PolyQuiz

Document d'architecture logicielle

Version 2.1

Historique des révisions

| **Date** | **Version** | **Description** | **Auteur** |
| --- | --- | --- | --- |
| 2024-09-14 | 1.0 | Introduction | Damaris Calestrov |
| 2024-09-15 | 1.1 | Diagramme de déploiement | Lina Belloui |
| 2024-09-19 | 1.2 | Version améliorée du diagramme de déploiement | Lina Belloui |
| 2024-09-25 | 1.3 | Diagramme de séquence (authentification 50%) | Motassembellah Mohamed Bassiouni |
| 2024-09-26 | 1.4 | Diagramme de séquence (clavardage - canaux de discussion) | Motassembellah Mohamed Bassiouni |
| 2024-09-27 | 1.5 | Diagramme de séquence (clavardage - intégration) | Motassembellah Mohamed Bassiouni |
| 2024-09-27 | 1.6 | Reste des diagrammes de séquence | Motassembellah Mohamed Bassiouni |
| 2024-09-27 | 1.7 | Vue des cas d’utilisation | Aymen Ghodbane |
| 2024-09-27 | 1.8 | Objectifs et contraintes architecturaux  Taille et performance | Damaris Calestrov |
| 2024-09-27 | 1.9 | Vue logique | Lina Belloui |
| 2024-09-27 | 2.0 | Vérifications et corrections | Damaris Calestrov  Manel Abroudj  Lina Belloui  Simon Cloutier  Aymen Ghodbane  Motassembellah Mohamed Bassiouni |
| 2024-11-24 | 2.1 | Finaliser la correction des cas d’utilisations | Motassembellah |

Table des matières

[**1. Introduction 4**](#_heading=h.gjdgxs)

[**2. Objectifs et contraintes architecturaux 4**](#_heading=h.30j0zll)

[**3. Vue des cas d’utilisation 6**](#_heading=h.3znysh7)

[**4. Vue logique 20**](#_heading=h.2et92p0)

[**5. Vue des processus 29**](#_heading=h.tyjcwt)

[**6. Vue de déploiement 40**](#_heading=h.3dy6vkm)

[**7. Taille et performance 41**](#_heading=h.1t3h5sf)

Document d'architecture logicielle

# 1. Introduction

L’architecture logicielle utilisée pour la solution proposée et détaillée dans le SRS sera illustrée dans le document ci présent. Pour commencer, les objectifs et les contraintes architecturaux seront présentés. Par la suite, plusieurs vues seront illustrées afin de présenter les différents éléments de l'architecture. La vue des cas d’utilisation sera la première vue présentée qui contiendra les aspects pertinents du modèle de cas d’utilisation. Ensuite, la vue logique et les aspects de l’architecture logique seront détaillés. Pour continuer, la vue des processus décrivant les interactions entre les différents processus sera présentée. Finalement, la vue de déploiement et les configurations de matériel physique seront décrites. Pour terminer, il sera question des différentes caractéristiques de la taille et de la performance qui impactent l’architecture logicielle.

# 2. Objectifs et contraintes architecturaux

**2.1 Performance :** La plateforme devra supporter plusieurs parties en même temps et sur plusieurs plateformes. Ainsi, il faudrait avoir une approche client-serveur qui supporte les connexions multiples.

**2.2 Confidentialité :** Auth0 est utilisé pour l’authentification et il chiffre les données, ce qui va protéger les données des utilisateurs. Ceci va simplifier l’architecture de l’application, car cette dernière ne s’occupera pas de la gestion de la sécurité et des comptes.

**2.3 Portabilité :** L’architecture logicielle devra être utilisée à la fois pour le client léger que le client lourd. Ceci nécessite d'adopter une approche centralisée afin de permettre la réutilisation des composants. Il faudra donc suivre un modèle MVP (Modèle-Vue-Présentateur) afin de séparer les responsabilités et ainsi avoir une architecture modulaire.

**2.4 Échéancier :** La réponse à l’appel d’offre sera le 27 septembre et la remise finale se fera le 19 novembre. Ceci étant dit, l'échéancier contraint de faire certains choix architecturaux pour permettre de terminer les fonctionnalités à temps. Il faudra donc réutiliser des composants existants, comme des librairies et des frameworks. De plus, le développement devra se faire de façon incrémentale afin de pouvoir livrer des fonctionnalités aux dates de remise.

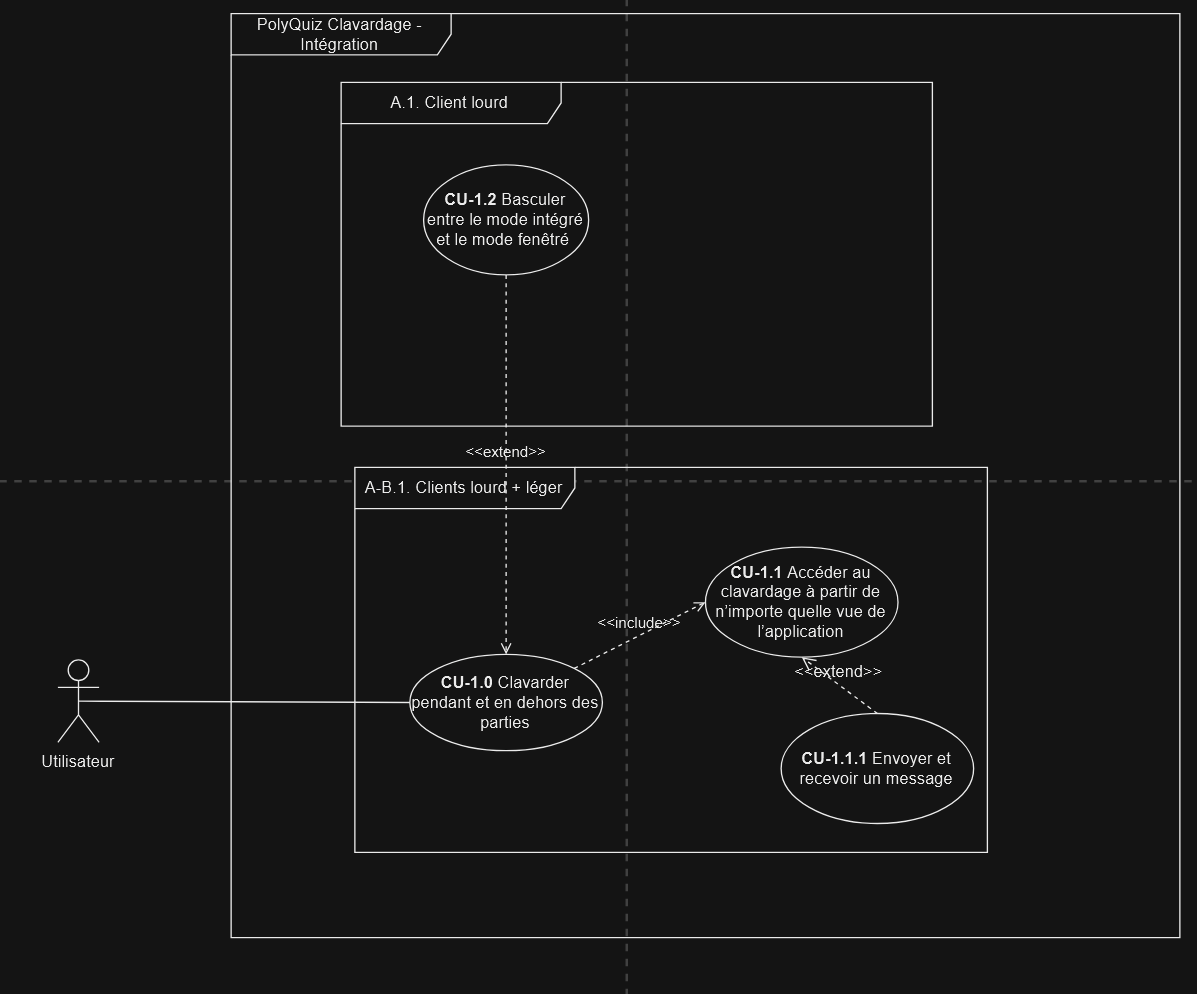
**2.5 Outils de développement :** Electron sera l’environnement d’exécution. Un modèle multi-processus sera utilisé. Le processus principal sera responsable de gérer l’application, le processus de l'authentification va gérer cette dernière et un dernier processus sera utilisé pour le mode fenêtré. Une communication inter-processus sera donc nécessaire afin de communiquer les données. Node.js sera utilisé pour le serveur et ceci permettra un un modèle d’exécution asynchrone et ainsi permettre le traitement simultané de beaucoup de requêtes. Il permet aussi la communication en temps réel à l’aide des sockets. MongoDB Atlas sera utilisé pour stocker les données et la gestion des données devra minimiser les coûts en évitant un usage inapproprié ou redondant.

**2.6 Langage de développement :** Le client lourd sera développé en TypeScript et à l’aide du cadriciel Angular. Ce dernier permet de séparer les responsabilités en utilisant les services et les modules. Le langage pour le client léger sera Kotlin qui utilisera une API pour établir la connexion avec le serveur et ainsi être synchronisé avec le client lourd.

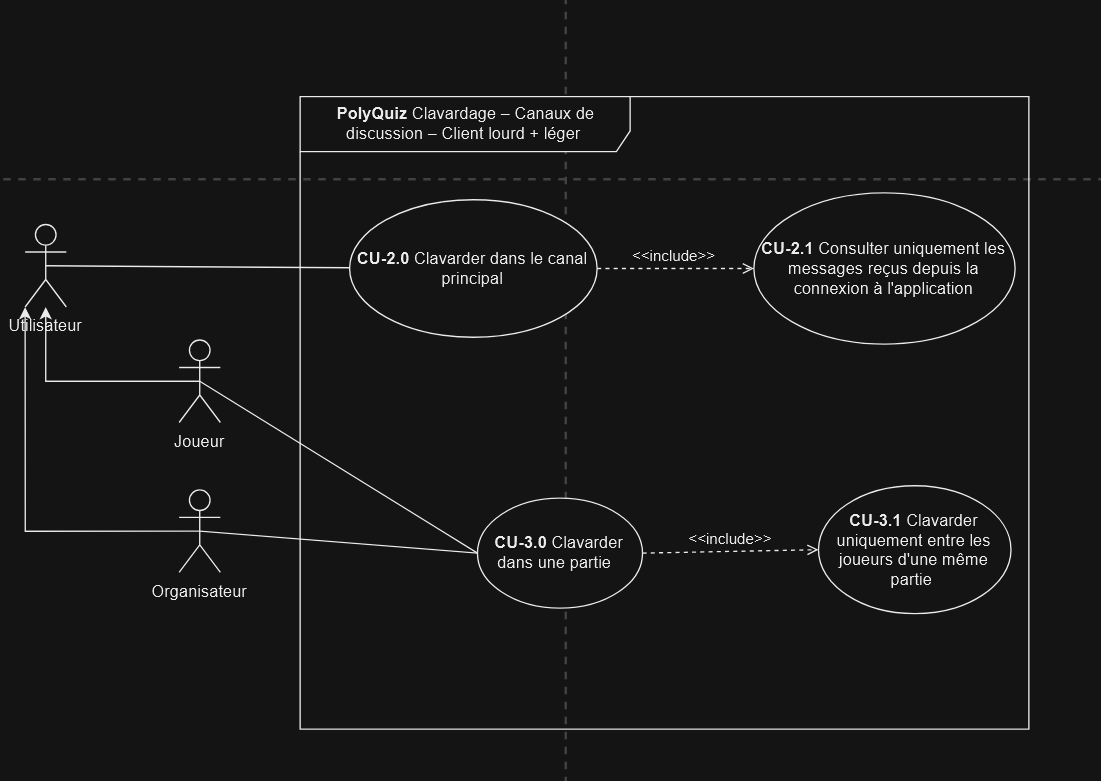
**2.7 Coût :** Si le nombre de comptes créés sur Auth0 dépasse 7500 comptes par mois, des frais de d’au moins 35$ devront être acquittés selon les besoins du système. De plus, si il y a un grand trafic sur AWS, des frais s'ajouteront selon ce dernier. Il faut donc éviter d’avoir des requêtes inutiles afin de minimiser le trafic.

