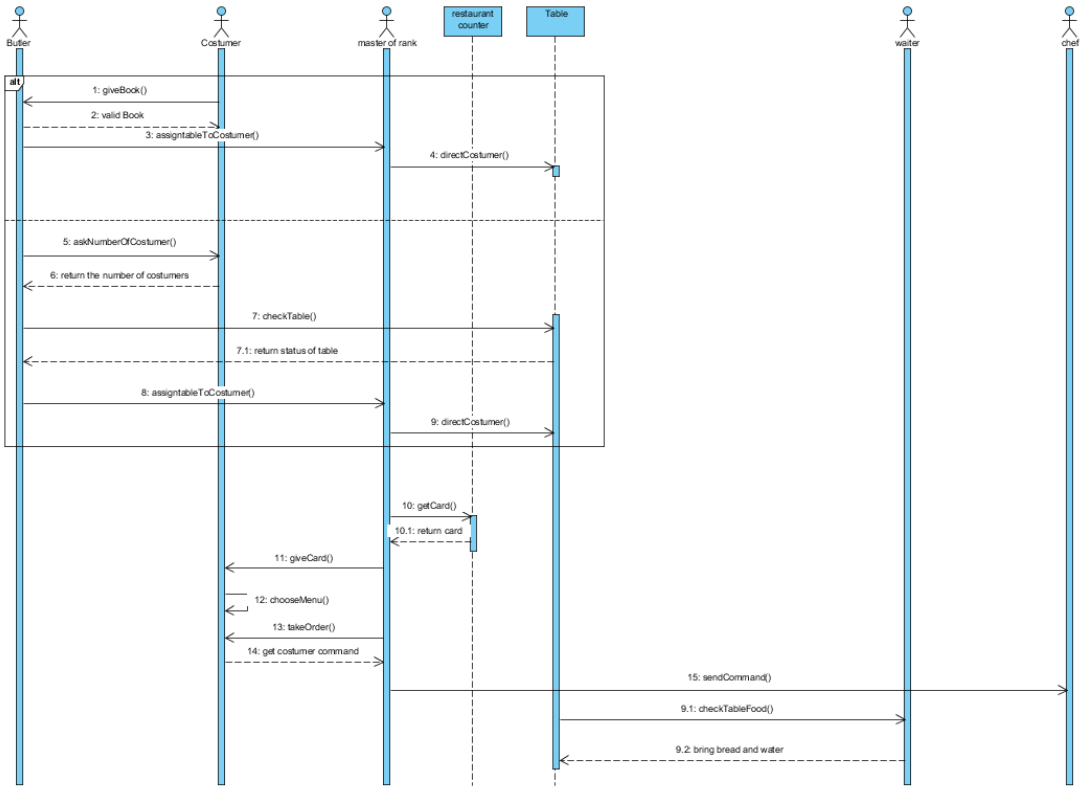
Le premier représentera le fonctionnement du système lors de l'arrivée d'un ou de plusieurs clients jusqu'au passage de leurs commandes en cuisine.

Le deuxième montrera les interactions des différents acteurs et élément de la cuisine après la réception d'une commande.

Le troisième schéma décrira le retour d'un plat en cuisine.

Et pour finir, le quatrième diagramme expliquera le comportement des éléments et acteur de la salle à manger.

Afin de justifier les différents diagrammes, nous rédigerons une description de celui-ci afin d'écrire ce que nous avons voulu montrer au travers de ces diagrammes.



1 : giveBook() / 2 : valid book / 3 : assignTableToCostumer() / 4 : directCostumers()

Pour commencer notre diagramme nous avons distingué plusieurs cas de figure.

Lorsqu'un ou des clients arrivent, ils ont soit une réservation ou ils n'en ont pas.

Dans le cadre ou le client possède une réservation, il la donne au maître d'hôtel, celui-ci donne la table qui est assignée au client au maître de rang qui emmène le client à sa table.

5 : askNumberOfCostumer() / 6 : return number of costumer

Dans le cas où il n'y a pas de réservation, le maître d'hôtel demande au client le nombre de clients et le client renvoie un nombre afin de déterminer la table qui lui sera attribuée.

7 : checkTable() / 7.1 : return status of table

Afin que le maître d'hôtel détermine la table à attribuer au client, il doit avant tout regarder s'il possède des tables disponibles, et trouver la plus proche. Le maître d'hôtel interroge ainsi l'objet Table et celui-ci renvoie l'état des tables au maître d'hôtel.

8 : assignTableToCostumer()

Une fois la table adéquate au nombre de client déterminé, le maître d'hôtel assigne une table au client.

9 : directCostumer()

Le maître de rang dirige le client à sa place.

10 : getCard() / 10.1 : return card

Le maître de rang récupère le nombre de cartes requise (selon le nombre de clients) auprès de l'objet qui stocke les cartes.

11 : giveCard()

Le maître de rang donne les cartes au client afin qu'il fasse son choix, il patiente 5 minutes puis il récupère la commande du client.

12 : chooseMenu()

Le client choisit son menu.

13 : takeOrder() / 14 : get costumer command

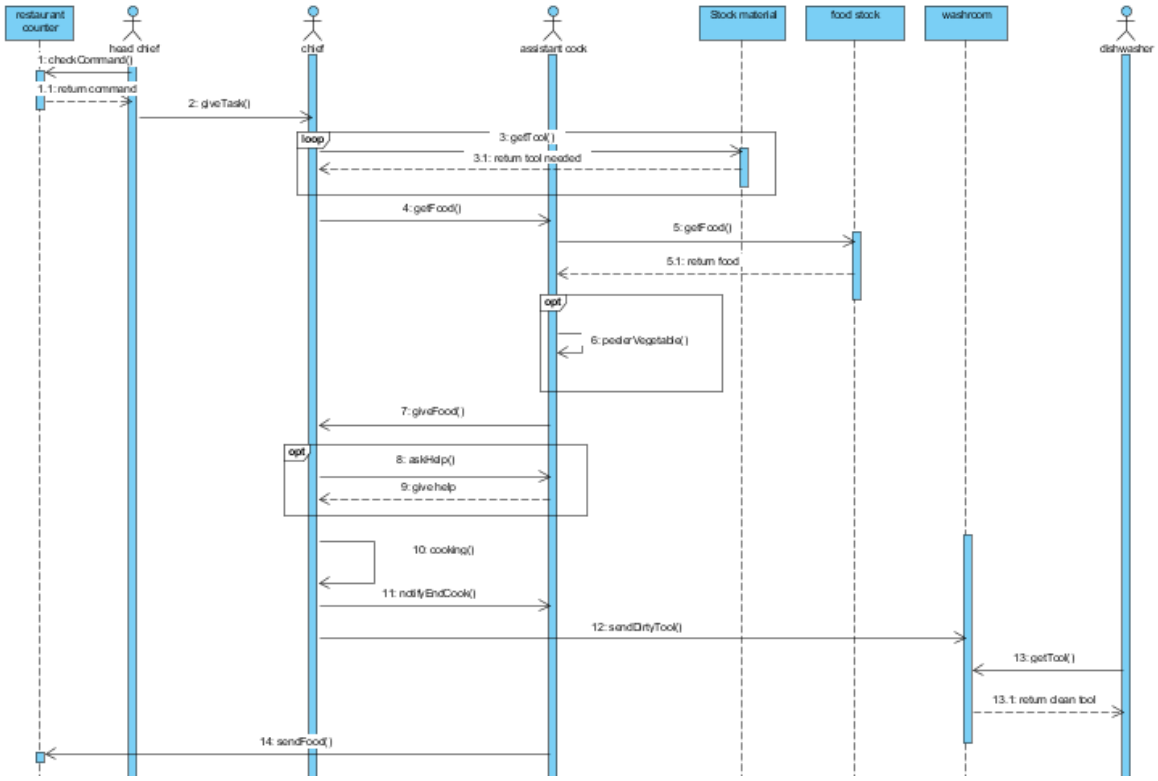
Le maître de rang récupère la commande du client.

