PROJET MASTERCHEF INFO

Dossier de modélisation

Agathe Lagache – Lucas Lartot-Da Luz Rijo
Axel Navarrot-Lavigne – Kévin Bundhun

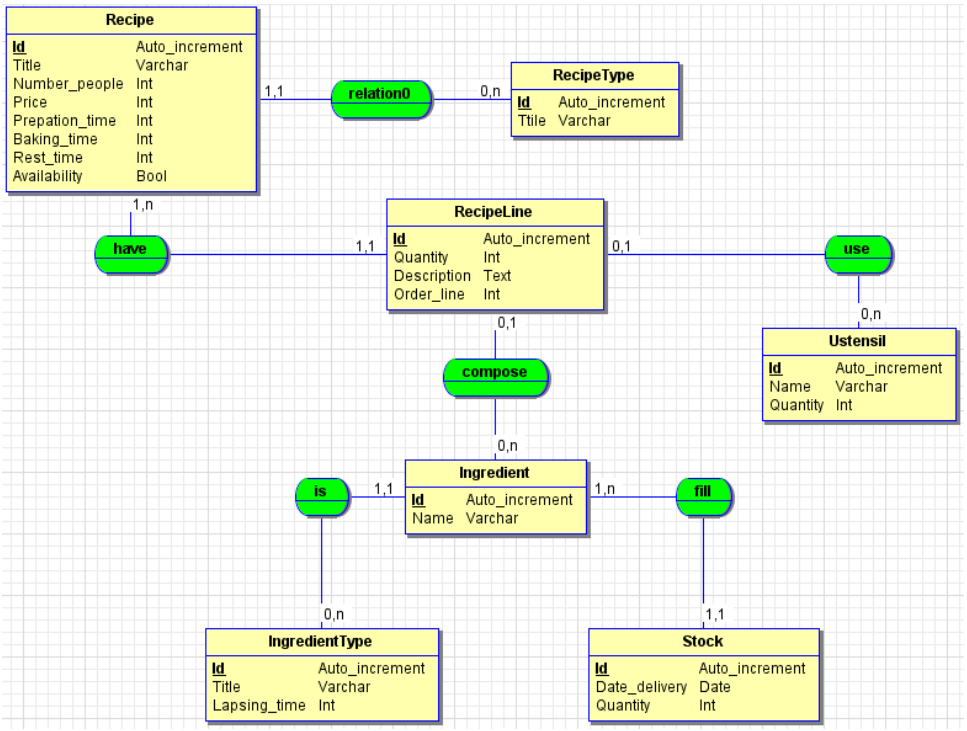
Table des matières

I. Modèle Conceptuel des Données

Dans le cadre de notre projet « Programmation système (.NET) », nous devions modéliser et construire une base de données afin de pouvoir stocker de multiples informations pour notre application.

L'indication qui nous était donné dans le sujet pour orienter la création de notre base de données était la suivante : « Le stock devra être géré avec une base de données et devra être actualisé en temps-réel tout au long du service. Pour une certaine simplification, vous pouvez calculer le stock par pièce et non par kilos (par exemple, 10 bavettes et non 1,5Kg de viande à bavette ; 3 pommes et non 1Kg de pommes) ».

Voici donc ci-dessous le MCD de la base de données que nous avons conçu :



Cela nécessite néanmoins quelques explications. Comme spécifié dans l'énoncé, nous avons créé une table « **Stock** » permettant de répertorier tous les ingrédients que possède le restaurant (**ATTENTION :** ici n'est géré que le stock « alimentaire »). Cette table « **Stock** » contient donc :

- Un **ID** (unique – clé primaire) ;
- Une **date de livraison** qui va nous permettre de savoir depuis quand est conservé un ingrédient ;
- Une **quantité** qui est la quantité livrée à chaque livraison.

Cette table est liée à la table « **Ingredient** » afin de pouvoir récupérer l'ID des ingrédients qui sont dans le stock. Cette dernière contient uniquement deux champs :

- Un **ID** (unique – clé primaire) ;
- Un **nom** (unique).

Cependant, un ingrédient est caractérisé par son type de conservation : la table « **Type** » assure cet aspect. La table « **Ingredient** » possède donc un nouveau champ « **Id_Type** ». En effet, elle contient :

- Un **ID** (unique – clé primaire) :
- **L'intitulé** (unique) qui correspondra aux types de conservation comme « basique », « frais » ou « surgelé » ;
- Le **temps** avant péremption.

Nous avons ensuite pris la décision de stocker aussi en base les différentes recettes disponibles dans le restaurant. Nous avons donc créé une table « Recipe » contenant :

- Un **ID** (unique – clé primaire) ;
- Le **titre** (unique) ;
- Le **nombre de personnes** ;
- Le **prix** ;
- Le **temps de préparation** ;
- Le **temps de cuisson** ;
- Le **temps de repos** ;
- La **disponibilité** de la recette au sein du restaurant.

Une recette est caractérisée par un type (entrée, plat, dessert). La création d'une table « **RecipeType** » était donc indispensable. Cette table possède uniquement :

- Un **ID** (unique – clé primaire) ;
- Le **titre** (unique) ;

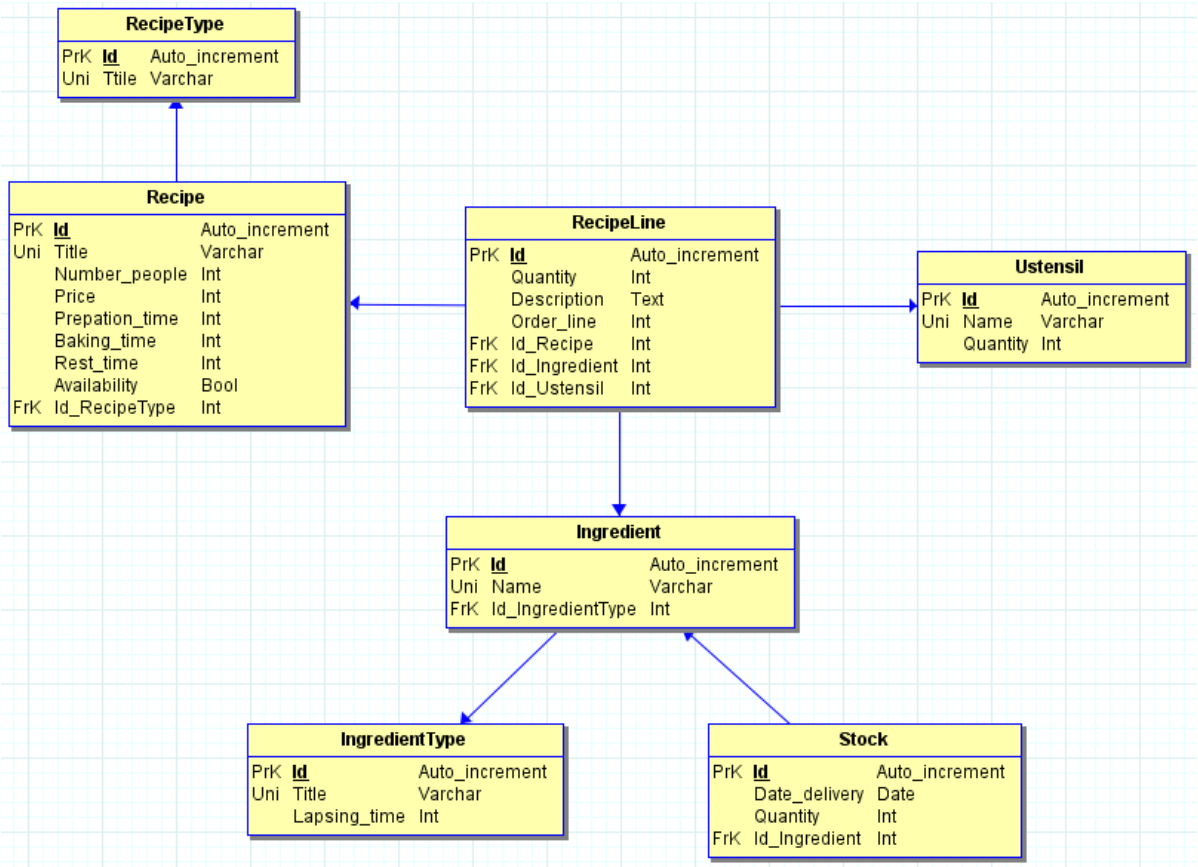
Or, une recette est composée de plusieurs ingrédients, donc de plusieurs lignes de recette, d'où la présence de la table « **RecipeLine** » qui va répertorier toutes les lignes recettes de toutes les recettes du restaurant. Elle contient donc :

- Un **ID** (unique – clé primaire) ;
- La **quantité** d'un ingrédient (ingrédient indiqué via son ID étant donné que cette table est aussi liée à la table « **Ingredient** ») ;
- La **description** (ce qui doit être fait) ;
- **L'ordre de la ligne** dans la recette.

Enfin, pour réaliser une recette, on a aussi besoin d'ustensiles de cuisine. Donc, nous avons créé une table « **Ustensil** », liée à la table « **RecipeLine** » contenant :

- Un **ID** (unique – clé primaire) ;
- Le **nom** (unique) ;
- La **quantité** de ces ustensiles dans la cuisine.

