

Software Requirements Document: Quoridor

MERIAN Emile, ABRAHAM Théo, VINOVRSKI Alexandre, DIMITROV Mark,
CHICA COBO Sofia, HORII Hisao, LEDUC Camille, BOLLENGIER Thomas

23 Novembre 2021



Contents

1	Introduction	4
1.1	But du projet	4
1.2	Glossaire	4
2	Besoins utilisateur: Fonctionnels	5
2.1	Liste d'amis	5
2.1.1	Ajouter un ami	6
2.1.2	Accepter une demande d'ami	6
2.1.3	Supprimer un ami	6
2.1.4	Discuter avec un ami	6
2.1.5	Afficher un profil	6
2.1.6	Barre de recherche	6
2.1.7	Ajouter un ami depuis un salon	6
2.2	Menu	6
2.2.1	Créer une partie	6
2.2.2	Charger une partie	6
2.2.3	Rejoindre une partie	7
2.2.4	Rejoindre un ami	7
2.2.5	Profil	7
2.2.6	Classement	7
2.3	Dans un salon	7
2.4	En partie	8
2.4.1	Abandon d'une partie en cours	8
2.4.2	Mettre une partie en pause et la sauvegarder	8
3	Besoins utilisateur: Non fonctionnels	9
3.1	Gestion du compte	9
3.1.1	Création de compte	9
3.2	Mot de passe	9
3.2.1	Connexion	9
3.2.2	Suppression de compte	9
3.3	Configurer une partie	9
3.3.1	Choisir un mode de jeu	10
3.3.2	Choisir le nombre de joueurs	10
3.3.3	Choisir les paramètres	10
3.4	En partie	10
3.4.1	Jouer en terminal	11
3.4.2	Jouer en interface	11
3.5	Fin de partie	12
4	Besoin système: Fonctionnels	13
4.1	Gestion des comptes	13
4.1.1	Création d'un compte	13
4.1.2	Suppression d'un compte	13
4.2	Mot de passe	13
4.2.1	Connexion	13
4.3	Gestion des profils	14
4.3.1	Affichage d'un profil	14
4.3.2	Sauvegarder les statistiques du joueur	14
4.4	Chat	14

4.5	Gestion d'une partie	14
4.5.1	Gestion du plateau	16
4.5.2	Gestion des pions	16
4.5.3	Gestion des murs	16
4.5.4	Gestion des tours	16
4.6	Rejoindre une partie	17
4.6.1	Matchmaking rapide: trouver un salon ouvert	17
4.6.2	Rejoindre un ami	17
4.7	Sauvegarde	18
4.8	Classement des joueurs	18
5	Besoins système: Non fonctionnels	19
5.1	OS	19
5.2	Réseau	19
5.3	Disponibilité	19
5.4	Performances	19
5.5	Capacité	19
5.6	Sécurité	19
5.6.1	Base de données	19
5.6.2	Mot de passe	19
5.7	Robustesse	19
6	Design et fonctionnement du Système	20
6.1	Design du système	20
6.2	Fonctionnement du système	20
6.2.1	Inscription	20
6.2.2	Connexion	21
6.3	Design des modes de jeux	21
6.3.1	DestruQtion	21
6.3.2	QQQuoridor	22
6.3.3	Blitz	22
6.3.4	Double Up	22
6.4	Design de l'IA	22
7	Implémentation des différentes parties	24
7.1	Jeu Quoridor	24
7.1.1	Version classique	24
7.1.2	Modes de jeu	25
7.2	Base de données	26
7.3	Serveur	26
8	Implémentations manquantes	27
9	Historique	29

1 Introduction

1.1 But du projet

Le but du projet est de nous apprendre au travers d'un objectif commun à nous familiariser avec l'UML et la confection d'un SRD, à travailler dans une équipe d'inconnus, à nous servir de serveurs, de bases de données et à perfectionner notre utilisation du langage C++.

Dans ce cadre, l'ULB nous a donc confié la mission de créer une application d'un jeu de Quoridor dans le but d'informatiser la ludothèque de l'ULB et permettre aux étudiants d'apprécier les joies de la ludothèque dans les normes de la crise sanitaire et durant des possibles futurs confinements.

1.2 Glossaire

Glossaire d'interactions :

- Utilisateur : un utilisateur est un individu qui interagit avec l'application.
- Joueur : un utilisateur qui souhaite jouer ou qui joue sur l'application.
- Compte : un compte rassemble toutes les informations de son utilisateur au sein de l'application.
- Profil : un profil est l'ensemble des informations rassemblées par un compte qui sont visibles par les autres utilisateurs.
- Pseudonyme (Pseudo) : un pseudonyme est le nom qui identifie l'utilisateur au sein de l'application. Il identifie également le compte sous lequel l'utilisateur est connecté dans la base de données du serveur.
- Salon : Un salon est un espace virtuel dans lequel des joueurs se regroupent afin de lancer une partie de Quoridor.
- includes : un cas d'utilisation incorpore explicitement et de manière obligatoire un autre cas d'utilisation à l'endroit spécifié.
- excludes : un cas d'utilisation incorpore implicitement de manière facultative un autre cas d'utilisation à l'endroit spécifié.

Glossaire système :

- Base de données : Une base de données est une méthode de stockage de données. Elle se trouve au niveau du serveur et permet d'accéder et de gérer facilement les données.

Glossaire de jeu :

- Jeu : un jeu est une partie de Quoridor ou d'un mode de jeu de l'application.
- Case/Cellule/Cell : une cellule représente une case du plateau de jeu sur lequel se déplacent les pions ou où peuvent être placés des murs. Les cellules sont séparées en deux catégories : PawnCell et WallCell.
- Mur : un mur est un élément que les joueurs peuvent placer sur deux cases WallCells pour empêcher le déplacement d'un pion entre les cases bloquées par le mur.

2 Besoins utilisateur: Fonctionnels

Cette section regroupe les fonctionnalités disponibles présentées par l'application à l'utilisateur.

Voici une structure générale d'interaction entre le joueur et l'application (voir Figure 1) :

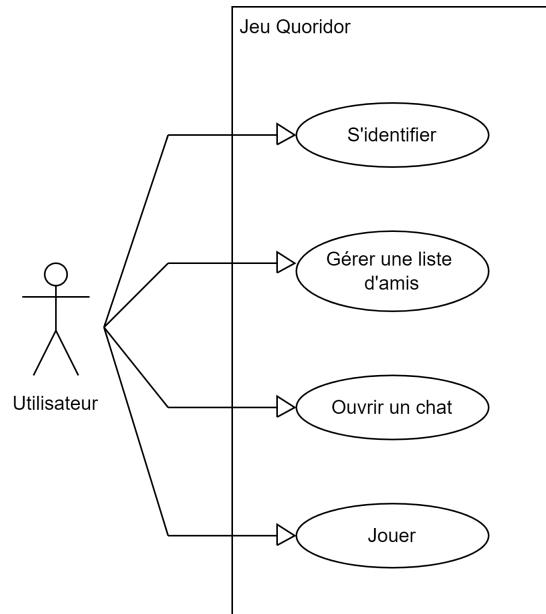


Figure 1: Diagramme use-case entre l'utilisateur et l'application

2.1 Liste d'amis

L'utilisateur a accès à une liste d'amis en permanence. Une fois qu'il est connecté à un compte, il peut voir les pseudonymes de ses amis dans sa liste d'amis (voir Figure 2).

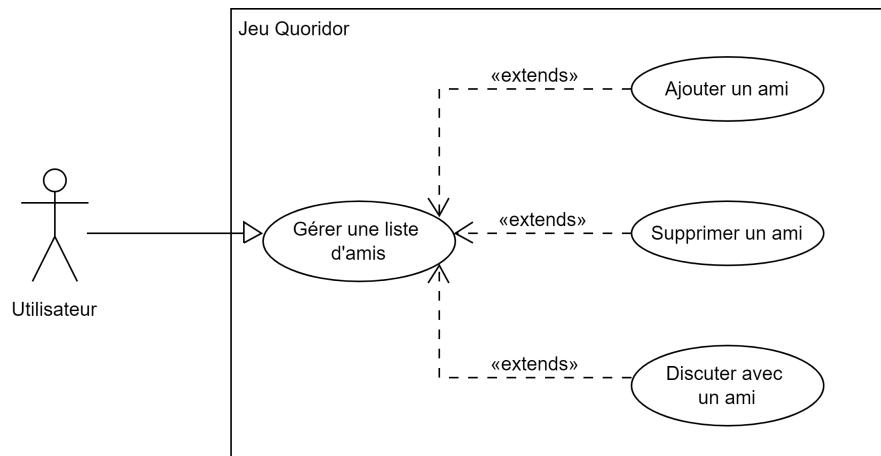


Figure 2: Diagramme use-case de l'interaction "Gérer une liste d'amis"

2.1.1 Ajouter un ami

Un utilisateur peut inviter un ami à sa liste d'amis en entrant son pseudonyme.

2.1.2 Accepter une demande d'ami

L'utilisateur qui a reçu une demande d'ami peut accepter la demande et ajouter le pseudonyme à sa liste d'amis ou la refuser et aucun pseudonyme ne sera ajouté dans la listes d'amis des utilisateurs.

2.1.3 Supprimer un ami

L'utilisateur A peut supprimer un utilisateur B de sa liste d'amis, ce qui supprime également l'utilisateur A de la liste d'amis de l'utilisateur B.

