Université des sciences et de la technologie Houari Boumédiène

Rapport

Gestion du suivi des commandes d'un restaurant-cafétéria

Présentation

Département Informatique
Section : ISIL A

Etudiants :

Hammani Aimene / 181831092088 / Groupe 02
Zitouni Lokmane / 181831091028 / Groupe 02

Enseignant :

Mme Samia Boulkrinate

Contenu du rapport :

- Choix du cycle de vie
- Cahier des charges
- Diagrammes UML
- Conception architectural
- Conception détaillée

I) Choix du cycle de vie :

Nous choisissons un cycle de vie en cascade car :

Le but du projet est de réaliser une application pour gérer les suivis des commandes d'un restaurant-cafeteria permettant aux clients de passer des commandes à manger à domicile o sur place ainsi que d'autres fonctionnalités, aux employés de valider leur affectation a une livraison et au chef de gérer les données traitées par l'application (employés, clients, plats, boissons, statistiques ect…).

La définition du produit est stable, bien détaillé et claire.

Un cycle de vie en cascade est utilisé lorsqu'on a un Environnement de développement solide, pour cela on a différents outilles, notamment draw.io pour la conception (diagrammes UML), Intellij IDEA (un IDE basé sur Java) pour le développement.

II) Cahier des charges :

Table des matières

1. INTRODUCTION :

Le présent document a pour objet de donner et définir les différentes prestations d'une application de gestion et suivi des commandes d'un restaurant-cafeteria.

2. OBJECTIF DU PROJET :

L'émergence d'applications spécialisées dans les restaurants a tous une révolution dans l'industrie. Les nombreux avantages qu'ils offrent et les services publics facilitent la gestion quotidienne des deux gestionnaires et propriétaires d'entreprises, et le client. Le projet a pour but de définir une application qui répond aux besoins des services nécessaire pour la bonne conduite et gestion d'un restaurant-cafeteria.

3. CONVIVIALITE :

L'application développée doit être simple, claire, conviviale, facile à manipuler et ne nécessite aucune exigence de l'utilisateur. N'importe qui doit pouvoir utiliser l'application.

4. ETENDU DU PROJET :

Dans des conditions optimales de rentabilité, de sécurité et de respect de l'environnement.

4.1. Gestion des clients :

L'application doit prendre en charge la gestion des :

4.1.1. Types des clients :

- Client régulier.
- Client inopiné.

4.1.2. Caractéristiques d'un client :

Un client régulier est identifié par son :

- ID-Client.
- Nom.
- Prénoms.
- Adresse à domicile.
- Numéro de téléphone.
- Adresse email.
- Mot de passe compte

4.1.3. Comptes des clients :

Un client régulier dispose en plus d'un compte privé pour accéder en ligne à l'application lui permettant de :

- Choisir en l'occurrence les plats à mettre dans leur panier.
- Commander des plats sélectionnés du panier pour être livrés à domicile.
- Suivre la livraison depuis son départ jusqu'à son arrivé à destination.

Le client devra spécifier la liste des plats et boissons ainsi que le nombre de portions.

- Les client non-réguliers serons représenté par un compte générique.

4.1.4. Actions Programmable :

L'application doit permettre :

- L'ajout d'un client.
- La recherche d'un client.
- La modification des informations d'un client.

4.2. Gestion des Plats et Boissons :

L'application doit prendre en charge la gestion des :

4.2.1. Caractéristiques d'un plat ou boisson :

Un plat ou boisson est identifier par :

- ID-plat.
- Sa Spécialité culinaire (Orientale, Occidentale, Traditionnelle, Asiatique Ou Autre).
- Sa désignation.
- Sa nature (chaude ou froide).
- Son conditionnement (assiette, récipient avec couvercle, boite, plateau, barquette, Goblet, bouteille, etc.)
- Son prix.

4.2.2. Ingrédients d'un plat :

Pour chaque plat/boisson on tient compte des ingrédients qui lui constitue, pour chaque ingrédient on tient compte de :

- ID-Ingrédient.
- Nom du produit.
- Quantité.
- Unité de mesure.
- Prix de l'unité.

4.2.3. Les méthodes de préparation :

Pour chaque plat on tient compte des méthodes de préparation nécessaires, une méthode de préparation est constituée de :

- ID-méthode
- Un nom.
- Temps de préparation.
- Description.

4.2.4. Actions programmables :

L'application doit permettre :

- L'ajout d'un plat au menu.
- Suppression d'un plat du menu.
- Recherche d'un plat.
- Modification des caractéristiques d'un plat.

4.3. Gestion des commandes :

L'application doit prendre en charge la gestion des :

4.3.1. Types des commandes :

- Commande à servir sur place.
- Commande à livrer à domicile.

4.3.2. Caractéristiques d'une commande :

- ID-commande.
- Date et heure.
- ID-Employé traitant la commande.
- ID-Client

4.3.3. Détails des commandes :

L'application doit permettre la sauvegarde du :

- ID-Commande.
- ID-Plat.
- Nombre de portion commandé.
- Montant de la commande.

4.3.4. Actions Programmables :

L'application doit permettre :

- La création d'une commande.
- L'annulation d'une commande.
- La modification d'une commande.
- La recherche dans l'historique des commandes.

4.4. Gestion des livraisons :

Pour chaque commande de livraison à domicile valide un livreur doit la faire parvenir à destination, pour chaque livraison on tient compte de :

- ID-Livraison.
- ID-Commande.
- Sa date et heure de départ.
- L'employé livreur.

Une livraison est validée par le client destinataire en apposant sa signature électronique sur appareil destiné à cet effet.

4.5. Gestion des employés :

Pour chaque employé on tient compte des :

4.5.1. Caractéristiques d'un Employé :

- ID-Employé.
- Type Employé (Caissier, Livreur, chef restaurant).
- Adresse électronique.
- Mot de passe compte.
- état (En service, démissionné, viré).

4.5.2. Actions programmables :

- Ajouter un employé.
- Modifier.
- Rechercher.

4.6. Gestion des comptes :

L'application doit prendre en charge la gestion des

- Comptes client (accès aux services client).
- Comptes Employé simple.
- Comptes administratif (accès aux statistiques et actions administratives).

4.7. Gestion des Services client :

l'application doit s'abstraire.

1/ Une commande sur place ou à distance.
2/ Un service sur place ou livraison à domicile.
3/ Access au menu avec possibilité de recherche de plats par nom ou spécialité culinaire.
4/ Possibilité de suivre la livraison du plat depuis son départ jusqu'à son arrivé à ladresse indiqué
5/ Service de notification pour informer le client du début et la durée approximative de la livraison.
6/ Service d'évaluation pour qu'ils puissent noter le service (de 1 à 5) + espace commentaire pour laisser son avis.

