# Software Requirements Document : Tetris Royal

Tao Chau
Juliete Cornu-Besser
Quentin Bernard Bouissières
Jonas Schellekens
Ethan Van Ruyskenvelde
Lucas Verbeiren
Ernerst Malysz
Rafaou Gajewicz

13 décembre 2024

# Table des matières

# 1 Introduction

## 1.1 But du projet

Ce projet a pour objectif de concevoir une application de jeu intégrant diverses fonctionnalités multijoueur. Les règles des parties reprendront celles existantes dans *Tetris Royal*, une extension de la version normale du jeu *Tetris*. Chaque joueur possède une grille vide qu'il doit essayer de remplir en laissant le moins d'espace possible avec des *tetrominoes*, des formes géométriques qui peuvent être tournées et déplacées avant leur place finale. Les pièces tombent depuis le haut de la grille et plus la partie avance, plus elles tombent rapidement. Une ligne remplie de *tetrominoes* se supprime et fait gagner des points au joueur. La partie se termine quand le joueur ne sait plus placer de formes géométriques sur la grille car il y a des *tetrominoes* sur toute la hauteur de la grille. Son score dépendra du temps de la partie et des lignes supprimées.

Dans cette application, quatre modes sont proposés au joueur. Il existe la version *Endless* avec un seul joueur où le but est de savoir placer des pièces sur la grille le plus longtemps possible. Selon les combinaisons des pièces qui se suppriment, les points attribués seront différents.

Puis nous implémenterons les modes multijoueurs. Une nouvelle notion doit être introduite : "*une ligne de penalty*". Un joueur peut envoyer un penalty à un adversaire. Le receveur du penalty retrouvera toutes ses lignes poussées vers le haut pour laisser la place à une ou plusieurs rangées avec un bloc manquant en dessous, qui ne peuvent pas être supprimées par les combinaisons.

La version *Classic* et la version *Duel* comprennent respectivement des parties de trois à neuf, et de deux joueurs. Chaque participant a sa propre grille avec ses *tetrominoes* qui tombent ; celui qui complète une ou plusieurs lignes en même temps peut envoyer des lignes de penalty selon sa combinaison à un adversaire de son choix.

Le dernier mode, *Royal Competition*, comprend des penalties multiples et des bonus. Ce sont des parties de trois à neuf joueurs. Chaque individu aura sa propre barre d'énergie initialement nulle. Au fil de la partie, la barre se remplit selon les combinaisons de lignes supprimées du joueur. Une fois cette dernière atteignant une valeur, le joueur peut acheter des penalties pour les envoyer à un adversaire ou s'octroyer un bonus qui lui facilite la partie pendant une période donnée. Dans la liste étendue de penalties et de bonus, nous retrouvons un ralentissement de la chute des *tetrominoes* pour soi-même, augmenter cette vitesse pour un adversaire ou encore inverser les commandes d'action d'un adversaire.

En dehors des parties de jeux, le joueur peut se connecter ou créer un compte pour accéder au *Tetris Royal*. Cela lui permet de gérer sa liste d'amis et de discuter avec eux via une messagerie intégrée à l'application. Il peut également consulter le classement des autres joueurs dans le mode *Endless*. Pour terminer, il peut créer ou rejoindre une partie dans un mode multijoueur sélectionné. Les participants peuvent être invités en mode observateur ou en mode joueur.

## 1.2 Glossaire

— **Blackout** : Pénalité qui "éteint la lumière" chez un adversaire ( le joueur ne voit plus son tableau pendant un court moment)
— **Bonus** : Avantage obtenu par le joueur lorsque celui-ci dépense son énergie. Un bonus peut prendre plusieurs formes :
— **Effect** : terme regroupant les pénalités et les bonus utilisés dans le mode royal
— **InputLock** : Pénalité qui bloque les commandes d'un adversaire pour le prochain tetromino.
— **Lightning** : Pénalité qui envoie un eclair qui supprime des blocs dans une zone de 2 x 2 chez un adversaire.
— **Matchmaking** : Système qui permet de connecter plusieurs joueurs ensemble pour pouvoir créer et lancer une partie en ligne.
— **Mini Tetrominoes** : Avantage qui fait que les 2 prochains tetrominoes se transforment en bloc de 1 x 1.
— **Penalty** : Pénalité infligée à un joueur par un de ses adversaires. Le penalty se manifeste par l'ajout d'une ou de plusieurs lignes supplémentaires en bas de la grille du receveur, poussant les blocs existants vers le haut et rapprochant le joueur de la défaite. Cette ligne de pénalité peut être supprimé de la grille si le joueur arrive à compléter avec ses tetrominoes.
— **Slow Down** : Avantage qui réduit la vitesse de chute des tetrominoes sur ton plateau.
— **Speed Up** : Pénalité qui augmente la vitesse de chute des tetrominoes sur le plateau d'un adversaire.
— **Reverse Controls** : Pénalité qui inverse les commandes d'un adversaire pour les trois prochaines tetrominoes.
— **Tetromino** : Pièce de jeu composée de quatre blocs carré connectés entre eux de manière à former différentes formes géométriques, respectivement (Z, L, O, S, I, J, T).
— **Username** : Nom utilisé par un utilisateur et qui peut ne pas être son nom officiel.

## 1.3 Historique

| Numéro de version | Nom | Modifications | Date |
|---|---|---|---|
| 0.1 | Quentin Bernard Bouissières / Jonas Schellekens / Ethan Van Ruyskenvelde / Lucas Verbeiren | Besoins utilisateurs fonctionnels | 20/11/2024 |
| 0.2 | Juliette Cornu-Besser / Jonas Schellekens | Introduction + Glossaire + commentaires Besoin utilisateur | 25/11/2024 |
| 0.3 | Ethan Van Ruyskenvelde | Ajout Besoins système non fonctionnels | 28/11/2024 |
| 0.4 | Quentin Bernard Bouissières / Lucas Verbeiren | Diagrammes de classe Game et Connexion | 01/12/2024 |
| 0.5 | Quentin Bernard Bouissières | Diagrammes de séquence Connexion et Inscription | 30/11/2024 |
| 0.7 | Juliette Cornu / Quentin Bernard Bouissières | Diagramme de classe Tetris | 08/12/2024 |
| 0.8 | Ernest Malysz | Relecture du SRD | 10/12/2024 |
| 1.0 | | rendu première version du projet | 12/12/2024 |
| 1.2 | Ernest Malysz / Jonas Schellekens / Rafaou Gajewicz | commencement diagrammes de classes | 26/02/2025 |
| 1.6 | Juliette Cornu | terminaison diagrammes de classes | 11/03/2025 |
| 1.7 | Juliette Cornu | avancement des diagrammes de séquence | 12/03/2025 |
| 1.8 | Juliette Cornu | rajout des descriptions pour les diagrammes de séquence et de classes | 13/03/2025 |
| 2.0 | Juliette Cornu | finir les diagrammes de séquence et relecture | 14/03/2025 |
| 2.5 | Juliette Cornu | corrections des diagrammes de classe et de use case | 04/05/2025 |
| 3.0 | | version finale | 09/05/2025 |

# 2 Besoins utilisateurs : Fonctionnels

## 2.1 Écran de connexion

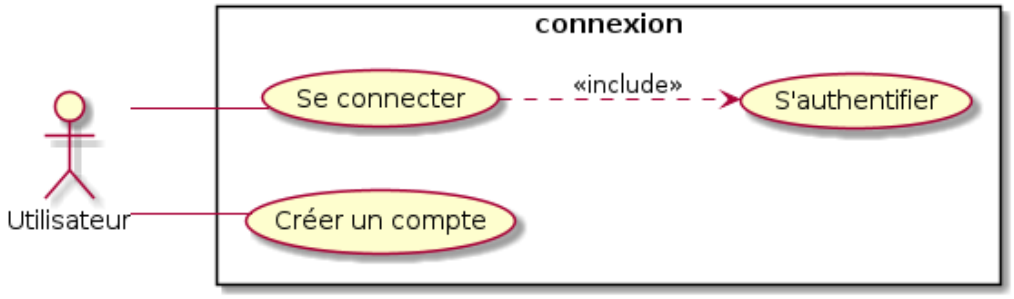



FIGURE 1 – Diagramme Use Case du Menu de Connexion

Le premier écran auquel l'utilisateur accède est un moyen de se connecter à la plateforme de jeu via un compte. Le moyen de s'authentifier est avec un compte existant et le mot de passe associé valide. Si l'utilisateur ne possède pas de compte existant, il peut en créer un nouveau avec un pseudo non existant sur la plateforme.

**Connexion**

- **Acteur** : Utilisateur
- **Use case** : Se connecter
- **Description** : Permet à l'utilisateur de se connecter au système.
- **Précondition** : L'utilisateur possède déjà un compte.
- **Postcondition** : L'utilisateur est connecté au système.
- **Cas exceptionnels** : Mauvais identifiant ou mot de passe, échec de la connexion réseau.

