



Conception

Projet GLA

Room reservation

Qiwei XIAN

Kelun CHAI

Zixi CHEN

Sommaire

1. Introduction

- 1.1 Objectif
- 1.2 Portée du projet
- 1.3 Définitions
- 1.4 Vue d'ensemble

2. Description générale

- 2.1 Perspective du produit
- 2.2 Fonctionnalités de logiciel
- 2.3 Exigences fonctionnelles
 - Gestion des demandes (Ajouter, Modifier, Supprimer, Consulter)
 - Gestion des clients (Ajouter, Modifier)
 - Rechercher les ressources
 - Gestion des ressources (Ajouter, Modifier, Supprimer)
 - Gestion de l'inventaire (Ajouter, Modifier, Supprimer)
 - Gestion des carte crédit (Ajouter, Modifier, Supprimer)
 - Paieement en ligne
 - L'émission des factures
- 2.4 Caractéristiques de l'utilisateur
 - 2.4.1 Client
 - 2.4.2 Gestionnaire
- 2.5 Contraintes
 - 2.5.1 Hardware
 - 2.5.2 Software
 - 2.5.3 Langue de haut niveau

3. Exigence spécifique

- 3.1 Exigence fonctionnelle
 - 3.1.1 Contraintes de ajouterDemande
 - 3.1.2 L'algorithme
 - 3.1.3 Pseudocode de accepteDemande
- 3.2 Exigence Non-fonctionnelle
 - 3.2.1 Sécurité
 - 3.2.2 Autre
- 3.3 Diagramme UML
- 3.4 Interface externe
 - 3.3.1 Interface Client
 - 3.3.2 Interface Gestionnaire
- 3.4 Diagramme des cas utilisations

3.4.1 Authentification

3.4.2 Gestion des demandes

3.4.3 Gestion des clients

3.4.4 Gestion des ressources

3.4.5 Gestion de l'inventaire

3.4.6 Gestion des carte crédit

3.4.7 Paiement en ligne

3.4.8 L'émission de facture

3.5 Les diagramme de séquence

3.5.1 Authentification

3.5.2 Gestion des réservations

3.5.2.1 findAll(Consultation)

3.5.2.2 ajouterDemande

3.5.2.3 modifierDemande

3.5.2.4 supprimerDemande

3.5.3 Gestion des ressources

3.5.3.1 ajouterClient

3.5.3.2 modifierClient

3.5.3.3 supprimerClient

3.5.3.4 ajouterRessources

3.5.3.5 modifierRessources

3.5.3.5 supprimerRessources

3.5.3.6 ajouterMeuble

3.5.3.7 modifierMeuble

3.5.3.8 supprimerMeuble

3.6 Diagramme d'activité

3.6.1 Consultation

3.6.2 Gestion des ressources

3.6.3 Gestion des clients

3.6.4 Gestion des réservations

4. Modèle Relationnel

4.1 Diagramme de classe

4.2 Règle de passage au modèle logique de donnée relationnel

4.3 Le modèle relationnel des données de l'application

1. Introduction

Le **SGH** (**S**ystème de la **G**estion **H**ôtelière) est un système qui joue un rôle très important dans le management d'un hôtel moderne, car son efficacité et la possibilité de réserver une ressource en ligne permet de garantir un bon fonctionnement pour une hôtel, Ce système possède 2 côtés, le côté administration permet aux administrateurs de gérer les informations de clients et réservation, autre côté permet aux utilisateurs(les clients) d'effectuer une réservation et finalement un paiement.

1.1 Objectif

La spécification des exigences logicielles (Software Requirements Specification (SRS)) fournira une description détaillée des exigences du Système de la Gestion Hôtelière (SGH) . Ce SGH permettra de comprendre facilement ce que l'on attend du nouveau système qui doit être mis en place. La compréhension claire du système et de ses fonctionnalités nous permettra de développer correctement le logiciel pour l'utilisateur final et le maintenir. SRS est la base de projet. A partir de SRS, SGH peut être conçu, construit et finalement testé.

SRS sera utilisé par l'équipe de développement qui construit le SGH et les utilisateurs finaux de l'hôtel. L'équipe de projet utilisera le SRS pour bien comprendre les attentes de ce SGH et construire correctement le logiciel. Les utilisateurs finaux de l'hôtel pourront utiliser SRS comme «black box test» pour déterminer si le système s'est bien conformé à leurs attentes. Si ce n'est pas conforme à leurs attentes, les utilisateurs finaux peuvent spécifier ce qui ne leur plaît pas et l'équipe modifiera le SRS pour répondre aux besoins des utilisateurs finaux.

1.2 Portée du projet

Ce logiciel sera implémenté pour la gestion quotidienne de l'hôtel. Il est séparé en deux interfaces, l'interface de gestionnaires et celle de client. Les gestionnaires ont le droit d'utiliser toutes les fonctionnalités. Néanmoins, les clients sont seulement autorisés à gérer leur information personnelle et leur demandes.

1.3 Définitions

SGH Système de la Gestion Hôtelière

SRS Software Requirements Specification

1.4 Vue d'ensemble

SRS est organisé en deux parties, la première est la description générale et la deuxième section, l'exigence spécifique.

La description générale décrira les exigences du **Système de Gestion Hôtelière**.

SRS décrit les détails du système.

2. Description générale

2.1 Perspective du produit

Le système de gestion hôtelière est un nouveau produit logiciel qui sera produit par l'équipe du projet afin de résoudre les problèmes rencontrés en raison du système manuel actuel.

Le système nouvellement introduit fournira un accès facile au système et contiendra des fonctions conviviales.

2.2 Fonctionnalités de logiciel

- Gestion des demandes (Ajouter, Modifier, Supprimer, Consulter)
- Gestion des clients (Ajouter, Modifier)
- Rechercher les ressources
- Arranger le table de demande en fonction de bénéfice
- Gestion des ressources (Ajouter, Modifier, Supprimer)
- Gestion de l'inventaire (Ajouter, Modifier, Supprimer)
- Gestion des carte crédit (Ajouter, Modifier, Supprimer)
- Paiement en ligne
- L'émission des factures

2.3 Exigences fonctionnelles

- Gestion des demandes (Ajouter, Modifier, Supprimer, Consulter)

Fonction 1	ajouterDemande	class Personne
Entree	new Demande (DateCheckin,DateCheckout,typeRessource)	
Sortie	True pour que la demande a été ajoutée dans le table des demandes avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et enregistrez les informations de la nouvelle demande dans la base de données	

Fonction 2	accepteDemande	class Gestionnaire
Entree	Demande	
Sortie	True pour que la demande a été acceptée avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et modifier le statut de demande dans la base de données	

Fonction 3	modifierDemande	class Personne
Entree	Demande	
Sortie	True pour que la demande a été modifiée dans le table des demandes avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et modifier les informations de demande dans la base de données	

Fonction 4	supprimerDemande	class Personne
Entree	Demande	
Sortie	True pour que la demande a été supprimée dans le table des demandes avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et supprimer les informations de demande dans la base de données	

Fonction 5	findAll	class Personne
Entree		
Sortie	Toutes les demandes de cette Personne	
Workflow	Retourner toutes les demandes de cette personne	

Fonction 6	findAll	class Gestionnaire
Entree	Client	
Sortie	Toutes les demandes de cette Client si ce client existe dans le table des clients. Sinon null	
Workflow	Retourner toutes les demandes de cette client	

Gestion des clients (Ajouter, Modifier)

Fonction 7	AjouterClient	class Gestionnaire
Entree	new Client (userName, motDePasse, nom, email, telephone, adresse)	
Sortie	True pour que le client a bien été ajouté dans le table des clients avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables données et enregistrez les informations de nouveau client dans la base de données	

Fonction 8	modifierClient	class Personne
Entree	Client	
Sortie	True pour que le client a été modifié dans le table des clients avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et modifier les informations de client	

- Rechercher les ressources

Fonction 9	rechercheChambre	class Personne
Entree	dateCheckIn, dateCheckOut, typeRessrouce	
Sortie	Chambre disponible ou null	
Workflow	Valider les variables données et vérifier si il y a des chambres disponibles dans une période donnée et retourner-les. sinon retourner null	

- Gestion des ressources (Ajouter, Modifier, Supprimer)

Fonction 10	ajouterRessource	class Gestionnaire
Entree	Ressoure	
Sortie	True pour que la ressource a été ajoutée dans le table des ressources avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les détails données et enregistrer les informations de ressource dans la base de données	

Fonction 11	modifierRessource	class Gestionnaire
Entree	Ressource(numChambre, price, type, taille, listMeuble)	
Sortie	True pour que la ressource a été modifiée dans le table des ressources avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les détails données et remplacer les informations de ressource par celle de paramètre dans la base de données	

Fonction 12	supprimerRessource	class Gestionnaire
Entree	numChambre	
Sortie	True pour que la ressource a été supprimée dans le table des ressources avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les détails données et supprimer les informations de ressource en fonction de numéro de chambre dans la base de données	

- Gestion de l'inventaire (Ajouter, Modifier, Supprimer)

Fonction 13	ajouterMeuble	class Gestionnaire
Entree	Meuble, Ressource	
Sortie	True pour que la meuble a été ajoutée dans le table des meuble de la ressource avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et modifier la liste des meubles de ressource dans la base de données	

Fonction 14	modifierMeuble	class Gestionnaire
Entree	Meuble, Ressource	
Sortie	True pour que la meuble a été modifiée dans le table des meuble de la ressource avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et modifier la meuble dans la base de données	

Fonction 15	supprimerMeuble	class Gestionnaire
Entree	Meuble, Ressource	
Sortie	True pour que la meuble a été supprimée dans le table des meuble de la ressource avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables des paramètres et supprimer la meuble dans la base de données	

- Gestion des carte crédit (Ajouter, Modifier, Supprimer)

Fonction 16	ajouterCarteCredit	class Client
Entree	new CarteCredit(numCarte, DateExp, nom, CVV)	
Sortie	True pour que la carte crédit a été ajoutée dans la liste de carte du client avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables données et ajouter les informations de carte crédit dans la base de données	

Fonction 17	modifierCarteCredit	
Entree	UserID, CarteCredit	
Sortie	True pour que la carte crédit a été remplacé par celle de paramètre dans la liste de carte du client avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables données et modifier les informations de carte crédit dans la base de données	

Fonction 18	supprimerCarteCredit	class Client
Entree	UserID, numCarte	
Sortie	True pour que la carte crédit a été supprimée par celle de paramètre dans la liste de carte du client avec succès, false sinon, puis afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès	
Workflow	Valider les variables données et supprimer les informations de carte crédit dans la base de données	

- Paiement en ligne

Fonction 19	payer	class Client
Entree	CarteCredit, montant	
Sortie	<p>si le paiement a été implémenté avec succès, rendre new Paiement (compteTransactionA, compteTransactionB, dateTransaction, devise, montant) et modifier le solde de la carte utilisé</p> <p>sinon rendre null, finalement afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès</p>	
Workflow	Valider les variables données et mettre à jour le solde de la carte utilisé du client dans la base de données	

Fonction 20	rembourser	class Gestionnaire
Entree	CarteCredit	
Sortie	<p>si le paiement a été implémenté avec succès, rendre new Paiement (compteTransactionA, compteTransactionB, dateTransaction,devise,montant) et modifier le solde de la carte utilisé</p> <p>sinon rendre null, finalement afficher une fenêtre contextuelle montrant que la base de données a été mise à jour avec succès</p>	
Workflow	Valider les variables données et mettre à jour le solde de la carte utilisé du client dans la base de données	

- L'émission des factures

Fonction 20	emissionFacture	class Paiement
Entree		
Sortie	Facture detaille	
Workflow	retourner une facture en fonction de l'information de paiement	

2.4 Caractéristiques de l'utilisateur

2.4.1 Client

Avant d'utiliser le système, les clients doivent entrer leur nom, sexe, âge, société et adresse pour s'enregistrer. Une fois l'inscription réussie, un numéro de client (`nClient`) sera attribué au client. Le client entre une période de temps prédéterminée (`DateCheckin`, `DateCheckout`), le nombre d'adultes et le nombre d'enfants pour faire une réservation.

Une fois que le client a sélectionné la chambre, il sera payé en entrant le mode de paiement (informations de carte de crédit). Les clients verront une facture détaillée et complète avant de débiter.

Les clients peuvent modifier ou annuler une réservation en prenant contact avec le gestionnaire.

2.4.2 Gestionnaire

Le gestionnaire est responsable de la gestion des ressources disponibles dans le système de gestion de l'hôtel. Le gestionnaire bénéficie également de la plupart des privilèges mentionnés ci-dessus, à l'exception de ce qui concerne le traitement des paiements.

L'utilisation du gestionnaire a pour but de réduire la charge de travail du propriétaire qui ne peut pas être affectée à la réceptionniste, ces tâches semblant très responsables.

