

# 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, Mamadou Balde

Decembre 2021

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	But	3
1.2	Glossaire	3
1.3	Historique du document	3
<b>2</b>	<b>Besoin d'utilisateur</b>	<b>5</b>
2.1	Besoins/Exigences fonctionnelles	5
2.1.1	Menu de bienvenu	5
2.1.2	Menu principal	5
2.1.3	Durant une partie	8
2.1.4	Humain contre ordinateur	10
2.1.5	Légalité	10
2.1.6	Messagerie	10
2.2	Exigences non-fonctionnelles	10
2.2.1	Inscription et connexion	10
2.2.2	Commande	12
2.2.3	Déconnexion en partie	12
<b>3</b>	<b>Besoins du système</b>	<b>13</b>
3.1	Besoins fonctionnels	13
3.1.1	Enregistrement	13
3.1.2	Connexion	15
3.1.3	Menu principal	16
3.1.4	Configuration de partie	16
3.1.5	Gestion de partie	16
3.1.6	Actualisation du classement	16
3.1.7	Base de données	17
3.1.8	Logging	17
3.1.9	Mode de jeu	17
3.1.10	Latence	17
3.1.11	API	17
3.1.12	diagramme de classe Server side	17
3.2	Besoins non-fonctionnels	17
3.2.1	Portabilité	18

3.2.2	Réseau . . . . .	18
3.2.3	Concurrence . . . . .	18
3.2.4	Sécurité . . . . .	18
3.2.5	Fiabilité . . . . .	18
3.2.6	Internationalisation et Localisation . . . . .	18
3.2.7	Légalité . . . . .	18
<b>4</b>	<b>Design et fonctionnement du système</b>	<b>18</b>
4.1	Jeu . . . . .	18
<b>5</b>	<b>Images d'illustration</b>	<b>19</b>
<b>6</b>	<b>Annexes</b>	<b>19</b>
<b>A</b>	<b>Style de programmation</b>	<b>19</b>
<b>B</b>	<b>Règles de base de Quoridor</b>	<b>19</b>
<b>C</b>	<b>Librairies</b>	<b>19</b>
<b>D</b>	<b>Protocole</b>	<b>19</b>
<b>E</b>	<b>Commentaires sur les diagrammes</b>	<b>19</b>

# 1 Introduction

## 1.1 But

Faire un portage du jeu de plateau classique multijoueur Quoridor <sup>1</sup> 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

<b>Client :</b>	Application client
<b>Game/Jeu :</b>	Partie de jeu Quoridor
<b>TLS :</b>	Transport Layer Security
<b>Bibliothèque header-only :</b>	Une bibliothèque composée uniquement de fichiers headers. Les définitions des fonctions, classes et macros sont donc dans des fichiers headers.
<b>Utilisateur :</b>	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
<b>Joueur :</b>	Individu ou machine qui participe à un jeu Quoridor

## 1.3 Historique du document

Version	Date	Auteur(s)	Modifications
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
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

---

1. les règles détaillées peuvent être consultées dans la section [Annexes](#)

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, correction 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, hyperreférences, 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 de bienvenu

En démarrant le jeu, un menu de bienvenu apparaîtra. Si l'utilisateur n'a pas de compte il aura la possibilité d'en créer un.

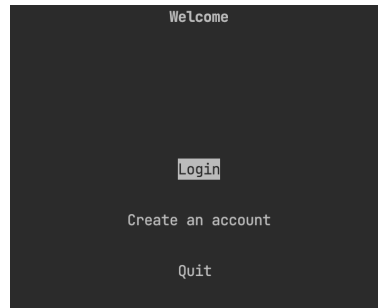


FIGURE 1 – Aperçu du menu de bienvenu

#### 2.1.2 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.

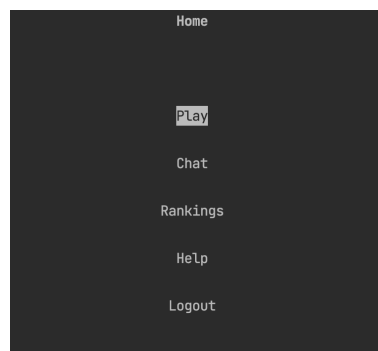


FIGURE 2 – Aperçu du menu principal

L'utilisateur pourra configurer sa partie en modifiant plusieurs paramètres. Il devra indiquer le nombre de participants qui seront dans la partie. Ils pourront être 2 ou 4. Par la suite, il devra indiquer quels utilisateurs, parmi la liste d'amis, pourront 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 que tous les autres participants soient présents 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.

