Quoridor Software Requirement Document

INFO-F209 - Projet d'informatique 2

Anton Romanova, Mohammad Secundar, Esteban Aguililla Klein,
Vlad Moruntale, Mathieu Van Den Bremt, Nabil Abdellaoui,
Ayman Boulaich, Noé Bourgeois

Avril 2022

Table des matières

1 Introduction **3**
1.1 But . . . 3
1.2 Glossaire . . . 3
1.3 Historique du document . . . 3

2 Besoin d'utilisateur **6**
2.1 Besoins/Exigences fonctionnelles . . . 6
2.1.1 Menu principal . . . 6
2.1.2 Durant une partie . . . 8
2.1.3 Commande . . . 12
2.1.4 Mode de jeu . . . 13
2.1.5 Messagerie . . . 13
2.2 Exigences non-fonctionnelles . . . 14
2.2.1 Inscription et connexion . . . 14

3 Besoins du système **14**
3.1 Besoins fonctionnels . . . 14
3.1.1 Enregistrement . . . 14
3.1.2 Connexion . . . 16
3.1.3 Rejoindre une partie . . . 17
3.1.4 Gestion de partie . . . 17
3.1.5 Actualisation du classement . . . 17
3.1.6 Base de données . . . 17
3.1.7 Logging . . . 18
3.1.8 Latence . . . 18
3.1.9 API . . . 19
3.2 Besoins non-fonctionnels . . . 20
3.2.1 Portabilité . . . 20
3.2.2 Réseau . . . 20
3.2.3 Concurrence . . . 20
3.2.4 Sécurité . . . 20
3.2.5 Légalité . . . 20

4 Design et fonctionnement du système **20**

5 Annexes **20**

A Règles de base de Quoridor **20**

B Librairies **21**

C Commentaires sur les diagrammes **21**

D Commande sous forme d'entrée clavier et souris **21**

E Galerie **22**

1 Introduction

1.1 But

Faire un portage du jeu de plateau classique multijoueur Quoridor [1] par le biais d'une interface client-serveur et l'ajout de fonctionnalités sociales modernes.

Pour que la partie se termine, dans le mode classique, il suffit d'atteindre le côté opposé du plateau tout en empêchant l'adversaire de faire de même à l'aide de murs plaçables.

Concernant les fonctionnalités sociales, il est possible de gérer une liste d'amis, discuter avec ces derniers, créer une partie privée et consulter un classement des joueurs.

De par sa nature simple, le jeu se veut tout public malgré ses fonctionnalités sociales non modérées.

1.2 Glossaire

Client :	Application client
Game/Jeu :	Partie de jeu Quoridor
TLS :	Transport Layer Security
Bibliothèque header-only :	Une bibliothèque composée uniquement de fichiers headers. Les définitions des fonctions, classes et macros sont donc dans des fichiers headers.
Utilisateur :	Un individu possédant les identifiants pour se connecter à un compte existant dans la base de données ou ayant l'intention d'en créer un
Joueur :	Individu ou machine qui participe à un jeu Quoridor

1.3 Historique du document

Version	Date	Auteur(s)	Modifications
0.1.32	29.03.22	Abdellaoui Nabil & Aguililla Klein Esteban & Mathieu Van Den Bremt & Boulaich Ayman	réctification des fonctionnalités pour les mode de jeux et restructuration de l'ensemble du SRD
0.1.31	05.03.22	Bourgeois Noé	ajout de la fonction MyIncludeGraphicsMaxSize et du diagramme de sequence de création de compte et mise à jour du diagramme de sequence de connexion
0.1.30	04.03.22	Mathieu Van Den Bremt	Modification des tableau des USe Case et modification de certains newpage
0.1.29	04.03.22	Ismaël Secundar	Ajout des aperçu des menus et modification du use case du menu
0.1.28	01.03.22	Bourgeois Noé	division du Diagramme de séquence de connexion en 3 sous-sections et integration avec le texte
0.1.27	27.02.22	Boulaich Ayman	Réaménagement des différents diagrammes dans leurs contextes dans le srd
0.1.26	25.02.22	Boulaich Ayman	Ajout protocole
0.1.25	25.02.22	Boulaich Ayman	Ajout des librairies

1. les règles détaillées peuvent être consultées dans la section Annexes

0.1.24	23.02.22	Anton Romanova & Ismaël Secundar & Abdellaoui Nabil	Description du menu actuel et aperçu du menu principal
0.1.24	23.02.22	Anton Romanova & Ismaël Secundar & Abdellaoui Nabil	Description du menu actuel et aperçu du menu principal
0.1.23	13.12.21	Anton Romanova & Bourgeois Noé	correction du Diagramme de séquence de connexion
0.1.22	13.12.21	Anton Romanova & Ismaël Secundar & Mathieu Van Den Bremt	Ajout du diagramme pseudo-UML de l'API + section API
0.1.21	10.12.21	Aguililla Klein Esteban & Moruntale Vlad	Ajout du diagramme de séquence expliquant le déroulement d'un tour
0.1.20	10.12.21	Aguililla Klein Esteban & Moruntale Vlad	Ajout de la description des modes de jeux
0.1.19	9.12.21	Romanova Anton, Ismaël Secundar	Relecture & corrections orthographiques et grammaticales
0.1.18	8.12.21	Mathieu Van Den Bremt, Abdellaoui Nabil	Ajout du diagramme des classes pour les menus
0.1.17	28.11.21	Bourgeois Noé	Ajout du diagramme de séquence de connexion
0.1.17	28.11.21	Bourgeois Noé	typos, amélioration du calcul d'actualisation du classement, corrections use case de gestion de compte
0.1.16	24.11.21	Anton Romanova	Ajout de Mamadou Balde dans la liste des auteurs + fix stylistiques mineurs
0.1.15	24.11.21	Abdellaoui Nabil	Cas de disqualification et sauvegarde de partie
0.1.14	24.11.21	Boulaich Ayman	Gestion de partie et connexion
0.1.13	24.11.21	Bourgeois Noé	Base De Données
0.1.12	23.11.21	Bourgeois Noé	Logging
0.1.11	21.11.21	Romanova Anton, Ismaël Secundar	Section "Style de programmation" et "Internationalisation et Localisation"
0.1.10	21.11.21	Romanova Anton, Ismaël Secundar	Ajout de la sous-section 3.2.3 (Concurrence)
0.1.9	21.11.21	Bourgeois Noé	Actualisation du classement, colorisation du texte, hyperreferences, use case de gestion de compte
0.1.8	20.11.21	Bourgeois Noé	Lancement, enregistrement, créer une partie, rejoindre une partie
0.1.7	20.11.21	Mathieu Van Den Bremt & Nabil Abdellaoui	Modification des UseCase et amélioration divers pour User requirements
0.1.6	19.11.21	Romanova Anton, Ismaël Secundar	Brouillon pour besoins non-fonctionnels des besoins système
0.1.5	19.11.21	Aguililla Klein Esteban & Moruntale Vlad	Ajout du but et d'un annexe
0.1.4	19.11.21	Mathieu Van Den Bremt	Tableau Use Case
0.1.3	19.11.21	Boulaich Ayman	Sous section des besoins fonctionnels du système et début

0.1.2	18.11.21	Anton Romanova	Ajout de la section "Annexes" et "Design"
0.1.1	17.11.21	Mathieu Van Den Bremt & Nabil Abdellaoui	Début Besoin d'utilisateur + diagrammes Use Case
0.1	16.11.21	Anton Romanova	Structure générale

2 Besoin d'utilisateur

2.1 Besoins/Exigences fonctionnelles

2.1.1 Menu principal

Une fois la connexion faite, l'utilisateur basculera sur le menu d'accueil. Il aura plusieurs choix. Il pourra notamment lancer une partie, ajouter un(e) ami(e) et échanger des messages avec, consulter la liste des meilleurs joueurs et leurs score. Dans le cas où l'utilisateur a besoin d'aide, il pourra accéder à un menu qui va l'aiguiller. De plus, si un utilisateur se déconnecte de son compte un autre peut se connecter avec un autre compte. Chacun aura évidemment son compte propre à lui.