# 

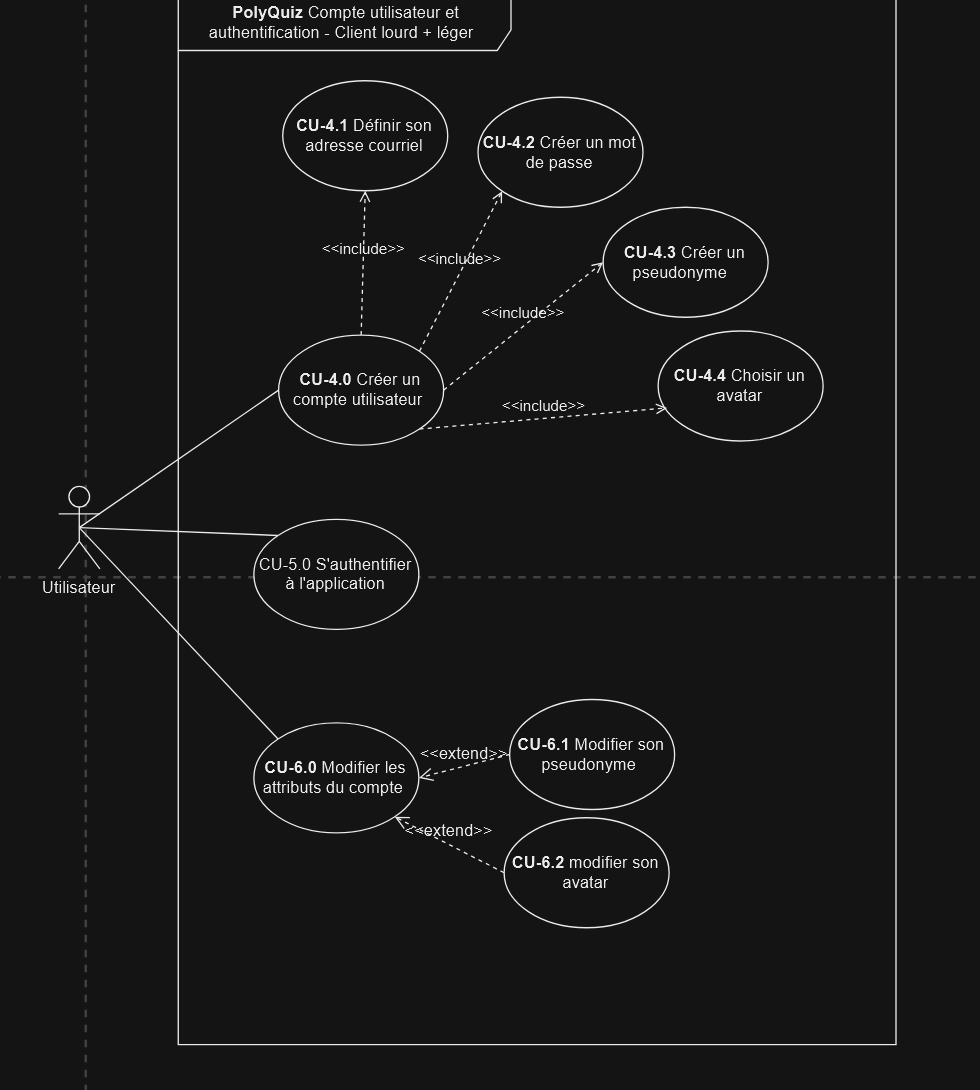
# 3. Vue des cas d’utilisation

**

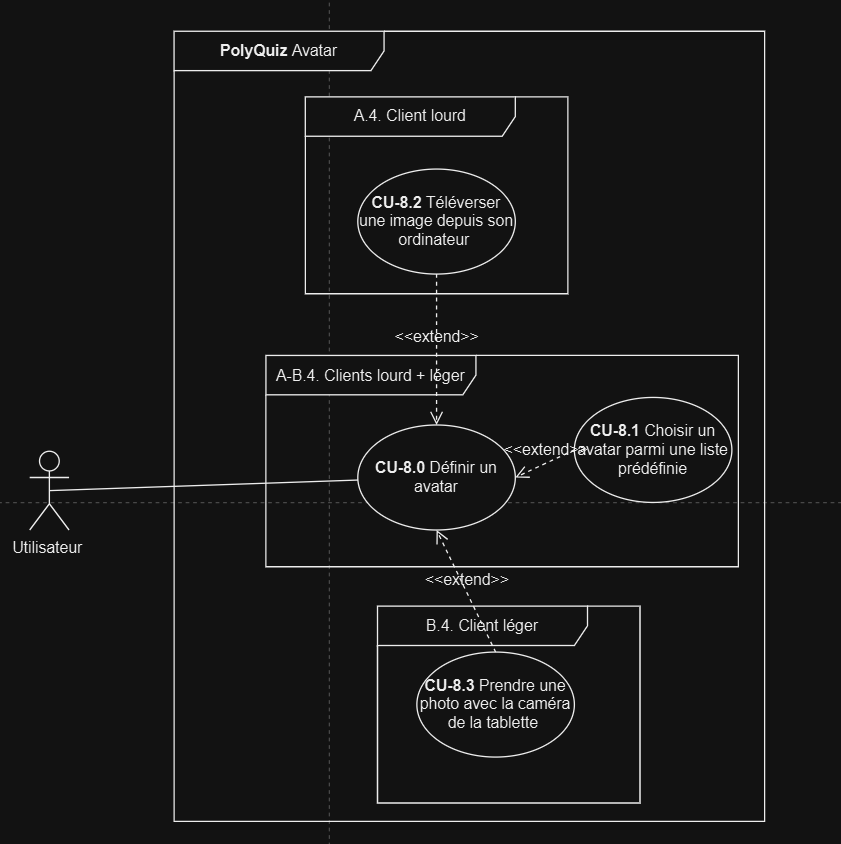
**Figure 3.0:** Diagramme de cas d’utilisation: Clavardage- Intégration



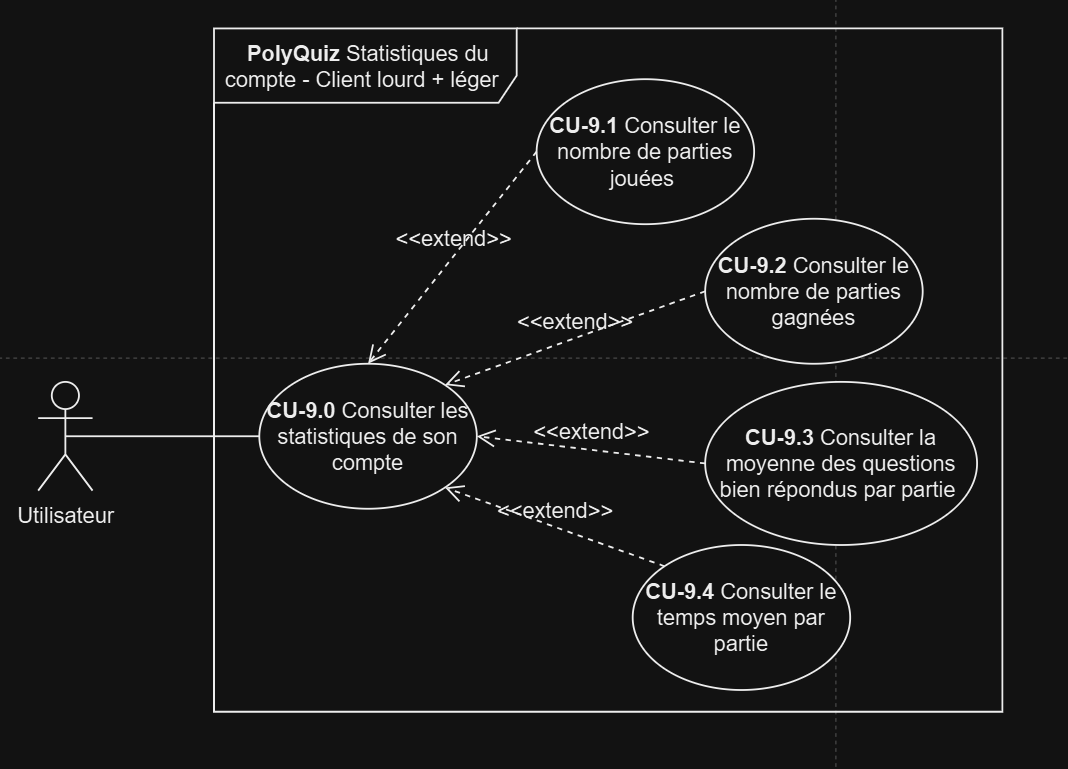
**Figure 3.1:** Diagramme de cas d’utilisation: Clavardage- Canaux de discussion



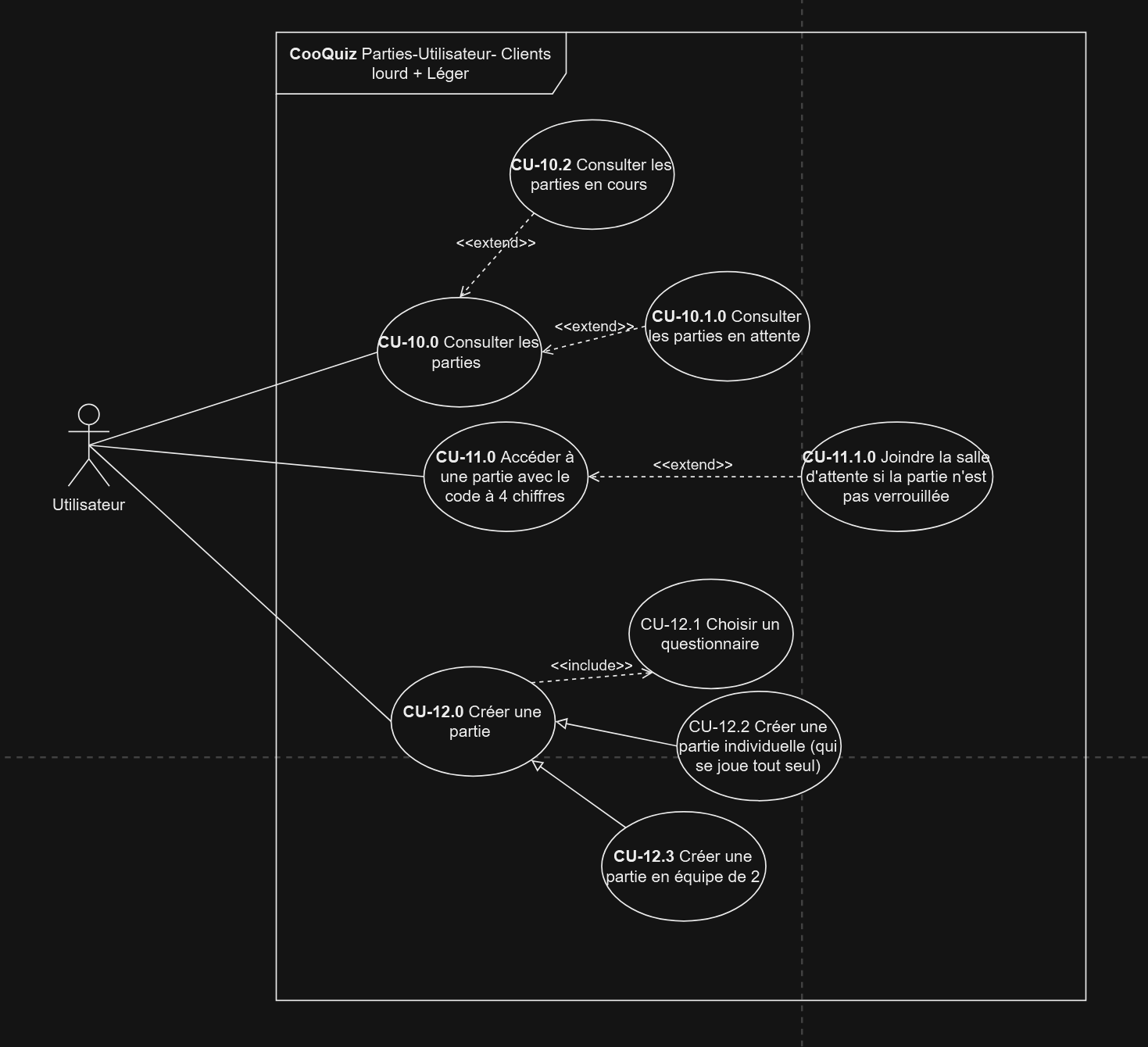
**Figure 3.2:** Diagramme de cas d’utilisation: Compte utilisateur et authentification



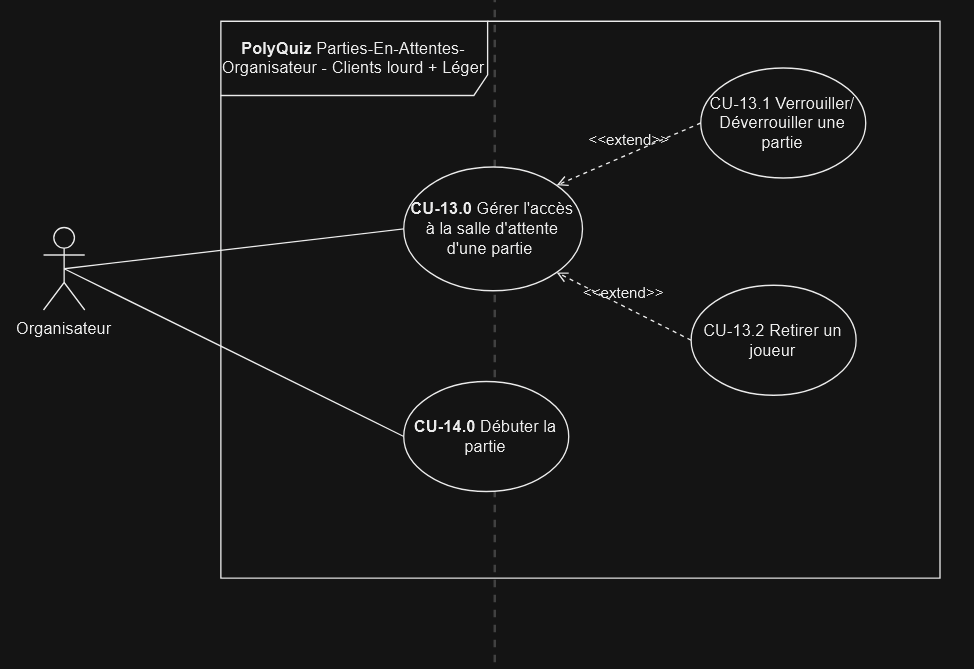
**Figure 3.3:** Diagramme de cas d’utilisation: Avatar