Le gestionnaire dispose d'autres capacités, telles que l'ajout de nouveaux membres du personnel au système, leur modification ou leur suppression, l'ajout de nouveaux clients au système, leur modification et leur suppression du système. Ajouter de nouveaux types de chambre au système, les modifier et les supprimer.

2.5 Contraintes

Nous fournis nos meilleurs efforts dans le développement du système. Afin de maintenir la fiabilité et la durabilité du système, certaines contraintes de conception et de mise en œuvre sont appliquées. Le système aura besoin d'une mémoire minimale de 512 Mo. Mais il est recommandé d'avoir une mémoire de 1 Go. Lors de la conception des interfaces de système, nous avons la possibilité de travailler avec de nouveaux outils tels que Django Framework.

2.5.1 Hardware

1. Système prend en charge tous les systèmes d'exploitation connus, tels que Windows, MacOS et Linux
2. Ordinateur 512 Mo + RAM, moniteur avec une résolution minimale de 1024x768, clavier et souris
3. Le disque dur doit être au système de fichiers NTFS et formaté avec au moins 10 Go d'espace libre
4. Une imprimante devra être utilisée pour imprimer ces rapports et notes

2.5.2 Software

1. Le système est conçu pour fonctionner sur n'importe quelle plate-forme à l'aide d'un navigateur Internet tel que Chrome.

2.5.3 Langue de haut niveau

1. framework Django (Python)
2. HTML, mySQL

3. Exigence spécifique

3.1 Exigence fonctionnelle

- Gestion des demandes (Ajouter, Modifier, Supprimer, Consulter)
- Gestion des clients (Ajouter, Modifier)
- Rechercher les ressources
- Arranger le table de demande en fonction de bénéfice
- Gestion des ressources (Ajouter, Modifier, Supprimer)
- Gestion de l'inventaire (Ajouter, Modifier, Supprimer)
- Gestion des carte crédit (Ajouter, Modifier, Supprimer)
- Paiement en ligne
- L'émission des factures

3.1.1 Contraintes de ajouterDemande

Ajouter la demande dans le table est une difficulté de la gestion. Il faut résoudre les conflits des demandes ayant les même besoins

(Leur périodes de séjour ont le conflit, les chambres ne sont plus disponibles, etc)

3.1.2 L'algorithme

On considère les demandes comme une forme conjonctive. (i.e. $d_{11} \cdot d_{21} \cdot (d_{31} \vee d_{32}) \cdot (d_{41} \vee d_{42}) \dots d_{n1}$), chaque clause est un seul littéral ou une forme disjonctive, le littéral est une demande. True pour un littéral ça veut dire que l'on accepte cette demande false sinon. On met d'abord $d_{11} \leftarrow \text{true}$, s'il n'y a pas conflit, on prend d_{21} , jusqu'à le conflit est apparu.

(On suppose que d_{ij} a conflit) Si d_{ij} est dans DNF, alors on passe d_{ij} et essaie à accepter $d_{i,j}+1$, si tous les littéraux de cette clause ont le conflit, alors il n'y a pas de solution pour l'instant, on doit rendre à $d_{(i-1)j}$, et reprend $d_{(i-1)(j+1)}$, etc.

Si pour tous les combinatoire il y a le conflit, alors il n'y a pas de solution, on doit abandonner une demande qui cause le moins d'influence.

3.1.3 Pseudocode de accepteDemande

par exemple, il y a demandes:

$$(d_{11} \vee d_{12} \vee d_{13}) \cdot (d_{21}) \cdot (d_{31} \vee d_{32}) \cdot (d_{41} \vee d_{42} \vee d_{43} \vee d_{44}) \cdot (d_{51})$$

allDemandes =

$$[[d_{11}, d_{12}, d_{13}], [d_{21}], [d_{31}, d_{32}], [d_{41}, d_{42}, d_{43}, d_{44}], [d_{51}]]$$

Pseudocode

```

acceptDemande(d: Demande)
    demandes = this.allDemandes
    d.status = "confirm"
    conflit = hasConflit(demandes, d)
    if not conflit:
        add d into demandes
        this.allDemandes = demandes
        return true
    else:
        demandes = init(demandes)
        stack = new stack()
        stack_pos = new stack()
        tmp = demandes
        i = j = 0
        while i < length demandes:
            next = false
            while j < length tmp[i] and not next:
                tmp[i][j].status = "confirm"
                conflit = hasConflit(tmp, d)
                if not conflit:
                    add tmp into stack
                    add [i,j] into stack_pos
                    next = true
                else:
                    tmp [i][j].status =
"attendu"
                    j = j + 1
            if next:
                if i == (length demandes) -1:
                    result = unstack stack

```

```

        add d into result

        this.allDemandes = result
        return true
    else:
        i++; j=0
else:
    if stack_cor == []:
        return false
    else:
        i,j = unstack stack_cor
        tmp = unstack stack
        tmp[i][j] = "attendu"
        j++

return false

```

Si `ajouteDemande(d)` est **true**,

`allDemandes` **devient** `[[d11, d12, d13], [d21], [d31, d32], [d41, d42, d43, d44], [d51], [d]]`

Sinon, on n'ajoute pas pour l'instant, il faut comparer les valeurs de chaque demandes, pour décider quelle demande qu'on doit abandonner.

3.2 Exigence Non-fonctionnelle

3.2.1 Sécurité

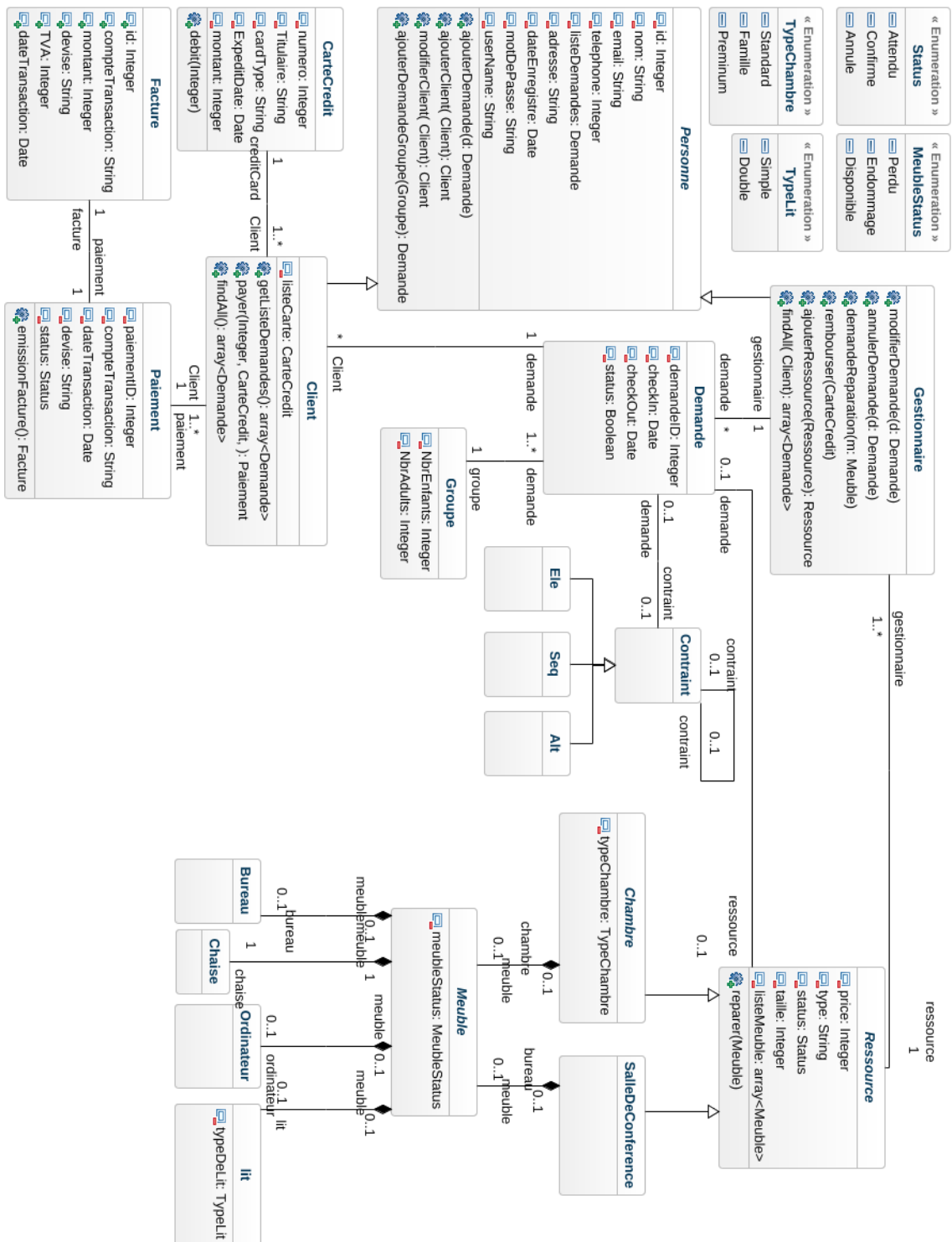
Il existe plusieurs niveaux d'utilisateur dans le **Système de Gestion Hôtelière**. L'accès aux différents sous-systèmes sera protégé par un nom d'utilisateur et un mot de passe. Cela donne

différentes vues et fonctions accessibles selon les niveaux d'utilisateur à travers le système.

3.2.2 Autre

Les gestionnaires pourront se connecter au système de gestion hôtelière. Le client aura accès à la réservation et aux sous-systèmes. Les gestionnaires auront accès au sous-système de gestion ainsi qu'aux sous-systèmes de réservation. L'accès aux différents sous-systèmes sera protégé par un nom d'utilisateur et un mot de passe.

3.3 Diagramme UML



3.4 Interface externe


3.3.1 Interface Client

L'interface de **connexion** permet de se connecter au système en utilisant le nom d'utilisateur et le mot de passe de trois utilisateurs différents.



A screenshot of a login interface. At the top, the text "MEMBRE ?" is centered. Below it are two input fields: the first is labeled "username" and the second is labeled "mot de passe". At the bottom, there is a button labeled "Connexion".

Inscription: Cette page est utilisée pour les inscriptions de compte. Les informations seront enregistrées dans la base de données.



A screenshot of a registration interface. It contains five input fields stacked vertically: "Nom", "Prenom", "E-mail", "Mot de passe", and "Confirmer mot de passe". At the bottom, there is a button labeled "Inscription".

Réservation: Cette page est utilisée par les clients pour rechercher les ressources dont ils ont besoin.

Lieu

Check-in Date

Check-out Date

Bed Type

Adults

Enfants

Recherche

findAll(Consultation des commandes): Les clients recherchent toutes les commandes de leur compte.

Trouver Commades

N Client

From

To

Find

Résultats de la requête (afficher les commandes existantes)

Commades			
N de commande	Date Checkin	Date Checkout	Nb de personne
C190321	22/03/2019	31/03/2019	2
C190403	03/04/2019	04/04/2019	1

Ajouter Carte Crédit: Les clients ajoutent une carte de crédit comme moyen de paiement

Moyen de Paiement

Type de Carte

☒ Visa ☐ Mastercard ☐ CB ☐ Amex

Titulaire de Carte

Numero de Carte

Date d'expiration

CVV

Valider

3.3.2 Interface Gestionnaire

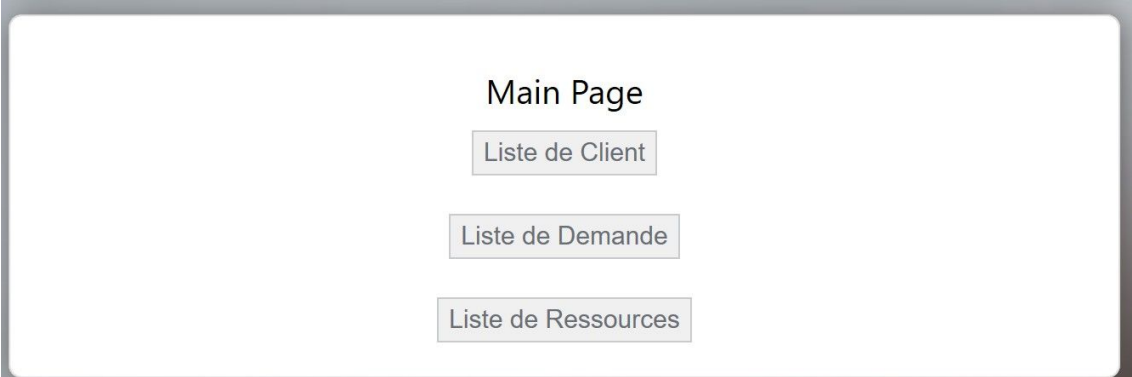
Interface de login, permet aux différentes utilisateurs à connecter sur en authentifiant avec Login et Mot de passe.



Bienvenu sur l'interface Administrateur de
HOTEL XXX

Login :

Mot de passe :



Main Page

L'interface de liste de Client est presente en premier lieu.