**Créer un compte**

- **Acteur** : Utilisateur
- **Use case** : Créer un compte
- **Description** : Permet à un utilisateur de s'inscrire et de se créer un compte.
- **Précondition** : L'utilisateur n'a pas encore de compte.
- **Postcondition** : Un nouveau compte est créé et l'utilisateur est authentifié.
- **Cas exceptionnels** : L'utilisateur existe déjà, erreurs de validation de données.

**S'authentifier**

- **Acteur** : Utilisateur
- **Use case** : S'authentifier
- **Description** : Vérifie les informations d'identification fournies par l'utilisateur.
- **Précondition** : L'utilisateur a soumis ses identifiants.
- **Postcondition** : Authentification réussie, session utilisateur créée.
- **Cas exceptionnels** : Identifiants incorrects, serveur d'authentification indisponible.

## 2.2 Menu Principal

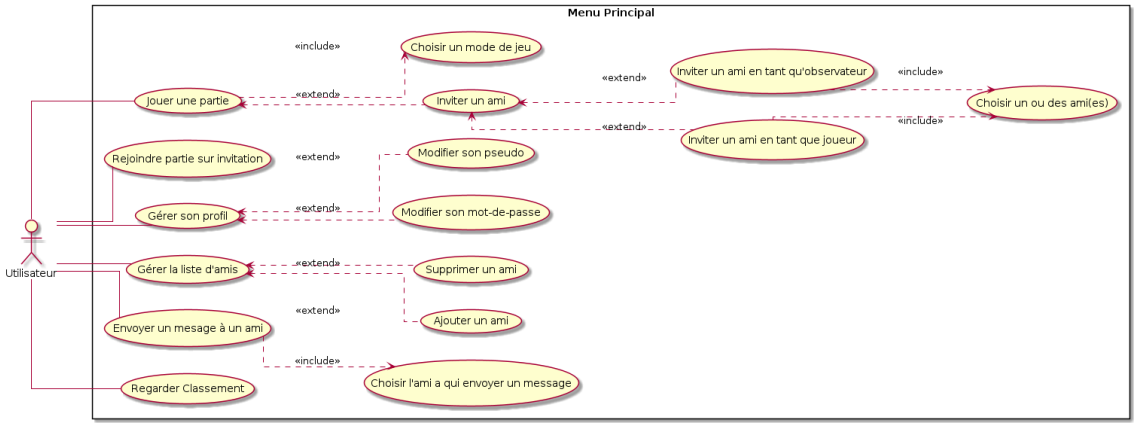



FIGURE 2 – Diagramme Use Case du Menu Principal

Une fois l'authentification validée, l'écran du menu principal est accessible. L'utilisateur peut lancer une partie, gérer son profil, communiquer avec ses amis par la messagerie interne, gérer sa liste d'amis et regarder le classement général du jeu.

**Jouer une partie**

- **Acteur** : Utilisateur
- **Use case** : Jouer une partie
- **Description** : Permet à l'utilisateur de lancer une partie.
- **Précondition** : L'utilisateur est authentifié.
- **Postcondition** : Une partie de jeu est démarrée.
- **Cas exceptionnels** : Problème de connexion ou erreurs de lancement.

**Rejoindre une partie existante**

- **Acteur** : Utilisateur
- **Use case** : Rejoindre partie existante
- **Description** : Permet à l'utilisateur de rejoindre une partie existante sur invitation d'un ami.
- **Précondition** : L'utilisateur a reçu une invitation.
- **Postcondition** : L'utilisateur rejoint la partie.
- **Cas exceptionnels** : Invitation expirée ou la partie a atteint le nombre maximum de personnes.

**Choisir mode de jeu**

- **Acteur** : Utilisateur
- **Use case** : Choisir mode de jeu
- **Description** : Permet à l'utilisateur de choisir le mode de jeu.
- **Précondition** : L'utilisateur veut créer une partie de jeu.
- **Postcondition** : Le mode de jeu est défini pour la partie.
- **Cas exceptionnels** : Aucun.

**Inviter un ami**

- **Acteur** : Utilisateur
- **Use case** : Inviter un ami

— **Description** : Permet à l'utilisateur d'inviter un de ses amis présents dans sa liste à une partie.
— **Précondition** : L'ami est connecté.
— **Postcondition** : L'invitation est envoyée à l'ami.
— **Cas exceptionnels** : L'ami est indisponible ou décline l'invitation.

**Gérer son profil**

— **Acteur** : Utilisateur
— **Use case** : Gérer son profil
— **Description** : Permet à l'utilisateur de modifier les informations de son profil.
— **Précondition** : L'utilisateur est authentifié.
— **Postcondition** : Les informations de profil sont mises à jour.
— **Cas exceptionnels** : Échec de la mise à jour des informations.

**Gérer la liste d'amis**

— **Acteur** : Utilisateur
— **Use case** : Gérer la liste d'amis
— **Description** : Permet à l'utilisateur de gérer sa liste d'amis.
— **Précondition** : L'utilisateur est connecté.
— **Postcondition** : La liste d'amis est mise à jour.
— **Cas exceptionnels** : Problèmes de synchronisation, ami introuvable.

**Envoyer un message à un ami**

— **Acteur** : Utilisateur
— **Use case** : Envoyer un message à un ami
— **Description** : Permet à l'utilisateur d'envoyer un message à un ami.
— **Précondition** : L'ami est dans la liste d'amis.
— **Postcondition** : Le message est envoyé avec succès.
— **Cas exceptionnels** : Ami déconnecté ou problème de réseau.

**Modifier son mot de passe**

— **Acteur** : Utilisateur
— **Use case** : Modifier son mot de passe
— **Description** : Permet à l'utilisateur de changer son mot de passe.
— **Précondition** : L'utilisateur est authentifié.
— **Postcondition** : Le mot de passe est modifié avec succès.
— **Cas exceptionnels** : Mot de passe actuel incorrect, validation échouée.

**Modifier son pseudo**

— **Acteur** : Utilisateur
— **Use case** : Modifier son pseudo
— **Description** : Permet à l'utilisateur de changer son pseudo.
— **Précondition** : L'utilisateur est authentifié.
— **Postcondition** : Le pseudo est modifié avec succès.
— **Cas exceptionnels** : Le pseudo est déjà pris, le changement de pseudo est invalidé.

**Ajouter un ami**

— **Acteur** : Utilisateur
— **Use case** : Ajouter un ami
— **Description** : Permet à l'utilisateur d'ajouter un autre utilisateur à sa liste d'amis.
— **Précondition** : Le potentiel nouveau ami existe et est accessible.
— **Postcondition** : L'utilisateur est ajouté en tant qu'ami dans sa nouvelle liste.

— **Cas exceptionnels** : Ami déjà présent dans la liste ou utilisateur introuvable.

**Supprimer un ami**

— **Acteur** : Utilisateur
— **Use case** : Supprimer un ami
— **Description** : Permet à l'utilisateur de retirer un ami de sa liste.
— **Précondition** : L'ami existe dans la liste d'amis.
— **Postcondition** : L'ami est supprimé de la liste.
— **Cas exceptionnels** : L'utilisateur à supprimer n'est pas dans la liste d'amis.

**Inviter un ami en tant que joueur**

— **Acteur** : Utilisateur
— **Use case** : Inviter un ami en tant que joueur
— **Description** : L'utilisateur a créé une partie et invite un joueur dans sa liste d'amis dans la partie.
— **Précondition** : L'ami est connecté.
— **Postcondition** : Invitation envoyée pour jouer.
— **Cas exceptionnels** : L'ami n'est pas disponible et l'invitation est rejetée.

**Inviter un ami en tant qu'observateur**

— **Acteur** : Utilisateur
— **Use case** : Inviter un ami en tant qu'observateur
— **Description** : L'utilisateur a créé une partie et invite un ami de sa liste d'amis à la partie en tant qu'observateur.
— **Précondition** : L'ami est connecté.
— **Postcondition** : Invitation envoyée pour observer.
— **Cas exceptionnels** : L'ami est indisponible et l'invitation est rejetée.

**Regarder Classement**

— **Acteur** : Utilisateur
— **Use case** : Regarder Classement
— **Description** : Permet à l'utilisateur de consulter les classements.
— **Précondition** : L'utilisateur est connecté.
— **Postcondition** : Le classement est affiché.
— **Cas exceptionnels** : Classement indisponible, problèmes de connexion.