15 : sendCommand()

Le chef de rang envoie au chef de cuisine la commande des clients.

9.1 : checkTableFood() / 9.2 : bring bread and water

Une fois la commande envoyée au chef de cuisine, le serveur doit mettre du pain et de l'eau sur la table.



1 : checkCommand()

Le chef de cuisine vérifie si une commande est arrivée au comptoir.

2 : giveTask()

Après avoir reçu les commandes du maître de rang, le chef de cuisine doit répartir les tâches et optimiser leur réalisation. Il les distribue ces tâches aux cuisiniers.

3 : getTool() / 3.1 return tool needed

Les cuisiniers ont besoin d'ustensile de cuisine pour réaliser les plats, il demande directement au stock de matériel de fournir le matériel.

4 : getFood()

Pour la récupération de la nourriture nécessaire à la réalisation du plat, le cuisinier demande au commis de lui chercher la nourriture.

5 : getFood() / 5.1 return food

Le commis demande au stock alimentaire de récupérer des aliments.

6 : peelerVegetable()

Ici, nous avons mis une option « éplucher légume » car il est possible que le commis de cuisine aide le cuisinier sur cette tâche.

7 : giveFood()

Le commis donne les aliments au cuisinier.

8 : getHelp() / 9 : give help

Dans le cas où le cuisinier a besoin d'aide, le commis peut l'aider. Le Cuisinier a donc la capacité de demander de l'aide et le commis l'aide en retour.

8 : cooking()

Correspond au temps de réalisation du plat fait par le cuisinier.

9 : notifyEndCook()

Une fois le plat fini par le cuisinier, il notifie le commis.

10 : sendDirtyTool()

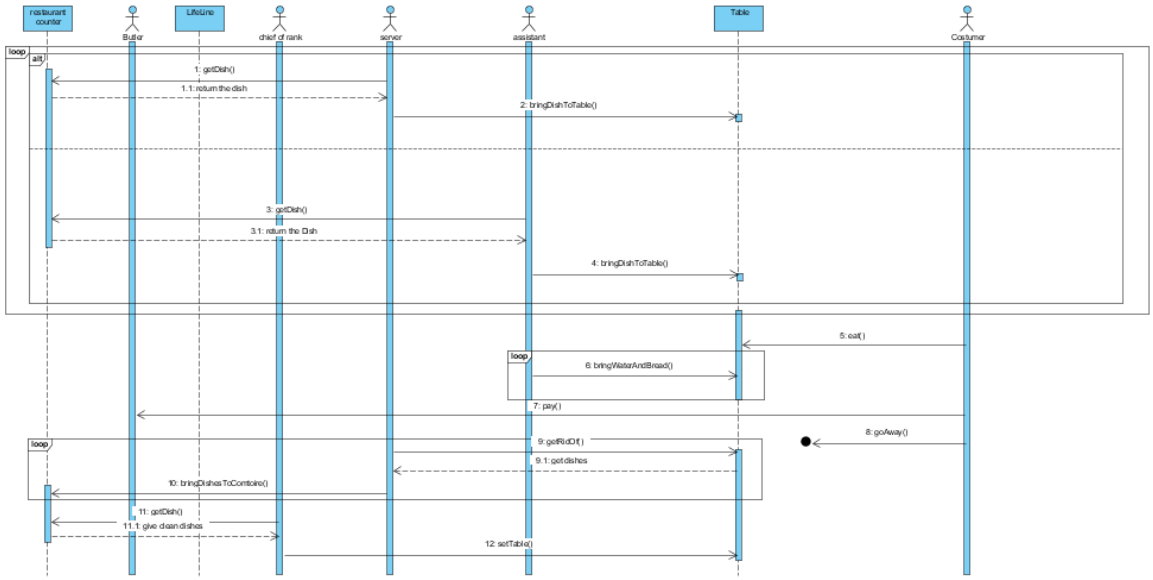
Le cuisinier envoie les outils dont il n'a plus besoin au lavabo qui stocke les outils/vaisselle salle.

11 : getTool() / 12 : return clean tool

Le plongeur récupère les outils sales et les nettoie.

13 : sendFood()

Pour finir le commis envoie au comptoir les plats terminés.



1 : getDish() /return the dish

Une fois le plat préparé par les cuisiniers arrivé au comptoir, c'est au tour du serveur. Celui-ci va aller chercher le plat au comptoir.

2 : bringDishToTable()

Le serveur va ensuite apporter le plat à la table correspondante à la commande.

3 : getDish /return the dish / 4 : bringDishToTable

Ces actions sont les mêmes que celle décrite plus haut. La seule différence réside en l'acteur qui n'est plus le serveur mais le commis de salle.

5 : eat()

Le plat reçu, le client commence à manger le plat sur la table.

6 : bringWaterAndBread()

Nous avons placé une boucle qui demande au commis de salle d'apporter de l'eau et du pain s'il n'y en a plus sur la table.

7 : pay()

Le plat terminé, le client va payer auprès du maître d'hôtel.

8 : goAway()

Le client s'en va.

9 : getRidOf() / 9.1 : get dishes

Le serveur débarrasse la table et récupère les assiettes et couvert sale jusqu'à qu'il n'y en ait plus.

10 : bringDishToCounter

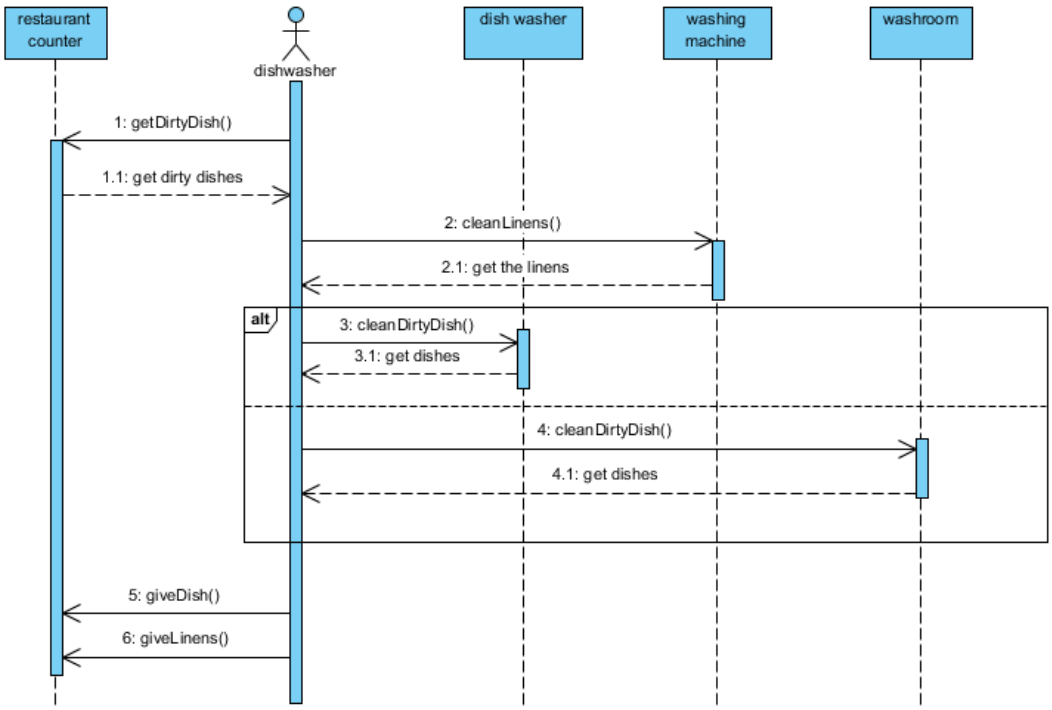
Apporte les assiettes au comptoir jusqu'à ce qu'il y en ait plus.