Liste de Client						
Nom	Prenom	email	Tel	adresse	password	date Enreg
表中无内容						
<div>Ajouter Client</div> <div>Modifier</div> <div>Voir son Commandes</div>						

En cliquant sur le bouton '*Ajouter Client*', une fenêtre s'affiche qui permet au gestionnaire à ajouter un nouveau

client à la base.

Information Generale

Information de Contact

Nom:

Prenom:

Adresse:

Numero telephone:

Login:

Password:

Confirmer

Annuler

En cliquant sur le bouton 'voir son commande', on rediret à une autre interface 'Liste de Commande de Client No.XXX'

Liste de Commande de Client No.001					
N de Client	N de commande	Date Checkin	Date Checkout	Nb de personne	Edit
001	C190321	22/03/2019	31/03/2019	2	<div>Modifier</div> <div>Supprimer</div>
001	C190403	03/04/2019	04/04/2019	1	<div>Modifier</div> <div>Supprimer</div>

Vue Liste de Ressource

L'interface de liste de ressource s'affiche en sélectionnant une commande et cliquant sur le bouton '*voir les ressource*'.

<

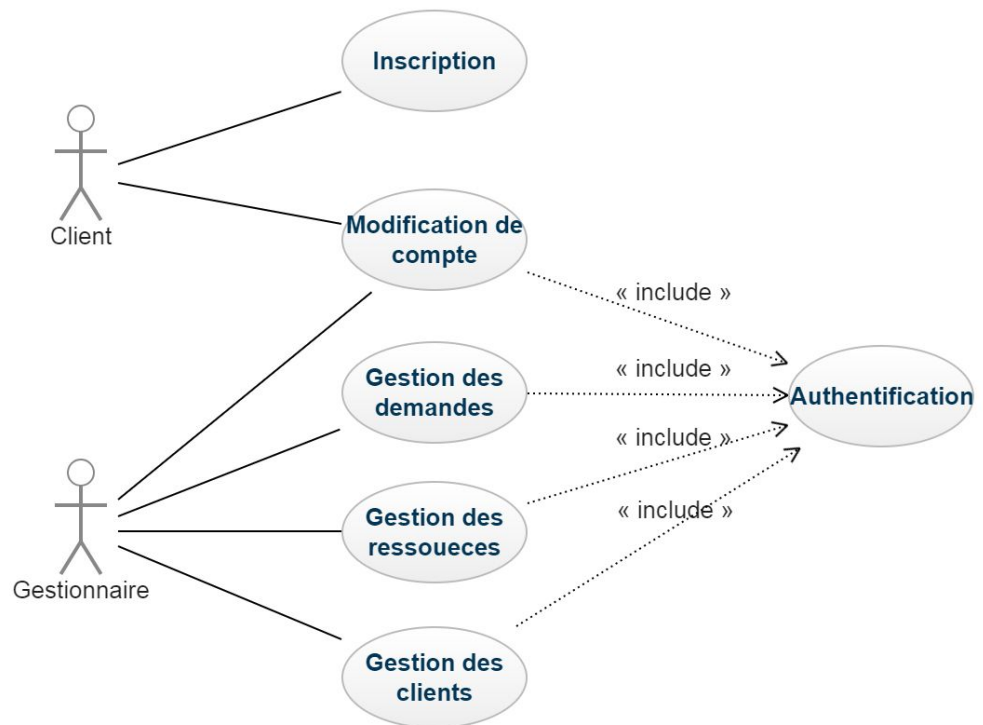
Vue liste de meuble de ressource.

Cette interface permet au gestionner de gerer les Meuble.

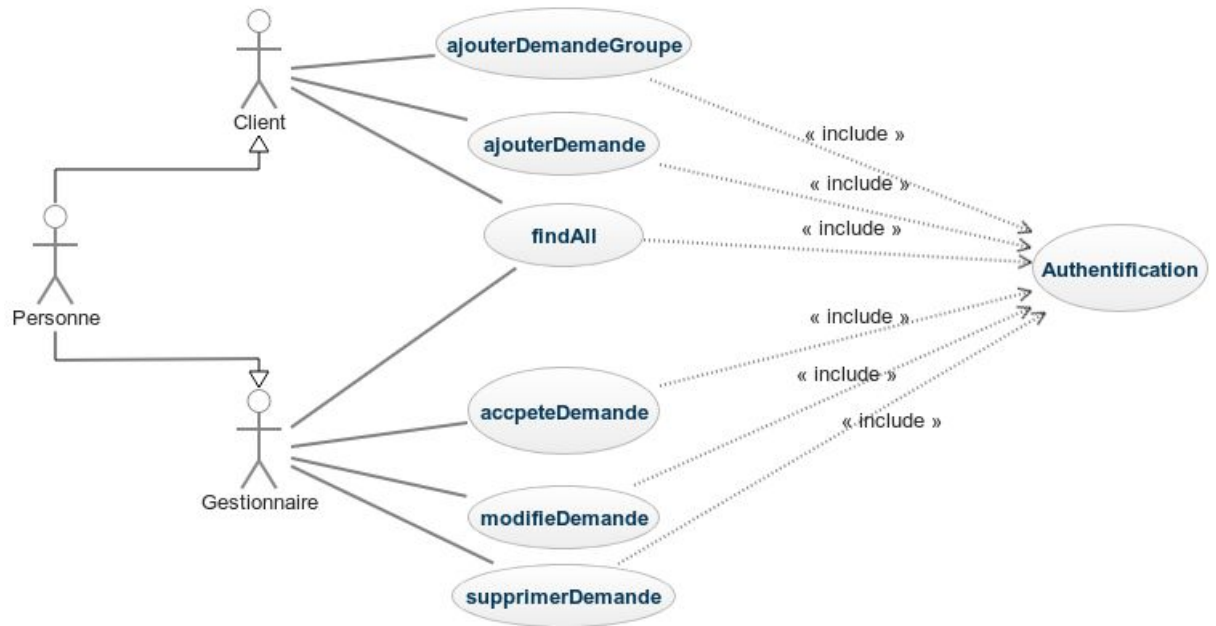
<u>Liste de Meuble de Ressource</u>		
<u>ID: XXXX</u>		
Panneau de Controle	ID Meuble	Nom Meuble
Ajouter	表中无内容	
Modifier		
Supprimer		