4.8. Evolutivité du système :

Le system doit au fur et à mesure construire un profil pour chaque client régulier, en prenant en compte ses habitudes de commandes et ses gouts de préférence culinaire, il pourra ainsi recommander au différant clients les dernières nouveautés du restaurant les plus compatible avec leurs préférences culinaires.
Le system doit tenir compte de l'activité du client, éventuellement lui bénéficier de bons gratuit pour déjeuner ou pour commander s'il atteint un certain nombre de commandes ou que le montant de ses commandes antécédentes aura dépassé un certain seuil.

4.9. Statistiques et rapports :

Le système doit établir les statistiques suivantes :

- Nombre de client trié par catégorie.
- Nombre de commande sur place vs à distance.
- Chiffre mensuel enregistré.
- Les cinq premiers plats les plus commandés.
- Le nombre de commande de chaque client.
- Le nombre de livraison de chaque livreur.

Ce faisant il sera possible de sélectionner :

- Le client le plus fidèle.

- Le livreur le plus consciencieux.

L'Access à Ces statistique est limité au chef du restaurant.

4.10 Caractéristiques fonctionnelles :

- Les outils qui seront proposés pour l'interaction avec l'éditeur devront être intuitifs et facilement utilisables.
- Les données et les informations affichées doivent être présentées de manière bien structurée bien organisée afin qu'elles soient facilement et clairement lisibles et exploitables.

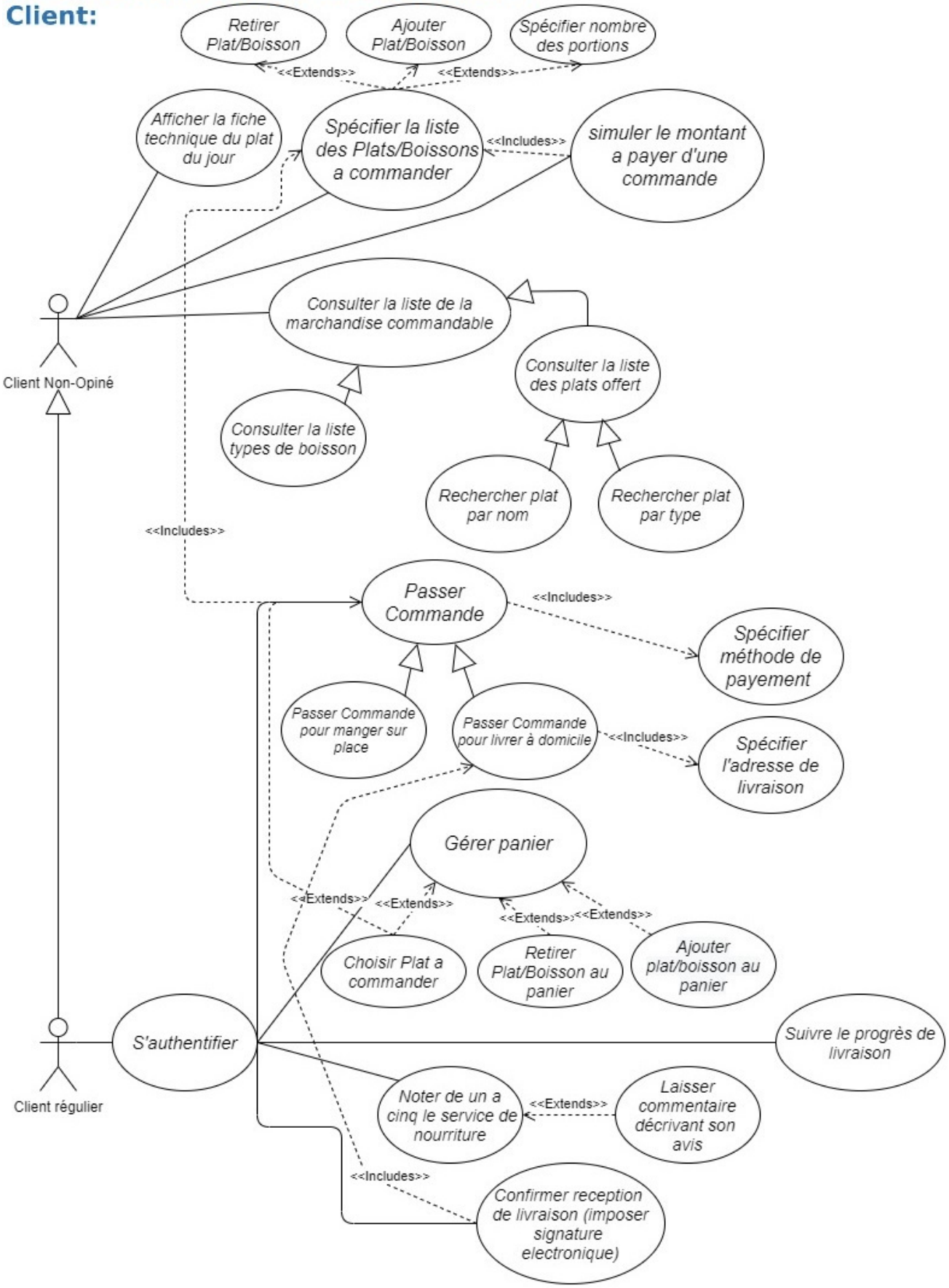
5. LIVERABLES

L'application doit être accompagné des procédures et guides d'installation, utilisation et maintenance.

