Sujet de TD/TP 2 : Modélisation UML des cas d'utilisation et des classes

Dans ce TD/TP, vous allez approfondir votre compréhension de la modélisation UML en vous basant sur la plateforme WAVE. Votre mission consiste à :

1. Analyser les cas d'utilisation identifiés précédemment pour chaque acteur et en décrire les interactions avec le système.

2. Créer un diagramme de cas d'utilisation représentant les fonctionnalités principales du système, incluant les acteurs et leurs interactions avec les cas d'utilisation.

3. Élaborer un diagramme de classes qui décrit la structure du système, en définissant les classes, les attributs, les méthodes et les relations entre les classes.

4. Définir les relations d'héritage, d'association et d'agrégation dans le diagramme de classes pour illustrer les interactions entre les différentes entités du système.

5. Discuter des implications de la modélisation UML sur le développement du système d'information de WAVE, en prenant en compte les enjeux, défis, et objectifs de la plateforme.

Réfléchissez également à la manière dont ces diagrammes peuvent servir de base pour la conception technique et le développement futur du système d'information.

-------------------------------------

Sujet de TD/TP 3 : Application de la méthode MERISE à la modélisation des données et des traitements

SOLUTION

**1.Analyser les cas d'utilisation identifiés précédemment pour chaque acteur et en décrire les interactions avec le système**

**CLIENT**

* Le client doit créer un compte.
* Le client doit pouvoir envoyer de l’argent
* Le client doit pouvoir recevoir de l’argent
* Consulter son solde
* Consulter son historique
* Faire les paiements de facture,

**AGENT DE GUICHET**

* L’agent de guichet doit gérer les dépôts client
* L’agent de guichet doit gérer les retraits client
* Valider une transaction, en validant les informations et s'assurant de leur exactitude avant de finaliser l’opération.
* Générer un reçu pour le client après avoir effectué une transaction

**GESTIONNAIRE AGENCE**

Le gestionnaire d’agence doit pouvoir :

* Gérer les dépôts client
* Gérer les retraits client
* Traiter les réclamations clients
* Valider une transaction, en validant les informations et s'assurant de leur exactitude avant de finaliser l’opération.
* Générer un reçu pour le client après avoir effectué une transaction

**RESPONSABLE**

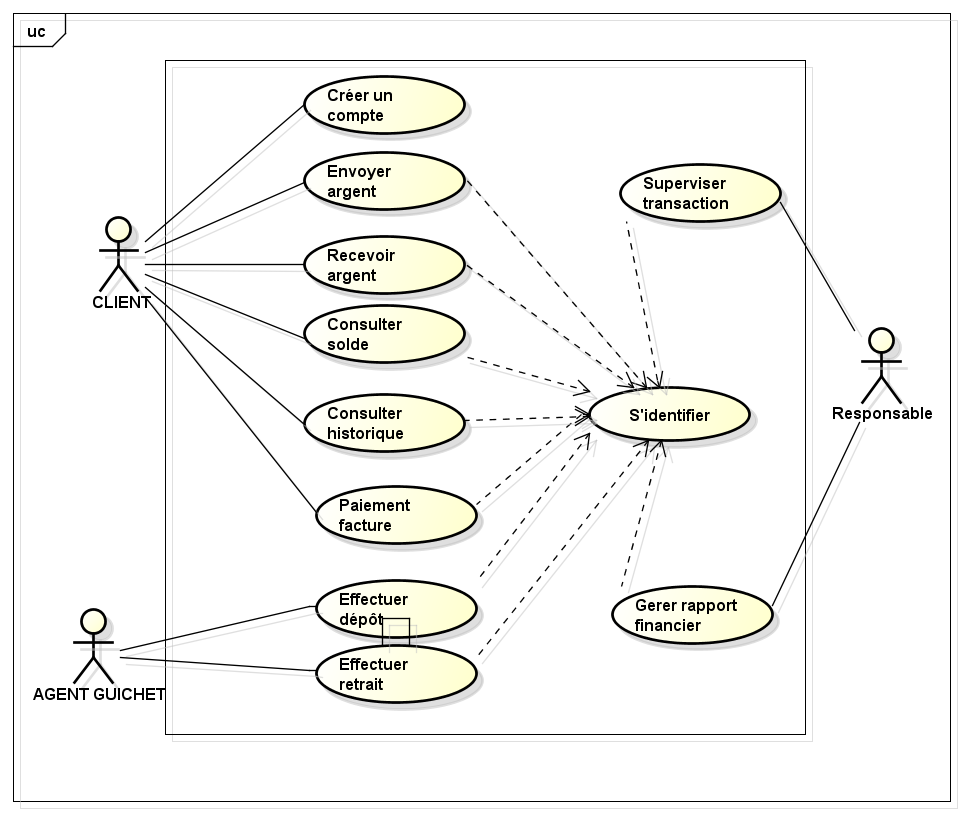
* Le responsable doit pouvoir superviser toutes les transactions
* Le responsable doit pouvoir gérer les rapports financiers
* Le responsable peut examiner des transactions signalées comme suspectes, les analyser, et prendre des mesures correctives si nécessaire.