2.1.4 Discuter avec un ami

L'utilisateur peut ouvrir et fermer une fenêtre de chat avec un ami à tout moment et lui envoyer un message. Une fenêtre de chat ne s'ouvre chez l'ami que si l'utilisateur lui envoie un message.

2.1.5 Afficher un profil

Un utilisateur peut afficher le profil d'un de ses amis pour connaître le nombre de parties qu'il a jouées, gagnées, perdues, son nombre d'amis et son classement.

2.1.6 Barre de recherche

L'utilisateur peut avoir plusieurs amis et doit pouvoir retrouver un autre utilisateur le plus simplement possible à l'aide d'une barre de recherche.

2.1.7 Ajouter un ami depuis un salon

L'utilisateur doit pouvoir inviter un joueur qui n'est pas son ami mais qui est dans le même salon que lui dans sa liste d'amis.

2.2 Menu

L'utilisateur qui se connecte à son compte ou qui quitte un salon de jeu à accès au menu du jeu. Il perd accès au menu du jeu une fois qu'il crée/rejoint un salon ou se déconnecte de son compte (voir Figure 3).

2.2.1 Créer une partie

L'utilisateur peut créer un salon de jeu, depuis le menu, en lui donnant un nom et s'il le souhaite, un mot de passe.

2.2.2 Charger une partie

L'utilisateur peut créer un salon à partir d'une partie sauvegardée dans la base de données du serveur. Si un joueur de la partie rejoint le salon, il sera assigné à la place qu'il avait. Un joueur qui n'avait pas joué la partie avant ne peut pas rejoindre une partie chargée. L'utilisateur peut lancer la partie sans que tous les joueurs soient présents, et les joueurs manquants seront remplacés par des IAs.

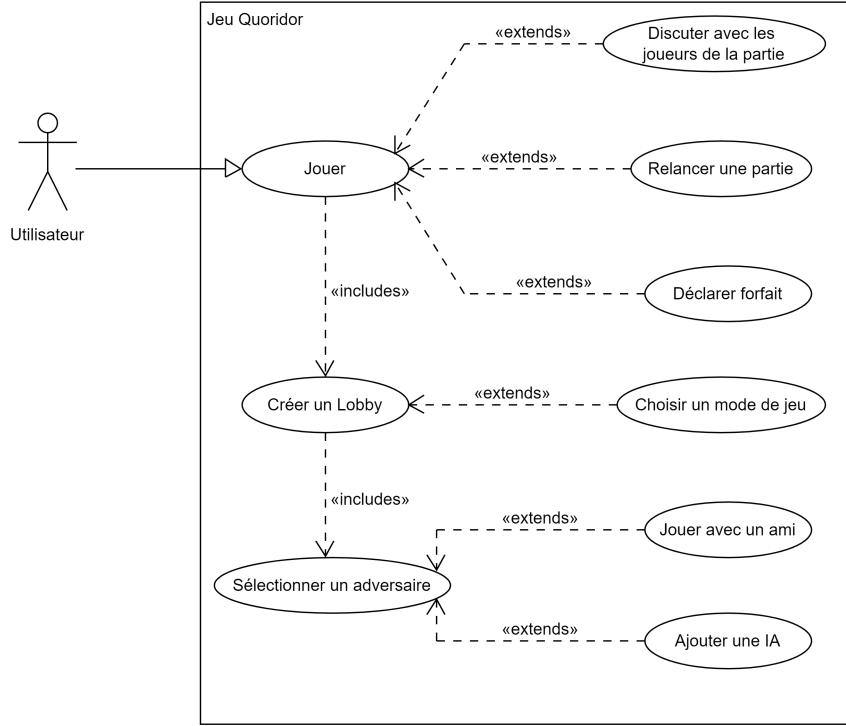


Figure 3: Diagramme Use-case de l'interaction "Jouer"

2.2.3 Rejoindre une partie

Un utilisateur peut rejoindre une partie depuis le menu via "Recherche Rapide" où il sera envoyé dans le premier salon libre. Il peut également faire "Voir Salons" afin de voir une liste des salons avec au moins une place libre.

2.2.4 Rejoindre un ami

Un utilisateur peut rejoindre une partie via "Rejoindre Ami" où il voit uniquement les salons créés par ses amis.

2.2.5 Profil

L'utilisateur peut consulter son profil depuis le menu, ainsi que se renommer, changer son mot de passe ou supprimer son compte.

2.2.6 Classement

L'utilisateur peut consulter un classement des comptes présent dans la base de données avec les plus haut scores de classement.

2.3 Dans un salon

Un chat de salon est automatiquement créé en même temps que le salon qui regroupe tous les joueurs du salon (voir Figure 4).

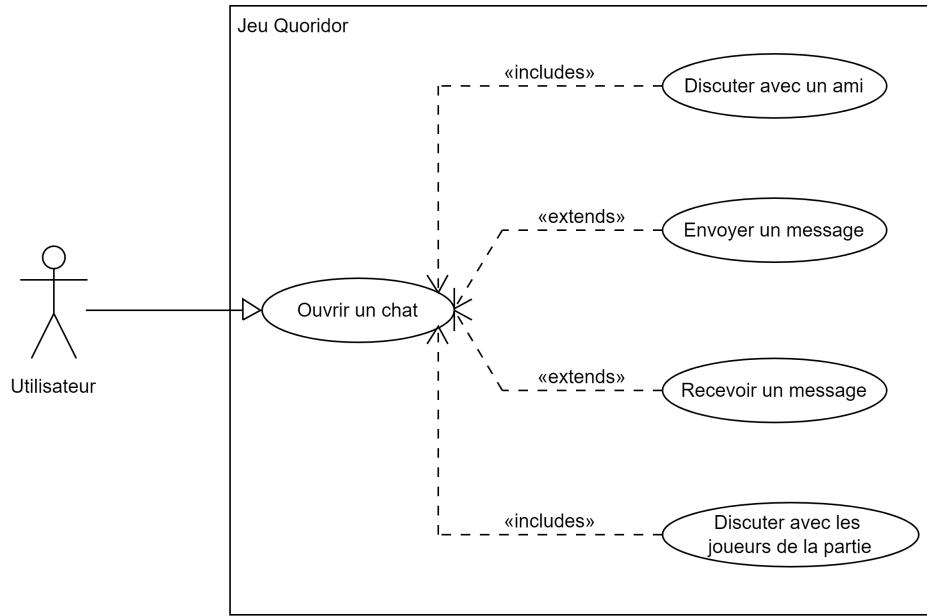


Figure 4: Diagramme use-case de l’interaction ”Ouvrir un chat”

2.4 En partie

2.4.1 Abandon d’une partie en cours

Un utilisateur peut déclarer forfait et être retiré de la partie. Il peut toujours voir comment la partie se déroule si elle n'est pas finie mais le serveur ne déclarera pas son tour.

2.4.2 Mettre une partie en pause et la sauvegarder

L'hôte d'une partie en cours peut décider de sauvegarder la partie dans la base de donnée du serveur et il pourra la charger plus tard depuis le menu.

3 Besoins utilisateur: Non fonctionnels

Cette section regroupe les fonctionnalités obligatoires à l'utilisateur pour assurer le bon fonctionnement de l'application.

3.1 Gestion du compte

Un utilisateur doit toujours créer un compte ou se connecter à un compte existant avant d'accéder au reste de l'application (voir Figure 5).

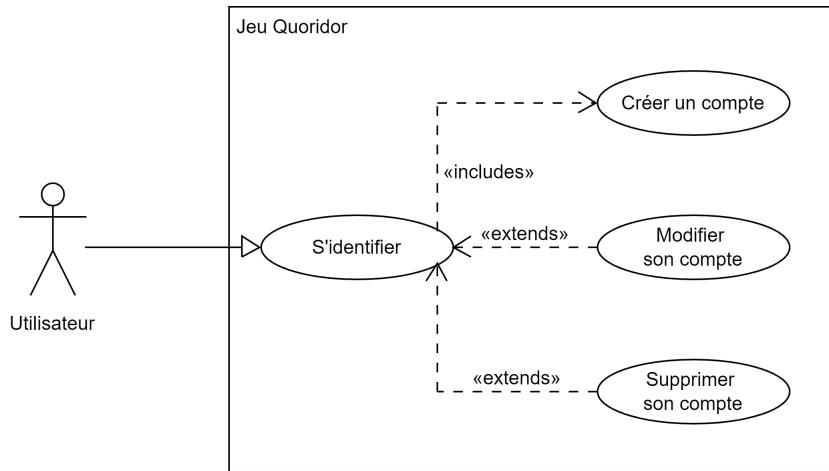


Figure 5: Diagramme use-case de la gestion de compte

3.1.1 Création de compte

Un utilisateur doit créer un compte en entrant un pseudonyme qui n'existe pas dans la base de données du serveur et un mot de passe associé au nouveau compte créé.

3.2 Mot de passe

Un mot de passe doit contenir entre 6 et 16 caractères avec au moins une majuscule, un chiffre et un caractère spécial.

3.2.1 Connexion

L'utilisateur doit se connecter à un compte en entrant un pseudonyme existant dans la base de données et le mot de passe qui lui est associé. Si le pseudonyme n'existe pas, il est proposé de créer un compte avec ce pseudonyme.

3.2.2 Suppression de compte

Si l'utilisateur souhaite supprimer son compte, il doit le faire dans la partie "Profil" du menu et entrer son mot de passe pour confirmer la suppression.

3.3 Configurer une partie

Le joueur doit pouvoir lancer une partie avec des règles qui lui conviennent.