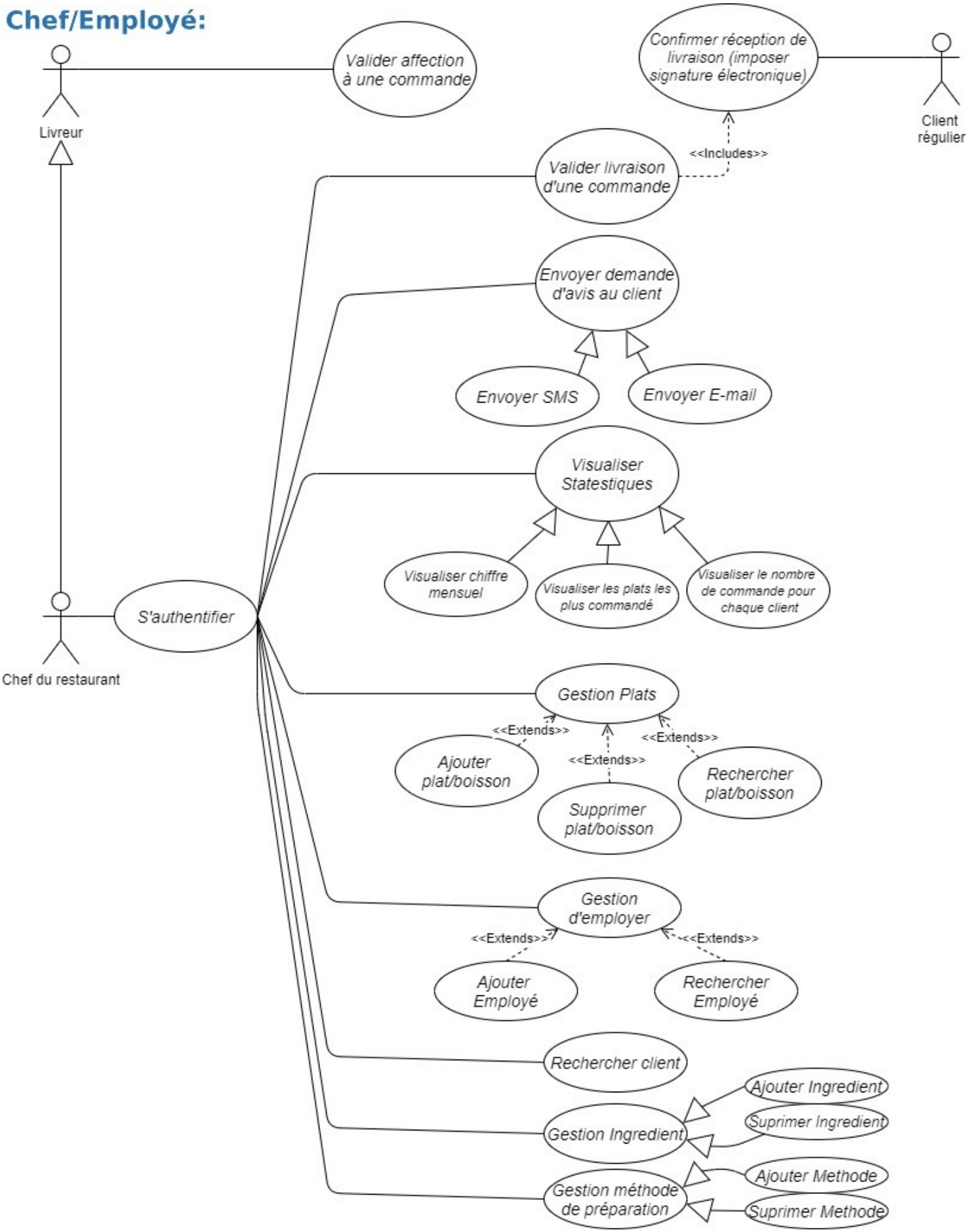
III) Les Diagrammes UML :

Les Diagramme des cas d'utilisation:

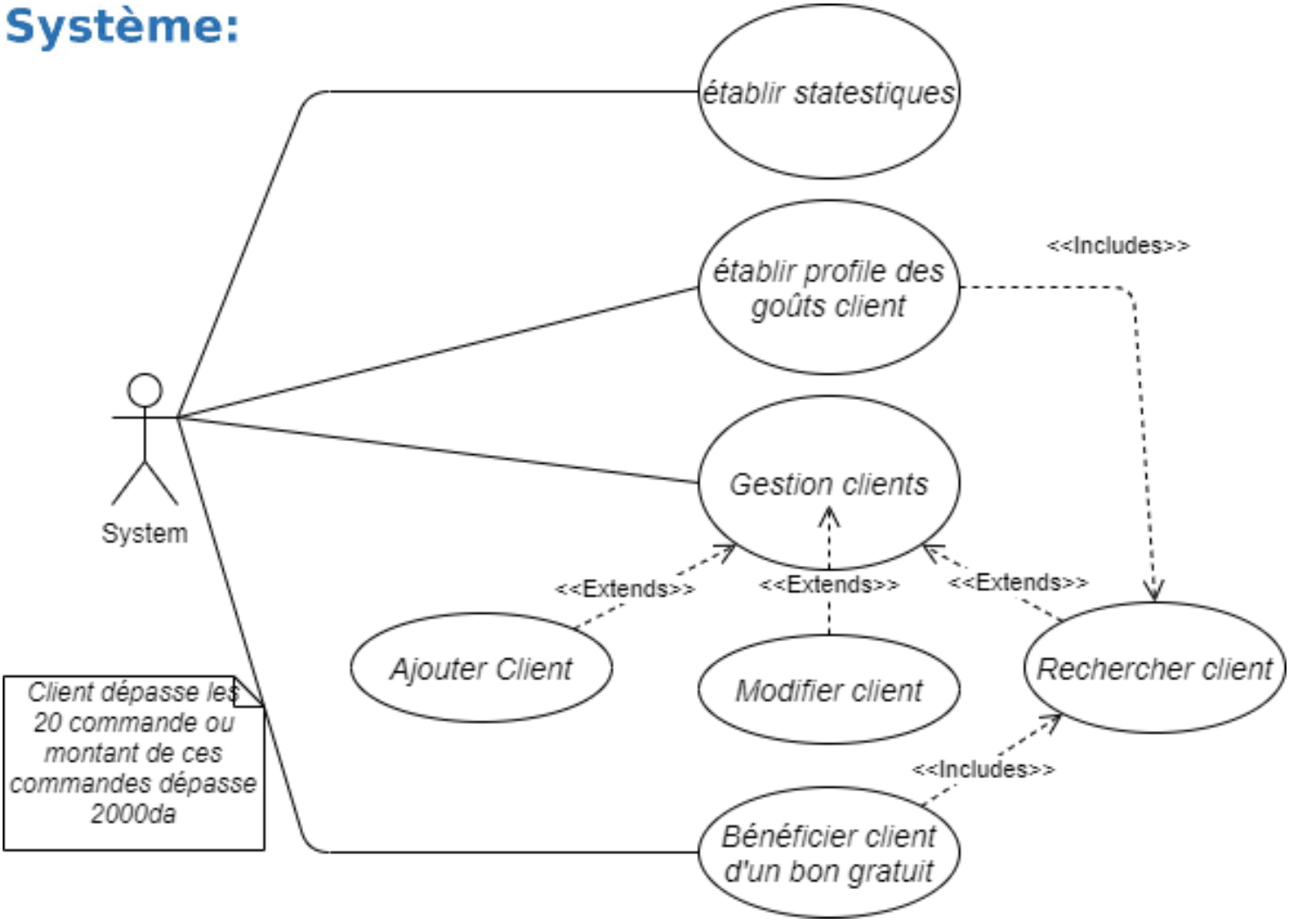
Client:

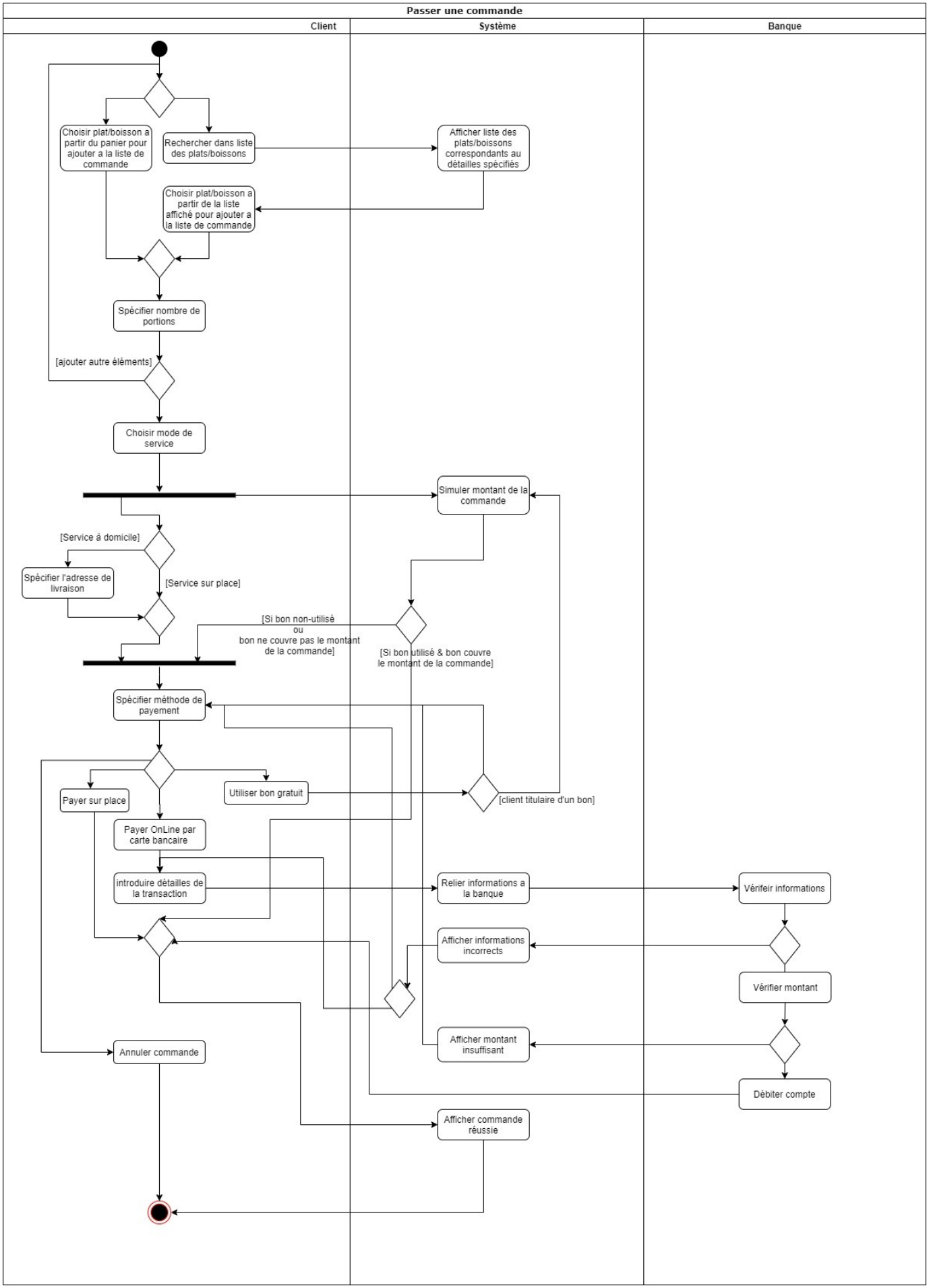


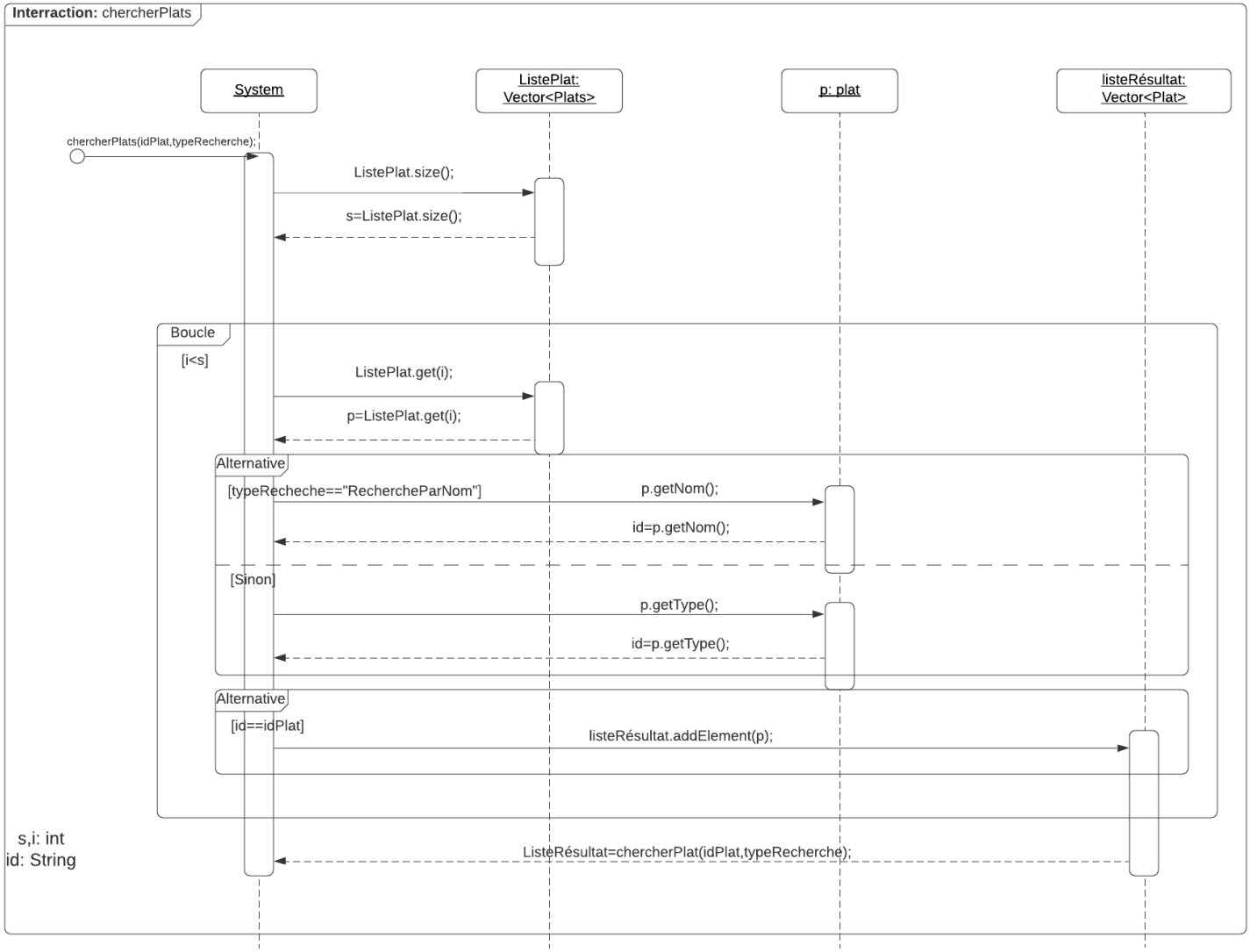
Chef/Employé:

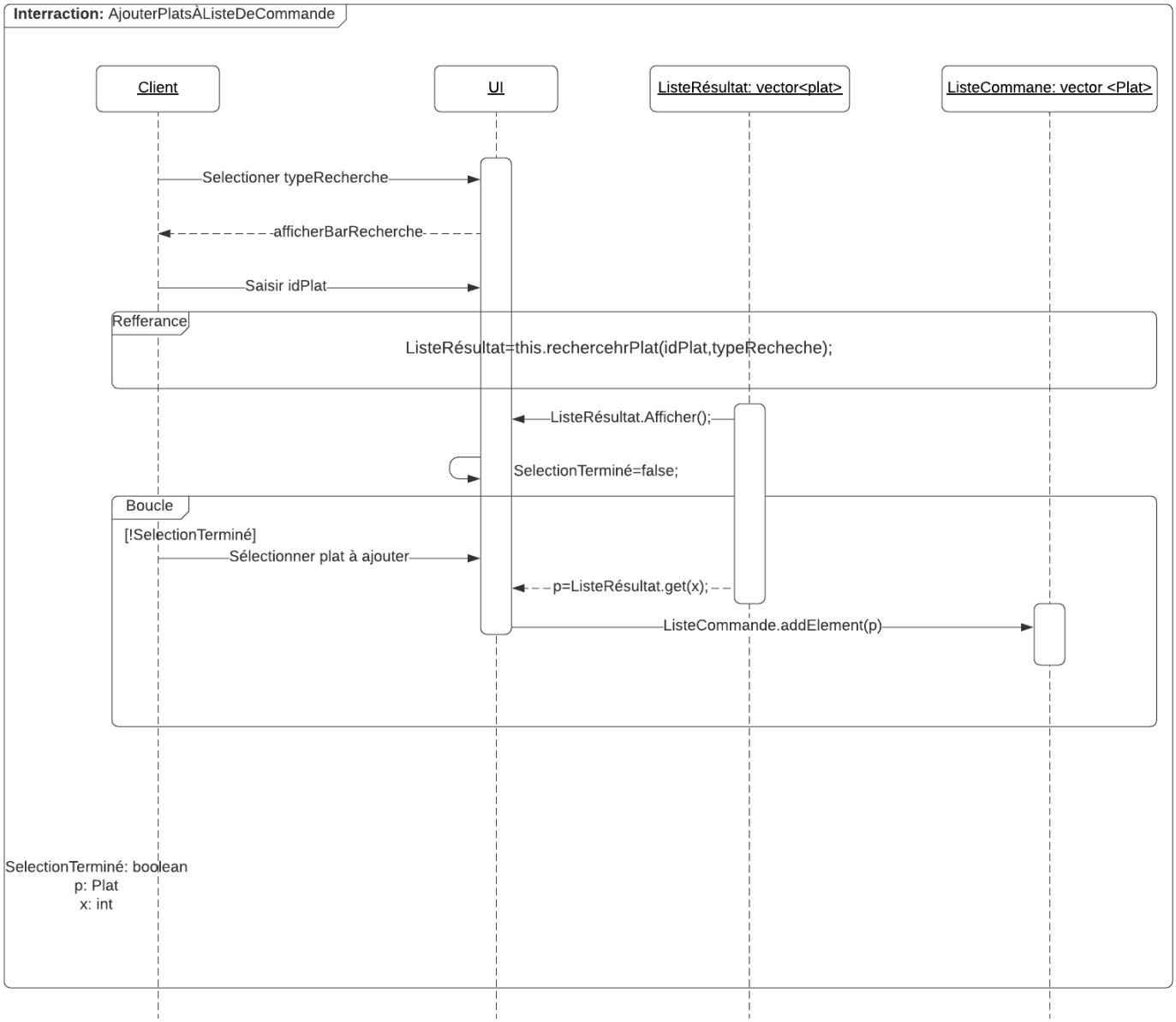


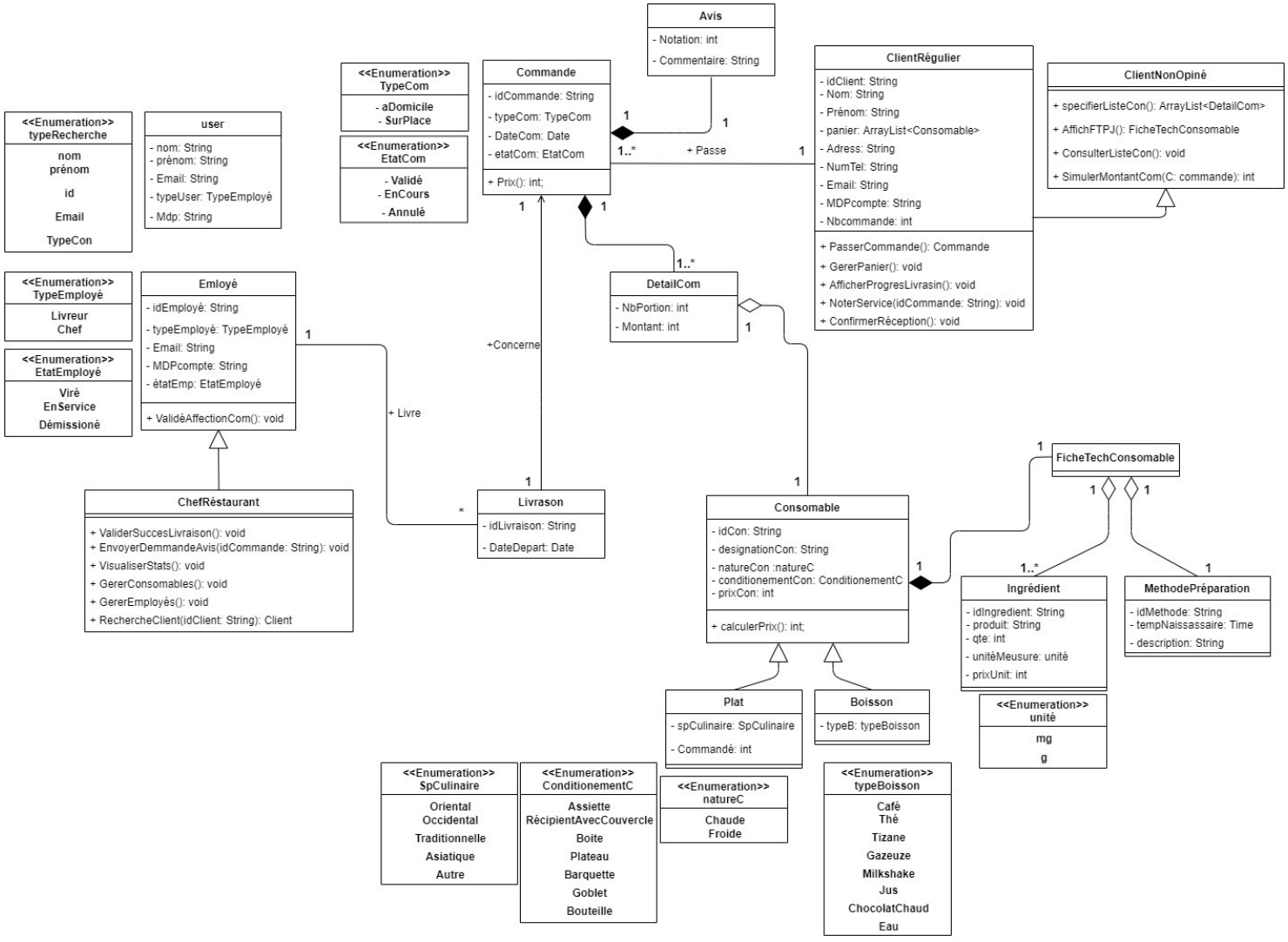
Système:

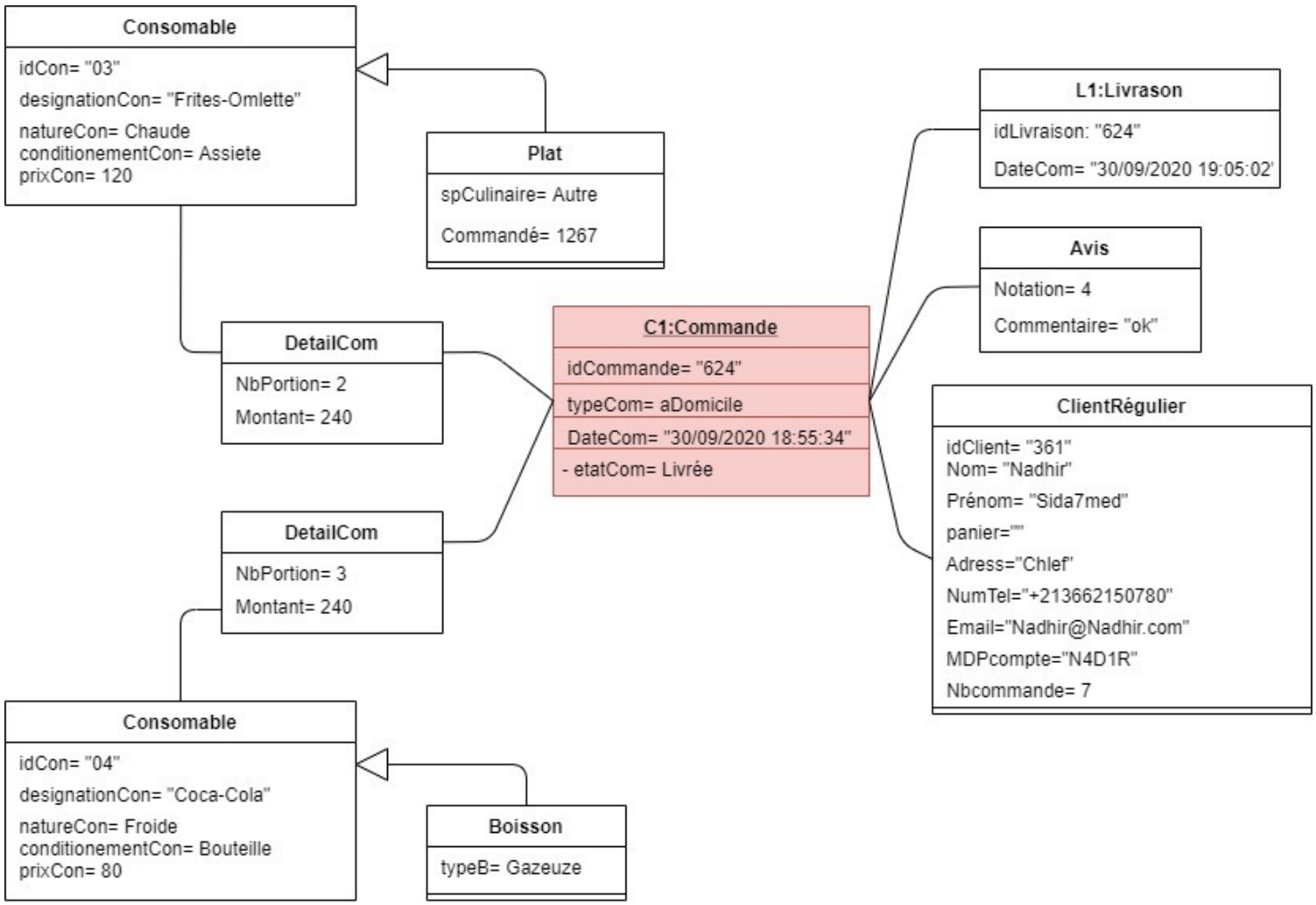

établir statestiques
établir profile des goûts client
<<Includes>>
System
Gestion clients
<<Extends>>
<<Extends>>
<<Extends>>
Ajouter Client
Modifier client
Rechercher client
Client dépasse les 20 commande ou montant de ces commandes dépasse 2000da
<<Includes>>
Bénéficier client d'un bon gratuit


Passer une commande
Client
Système
Banque
Choisir plat/boisson a partir du panier pour ajouter a la liste de commande
Rechercher dans liste des plats/boissons
Afficher liste des plats/boissons correspondants au détailles spécifiés
Choisir plat/boisson a partir de la liste affiché pour ajouter a la liste de commande
Spécifier nombre de portions
[ajouter autre éléments]
Choisir mode de service
Simuler montant de la commande
[Service à domicile]
Spécifier l'adresse de livraison
[Service sur place]
[Si bon non-utilisé ou bon ne couvre pas le montant de la commande]
[Si bon utilisé & bon couvre le montant de la commande]
Spécifier méthode de payement
Payer sur place
Utiliser bon gratuit
[client titulaire d'un bon]
Payer OnLine par carte bancaire
introduire détailles de la transaction
Relier informations a la banque
Vérifeir informations
Afficher informations incorrects
Vérifier montant
Afficher montant insuffisant
Débiter compte
Annuler commande
Afficher commande réussie