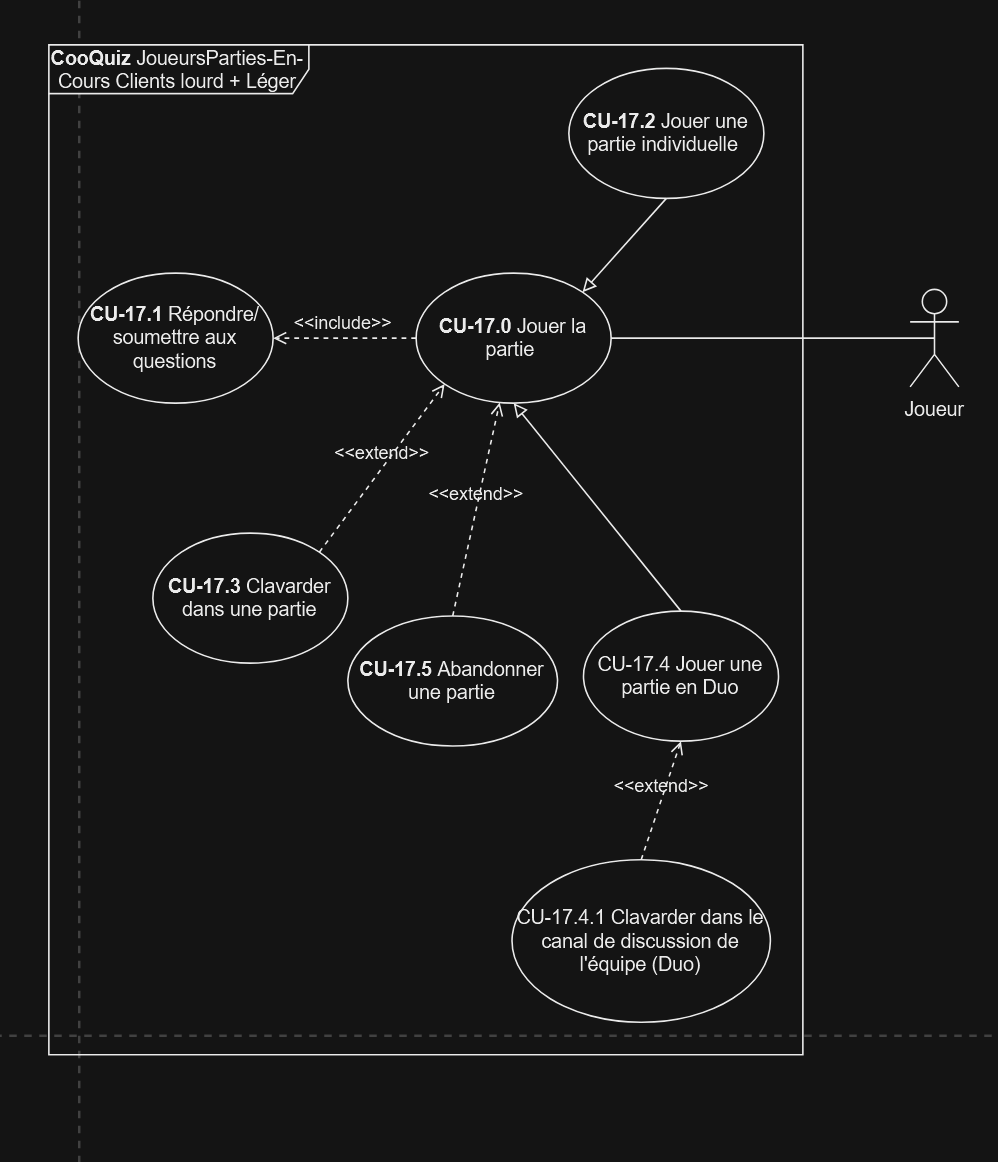
**Figure 3.4:** Diagramme de cas d’utilisation: Statistiques du compte



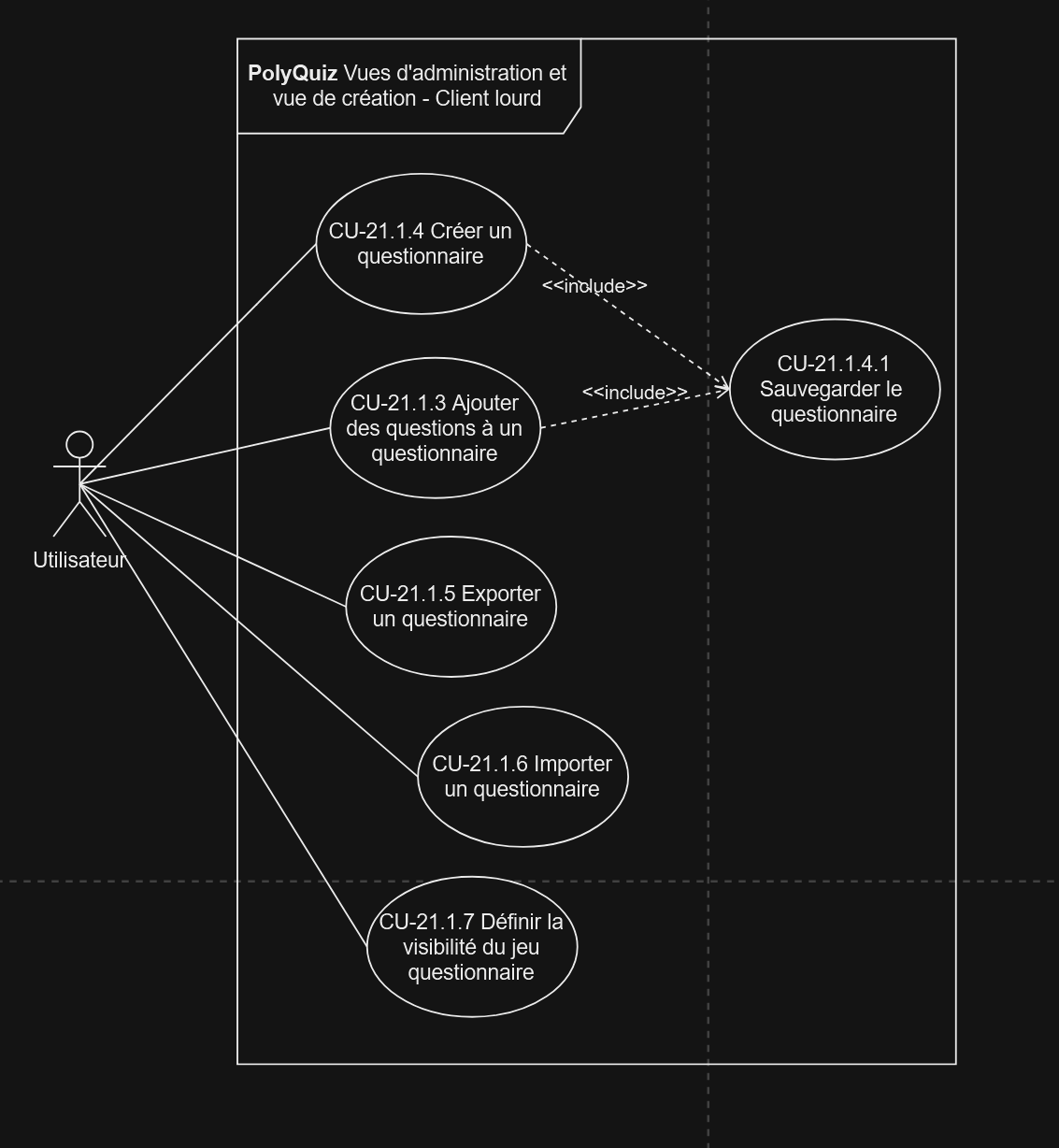
**Figure 3.6:** Diagramme de cas d’utilisation: Parties-Utilisateurs



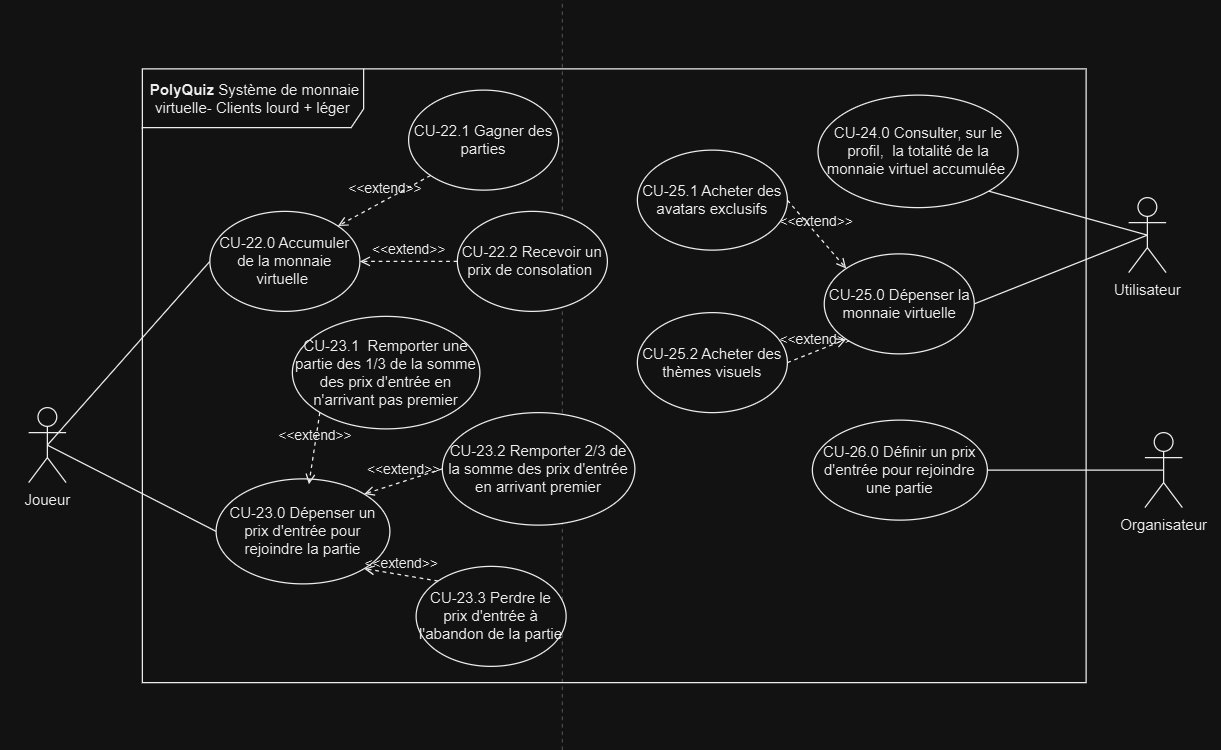
**Figure 3.7:** Diagramme de cas d’utilisation: Parties-Organisateurs-En-Attentes



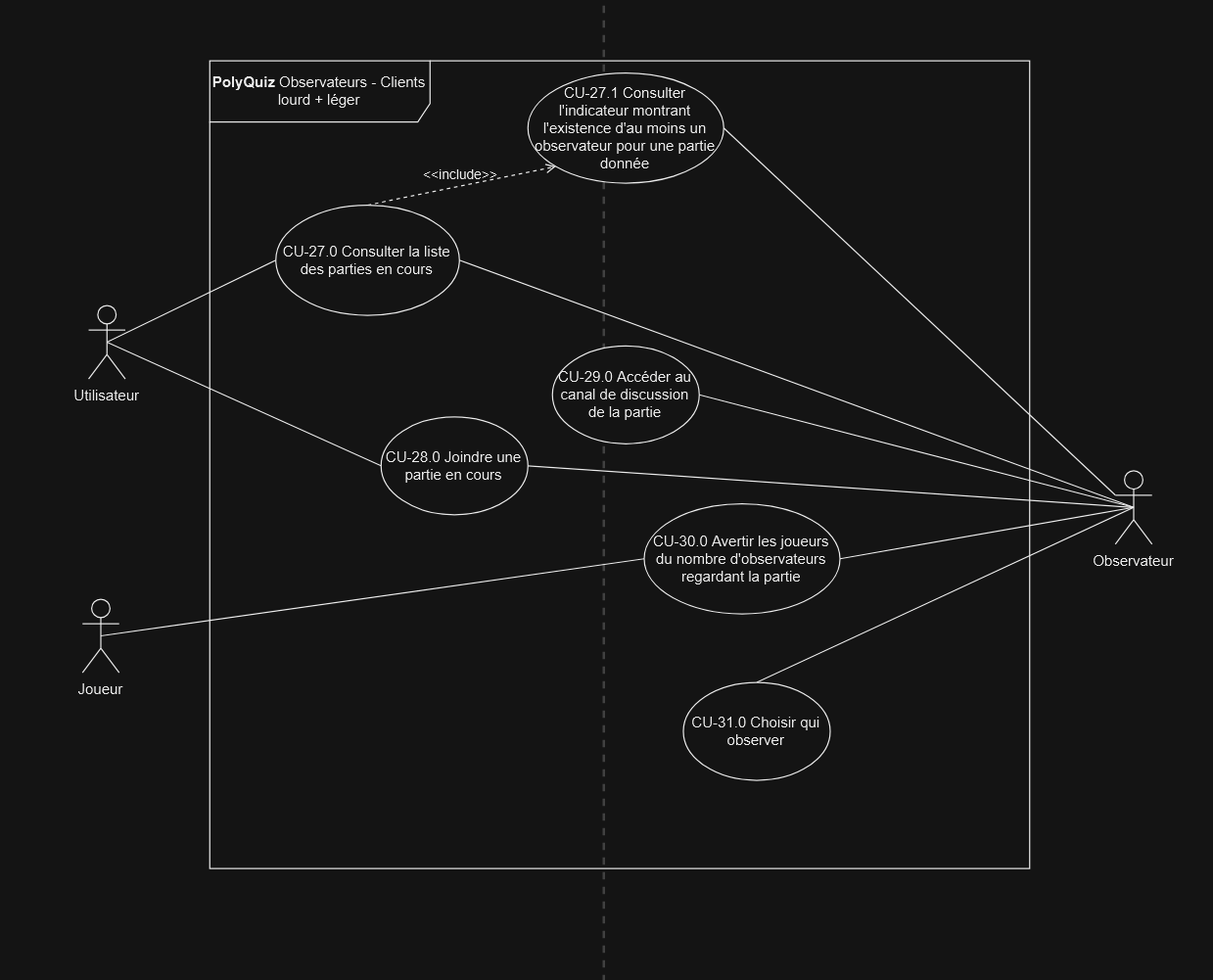
**Figure 3.8:** Diagramme de cas d’utilisation: Joueurs - En salle d’attente