L'utilisateur pourra naviguer entre les différentes options grâce aux Touches directionnelles du clavier.

En configurant sa partie, l'utilisateur pourra indiquer quel autre joueur, parmi la liste d'amis, pourra rejoindre la partie. Les joueurs invités devront évidemment confirmer leur participation. Une fois que tout ceci est fait, les joueurs attendront que leurs adversaires rejoignent la partie avant de commencer à jouer.

Si le cours d'une partie a été précédemment sauvegardé, l'utilisateur pourra la reprendre là où elle s'était arrêtée à condition le ou les autres participants soient présent(s) pour continuer.

Si il le désire, l'utilisateur pourra aussi demander de l'aide, l'application affichera le fonctionnement du jeu et les différentes fonctionnalités de cette dernière.

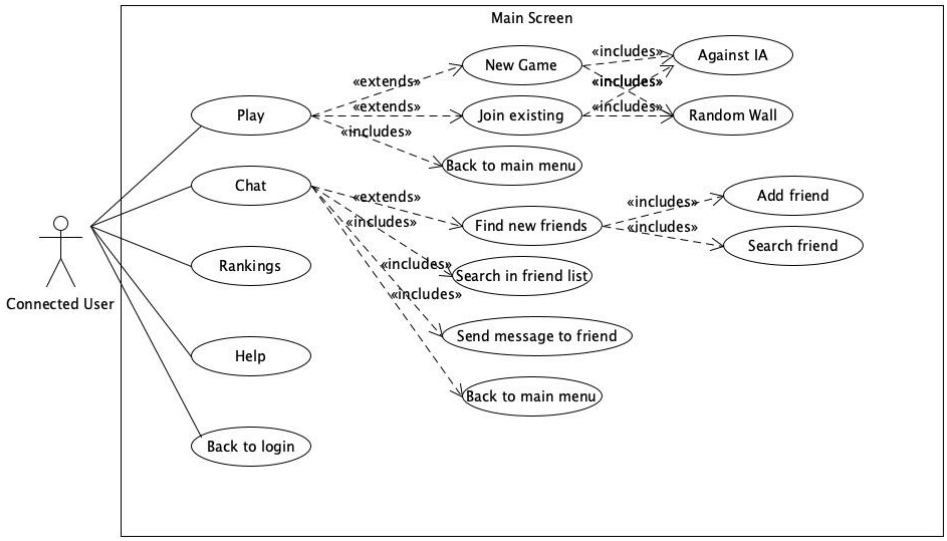
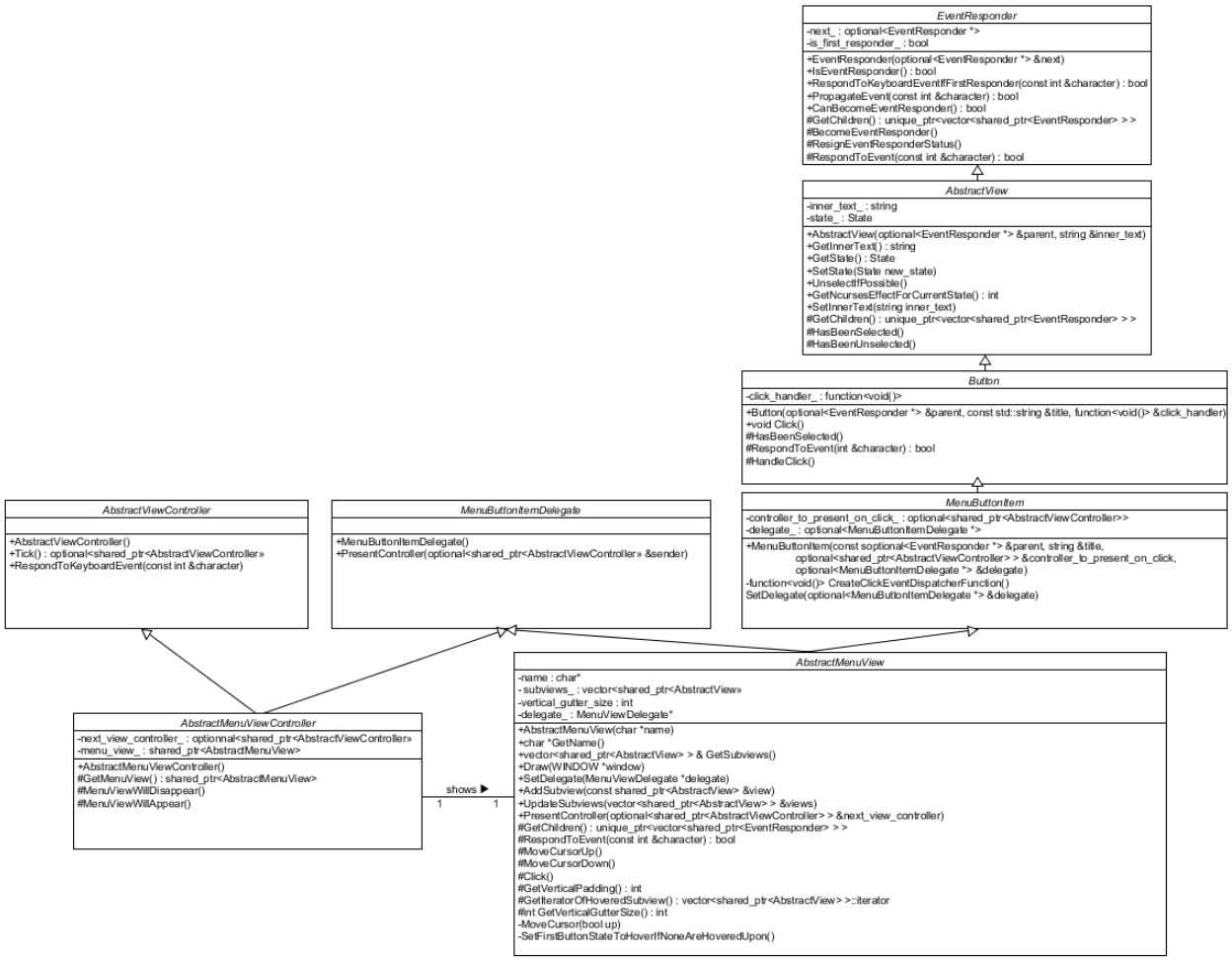


Figure 1 – Diagramme Use Case Des Menus. On peut y voir toute les actions possible depuis le menu principal.