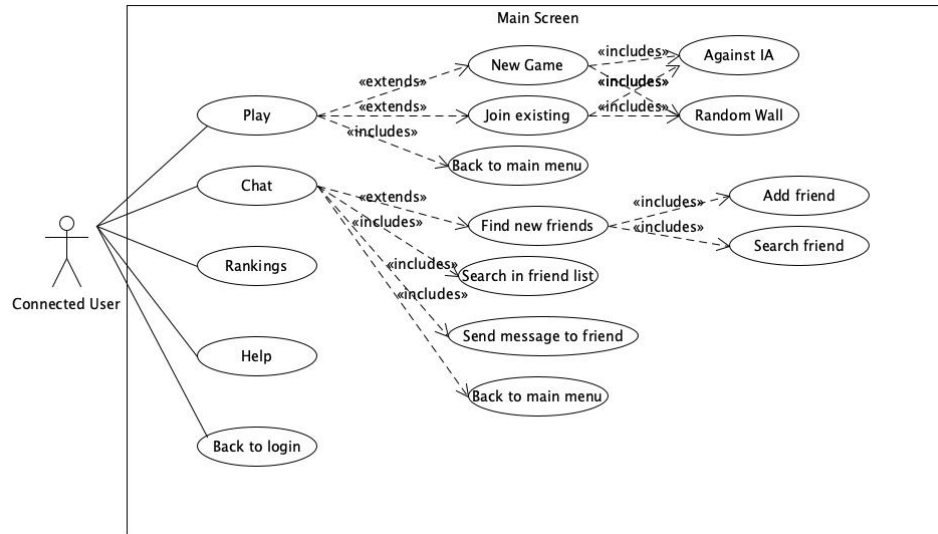
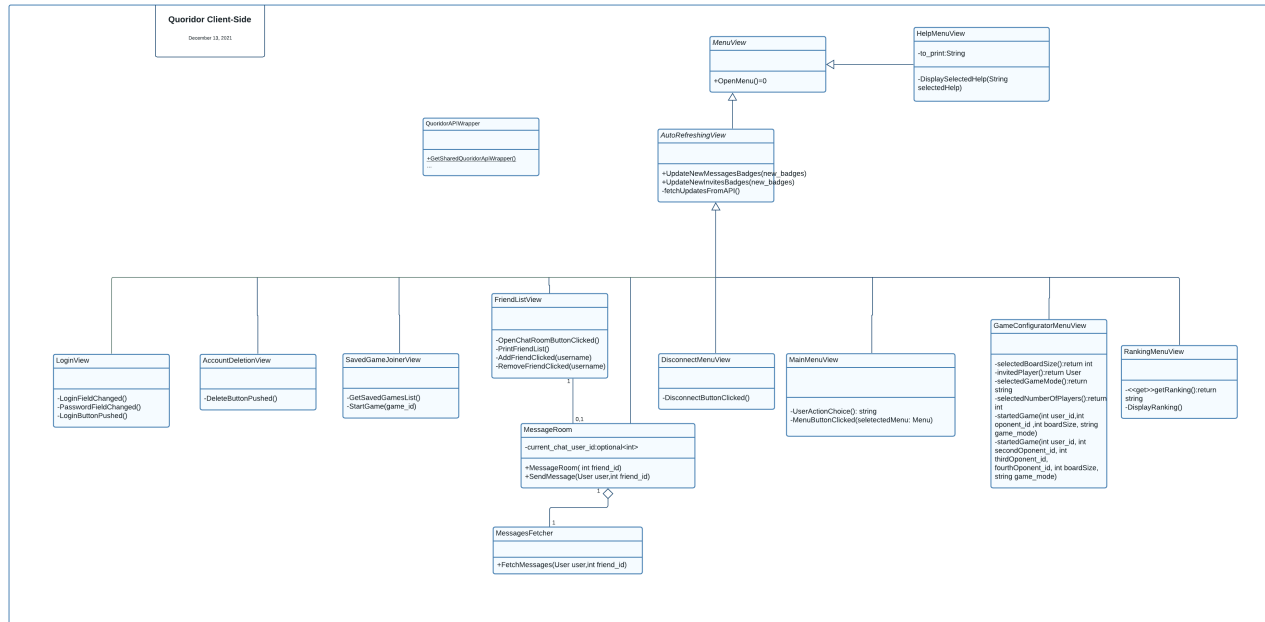


