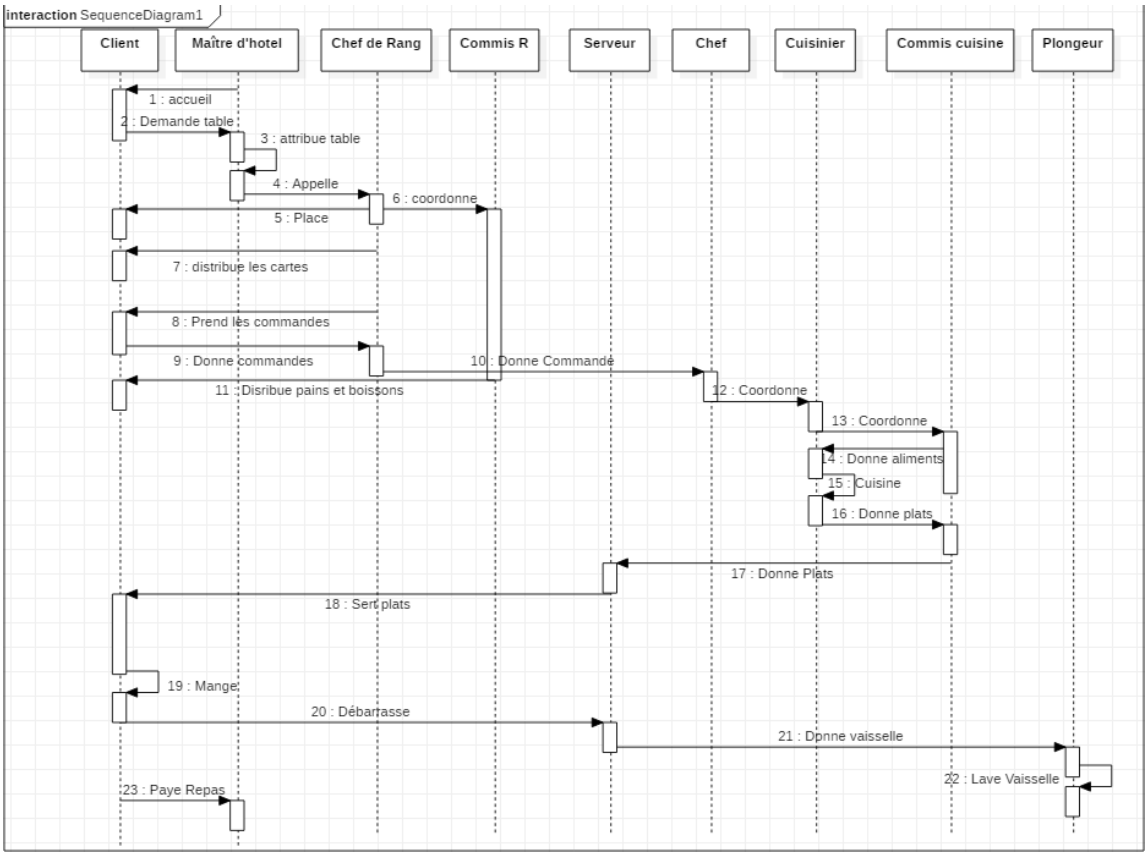
Livrable du 5 décembre 2018

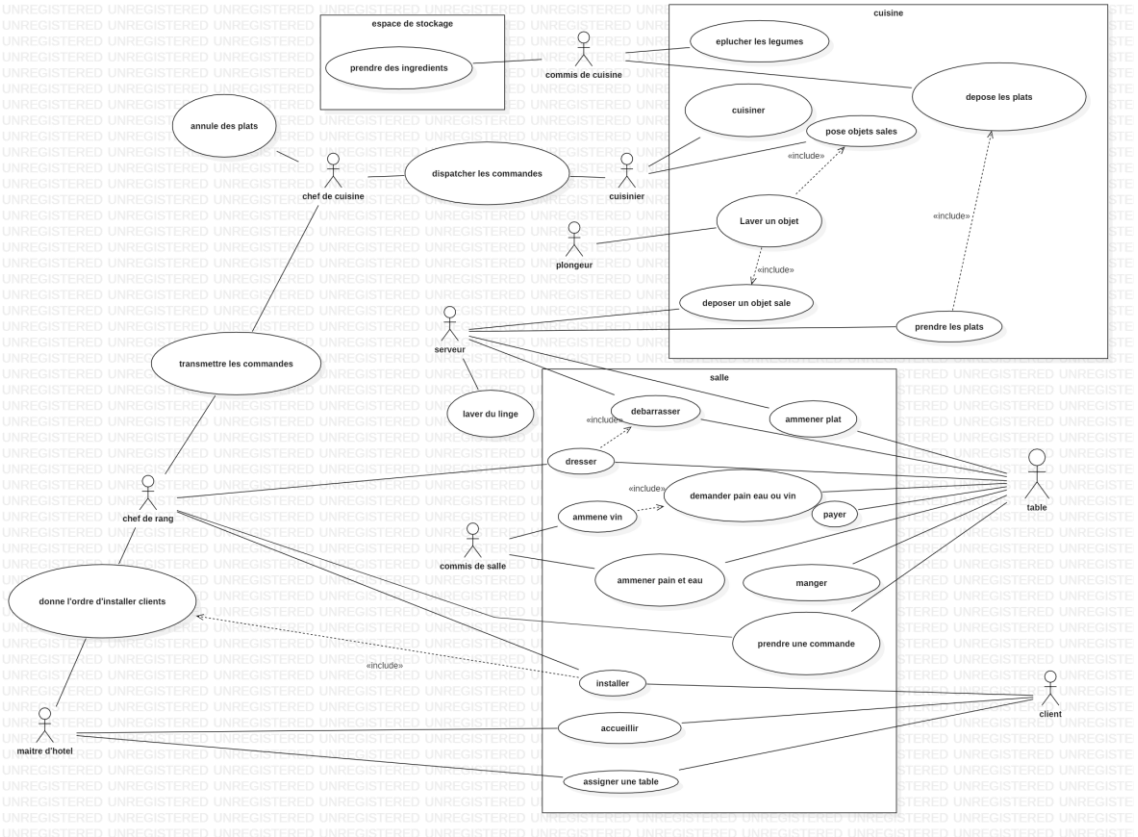
I. Table des matières

II. Diagrammes UML

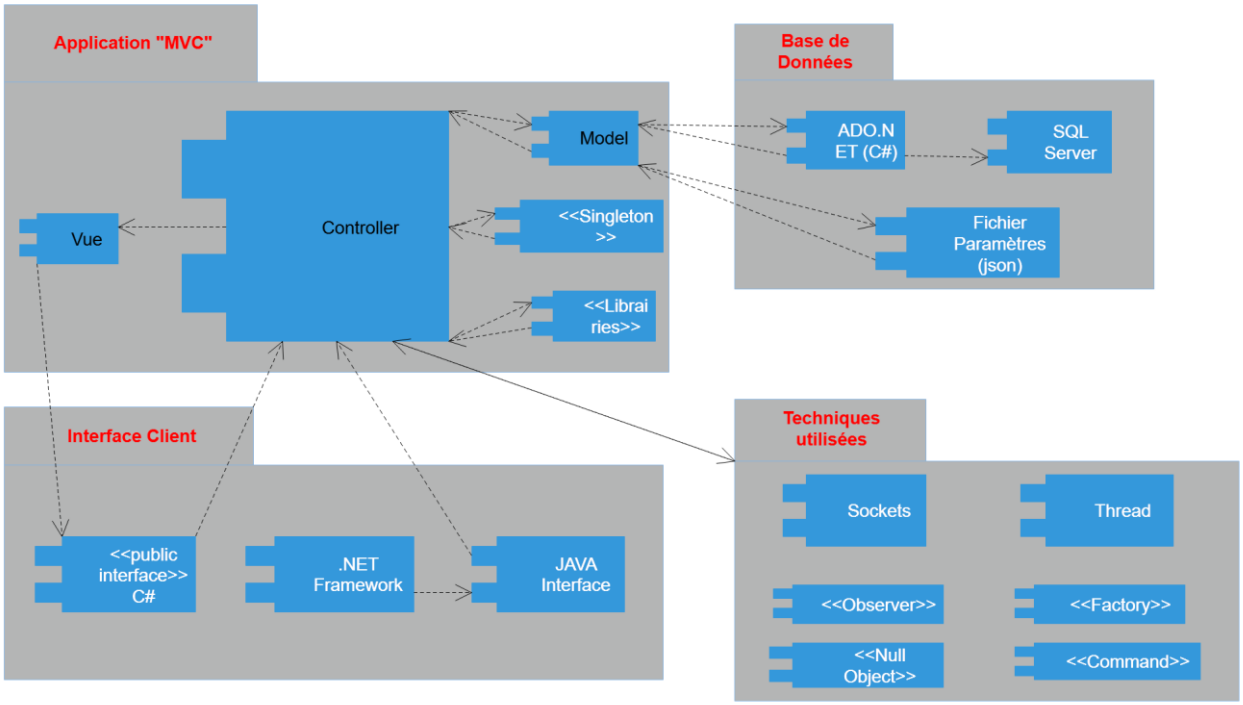
A. Diagramme de séquence



B. Diagramme des cas d'utilisation

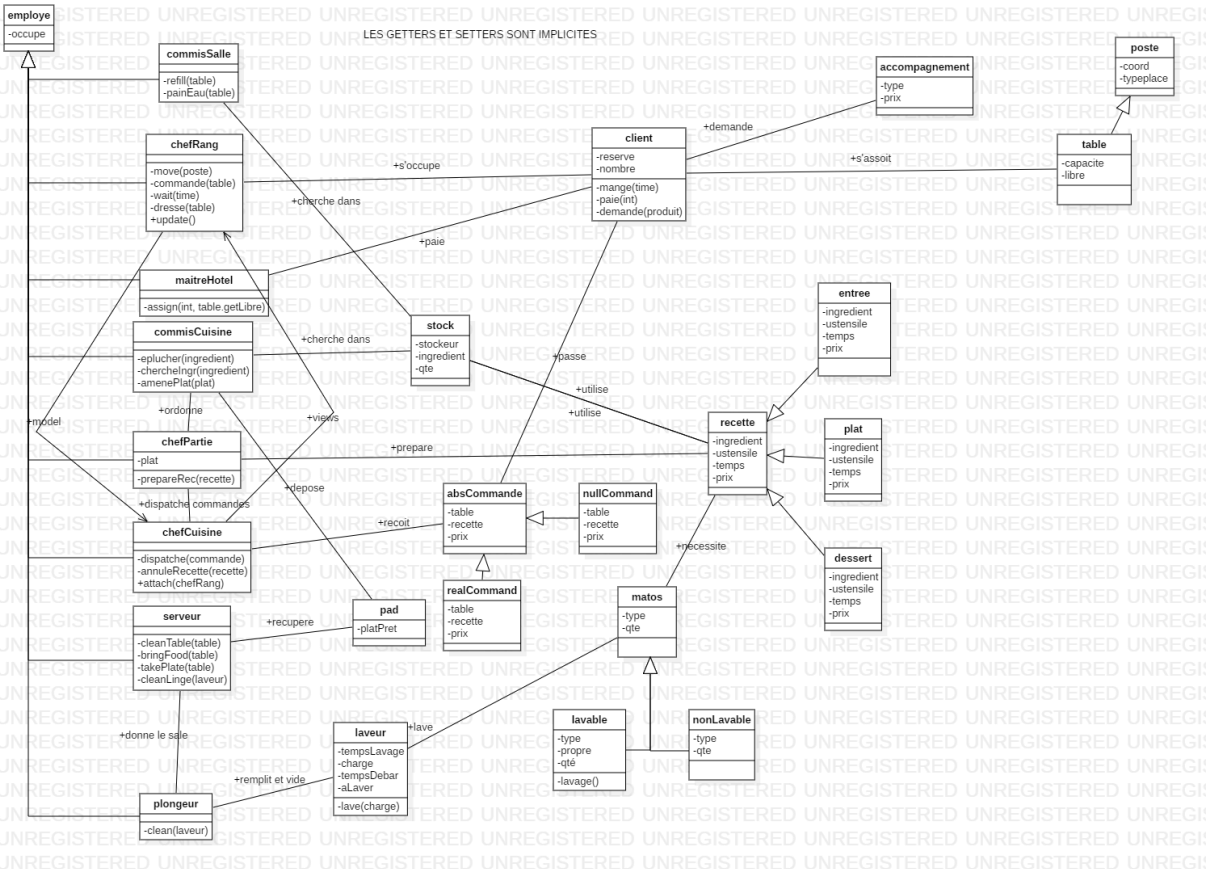


C. Diagramme des composants

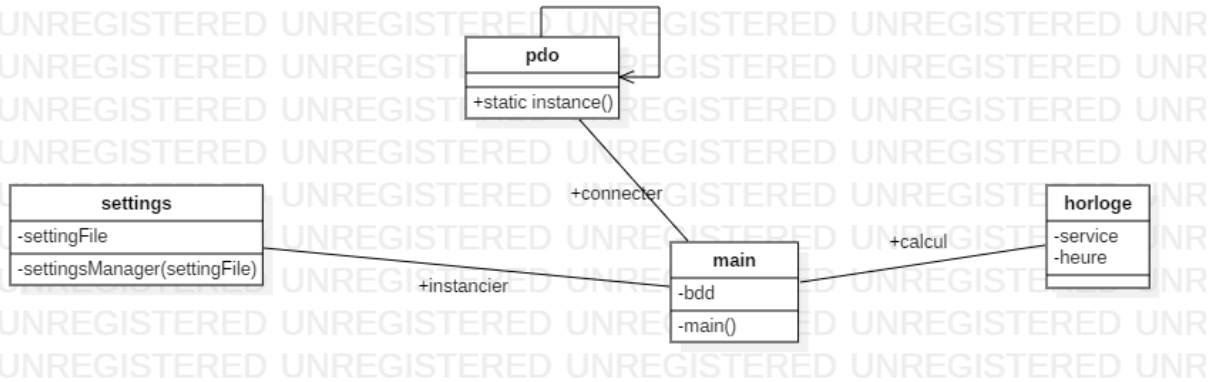


D. Diagramme des classes

1. Model



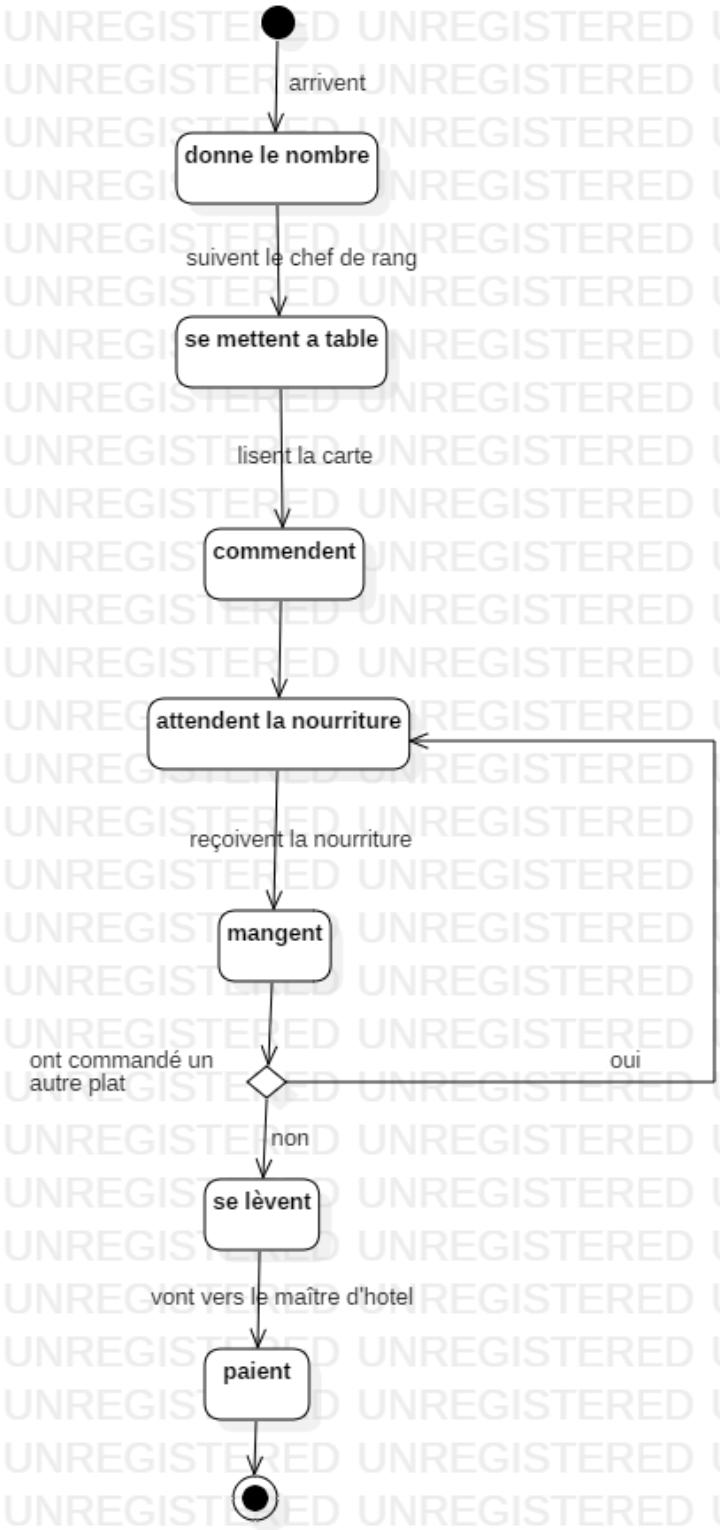
2. Controller



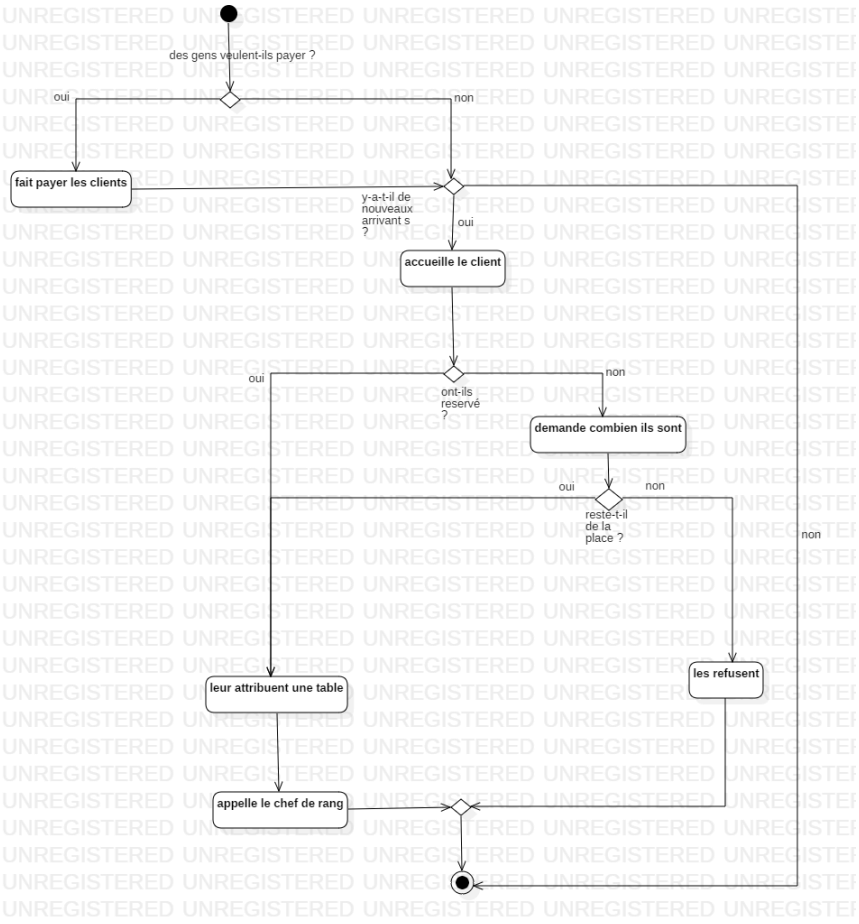
3. Vue

E. Diagramme d'activité

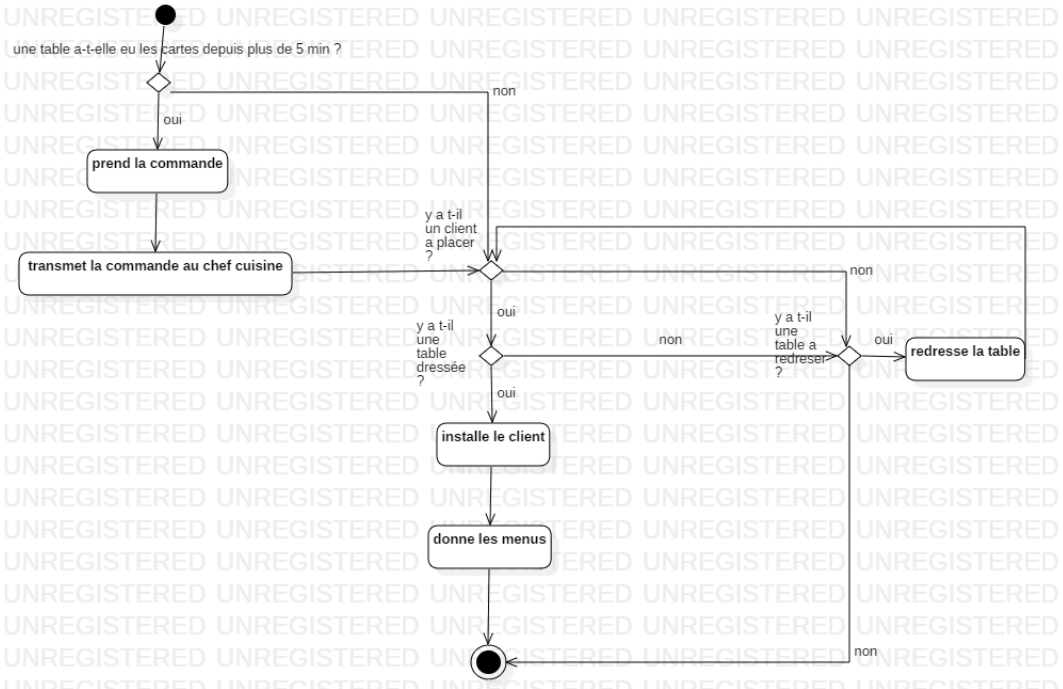
1. Client



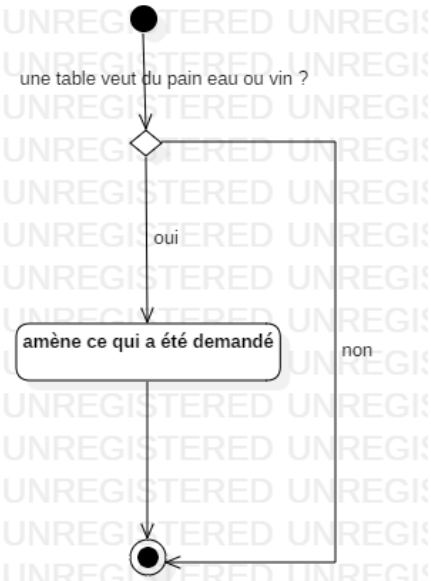
2. Maître d'hôtel



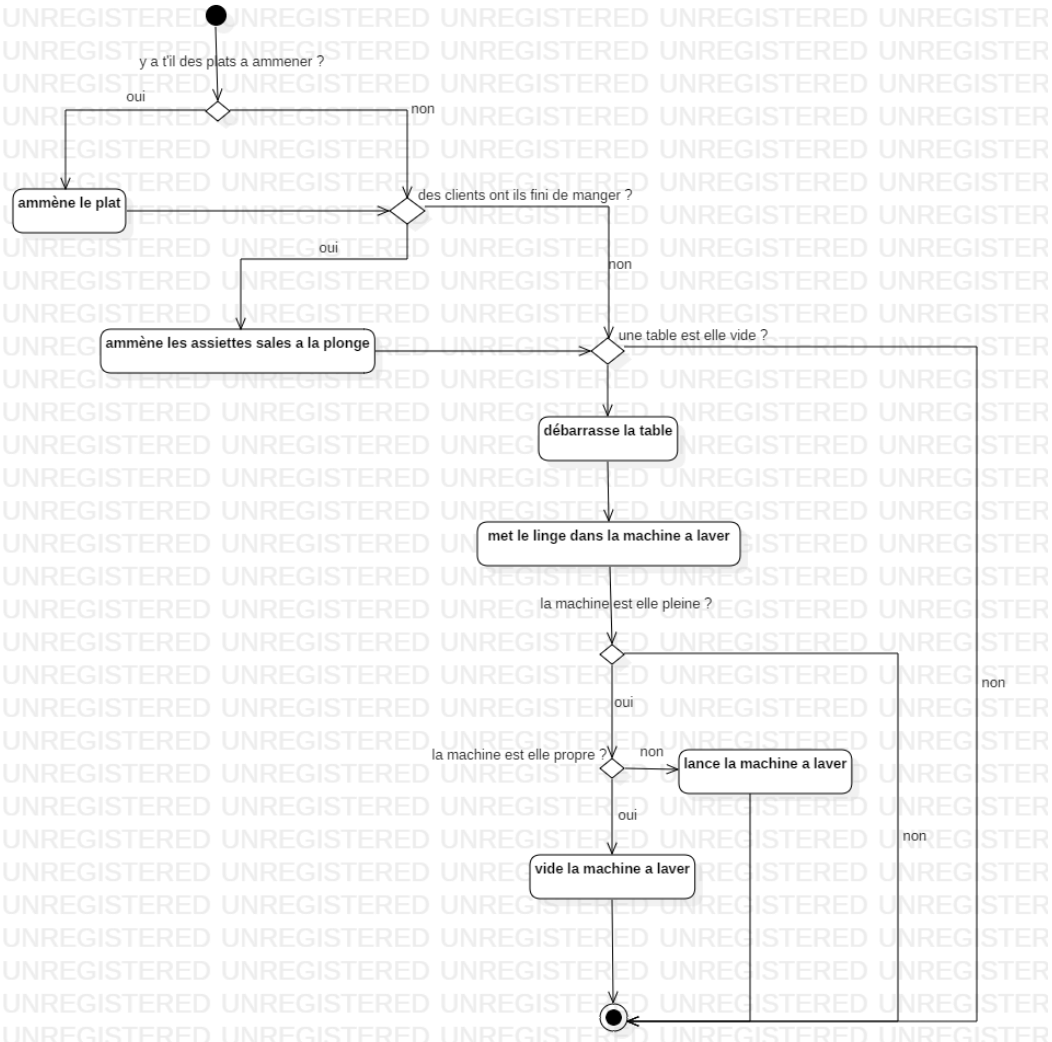
3. Chef de Rang



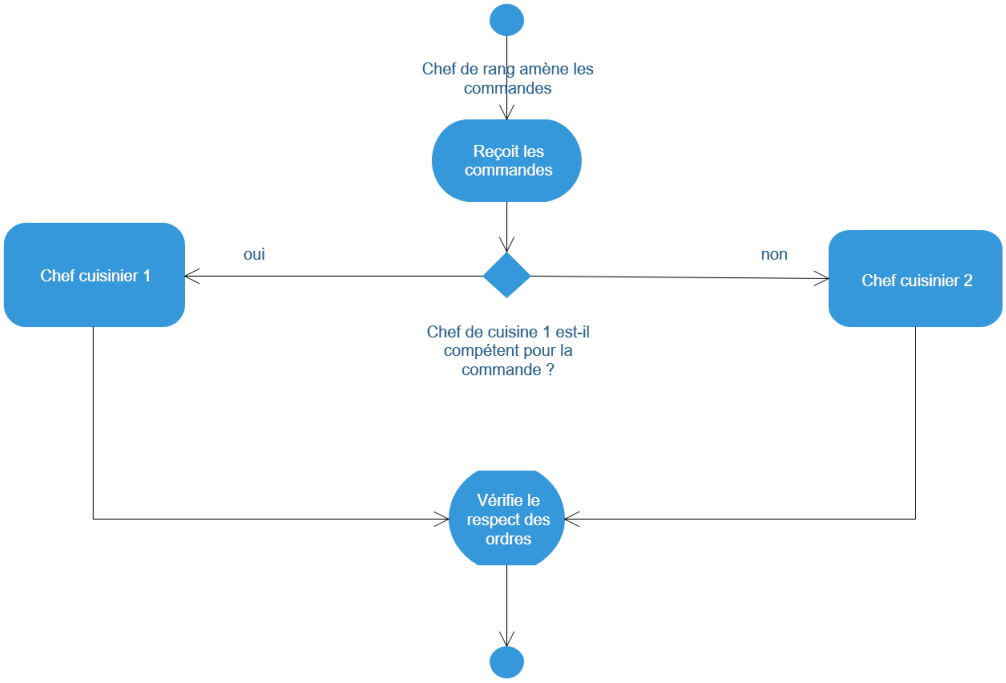
4. Commis de salle



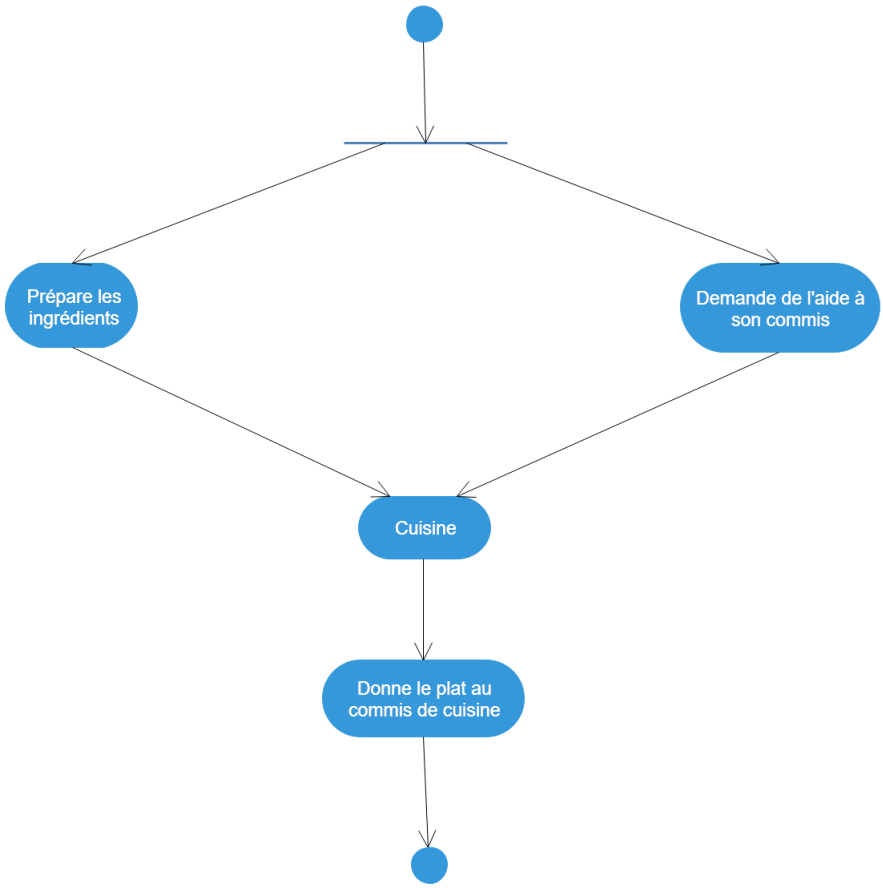
5. Serveur



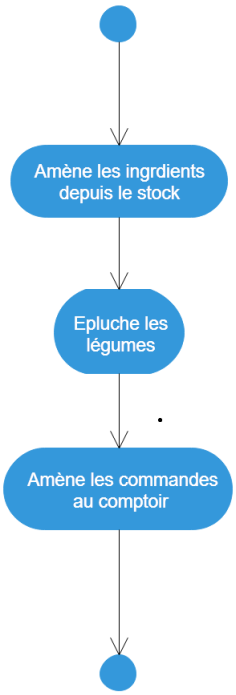
6. Chef de cuisine



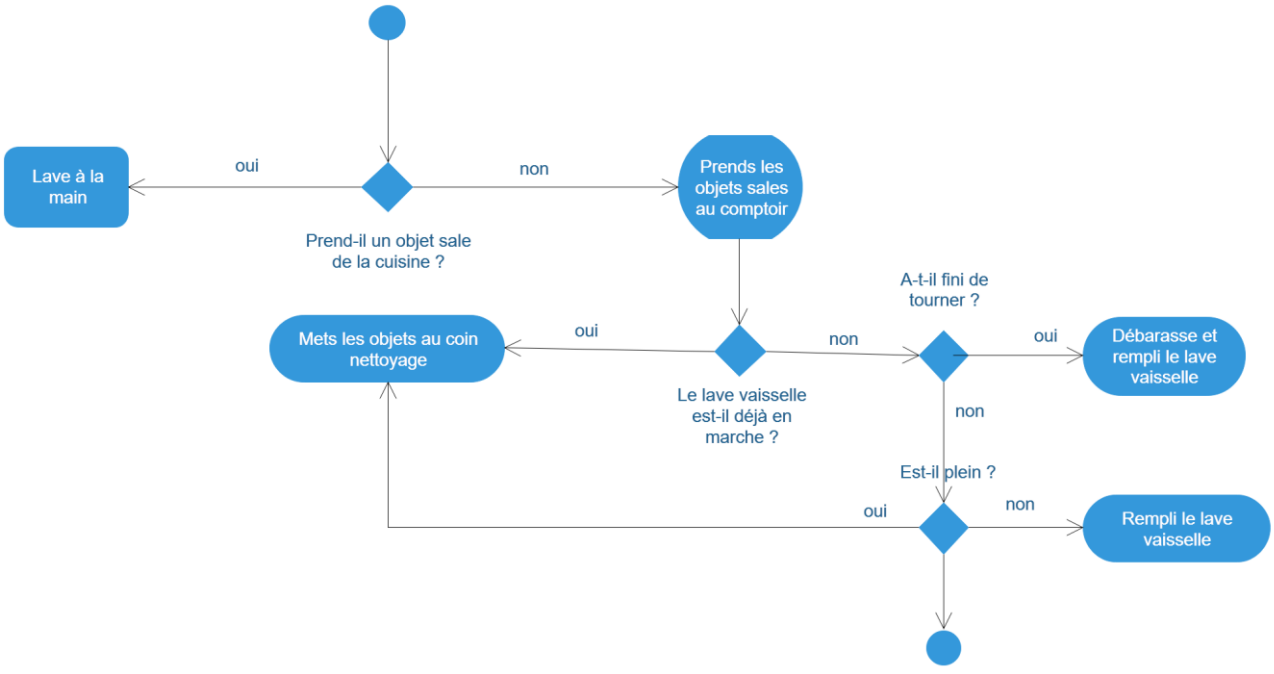
7. Chef de partie



8. Commis de cuisine

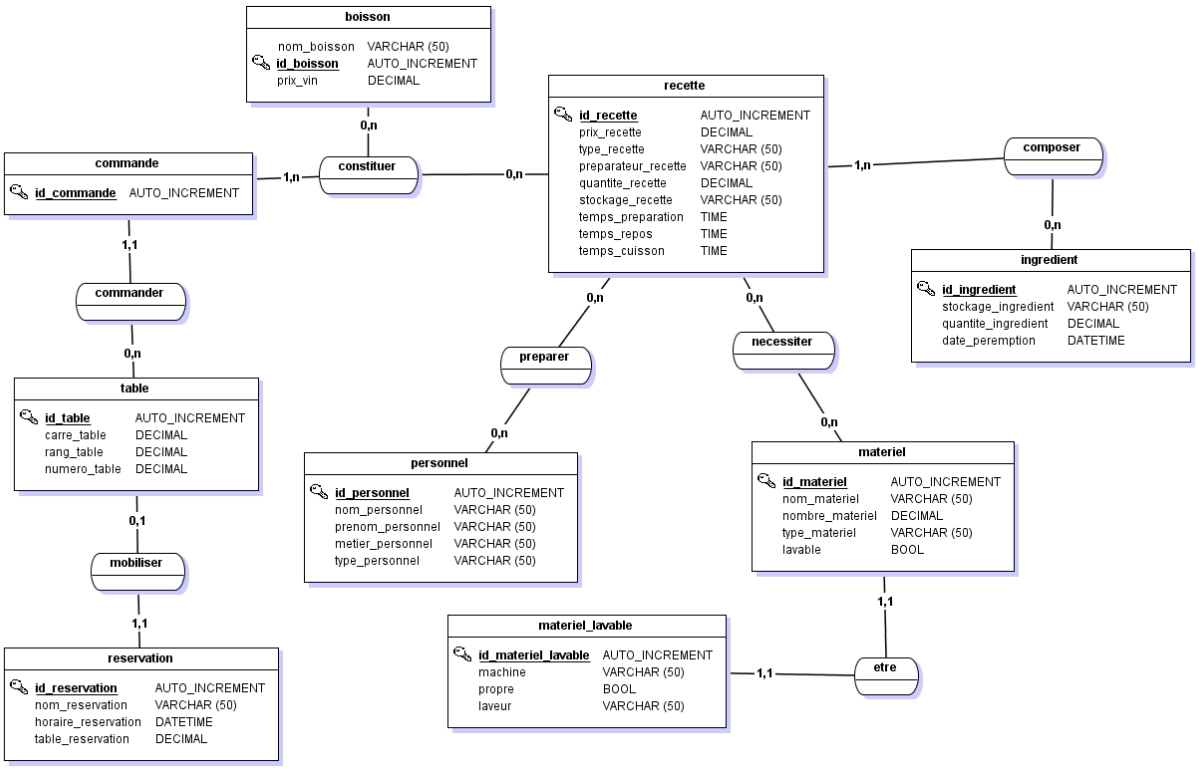


9. Plongeur



III. MCD et script SQL

A. MCD



B. Script SQL

```
#------------------------------------------------------------
#        Script MySQL.
#------------------------------------------------------------

#------------------------------------------------------------