3.4 Diagramme des cas utilisations

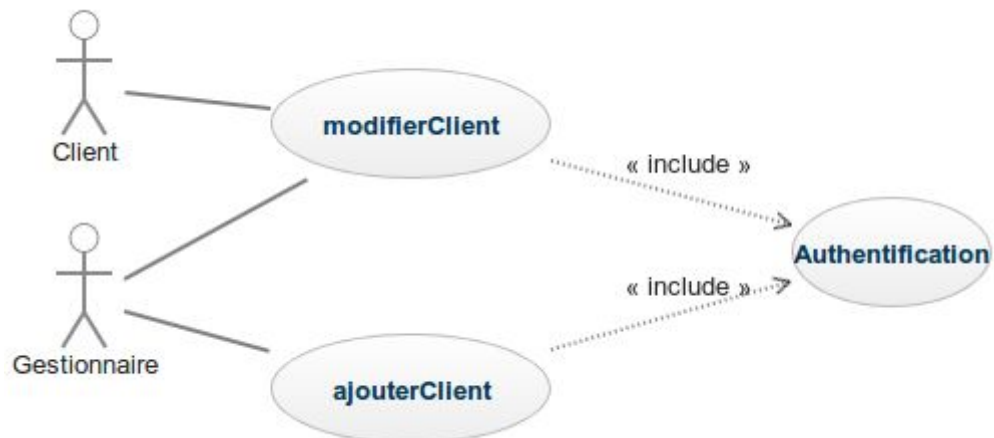
3.4.1 Authentification



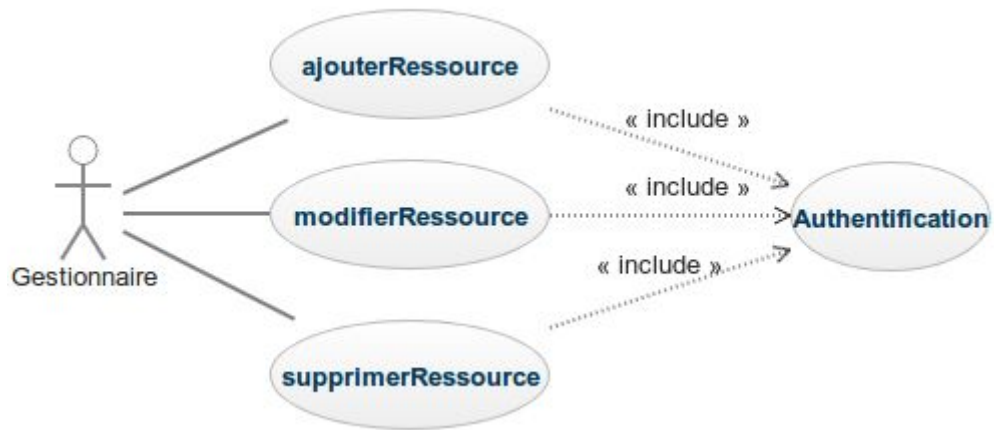
3.4.2 Gestion des demandes



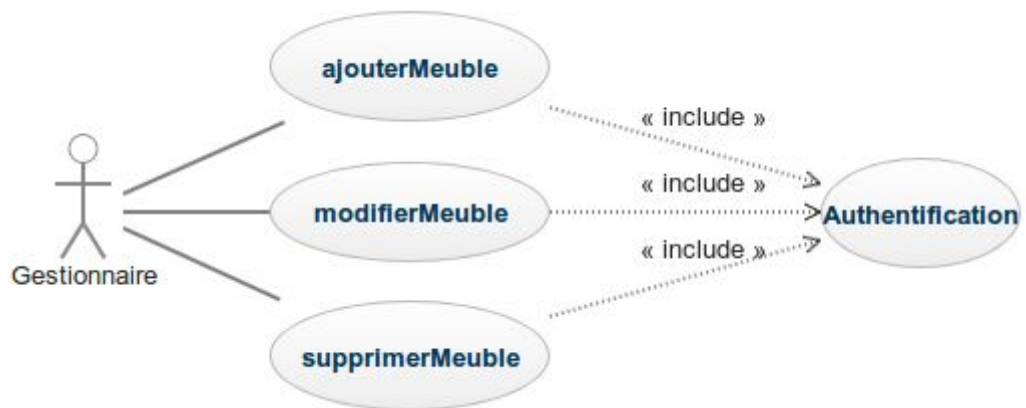
3.4.3 Gestion des clients



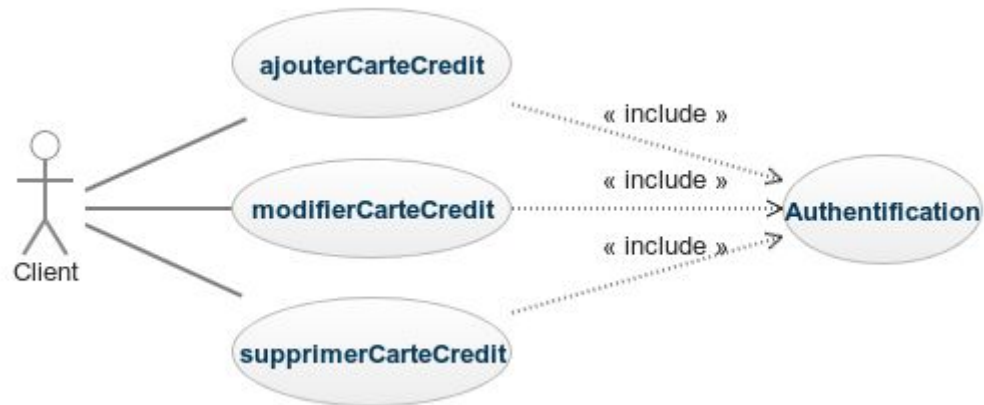
3.4.4 Gestion des ressources



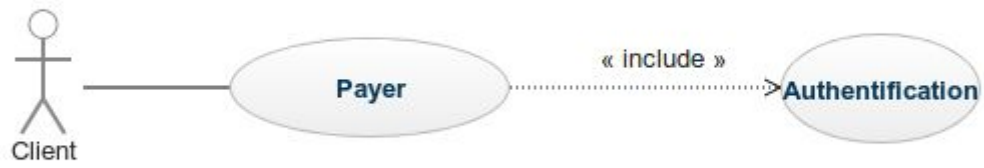
3.4.5 Gestion de l'inventaire



3.4.6 Gestion des carte crédit



3.4.7 Paiement en ligne

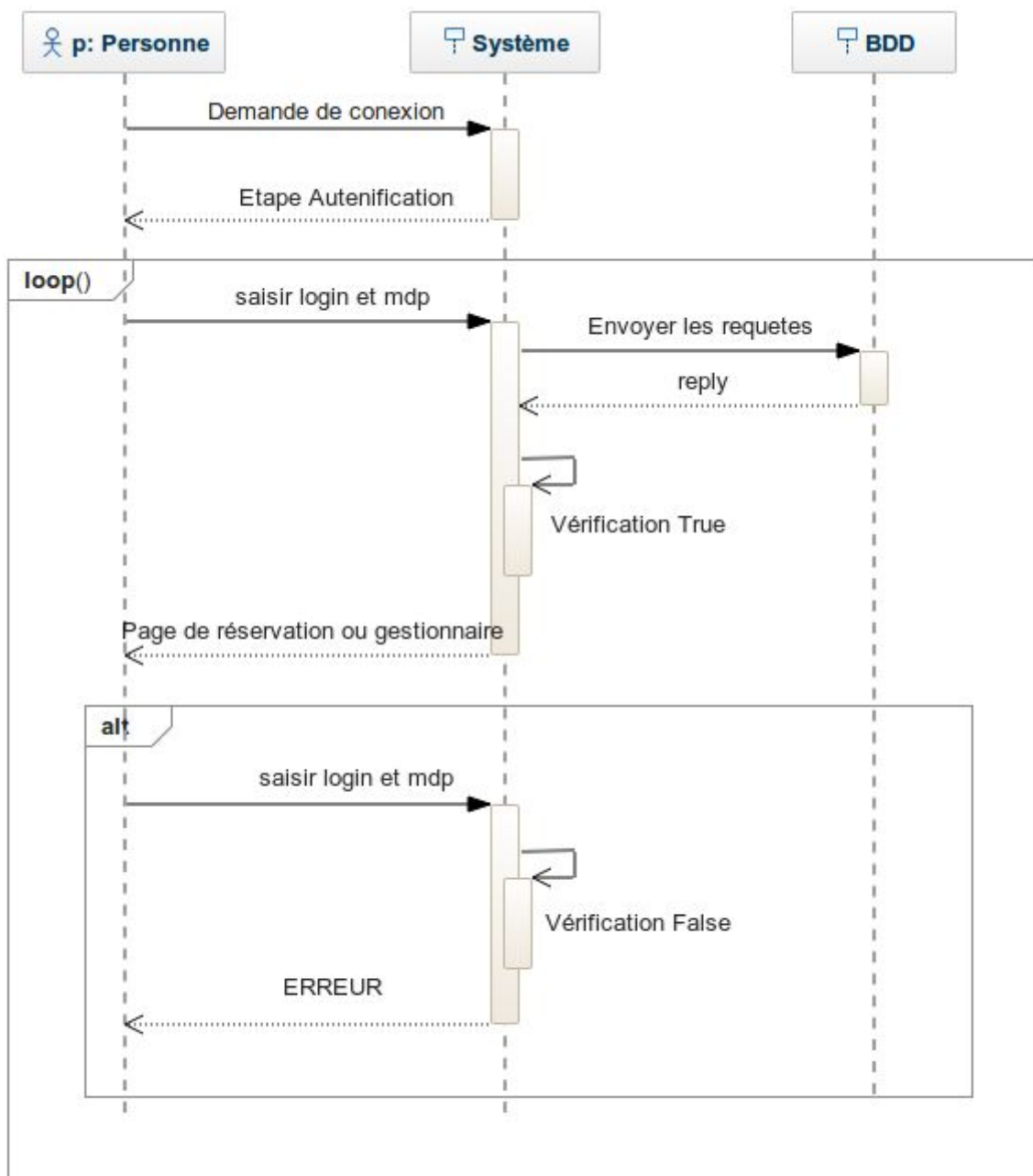


3.4.8 L'émission de facture



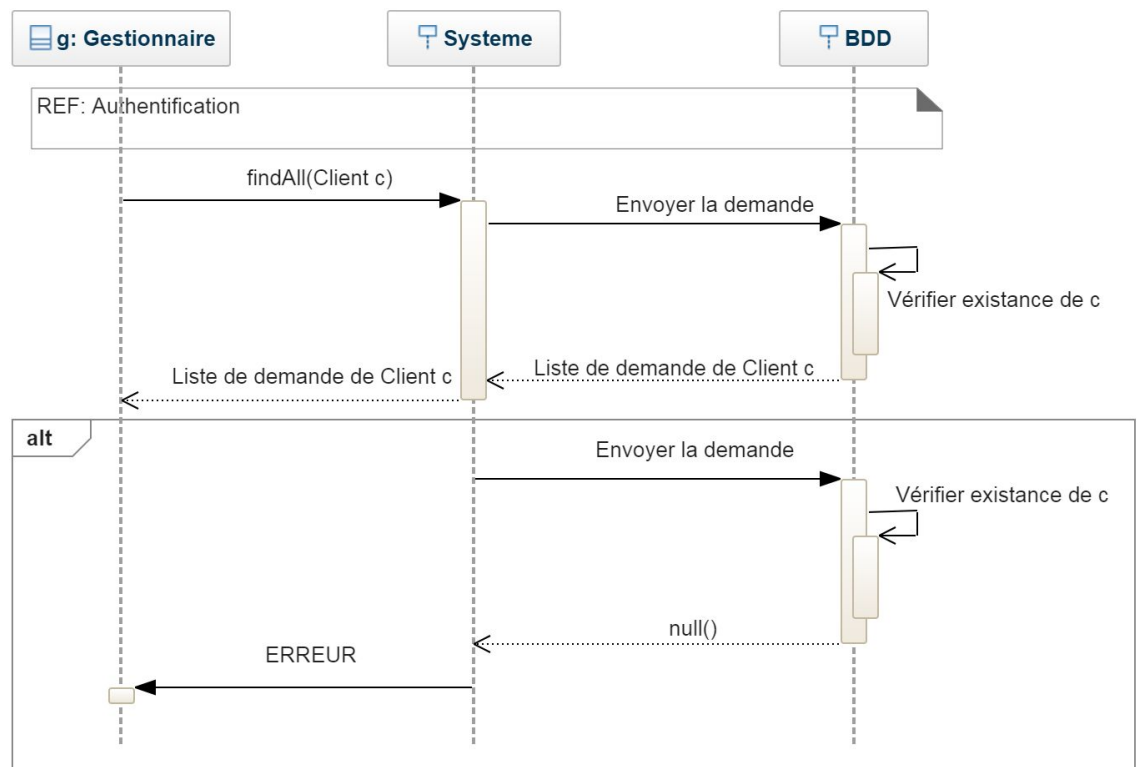
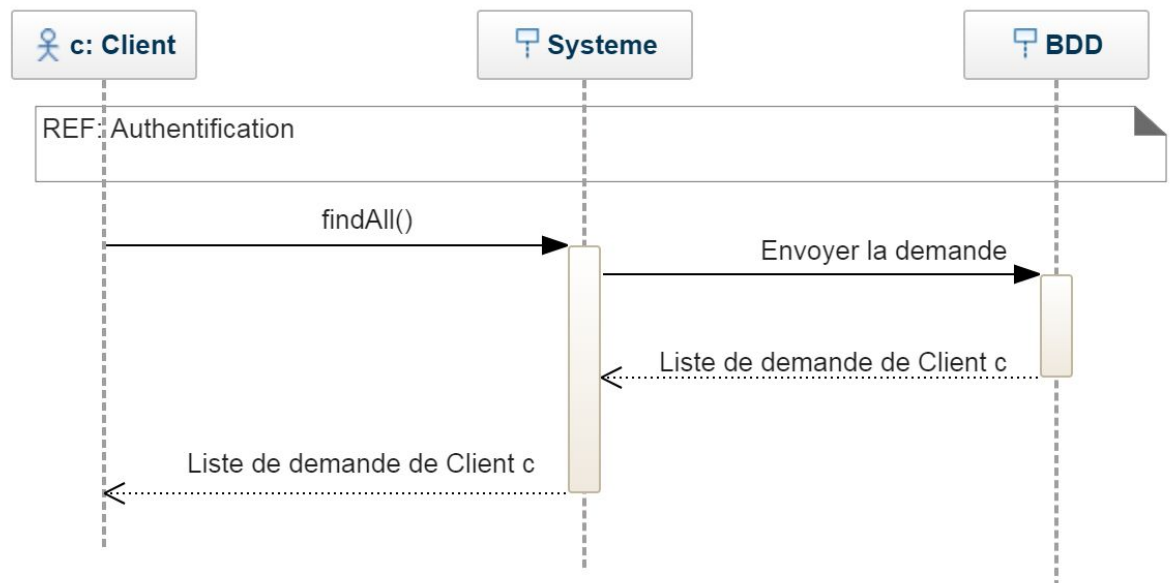
3.5 Les diagramme de séquence