3.3.1 Choisir un mode de jeu

Le joueur qui crée le salon doit choisir un mode de jeu, soit une partie de Quoridor normale ou bien un des modes de jeu présentés dans "Design des modes de jeu" (6.3).

3.3.2 Choisir le nombre de joueurs

Le joueur qui crée le salon doit aussi choisir le format de la partie (2 ou 4 joueurs).

3.3.3 Choisir les paramètres

Les paramètres sont les suivants:

1. Taille du plateau
2. Limite de temps par tour

3.4 En partie

Une partie fait jouer les joueurs tour à tour. Un utilisateur doit attendre son tour avant de déplacer son pion ou placer un mur valide sur le plateau (voir Figure 6).

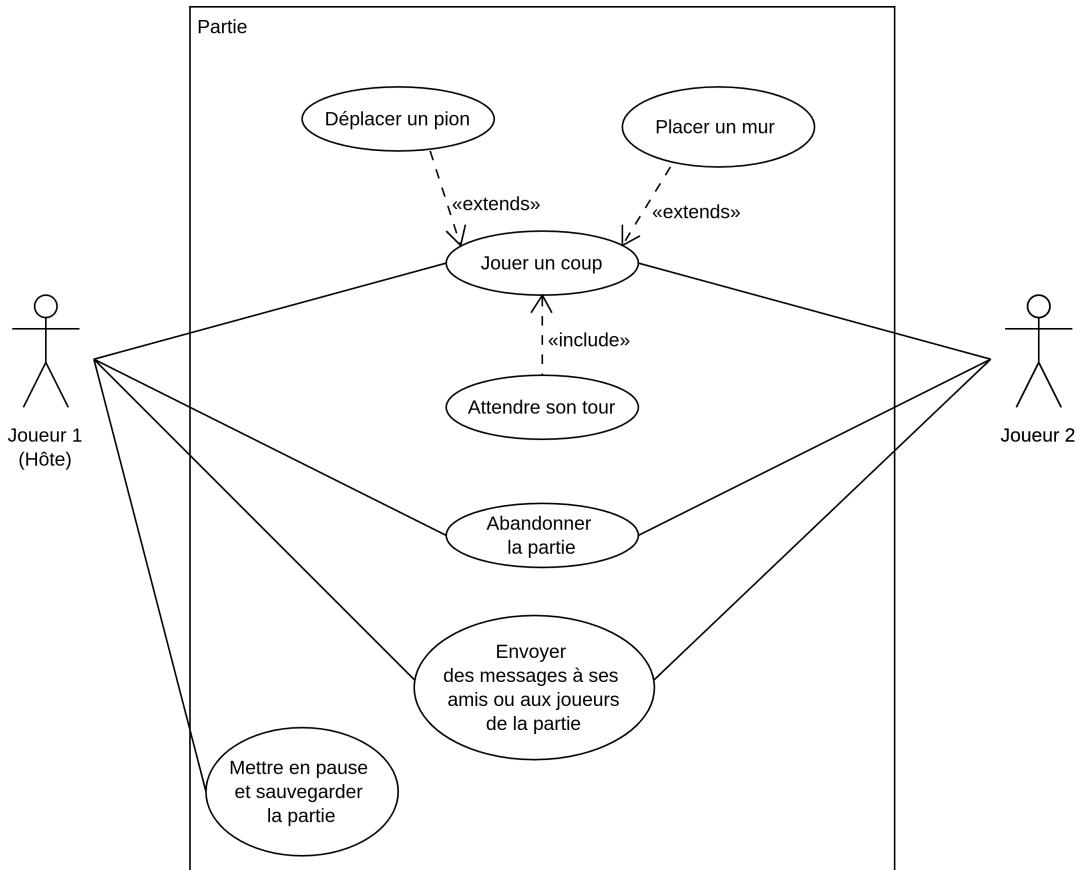


Figure 6: Diagramme use-case d'une partie entre 2 joueurs

3.4.1 Jouer en terminal

Un utilisateur doit entrer en premier le type de coup à effectuer puis la coordonnée sur laquelle effectuer son coup. Les types de coups sont :

- P : déplacer son pion
- H : placer un mur horizontal
- V : placer un mur vertical

Les coordonnées valides ne doivent pas sortir du plateau et doivent respecter un format lettre:nombre, par exemple : a1, f3, b10, etc.

Un déplacement du pion valide doit cibler une cellule adjacente à celle occupée par le pion et qui n'est pas bloquée par un mur.

Un mur horizontal bloque toujours le haut de la cellule correspondant à la coordonnée entrée et le haut de sa voisine de droite. Un mur vertical bloque toujours la droite de la cellule correspondant à la coordonnée entrée et la droite de sa voisine du dessus.(voir Figure 7)

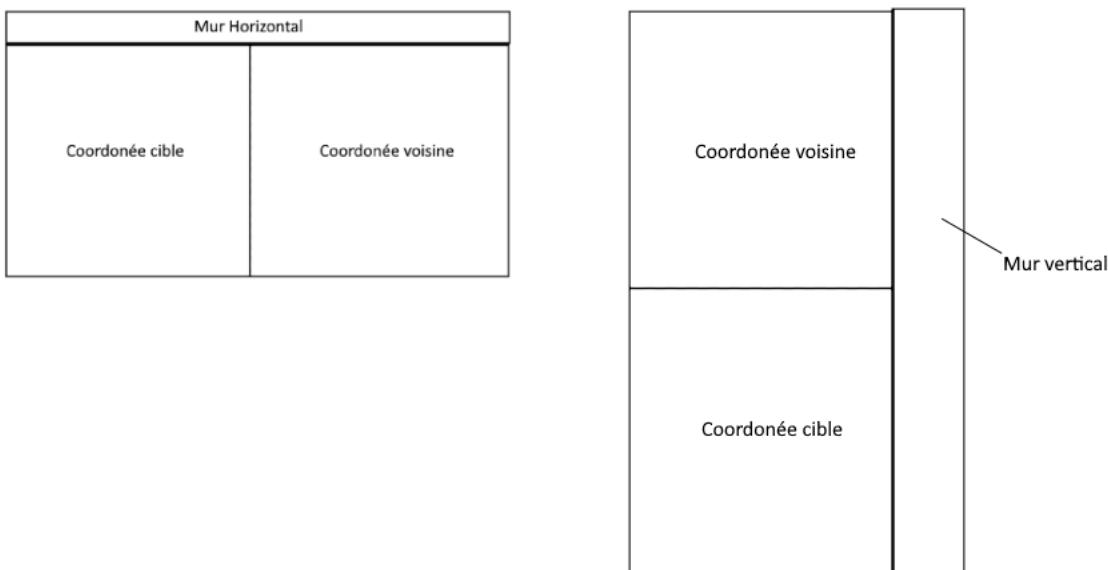


Figure 7: Schéma du placement d'un mur sur la coordonnée cible

Un mur valide doit donc être placé sur une cellule qui possède une voisine de droite s'il est horizontal ou une voisine du dessus s'il est vertical. De plus, tour mur ne peut être placé qu'entre deux paires de cellules. Il est donc impossible de poser un mur sur un des bords du plateau. Les cellules voisines doivent également ne pas avoir déjà été bloquées de la même manière.

Un coup valide peut donc ressembler à : H b4, V e7, P h12, etc.

3.4.2 Jouer en interface

Un utilisateur peut cliquer sur son pion puis sur une cellule valide de son choix (les cellules valides changent de couleur pour faciliter l'affichage des coups possibles du joueur). Il peut cliquer sur un bouton "placer mur" qui lui permet d'afficher les murs possibles lorsqu'il passe sa souris sur le plateau. Il peut cliquer sur un bouton "tourner mur" ou à l'aide d'un raccourci afin de changer l'axe du mur.

3.5 Fin de partie

Les joueurs reviennent toujours dans le salon de la partie une fois qu'elle est terminée. Il faut créer un nouveau salon si les paramètres ou le mode de jeu souhaitent être changés.

4 Besoin système: Fonctionnels

Cette section explique le fonctionnement des implémentations nécessaires pour subvenir aux besoins de l'utilisateur.

4.1 Gestion des comptes

Afin de gérer les comptes, l'application fait appel à une base de données sous la forme d'une bibliothèque de fichiers .txt où seront rangées toutes les informations stockées de l'application (comptes, profils, parties sauvegardées). L'application utilise une classe Database pour lire et écrire sur les fichiers de la base de données afin de former le pont entre les processus de l'application et la base de données. (voir Figure 8)

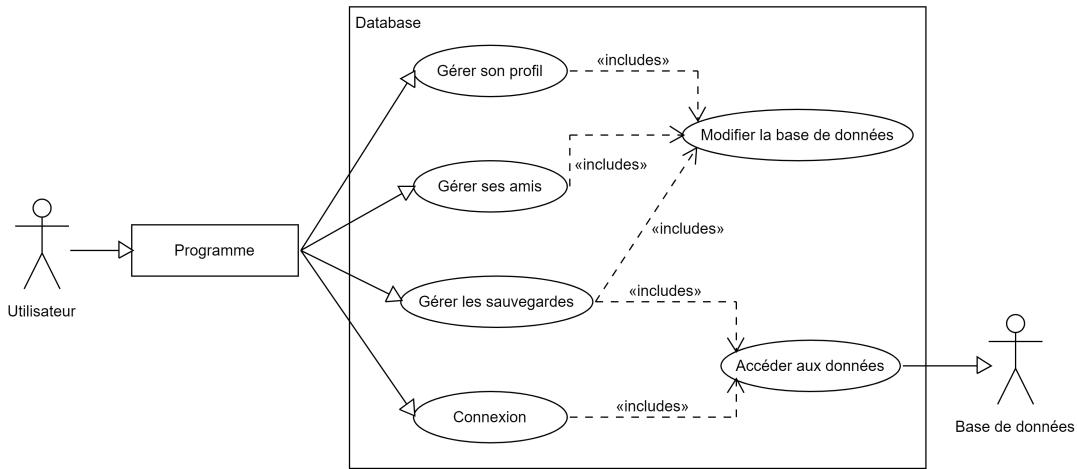


Figure 8: Diagramme use-case de la classe Database sur la base de données en fonctions des actions utilisateurs

4.1.1 Création d'un compte

L'application exécute une lecture des fichiers de la base de données afin de vérifier l'existence du pseudonyme donné. Elle crée le compte si le pseudo donné n'existe pas.