# Table: boisson
#------------------------------------------------------------

CREATE TABLE boisson(
        id_boisson  Int  Auto_increment  NOT NULL ,
        nom_boisson Varchar (50) NOT NULL ,
        prix_vin    Decimal NOT NULL
            ,CONSTRAINT boisson_PK PRIMARY KEY (id_boisson)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: table
#------------------------------------------------------------

CREATE TABLE table(
        id_table     Int  Auto_increment  NOT NULL ,
        carre_table  Decimal NOT NULL ,
        rang_table   Decimal NOT NULL ,
        numero_table Decimal NOT NULL
            ,CONSTRAINT table_PK PRIMARY KEY (id_table)
```

```
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: reservation
#------------------------------------------------------------

CREATE TABLE reservation(
        id_reservation      Int  Auto_increment  NOT NULL ,
        nom_reservation     Varchar (50) NOT NULL ,
        horaire_reservation Datetime NOT NULL ,
        table_reservation   Decimal NOT NULL ,
        id_table            Int NOT NULL
                ,CONSTRAINT reservation_PK PRIMARY KEY (id_reservation)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: commande
#------------------------------------------------------------

CREATE TABLE commande(
        id_commande Int  Auto_increment  NOT NULL ,
        id_table    Int NOT NULL
                ,CONSTRAINT commande_PK PRIMARY KEY (id_commande)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: personnel
#------------------------------------------------------------

CREATE TABLE personnel(
        id_personnel     Int  Auto_increment  NOT NULL ,
        nom_personnel    Varchar (50) NOT NULL ,
        prenom_personnel Varchar (50) NOT NULL ,
        metier_personnel Varchar (50) NOT NULL ,
        type_personnel   Varchar (50) NOT NULL
                ,CONSTRAINT personnel_PK PRIMARY KEY (id_personnel)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: recette
#------------------------------------------------------------

CREATE TABLE recette(
        id_recette          Int  Auto_increment  NOT NULL ,
        prix_recette        Decimal NOT NULL ,
        type_recette        Varchar (50) NOT NULL ,
        preparateur_recette Varchar (50) NOT NULL ,
        quantite_recette    Decimal NOT NULL ,
        stockage_recette    Varchar (50) NOT NULL ,
        temps_preparation   Time NOT NULL ,
        temps_repos         Time NOT NULL ,
        temps_cuisson       Time NOT NULL
                ,CONSTRAINT recette_PK PRIMARY KEY (id_recette)
```

```
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: ingredient
#------------------------------------------------------------