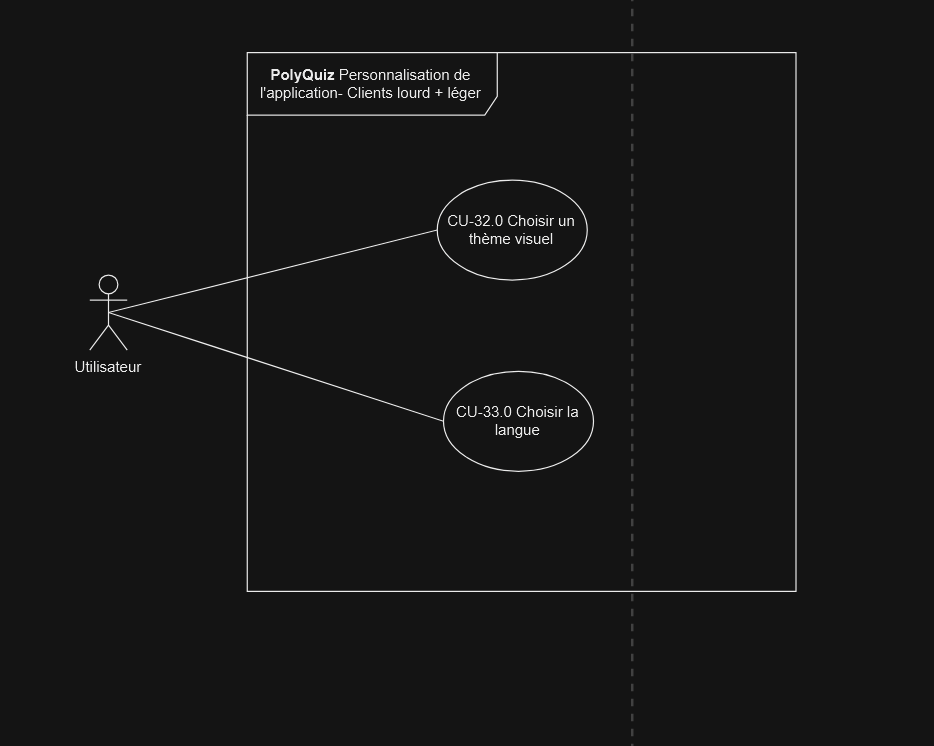
**Figure 3.9:** Diagramme de cas d’utilisation: Vue d’administration et vue de création



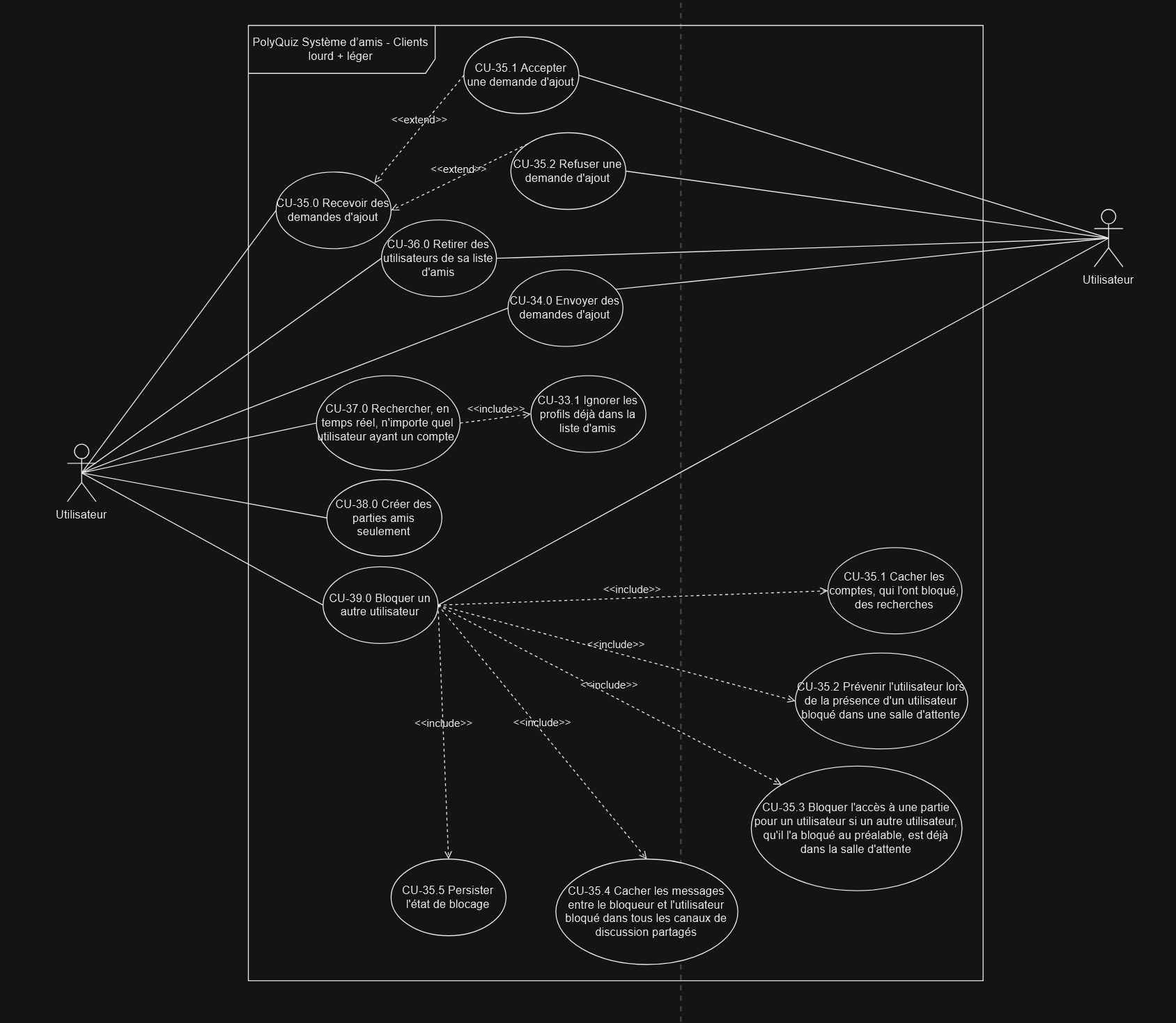
**Figure 3.10:** Diagramme de cas d’utilisation: Système de monnaie virtuelle



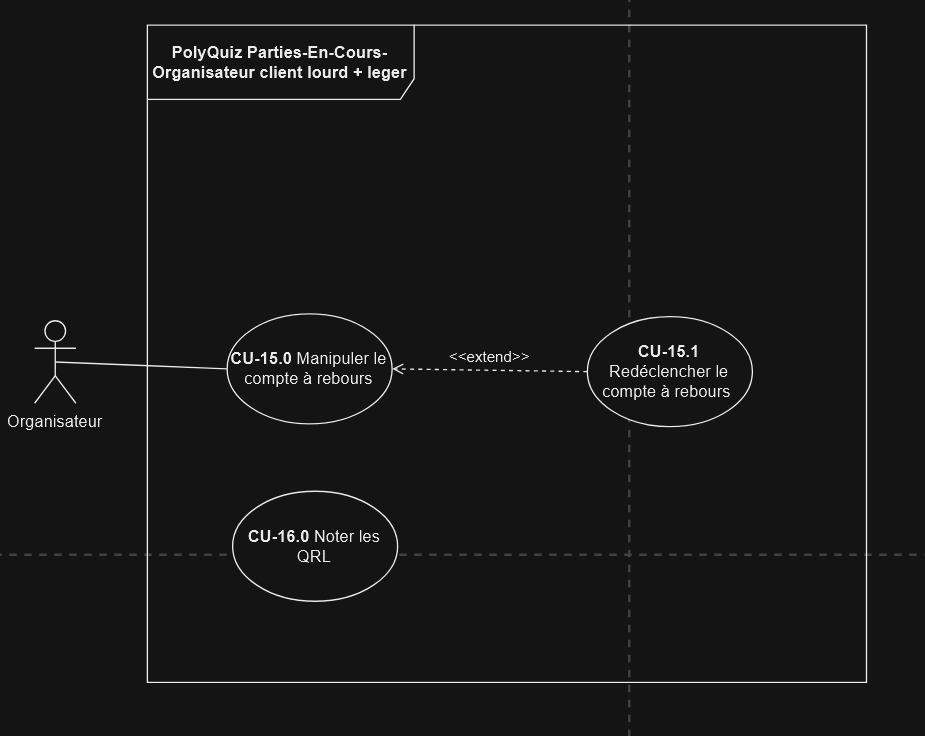
**Figure 3.11:** Diagramme de cas d’utilisation: Observateurs



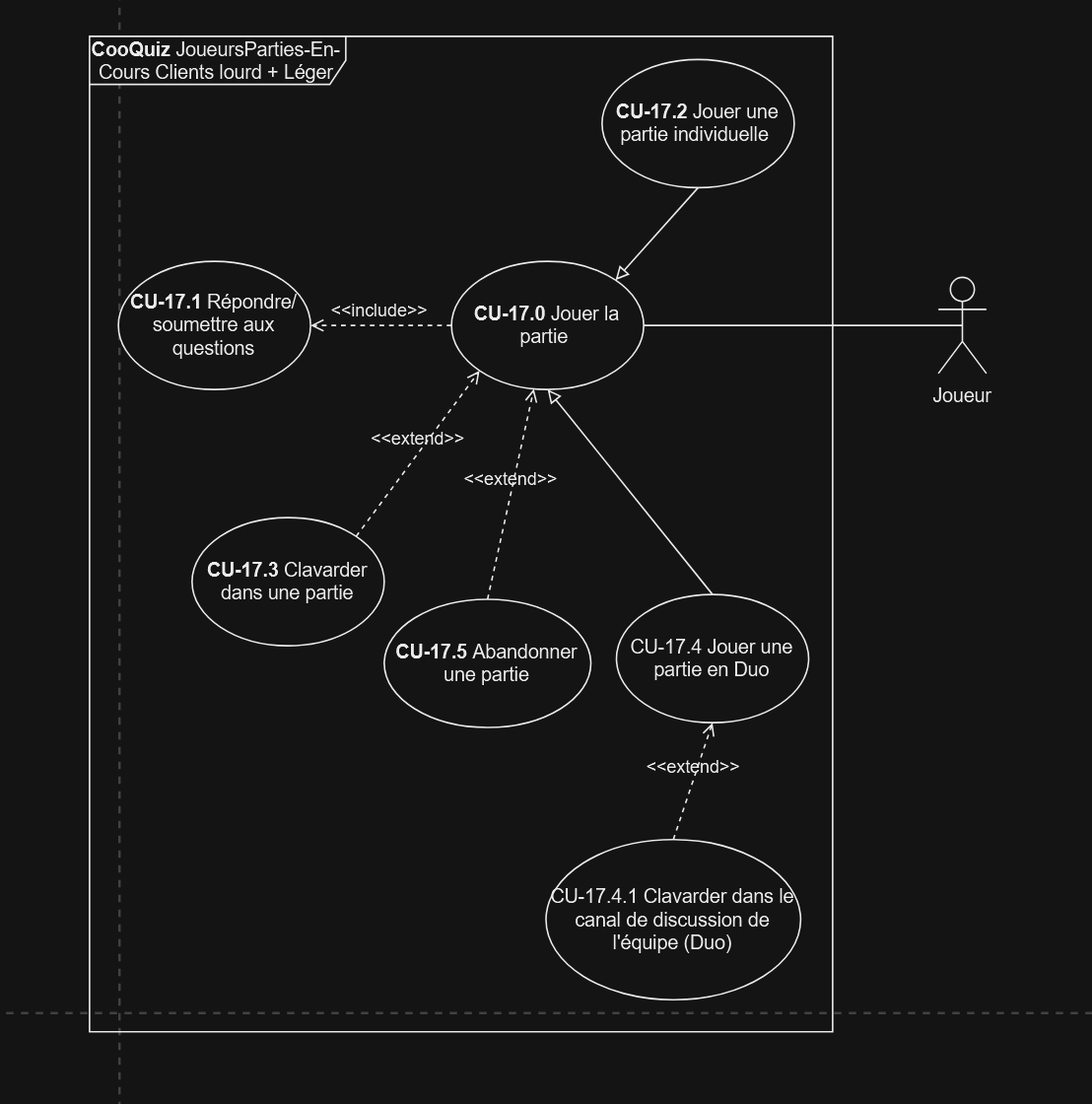
**Figure 3.12:** Diagramme de cas d’utilisation: Personnalisation de l’application