4.1.2 Suppression d'un compte

L'application supprime le compte de l'utilisateur de la base de données.

4.2 Mot de passe

Un mot de passe doit contenir entre 6 et 16 caractères avec au moins une majuscule, un chiffre et un caractère spécial.

4.2.1 Connexion

L'utilisateur doit se connecter à un compte en entrant un pseudonyme existant dans la base de données et le mot de passe qui lui est associé. (voir Figure 9)

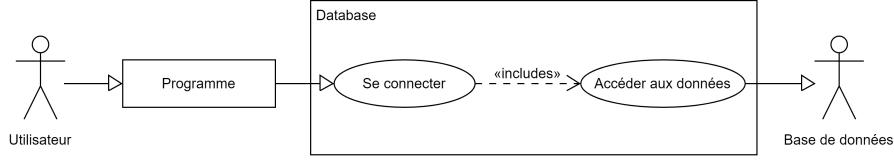


Figure 9: Diagramme use-case de l'interaction "se connecter" de la classe Database

4.3 Gestion des profils

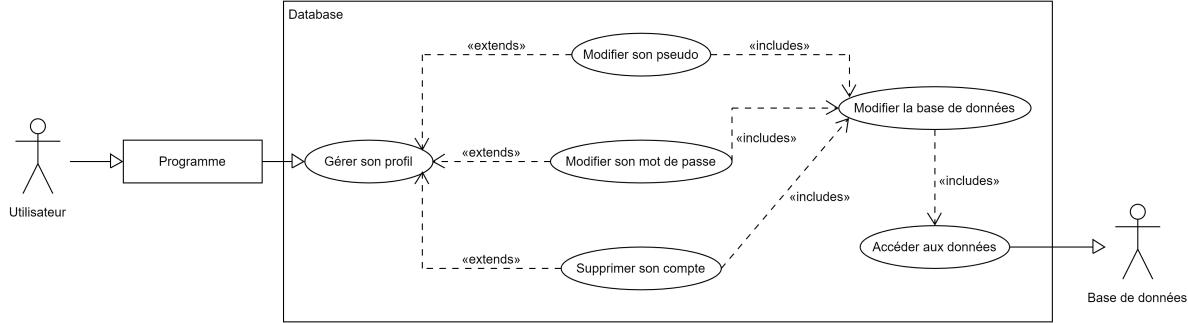


Figure 10: Diagramme use-case de la classe Database sur la base de données en fonction des actions de l'utilisateur

4.3.1 Affichage d'un profil

L'application accède aux données d'un utilisateur dans la base de données afin d'afficher son profil (voir Figure 11).

```

Nom: Pseudo
Parties jouées: Nombre total (Victoires, Défaites)
Nombre d'amis: Nombre
Classement: nème
    
```

Figure 11: Exemple d'affichage d'un Profil en Terminal

4.3.2 Sauvegarder les statistiques du joueur

Après chaque partie, l'application accède au profil de l'utilisateur situé dans la base de données afin de modifier ses statistiques.

4.4 Chat

Le chat permet une communication entre deux utilisateurs via le serveur selon le diagramme de la Figure 12.

4.5 Gestion d'une partie

Cette section explique en détails le fonctionnement d'une partie au sein du système. Voici un diagramme de classe démontrant les éléments en jeu lors d'une partie (voir Figure 13).

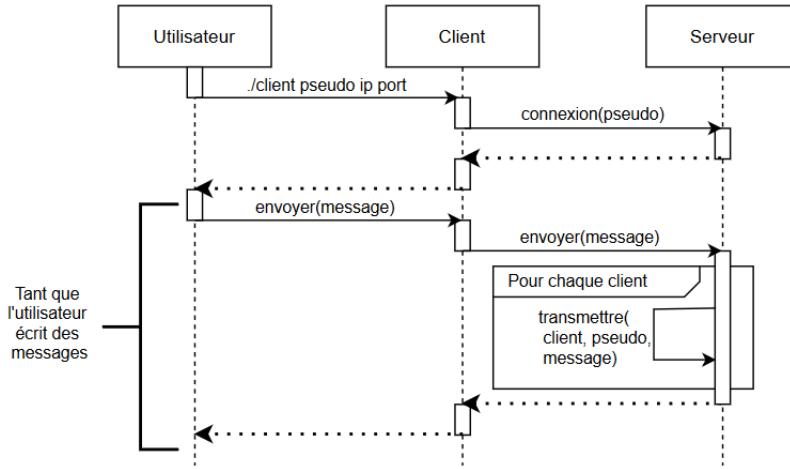


Figure 12: Diagramme de séquence de Chat Client-Serveur
(Ce diagramme provient de l'énoncé du projet 2 du cours de système d'exploitation)

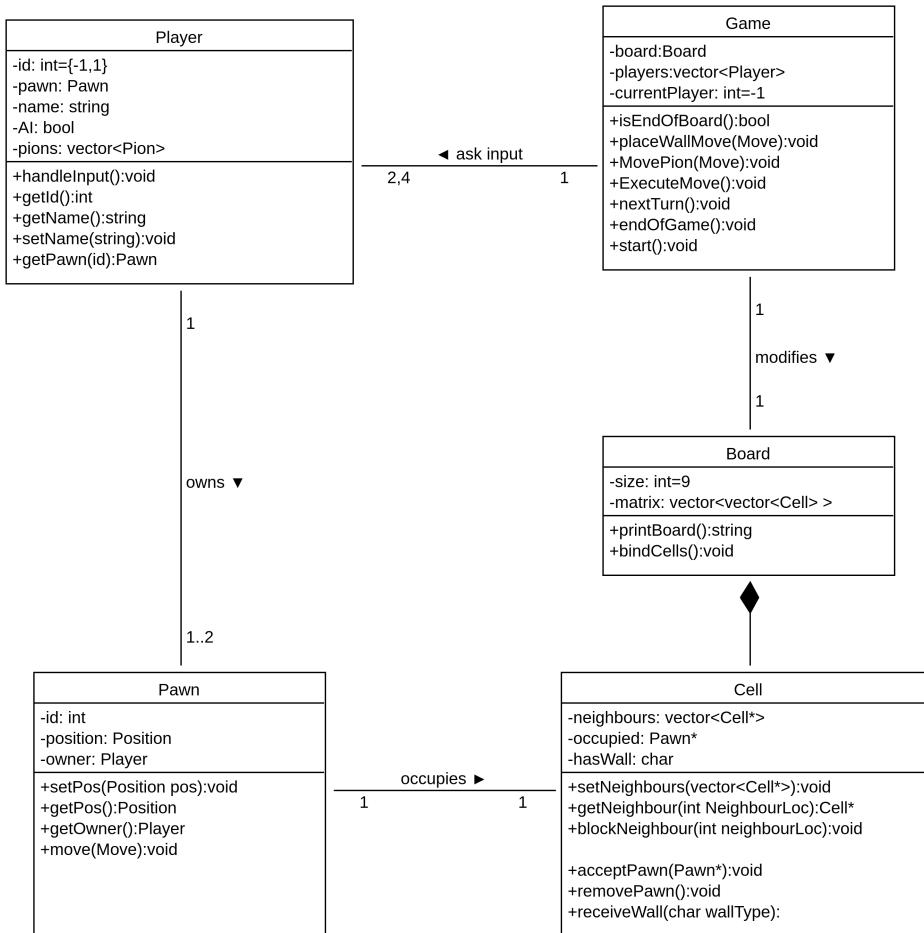


Figure 13: Diagramme de classe du jeu Quoridor

4.5.1 Gestion du plateau

Le plateau est représenté par une matrice n x n (n=9 de base). Le plateau ne peut être modifié que par l'action d'un joueur qui donne un coup valide. Chaque position dans la matrice correspond à une case du plateau et chaque case est représentée par une cellule (MotherCell) qui a comme attributs des pointeurs vers ses voisins directs.

4.5.2 Gestion des pions

Un pion possède un Id correspondant au joueur auquel il appartient et une position représentant sa position sur le plateau. Un pion ne peut que se déplacer verticalement et horizontalement.

4.5.3 Gestion des murs

Lorsqu'un joueur place un mur sur une WallCell entre deux PawnCells, ces dernières se suppriment l'une l'autre de leur liste de voisines respectives (voir Figure 14). Avant chaque mur placé, on vérifie que tous les joueurs ont encore un chemin possible vers l'autre bout du plateau. Si oui, c'est au prochain joueur de choisir une action. Si non, un message s'affiche pour annoncer au joueur que son coup est invalide et une nouvelle action est demandée.