FIGURE 3 – Diagramme Use Case Des Menus. On peut y voir toute les actions possible depuis le menu principal.

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

Diagramme de classe des menus :



Voici le diagramme de classe centré sur le côté client et les menus, qui sont des entités à travers lesquelles les utilisateurs naviguent pour effectuer des actions telles que configurer et lancer une partie. Celles-ci s'occupent aussi de l'affichage sur le terminal.

Dans ce diagramme nous faisons la distinction entre les menus qui nécessitent d'avoir un accès à un serveur et les autres qui n'en ont pas besoin.

La ChatRoom utilise un thread pour pouvoir envoyer et recevoir des messages en même temps(l'envoi est géré dans la ChatRoom et la réception dans le thread).

### 2.1.3 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. En plein duel, l'un des joueurs peut proposer au reste des participants de mettre le jeu en pause et de sauvegarder la partie en cours pour pouvoir la continuer plus tard. Les joueurs peuvent aussi déclarer forfait et donc se retirer.

Si l'un des joueurs se déconnecte sans proposer de sauvegarder la partie, il est considéré comme disqualifié, la partie est interrompue et son opposant ressort gagnant. Il en est de même pour une partie à 4 sauf qu'aucun gagnant n'est déterminé.



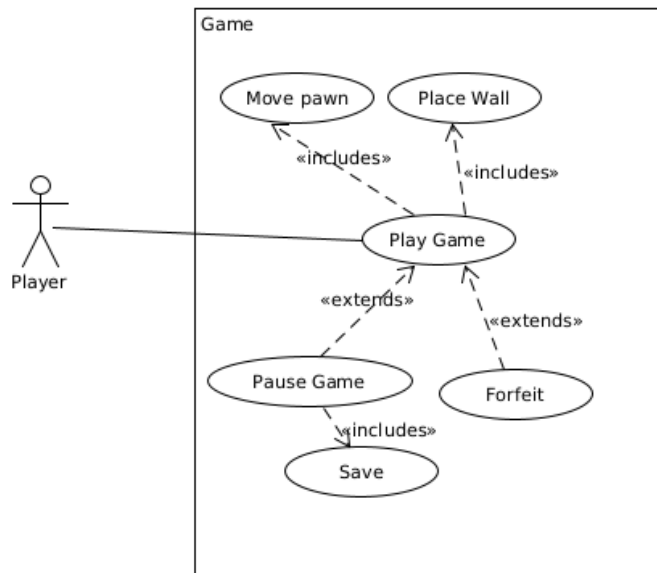


FIGURE 4 – Diagramme Use Case d’une partie de jeu

Use Case	Pré condition	Post condition	Cas général	Cas exceptionnels
Play Game	L'utilisateur à lancer et configurer 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
Forfeit	Néant	La partie est terminée sur une victoire adverse	Le programme termine la partie	Néant

**Mode de jeux** L'utilisateur peut choisir un mode de jeux différent au mode classique parmi les modes suivants.

**Murs aléatoires** Des murs apparaissent de façon aléatoire<sup>2</sup> sur le plateau.

#### 2.1.4 Humain contre ordinateur

L'adversaire est contrôlé par l'ordinateur. Ce dernier est capable des mêmes actions qu'un joueur normal.