**Figure 3.13:** Diagramme de cas d’utilisation: Système d’amis

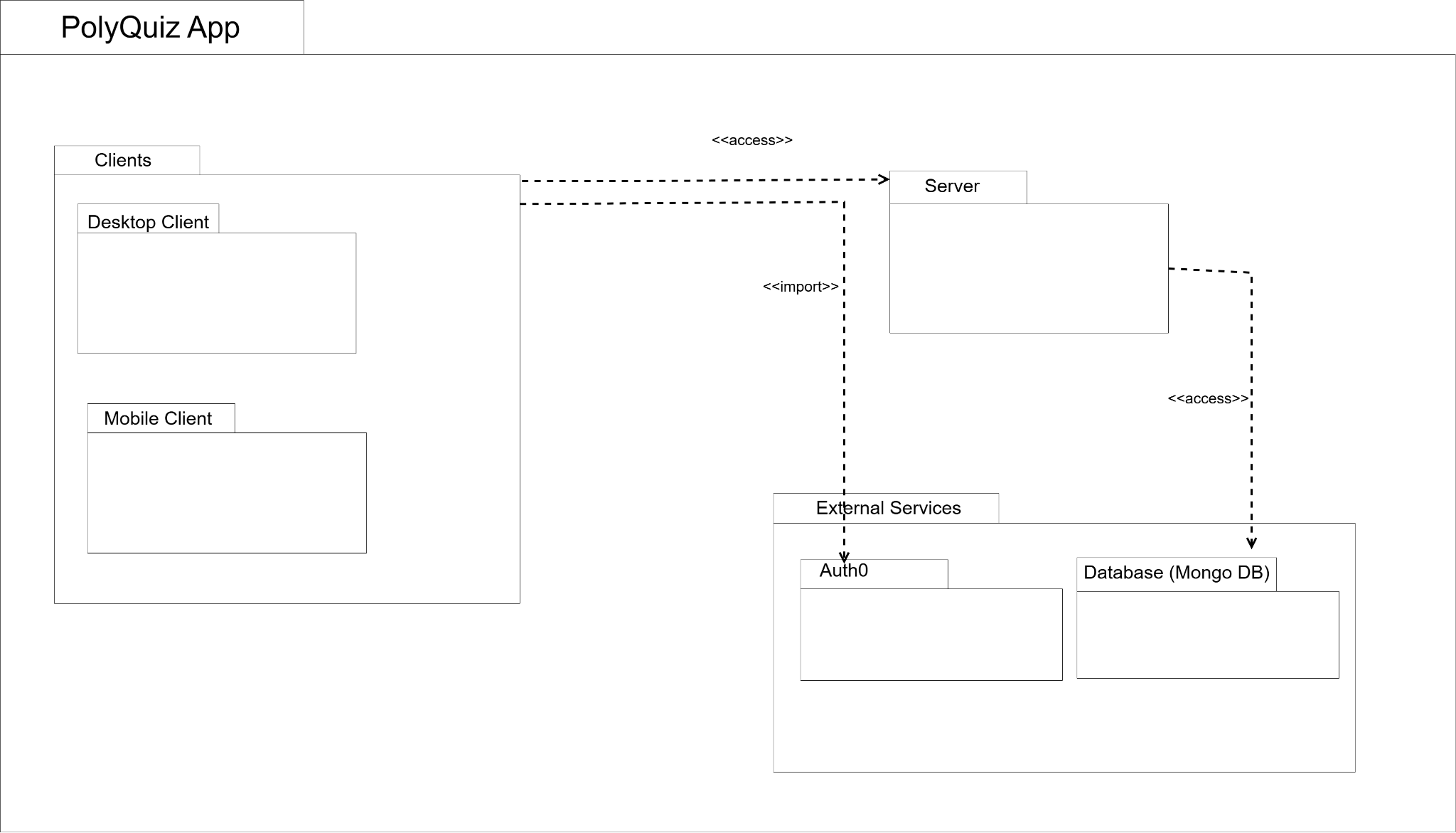
****

**Figure 3.14:** Diagramme de cas d’utilisation: Parties-En-Cours-Organisateur



**Figure 3.15:** Diagramme de cas d’utilisation: Joueurs - Parties En Cours

# 4. Vue logique

**

**Figure 4.1:** Diagramme de paquetage du logiciel PolyQuiz

| **<Clients>** |
| --- |
| *Ce paquetage inclut le client lourd et le client léger. Il représente toutes les fonctionnalités que les deux clients ont en commun.* |

| **<Desktop Client>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires au client lourd.* |

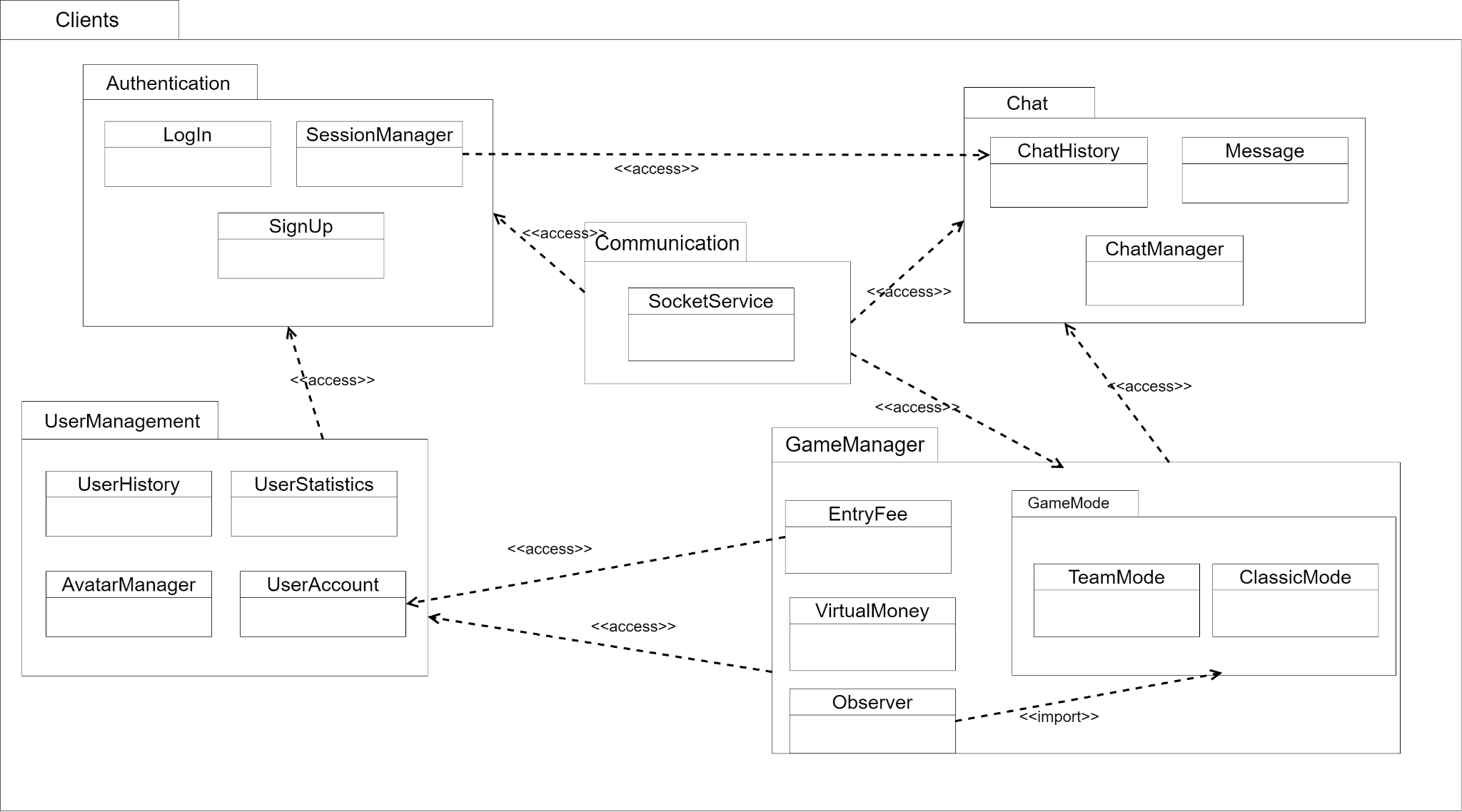
| **<Mobile Client>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires au client léger.* |

| **<Server>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires au serveur.* |

| **<External Services>** |
| --- |
| *Ce paquetage contient tous les services externes nécessaires au bon fonctionnement du processus de l’application.* |

| **<Auth0>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires pour la gestion de l’authentification des utilisateurs* |

| **<Database>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires à la persistance des données* |



**Figure 4.2:** Diagramme de Paquetage générique des éléments logiques **communs** entre le client lourd et léger

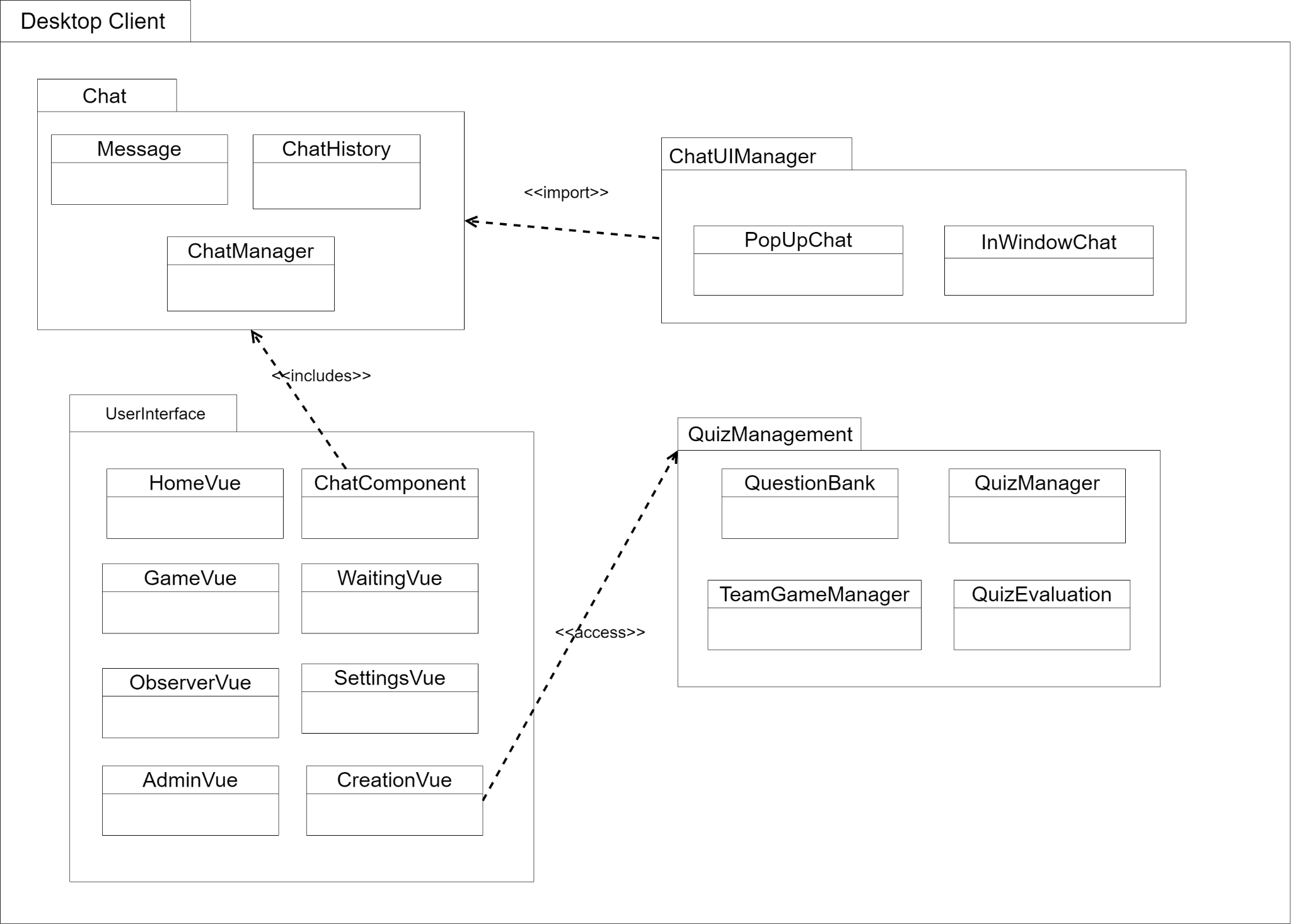
| **<Chat>** |
| --- |
| *Ce paquetage contient toutes les classes nécessaires pour la fonctionnalité de communication entre les utilisateurs.* |

| **<Authentication>** |
| --- |
| *Ce paquetage contient toutes les interfaces utilisateurs et services liés à l’authentification et la gestion des sessions des utilisateurs.* |

| **<Communication>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires à la communication et l’envoi de données de manière dynamique en temps réel.* |

| **<GameManager>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires au fonctionnement des jeux pour les joueurs, organisateurs et observateurs ainsi que la gestion des prix de parties.* |

| **<UserManagement>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires à la gestion des comptes et leurs données suite à l’authentification* |