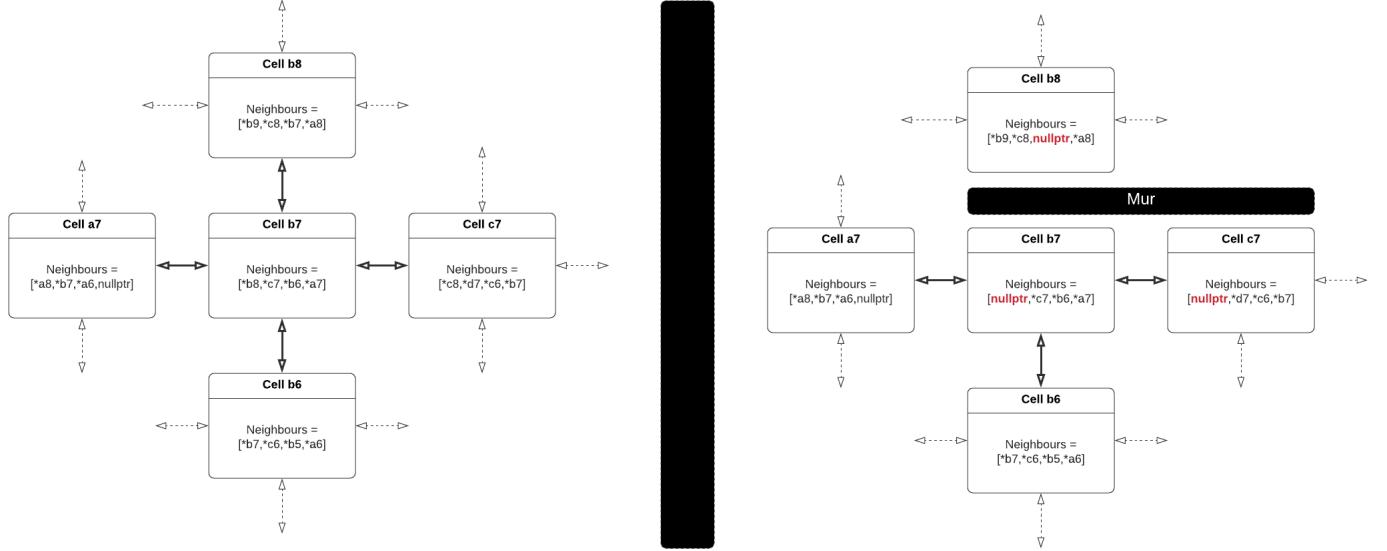


Figure 14: Schéma des voisins d'une cellule (b7) avec et sans mur

4.5.4 Gestion des tours

Chaque tour, le joueur a le choix entre placer un mur ou déplacer son pion dans la direction valide de son choix. Les tours sont gérés par la classe Game. Tant que le coup n'est pas valide, le processus continue de demander une nouvelle action (voir Figure 15). Rappel des actions illégales:

1. Choisir un déplacement entre deux cases non jointes
2. Poser un mur sur un autre mur
3. Poser un mur qui enfermerait un pion et/ou l'empêcherait d'atteindre l'autre bout du plateau
4. Sortir son pion du plateau ou placer un mur hors du plateau

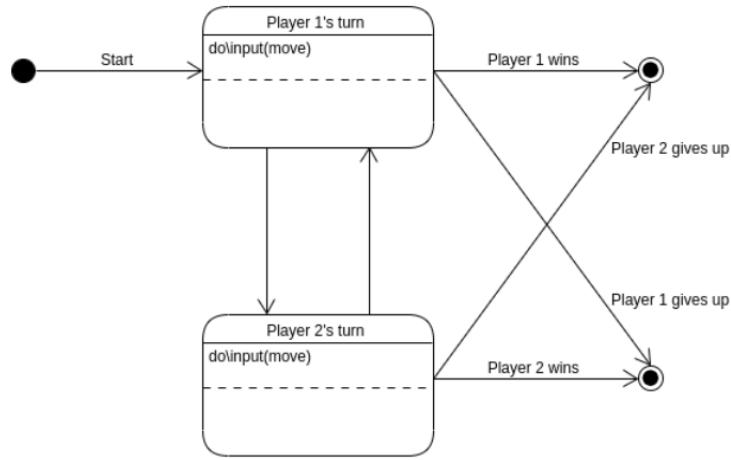


Figure 15: Diagramme d'état de la classe Game faisant jouer 2 joueurs

4.6 Rejoindre une partie

Soit un joueur déjà présent dans le salon invite un autre joueur via son pseudo, soit un joueur extérieur demande à rejoindre le salon via l'identifiant de cette dernière.

4.6.1 Matchmaking rapide: trouver un salon ouvert

Trouver un salon où un joueur est explicitement demandé et où il reste de la place.

4.6.2 Rejoindre un ami

Demandeur à un joueur dans un salon pour rejoindre sa partie même sans connaître l'identifiant du salon. (voir Figure 16)

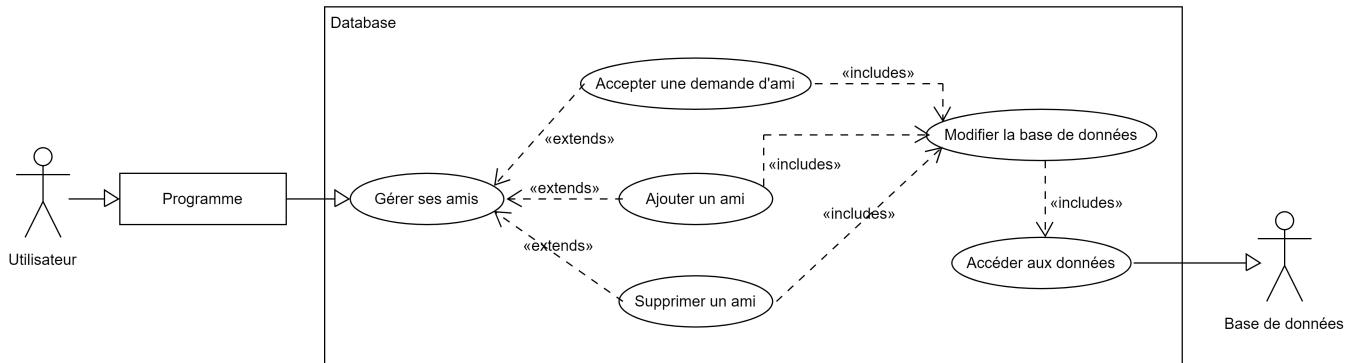


Figure 16: Diagramme use-case de l'interaction "gérer ses amis"

4.7 Sauvegarde

Il y a la possibilité de sauvegarder une partie en cours et de la reprendre à tout moment. (voir Figure 17)

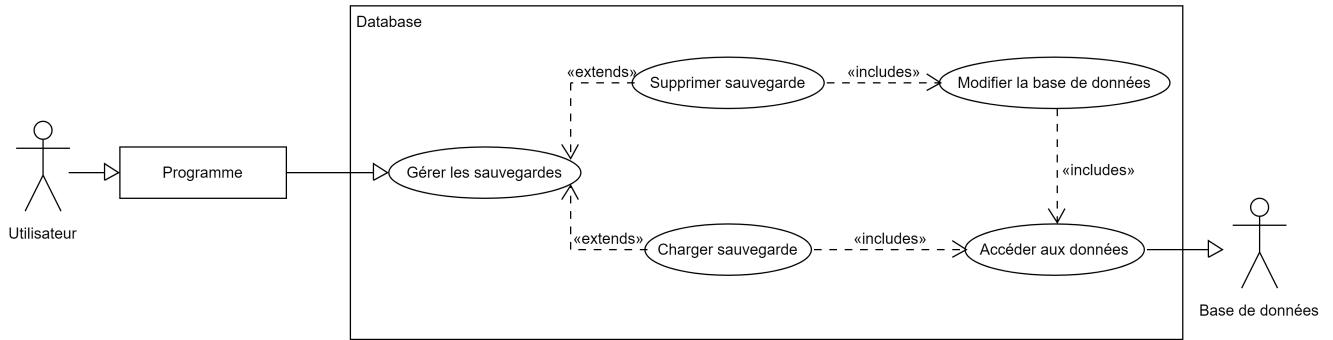


Figure 17: Diagramme use-case de la sauvegarde

4.8 Classement des joueurs

Les joueurs pourront à la fin de chaque partie gagner ou perdre un score de classement. La classe Database traque les meilleurs scores de la base de données pour les trier dans l'ordre croissant et les présenter dans la section "Classement" du Menu.

5 Besoins système: Non fonctionnels

5.1 OS

L'application doit être lancée sur une machine GNU/Linux.

5.2 Réseau

L'application requiert une adresse Ipv4/Ipv6 valide ainsi qu'un port valide par lequel les joueurs se connecteront.

5.3 Disponibilité

L'application est gratuite.

5.4 Performances

Les performances requises par l'application sont encore à déterminer au cours des prochaines phases du projet, mais ne devraient pas avoir de lourds requis pour fonctionner.

5.5 Capacité

La capacité requise par l'application est encore à déterminer au cours des prochaines phases du projet.

5.6 Sécurité

5.6.1 Base de données

La base de données du serveur sera crypté selon un chiffrement RSA. Vous pouvez en apprendre plus sur le chiffrement RSA [ici](#).

5.6.2 Mot de passe

Lorsque l'utilisateur crée ou modifie son mot de passe, il doit l'écrire une première fois puis le réécrire pour le confirmer.