11 : getDish() / 11.1 : give clean dishes

Le chef de rang demande au comptoir des assiettes et le comptoir lui en envoie.

12 : setTable()

Le chef de rang va mettre la table.



1 : getDirtyDish() /1.1 : get dirty dishes

Le plongeur va chercher les assiettes sales au comptoir.

2 : cleanLinens() / get the linens

Le plongeur va ensuite donner le linge sale au lave-linge.

3 : cleanDirtyDish() /get dishes

Le plongeur envoie les assiettes et couverts sales au lave-vaisselle (attend 30 minutes) puis les récupère propre.

4 : cleanDirtyDish() / 4.1 : get dishes

Le plongeur envoie les outils de cuisine sale à l'évier (attend 30 minutes) puis les récupère propre.

5 : giveDish() / 6 : giveLinens()

Le plongeur envoie les assiettes propre et le linge propre au comptoir.

c. Diagramme d'activité

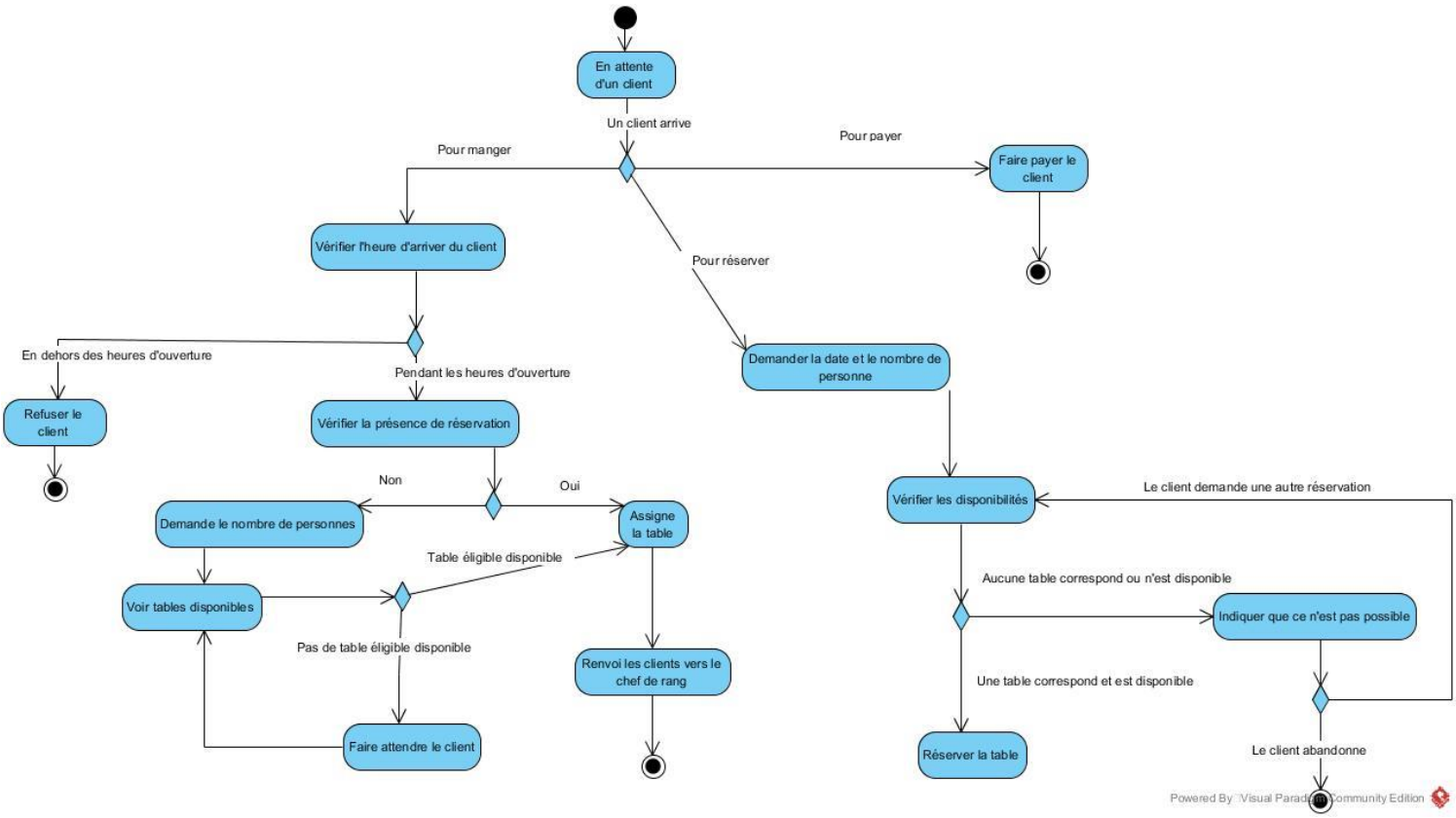
Le diagramme d'activité est un diagramme comportemental qui permet de modéliser des comportements et des déclenchements d'événements en fonction des états du système.

Ce diagramme est utilisé ici pour décrire le flux de travail (workflow) de chaque poste de travail.

Tous les diagrammes d'activités sont en français, car ils permettent de mieux comprendre le rôle de chaque métier. Ils ne seront pas retranscrits directement en code. A l'inverse du diagramme de classes (voir diagramme de classes), celui d'activités ne définit aucun nommage ou convention de nommage, nous l'avons donc réalisé en français pour en faciliter la compréhension.

Voici les différents diagrammes d'activités que nous avons réalisés.