CREATE TABLE ingredient(
        id_ingredient       Int  Auto_increment  NOT NULL ,
        stockage_ingredient Varchar (50) NOT NULL ,
        quantite_ingredient Decimal NOT NULL ,
        date_peremption     Datetime NOT NULL
                ,CONSTRAINT ingredient_PK PRIMARY KEY (id_ingredient)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: composer
#------------------------------------------------------------

CREATE TABLE composer(
        id_recette    Int NOT NULL ,
        id_ingredient Int NOT NULL
                ,CONSTRAINT composer_PK PRIMARY KEY (id_recette,id_ingredient)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: constituer
#------------------------------------------------------------

CREATE TABLE constituer(
        id_boisson  Int NOT NULL ,
        id_recette  Int NOT NULL ,
        id_commande Int NOT NULL
                ,CONSTRAINT constituer_PK PRIMARY KEY (id_boisson,id_recette,id_commande)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: preparer
#------------------------------------------------------------

CREATE TABLE preparer(
        id_recette   Int NOT NULL ,
        id_personnel Int NOT NULL
                ,CONSTRAINT preparer_PK PRIMARY KEY (id_recette,id_personnel)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: materiel
#------------------------------------------------------------

CREATE TABLE materiel(
        id_materiel         Int  Auto_increment  NOT NULL ,
        nom_materiel        Varchar (50) NOT NULL ,
        nombre_materiel     Decimal NOT NULL ,
```

```
    type_materiel       Varchar (50) NOT NULL ,
    lavable             Bool NOT NULL ,
    id_materiel_lavable Int NOT NULL
        ,CONSTRAINT materiel_PK PRIMARY KEY (id_materiel)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: materiel_lavable
#------------------------------------------------------------

CREATE TABLE materiel_lavable(
    id_materiel_lavable Int  Auto_increment  NOT NULL ,
    machine             Varchar (50) NOT NULL ,
    propre              Bool NOT NULL ,
    id_materiel         Int NOT NULL ,
    id_personnel        Int NOT NULL
        ,CONSTRAINT materiel_lavable_PK PRIMARY KEY (id_materiel_lavable)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: etre
#------------------------------------------------------------

CREATE TABLE etre(
    id_materiel   Int NOT NULL ,
    id_materiel_lavable Int NOT NULL
        ,CONSTRAINT etre_PK PRIMARY KEY (id_materiel,id_materiel_lavable)
)ENGINE=InnoDB;


#------------------------------------------------------------
# Table: necessiter
#------------------------------------------------------------