#### 2.1.5 Légalité

L'utilisateur peut supprimer son compte selon le GDPR.

#### 2.1.6 Messagerie

L'utilisateur pourra voir la date et l'heure d'envoi d'un message.

## 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.

---

2. les murs ne pourront pas bloquer totalement le chemin d'un joueur vers son objectif

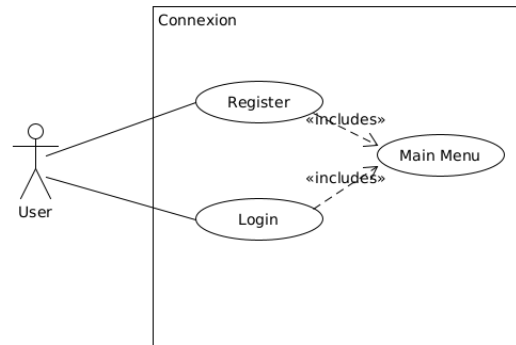


FIGURE 5 – Diagramme Use Case pour la connexion

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

### 2.2.2 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 : "case de départ">"case d'arrivée"

Pour mettre un mur : "première case">"deuxième case">"Direction du mur"

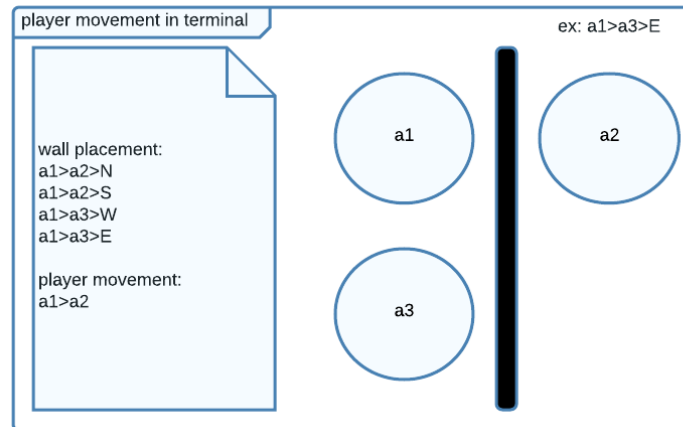


FIGURE 6 – Exemple de mouvement lors d'une partie

### 2.2.3 Déconnexion en partie

Dans le cas d'une déconnexion durant une partie, le jeu est interrompu et tout les joueurs sont renvoyés au menu principal.