Maître d'hôtel



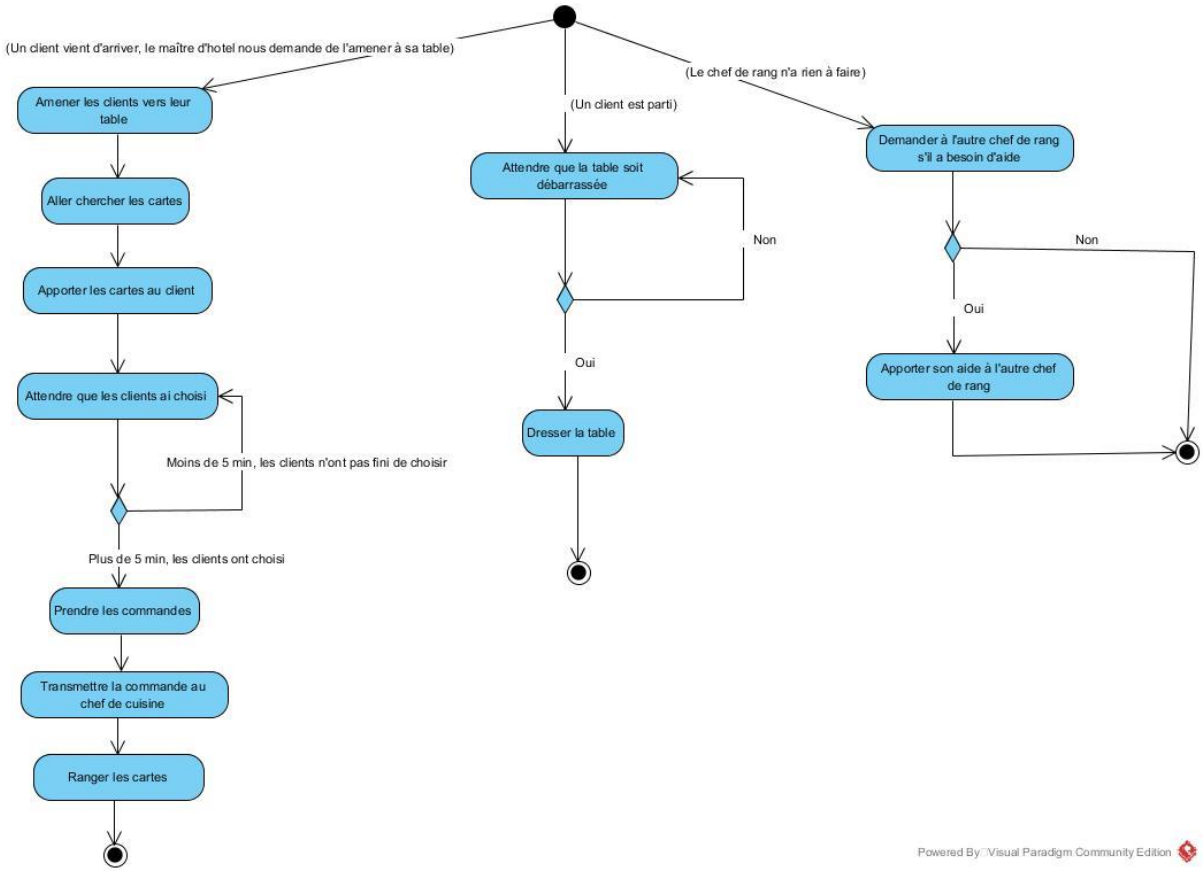
Le maître d'hôtel accueille les clients. Il peut y avoir 3 cas de figure :

- 1 : Le client arrive pour manger.
  Si le client est arrivé en dehors des horaires d'ouverture, celui-ci n'est pas accepté, sinon, le maître d'hôtel lui demande s'il a une réservation. Si c'est le cas, il assigne la table

correspondante au client et transfère les infos au chef de rang. Si le client n'a pas réservé, le maître d'hôtel regarde si une table peut correspondre (nombre de places supérieures ou égales au nombre de clients). Si aucune table n'est disponible, le client doit patienter, le temps qu'une table se libère. Puis le maître d'hôtel assigne la table correspondante au client et transfère les infos au chef de rang.

- 2 : Le client arrive pour réserver.
  Le maître d'hôtel demande au client le nombre de personnes et la date pour la réservation. Il vérifie ensuite les disponibilités, assigne une table s'il y a des disponibilités qui conviennent ou propose au client de changer leur demande si aucune table ne convient.
- 3 : Le client arrive pour payer.
  Le maître d'hôtel encaisse les clients.

Chef de rang



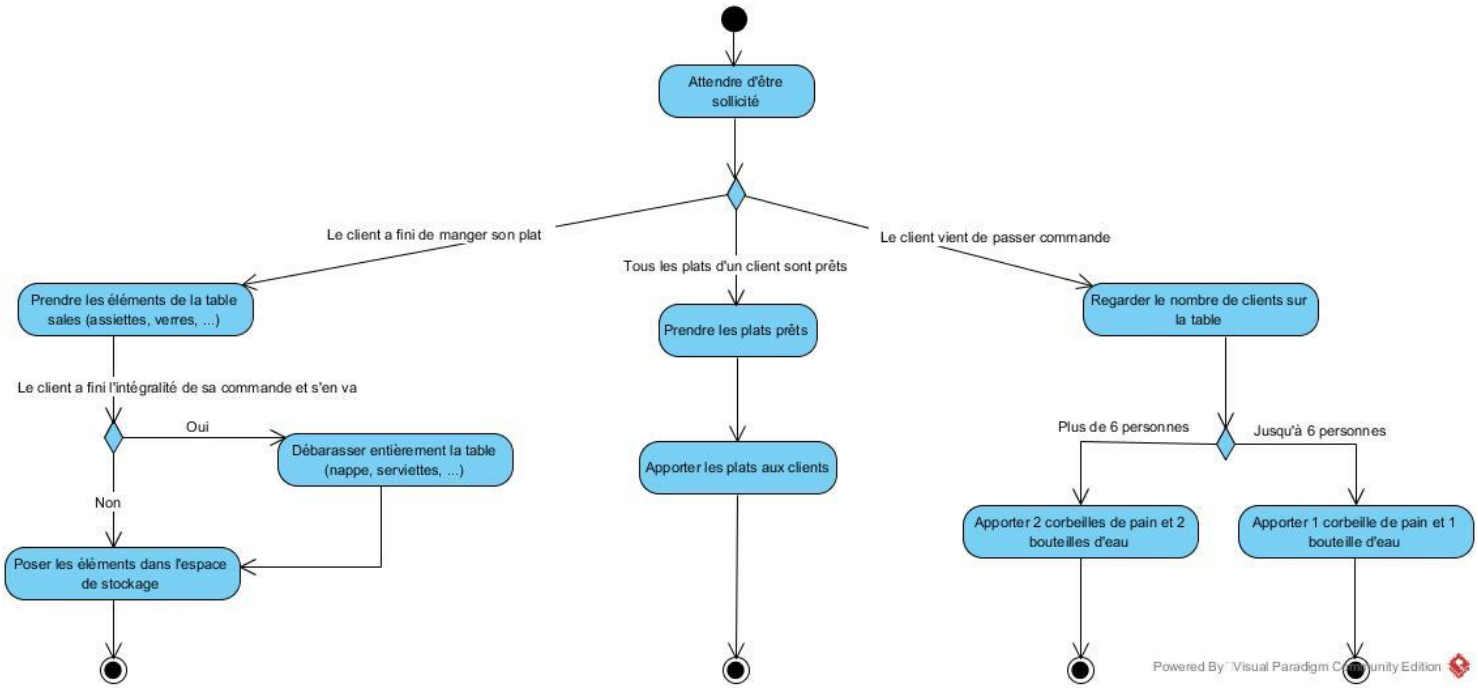
Le chef de rang possède 3 comportements possibles :

- 1 : Un client vient d'arriver, le maître d'hôtel demande au chef de rang de l'amener à sa table. Le chef de rang amène le client à sa table, va chercher les cartes et les apporte au client puis attend que le client choisisse les plats (il peut effectuer autre chose en attendant). Puis il prend les commandes et les transmet au chef de cuisine, puis range les cartes.
- 2 : Un client est parti.

Le chef de rang attend que le serveur ait débarrassé la table avant de dresser de nouveau la table.

- 3 : Le chef de rang n'a rien à faire (si les 2 autres cas ne sont pas réalisables).
  Le chef de rang apporte son aide à l'autre chef de rang s'il a besoin de son aide.