CREATE TABLE necessiter(
    id_recette   Int NOT NULL ,
    id_materiel Int NOT NULL
        ,CONSTRAINT necessiter_PK PRIMARY KEY (id_recette,id_materiel)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: commander
#------------------------------------------------------------

CREATE TABLE commander(
    id_commande   Int NOT NULL ,
    id_table Int NOT NULL
        ,CONSTRAINT commander_PK PRIMARY KEY (id_commande,id_table)
)ENGINE=InnoDB;

#------------------------------------------------------------
# Table: mobiliser
#------------------------------------------------------------
```

```
CREATE TABLE mobiliser(
      id_table   Int NOT NULL ,
      id_reservation Int NOT NULL
           ,CONSTRAINT mobiliser_PK PRIMARY KEY (id_table,id_reservation)
)ENGINE=InnoDB;
```

IV. Design Pattern

A. MVC

Un des plus célèbres *design patterns* s'appelle MVC, qui signifie **Modèle - Vue - Contrôleur**. C'est celui que nous allons découvrir maintenant.

Le pattern MVC permet de bien organiser son code source. Il va vous aider à savoir quels fichiers créer, mais surtout à définir leur rôle. Le but de MVC est justement de séparer la logique du code en trois parties que l'on retrouve dans des fichiers distincts.