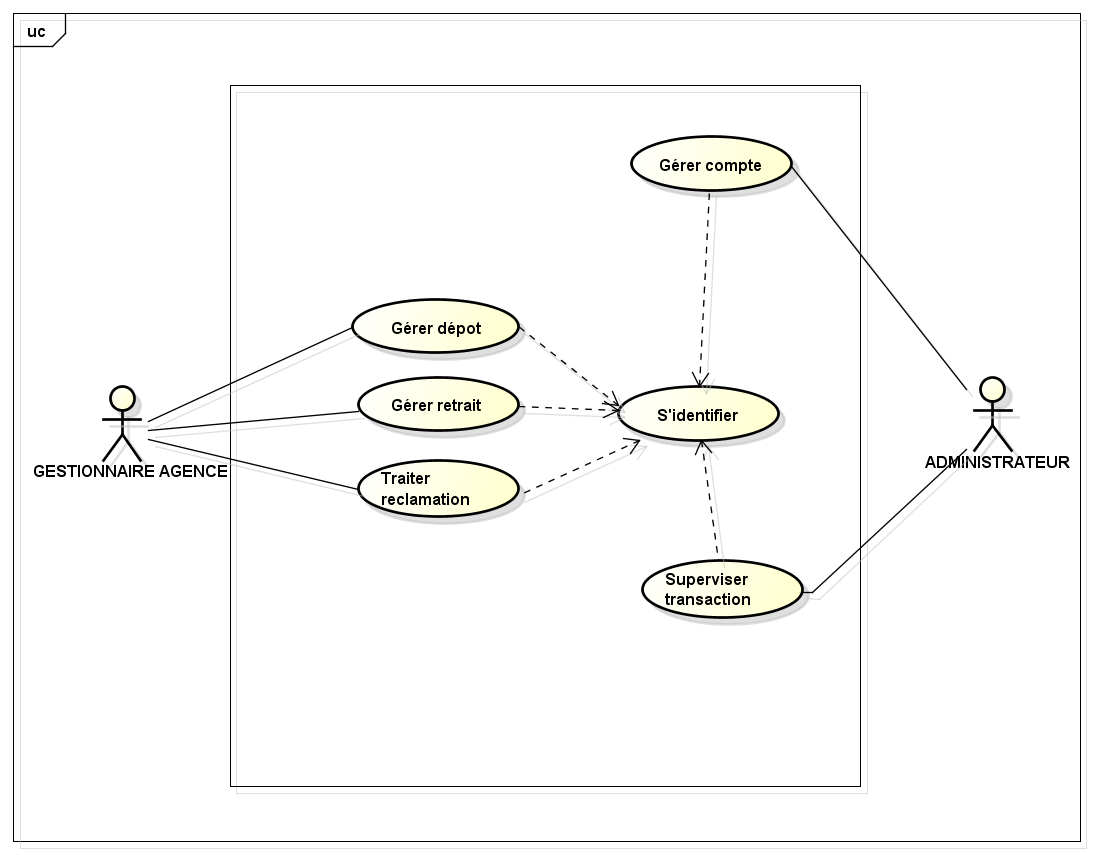
**ADMINISTRATEUR**

L'administrateur du système doit pouvoir :