Serveur



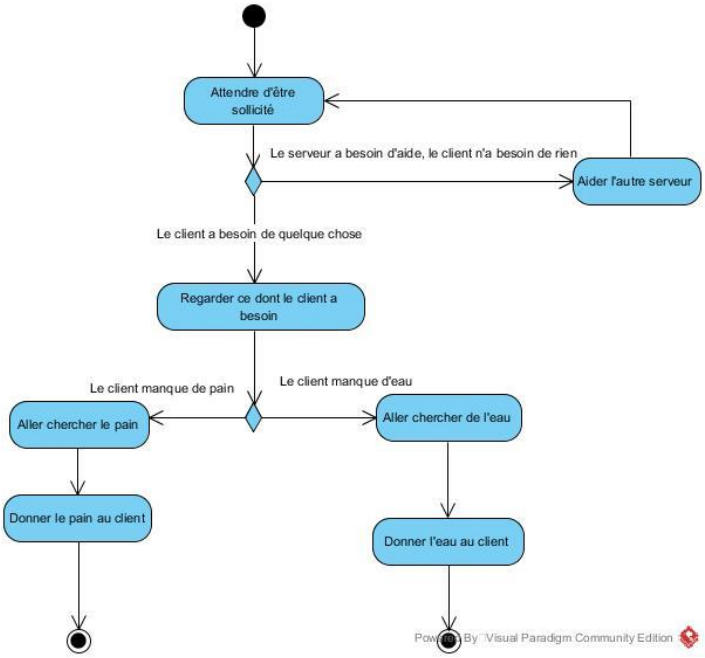
Le serveur attend d'être sollicité.

Si le client a fini de manger, le serveur débarrasse la table des éléments sales (assiettes, couverts, …). Si le client a fini l'intégralité de sa commande et s'en va, alors le serveur débarrasse l'intégralité de la table (serviette et nappes incluse). Il pose tous les éléments récupérés dans l'espace de stockage.

Le serveur intervient également si tous les plats d'une table sont prêts à être amené aux clients, il les apporte donc à la table correspondante.

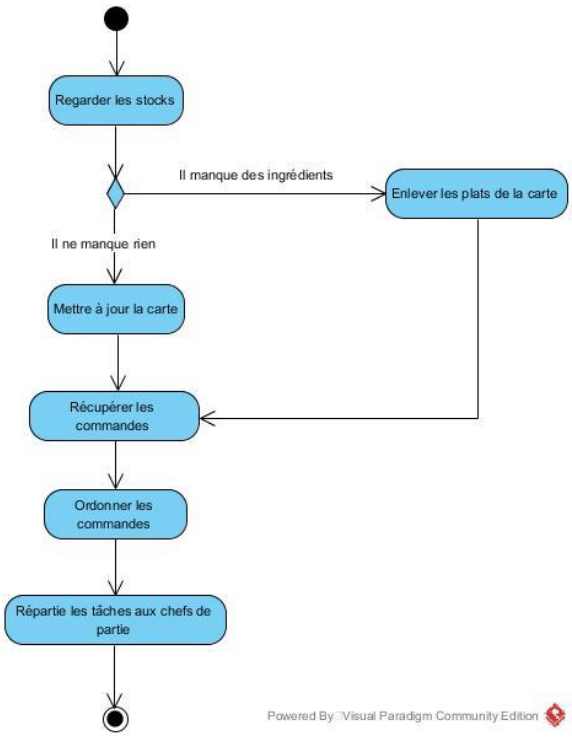
Si le client vient de passer sa commande, le serveur apporte à leur table une corbeille de pains et une bouteille d'eau (2 de chaque si le nombre de personnes sur la table est supérieur à 6).

Commis de salle



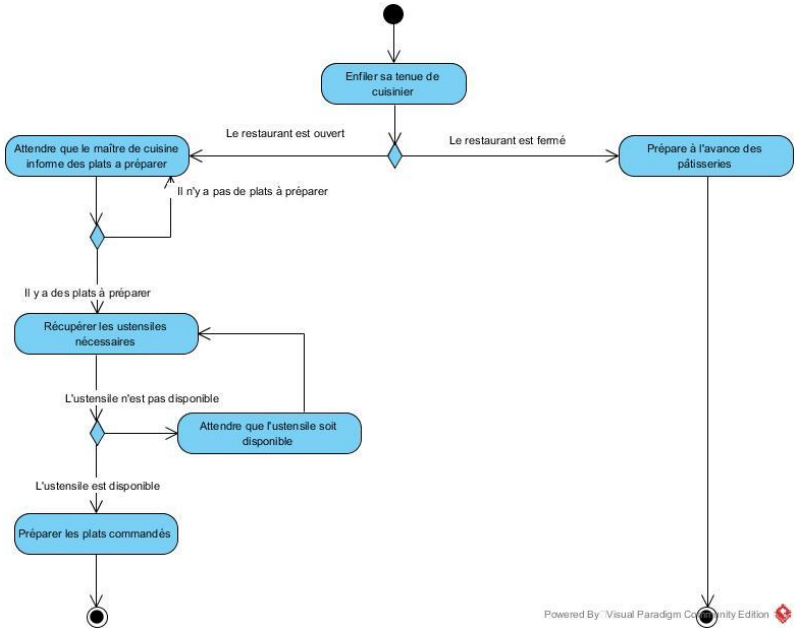
Le commis de salle attend d'être sollicité. Il y a 2 cas de figure : soit le serveur a besoin d'aide et le client n'a besoin de rien, dans ce cas le commis de salle va aider le serveur, soit le client a besoin de quelque chose, dans ce cas le commis de salle regarde les besoins du client et apporte en conséquence du pain ou de l'eau.

Chef de cuisine



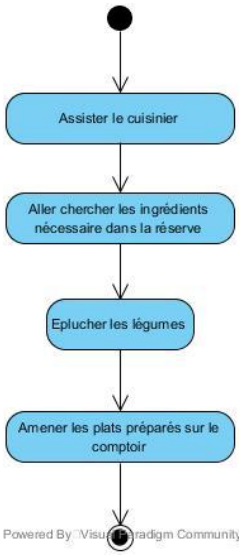
Le chef de cuisine se charge de surveiller les stocks. Si un plat de la carte ne peut plus être réalisé à cause d'un stock insuffisant, alors le chef de cuisine enlève ce plat de la carte. Il récupère aussi les commandes, les ordonne et répartit les tâches aux chefs de partie.

Cuisinier



Le cuisinier agit différemment selon l'heure. Si le restaurant est fermé, le cuisinier prépare des pâtisseries en amont de son service. Si le restaurant est ouvert, le cuisinier attend que le maître de cuisine l'informe des plats à préparer, récupère les ustensiles de cuisine qui lui sont nécessaires à la réalisation du plat. Si ces ustensiles ne sont pas disponibles, il attend qu'ils le soient puis prépare le plat.