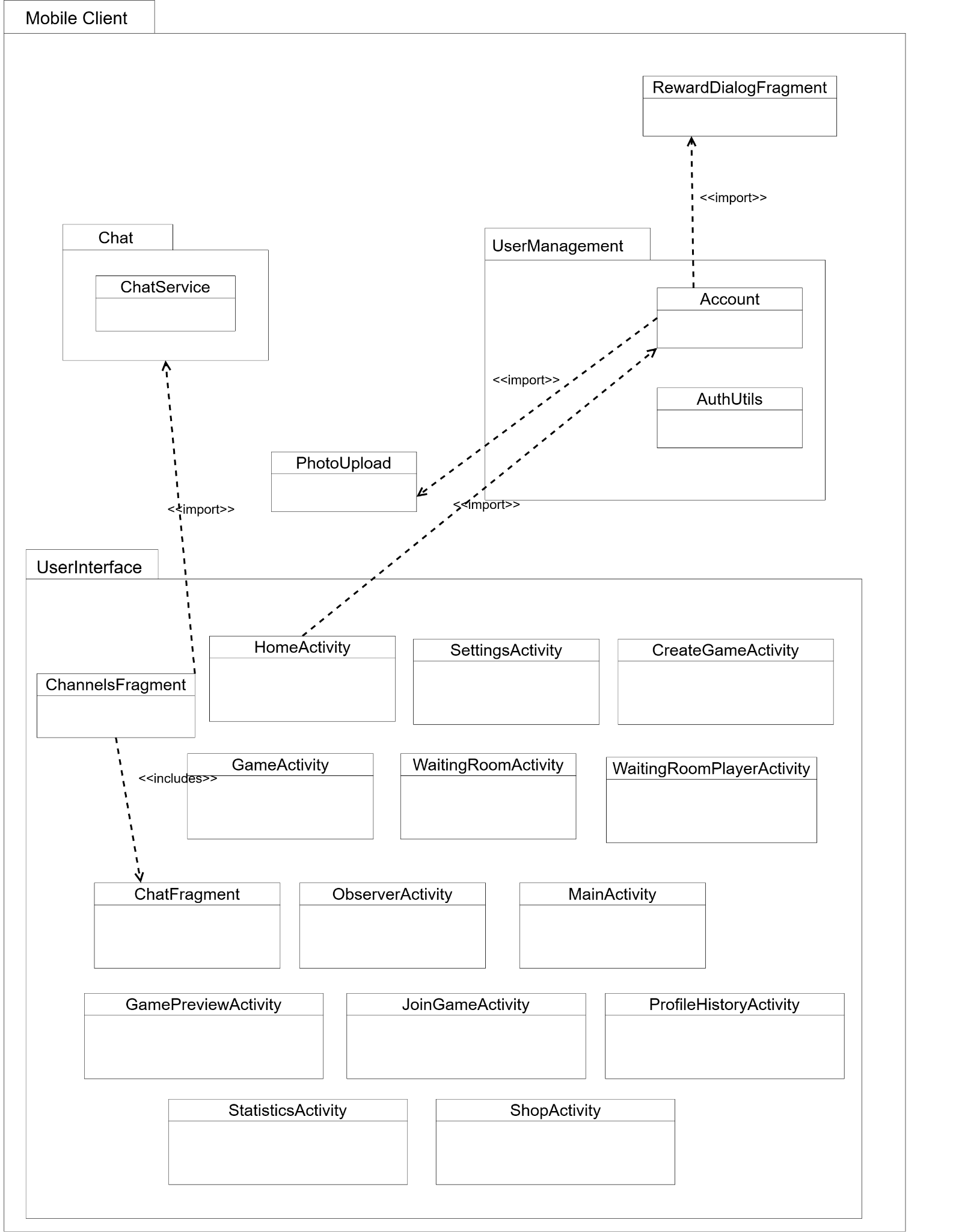


**Figure 4.3:** Diagramme de Paquetage générique du client lourd

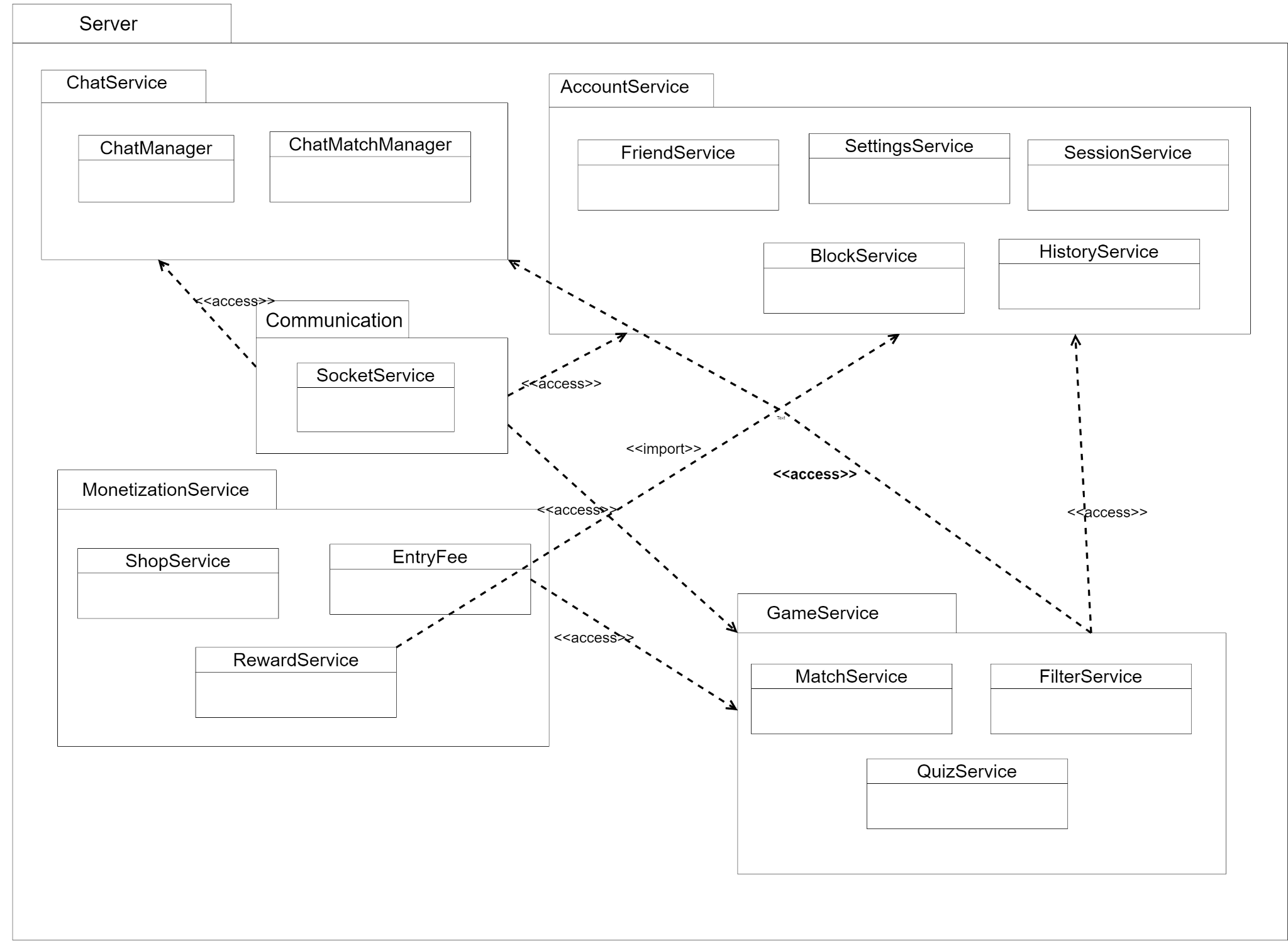
| **<ChatUIManager>** |
| --- |
| *Ce paquetage contient les fonctionnalités de gestion du chat pour le chat intégré et fenêtré* |

| **<QuizManagement>** |
| --- |
| *Ce paquetage contient toutes les fonctionnalités nécessaires au fonctionnement des jeux pour les joueurs, organisateurs et observateurs, de la création et modifications de quiz ainsi que la gestion des prix de parties.* |

| **<UserInterface>** |
| --- |
| *Ce paquetage regroupe les interfaces incluses dans le client lourd. Ce paquetage est différent de celui du client léger puisqu’il inclut des vues et non des ‘’Activities’’ comme sur Kotlin.* |



**Figure 4.4** Diagramme de paquetage du client léger



**Figure 4.5** Diagramme de paquetage du serveur

| **<ChatService>** |
| --- |
| *Ce paquetage contient toutes les classes nécessaires pour la fonctionnalité de communication entre les utilisateurs.* |

| **<AccountService>** |
| --- |
| *Ce paquetage contient toutes les classes nécessaires à la gestion des comptes pour leurs préférences, leurs amitiés, leurs utilisateurs bloqués, et leur historiques de parties.* |

| **<GameService>** |
| --- |
| *Ce paquetage contient toutes les classes nécessaires pour gérer la création et la modification de Quiz, la gestion des matchs ainsi que la gestion des notes attribués aux quiz.* |

| **<MonetizationService>** |
| --- |
| *Ce paquetage contient les classes ayant un lien avec la gestion de monétisation (monnaie virtuelle) comme les prix d’entrée aux parties, les éléments à acheter avec sa monnaie et le système de récompense du client léger.* |

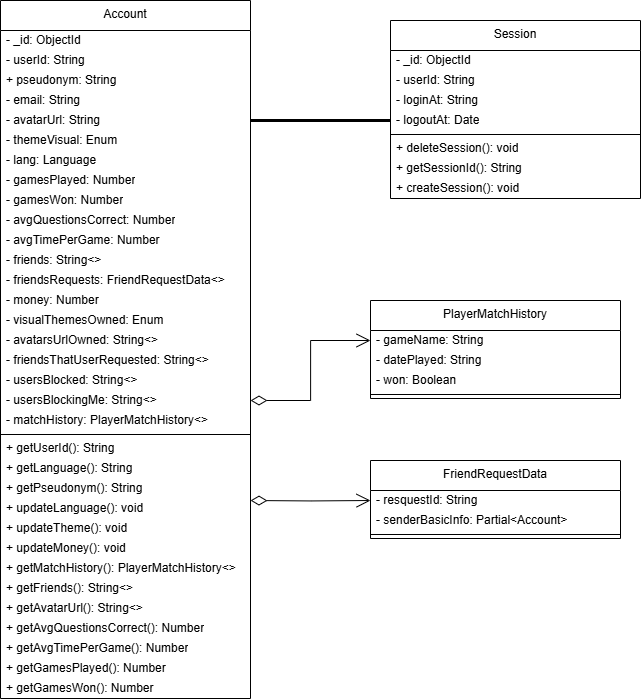
# 

# 

# 

# 

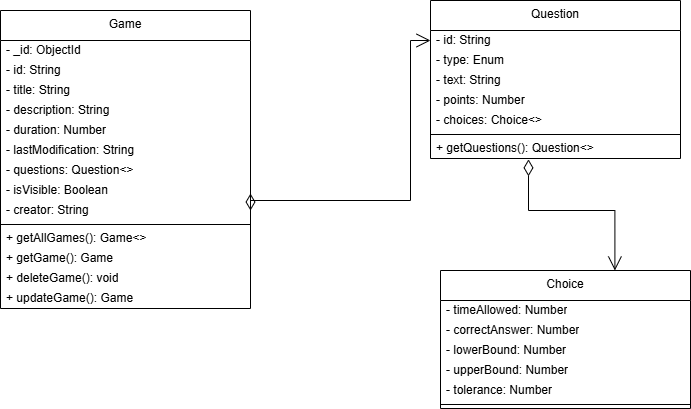
# 



\*\*\* les attributs \_id correspondent à une clé primaire générée par MongoDB à la création du document.

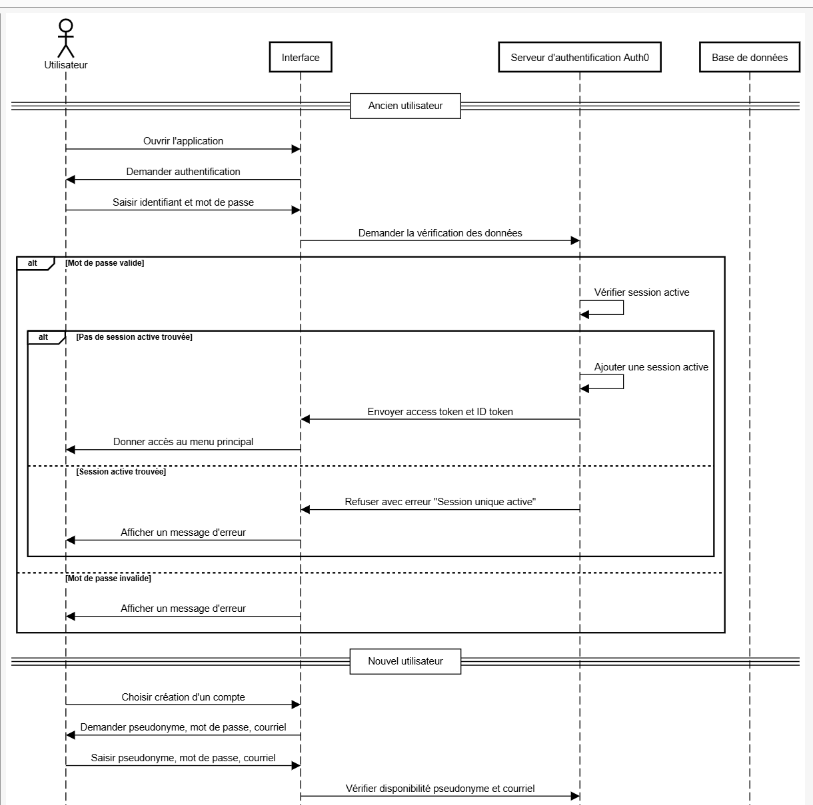
**Figure 4.7** Diagramme de classe des comptes utilisateurs et sessions au niveau de la base de donnée

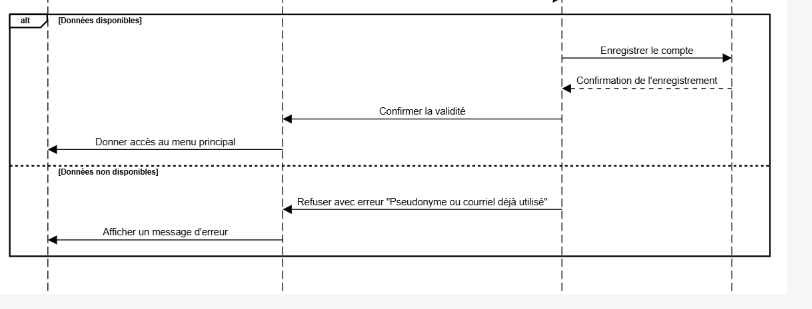
# 



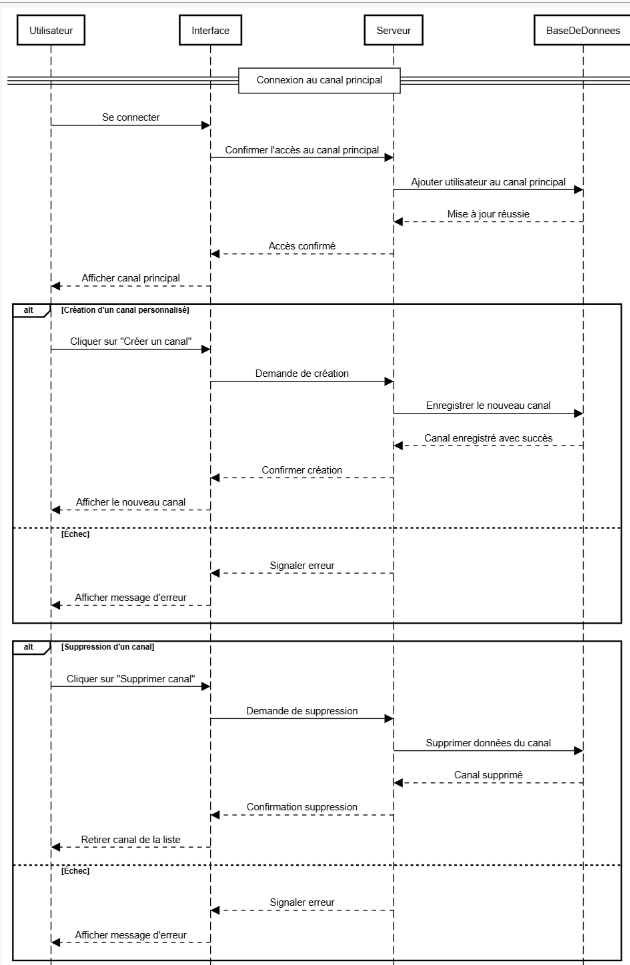
**Figure 4.7** Diagramme de classe des jeux au niveau de la base de donnée