## 3 Besoins du système

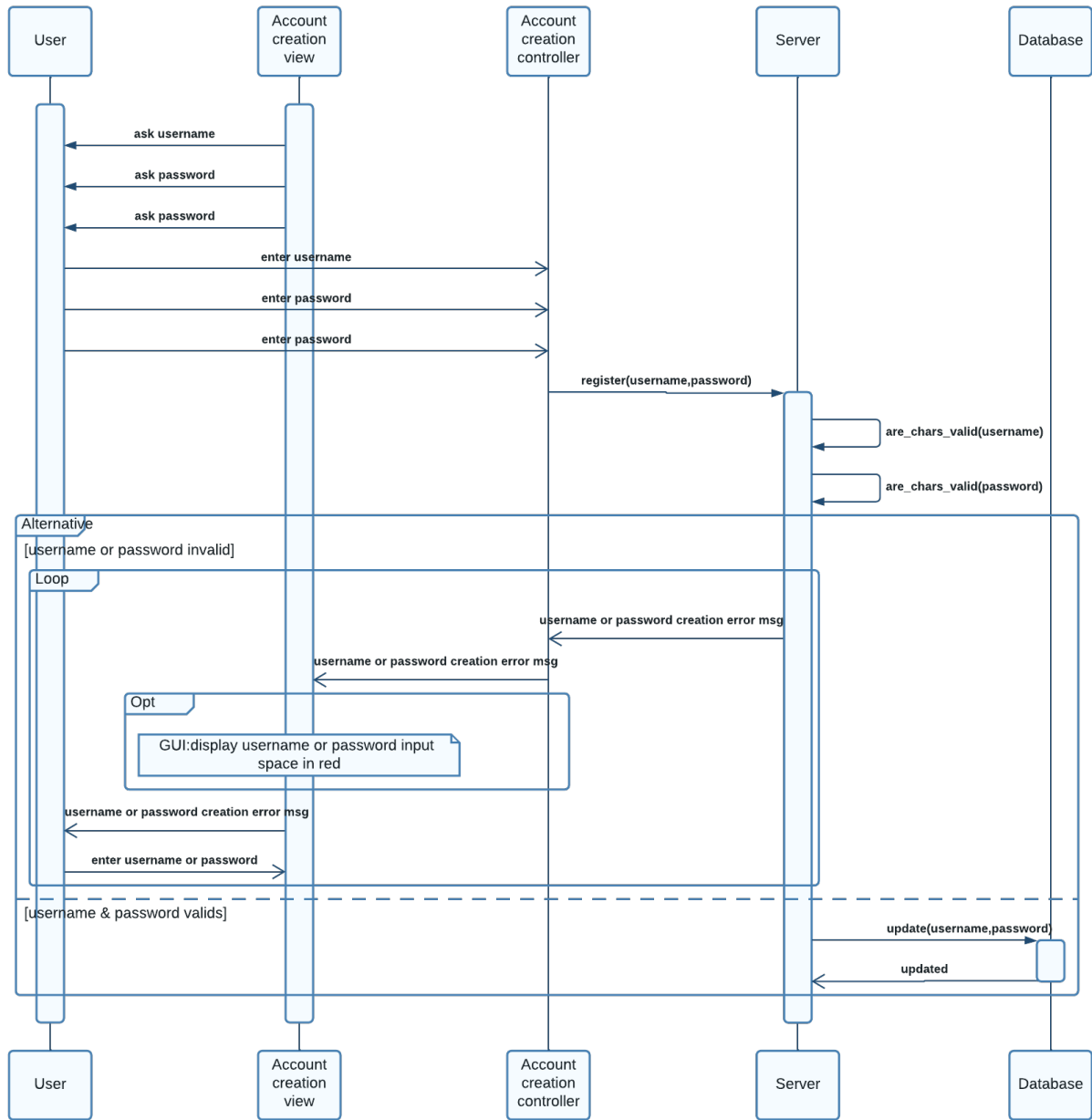
### 3.1 Besoins fonctionnels

#### 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
  - 4 caractères spéciaux

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 [Menu principal](#) apparaît.