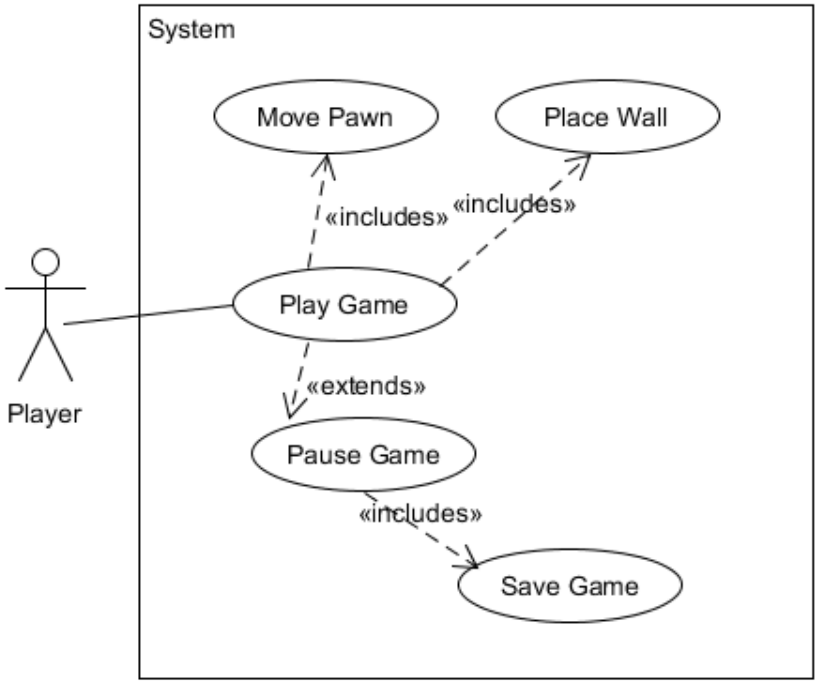
Use Case	Pré condition	Post condition	Cas général	Cas exceptionnels
Play	Néant	En fin de partie les résultat sont mis à jour dans le Ranking	Le programme lance la configuration d'une partie puis lance celle-ci après avoir lancé l'invitation d'un joueur	L'invité refuse l'invitation, l'utilisateur est notifié que l'invité a refusé l'invitation
Chat	Néant	L'utilisateur voit sa liste d'ami(e)s	L'utilisateur cible l'ami avec qui il veut parler	L'utilisateur n'est pas ami avec la cible, il peut chercher l'ami en l'ajoutant
Ranking	Un classement existe	L'utilisateur voit le classement	Le programme montre le classement des meilleurs joueurs	Néant
Help	L'utilisateur a besoin d'aide	Néant	Une explication brève du jeu s'affiche	Néant
Send message to friend	l'utilisateur cible existe	Le message est envoyé	Le programme envoie un message écrit par l'utilisateur à une personne de la liste d'amis de celui-ci	l'utilisateur s'envoie un message à lui-même, une erreur s'affiche
Search in friend list	l'utilisateur cherche un ami	Néant	L'utilisateur trouve l'ami qu'il cherchait dans sa liste d'amis	Néant
Search friend	l'utilisateur cherche l'ami qui doit être ajouté	L'utilisateur ajoute l'ami	Le programme sauvegarde le contact du nouvel ami dans la liste	Néant
Invite Friend to Game	L'ami invité doit faire partie de la liste d'ami	L'ami rejoint la partie	Le programme permet à l'utilisateur d'inviter un ami à jouer avec lui	Si l'ami ciblé refuse l'invitation, le programme notifie l'utilisateur que l'ami ciblé a refusé
Join Game	Un autre utilisateur a invité l'utilisateur	L'utilisateur rejoint la partie	Le programme invite l'utilisateur à rejoindre une partie	Si l'utilisateur refuse alors le programme envoie un message à l'utilisateur qui a envoyé l'invitation
Play Game	Néant	En fin de partie les résultat sont mis à jour dans le classement	Le programme démarre la partie	Si la partie est quittée en cours de jeu, alors le classement n'est pas mis à jour
Logout	Néant	Le programme retourne à l'écran de connexion	Le programme déconnecte le joueur	Néant

Le diagramme ci-dessus représente tout le fonctionnement orienté MVC des menus. D'une part il y a les contrôleurs qui interagissent avec l'utilisateur et qui attendent une entrée de la part de ces derniers, et d'autre part il y a les vues qui, comme leurs noms l'indiquent, sont la partie visuelle des menus. Cette une version des menus propre aux terminaux.



2.1.2 Durant une partie

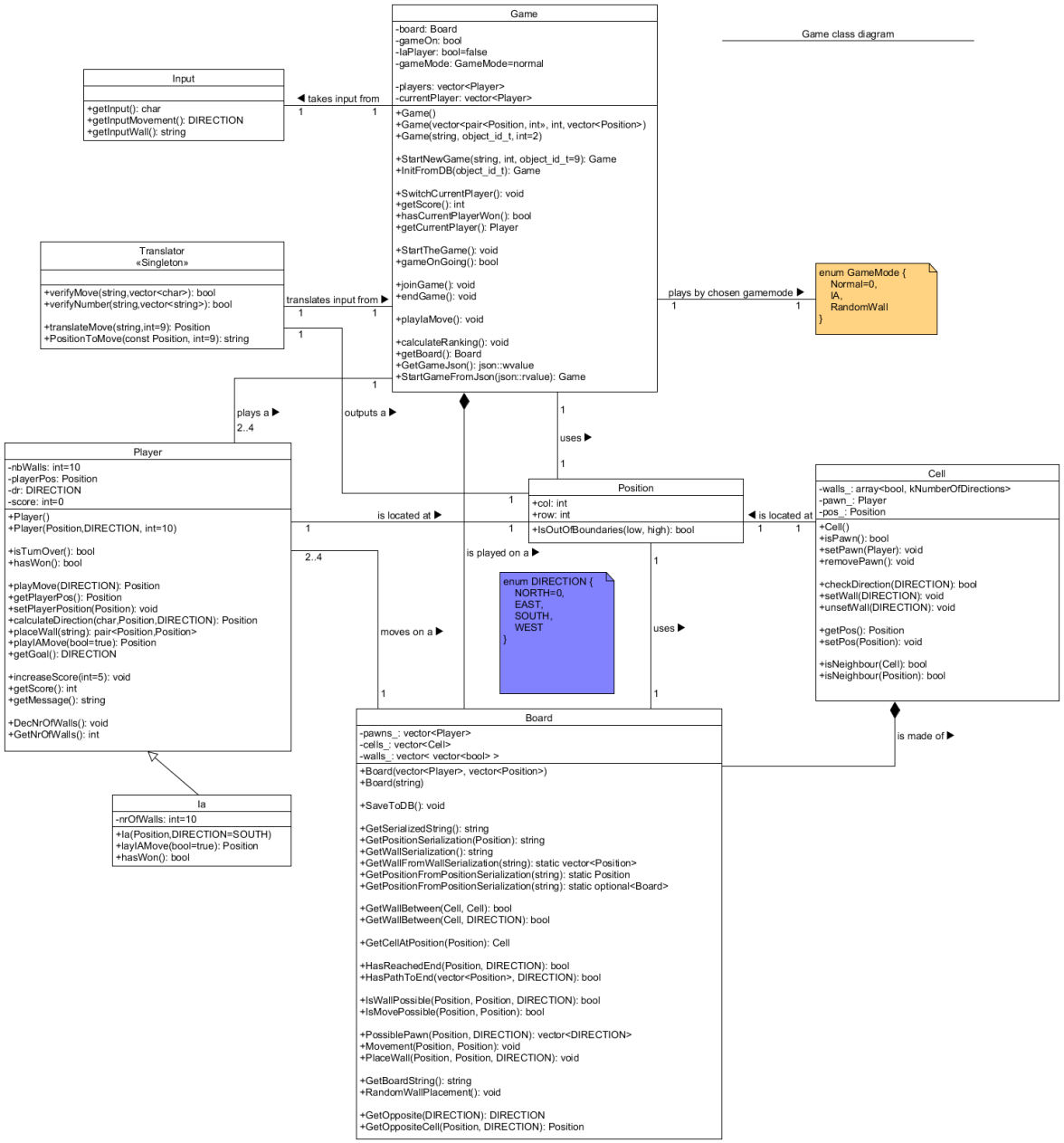
Quand c'est son tour, le joueur doit pouvoir effectuer une action, déplacer un pion ou mettre un mur. Cette action est représentée comme un message envoyé à l'application qui agira sur lui-même en conséquence. Un plateau de jeu est défini comme une matrice de cellule qui partagent tous une position précise, ces positions sont soit vides, soit occupés par un pion.