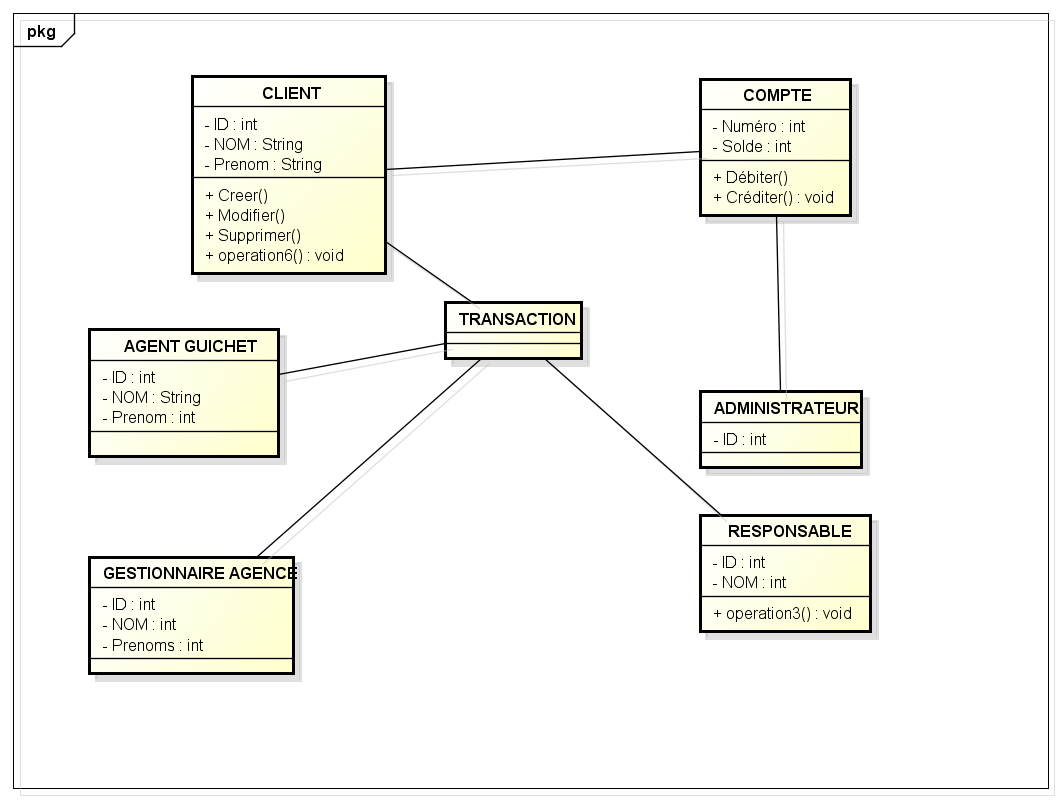
* Effectuer toutes les actions affectées au responsable
* Gérer le compte des clients
* Créer et gérer les comptes des agents de guichet

**2. Créer un diagramme de cas d'utilisation représentant les fonctionnalités principales du système, incluant les acteurs et leurs interactions avec les cas d'utilisation.**





**3. Élaborer un diagramme de classes qui décrit la structure du système, en définissant les classes, les attributs, les méthodes et les relations entre les classes**



**Explication**

**Client**

Attributs :

nom: string

identifiant: string

solde: float

historiqueTransactions: List<Transaction>

numéroTéléphone: string

Méthodes :

envoyerArgent(destinataire: Client, montant: float): bool

consulterSolde(): float

consulterHistorique(): List<Transaction>

**Agent de Guichet**

Attributs :

nom: string

matricule: string

agence: Agence

Méthodes :

effectuerTransaction(client: Client, montant: float): bool

validerTransaction(transaction: Transaction): bool

générerReçu(transaction: Transaction): string

**Responsable**

Attributs :

nom: string

identifiant: string

agence: Agence

Méthodes :

genererRapportMensuel(): Rapport

superviserAgent(agent: AgentDeGuichet): void

analyserTransaction(transaction: Transaction): bool

**Transaction**

Attributs :

identifiant: string

montant: float

date: DateTime

statut: string

client: Client

Méthodes :

valider(): bool

annuler(): bool

**Administrateur**

Attributs :

nom: string

identifiant: string

email: string

motDePasse: string

dateDernièreConnexion: DateTime

Méthodes :

créerCompteUtilisateur(utilisateur: Utilisateur): bool

supprimerCompteUtilisateur(utilisateur: Utilisateur): bool

modifierDroitsUtilisateur(utilisateur: Utilisateur, droits: List<string>): bool

consulterRapports(agence: Agence): List<Rapport>

gérerParamètresSystème(paramètre: string, valeur: any): bool

**Gestionnaire agence**

Attributs :

nom: string

matricule: string

agence: Agence

Méthodes :

effectuerTransaction(client: Client, montant: float): bool

validerTransaction(transaction: Transaction): bool

générerReçu(transaction: Transaction): string

**4. Définir les relations d'héritage, d'association et d'agrégation dans le diagramme de classes pour illustrer les interactions entre les différentes entités du système.**

**Héritage**

L'héritage est une relation, où une classe dérive d'une classe plus générale

**Association**

L'association représente une relation où une classe dépend d'une autre, souvent indiquée par des flèches simples.

**Agrégation**

L'agrégation est une relation, où une classe contient une autre classe, mais les objets peuvent exister indépendamment.

**5. Discuter des implications de la modélisation UML sur le développement du système d'information de WAVE, en prenant en compte les enjeux, défis, et objectifs de la plateforme.**