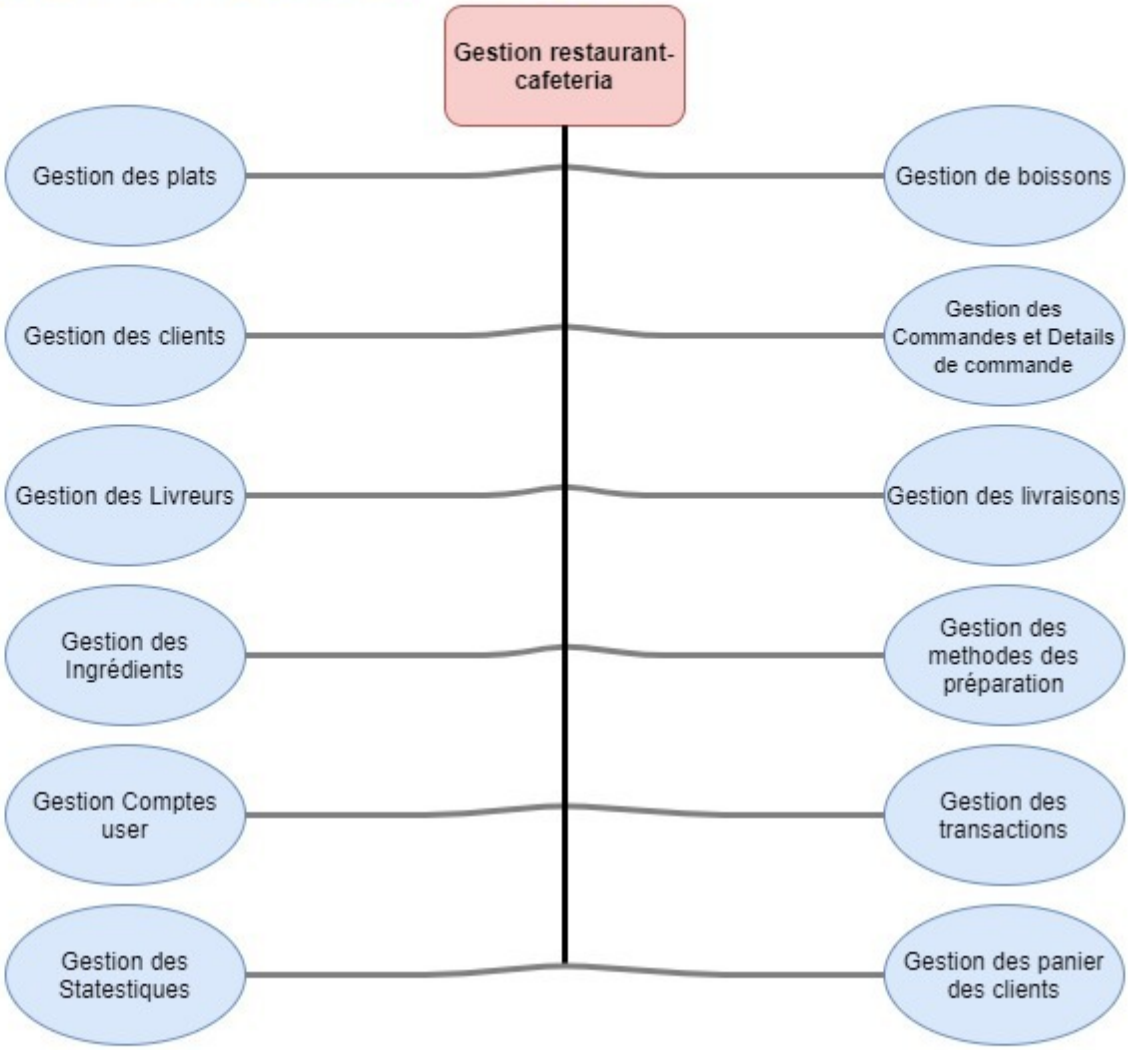

Interraction: chercherPlats
System
ListePlat: Vector<Plats>
p: plat
listeRésultat: Vector<Plat>
chercherPlats(idPlat,typeRecherche);
ListePlat.size();
s=ListePlat.size();
Boucle
[i<s]
ListePlat.get(i);
p=ListePlat.get(i);
Alternative
[typeRecheche=="RechercheParNom"]
p.getNom();
id=p.getNom();
[Sinon]
p.getType();
id=p.getType();
Alternative
[id==idPlat]
listeRésultat.addElement(p);
s,i: int
id: String
ListeRésultat=chercherPlat(idPlat,typeRecherche);


Interraction: AjouterPlatsÀListeDeCommande
Client
UI
ListeRésultat: vector<plat>
ListeCommane: vector <Plat>
Selectioner typeRecherche
afficherBarRecherche
Saisir idPlat
Refferance
ListeRésultat=this.rechercehrPlat(idPlat,typeRecheche);
ListeRésultat.Afficher();
SelectionTerminé=false;
Boucle
[!SelectionTerminé]
Sélectionner plat à ajouter
p=ListeRésultat.get(x);
ListeCommande.addElement(p)
SelectionTerminé: boolean
p: Plat
x: int