## 2.3 Différents modes de jeux

### 2.3.1 En mode Endless

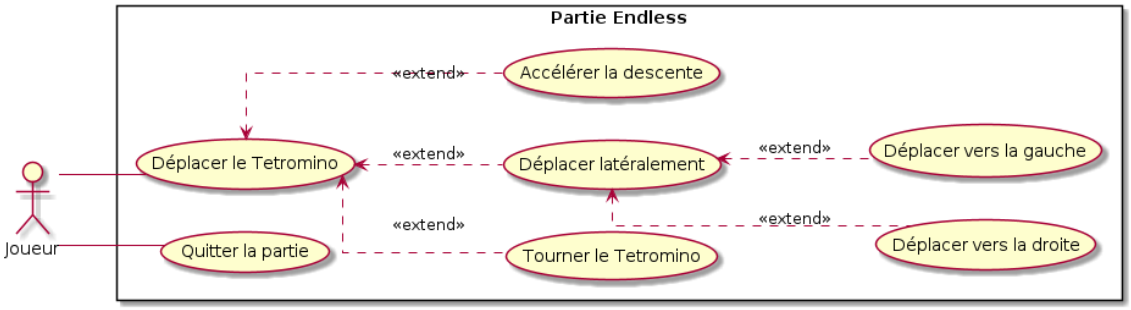



FIGURE 3 – Diagramme Use Case en mode Endless

**Déplacer le Tetromino**

- **Acteur** : Joueur
- **Use case** : Déplacer un Tetromino
- **Description** : Permet au joueur de déplacer un Tetromino sur le plateau.
- **Précondition** : Une partie est en cours et le joueur contrôle un Tetromino.
- **Postcondition** : Le Tetromino est déplacée dans une direction choisie.
- **Cas exceptionnels** : Limite du plateau atteinte, collision avec d'autres Tetromino.

**Tourner le Tetromino**

- **Acteur** : Joueur
- **Use case** : Tourner le Tetromino
- **Description** : Permet au joueur de tourner le Tetromino dans une direction donnée.
- **Précondition** : Une partie est en cours et un Tetromino est actif.
- **Postcondition** : Le Tetromino est tournée en conséquence.
- **Cas exceptionnels** : La rotation amène le Tetromino en dehors des limites.

**Quitter la partie**

- **Acteur** : Joueur
- **Use case** : Quitter la partie
- **Description** : Permet au joueur de quitter la partie en cours.
- **Précondition** : La partie est en cours.
- **Postcondition** : Le joueur quitte la partie et retourne au menu principal.
- **Cas exceptionnels** : Aucun.

### 2.3.2 En mode Duel

**Déplacer le Tetromino**

(identique au mode Endless)

**Tourner le Tetromino**

(identique au mode Endless)

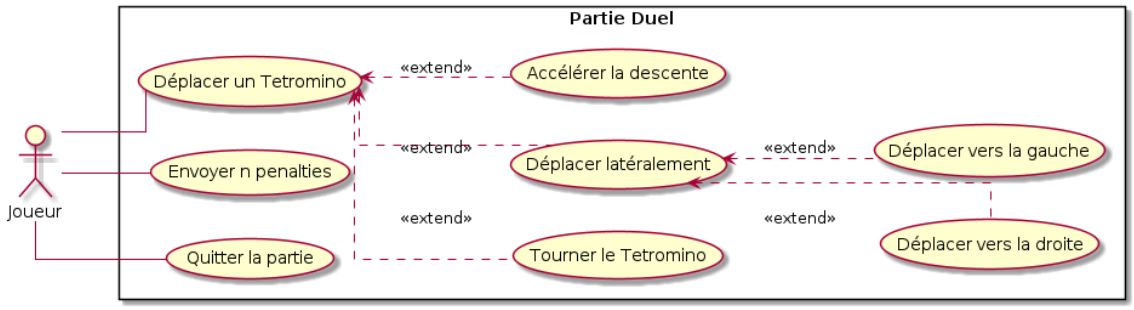



FIGURE 4 – Diagramme Use Case en mode Duel

**Envoyer n penalties**

- **Acteur** : Joueur
- **Use case** : Envoyer n penalties
- **Description** : Envoie un ou plusieurs penalties à l'adversaire.
- **Précondition** : Le joueur a complété au moins deux lignes dans sa grille.
- **Postcondition** : L'adversaire reçoit un ou plusieurs penalties.
- **Cas exceptionnels** : Échec de l'envoi du penalty, conditions du penalty non remplies.

**Quitter la partie**

- **Acteur** : Joueur
- **Use case** : Quitter la partie
- **Description** : Permet au joueur de quitter la partie en cours.
- **Précondition** : La partie est en cours.
- **Postcondition** : Le joueur quitte la partie et retourne au menu principal.
- **Cas exceptionnels** : Aucun.

### 2.3.3 En mode Classique

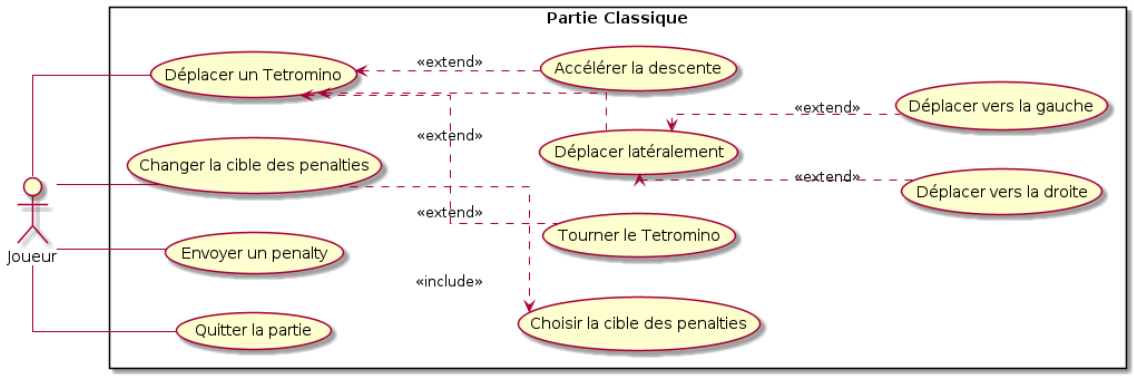



FIGURE 5 – Diagramme Use Case en mode Classique

**Déplacer le Tetromino**

(identique au mode Endless)

**Tourner le Tetromino**

(identique au mode Endless)

**Accélérer la descente**

(identique au mode Endless)

**Déplacer latéralement**

(identique au mode Endless)

**Changer la cible des penalties**

- **Acteur** : Joueur
- **Use case** : Changer la cible des penalties
- **Description** : Permet au joueur de changer l'adversaire recevant le penalty
- **Précondition** : La partie est en cours et le joueur peut envoyer un ou plusieurs penalties.
- **Postcondition** : Le joueur a sélectionné une nouvelle cible pour envoyer ses penalties.
- **Cas exceptionnels** : Il n'y a plus d'autres joueurs dans la partie.

**Choisir la cible des penalties**

- **Acteur** : Joueur
- **Use case** : Choisir la cible des penalties
- **Description** : Le joueur choisit parmis ses adversaires sa nouvelle cible pour les penalties.
- **Précondition** : La partie est en cours et le joueur peut envoyer un penalty.
- **Postcondition** : Le joueur change de cible et envoie un ou plusieurs penalties à l'adversaire nouvellement désigné.
- **Cas exceptionnels** : Il n'y a plus d'autres joueurs dans la partie.

**Envoyer n penalties**

(identique au mode Duel)

**Quitter la partie**

(identique au mode Duel)

### 2.3.4 En mode Royal Competition

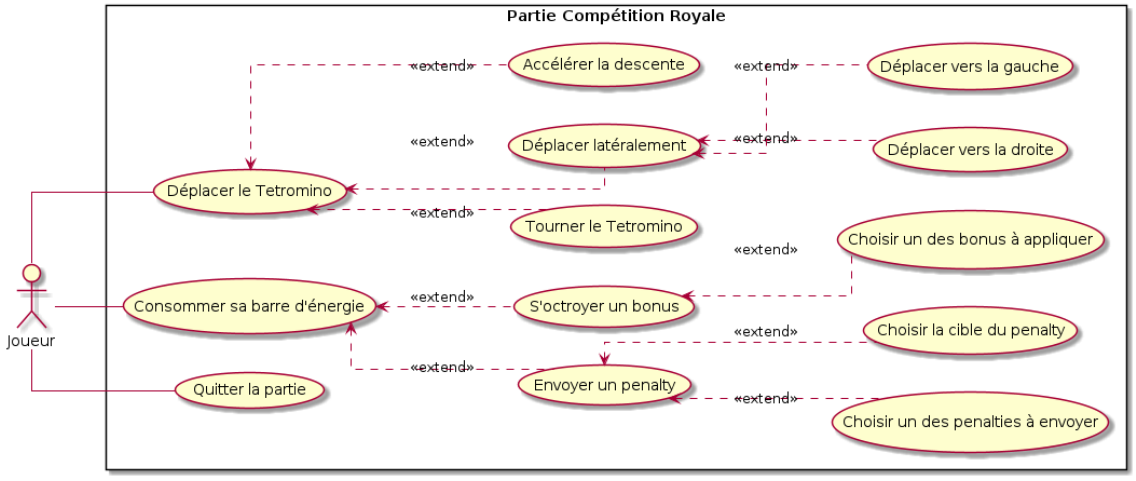



FIGURE 6 – Diagramme Use Case en mode Royal Competition

**Déplacer le Tetromino**

(identique au mode Endless)

**Tourner le Tetromino**

(identique au mode Endless)

**Déplacer latéralement**

(identique au mode Endless)

**Accélérer la descente**

(identique au mode Endless)

**Consommer sa barre d'énergie**

- **Acteur** : Joueur
- **Use case** : Consommer sa barre d'énergie
- **Description** : Le joueur a accumulé assez d'énergie dans sa barre avec des lignes complétées dans sa grille et consomme une partie de son énergie.
- **Précondition** : La partie est en cours et le joueur a accumulé assez de lignes complétées.
- **Postcondition** : Le joueur consomme sa barre d'énergie parmis les différents actions proposées.
- **Cas exceptionnels** : Il n'y a plus d'autres joueurs dans la partie.