Ci-dessous, voici le MLD (Modèle Logique des Données) découlant de la conception du MCD.



II. Diagrammes

1. Diagramme de cas d'utilisation

Nous devions réaliser un diagramme de cas d'utilisation (en anglais, use case diagram) nous permettant de schématiser les différentes actions de chaque acteur au sein de notre application. Pour construire ce diagramme, nous avons donc commencé par séparer les types d'utilisateurs :

- Le personnel ; ce type d'utilisateur est aussi divisé en en sous-type en fonction de leur emplacement dans le restaurant : « Salle » ou « Cuisine » ;
- Les clients.

Cependant, ce n'est pas si simple. Chaque salle comporte 4 rôles que nous avons donc bien distingué dans notre diagramme.

Chacun des rôles possède des actions précises à exécuter : ces actions sont aussi réparties entre deux « systèmes » qui sont les différentes parties du restaurant, c'est-à-dire la cuisine ainsi que la salle de restauration.

Voici la liste des actions que nous avons répertorié en fonction de l'acteur et du lieu :

Salle de restaurant :

- **Maître d'hôtel :**
  - Accueillir ;
  - Appeler le chef de rang ;
  - Assigner une table ;
  - Encaisser.
- **Chef de rang :**
  - Superviser / coordonner les actions du commis ;
  - Distribuer les cartes ;
  - Amener le client ;
  - Prendre les commandes ;
  - Dresser la table ;
  - Se déplacer.
- **Serveur :**
  - Se déplacer ;
  - Servir ;
  - Débarrasser la table ;
  - Emmener la vaisselle en plonge.
- **Commis :**
  - Ce dernier possède les même comportements qu'un serveur. En effet, il peut le remplacer s'il n'est pas disponible. Donc, le commis hérite du serveur.

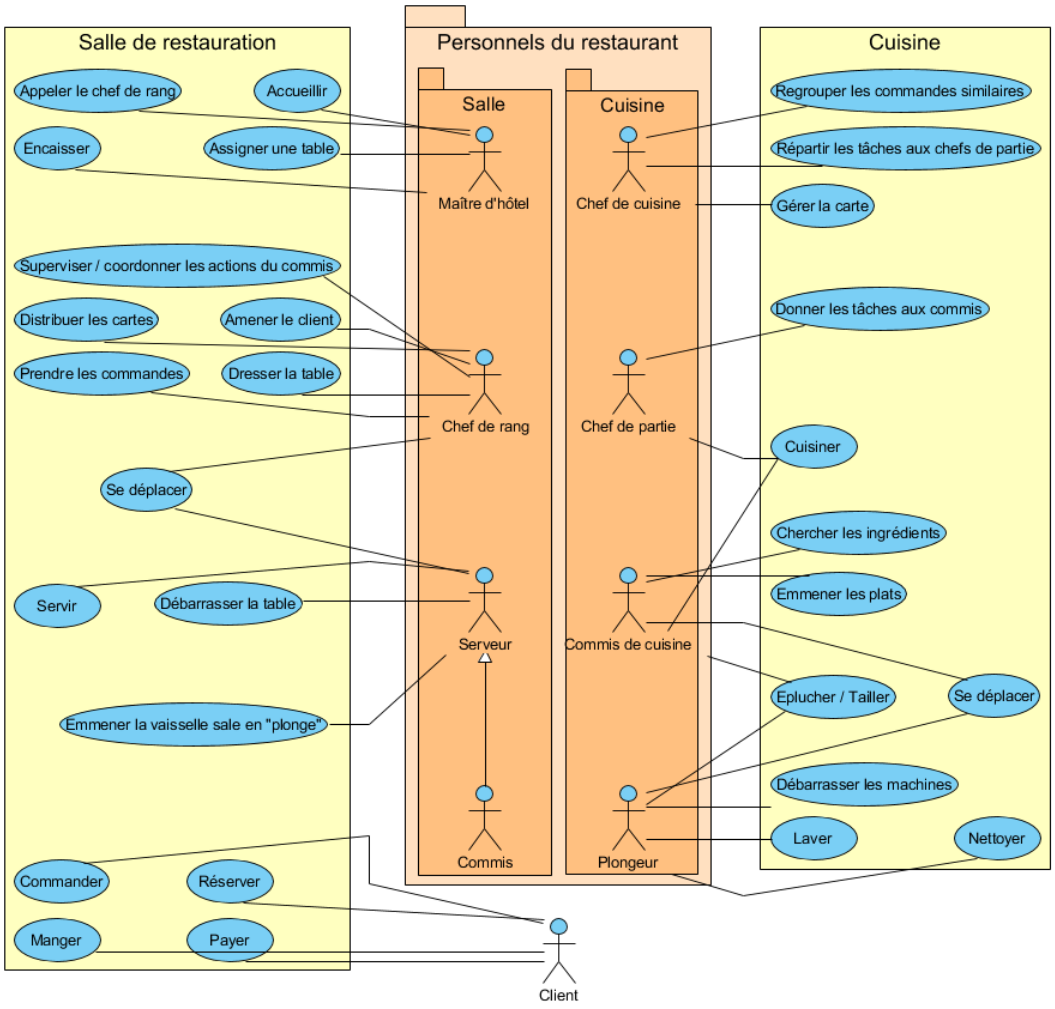
Cuisine :

- **Chef de cuisine :**
  - Regrouper les commandes similaires ;
  - Répartir les tâches ;
  - Gérer la carte.
- **Chef de partie / cuisinier :**
  - Donner les tâches aux commis ;
  - Cuisiner.
- **Commis de cuisine :**
  - Cuisiner ;
  - Chercher les ingrédients ;
  - Emmener les plats ;
  - Se déplacer ;
  - Eplucher / Tailler.
- **Plongeur :**
  - Eplucher / Tailler ;
  - Débarrasser les machines ;
  - Laver ;
  - Nettoyer.

Autre :

- **Client :**
  - Commander ;
  - Réserver ;
  - Manger ;
  - Payer.

Ce qui nous donne, de manière schématisée, ce diagramme de cas d'utilisation :

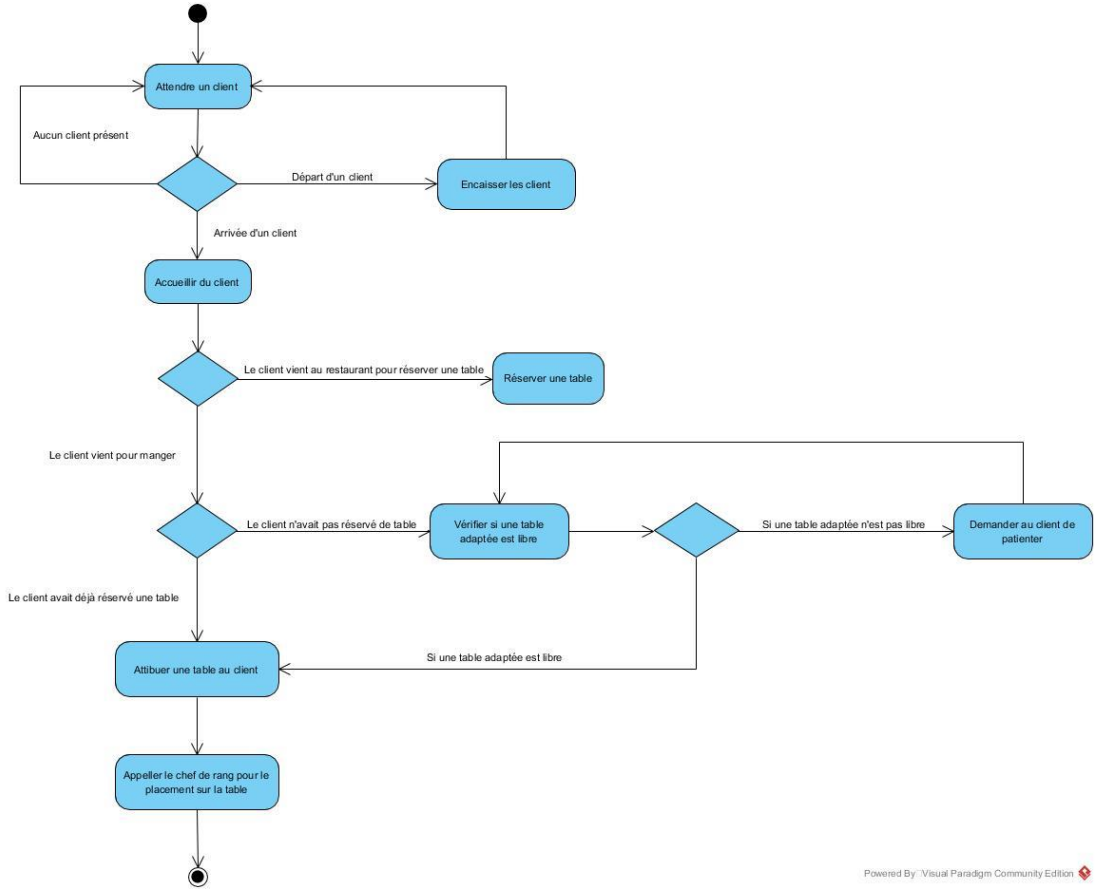


2. Diagrammes d'activité

Le diagramme d'activité est un diagramme décrivant le comportement et permettant de représenter le déclanchement d'évènements selon l'état du système étudié. Ce diagramme permet également de modéliser des comportements exécutés simultanément. Il nous permet également de décrire le flux de travail (workflow) de chaque poste de travail.