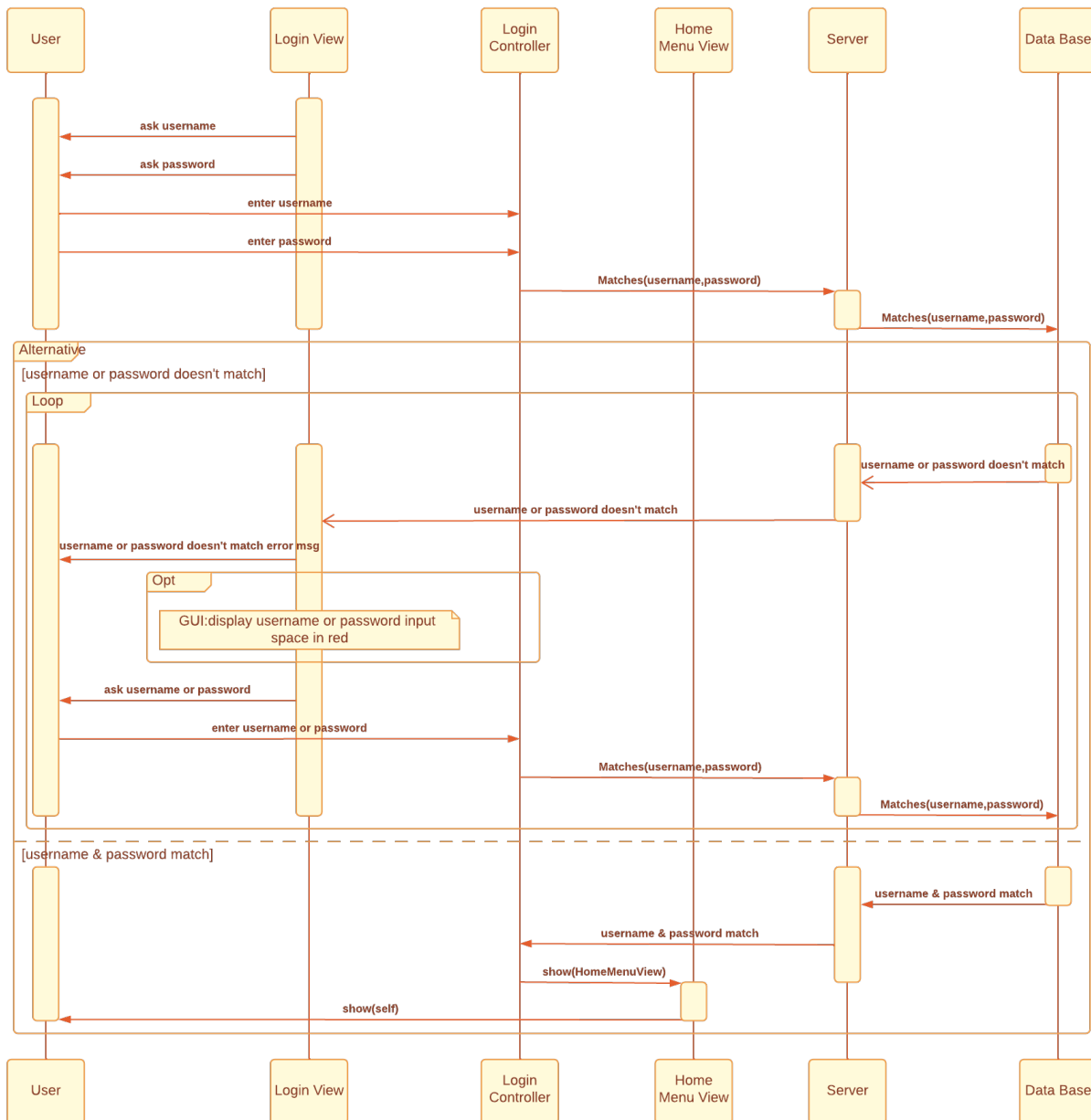


### 3.1.2 Connexion

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. Le programme donnera alors à l'utilisateur la possibilité de retrouver son mot de passe via la réponse à une question secrète stockée dans la base de données lors de l'enregistrement du compte :



### 3.1.3 Menu principal

Use case : [Menu principal](#)

### 3.1.4 Configuration de partie

**Création de partie** Si l'utilisateur sélectionne "new game", une fenêtre de configuration de jeu apparaît contenant les paramètres par défaut :

- la taille du plateau : 9
  - l'apparition aléatoire de murs sur le plateau : "on" (switch sur GUI)
  - le mode de jeu
  - le ou les autre(s) joueur(s) :
    - un ou des humain(s) qui ont rejoint
  - l'invitation d'autres joueurs humains sous forme de commande pour ouvrir une fenêtre affichant les joueurs connectés et un espace de recherche
- L'utilisateur peut les modifier et, à tout moment, démarrer la partie avec les paramètres affichés

**Rejoindre une partie** Si l'utilisateur sélectionne "join", une liste des joueurs créant une partie apparaît. Il peut en sélectionner une pour la rejoindre.  
Si l'utilisateur reçoit une invitation, celle-ci est affichée superposée à la fenêtre actuellement affichée sauf si il est en train de jouer.