3.5.1 Authentification

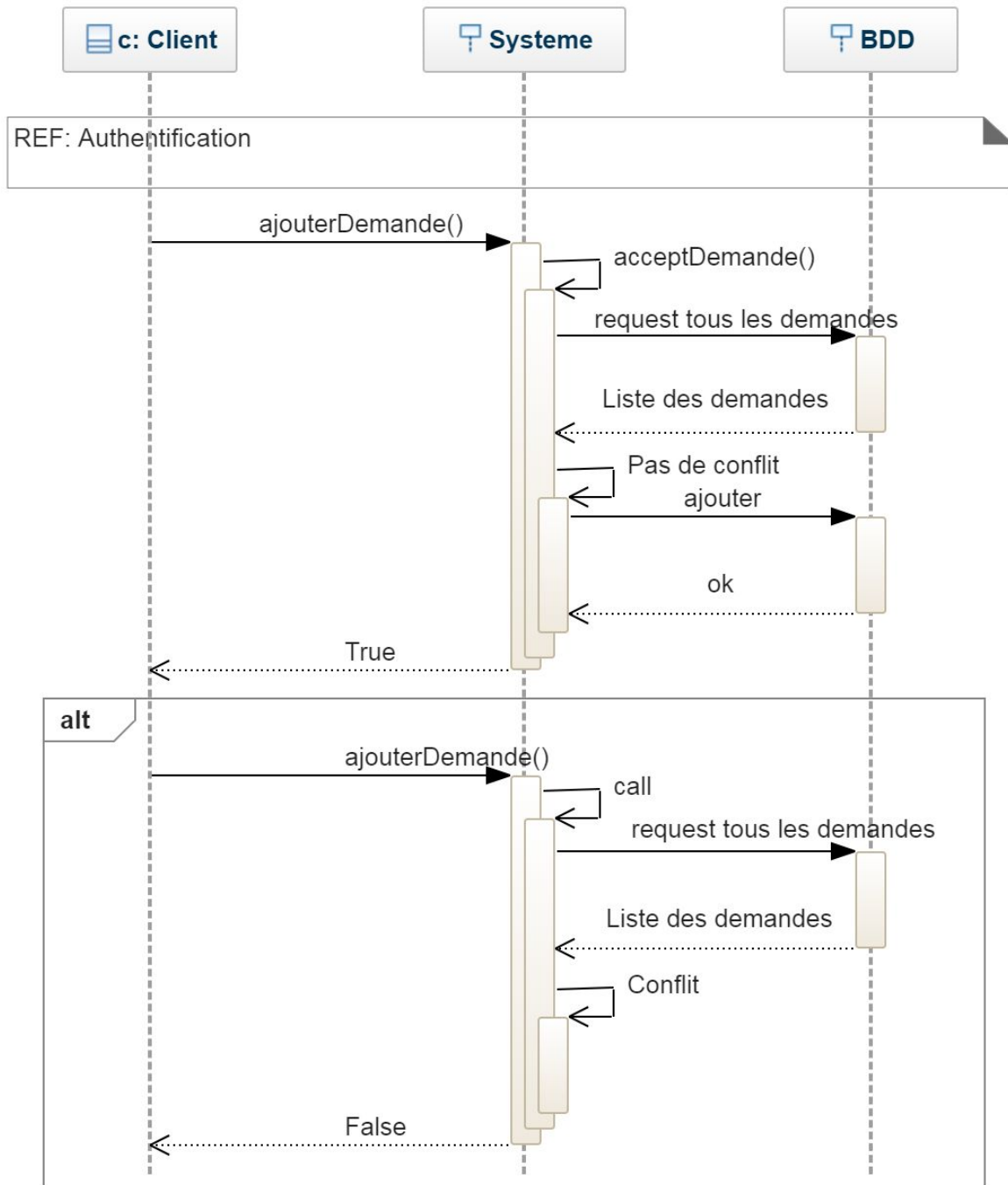


3.5.2 Gestion des réservations

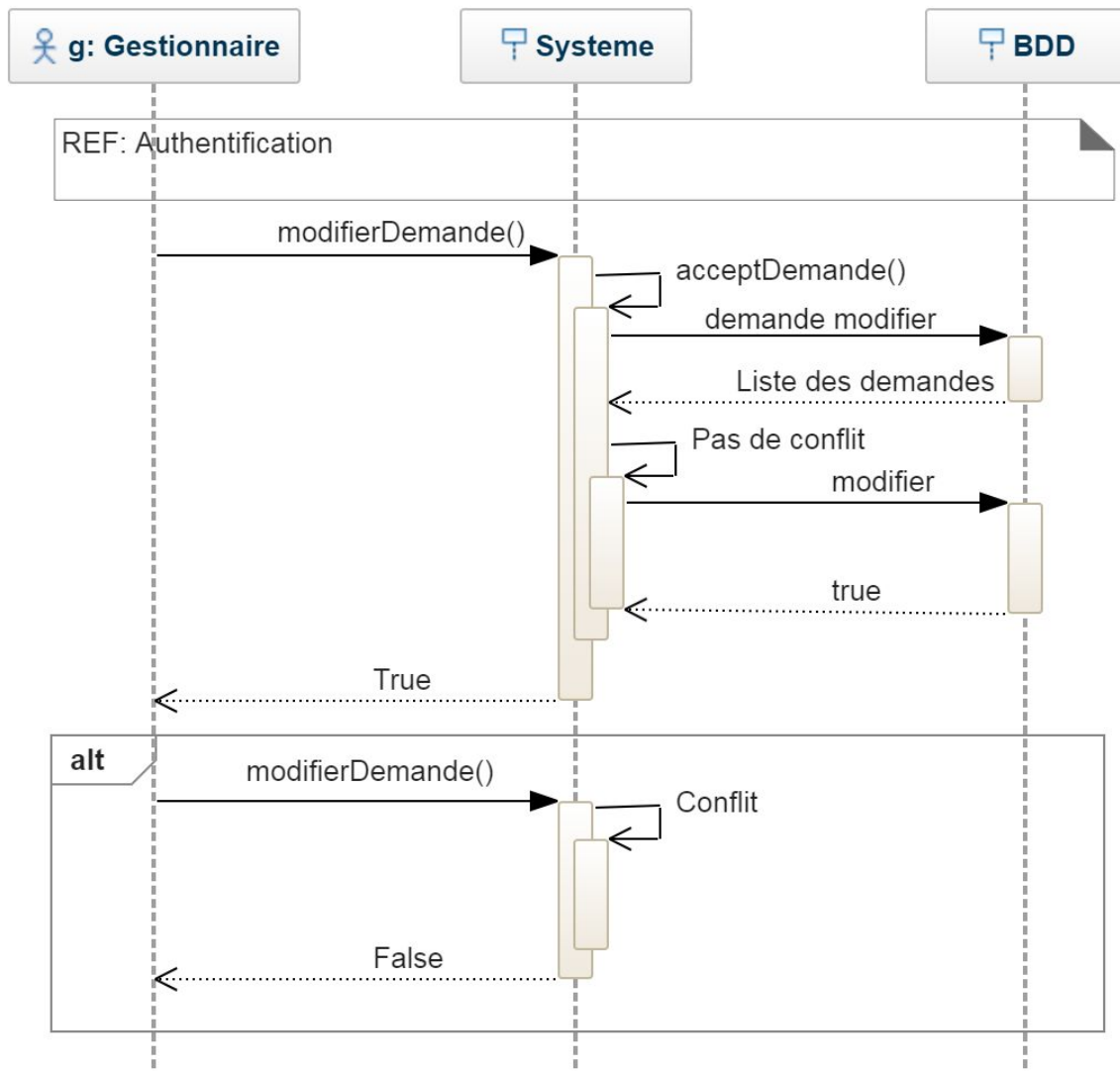
3.5.2.1 findAll(Consultation)