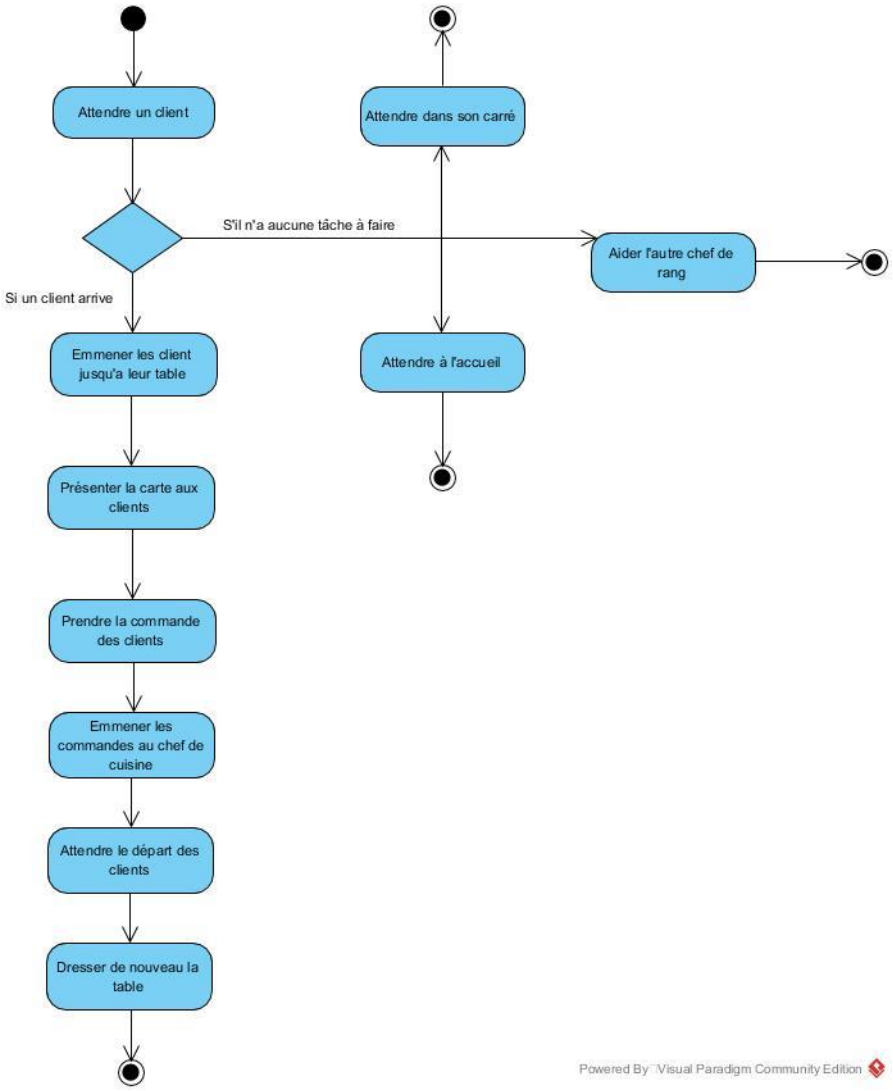
Voici en suivant les différents diagrammes d'activités demandés.

A. Diagramme d'activité du maître d'hôtel



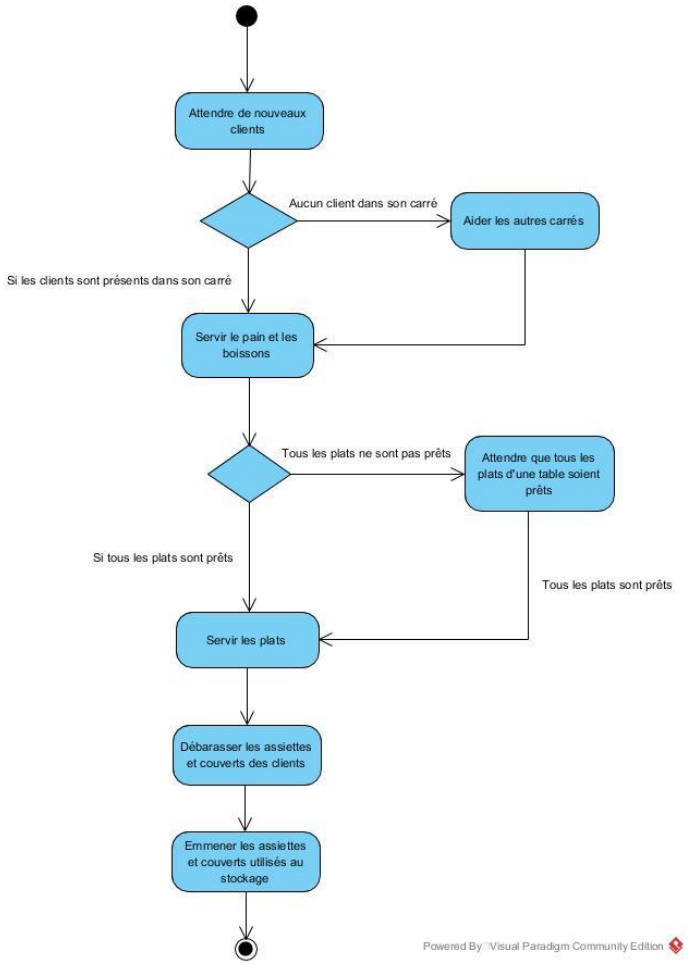
D'après le texte du projet : « Le maître d'hôtel attend de nouveaux clients. Lorsque des clients se présentent, il les accueille. Ensuite, si le client vient seulement pour réserver une table, il réserve la table au client. Si le client vient pour manger et qu'il avait réservé une table, le maître d'hôtel lui attribue une table et appelle le chef de rang pour aller le placer. Si le client vient pour manger mais qu'il n'a pas réservé de table, le maître d'hôtel vérifie si une table adaptée au nombre de client est disponible. Si cela est le cas, il appelle le chef de rang pour aller les placer. Mais si aucune table n'est disponible, il demande au client de patienter. »

B. Diagramme d'activité du chef de rang



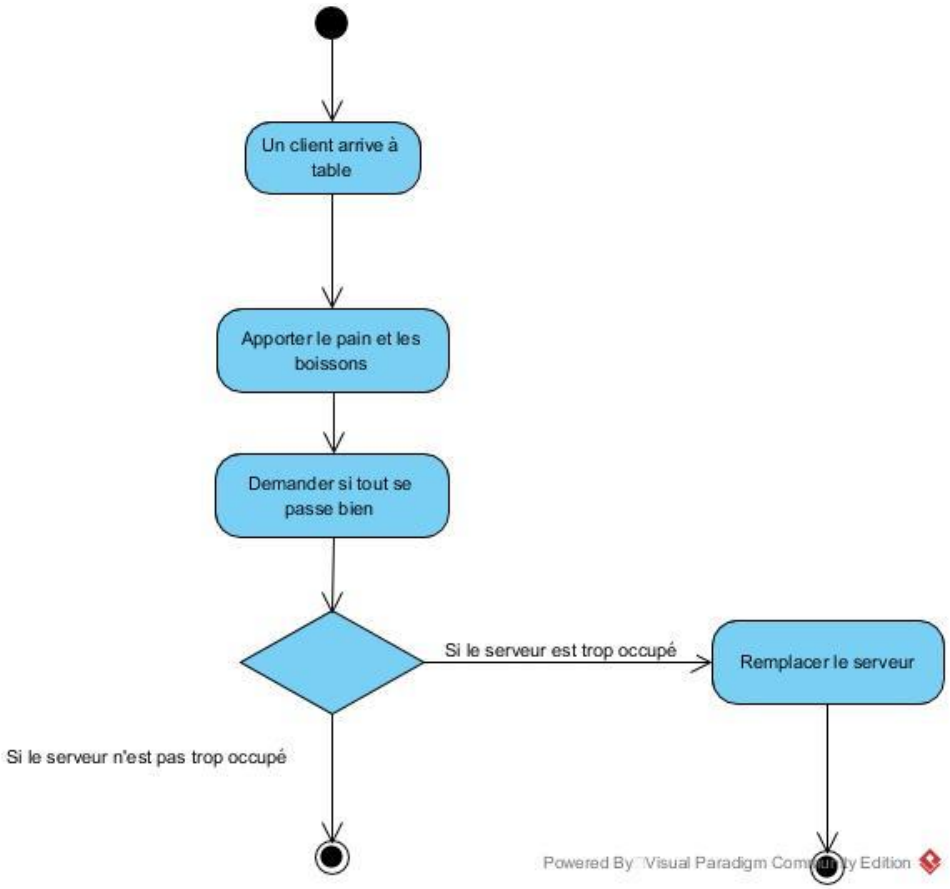
D'après le texte du projet : « Le chef de rang est responsable d'un certain nombre de tables, qui composent un rang. Lorsqu'un (des) client(s) arrive(nt), il l'emmène jusqu'à sa table, lui présente la carte et prend les commandes. Ensuite, il emmène les commandes des clients au chef de cuisine. Puis, il attend que les clients partent, lorsque cette situation arrive, il dresse une nouvelle table pour accueillir de nouveaux clients. Quand le chef de rang n'a pas d'occupation, il peut attendre dans son carré soit attendre à l'accueil, soit aider le chef de rang d'un autre carré. »