**S'octroyer un bonus**

- **Acteur** : Joueur
- **Use case** : S'octroyer un bonus
- **Description** : Le joueur applique un bonus sur son jeu.
- **Précondition** : La partie est en cours et le joueur peut consommer sa barre d'énergie.
- **Postcondition** : Le joueur reçoit le bonus applicable durant un temps donné.
- **Cas exceptionnels** : Il n'y a plus d'autres joueurs dans la partie.

**Envoyer un penalty**

- **Acteur** : Joueur
- **Use case** : Envoyer un penalty
- **Description** : Le joueur envoie un penalty à l'adversaire choisi.
- **Précondition** : La partie est en cours et le joueur peut envoyer un penalty à un adversaire.
- **Postcondition** : Le joueur envoie le penalty à l'adversaire choisi.
- **Cas exceptionnels** : Il n'y a plus d'autres joueurs dans la partie.

**Choisir un des penalties à envoyer**

- **Acteur** : Joueur
- **Use case** : choisir un des penalties à envoyer
- **Description** : Le joueur a plusieurs options de penalties à envoyer à l'adversaire choisi.
- **Précondition** : La partie est en cours et le joueur peut consommer sa barre d'énergie.
- **Postcondition** : Le joueur envoie un des penalties proposés à l'adversaire visé.
- **Cas exceptionnels** : Il n'y a plus d'autres joueurs dans la partie.

**Choisir la cible du penalties**

(identique au mode Classique)

**Quitter la partie**

(identique au mode Duel)

# 3 Besoin système : Fonctionnels

## 3.1 Connexion

Quand l'utilisateur ouvre l'application, il doit avoir un compte valide pour s'authentifier. Il doit fournir le nom du compte et le mot de passe.
Ses entrées sur les deux champs sont envoyées au serveur de l'application. Les données sont vérifiées en recherchant dans la base de données un profil correspondant. Si un compte est trouvé avec le mot de passe correctement entré, le serveur connecte l'utilisateur au compte associé. Mais s'il ne trouve pas dans la base de données, l'utilisateur est notifié d'un message d'erreur du serveur.

## 3.2 Gestion des comptes

### 3.2.1 Création d'un compte

Quand un utilisateur crée un compte, il doit fournir un nom de compte et un mot de passe. Chaque entrée saisie a une taille minimale, maximale et une restriction sur les caractères autorisés. Le nom utilisé doit être unique pour chaque compte.
Le serveur vérifie qu'il n'existe pas déjà une autre entrée possédant le même nom.

### 3.2.2 Suppression d'un compte

L'option de suppression du compte doit être possible sur le système. Quand l'utilisateur fait une telle demande en confirmant avec son mot de passe associé, le programme client le questionne une dernière fois avant d'envoyer la requête au serveur. Le serveur supprime l'entrée sur la base de données du compte en question et l'utilisateur est déconnecté de la session.

## 3.3 Consulter le classement

Quand l'utilisateur est connecté et en dehors d'une partie de jeu, il peut consulter le classement général des joueurs de la plateforme pour le mode de jeu *Endless*. Le serveur l'actualise à chaque fin de partie et l'envoie sous forme de requête à chaque utilisateur concernant le classement.

## 3.4 Gestion de la partie

De la création à la fin de la partie, toutes les actions de l'utilisateur sont envoyées sous forme de requête au serveur et ce dernier applique l'action si elle est possible (ex. placer une pièce à tel endroit de la grille, envoyer un penalty à un adversaire, etc.). Le serveur permet de valider les actions et de vérifier s'il n'y a pas de triche en cours. La partie modifiée par le serveur avec la nouvelle action intégrée ou non sera récupérée et affichée à l'utilisateur.

## 3.5 Gestion des amis

Quand un utilisateur veut rajouter un autre utilisateur existant sur la plateforme à sa liste d'amis, le programme client enverra une requête au serveur. Si cette entrée saisie se trouve dans la base de données de l'application, le serveur enverra une requête au programme client de l'autre utilisateur. Ce dernier recevra une requête d'une demande d'ami qu'il pourra accepter ou non.

### 3.5.1 Gestion du chat

L'utilisateur peut utiliser la messagerie interne de l'application quand il possède au minimum un ami dans sa liste. Quand un message est envoyé de l'utilisateur, il est envoyé sous forme de requête utilisateur au serveur. Le ou les destinataires du message sont transmis au programme client de ces derniers. L'opération s'effectue sous condition d'avoir une connexion stable au serveur.

## 3.6 Fin de la partie

### 3.6.1 Fin de la partie *Endless*

Lorsque le joueur ne peut plus placer un *tetromino* sur sa grille ou demande volontairement d'arrêter de jouer, la partie se termine. Le serveur modifie le meilleur score associé au compte de l'utilisateur s'il a battu son propre record. Le classement des joueurs de l'application est mis à jour s'il y a eu modifications.

### 3.6.2 Fin de la partie *Multijoueurs*

Dans une partie de *Tetris* à plusieurs joueurs, le gagnant est celui qui peut toujours placer des *tetrominoes* sur sa grille tandis que les autres joueurs n'ont plus d'espace pour en déposer un.

# 4 Besoins système : Non fonctionnels

## 4.1 Besoins système

### 4.1.1 Réseau

L'application demande une connexion internet avec le protocole TCP/IP pour connecter le client au serveur. Ce n'est pas requis quand les deux se trouvent sur la même machine ; cela n'est pas attendu dans le cadre d'utilisation de l'application.

### 4.1.2 Système d'exploitation

Les machines du client et du serveur doivent pouvoir s'exécuter sur un système d'exploitation `Linux`.

### 4.1.3 Accessibilité

Le client doit être connecté à internet pour accéder au serveur. Son utilisation est gratuite pour tout utilisateur qui s'authentifie à un compte valable. S'il n'en possède pas un existant, il peut facilement s'en créer un.

### 4.1.4 Performances

Afin de minimiser le nombre de requêtes du client au serveur, des conditions seront vérifiées par le client avant d'envoyer ces dernières au serveur.

*Exemple* : l'utilisateur a bien fait au minimum deux combinaisons de lignes avant de pouvoir envoyer un ou plusieurs penalties à un de ses adversaires.

### 4.1.5 Capacité

Plusieurs parties de maximum neuf joueurs peuvent se tenir en même temps sur le serveur. De plus, l'espace de stockage pour chaque partie doit être au maximum de quelques mégaoctets.

### 4.1.6 Sécurité

Pour accéder au jeu, l'utilisateur devra s'authentifier à l'aide d'un nom d'utilisateur et de son mot de passe associé. Notamment, le serveur devra vérifier l'application correcte de toutes les règles du jeu durant la partie afin d'éviter des triches de la part des utilisateurs.

### 4.1.7 Robustesse

S'il reste au moins trois joueurs et que l'un d'entre eux se déconnecte de la partie ou du serveur, le jeu doit pouvoir continuer fluidement sans interruption. Celui qui vient de quitter la partie se retrouve perdant et est placé à la bonne position du classement de cette dernière.

## 4.2 Besoins utilisateur

L'application doit proposer une interface graphique intuitive à utiliser pour le joueur. Cela comprend un choix d'apparence attrayante en gardant une orientation d'une implémentation d'un jeu.
De plus, pour permettre une meilleure expérience de l'application à l'utilisateur, le lancement d'une partie locale ou avec une connexion au serveur doit être fluide et stable. La même chose est attendue durant tout le jeu. Le temps de rafraîchissement de l'interface du client doit se faire dans l'ordre du dixième de seconde.