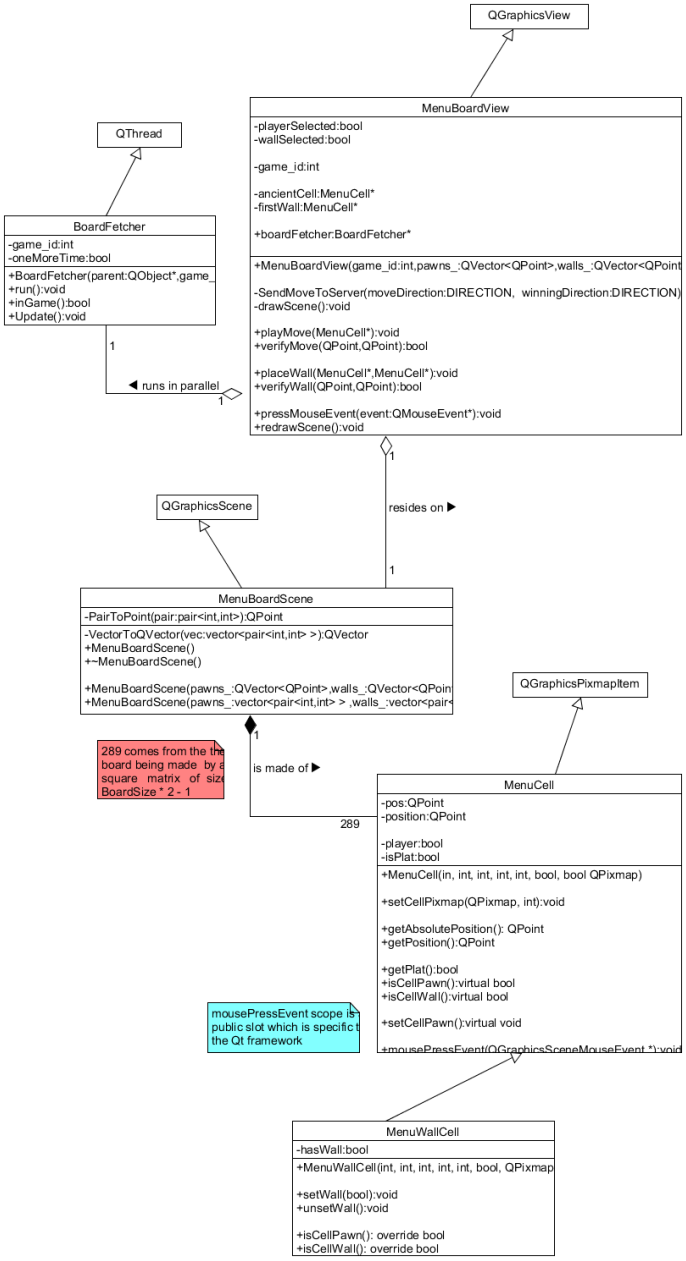
Use Case	Pré condition	Post condition	Cas général	Cas exceptionnels
Play Game	L'utilisateur a lancé et configuré une partie et il y a un nombre requis de Joueur	A la fin de la partie on retourne sur l'écran principal	Le programme lance le jeu Quoridor et demande tour par tour, quel actions les joueur veulent faire	Si un joueur quitte en cours de partie un message est envoyé à l'adversaire
Move pawn	Le pion peut être déplacé comme le veut l'utilisateur	Le plateau est modifié avec la nouvelle position du pion	Le programme déplace le pion	Néant

Place Wall	L'emplacement désigné pour le mur est libre et ne bloque pas entièrement un pion	Le plateau est modifié avec le nouveau mur	Le programme place un mur dans la position choisie	Néant
Pause Game	Néant	Le jeu est mis en pause, sauvegardé et ensuite terminé	Le programme demande à l'adversaire si celui-ci accepte de mettre en pause le jeu	Néant
Save	Le jeu a été mis en pause sur l'accord des deux joueurs	La partie est sauvegardée	Le programme sauvegarde la partie tel qu'elle est	Si le programme n'a pas réussi à sauvegarder les données, alors celui-ci envoie un message d'erreur

Comment la structure d'un jeu, son plateau, ses mouvements sont disposés ainsi que leurs interactions :



Une version alternative correspondant à la partie "Interface graphique" écrite grâce à la librairie Qt :



2.1.3 Commande

L'utilisateur devra entrer une commande pour chaque action qu'il effectuera. Par exemple les commandes en cours de jeu seront :
Pour déplacer un pion : M->"direction (t.q F, B, E et W)" (ex : "M->F")
Si un autre pion se trouve sur la case d'arrivée le joueur peut sauter par dessus et continuer en avant ou tourner à droite ou à gauche si il est impossible d'aller tout droit. Format de la commande :
M->"direction"->"direction" Pour mettre un mur : W->"première case"->"deuxième case"->"direction du

mur"
Pour la partie interface graphique, l'utilisateur devra juste cliquer, avec sa souris sur le plateau de jeu pour effectuer une action qui sera traduite par le programme comme une commande écrite.

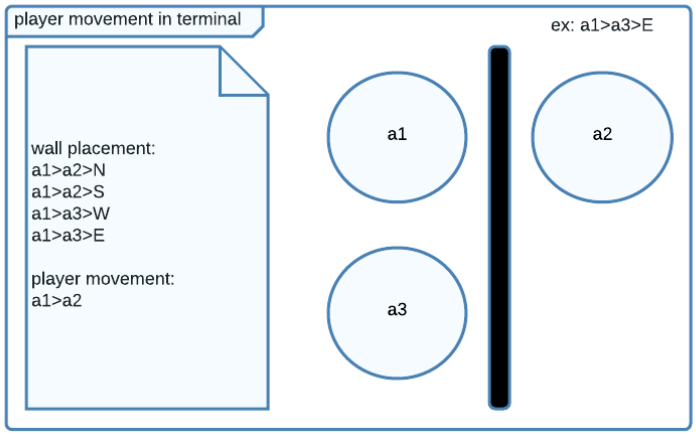


FIGURE 2 – Exemple de mouvement lors d'une partie

2.1.4 Mode de jeu

L'unique mode de jeu présent est celui d'un classique 1 versus 1.

2.1.5 Messagerie

L'utilisateur pourra échanger des messages uniquement avec ses amis qu'il devra ajouter au travers d'un menu spécifique.