C. Diagramme d'activité du serveur



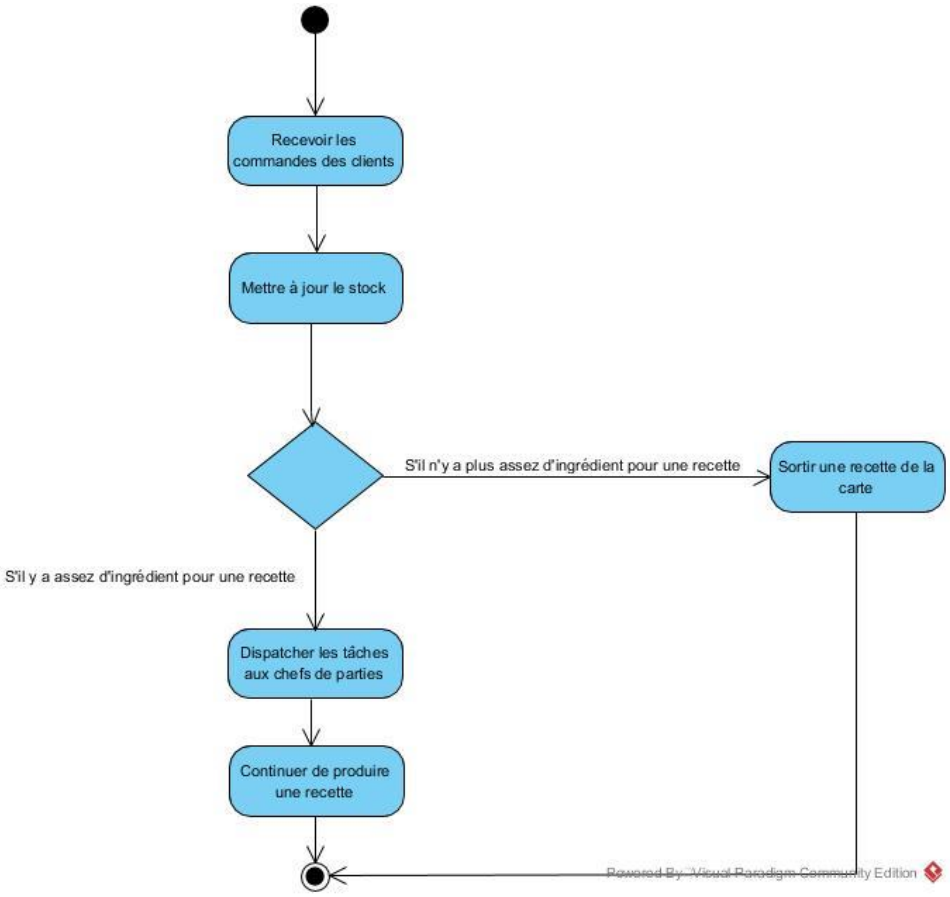
D'après le texte du projet : « Lorsqu'un client arrive et que sa commande est prise, le serveur met le pain et l'eau sur la table. Pour les tables de plus de 6 personnes, 2 corbeilles et 2 bouteilles. Ensuite, si tous les plats d'une table sont prêts, le serveur sert les plats à la tables correspondante. Lorsque les clients partent, le serveur débarrasse les couverts et les assiettes des clients et les emmène au stockage. Cependant, lorsqu'il n'y a pas de client dans son carré, il peut aider les serveurs des autres carrés. »

D. Diagramme d'activité du commis de salle



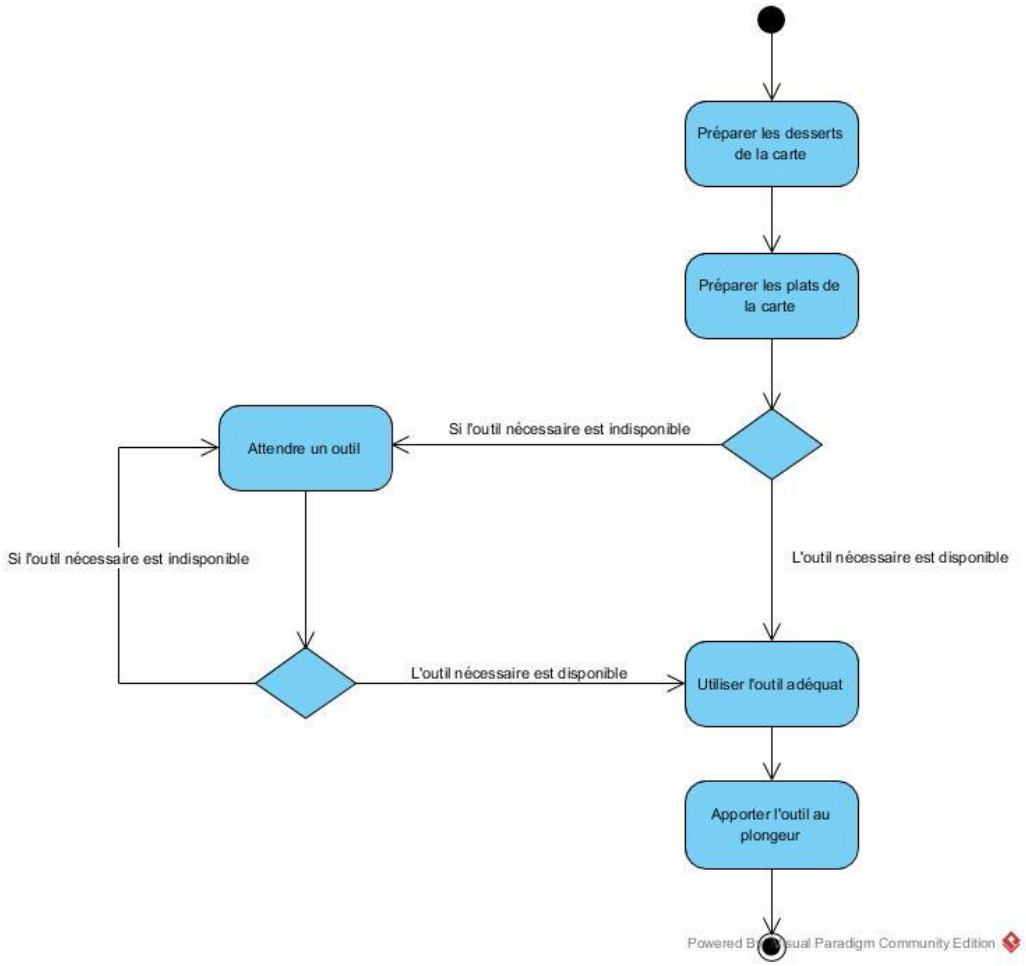
D'après le texte du projet : « Le commis de salle possède le comportement suivant. Lorsqu'un client arrive et qu'il est assis à la table qui lui est indiquée, le commis de salle apporte les boissons et le pain sur la table. Ensuite il doit s'assurer que tout se passe pour le mieux pour le client. Cependant, il peut avoir les mêmes fonctions du serveur si celui-ci est trop occupé. »

E. Diagramme d'activité du chef de cuisine



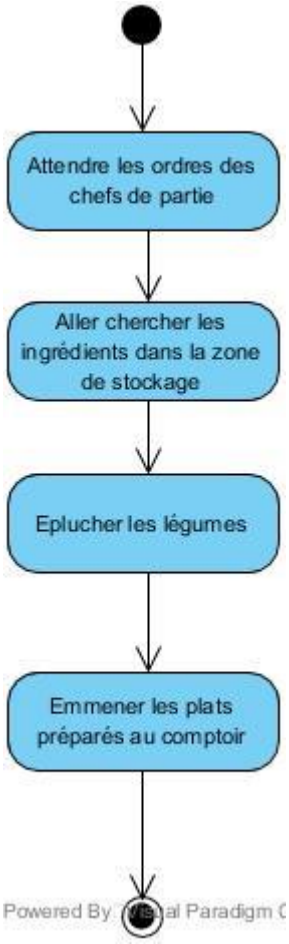
D'après le texte du projet : « Le chef de cuisine coordonne, dirige et supervise l'ensemble de l'activité de la cuisine. Il dispatche les tâches aux chefs de parties. De plus, il reçoit les commandes et son rôle est de les ordonner pour minimiser le temps d'attente du client et de dispatcher les tâches entre les chefs de parties/cuisiniers. Le chef de cuisine peut découper la recette en plusieurs sous-tâches en parallèle pour accélérer le service. Il gère également le stock d'aliments, lorsqu'il n'y a plus assez d'ingrédients, il doit enlever la recette de la carte. »