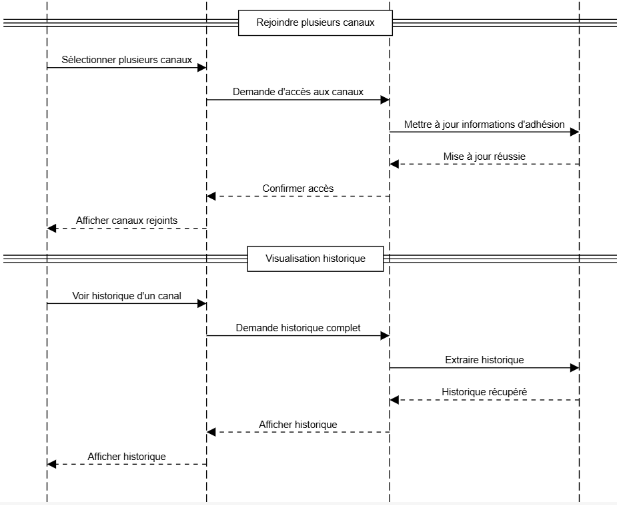
# 5. Vue des processus



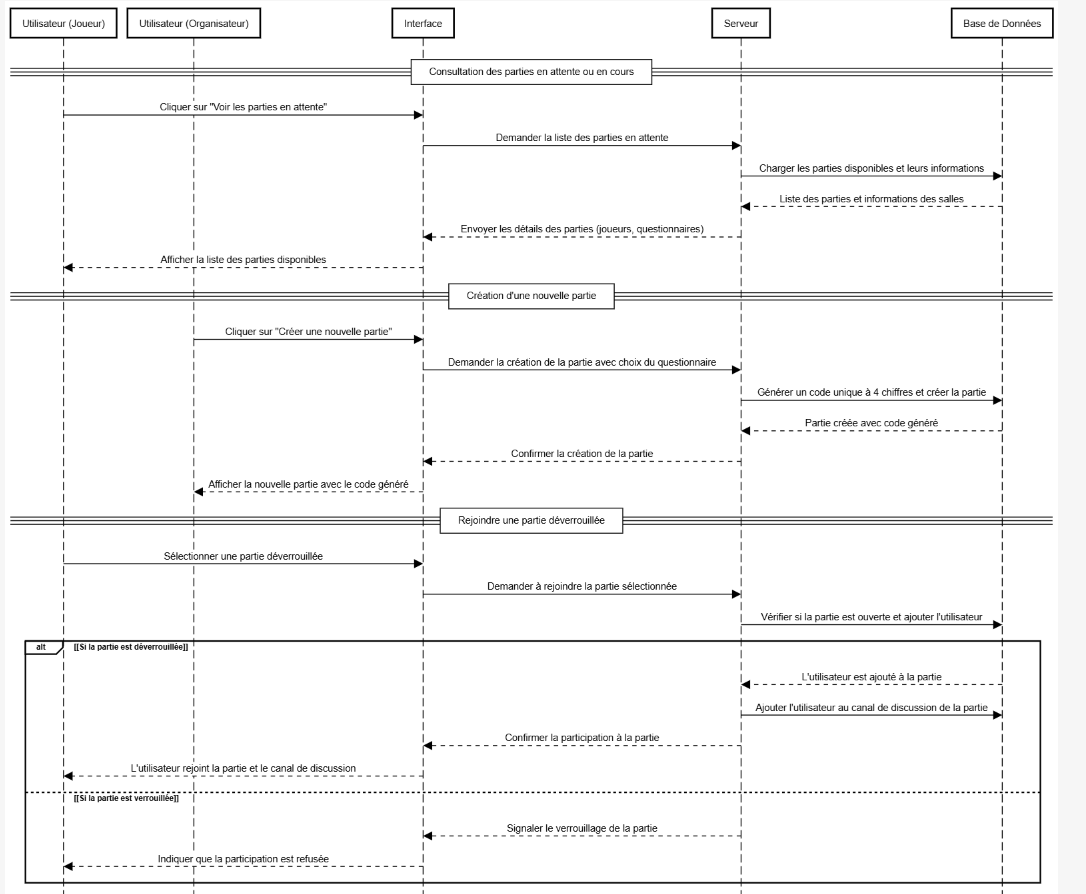


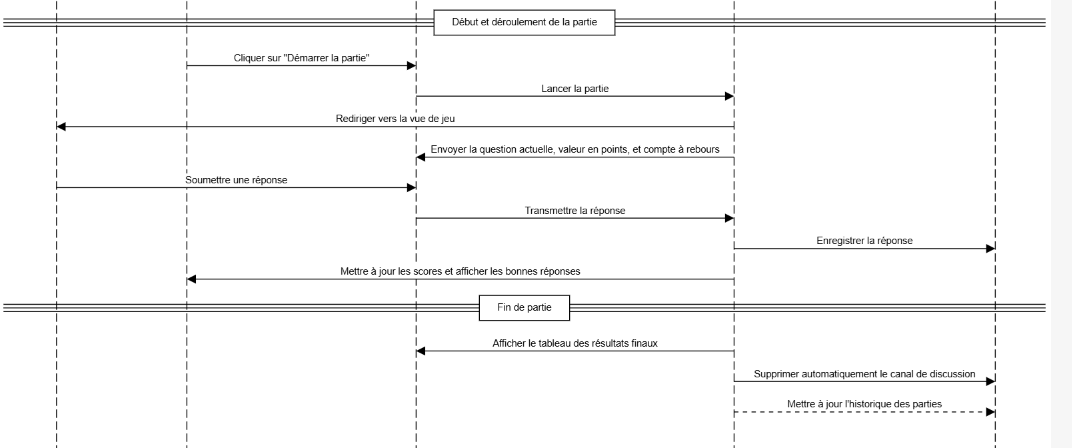
**Figure 5.1** Diagramme de processus pour la création d’un compte utilisateur et connexion

**

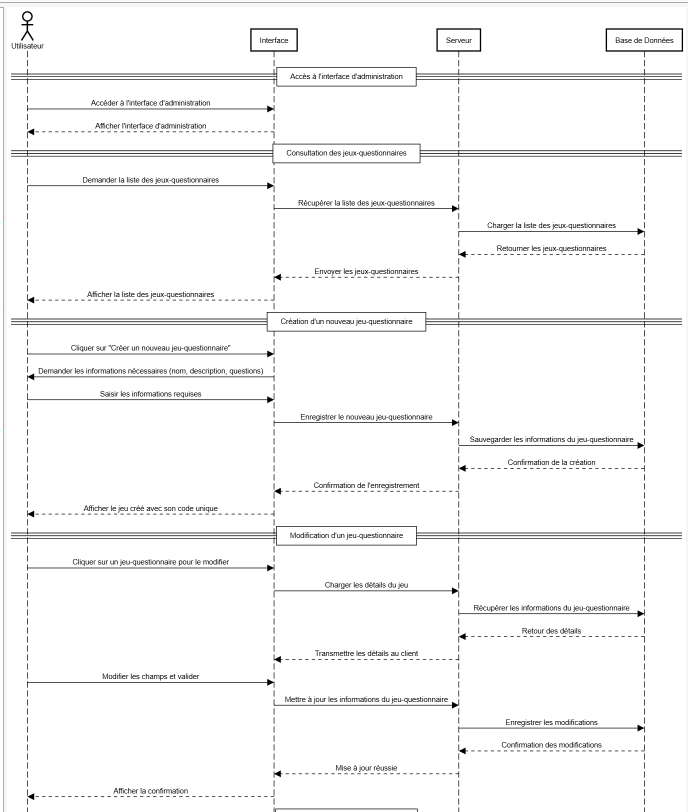
**

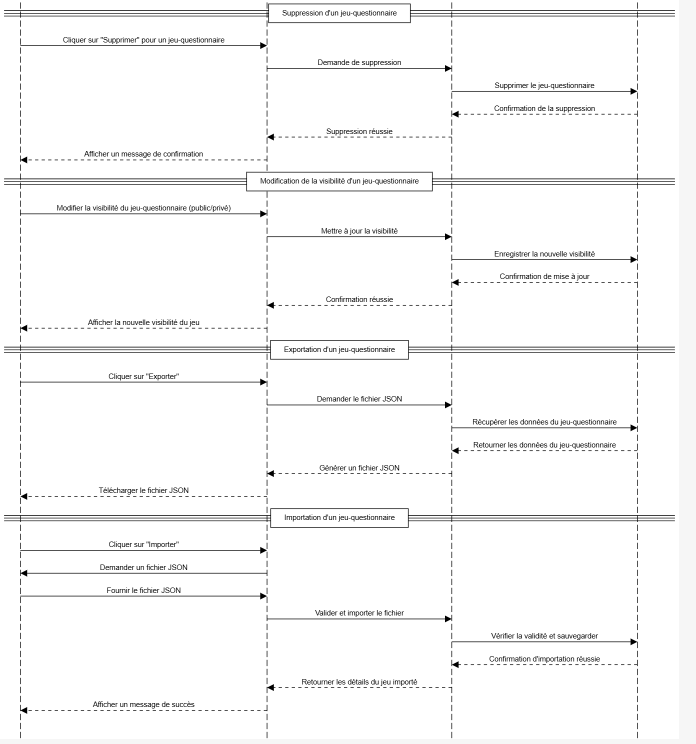
**Figure 5.2** Diagramme de processus pour le système de clavardage

**

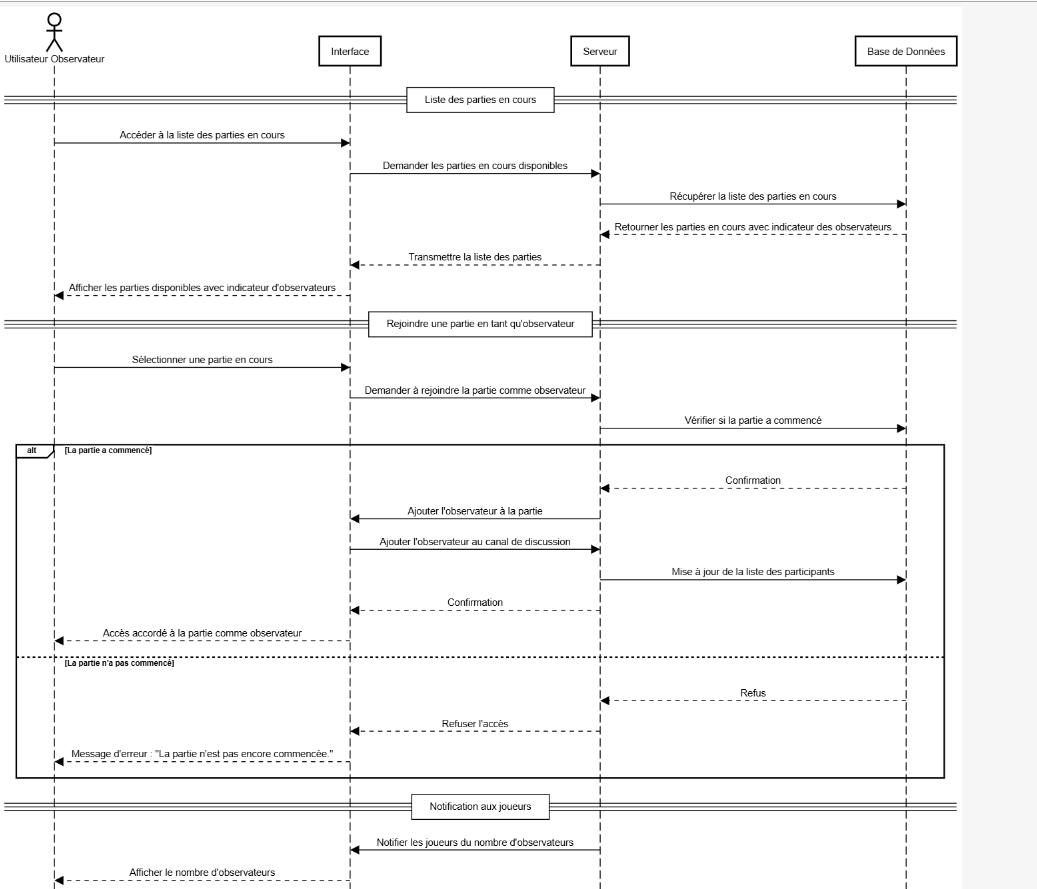
**

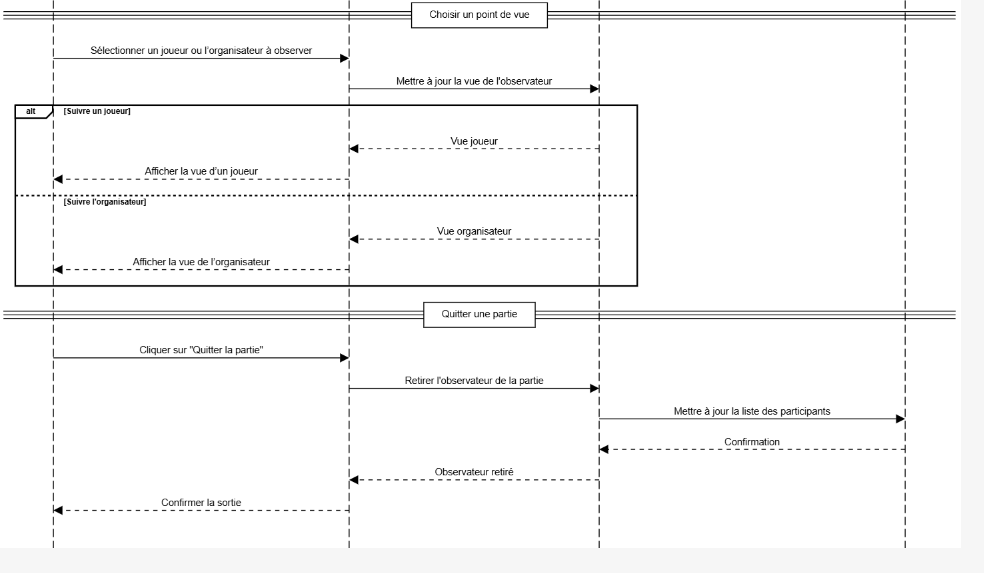
**Figure 5.3** Diagramme de processus pour le mode de jeu