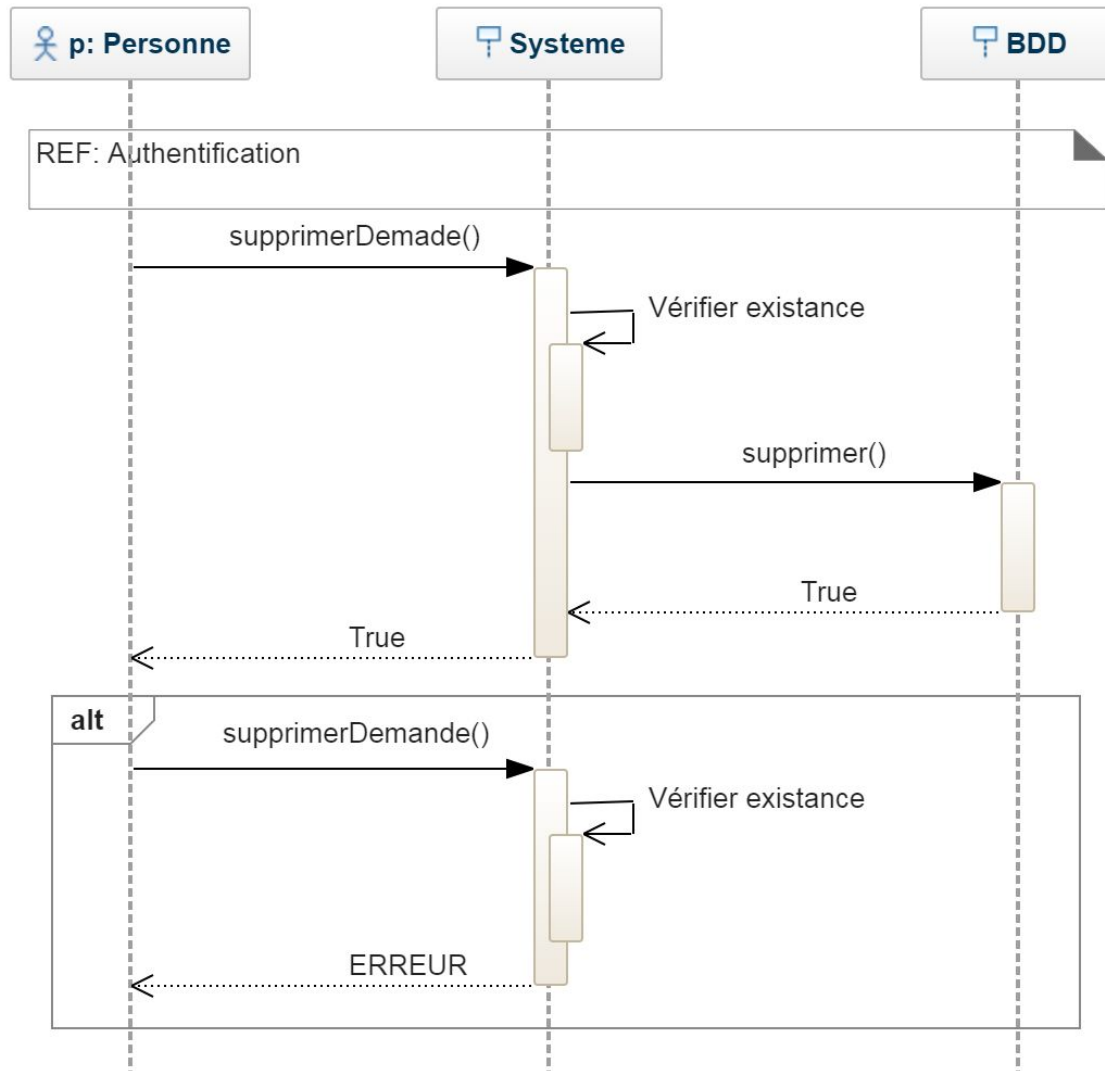
3.5.2.2 ajouterDemande



3.5.2.3 modifierDemande

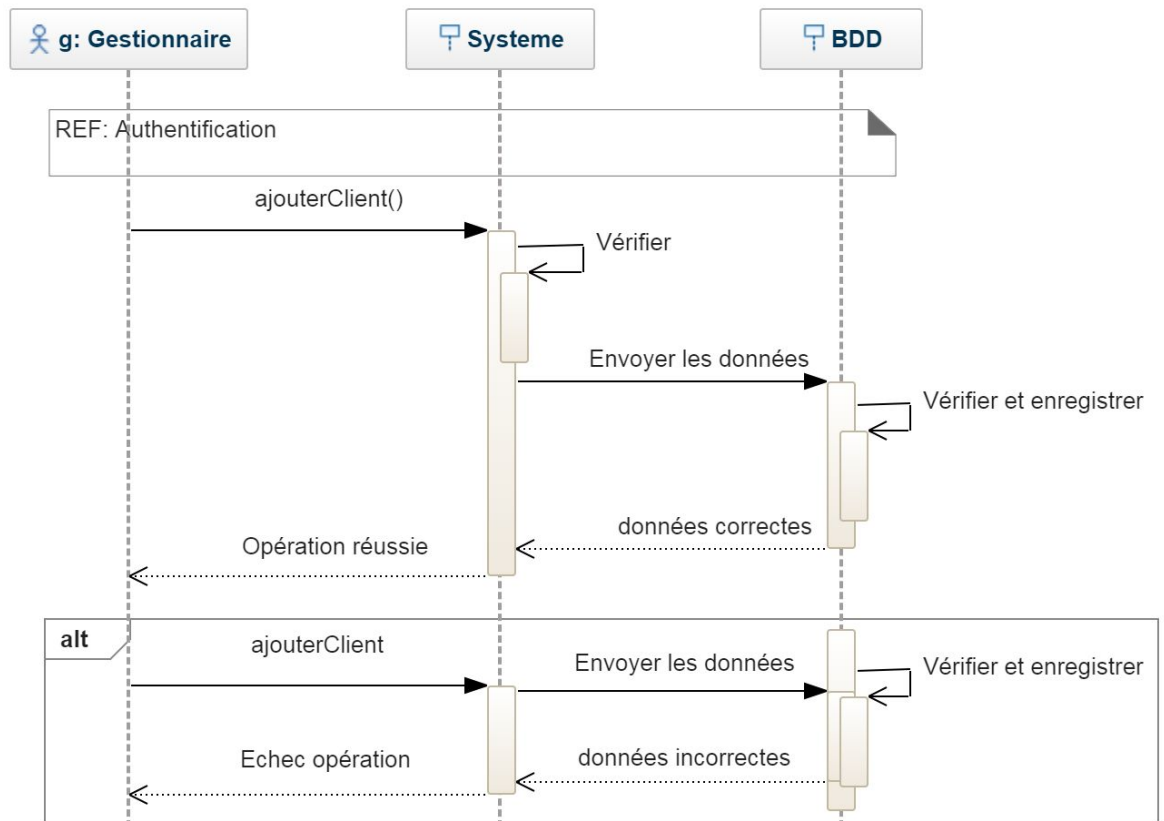


3.5.2.4 supprimerDemande

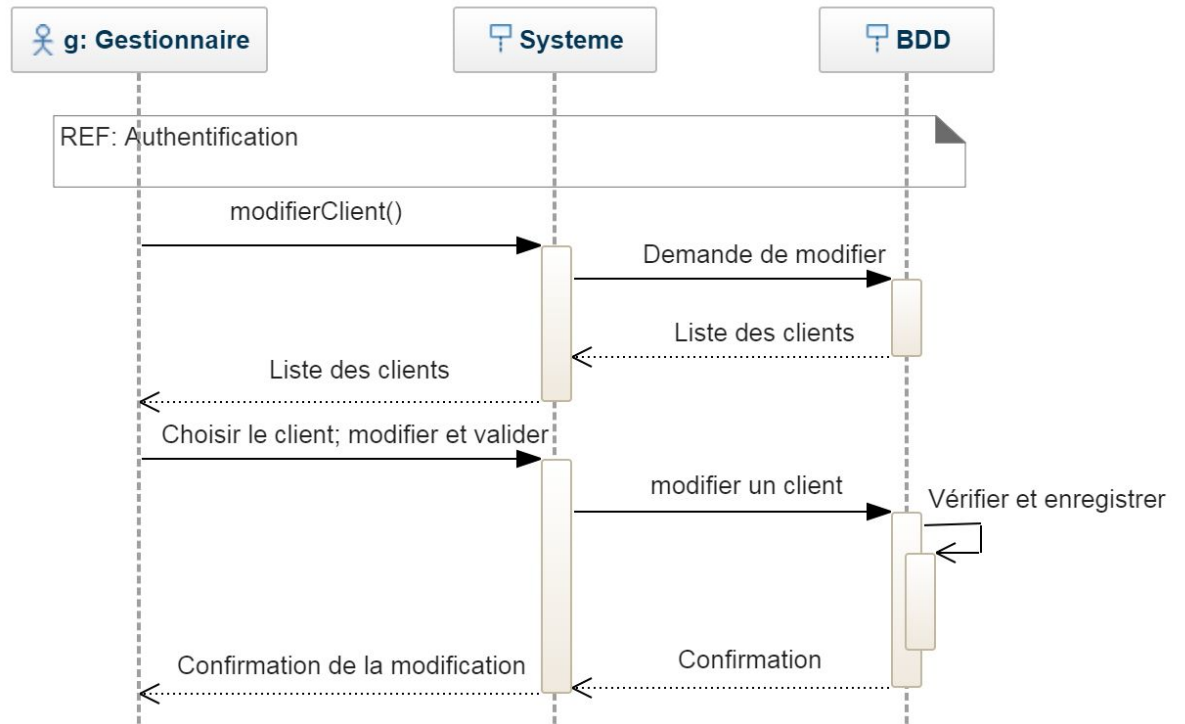


3.5.3 Gestion des ressources

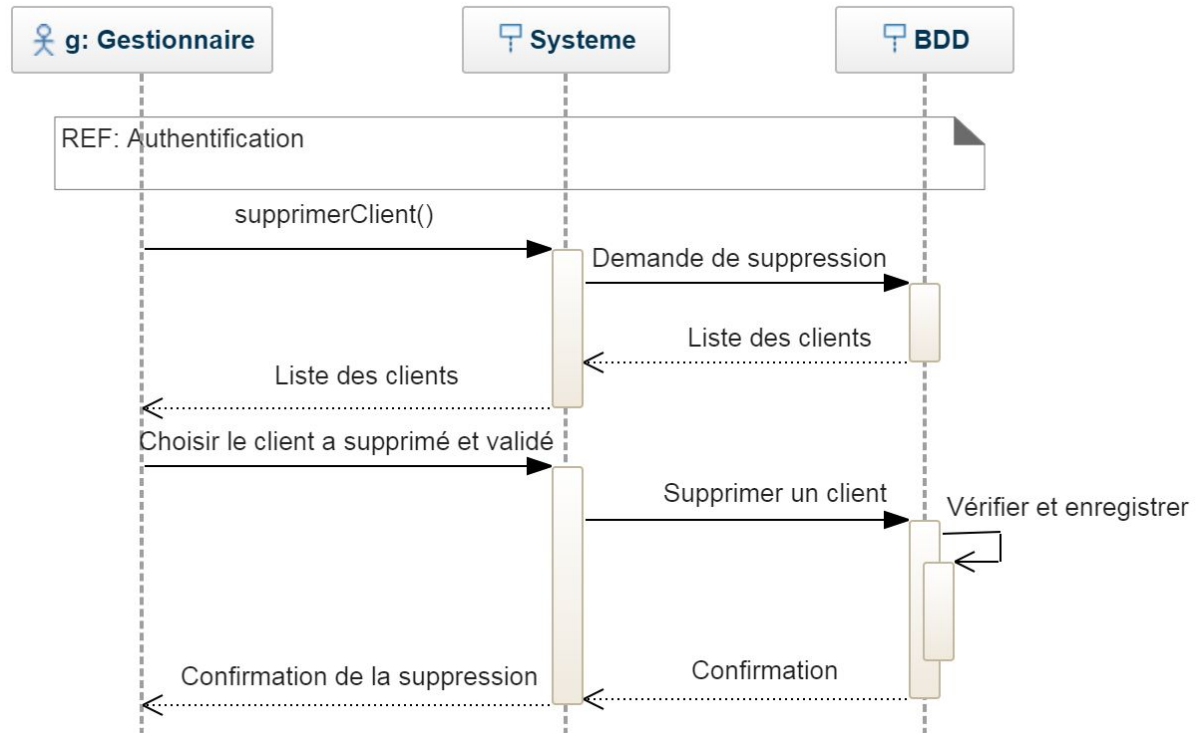
3.5.3.1 ajouterClient



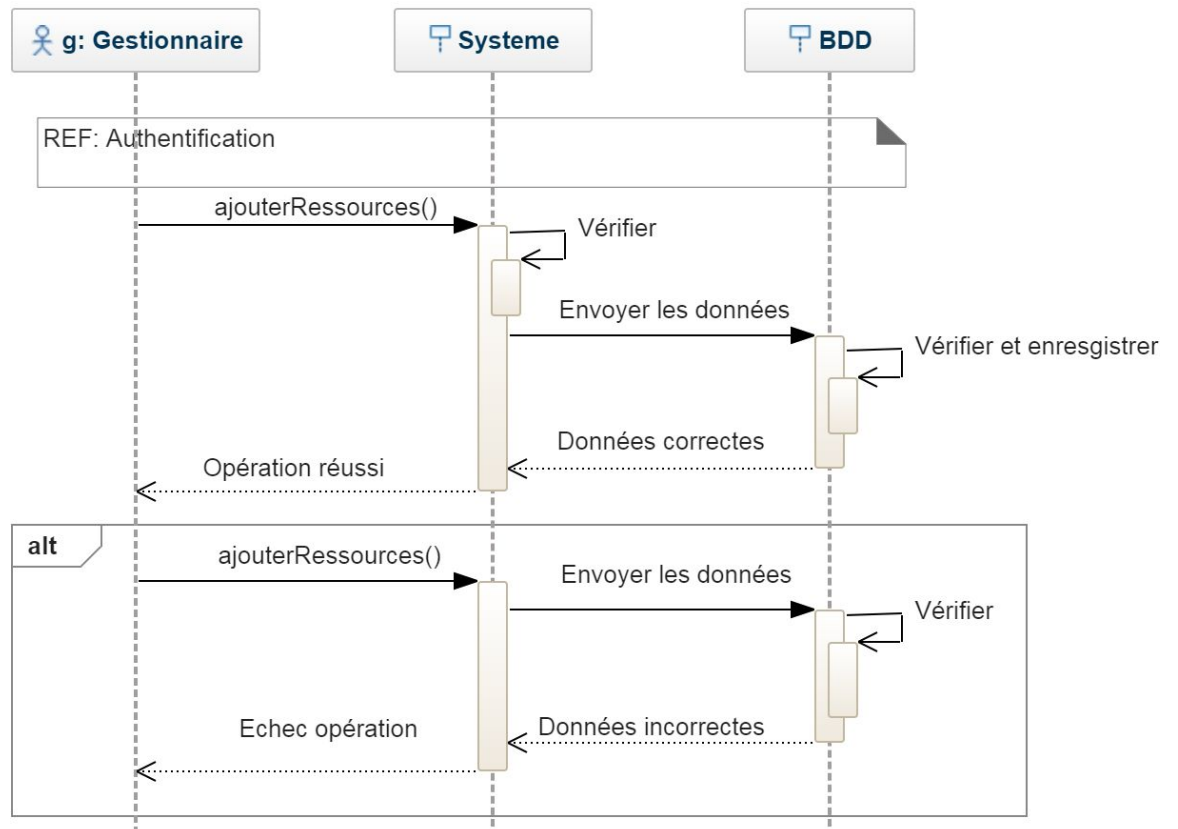
3.5.3.2 modifierClient



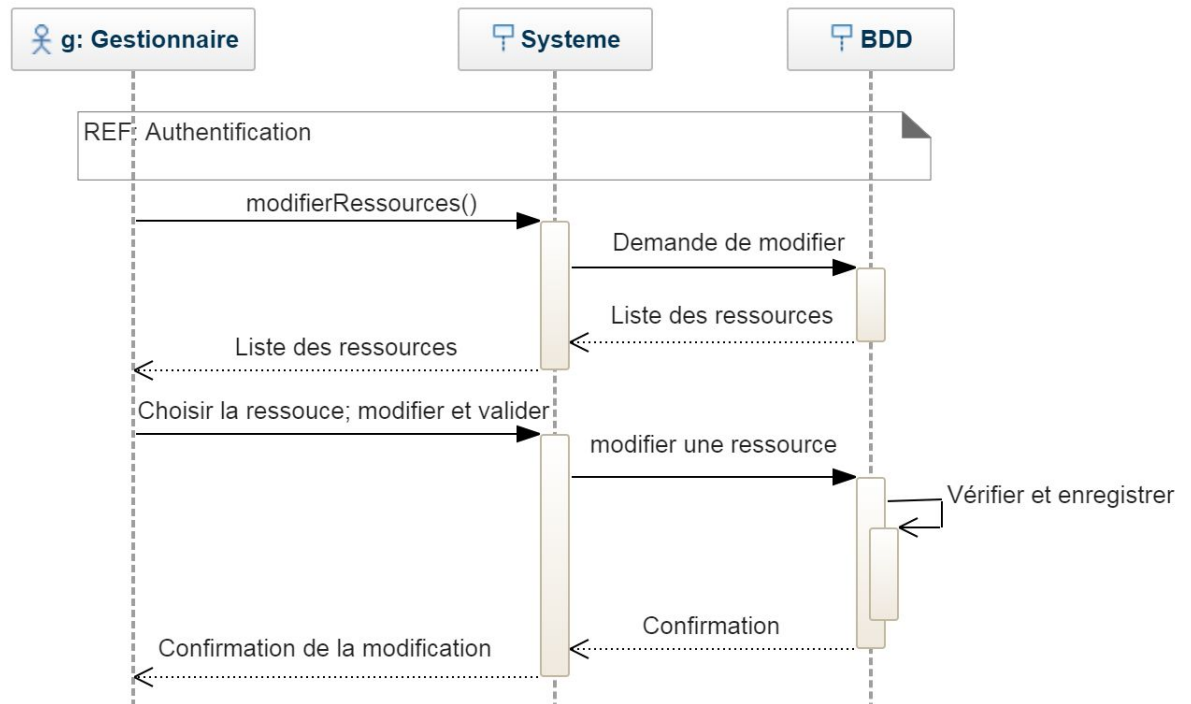
3.5.3.3 supprimerClient



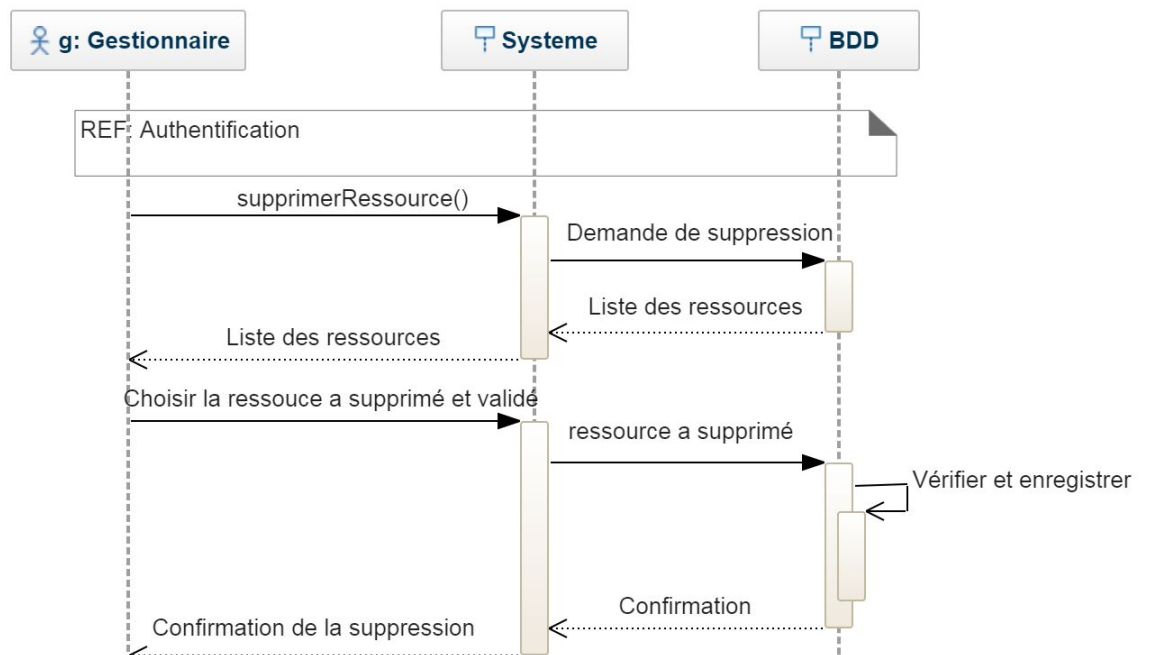
3.5.3.4 ajouterRessources



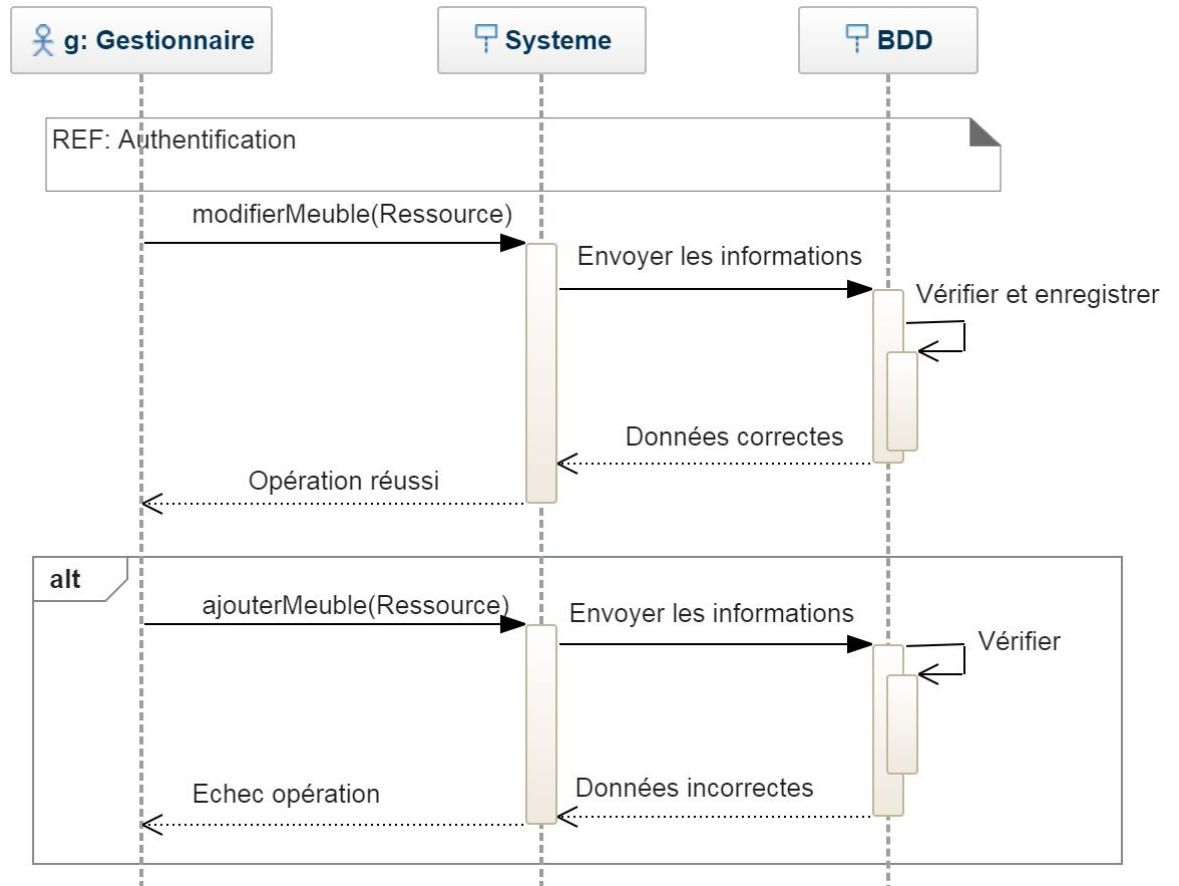
3.5.3.5 modifierRessources