F. Diagramme d'activité du chef de partie/cuisinier



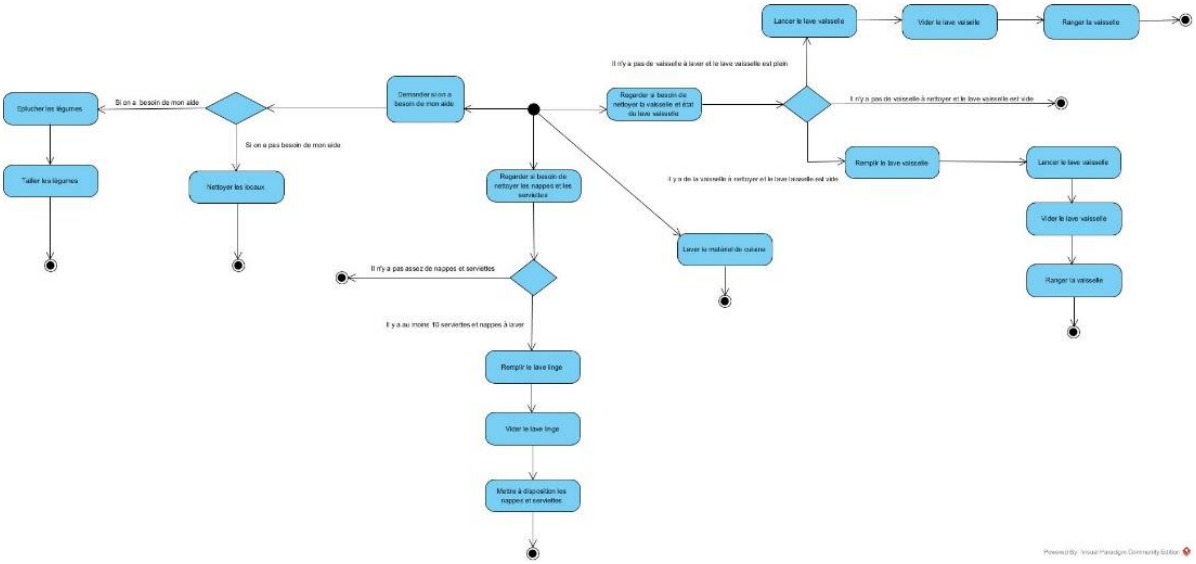
D'après le texte du projet : « Le chef de partie pâtisserie doit préparer, avant l'ouverture au public, une certaine quantité de desserts qui seront dans la carte. De plus il doit préparer les plats présents sur la carte. Cependant, il doit utiliser certains outils si ceux-ci sont disponibles. Lorsqu'ils le sont, le cuisinier peut l'utiliser et ensuite passer à l'étape suivante de la recette et apporter l'outil utilisé au plongeur afin de le nettoyer. »

G. Diagramme d'activité du commis de cuisine



D'après le texte du projet : « Le commis de cuisine possède la trame d'action suivante. En premier lieu, il attend les ordres des chefs de partie. Ensuite, il va chercher les ingrédients nécessaires à une recette dans la zone de stockage, puis éplucher les légumes. Et enfin, le commis de cuisine apporte les plats entièrement préparés au comptoir afin que les serveurs puissent les récupérer. »

H. Diagramme d'activité du plongeur



Le plongeur possède quatre scripts d'actions :

En effet, il doit laver le matériel de cuisine.

Le deuxième script démarre par la demande si les commis ont besoin d'aide. S'ils ont besoin d'aide, le plongeur doit aider à éplucher les légumes ainsi que les tailler. S'ils n'ont pas besoin d'aide, le plongeur doit nettoyer les locaux.

Dans le troisième script, le plongeur regarde s'il y a besoin de nettoyer la vaisselle ainsi que l'état du lave-vaisselle. S'il n'y a pas de vaisselle à nettoyer et que le lave-vaisselle est plein (au moins un objet), le plongeur met en marche le lave-vaisselle. Ensuite, il vide le lave-vaisselle et range la vaisselle. S'il n'y a pas de vaisselle à nettoyer et le lave-vaisselle est vide, il ne fait rien. Enfin s'il y a de la vaisselle à nettoyer et que le lave-vaisselle est vide, le plongeur remplit le lave-vaisselle, le met en marche, le vide et range la vaisselle.

Dans le dernier script, le plongeur regarde s'il y a besoin de nettoyer les nappes et les serviettes. Dans le cas où il n'y a pas assez de serviettes et/ou de nappes (moins de 10), il ne fait rien. Au contraire, s'il y en a assez, il remplit le lave-linge, il vide le lave-linge et enfin, il met à disposition les nappes et les serviettes.

3. Diagrammes de classe

Notre diagramme de classe indique toutes les classes, interfaces et relations de notre programme. Nous avons utilisé les Design Pattern Modèle/Vue/Contrôleur pour séparer notre code de manière plus simple et plus évolutive.

Dans la vue, nous retrouvons toutes les méthodes utiles à l'interface graphique, permettant de mettre en place celle-ci et de lancer le début du programme. Il s'agit uniquement d'une classe réalisant une interface, où les méthodes définissent la fenêtre.

METTRE IMAGE VUE

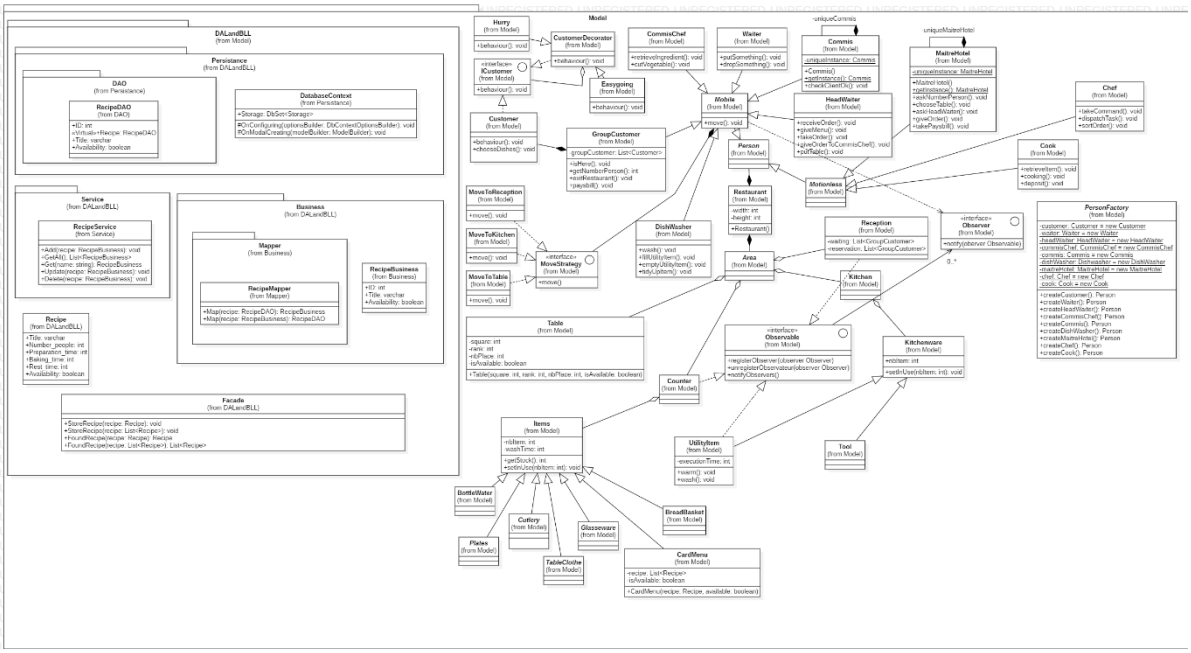
Dans le contrôleur, nous gérons toutes les fonctionnalités de l'application, le fait de démarrer la simulation du restaurant, d'accélérer le temps, ou alors de le mettre en pause. Toutes les mécaniques de l'application se trouvent ici. Ce composant permet de mettre en relation l modèle et la vue

METTRE IMAGE CONTROLEUR

Enfin, pour le modèle, nous implémentons une architecture en couches, avec les couches DAL (Data Access Layer, la couche d'accès aux données) et BLL (Business Logic Layer, la couche des processus métiers). Chacune de ses couches regroupe les composants partageant les mêmes fonctionnalités.

A l'intérieur de ces deux couches reliées en un seul package, nous trouvons un package nommé « Persistance », qui contient les informations liées à la BDD et sa configuration. Ensuite, nous avons un package « Service » qui contient une classe mettant à disposition les méthodes pour modifier la base de données (Add, Update et Delete). Puis vient le package « Business » qui fait la liaison entre le BDD et le client, grâce au package « Mapper » qui contient une classe permettant de faire la connexion entre les couches Persistance, Service et Business. Pour finir avec le package DAL/BLL, nous avons la classe « Facade » qui permet de simplifier la liaison avec la BDD.

Dans l'autre partie nous avons toutes les classes de nos éléments apparaissant dans la simulation, les acteurs (chef de cuisine, client, commis, etc…) et les éléments immobiles (le menu, les tables, etc…).