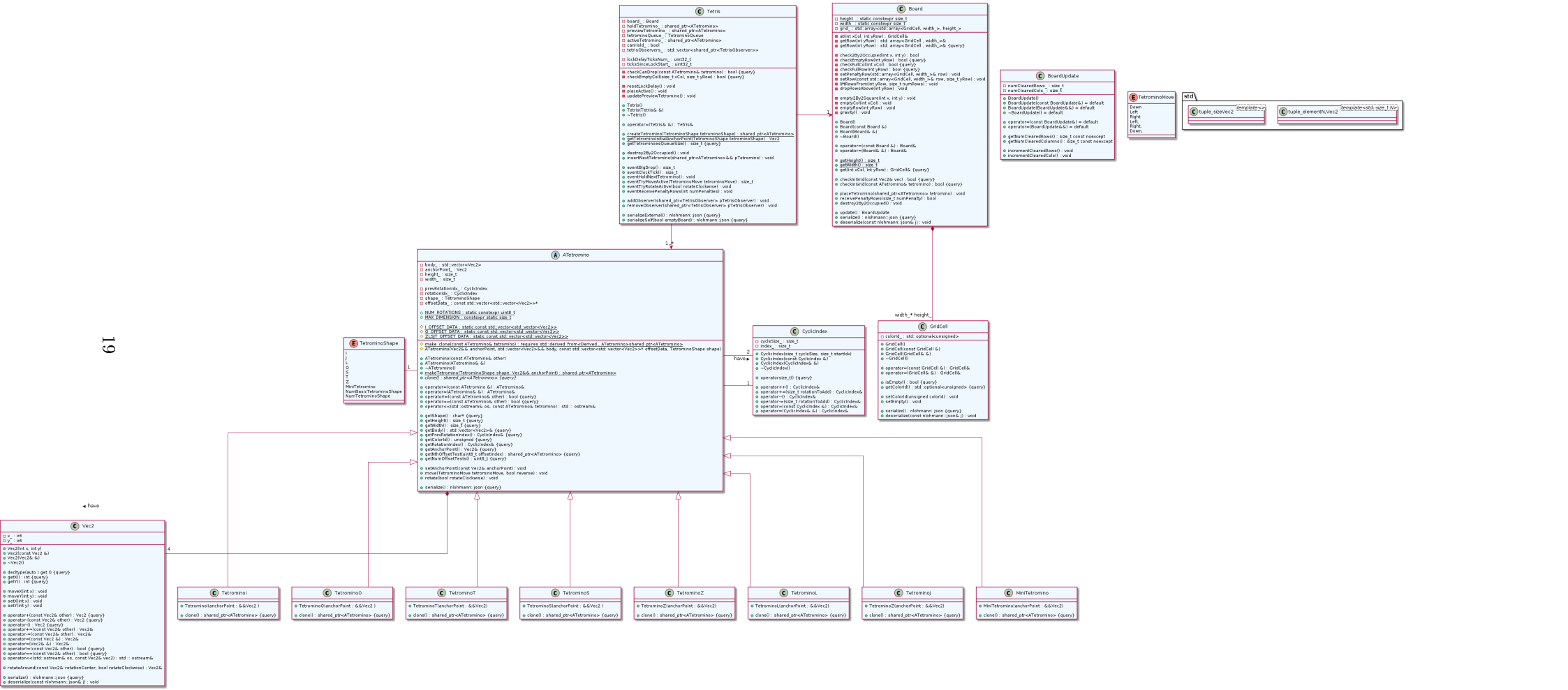
# 5 Architecture du système

## 5.1 Architecture du jeu

### 5.1.1 Classes du jeu

Au lancement d'une partie sur l'application, chaque joueur possède un objet de la classe Tetris. Dans la Board et la *queue-list* de tetrominoes, attributs de Tetris, le joueur placera les formes sur la Board. En reprenant l'algorithme pré-existant dans d'autres implémentations du jeu *Tetris*, le tetromino effectue une rotation similaire au fonctionnement de *Super Rotation System*. À chaque action du joueur envoyé pour changer la position du tetromino, Tetris calcule la nouvelle position selon celle renvoyée par le tetromino et puis vérifie la validité de cette position dans la Board.

Au lancement d'une partie avec plusieurs joueurs ou non, pour chaque joueur, nous avons la création d'un objet Tetris et PlayerState réunis dans un PlayerTetris. Pour l'ensemble de la partie, nous aurons un objet GameState qui contient toutes les informations nécessaires pour connaitre la situation de la partie en cours. Puis l'objet GameEngine permettra de valider ou non les actions demandées par les joueurs (*ex. changer le tetromino de rotation*).


Tetris
Board
BoardUpdate
TetrominoMove
std
tuple_sizeVec2
tuple_elementN,Vec2
ATetromino
TetrominoShape
CyclicIndex
GridCell
Vec2
Tetromino I
Tetromino O
Tetromino T
Tetromino S
Tetromino Z
Tetromino L
Tetromino J
MiniTetromino
1
1..*
width_* height_
have ▸
◂ have
4

### PlayerState

Le PlayerState permet d'associer le joueur, ses informations sur son avancemment durant la partie et celle de son profil.

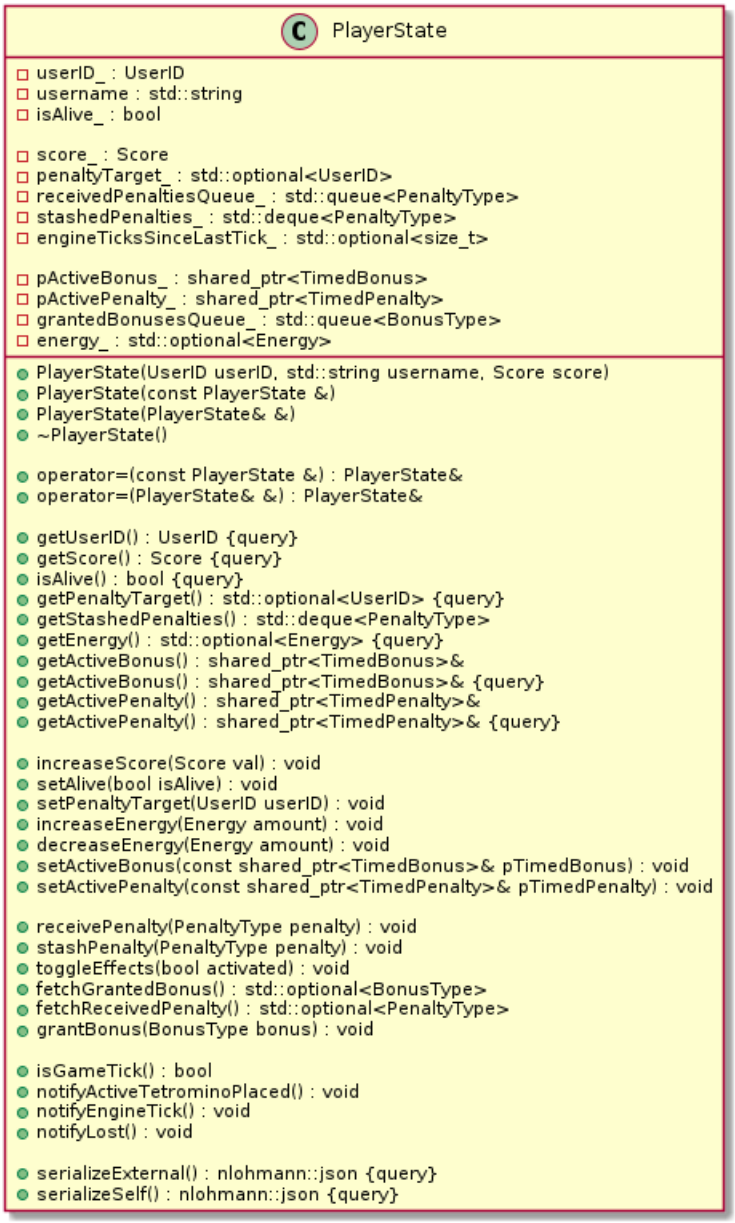



FIGURE 8 – Diagramme de classe de PlayerState

### PlayerTetris

Le PlayerTetris permet d'associer un PlayerState et un Tetris.

### GameState

Le GameState permet d'associer à une partie multijoueur ou non, un vecteur de PlayerState selon le nombre de participants. Il permet également d'accéder à un PlayerState selon l'identifiant du joueur pour pouvoir appliquer des actions dessus.

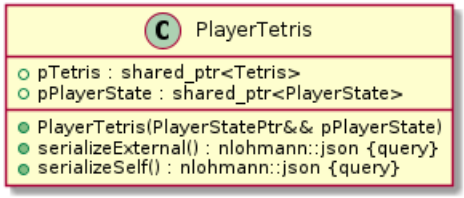



FIGURE 9 – Diagramme de classe de PlayerTetris

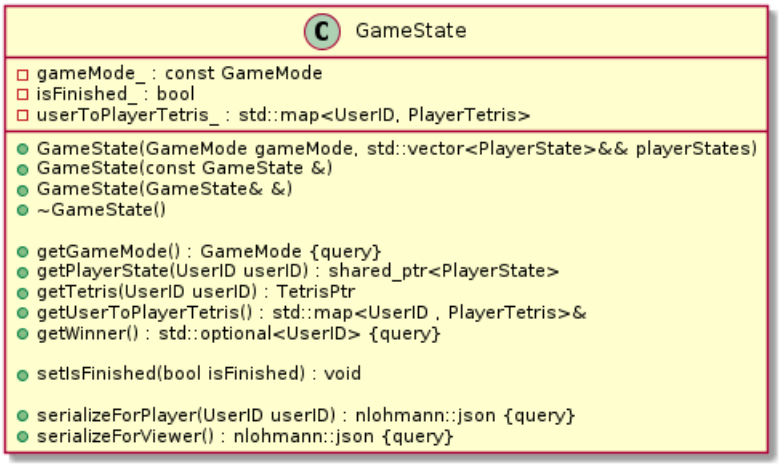



FIGURE 10 – Diagramme de classe de GameState

### GameEngine

Le GameEngine permet de changer l'état de GameState selon les méthodes appélées pour faire des actions sur la grille d'un joueur de la partie (*ex. changer la roation du tetromino active*) ou entre eux (*ex. envoyer une pénalité à un adversaire*).