2.2 Exigences non-fonctionnelles

2.2.1 Inscription et connexion

L'utilisateur doit être capable de présenter un nom de compte ainsi qu'un mot de passe au démarrage du jeu pour pouvoir se connecter et accéder à son menu principal. Si il n'a pas de compte ou si il désire en recréer un, il lui est possible d'en créer un nouveau. Pour la création d'un compte, aucun mail n'est nécessaire à introduire, l'utilisateur doit juste présenter un nouveau pseudonyme accompagné d'un mot de passe pour terminer le processus de création.

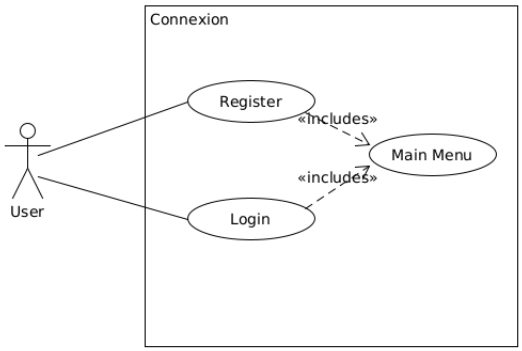


Figure 3 – Diagramme Use Case pour la connexion

Use Case	Pré condition	Post condition	Cas général	Cas exceptionnels
Register	l'utilisateur n'a pas encore un compte	On enregistre le nouveau compte	Le programme demande un nom et un mot de passe	Si le nom est déjà utilisé alors on envoie une erreur
Connect	L'utilisateur a déjà un compte	L'utilisateur rentre dans le programme	Le programme vérifie les informations données par le client ensuite, il lui permet de continuer sur le programme	Si les informations sont incorrectes, alors on envoie une erreur

3 Besoins du système

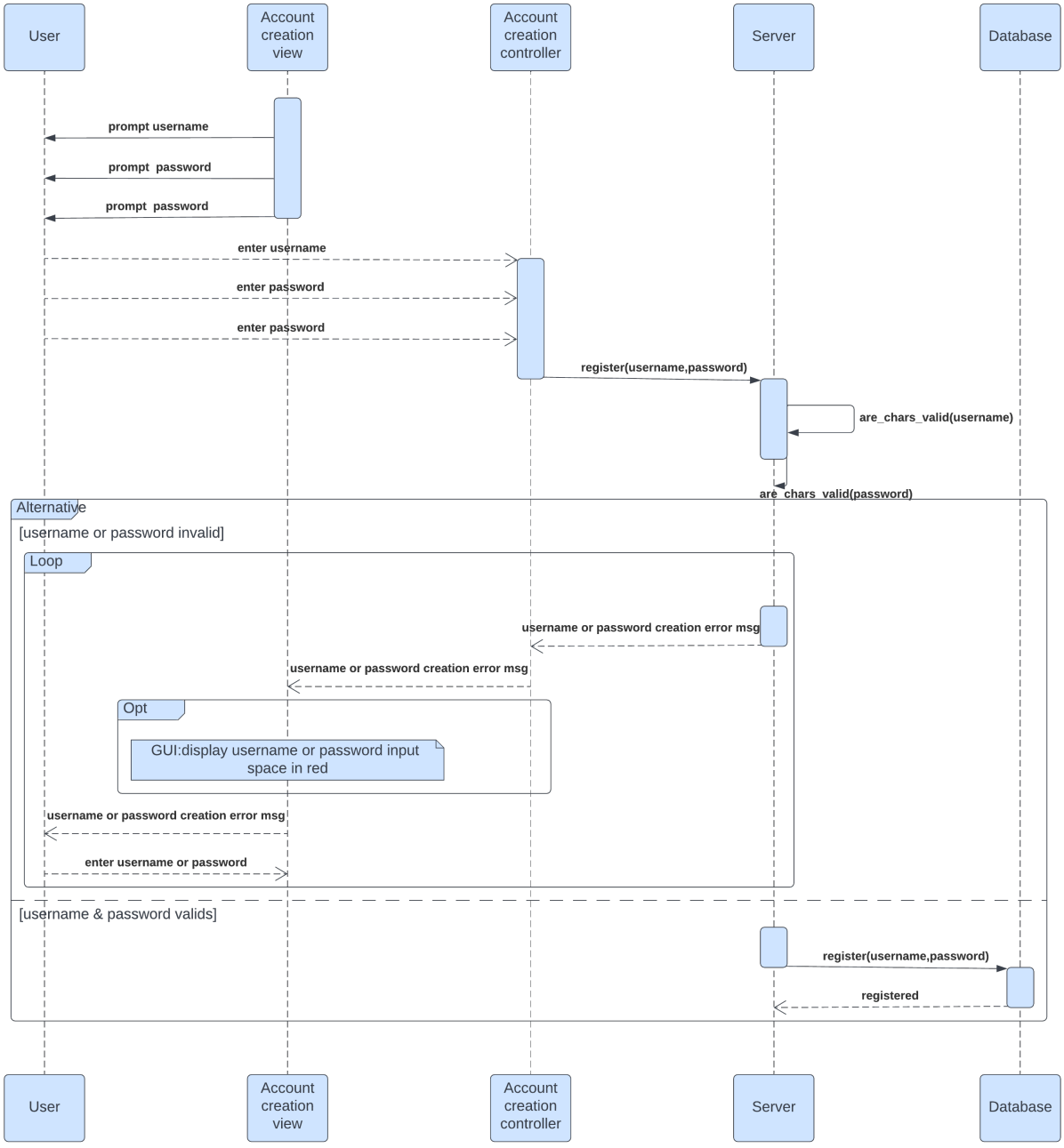
3.1 Besoins fonctionnels

3.1.1 Enregistrement

Conditions :
- Le pseudo n'est pas présent dans la Base De Donnée et contient :
  — Uniquement des caractères alphanumériques + "_" + "-"
  — Un nombre de caractères entre 3 et 25