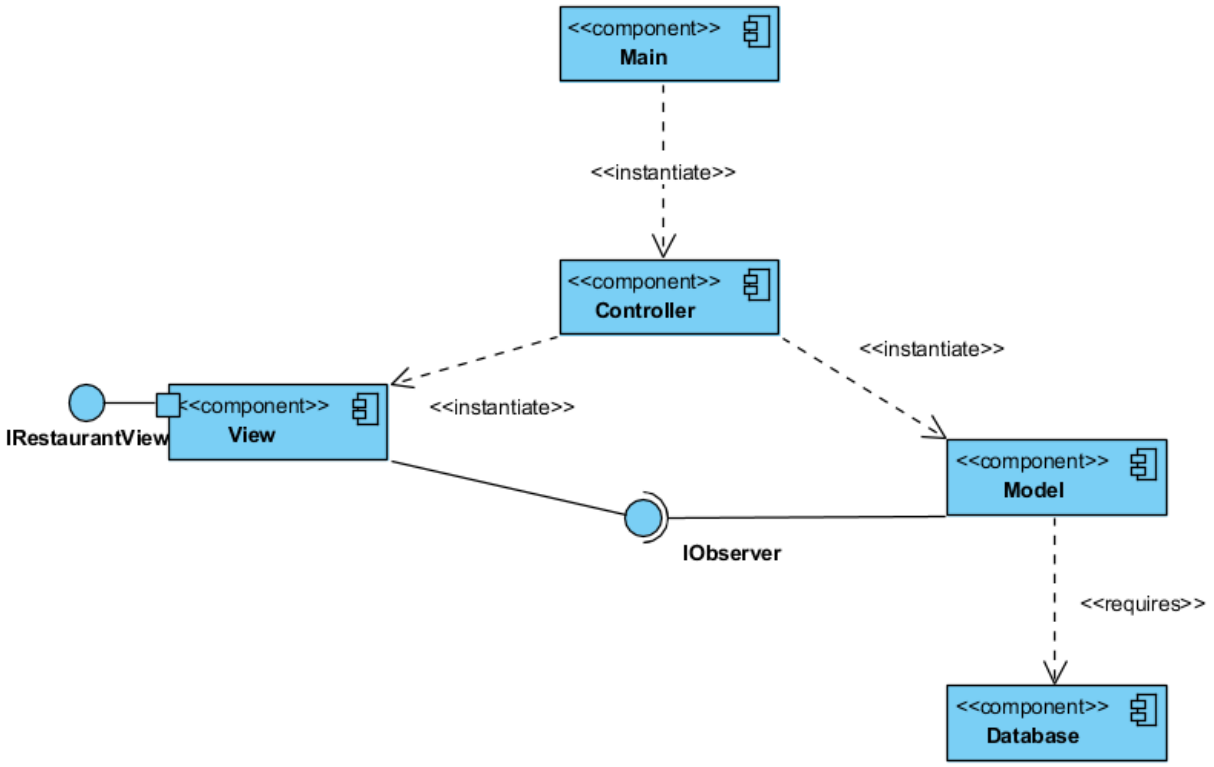


4. Diagramme de composants

Le diagramme de composants permet de représenter les relations entre les différents composants de notre système.

Après avoir réalisé notre diagramme de classe, nous avons construit un diagramme de composants permettant de schématiser nos différents composants :

- Le **Main**.
- Le **Controller**.
- La **View**.
- Le **Model**.
- La **Database**.



Comme l'indique le diagramme ci-dessus, le Main va instancier le Controller qui va lui-même instancier le Model ainsi que la View. Lorsqu'il va y avoir des changements dans le Model, ces changements seront notifiés grâce à l'interface IObserver dans la vue. Enfin, le Model requière l'accès à la base de données afin de pouvoir utiliser les données requises pour le bon fonctionnement de l'application.

5. Diagrammes de séquences

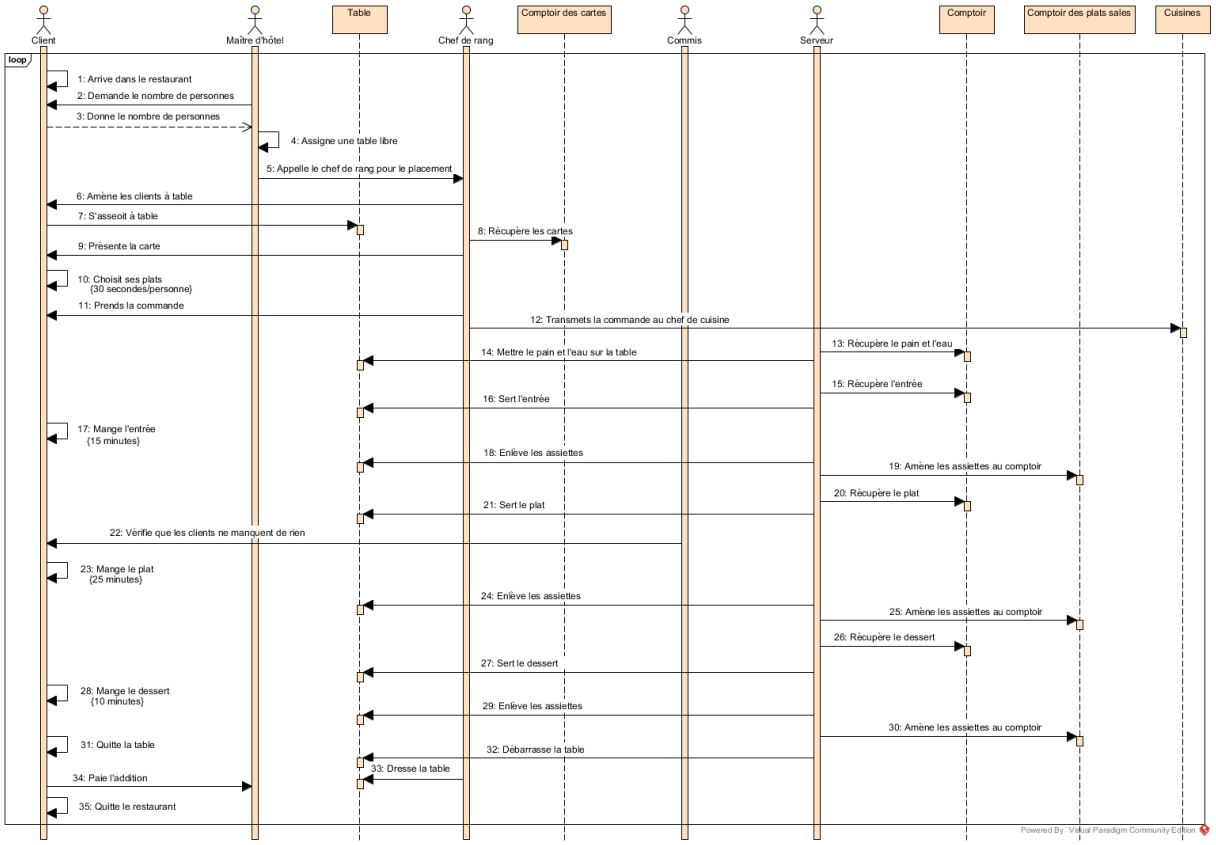
Le diagramme de séquence représente de manière graphique les interactions entre les acteurs et le système de manière chronologique.

Nous avons donc réalisé trois diagrammes différents :

- Un général, qui comporte tout le système ;
- Un se concentrant sur les actions ayant lieues à l'intérieur de la salle du restaurant ;
- Et un se concentrant sur les actions ayant lieues dans la cuisine.

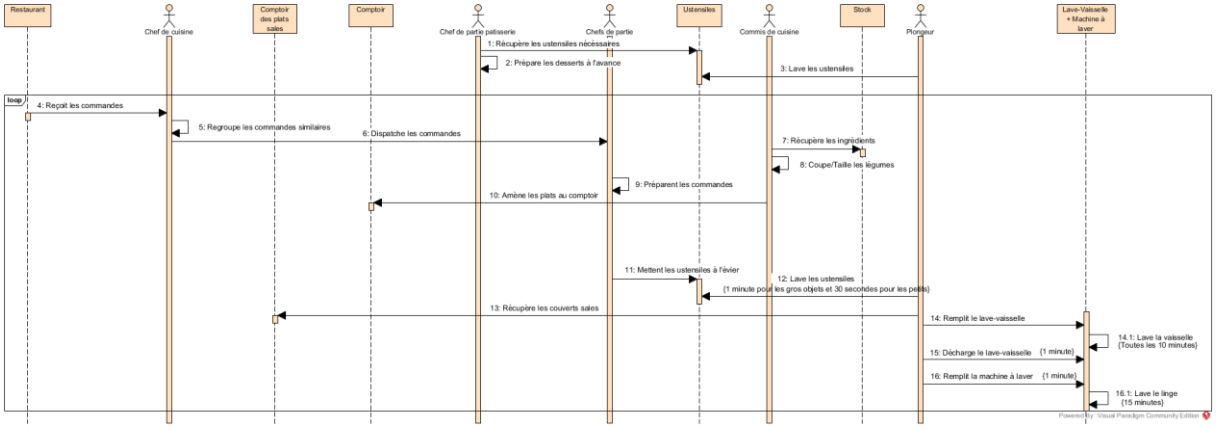
Le général comporte toutes les actions du système. Il permet de visualiser globalement l'ensemble des actions ayant lieues dans tout le restaurant. Afin de le rendre plus lisible, il a été séparé en deux : une partie qui visualise les interactions dans la salle du restaurant, et l'autre qui se concentre sur les actions dans les cuisines.

Vous pouvez retrouver les versions des diagrammes en grand format dans le dossier de modélisation (voir leur nom dans la partie Annexe de ce rapport).