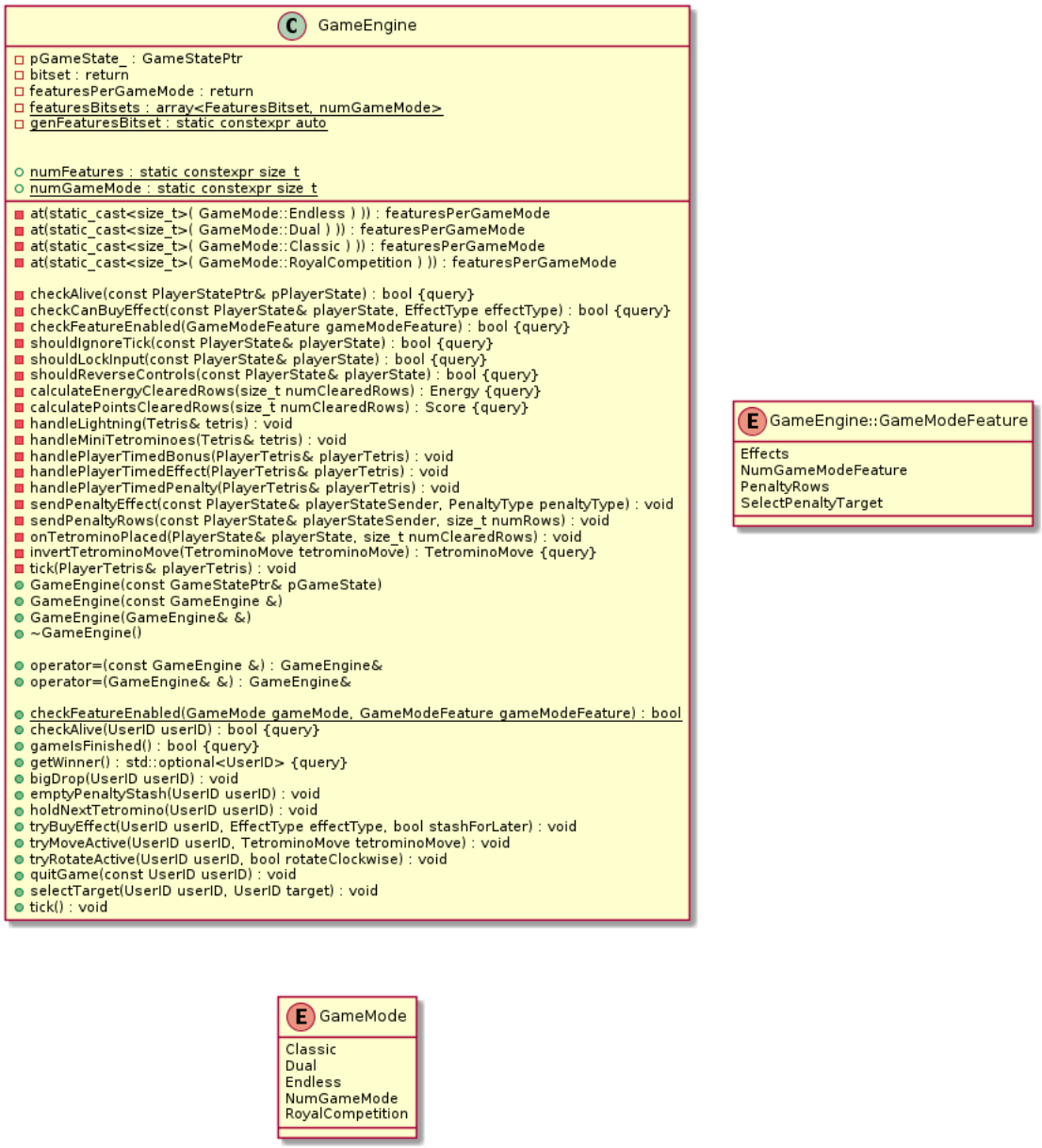



FIGURE 11 – Diagramme de classe de GameEngine

### 5.1.2 Partie multijoueur en mode royal

Au lancement d'une partie mulitjoueur dans le mode royal, pour chaque joueur, nous avons la création d'un objet Tetris et PlayerState. Contrairement aux autres modes, le PlayerState contient des informations supplémentaires tel que le niveau de la barre d'énergie et une queue de penalités à venir sur le joueur en question. Pour l'ensemble de la partie, GameEngine permettra de valider les actions demandées par les joueurs comprenant également les conditions pour acheter un effect spécifique.

#### Les effects

Pour l'implémentation des effects, nous avons distingué trois grandes familles, les effects qui s'écoulent sur une période de temps, les effects qui s'écoulent au bout d'un nombre de mouvements du joueur et les effects qui vont uniquement changer la grille d'un joueur une seule fois.

Nous avons une classe abstraite qui déclare des méthodes pour avoir le temps écoulé et le

nombre de mouvements du joueur. Ces méthodes sont concrétisées dans l'implémentation de deux classes héritantes, une pour les nombres d'actions restantes avant que l'effet se termine et une autre pour la durée restante. Puis les effects sont séparés en deux catégories, des bonus ou des penalités regroupées dans leur enumération de type respective.