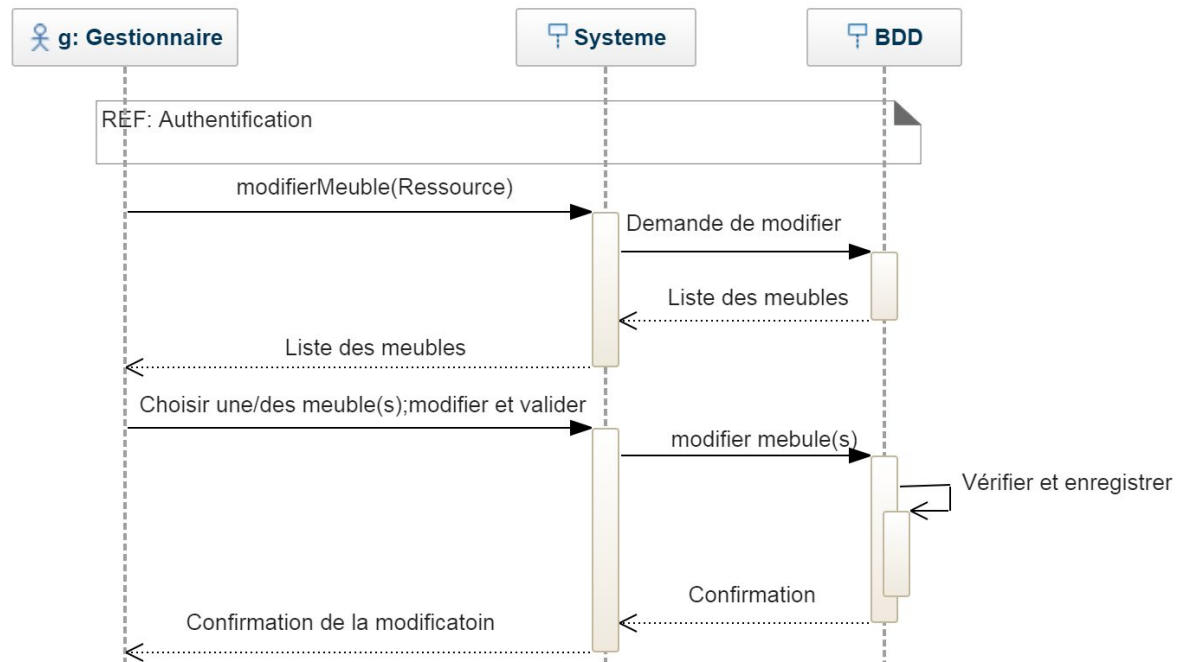
3.5.3.5 supprimerRessources



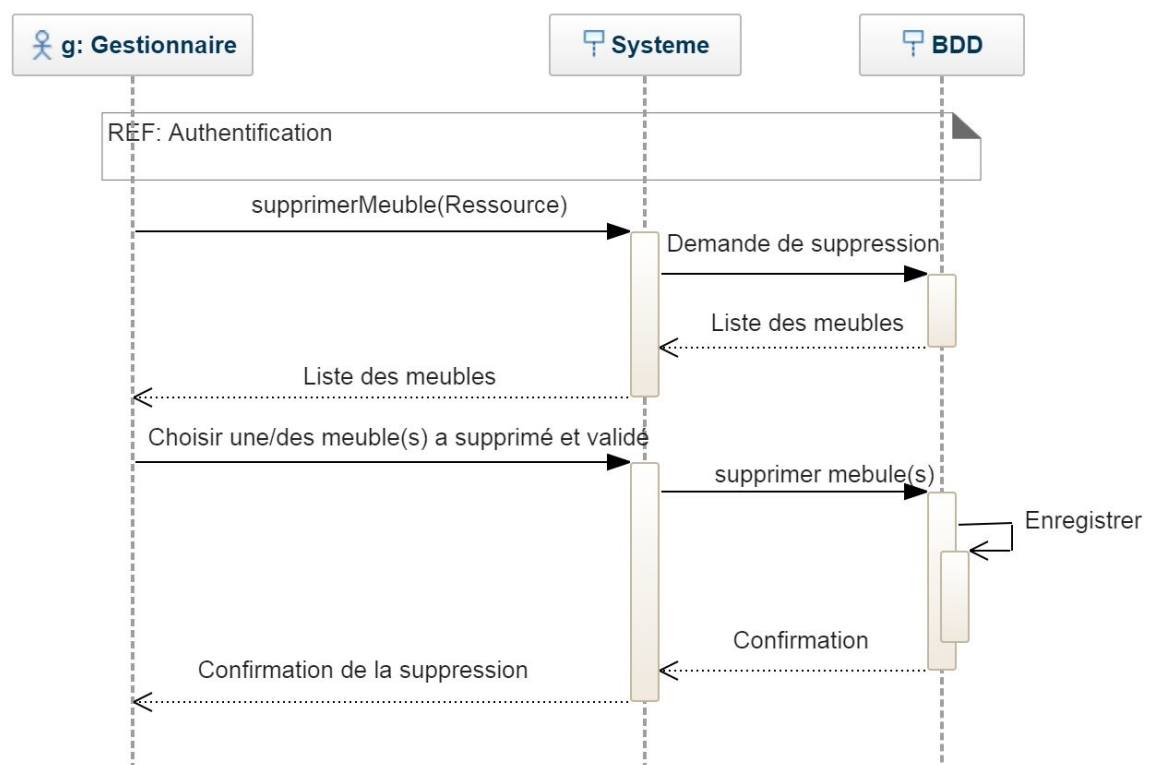
3.5.3.6 ajouterMeuble



3.5.3.7 modifierMeuble



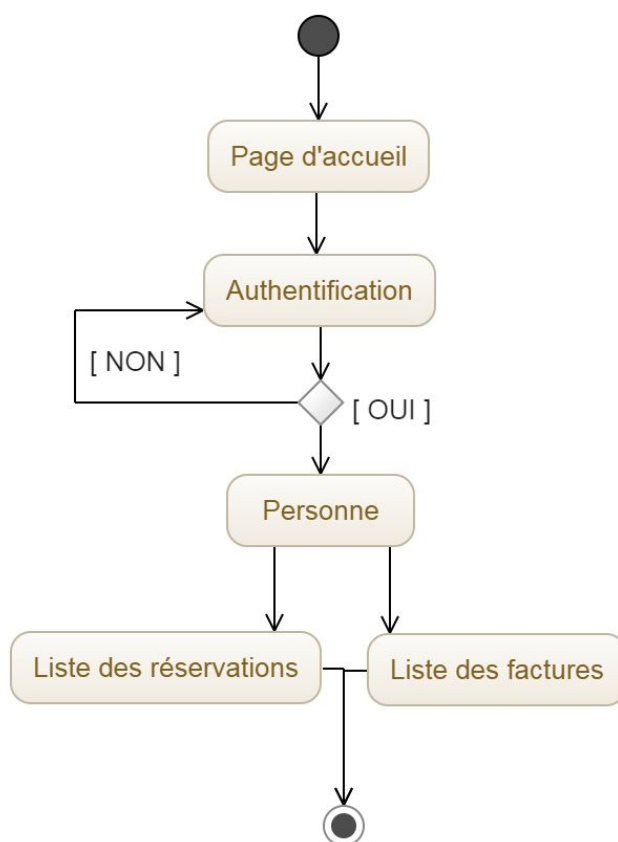
3.5.3.8 supprimerMeuble



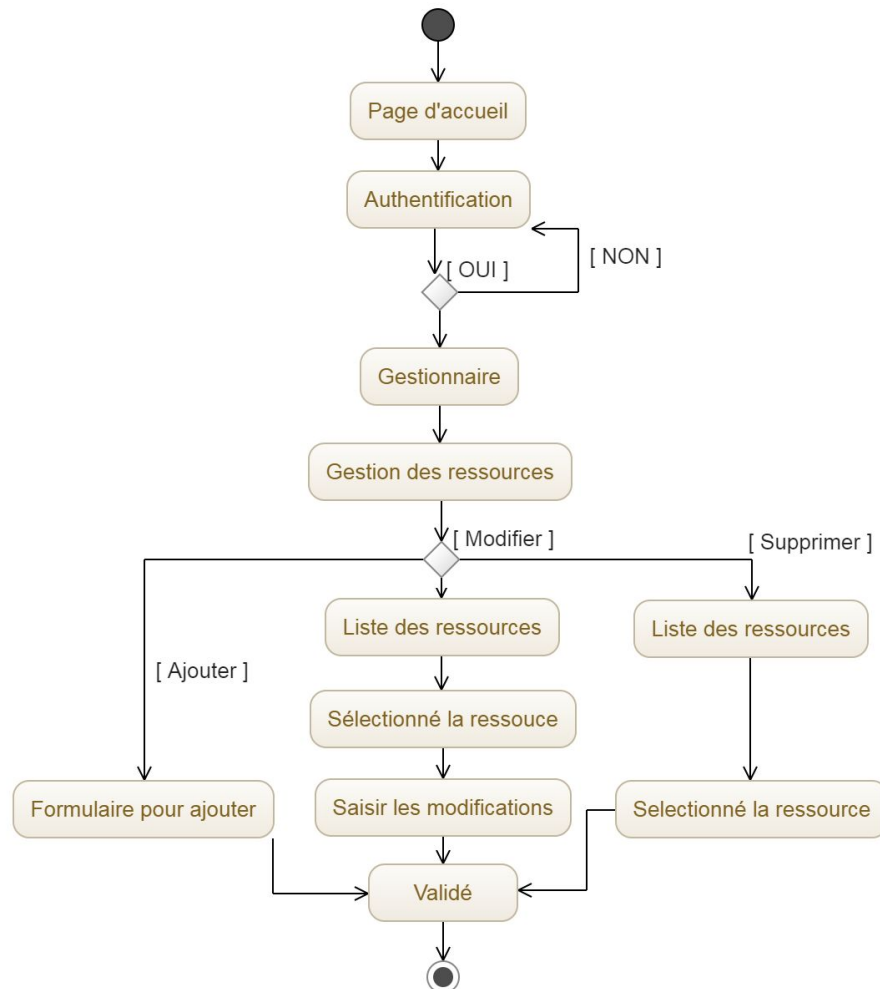
3.6 Diagramme d'activité

Le diagramme d'activité représente la dynamique du système. Il montre l'enchaînement des activités d'un système ou d'une opération. Il représente le flot de contrôle qui retrace le fil d'exécution et qui transite d'une activité à une autre dans le système.