Au sujet de la partie restaurant, on y voit les différentes interactions entre le client et les différents postes, ainsi que celles entre les différents postes et les cuisines et les autres éléments (tables, cartes, eau. Ceci détermine ce qui va se passer durant toute la durée d'ouverture, la boucle se répétant pour chaque client.

Pour la partie traitant des actions dans les cuisines on y voit les différentes interactions des postes avec le matériel en particulier, car les différents postes n'ont que très peu d'interactions entre eux. Une boucle tourne tout le temps de l'ouverture du restaurant, excluant ainsi le chef de pâtisserie qui prépare les gâteaux avant l'ouverture.



III. Explication des Design Pattern

1. Factory

Le Design Pattern Factory permet de créer des objets sans spécifier la classe concrète de celui-ci, et ce en fonction de paramètres fournis.

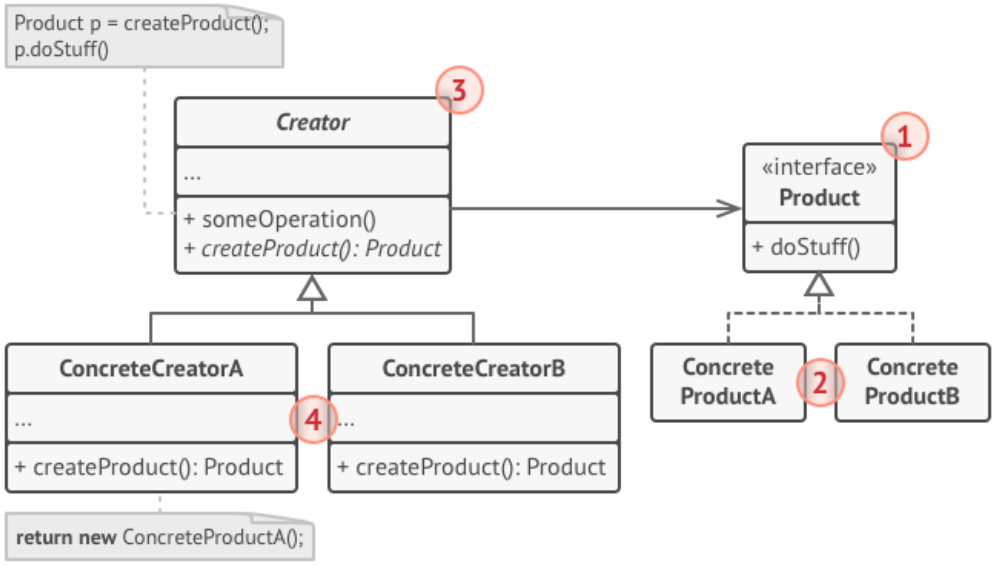
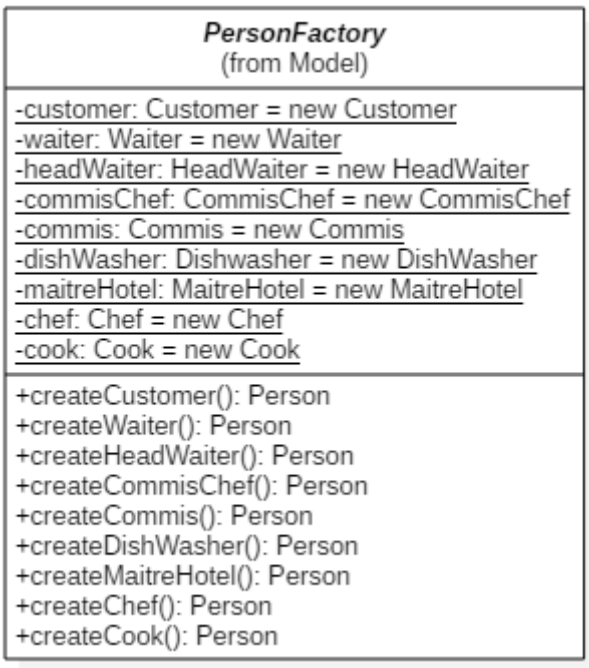


Figure 1 : Structure du DP Factory

Dans notre cas, nous utilisons la Factory pour instancier tous les éléments de notre simulation, comme par exemple les clients, les tables, les chefs de cuisines, etc…



2. Singleton

Le Design Pattern Singleton permet de restreindre l'instanciation d'une classe à un seul objet.

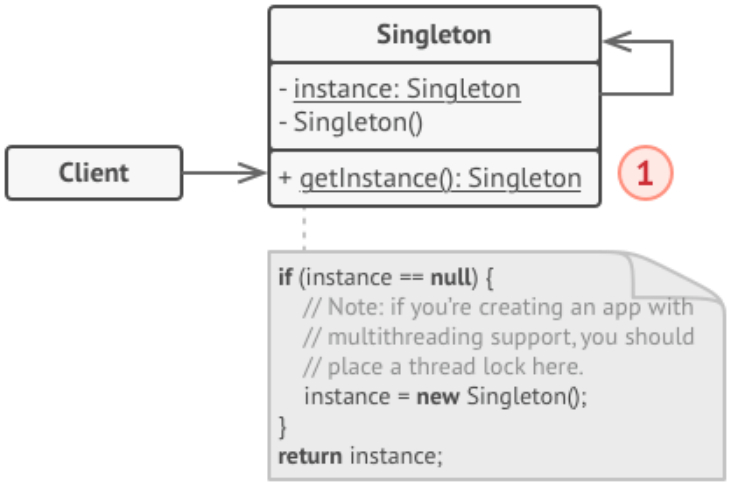
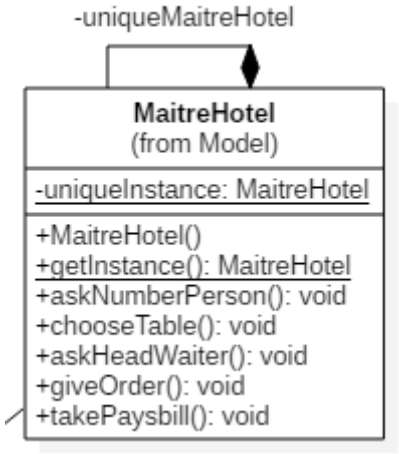


Figure 2 : Structure du DP Singleton

Dans notre cas, nous utilisons le Singleton pour gérer les objets qui ne nécessitent qu'une seule instance, par exemple le maître d'hôtel, le plongeur, etc…



3. Strategy

Le Design Pattern Strategy permet de changer d'algorithme utilisé de façon dynamique au cours du temps d'exécution.

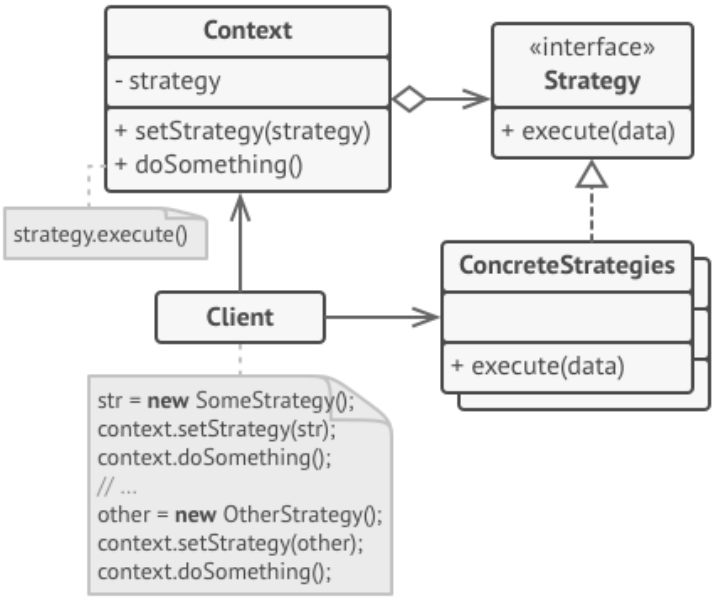
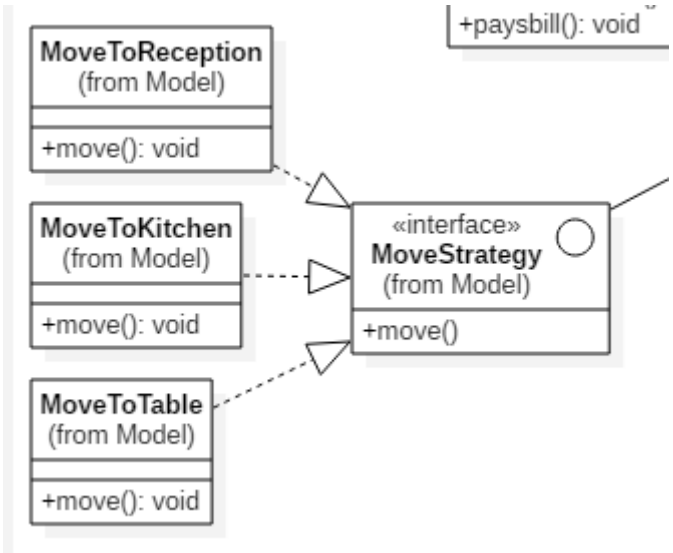


Figure 3 : Structure du DP Strategy

Dans notre cas, nous utilisons le Strategy pour attribuer un caractère aux différents types de clients, pour qu'ils aient un comportement propre.