- **Modèle** : cette partie gère les *données* de votre site. Son rôle est d'aller récupérer les informations « brutes » dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur. On y trouve donc entre autres les requêtes SQL.
- **Vue** : cette partie se concentre sur l'*affichage*. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions PHP très simples, pour afficher par exemple une liste de messages.
- **Contrôleur** : cette partie gère la logique du code qui prend des *décisions*. C'est en quelque sorte l'intermédiaire entre le modèle et la vue : le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue. Le contrôleur contient exclusivement du PHP. C'est notamment lui qui détermine si le visiteur a le droit de voir la page ou non (gestion des droits d'accès).

B. Singleton

Le singleton est un patron de conception *(design pattern)* dont l'objectif est de restreindre l'instanciation d'une classe à un seul objet (ou bien à quelques objets seulement). Il est utilisé lorsqu'on a besoin exactement d'un objet pour coordonner des opérations dans un système. Le modèle est parfois utilisé pour son efficacité, lorsque le système est plus rapide ou occupe moins de mémoire avec peu d'objets qu'avec beaucoup d'objets similaires.

C. Observer

Il est utilisé pour 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*).

D. Factory machine

La fabrique (*factory method*) est un patron de conception utilisé en programmation orientée objet. Elle 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.

Plusieurs fabriques peuvent être regroupées en une fabrique abstraite permettant d'instancier des objets dérivant de plusieurs types abstraits différents.

Les fabriques étant en général uniques dans un programme, on utilise souvent le patron de conception singleton pour les implémenter.

E. Null object

Dans la programmation orientée objet, un objet null est un objet sans valeur référencée ou qui a un comportement défini comme neutre (null). Le **patron de conception de l'objet null** décrit l'utilisation de ces objets et de leurs comportements.

Dans la plupart des langages orientés objet, les références peuvent être null. Il est nécessaire de vérifier que ces références ne sont pas null avant d'en invoquer les méthodes. En effet, il n'est pas possible d'invoquer une méthode sur une référence null.

F. Strategy

Le patron de conception stratégie est utile pour des situations où il est nécessaire de permuter dynamiquement les algorithmes utilisés dans une application. Le patron stratégie est prévu pour fournir le moyen de définir une famille d'algorithmes, encapsuler chacun d'eux en tant qu'objet, et les rendre interchangeables. Ce patron laisse les algorithmes changer indépendamment des clients qui les emploient.