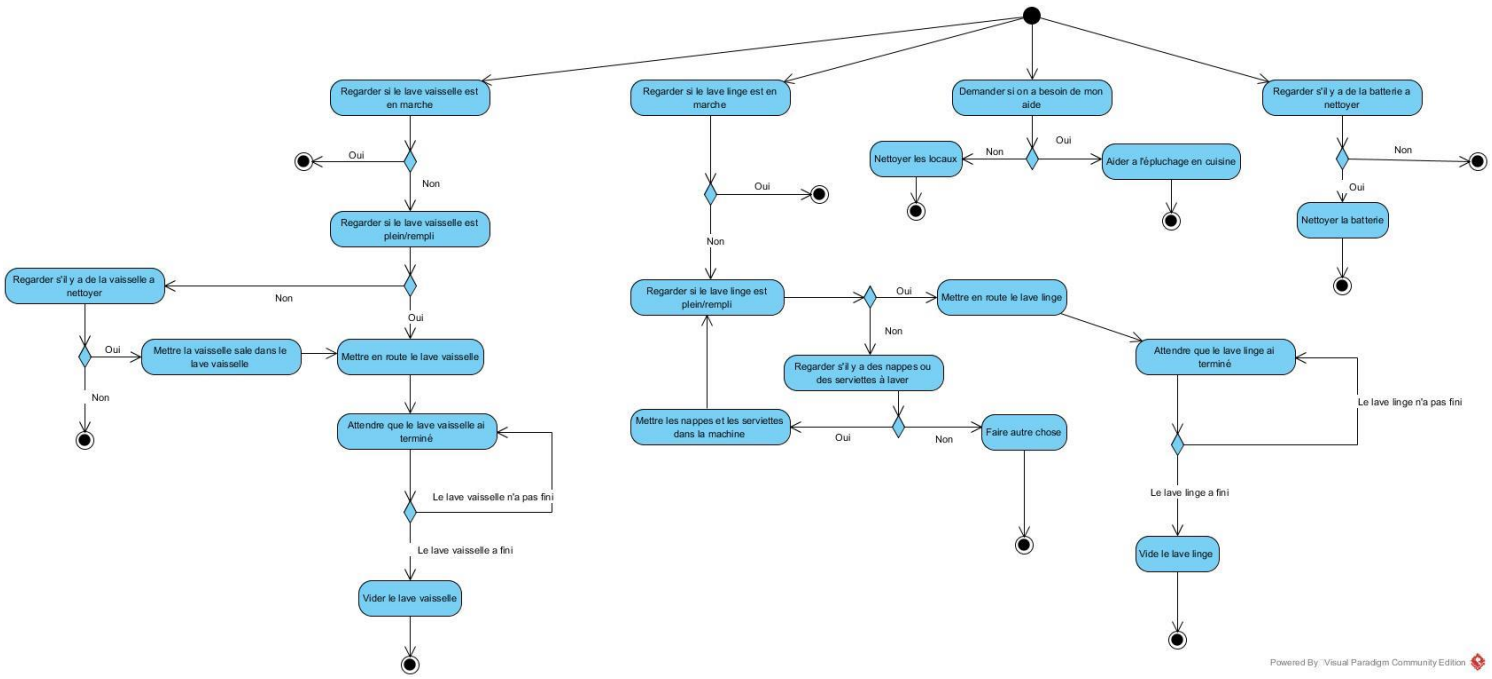
Commis de cuisine



Le commis de cuisine assiste le cuisiner dans son travail. Il va chercher les ingrédients nécessaires pour la réalisation du plat dans la réserve.

Il épluche également les légumes puis amène les plats préparés sur le comptoir afin qu'ils puissent être amenés aux clients.

Plongeur



Le plongeur possède 4 trames d'actions :

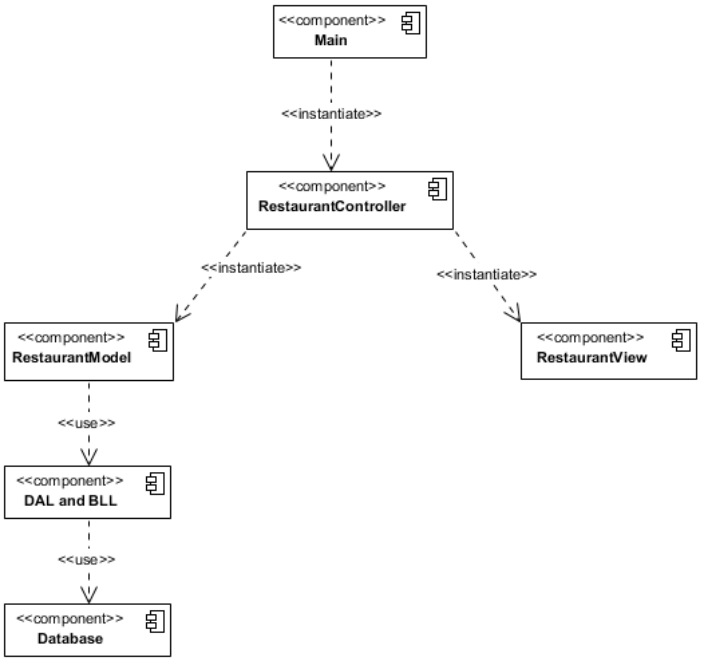
- 1 : Le plongeur s'occupe du lave-vaisselle.
  Si le lave-vaisselle est en marche, alors le plongeur effectue une autre tâche, sinon, il regarde si le lave-vaisselle est rempli ou non. Si c'est le cas, il le met en route, sinon il regarde s'il y a de la vaisselle à nettoyer. S'il n'y a rien à laver, alors le plongeur effectue autre chose, sinon il met les éléments dans le lave-vaisselle, puis le met en route. Une fois le cycle de lavage terminé, le plongeur vide le lave-vaisselle.
- 2 : Le plongeur s'occupe du lave-linge.
  Le plongeur agit de la même manière que pour le lave-vaisselle (voir 1.), la seule différence est que le lave-linge n'est pas mis en route tant qu'il n'y a pas un minimum de chose à laver chargé à l'intérieur (minimum de 10 pièces), au contraire du lave-vaisselle, qui peut être mis en route avec seulement 1 éléments.
- 3 : Le plongeur apporte son aide.
  Le plongeur demande en cuisine si on a besoin de son aide. Si c'est le cas, il aide en coupant des légumes ou en les épluchant, sinon il nettoie les locaux.
- 4 : Le plongeur nettoie la batterie.
  S'il y a de la batterie à nettoyer, le plongeur s'en charge, sinon il fait autre chose.

d. Diagramme de composant

Le diagramme de composant va nous servir à décrire les différents composants de notre système ainsi que leurs dépendances. Un composant va être un élément physique, c'est-à-dire qu'il va s'agir d'un élément de code. Il va représenter une partie implémentée de notre système.

Ainsi, dans notre diagramme de composant, nous allons avoir un total de 6 composants :

- Main
- RestaurantController
- RestaurantView
- RestaurantModel
- DAL and BLL (Data Access Layer et Business Logic Layer)
- Database

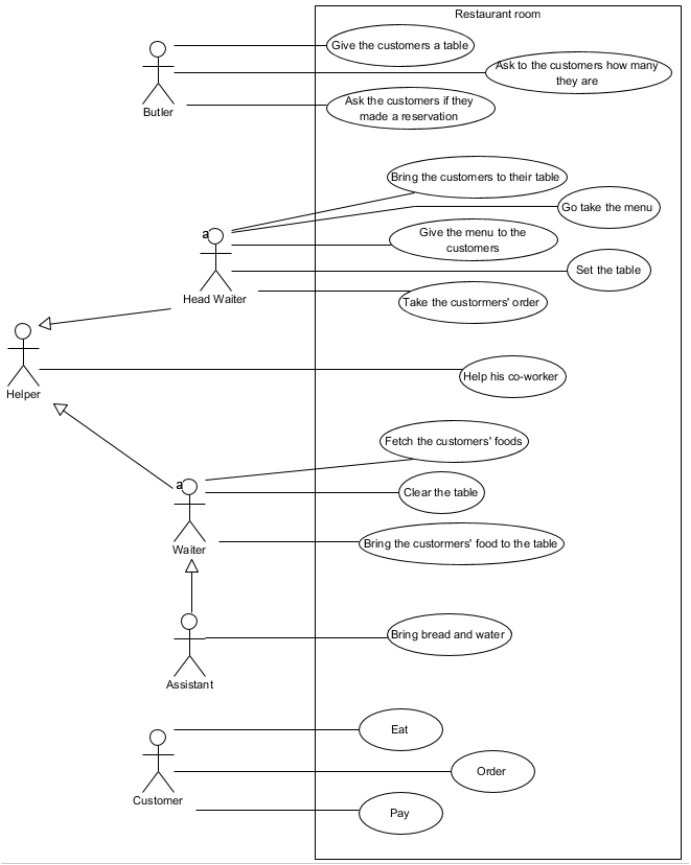


Nous avons donc notre composant Main qui va instancier notre RestaurantController qui est le contrôleur de notre programme. Le RestaurantController instancie ensuite RestaurantView, qui est notre vue, et RestaurantModel, qui est notre modèle. Puis nous allons avoir RestaurantModel qui va utiliser le composant DAL and BLL pour communiquer avec la base de données. Il utilise donc la base de données pour récolter les informations dont le modèle va avoir besoin.