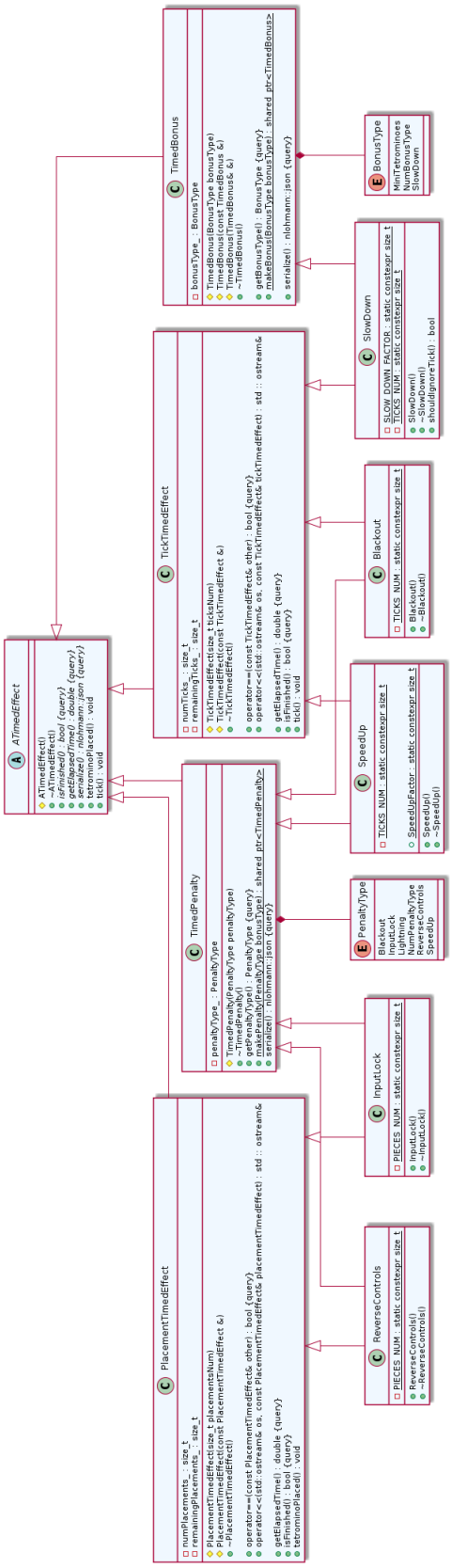



FIGURE 12 – Diagramme de classes des effects

### 5.1.3 Diagramme complet du jeu

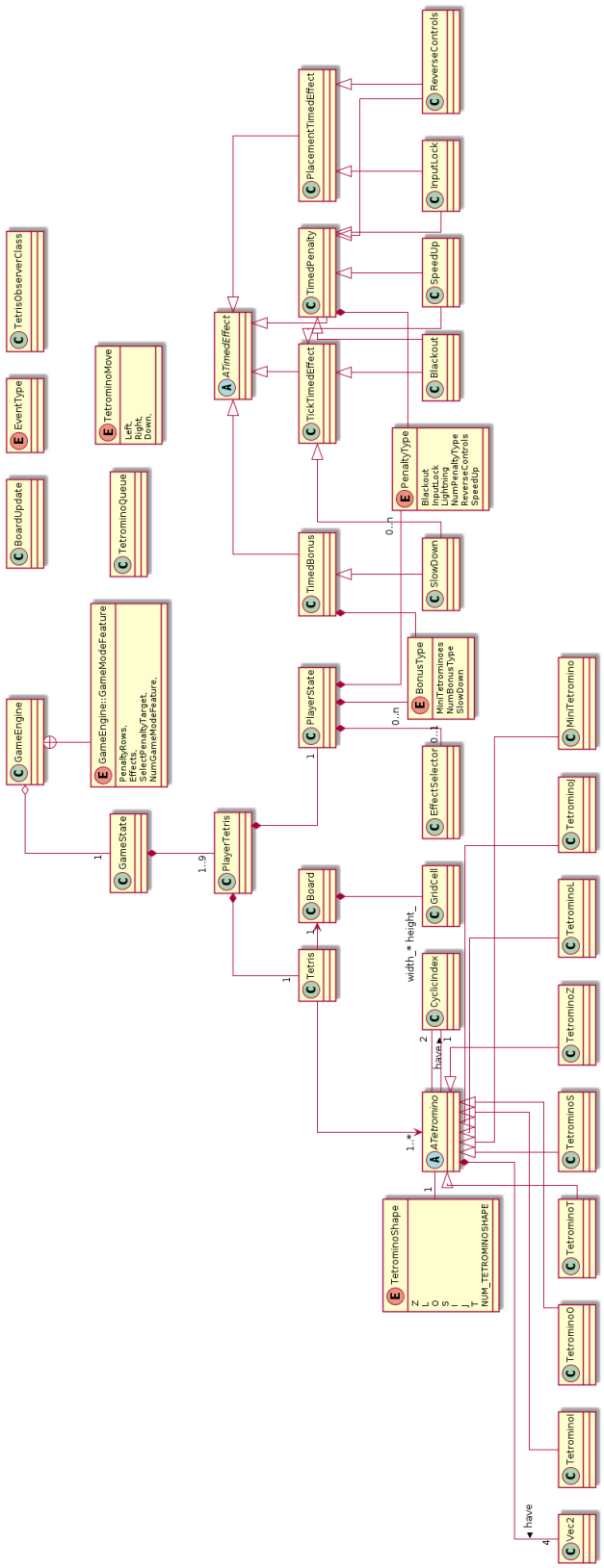



FIGURE 13 – Diagramme de classes simplifiées du jeu

## 5.2 Architecture du serveur

Le serveur est composé de quatre parties différentes. Chacune a un but précis pour permettre la communication entre différents utilisateurs.

### 5.2.1 La base de données

La base de données permet de stocker des informations sur les différents utilisateurs qui possèdent un compte dans l'application. Nous avons quatre différentes classes pour la base de données.

— AccountManager regroupe les informations du profil d'un utilisateur et permet de les modifier.
— FriendsManager permet de modifier ou de regarder la liste d'amis selon l'identifiant de l'utilisateur.
— MessagesManager gère le stockage de la messagerie de tous les utilisateurs et permet d'ajouter des messages à un fil de discussion existante ou qui va être créée sous certaines conditions.
— DatabaseManager permet de faire des redirections de requêtes pour les objets extérieurs qui vont faire une demande indirecte de changement de contenu de la base de données.

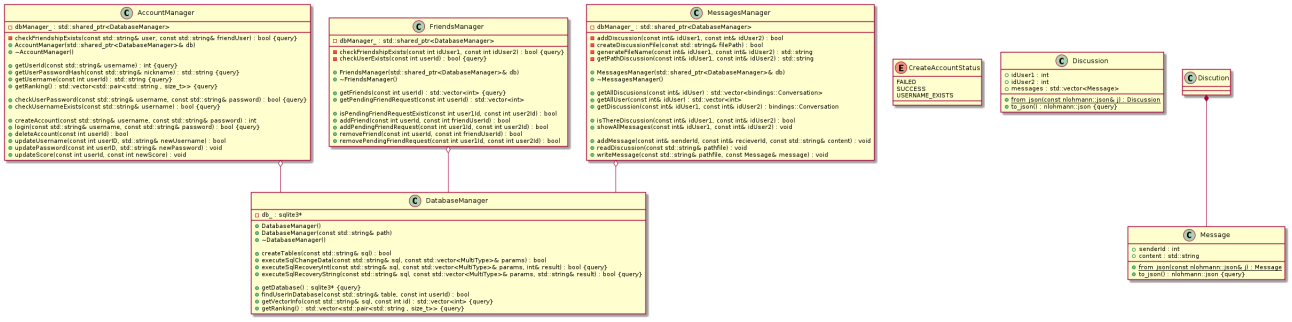



FIGURE 14 – Diagramme de classes de la base de données

### 5.2.2 Le network

#### Les bindings

Nous avons implémenté un espace de nommage *bindings*. Cela permet d'avoir un standard sur les paquets que le serveur peut recevoir et envoyer.

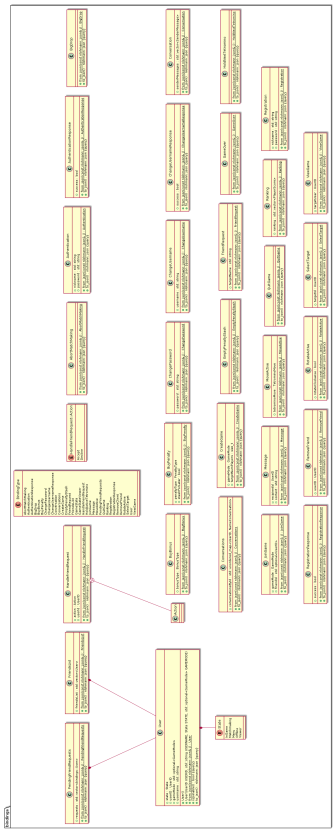


FIGURE 15 – L'espace de nommage de *bindings*

### Structure de Network

Pour chaque utilisateur qui se connecte au réseau, une instance de ClientLink est créée avec son socket. Au fil de l'état de l'utilisateur, le ClientManager le rajoute ou non dans le vecteur d'utilisateur authentifiés.

Nous pouvons voir le ClientLink comme une classe pour transmettre les paquets reçus d'un utilisateur en question puis ClientManager les traite pour que le serveur prenne en compte un événement.


FIGURE 16 – Diagramme de classes de Network

### AccountService

La classe AccountService permet de récuperer des informations dans la partie de la base de données qui s'occupe des comptes de l'application. En recevant des bindings précis, elle peut changer des champs dans la base de données ou vérifier les entrées données pour s'authentifier ou se connecter à un compte.

### SocialService

La classe SocialService permet de récupérer des informations dans la partie de la base de données concernant les amis d'un compte donnée ou les messages existants entre deux comptes différents. Nous pouvons également faire des opérations dans la base de données avec des méthodes de la classe. *ex. enlever un ami de sa liste d'amis*

## 5.2.3 Le Matchmaking

La classe Matchmaking est une classe que possède ClientManager. Elle est appelée quand ClientManager reçoit des paquets concernant une demande de création ou pour rejoindre une partie de jeu. Quand une demande de création de partie est dans le paquet, Matchmaking insère une nouvelle instance de GameCandidate dans un de ses attributs de gamesCandidate*Mode de jeu*. Pour GameCandidate, cela représente une partie en attente selon un mode de jeu sélectionné par le premier utilisateur, un nombre maximum d'utilisateurs, le nombre d'utilisateurs en train d'attendre.
Quand un nouveau utilisateur veut se rejoindre une possible partie en attente créée par un de ses amis ou aléatoirement, Matchmaking va chercher possiblement la partie en question et essayer de l'ajouter dans GameCandidate de cette dernière. Si le GameCandidate a atteint le quota de joueurs dans la partie en attente, Matchmaking appelle GamesManager dans une méthode privée pour lancer la partie en attente.