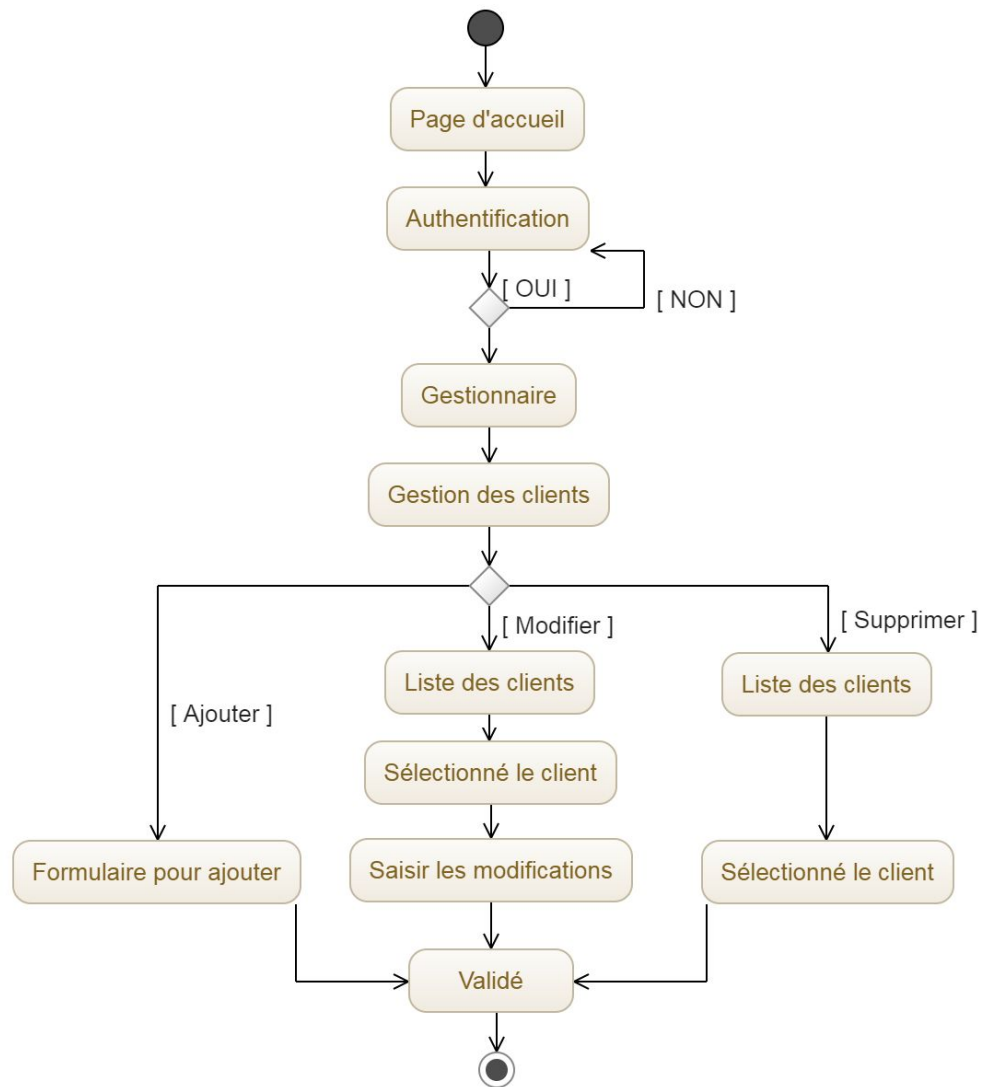
3.6.1 Consultation



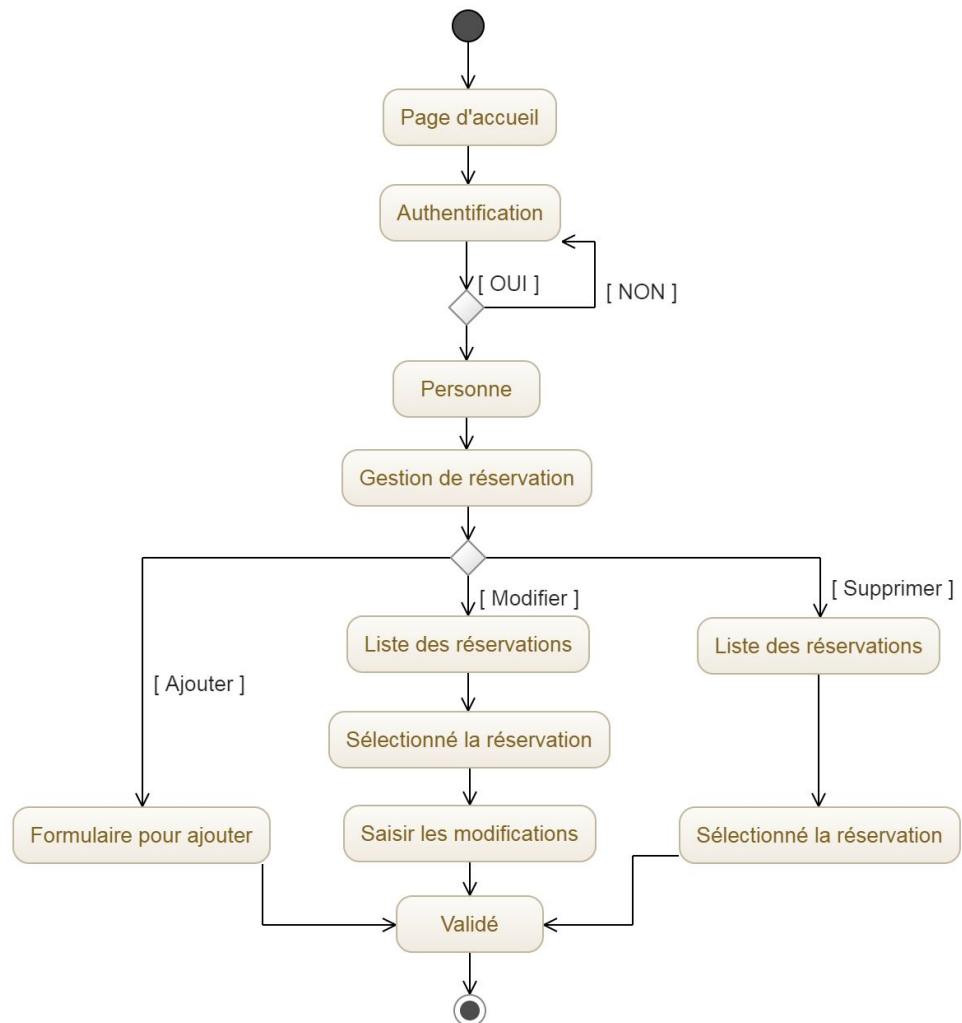
3.6.2 Gestion des ressources



3.6.3 Gestion des clients



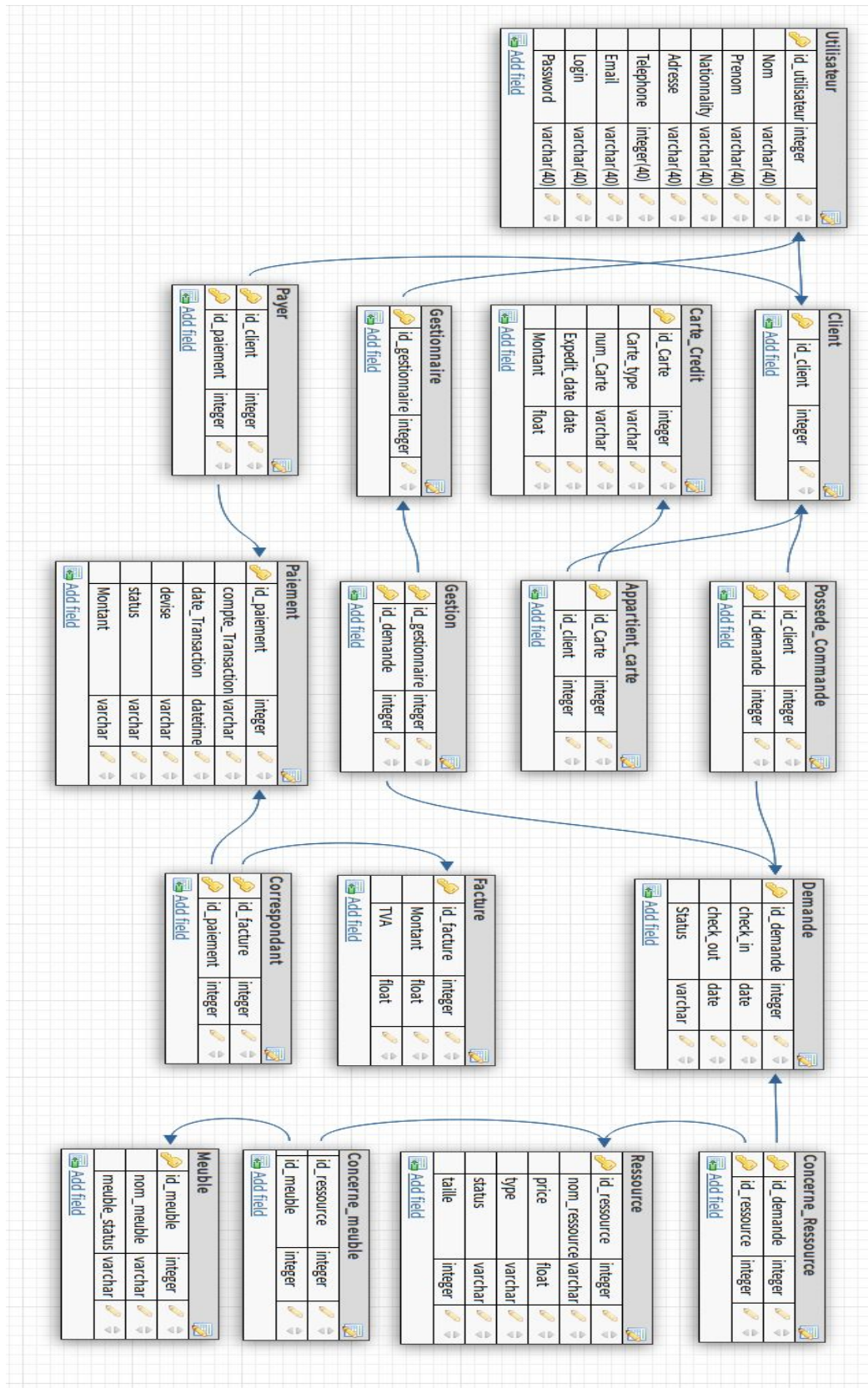
3.6.4 Gestion des réservations



4. Modèle Relationnel

Le modèle relationnel est basé sur une organisation des données sous forme de tables. La manipulation des données se fait selon le concept mathématique de relation de la théorie des ensembles « l'algèbre relationnelle ». Elle est constituée d'un ensemble d'opérations formelles sur les relations. Les opérations relationnelles permettent de créer une nouvelle relation (table) à partir d'opérations élémentaires sur d'autres tables.

4.1 Diagramme de classe



4.2 Règle de passage au modèle logique de donnée relationnel

Règle 1 : Transformation des classes

Chaque classe devient une relation. L'identifiant (respectivement les attributs) de la classe devient la clé primaire (respectivement des attributs) de la relation.

Règle 2 : Association un-à-plusieurs

Il faut ajouter un attribut de type clé étrangère dans la relation fils de l'association. L'attribut porte le nom de la clé primaire de la relation père de l'association.

Règle 3 : Associations plusieurs-à-plusieurs ou classes-associations

L'association (classe-association) devient une relation dont la clé primaire est composée par la concaténation des identifiants des classes connectés à l'association (classe association). Les attributs de l'association (classe-association) doivent être ajoutés à la nouvelle relation. Ces attributs ne sont ni clé primaire, ni clé étrangère.

Règle 4 : Association un-à-un

Il faut ajouter un attribut clé étrangère, dans la relation dérivée de la classe ayant la multiplicité minimale égale à un. L'attribut porte le nom de la clé primaire de la relation dérivée de la classe connectée à l'association.

4.3 Le modèle relationnel des données de l'application

Le modèle relationnel des données de l'application est représenté comme suit:

```

Utilisateur (id_Utilisateur, Nom, Prénom, Nationalité,
Adresse, Téléphone, Emailda, Login, Password)
Gestionnaire(id_Gestionnaire;)
Client(id_Client;)
Demande(id_Demande, checkin, checkout, status)
Carte_Credit(id_Carte, num_Carte, Carte_type, Expedit_date, M
ontant)
Paiement(id_Paiement, compte_Transaction, date_Transaction,
devise, status)
Facture(id_facture, motant, TVA) ;
Ressource(id_Ressource, nom_Ressource, price, type, status, ta
ille)
Meuble(id_Meuble, nom_meuble, meuble_status)

PossedeDeamnde(id_Client#, id_Demande;)
Gestion(id_Gestionnaire#, id_Demande;)
Appartient_Carte(id_Carte#, id_Client;)
Payer(id_Clinet#, id_Paiement;)
Correspondant(id_facture#, id_paiement;)
Concerne_Ressource(id_Demande#, id_Ressource;)
Concerne_Meuble(id_Ressource#, id_Meuble;)
    
```

Pour rendre cette structure à 3FN, il faut créer en total au moins 15 tables.