e. Diagramme de cas d'utilisation

Pour pouvoir comprendre le comportement de chacun des acteurs de notre simulation, nous allons utiliser le diagramme de cas d'utilisation. Il va nous servir à capturer le comportement de notre système. Il nous donne la possibilité de fractionner les fonctions de notre simulateur en unités cohérentes qui vont être les cas d'utilisation. Ces derniers vont avoir un sens pour les acteurs. Ainsi, un cas d'utilisation décrit une interaction entre un acteur, qui dans notre cas est l'un des métiers de la simulation ou le client, avec le reste du système, donc la salle de restauration ou la cuisine.

Ainsi, nous allons avoir deux diagrammes de cas d'utilisation : le premier représente la situation dans la salle de restauration et le second représente la situation dans la cuisine.



Nous avons un total de 5 acteurs. Tout d'abord, notre maître d'hôtel (Butler) accueille les clients en leur souhaitant la bonne journée, il leur demande combien ils sont puis leur attribut une table. Il peut également leur demander s'ils ont une réservation.

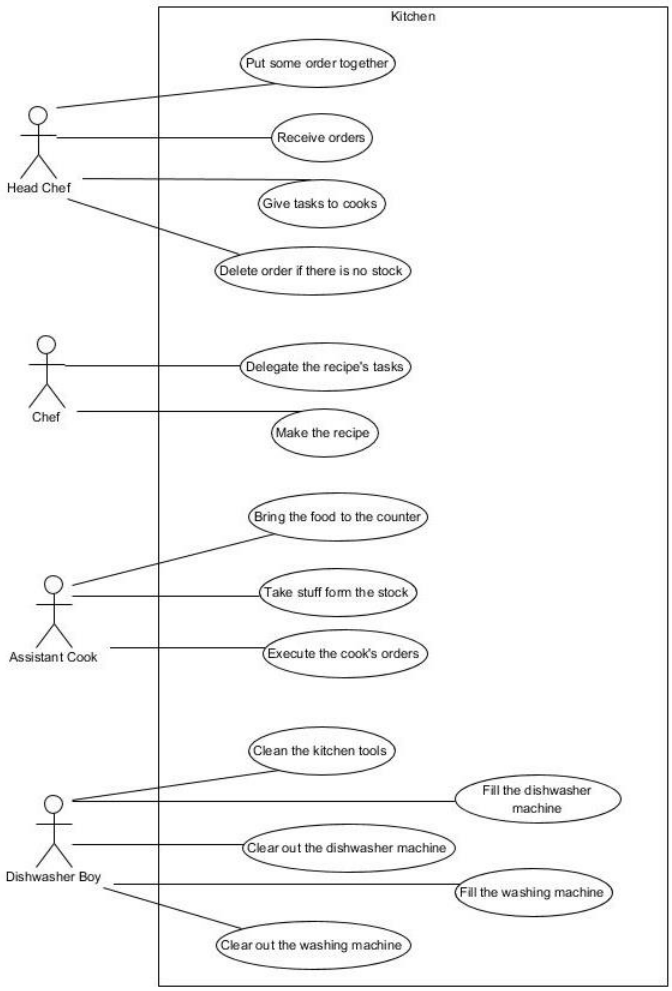
Le chef de rang (Head Waiter) amène les clients jusqu'à leur table, il va chercher les menus, les donne aux clients et enfin prend leur commande.

Le serveur (Waiter) va apporter les plats aux clients, leur apporte du pain et de l'eau et nettoie les tables.

Dans le cas où ils n'auraient rien à faire, le chef de rang et le serveur peuvent aller aider leurs collègues. Nous avons donc placé le cas d'utilisation « aider ses collègues ». Pour éviter de surcharger le diagramme en relier le cas à deux acteurs, nous avons fait le choix de créer un acteur intermédiaire « Renfort » (Helper) dont nos deux acteurs vont hériter.

Nous avons ensuite le Commis de rang (Assistant) qui va apporter l'eau et le pain sur les tables. Il peut aussi remplacer un serveur. Donc nous le faisons hériter de l'acteur Serveur pour qu'il est les mêmes cas d'utilisation.

Enfin, il y a le client (Customer) qui va pouvoir commander, manger et payer.



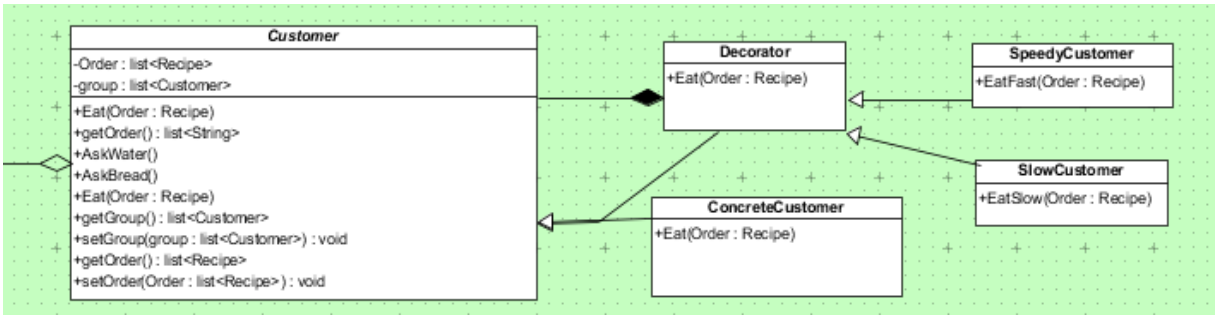
Pour le deuxième diagramme de cas d'utilisation, nous l'appliquons à la cuisine. Nous avons d'abord le chef de cuisine (Head Chef) qui va recevoir les commandes, assembler des commandes ensemble si besoin, distribuer des tâches aux cuisiniers et enfin supprimer la préparation d'un plat si un ingrédient n'est plus en stock.
Puis, vient l'acteur cuisinier (Chef) qui va être chargé de déléguer des parties de la recette et de réaliser la recherche.
L'acteur Commis de cuisine (Assistant Cook) va aller chercher les ingrédients dans le stock, réaliser les tâches qui lui sont données par le cuisinier et apporter les plats qui sont prêt au comptoir.
Nous avons enfin l'acteur Plongeur (Dishwasher Boy) qui va nettoyer les outils de cuisines, remplir puis ranger le lave-vaisselle, remplir puis ranger la machine à laver.

II. Les Design Patterns:

a. Decorator

Ce design pattern de structure, permet d'attacher dynamiquement des responsabilités à un objet.

Grâce à ce design pattern, nous pourrons attacher à chaque client un type pressé ou lent. Cela influera sur son comportement ; sa vitesse pour manger.



b. Observer

Le Design Pattern Observateur (de type Comportement) 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.