- Le mot de passe contient :
    — Plus de 8 caractères

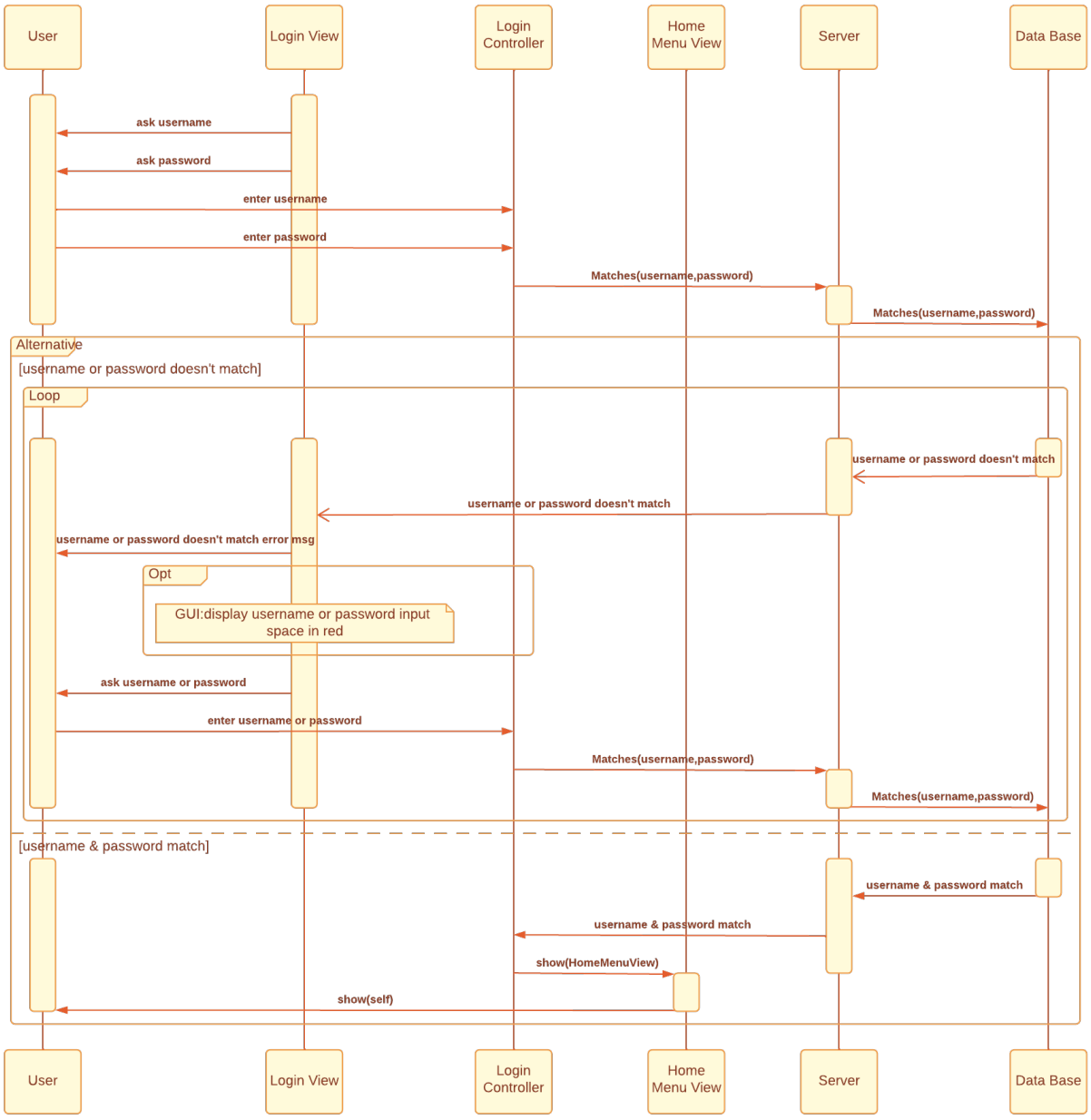
Si les conditions sont respectées, le nom d'utilisateur et le mot de passe sont enregistrés dans la base de données et la fenêtre du **??** apparaît.



3.1.2 Connexion

Conditions :

L'utilisateur possède un compte :
Si le pseudo est trouvé et que le mot de passe entré par le joueur correspond alors l'utilisateur sera connecté. Dans le cas contraire,l'utilisateur est informé par le programme que son mot de passe ne correspond pas au pseudo inscrit :



3.1.3 Rejoindre une partie

Si l'utilisateur sélectionne "Play Game" en ayant d'abord indiquer le nom de la partie, il la rejoindra si celle-ci est enregistrée dans la base de données.
Si il reçoit une invitation, le jeu lié à cette dernière lui sera disponible.

3.1.4 Gestion de partie

Lorsque le joueur lance la partie le plateau du jeu s'affiche en fonction des options de jeu choisi par les joueurs. Le système gère alors les différentes manipulations que peuvent faire les joueurs . Le système doit aussi actualiser le plateau lorsque le joueur bouge ou place un mur. Le nombre de murs de chaque joueur est connue par le système et est décrémenté à chaque fois que le joueur place un mur sur le terrain.

Voici le diagramme des séquences d'évènements qui se déroulent durant un tour normal pour le joueur. On y retrouve la connexion entre le client et le serveur lors d'une action demandée par le joueur. Diagramme de séquence représentant la création d'un jeu

3.1.5 Actualisation du classement

Après une partie, le classement est mis à jour :
- Si c'est une égalité, les scores ne changent pas.

Sinon, les classements du gagnant et du perdant sont respectivement augmentés et diminués de

$$\frac{\text{(Nombre de mouvements effecutés * 10) + (100 points de bonus)}}{}$$

- Le classement d'un joueur ne descend pas sous 0

3.1.6 Base de données

La base de donnée SQL contient :

— Les données correspondant à chaque joueur :

— Les pseudos (:clé)

— Les mots de passe

— Un classement

— Une liste d'amis

— Une liste de conversations

— Une liste de parties enregistrées

— Les parties enregistrées

FIGURE 4 – Représentation de la base de donnée.

3.1.7 Logging

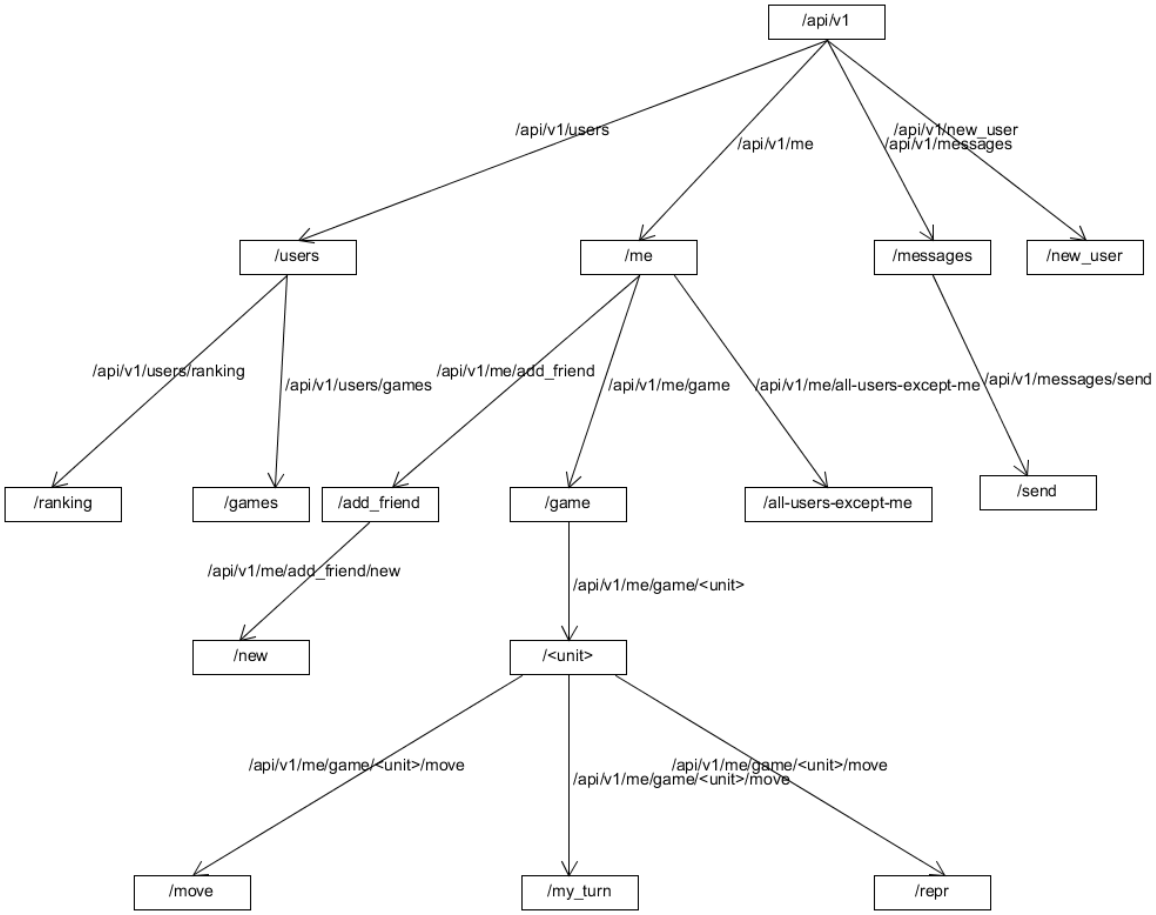
Tout évènement du programme incluant un accès à la base de données est enregistré sous forme ASCII ou binaire dans un log file et est consultable par l'administrateur système sous forme ASCII.