### 3.1.5 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 joueur peut mettre pause au jeu et déclarer forfait via des commandes . 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

### 3.1.6 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

$$\frac{(\text{taille du plateau} + \text{nombre de murs}) * (\text{score perdant} + 1)}{\text{nombre de tours} * \text{score gagnant} + 1}$$

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



### 3.1.7 Base de données

La base de donnée [SQL](#) contient :

- les données correspondant à chaque joueur :
  - pseudo ( :clé)
  - mot de passe
  - chemin d'accès vers le fichier de l'image du profil
  - classement
  - liste d'amis
  - liste de conversations
  - liste de parties enregistrées
  - les conversations
- les parties enregistrées
- le classement

### 3.1.8 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 consultable par l'administrateur système sous forme ASCII.

### 3.1.9 Mode de jeu

Le système doit adapter les règles du jeu selon le mode de jeu choisi par l'utilisateur.

### 3.1.10 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.

### 3.1.11 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". L'authentification de l'utilisateur se fait à l'aide d'un token d'accès temporaire régénéré à chaque nouvelle connexion de la part du client. 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, ou tout simplement consulter les informations sur d'autres joueurs (par exemple, leur biographie).

Dans ce diagramme on remarque, est représenté chaque connexion au serveur par l'intermédiaire de l'api. Chaque chemin ici présent fonctionne comme une uml

### 3.1.12 diagramme de classe Server side

## 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. Lorsqu'un joueur se déconnecte subitement du serveur, le serveur doit en avertir d'autres serveurs clients qui ont une partie en cours avec le joueur déconnecté.

### 3.2.3 Concurrency

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

Pour que l'envoi de données sensibles à travers internet ne puisse pas être intercepté, la communication devra se faire à l'aide d'un protocole de communication crypté. Un choix évident serait le TLS.

Une session pourra rester active à l'aide d'un access token. Ce token aura une date d'expiration. Le programme client gardera ce token dans la RAM. Ainsi, après le redémarrage du programme client, l'utilisateur devra se reconnecter avec son nom d'utilisateur et son mot de passe.

En ce qui concerne le programme serveur, celui-ci gardera un salted hash du mot de passe dans la base de données.

La récupération de mots de passe pourra se faire avec des questions secrètes ou éventuellement avec des mots de passes à usage unique générés à l'aide du standard TOTP (le client pourra, par exemple, les générer avec Google Authenticator ou l'alternative open-source, RaviOTP).

### 3.2.5 Fiabilité

Lorsque le client se déconnecte du serveur au cours d'une partie, l'adversaire doit en être averti. Après une minute d'attente, le joueur déconnecté est déclaré perdant suite à un abandon.

### 3.2.6 Internationalisation et Localisation

Dans le contexte de ce projet d'année, le programme ne sera pas internationalisé. Les timestamps (date des messages, création de parties, ...) seront enregistrés en timestamp unix. Ainsi, les clients se trouvant dans un fuseau horaire différent pourront voir ces timestamps en heure locale

### 3.2.7 Légalité

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

## 4 Design et fonctionnement du système

### 4.1 Jeu

Voir Annexe I

Diagramme de séquence expliquant les interaction client-serveur lorsqu'un joueur joue son tour

## 5 Images d'illustration

## 6 Annexes

### A Style de programmation

Dans le cadre de ce projet, le style de programmation défini par Google :

<https://google.github.io/styleguide/cppguide.html>

voir [Google Naming conventions](#)

doit être utilisé.