5.7 Robustesse

La robustesse de l'application désignera la capacité du système à ne pas planter, perdre, ou corrompre ses données. Elle sera à déterminer au cours des prochaines phases du projet.

6 Design et fonctionnement du Système

6.1 Design du système

Le système sera séparé en 2 sous-applications : une application principale serveur, qui s'occupe de toutes les fonctionnalités expliquées plus haut, et une application client, qui s'occupe de proposer ces fonctionnalités à l'utilisateur, d'analyser et transmettre les actions de l'utilisateur vers le serveur afin d'assurer l'interaction entre utilisateurs et serveur. La Figure 18 est une ébauche conceptuelle de l'organisation des composants du système.

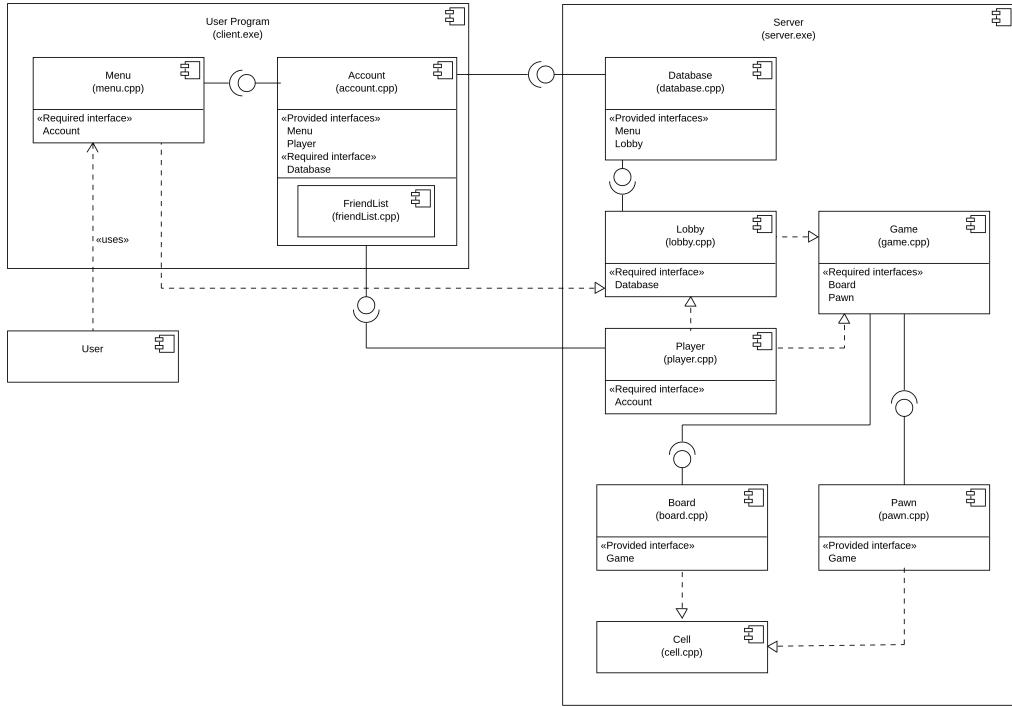


Figure 18: Component Diagram des fichiers système

6.2 Fonctionnement du système

6.2.1 Inscription

Le système gère une inscription de la façon suivante (voir Figure 19):

- Il demande et lit un pseudonyme au près de l'utilisateur.
- Il vérifie, dans la base de données, si un dossier existe avec le pseudonyme donné, s'il en trouve un, retour à l'étape 1 en demandant à l'utilisateur un pseudonyme différent, car celui donné existe déjà.
- Il demande et lit un mot de passe auprès de l'utilisateur. Le mot de passe est demandé une 2ème fois en confirmation.
- Il vérifie que les 2 mots de passe écrits correspondent. S'ils ne correspondent pas, retour à l'étape 3 en demandant à l'utilisateur de vérifier ses mots de passe.
- Le système crée un dossier de données lié au pseudonyme donné et y inclut son mot de passe crypté. L'inscription est complétée et l'utilisateur peut accéder à l'application avec son compte.

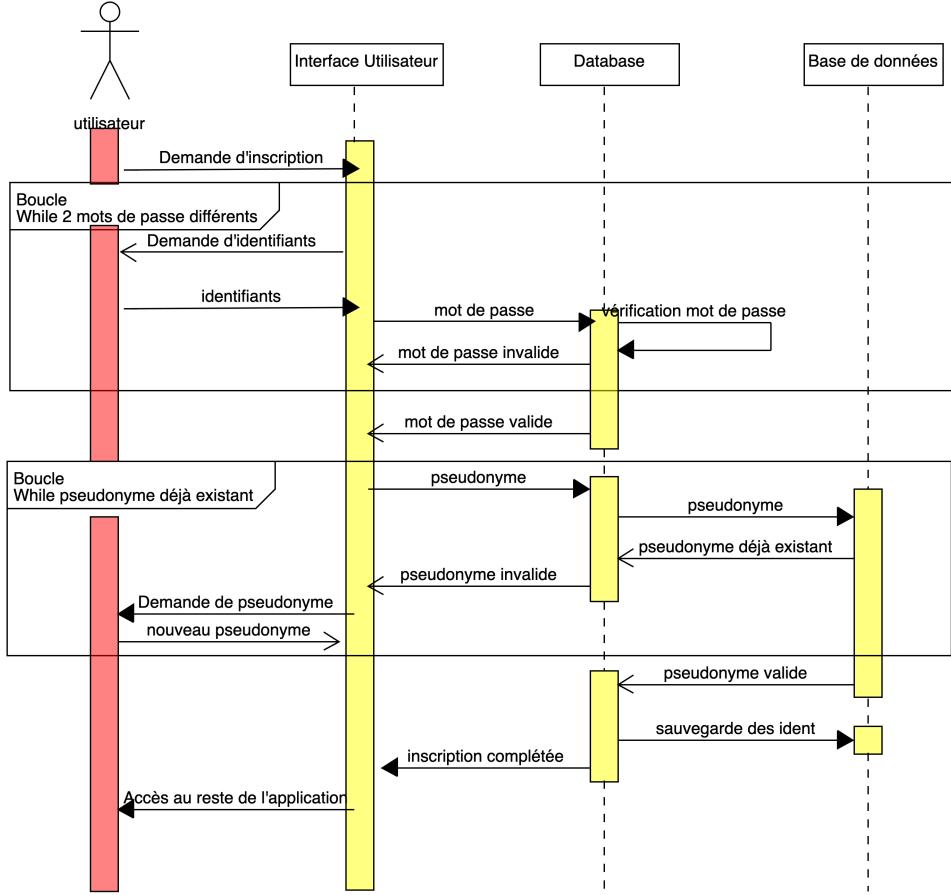


Figure 19: Diagramme de séquence d'une inscription

6.2.2 Connexion

Le système gère une connexion de la façon suivante (voir Figure 20):

- Il vérifie que le pseudonyme donné existe dans la base de données. S'il ne l'est pas, le système demande à l'utilisateur s'il veut créer un compte associé au pseudonyme donné.
- Il vérifie que le mot de passe entré correspond au pseudonyme donné. S'il ne l'est pas, le système demande à l'utilisateur de vérifier son mot de passe.
- Le système donne accès à l'utilisateur au reste de l'application avec son compte.

6.3 Design des modes de jeux

Les modes de jeux sont implémentés selon le principe d'héritage. La classe Board est réimplémentée, ainsi que d'éventuelles autres classes, en sous-classes qui représentent les différents modes de jeu. Voici une liste des modes de jeux proposés:

6.3.1 DestruQtion

Chaque joueur a plus de murs, et les murs peuvent servir à détruire ceux déjà existants.

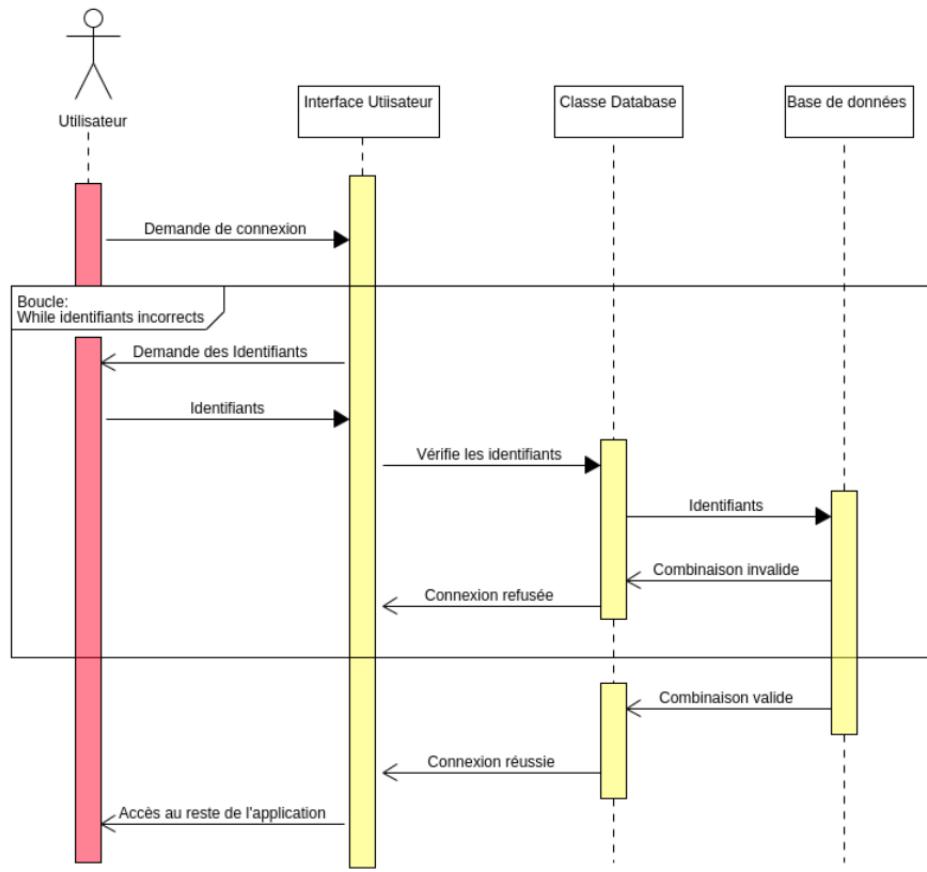


Figure 20: Diagramme de séquence d'une connexion

6.3.2 QQQuoridor

Chaque joueur a 2 fois plus de murs et joue 2 coups durant son tour. Le plateau est aussi plus grand.