3.1.8 Latence

Toutes les actions doivent passer d'un utilisateur à un autre avec une latence minimale en mettant à jour le plateau le plus rapidement possible et ainsi donner l'illusion que les actions se font instantanément.

3.1.9 API

Le programme utilise une REST API pour les communications entre le client et le serveur. L'API qui sera implémentée à l'aide de la bibliothèque Craw. Cette bibliothèque est header-only, ce qui permettra de faciliter la portabilité. Les appels se font sous forme de requêtes "GET". Les appels API assurent toute la communication client-serveur et permettent entre-autres de garder les parties des adversaires synchronisées, mais aussi avoir un des chats avec d'autres utilisateurs.



Dans ce diagramme on remarque que chaque connexion au serveur par l'intermédiaire de l'api nous permet de récupérer une donnée précise. Et que il y a un système d'arborescence de route d'URL qui sert de chemin pour atteindre cette même donnée.

3.2 Besoins non-fonctionnels

Contraintes liées au matériel.

3.2.1 Portabilité

Le programme doit fonctionner sur Linux. Si possible sans trop de modifications, il devrait également être compatible avec Windows.

3.2.2 Réseau

Les clients et le serveur doivent être connectés à un même réseau.

3.2.3 Concurrence

Afin de pouvoir gérer plusieurs connexions, le programme serveur doit s'exécuter en parallèle. Les transactions de la base de données effectuées en concurrence ne peuvent pas violer l'intégrité des données. Le contrôle optimiste de la concurrence sera utilisé, pour améliorer les performances et éviter les deadlocks.

Du côté client, la concurrence doit également être utilisée pour avoir une interface graphique utilisateur (GUI) réactive.

3.2.4 Sécurité

Le protocole SSH nous permet de passer par des websockets pour interagir avec l'API de manière sécurisé.

3.2.5 Légalité

Le système ne garantit pas le respect du GDPR excepté pour la suppression de compte.

4 Design et fonctionnement du système

5 Annexes

A Règles de base de Quoridor

Ces dernières peuvent être consultées sur le site de son éditeur Gigamic
https://www.gigamic.com/files/catalog/products/rules/quoridor-classic-fr.pdf

B Librairies

Dans le cadre de ce projet d'année , les librairies utilisées sont :

- ncurses
- boost
- uuid
- Crow
- sqlite3
- curl
- Qt6

C Commentaires sur les diagrammes

Les diagrammes séquentiels ont un rappel des classes à la fin de chaque ligne. Cela a pour but d'augmenter la lisibilité.

D Commande sous forme d'entrée clavier et souris

Entrée	Sortie
En terminal : $\leftarrow, \uparrow, \rightarrow, \downarrow$	Permet de naviguer dans les menus ou/et sélectionner différentes options
En terminal : [Enter]	Permet d'appuyer sur sur un bouton ou d'entrer dans un champ de texte
Avec l'interface graphique : Clic de souris	Permet d'appuyer sur un bouton dans l'interface ou d'effectuer un mouvement en plein jeu

E Galerie

FIGURE 5 – Sur terminal : Menu où l'on peut se connecter

FIGURE 6 – Sur terminal : Menu où l'on peut naviguer pour choisir une option.

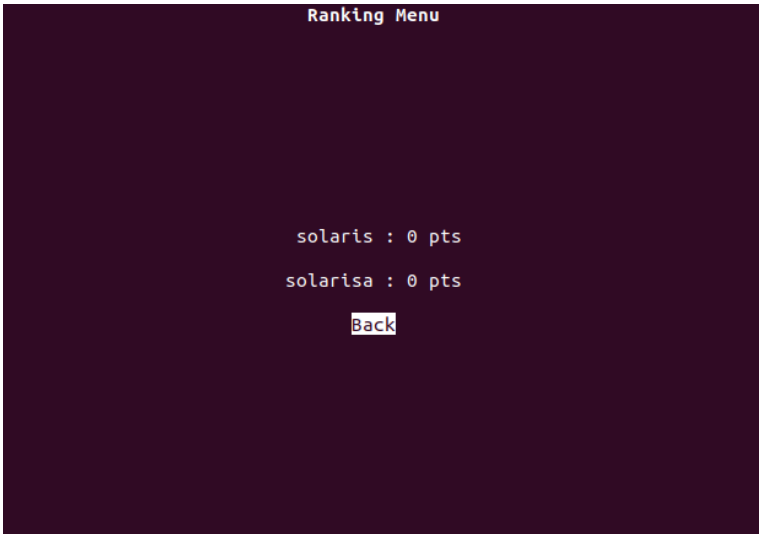


FIGURE 7 – Sur terminal : Menu où l'on peut regarder le classement local.

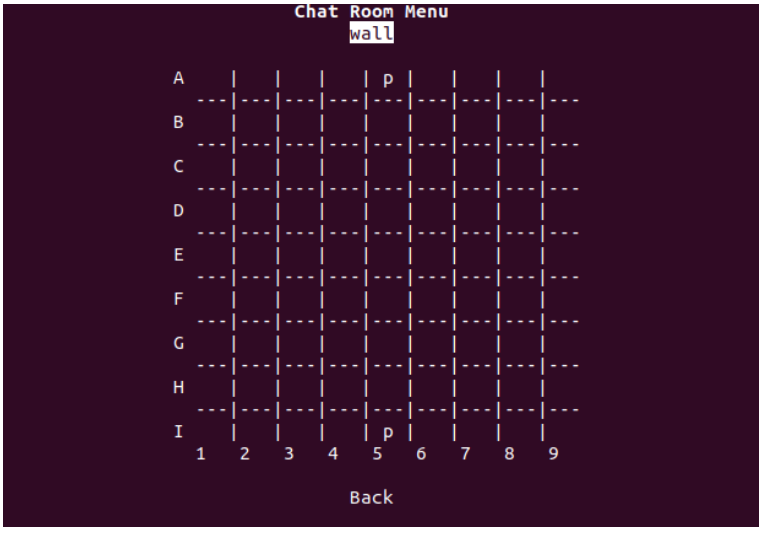


FIGURE 8 – Sur terminal : L'écran de la partie.