**

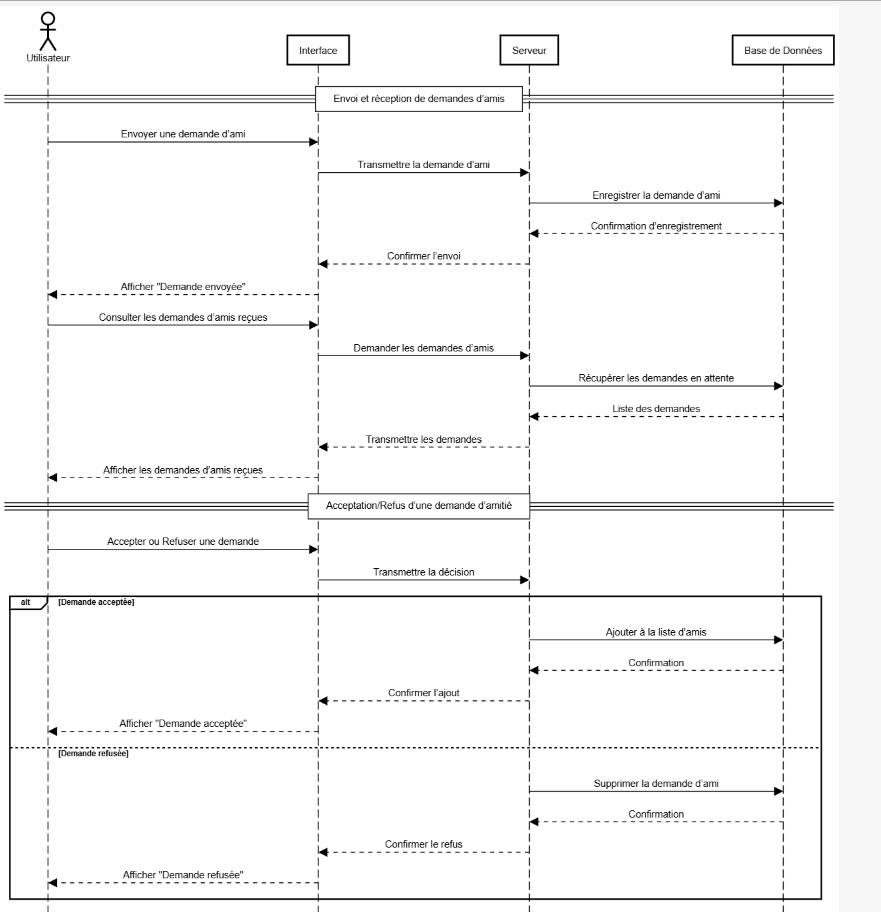
**

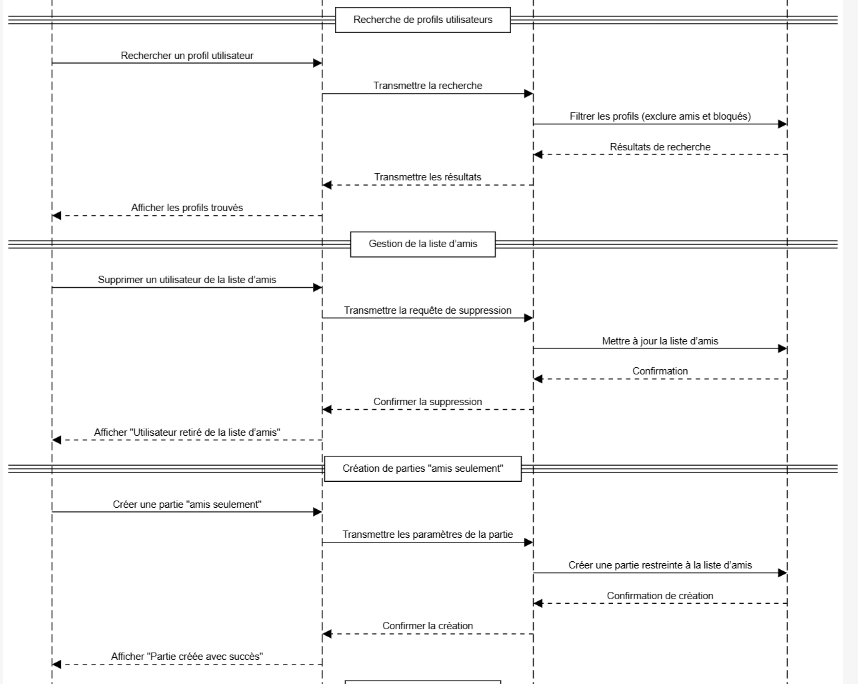
**Figure 5.4**  Diagramme de processus pour la vue d’administrateur

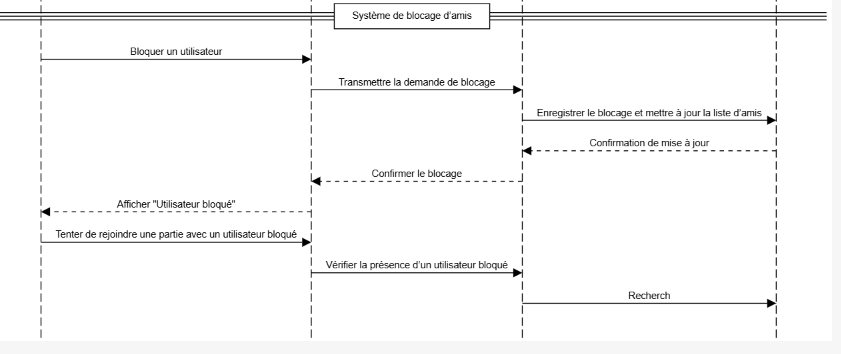
**

**

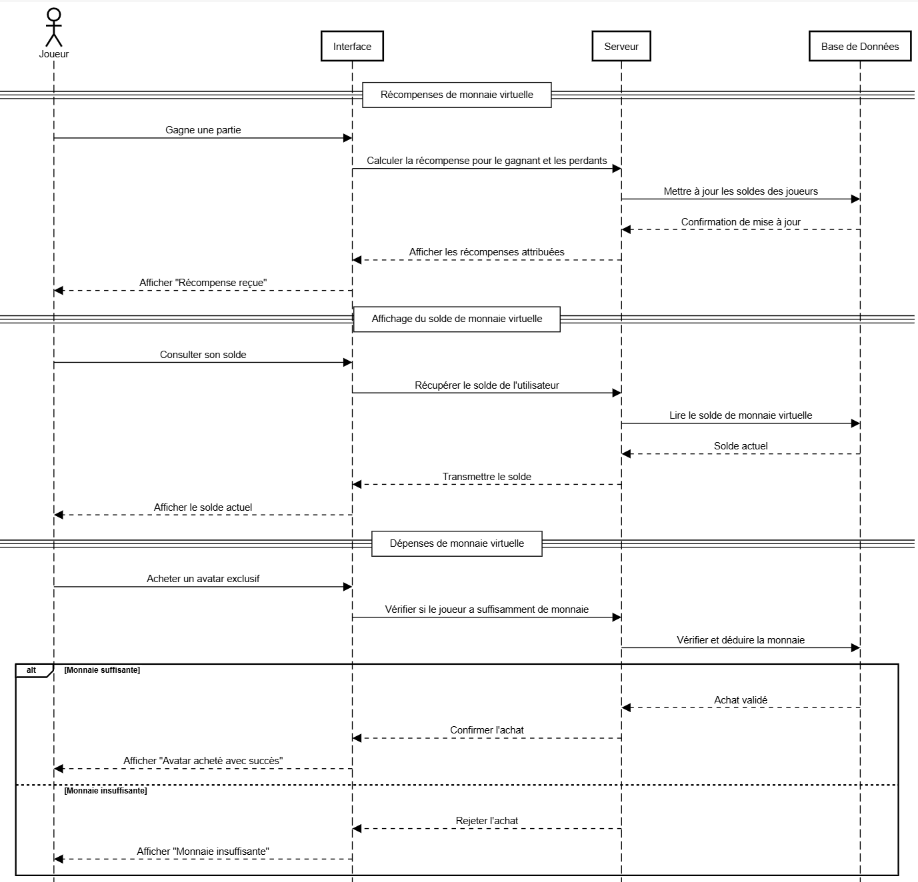
**Figure 5.5** Diagramme de processus pour l’observateur

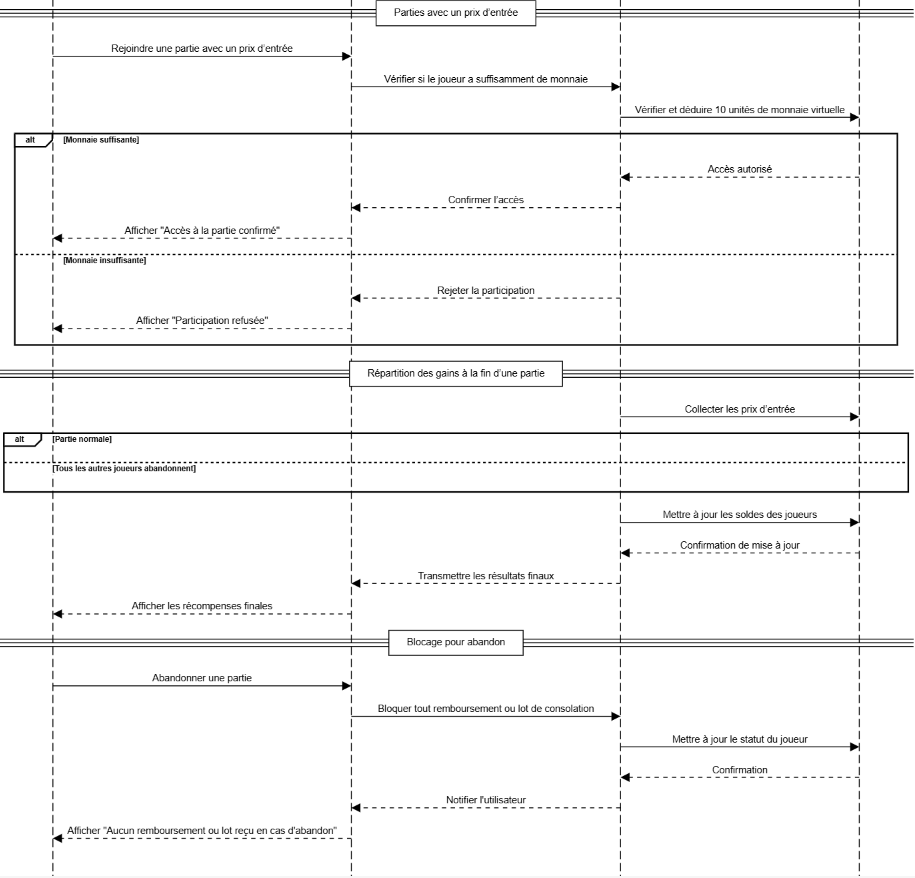
**

**

**

**Figure 5.6** Diagramme de processus pour le système d’amis





**Figure 5.7** Diagramme de processus pour le système de monnaie virtuelle

# 6. Vue de déploiement

# 

**Figure 6.1** Diagramme de vue de déploiement

# 7. Taille et performance

Étant donné que la plateforme sera exécutée sur 2 clients (un client lourd et un client léger) et un serveur, il faut prendre en compte les caractéristiques des appareils énumérés dans le plan de projet sur lesquels la plateforme sera utilisée et les différentes exigences décrites dans le SRS.

Premièrement, la plateforme devra être fluide et performante pour permettre aux utilisateurs une utilisation agréable. Un taux de rafraîchissement de 60FPS est suffisant pour garantir la fluidité de la navigation dans l’application. L’application devra pouvoir supporter 10 utilisateurs connectés en même temps. Les requêtes au serveur serveur doivent être optimisées afin de permettre d’afficher les informations à l’utilisateur en moins d’une seconde. De plus, le délai de réception d’un message dans le clavardage doit être de maximum 100ms. En ce qui concerne le temps de réception des informations de la base de données, il ne faut pas qu’il dépasse 500ms. Le temps de réponse du serveur vers le client devra se situer entre 50ms et 70ms. La vitesse de téléversement devrait être d’au moins 10Mbps. En respectant tous les délais mentionnés, la plateforme assure à l’utilisateur une utilisation agréable et sans délai notable.

Ensuite, en ce qui concerne le client lourd, la mémoire vive 200Mo utilisée ne devrait pas dépasser, la mémoire occupée sur le disque dur devra être au plus 500Mo. Pour le client léger, l’utilisation de mémoire vive devra être au plus de 150Mo et la taille de l’APK ne devra pas dépasser 100Mo.

Pour continuer, l’utilisation du CPU devra être de maximum 30% pour les deux clients. Le CPU utilisé sur la machine roulant le client lourd devrait avoir au moins 2 coeurs, ayant une fréquence de 3 GHz ou plus. Les versions des processeurs recommandés seraient donc Intel Core i5 ou AMD Ryzen 5 ou bien des versions suivant celles mentionnées.

En ce qui concerne le serveur, il devrait avoir 2 coeurs pour le vCPUS et une RAM de 8 Go. De plus, le stockage devrait être au moins de 100 Go afin de stocker les données de l’application.

Pour la base de données, l’espace en disque nécessaire est d’au moins 50 Go pour pouvoir gérer les besoins de l’application.