6.3.3 Blitz

Chaque joueur a un temps très limité pour jouer son coup.

6.3.4 Double Up

Mode de jeu 2v2, les deux joueurs d'une même équipe commencent du même côté du plateau et doivent tous deux arriver de l'autre côté pour gagner.

6.4 Design de l'IA

Nous allons implémenter une intelligence artificielle qui va suivre le principe de l'algorithme récursif minimax. Selon l'état d'un plateau, on trouve le meilleur état en simulant tous les scénarios possibles et en choisissant le meilleur scénario. Le meilleur scénario est celui avec la meilleure situation finale en assumant que :

- on fait toujours le meilleur mouvement pour maximiser son score et
- l'adversaire fait toujours le mouvement qui est le meilleur pour lui (et donc le pire pour nous), minimisant notre score.

Ce principe de maximisation et de minimisation est la raison pour laquelle l'algorithme se nomme Minimax. Le score ici représentera la différence entre le nombre de cases séparant l'IA de la victoire et le nombre de cases séparant son adversaire de la victoire.

7 Implémentation des différentes parties

7.1 Jeu Quoridor

7.1.1 Version classique

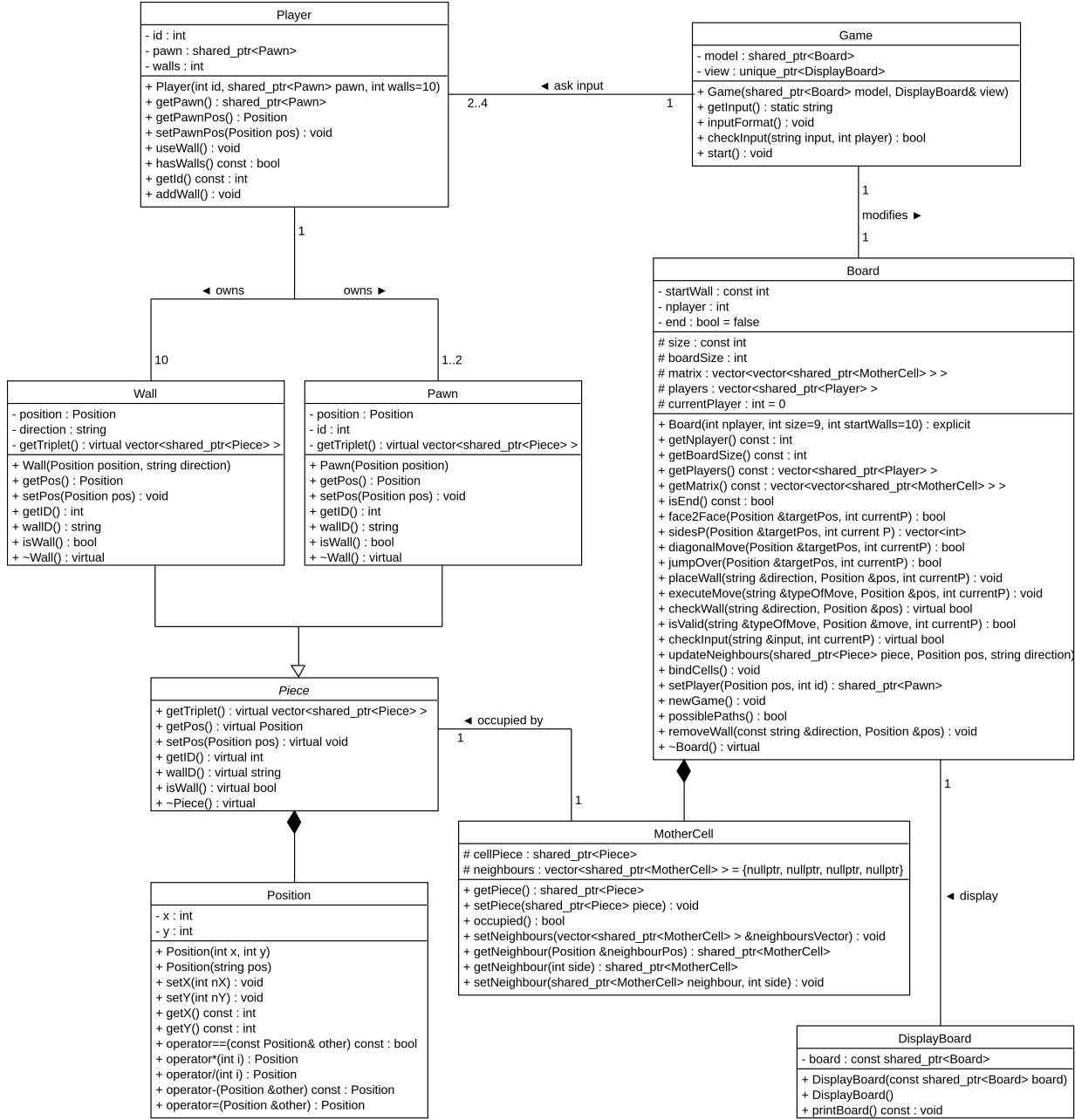


Figure 21: Diagramme de classe du jeu Quoridor

La classe Game est le contrôleur qui interagit avec les joueurs (Player) et le plateau (Board). Le plateau est affiché sur le terminal par la classe DisplayBoard.

Chaque joueur possède un identifiant, un pion (Pawn) et un nombre de murs égal à 10 au départ. Il a la possibilité de déplacer son pion ou de placer un mur.

Le plateau est composé de cellules (MotherCell) qui peuvent être occupées par une pièce (Piece), c'est à dire un mur (Wall) ou un pion. La classe Board gère tous les coups.

La classe Piece est une classe abstraite dont héritent les classes Wall et Pawn. Une pièce est définie par une position. Un mur a une direction comme attribut supplémentaire et un pion a un id qui référence à quel joueur il appartient.

La fonction start de Game imprime une première fois le format de coup accepté et le plateau. Tant que la partie n'est pas finie :

- le joueur dont c'est le tour reçoit un message.
- la fonction checkInput est appelée avec le coup que le joueur a entré. Si elle retourne true, c'est le tour du joueur suivant, si ce n'est pas le cas, le format de coup accepté est affiché.
- le plateau est de nouveau affiché avec les modifications du coup du joueur ou inchangé si le coup n'était pas valide.

La fonction checkInput de Game fait appel à la fonction checkInput de Board. Cette fonction :

- vérifie que la taille du coup et retourne false, si la taille ne correspond pas;
- récupère le type de coup et les coordonnées du coup;
- si le coup est valide, elle exécute le coup;
- si le coup était un placement de mur, elle vérifie qu'il y ait toujours un chemin possible jusqu'à l'autre bout du plateau;
- si ce n'est pas le cas, le mur est retiré et elle retourne false.
- si tout s'est bien passé, elle retourne true.

7.1.2 Modes de jeu

Les modes de jeu sont construits sur base d'héritage (voir Figure 22). Pour le moment, seuls les modes de jeu DestruQtion et QQQuoridor sont fonctionnels. Les méthodes virtuelles de Board, Wall et Position permettent le traitement spécifique que requièrent ces modes de jeu.

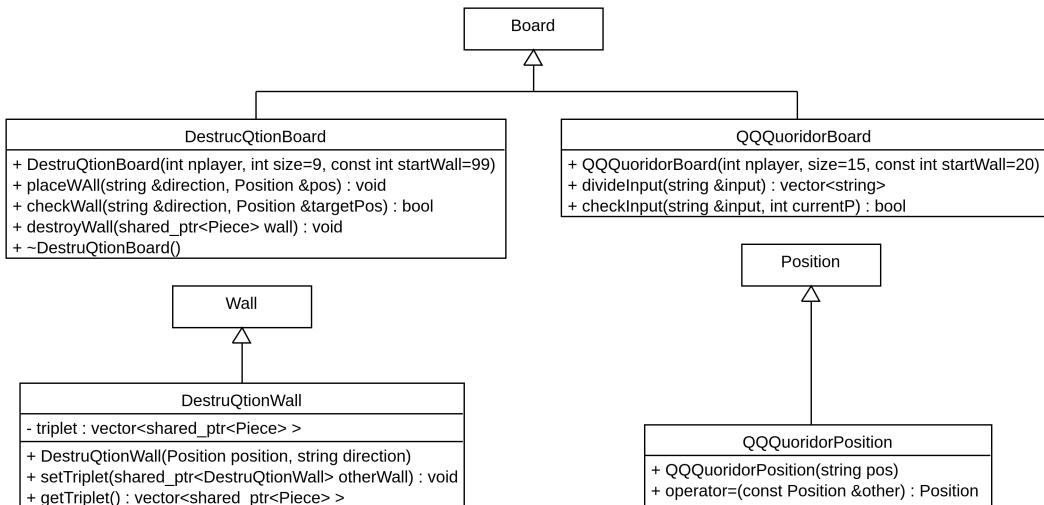


Figure 22: Diagramme de classe des modes de jeu

7.2 Base de données

La base de données est un ensemble de fichiers texte (.txt). Le nom d'un fichier est le pseudonyme qui lui est associé. Chaque fichier représente un compte qui contient:

- le mot de passe crypté associé au compte
- un plateau vide ou celui de la dernière partie sauvegardée
- le nombre de parties gagnées
- le nombre de parties perdues
- la liste des amis à ajouter
- la liste d'amis

Le classement des joueurs n'est pas encore opérationnel. La classe DatabaseHandler (Figure 23) permet de manipuler les données. Le programme doit être lancé en local pour garantir la synchronisation des données (exemple : amis).