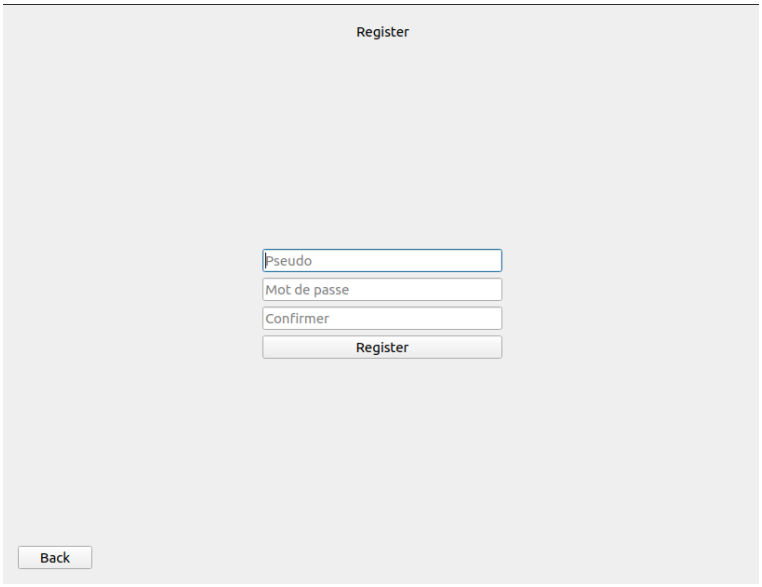


FIGURE 9 – Sur interface graphique : L'écran de la partie.

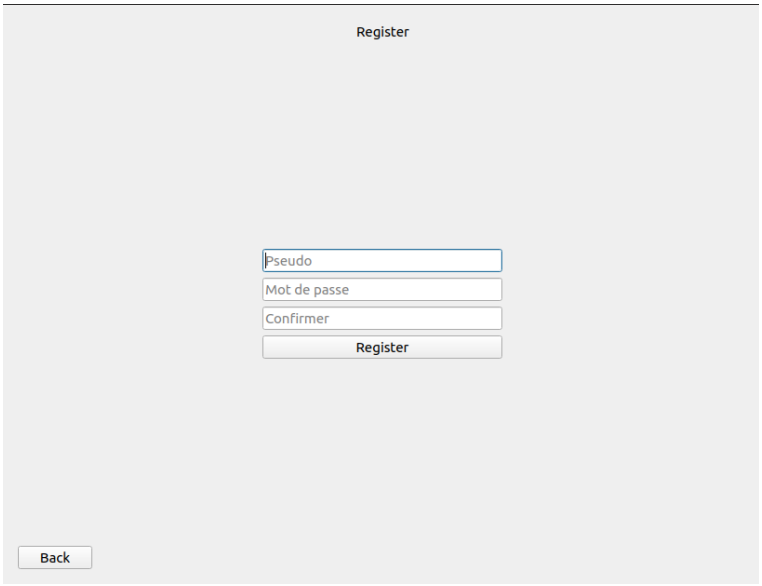


FIGURE 10 – Sur interface graphique : Menu où l'on peut se connecter.

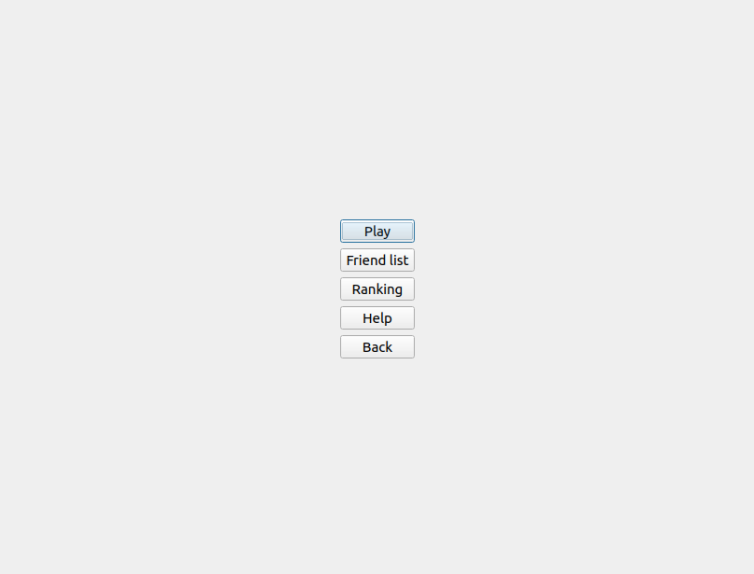


FIGURE 11 – Sur interface graphique : Menu sur lequel on peut appuyer sur un bouton pour en ouvrir un autre

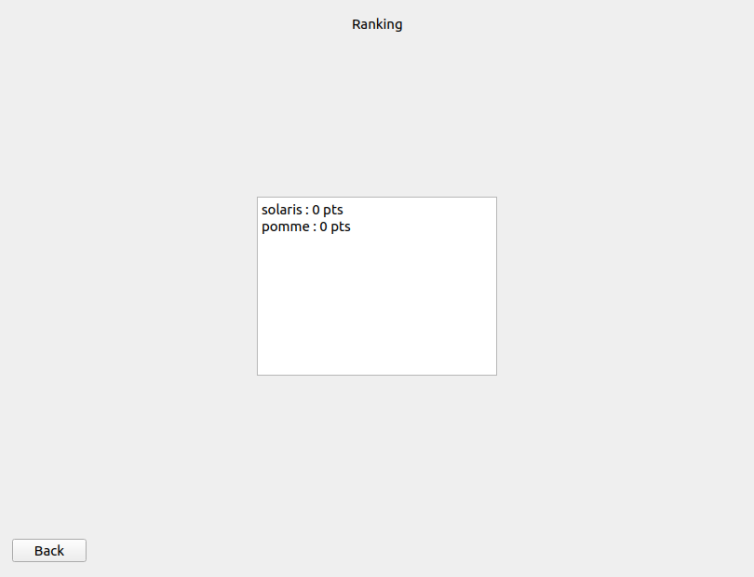


FIGURE 12 – Sur interface graphique : Menu où l'on peut regarder le classement local.

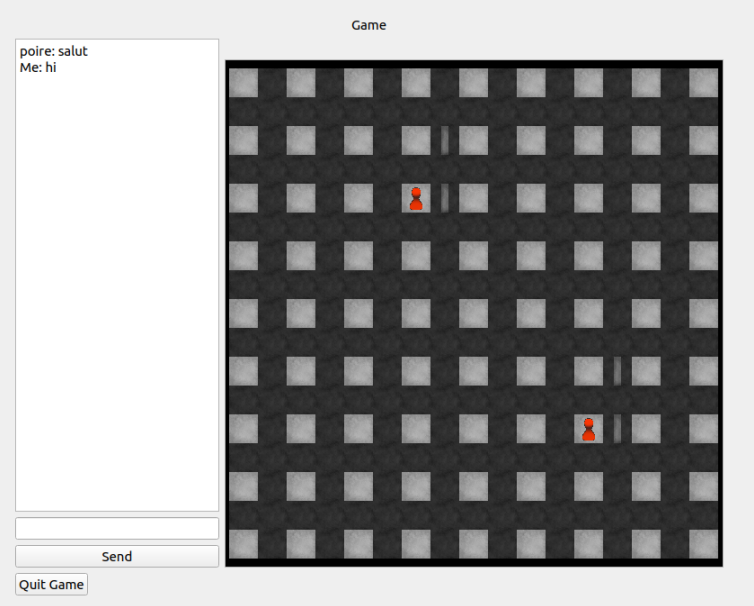


FIGURE 13 – Sur interface graphique : Menu où on peut bouger les pions et discuter avec son adversaire.