Dans notre contexte ce patron de conception permettra :

- Notifier les serveurs (waiter) lorsque les plats d'une table seront terminés.
- Notifier le plongeur (dishwasher boy) lorsqu'un nombre conséquent d'assiettes sales, de couverts sales, de nappes sales (ou autres…) sont présents sur le comptoir.
- Mettre à jour la vue selon les différents changements

c. Singleton

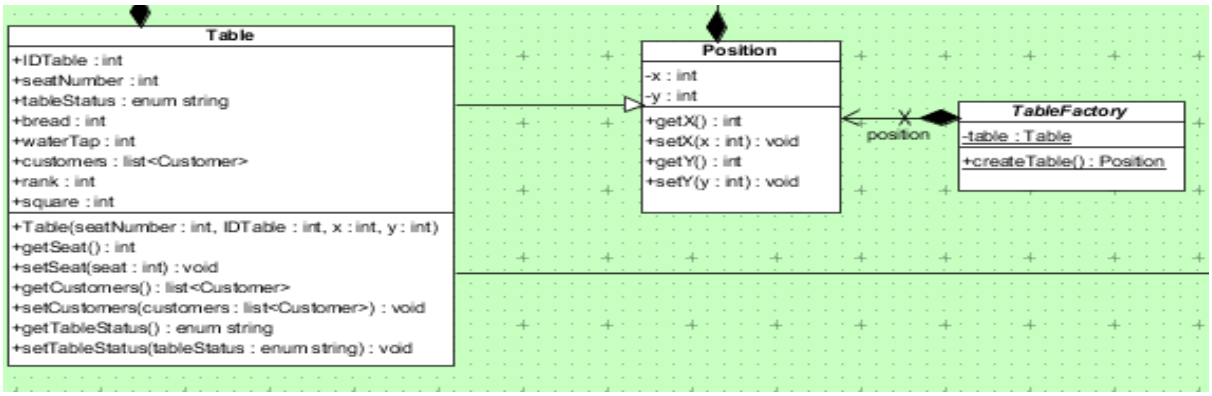
Le Singleton (Création) garantit qu'une classe n'a qu'une seule instance et fournit un point d'accès global à cette instance.

Le Maitre d'hôtel (Butler) et le chef de cuisine (head chef) seront en singleton, pour qu'il ne soit pas instancié plusieurs fois, c'est-à-dire appelé une nouvelle fois alors que ceux-ci sont normalement occupé.

d. Factory

Ce design pattern de création, est un modèle de conception qui fournit une interface pour créer des objets dans une superclasse, mais permet aux sous-classes de modifier le type d'objets qui seront créés.

Dans notre contexte, nous avons créé une usine (factory) de table pour créer toutes les tables de la salle de restauration, avec pour chacune des tables leurs spécificités (id, nombre de chaise…).



e. Strategy

Le design pattern Strategy est un design pattern de conception. Strategy nous permet de sélectionner des algorithmes au cours du temps d'exécution selon certaines conditions. Ce Design Pattern est souvent utilisé dans des cas où il va être nécessaire de permuter dynamiquement des algorithmes utilisés à l'intérieur d'une application. Ainsi, dans notre cas, nous utilisons Strategy pour nos éléments mobiles. Le but est que chaque élément que nous considérons comme mobile puis avoir des méthodes pour se déplacer vers l'accueil, le comptoir ou une table.

f. Facade

Ce design pattern est un design pattern de structure et nous donne la possibilité de rendre plus simple une partie de notre code en utilisant une interface.

Ainsi, dans notre cas, nous utilisons un design pattern Facade pour l'accès à notre Data Access Layer et Business Logique Layer qui est la partie de notre code qui va nous permettre d'échanger des informations avec notre base de données. Il s'agit donc d'une partie complexe et nous cherchons à la simplifier avec l'utilisation de la Facade et ces méthodes.

g. MVC

Le design pattern MVC pour Model – Vue – Contrôleur est un design pattern d'architecture logicielle composé de trois types de modules avec pour chacun une responsabilité différente :

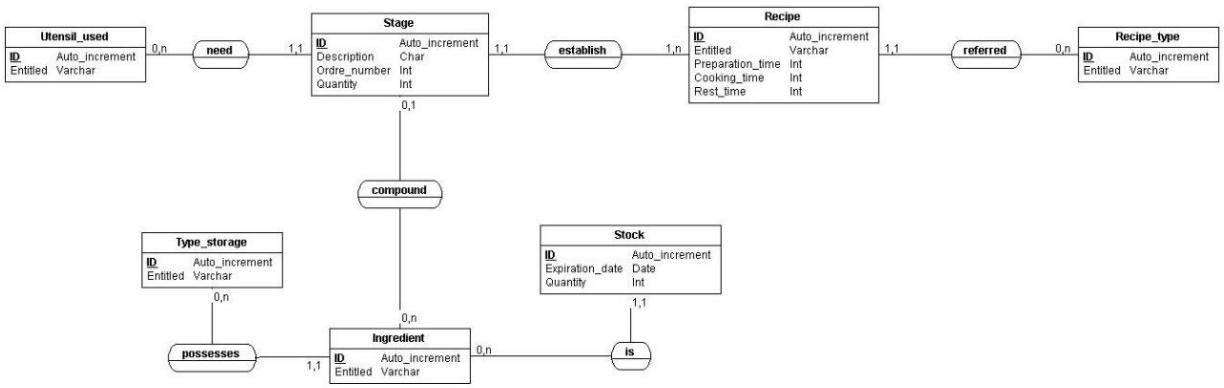
- Le contrôleur contient la logique concernant les actions à effectuer.
- Le modèle contient les données à afficher.
- La vue contient la présentation de l'interface graphique.

Pour notre application, nous avons le modèle, RestaurantModel, qui va contenir toutes les informations ainsi que les traitements des informations de notre simulation. Il va aussi être la partie qui contient les classes qui vont nous permettre de nous connecter à la base de données. Ensuite, nous avons le contrôleur, RestaurantControler, qui va se charger de mettre à jour la vue en faisant transiter des informations de la vue au modèle et inversement. Il va également s'agir de la partie qui va nous permettre de gérer les paramètres de l'applications. Et enfin nous avons la vue, RestaurantView, qui va afficher notre interface.

III. MCD :

Le Modèle Conceptuel des Données a pour but de représenter les données de manière compréhensible et permet de décrire le système d'informatique à l'aide d'entités.

Voici notre MCD :



Nous avons une entité *Recipe* qui regroupe toutes les recettes, avec un nom et le temps de préparation, de cuisson et de repos.

Cette recette possède un type (entité *Recipe_type*) qui est composé d'un intitulé.

Cette recette possède également des étapes (entité *Stage*) qui sont composés d'une description (c'est-à-dire les instructions à suivre pour réaliser la recette), un numéro ordre pour connaître l'ordre d'exécution de chaque étape ainsi qu'une quantité. Il s'agit de la quantité de l'ingrédient nécessaire pour cette étape.

Chaque étape utilise un ustensile de cuisine, nous avons donc une entité *Utensil_used* avec un intitulé.

L'entité *Ingredient* contient l'intitulé de l'ingrédient. Nous avons aussi l'entité *Stock*, avec pour chaque ingrédient, la date d'expiration et la quantité d'ingrédient.

Type_storage possède un intitulé. Cette entité permet de défini dans quel type de stockage figure les différents ingrédients. En effet, certains ingrédients sont stockés en le congélateur par exemple.