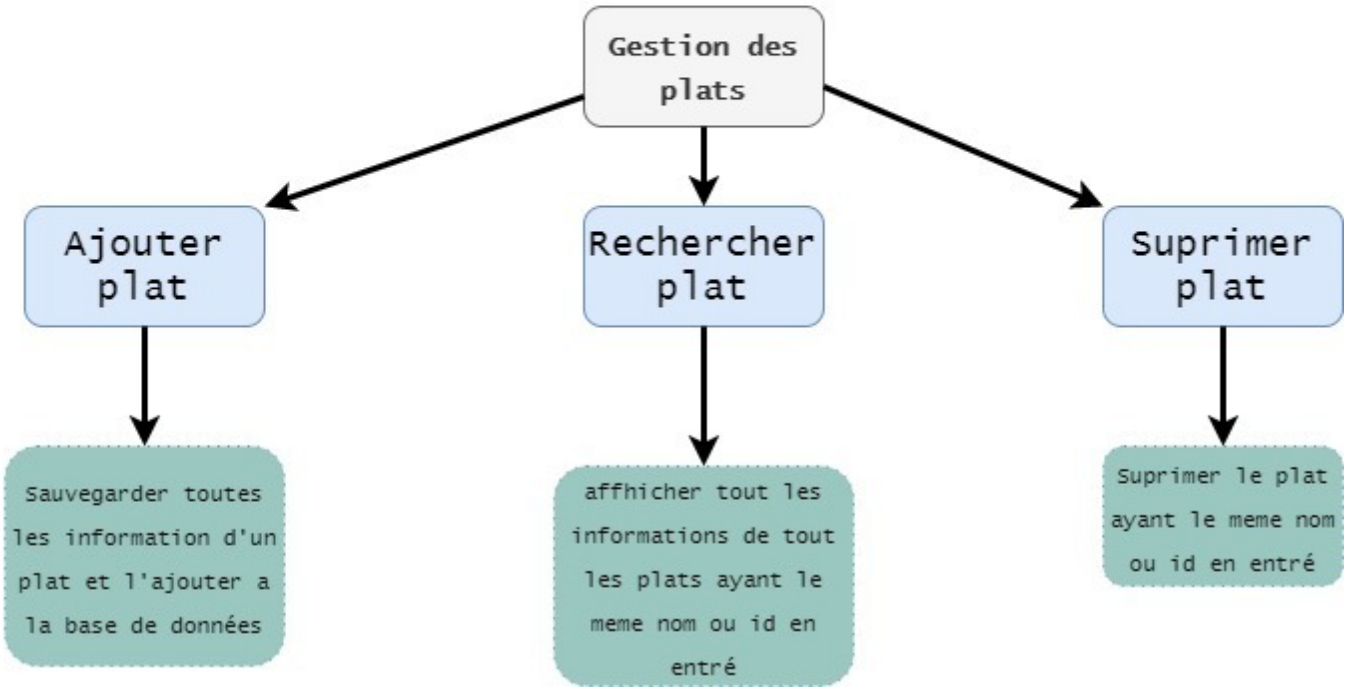

Avis
- Notation: int
- Commentaire: String
<<Enumeration>>
TypeCom
- aDomicile
- SurPlace
<<Enumeration>>
EtatCom
- Validé
- EnCours
- Annulé
Commande
- idCommande: String
- typeCom: TypeCom
- DateCom: Date
- etatCom: EtatCom
+ Prix(): int;
ClientRégulier
- idClient: String
- Nom: String
- Prénom: String
- panier: ArrayList<Consomable>
- Adress: String
- NumTel: String
- Email: String
- MDPcompte: String
- Nbcommande: int
+ PasserCommande(): Commande
+ GererPanier(): void
+ AfficherProgresLivrasin(): void
+ NoterService(idCommande: String): void
+ ConfirmerRéception(): void
ClientNonOpiné
+ specifierListeCon(): ArrayList<DetailCom>
+ AffichFTPJ(): FicheTechConsomable
+ ConsulterListeCon(): void
+ SimulerMontantCom(C: commande): int
<<Enumeration>>
typeRecherche
nom
prénom
id
Email
TypeCon
user
- nom: String
- prénom: String
- Email: String
- typeUser: TypeEmployé
- Mdp: String
1
1..*
+ Passe
1
1
1
1..*
DetailCom
- NbPortion: int
- Montant: int
1
+Concerne
<<Enumeration>>
TypeEmployé
Livreur
Chef
<<Enumeration>>
EtatEmployé
Viré
EnService
Démissioné
Emloyé
- idEmployé: String
- typeEmployé: TypeEmployé
- Email: String
- MDPcompte: String
- étatEmp: EtatEmployé
+ ValidéAffectionCom(): void
1
+ Livre
ChefRéstaurant
+ ValiderSuccesLivraison(): void
+ EnvoyerDemmandeAvis(idCommande: String): void
+ VisualiserStats(): void
+ GererConsomables(): void
+ GererEmployés(): void
+ RechercheClient(idClient: String): Client
*
1
Livrason
- idLivraison: String
- DateDepart: Date
1
Consomable
- idCon: String
- designationCon: String
- natureCon :natureC
- conditionementCon: ConditionementC
- prixCon: int
+ calculerPrix(): int;
1
1
FicheTechConsomable
1
1
1..*
1
Ingrédient
- idIngredient: String
- produit: String
- qte: int
- unitéMeusure: unité
- prixUnit: int
MethodePréparation
- idMethode: String
- tempNaissassaire: Time
- description: String
<<Enumeration>>
unité
mg
g
Plat
- spCulinaire: SpCulinaire
- Commandé: int
Boisson
- typeB: typeBoisson
<<Enumeration>>
SpCulinaire
Oriental
Occidental
Traditionnelle
Asiatique
Autre
<<Enumeration>>
ConditionementC
Assiette
RécipientAvecCouvercle
Boite
Plateau
Barquette
Goblet
Bouteille
<<Enumeration>>
natureC
Chaude
Froide
<<Enumeration>>
typeBoisson
Café
Thé
Tizane
Gazeuze
Milkshake
Jus
ChocolatChaud
Eau


Consomable
idCon= "03"
designationCon= "Frites-Omlette"
natureCon= Chaude
conditionementCon= Assiete
prixCon= 120
Plat
spCulinaire= Autre
Commandé= 1267
L1:Livrason
idLivraison: "624"
DateCom= "30/09/2020 19:05:02'
Avis
Notation= 4
Commentaire= "ok"
DetailCom
NbPortion= 2
Montant= 240
C1:Commande
idCommande= "624"
typeCom= aDomicile
DateCom= "30/09/2020 18:55:34"
- etatCom= Livrée
ClientRégulier
idClient= "361"
Nom= "Nadhir"
Prénom= "Sida7med"
panier=""
Adress="Chlef"
NumTel="+213662150780"
Email="Nadhir@Nadhir.com"
MDPcompte="N4D1R"
Nbcommande= 7
DetailCom
NbPortion= 3
Montant= 240
Consomable
idCon= "04"
designationCon= "Coca-Cola"
natureCon= Froide
conditionementCon= Bouteille
prixCon= 80
Boisson
typeB= Gazeuze

Conception Architectural du System :



Conception Détaillé (Gestion des plats) :



Les Méthodes de relation entre classes :

Classe ClientRégulier :

```
// methodes de la relation avec commande
public void ajouterCommande(Commande commande){
    if(!this.getCommandes().contains(commande)){
        commandes.add(commande);
        commande.setClientRégulier(this);
        nbCommande++;
    }
}
public void AnullerCommande(Commande commande){
    this.commandes.remove(commande);
    commande.setClientRégulier(null);
    nbCommande--;
}
```

Classe Employé :

```
    //Methodes Livreur
    public void ajouterLivraison(Livraison livraison){
        if(!this.livraisons.contains(livraison)){
            this.livraisons.add(livraison);
            livraison.setEmployé(this);
        }
    }
    public void anulerLivraison(Livraison livraison){
        if(this.livraisons.remove(livraison)){
            livraison.setEmployé(null);
        }
    }
```

Classe Commande :

```
//ClientRégulier:
public void ajouterClient(ClientRégulier clientRégulier){
    this.clientRégulier=clientRégulier;
    if(!clientRégulier.getCommandes().contains(this)){
        clientRégulier.getCommandes().add(this);
    }
}
public void suprimerClient(){
    if(this.clientRégulier.getCommandes().remove( o: this)){
        this.clientRégulier=null;
    }
}
```

Classe DétailCom :

```
public void AjouterCon(Consomable consomable,int i){
    this.consomable=consomable;
    this.setNbportion(i);
    this.montant=this.Nbportion*consomable.getPrixCon();
}
public  void suprimerCon(){
    this.consomable=null;
    this.setNbportion(0);
    this.setMontant(0);
}
```

Classe Livraison:

```
// Methode pour la relation avec un Employé:
    public void setEmployé(Employé employé) { this.employé = employé; }
    public void ajouterLivreur(Employé employé){
        this.setEmployé(employé);
        if (!employé.getLivraisons().contains(this))employé.getLivraisons().add(this);
    }
    public void anulerLivraison(){
        this.employé.getLivraisons().remove( o: this);
        this.employé=null;
    }
    // Methode pour la relation avec un Commande:
    public void ajouterCommande(Commande commande) { this.setCommande(commande); }
    public void suprimmerCommande(){
        this.setCommande(null);
        this.setEmployé(null);
    }
```

Classe FicheTechCon :

```
//Methode pour la relation avec une methode de préparation
public void modifierMethode(MethodePréparation methodePréparation){
    this.methodePréparation=methodePréparation;
}
//Methodes pour la relation avec un Ingrédient
public void ajouterIngredient(Ingrédient ingrédient){
    if(!this.getIngrédients().contains(ingrédient)){
        this.getIngrédients().add(ingrédient);
    }
}
public void suprimerIngredient(Ingrédient ingrédient) { this.getIngrédients().remove(ingrédient); }
```