Figure 23: Diagramme de classe du gestionnaire de la base de données

7.3 Serveur

Lorsque le serveur (Server) est lancé, il attend de nouvelles connections, c'est à dire un message envoyé au masterSocket. Si un message reçu provient d'un client (Client), le serveur enregistre ce client comme un utilisateur (userT). Quand assez d'utilisateurs pour commencer une partie sont connectés, le serveur alerte tous les utilisateurs et annonce le premier joueur. Le serveur interprète un message comme un coup seulement s'il commence par "/", autrement il sera considéré comme un message envoyé aux autres utilisateurs. Un

message vide est interprété comme une demande de déconnexion. Si le joueur actuel joue un coup légal, le joueur suivant est annoncé. Lorsque un joueur gagne, le serveur arrête d'interpréter les coups.

Lorsqu'un utilisateur lance un client, une fenêtre ncurses s'initialise et se divise en trois objets WINDOW (boardWindow en haut à gauche, chatWindow en haut à droite et inputWindow en bas). Plusieurs choix sont affichés : "Login and Play" et "Choose Gamemodes". Un utilisateur ne peut pas jouer tant qu'il ne s'est pas connecté.

- Login : cette action fait appel à la fonction loginRoutine. Cette fonction demande à l'utilisateur son pseudo et son mot de passe. Il vérifie dans la base de données que le compte existe et que le mot de passe est correct. Si le pseudo n'existe pas, il se voit proposer de créer un compte avec ce pseudo. L'utilisateur a la possibilité d'ajouter des amis à cette étape. Une fois que l'utilisateur est connecté, l'action play est lancée.
- Play : cette action fait appel à la fonction runGame. runGame lance le multi-threading et connecte le client au serveur. La partie se déroule comme décrit plus haut.
- Gamemodes : Pour le choix du mode de jeu, il faut modifier le type de plateau construit lors de l'initialisation de l'objet "Server". Ce plateau est initialisé à partir d'un plateau spécifique à un mode de jeu parmi 3 (Board, DestruQtion, QQQuoridor).

La Figure 24 représente l'implémentation du serveur. Le mode de jeu est choisi lors du lancement de la partie. Pour le moment, le serveur ne peut gérer qu'une seule partie à la fois. Avant le lancement du client, il faut veiller à avoir le plus grand terminal possible pour qu'il n'y ait pas problème d'affichage.

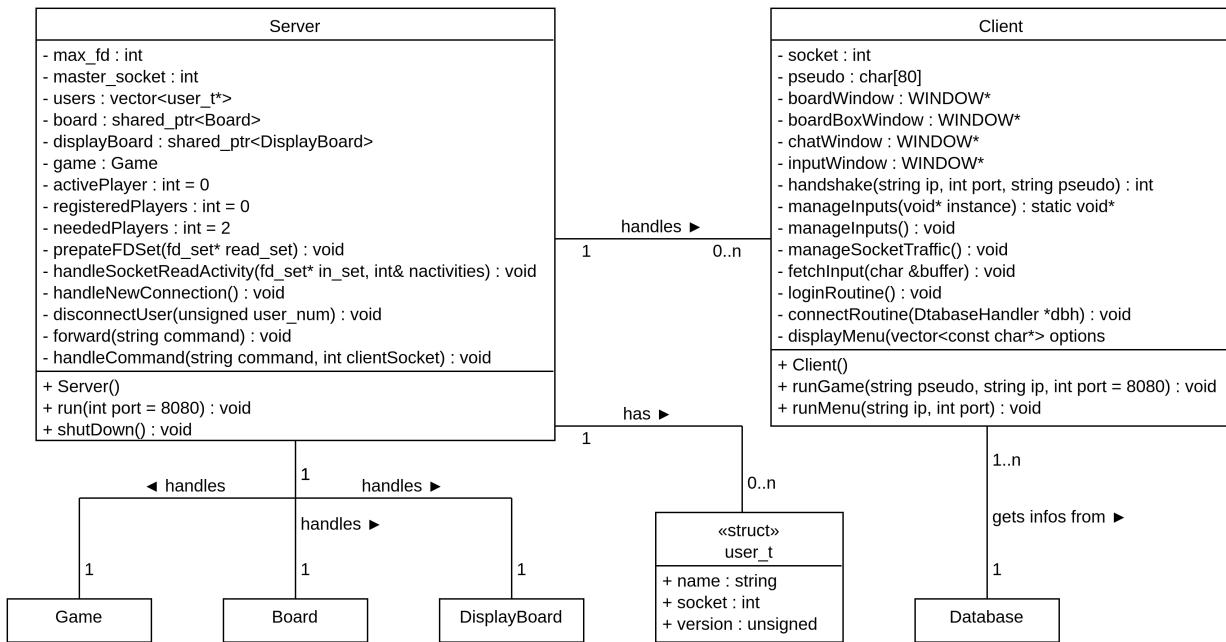


Figure 24: Diagramme de classe du serveur

8 Implémentations manquantes

- Le choix du mode de jeu depuis le menu affiché par client.cpp s'affiche, mais ne change pas le mode de jeu qui sera joué. La modification du mode de jeu doit se faire par l'intérieur du code, ou il suffit de remplacer la création d'un Board par un DestruQtionBoard ou un QQQuoridorBoard.

- Le programme manque la capacité de rejouer des parties, a chaque fin de partie, le client doit relancer le serveur.
- Pas de classement, ni de score de joueur (elo).
- La friendlist est présente, mais ne sert pas d'utilité.

9 Historique

Version	Ajout(s)	Auteur(s)	Date
version 0.0	”Table des matières”	Groupe	24/11/2021
version 0.1	”Besoin utilisateur: Fonctionnels” + ”Glossaire”	Merian Emile	29/11/2021
version 0.2	”Besoins utilisateur: Non Fonctionnels”	Abraham Théo	01/12/2021
version 0.2.1	”Besoins systèmes: Fonctionnels”	Abraham Théo	01/12/2021
version 0.2.2	Mise en page de l'historique	Abraham Théo	02/12/2021
version 0.3	”Amélioration ”Besoins système: Fonctionnels”	Merian Emile, Dimitrov Mark	02/12/2021
version 0.3.1	”Glossaire”	Merian Emile	02/12/2021
version 0.4	Correction de l'orthographe et de la syntaxe	Leduc Camille	02/12/2021
version 0.5	”Component Diagram des fichiers système” + ”Exemple d'affichage d'un Profil en Terminal”	Leduc Camille	04/12/2021
version 0.5.1	”Diagramme Use-Case d'une partie entre 2 joueurs”	Leduc Camille	04/12/2021
version 0.5.2	”Diagramme Use-Case de la classe Database sur la base de données en fonctions des actions utilisateurs”	Dimitrov Mark	05/12/2021
version 0.5.3	”Design du système”	Merian Emile	05/12/2021
version 0.5.4	”Diagramme Voisins avec et sans mur”	Dimitrov Mark	05/12/2021
version 0.6	”Design de l'IA”	Leduc Camille	06/12/2021
version 0.7	”Diagramme d'état de la classe Game”	Chica Cobo Sofia	06/12/2021
version 0.8	”Pseudocodes”	Leduc Camille, Dimitrov Mark	07/12/2021
version 0.9	”Fonctionnement système: Inscription, Connexion”	Merian Emile	08/12/2021
version 0.9.1	Retrait des ”Pseudocodes”	Dimitrov Mark	10/12/2021
version 0.10	”Diagramme de séquence d'une connexion”	Chica Cobo Sofia	12/12/2021
version 0.10.1	”Diagramme de séquence d'une inscription”	Dimitrov Mark	12/12/2021
version 1.0	”Update des pseudocodes et première version des headers”	Horii Hisao, Bollengier Thomas	12/12/2021
version 1.1	”Division du use-case général” + ”Ajouts au glossaire”	Leduc Camille	12/02/2022
version 1.1.1	”Division du use-case de Database”	Leduc Camille	15/02/2022
version 1.2	”Diagramme de classe basé sur l'implémentation actuelle”	Leduc Camille	17/02/2022
version 1.3	”Adaptation des parties 'Connexion' et 'Inscription' suivant l'implémentation”	Leduc Camille	23/02/2022
version 1.4	”class diagram Quoridor” + ”correction du texte pour coller à l'implémentation”	Leduc Camille	02/03/2022
version 2.0	”Implémentation des parties” + ”diagrammes de classe associés”	Leduc Camille	06/03/2002