4. Observer

Le Design Pattern Observer permet d'envoyer un signal à des observateurs afin qu'ils effectuent alors l'action adéquate en fonction des informations qui parviennent depuis les observables.

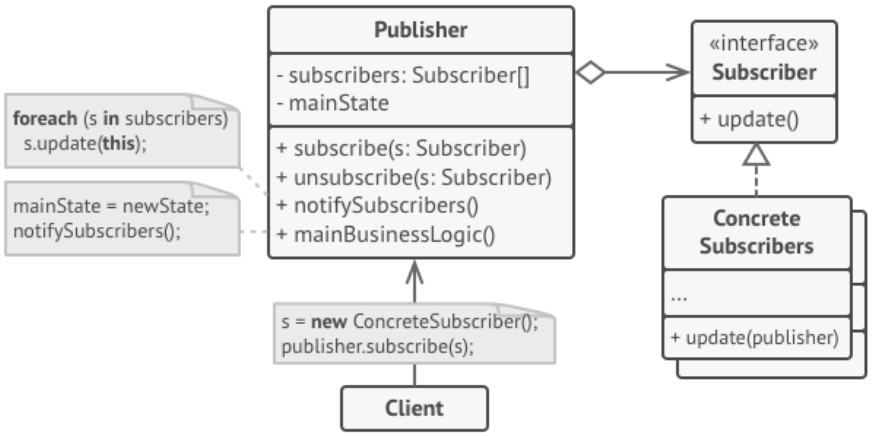
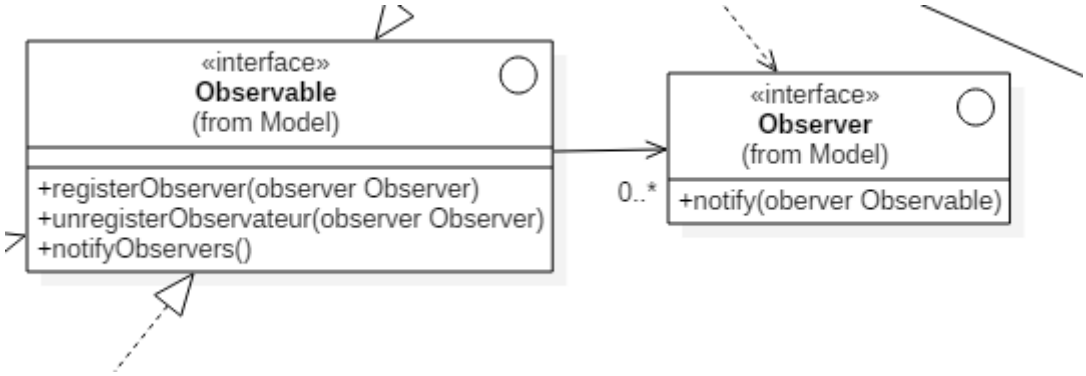


Figure 4 : Structure du DP Observer

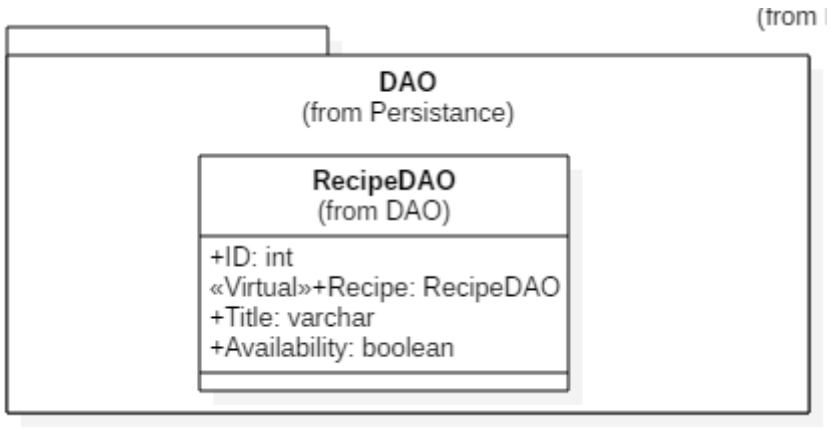
Dans notre cas, nous utilisons l'Observer pour avertir des déplacements dans la carte de la simulation.



5. Data Access Object

Le Design Pattern DAO (Data Access Object) permet de regrouper les accès aux données persistantes stockées en base de données dans des classes à part plutôt que de les disperser afin d'isoler le code responsable du stockage des données.

Dans notre cas, nous utilisons le DAO afin d'isoler les données et la manière dont les données sont stockées.



6. Façade

Le Design Pattern Façade permet de cacher une conception et une interface complexe difficile à comprendre afin de simplifier son utilisation.

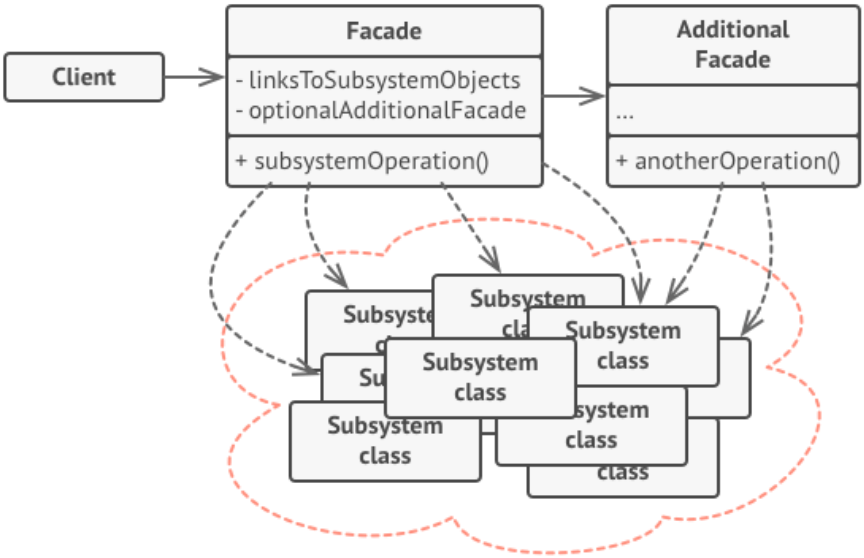
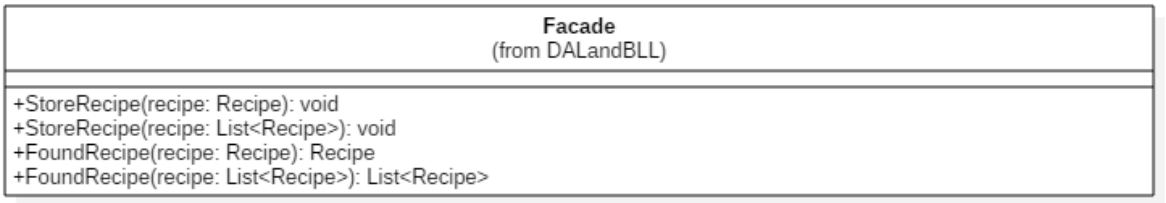


Figure 5 : Structure du DP Façade

Dans notre cas, nous utilisons la Façade pour effectuer le lien entre la gestion de BDD et les autres classes du modèle, de manière plus simplifiée.



7. Modèle/Vue/Controleur

Le motif d'architecture logicielle Modèle/Vue/Contrôleur est composé de trois types de modules : le modèle, qui contient les données, la vue, qui correspond à l'interface graphique, et le contrôleur, qui contient la logique des interactions utilisateurs.

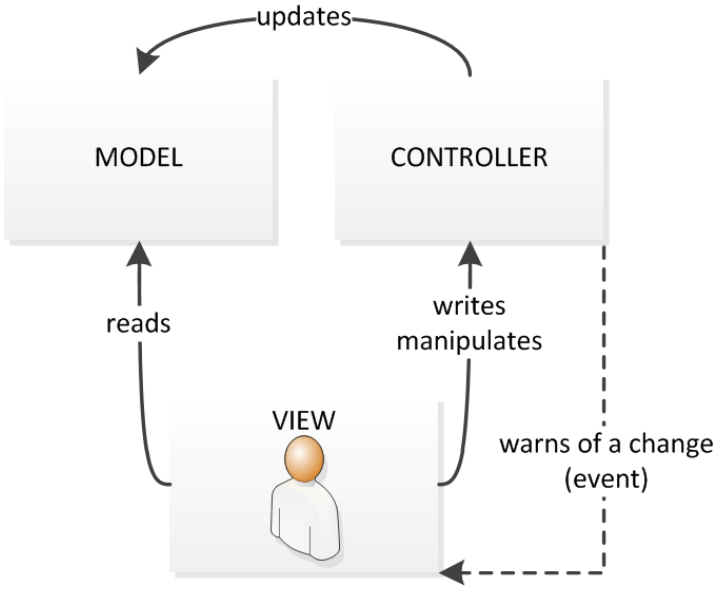


Figure 6 : Structure de l'architecture MVC

Dans notre cas, nous utilisons l'architecture Modèle/Vue/Contrôleur afin de séparer les différents composants de notre application, améliorant ainsi la clarté du code et son évolution.