En résumé,

- L'indentation est de 2 espaces,
- Les noms des fichiers en minuscule avec des “.” et des “\_”
- Les noms des classes, des structs et des méthodes sont en CamelCase et commencent par une majuscule
- Les noms des variables sont en minuscules et séparés par des “\_”
- Les attributs privés terminent par un “\_”
- les constantes sont en CamelCase et commencent par un k minuscule (ex : kRefreshRate)

### B 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>

### C Librairies

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

- ncurses
- Algorithm
- boost

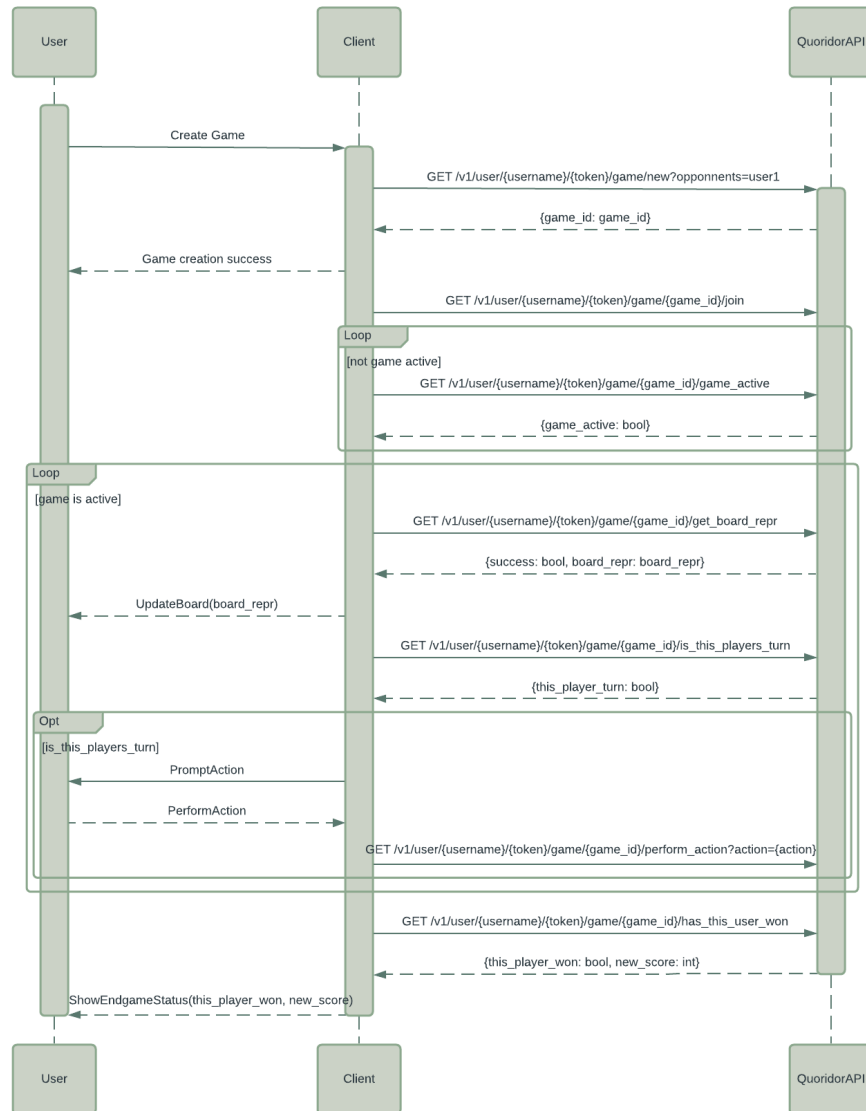
### D Protocole

### E 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.

**Diagramme de séquence  
représentant la  
création d'un jeu**

Anton & Ismael  
December 11, 2021



## Quoridor API

Anton & Ismael | December 12, 2021

PSEUDO-UML To represent our REST API service

ex. call:

[https://ip:port/v1/user/myusername123f00b4567d48628b40fff/set\\_bio?new\\_bio>Hello%2C%20this%20is%20my%20bio%21%20%F0%9F%A5%B8](https://ip:port/v1/user/myusername123f00b4567d48628b40fff/set_bio?new_bio>Hello%2C%20this%20is%20my%20bio%21%20%F0%9F%A5%B8)

This diagram was inspired by the following article:  
[https://firstinfinity.wordpress.com/modeling\\_rest\\_web\\_services/](https://firstinfinity.wordpress.com/modeling_rest_web_services/)