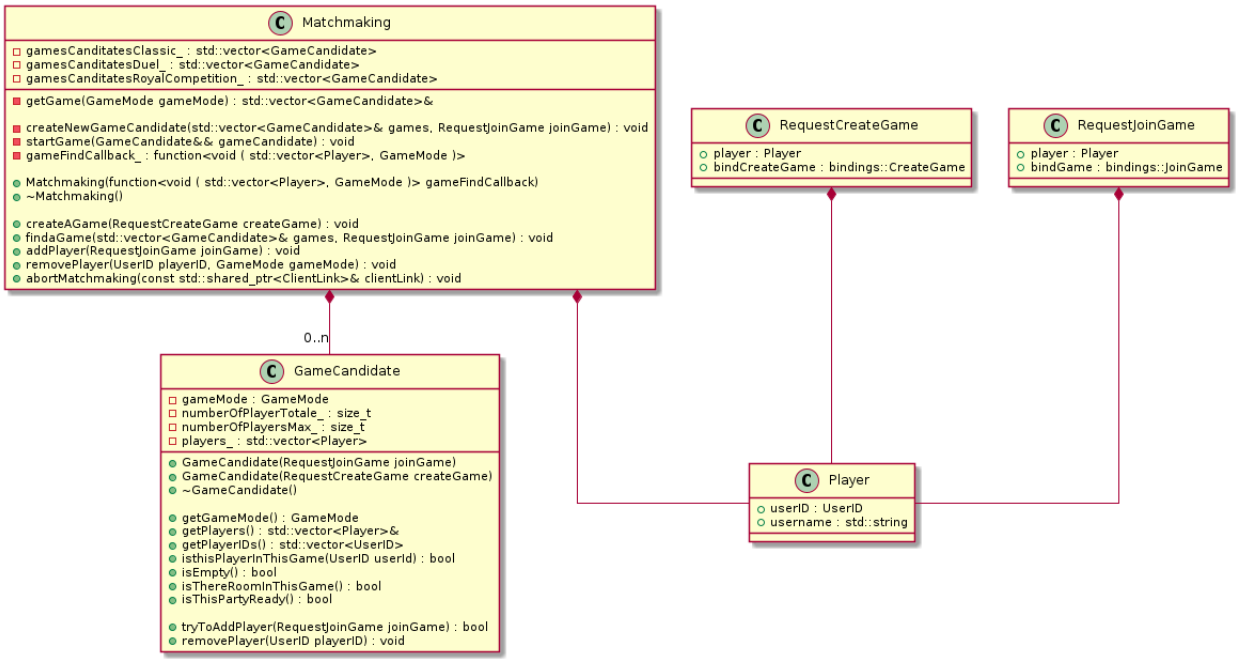



FIGURE 17 – Diagramme de classes de Matchmaking

### 5.2.4 GameServer

la classe GamesManager contient plusieurs maps pour permettre de trouver le gameID et le gameServer correspondant, de même pour le gameID et le thread, et le PlayerID et le gameID. GamesManager crée des instances de GameServer quand Matchmaking l'appelle selon un mode de jeu et une liste de PlayerID de la partie créée.
Quand ClientManager reçoit un paquet pour le jeu avec un playerID, le paquet est envoyé à GamesManager et puis à GameServer avec enqueueBinding. GameServer le traite à son tick interne correspondant pour que le gameEngine puisse prendre en compte le paquet reçu du serveur. Quand toutes les demandes en attente durant la période de temps accordée sont traitées, le GameServer envoie le GameState changé à ClientManager.


FIGURE 18 – Diagramme de classes de GameServer

### 5.2.5 MainServer

La classe TetrisMainServer permet de lancer les instances tels que ClientManager, la base de données et DatabaseManager lorsqu'on lance le serveur avecou non le choix d'un port spécifique.

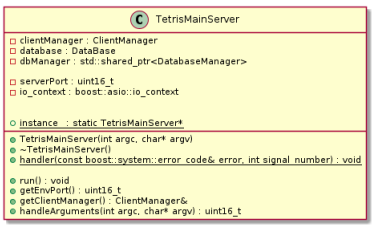



FIGURE 19 – Diagramme de classe de TetrisMainServer

### 5.2.6 Diagramme complet du serveur

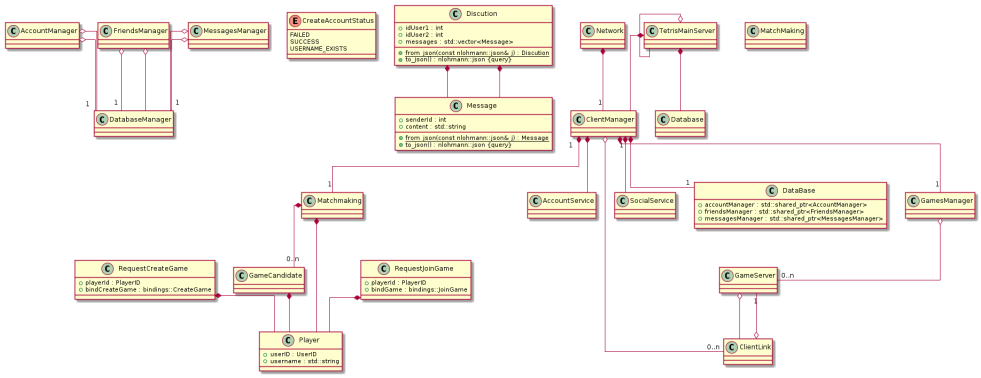



FIGURE 20 – Diagramme de classes simplifiées du Serveur

## 5.3 Architecture du client

L'architecture du client possède plusieurs parties qui permettent de communiquer avec le serveur et de plus afficher les informations sur l'écran de l'utilisateur. Il possède également tous les éléments nécessaires pour avoir localement la logique de Tetris dans tous les modes.

### 5.3.1 le Graphisme

La disposition sur l'écran des informations reçues du serveur est séparée par les classes pour le graphisme et une classe MainTui ou MainGui selon le choix de l'utilisateur lors du lancement de l'application.

Pour le GUI et le TUI, nous avons suivi une architecture similaire pour chaque avec en commun l'utilisation de la classe abstraite AbstractGameDisplay. Cette dernière permet d'avoir une implémentation existante pour certaines fonctions qui extraient les informations du GameState.


FIGURE 21 – Diagramme de classe de AbstractGameDisplay

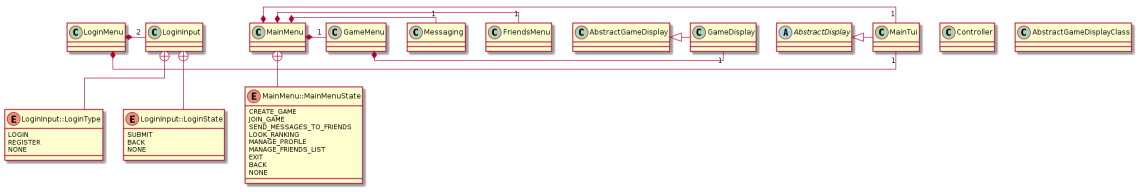



FIGURE 22 – Diagramme de classes des classes utilisées pour la partie graphique

### 5.3.2 le NetworkManager

La partie réseau du client est faite par la classe NetworkManager. Elle établit et met fin correctement la connexion avec le serveur. De plus, elle reçoit du serveur tous types de paquets et peut envoyer des paquets au serveur. Le traitement de ces paquets est fait dans une autre classe du client.

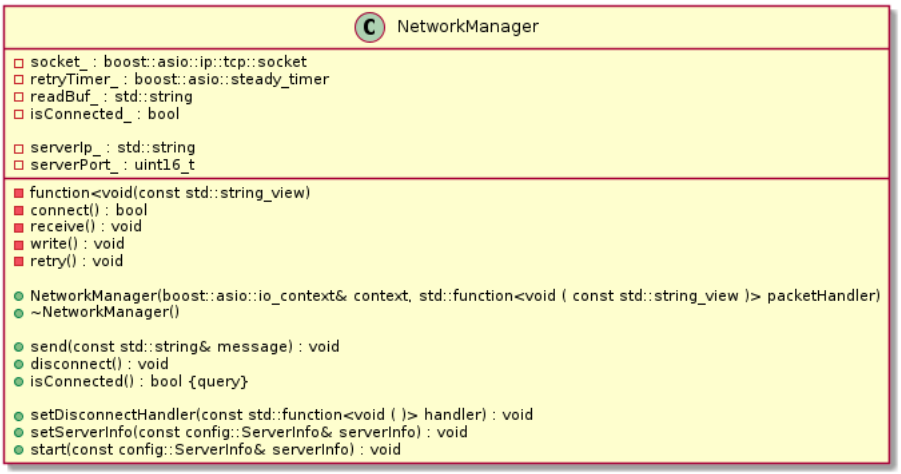



FIGURE 23 – Diagramme de classe de NetworkManager

### 5.3.3 le Controller

Le Controller réunit la partie réseau et l'affichage des informations à l'utilisateur. Il possède une instance de chaque partie et d'autres informations que nous aurions besoin d'afficher à l'utilisateur (*ex. la liste d'amis de l'utilisateur pour pouvoir lui afficher*).

Le Controller permet également de traiter les paquets reçus de NetworkManager pour changer les informations que possède l'utilisateur jusque lors. Quand le client veut envoyer une information au serveur, le paquet est créé dans Controller et puis, NetworkManager l'envoie au serveur.

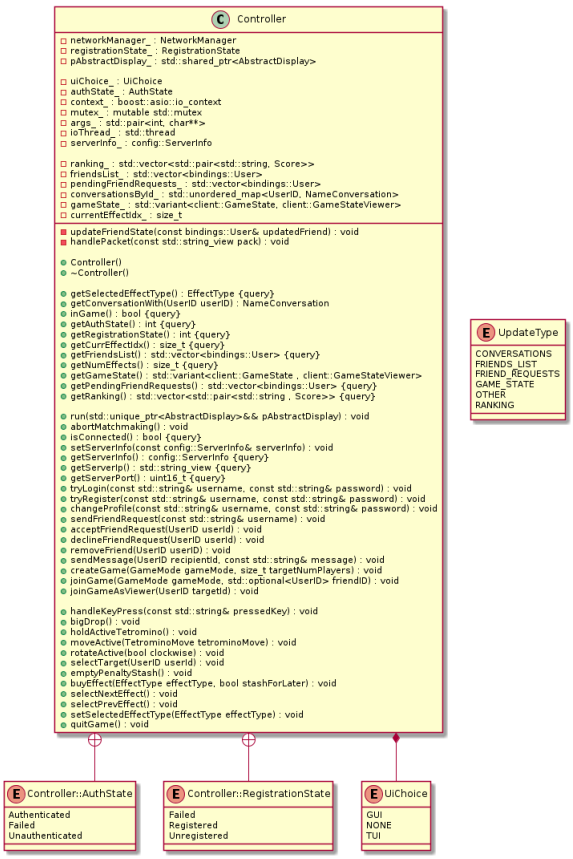



FIGURE 24 – Diagramme de classe de Network

### 5.3.4 Diagramme complet du client

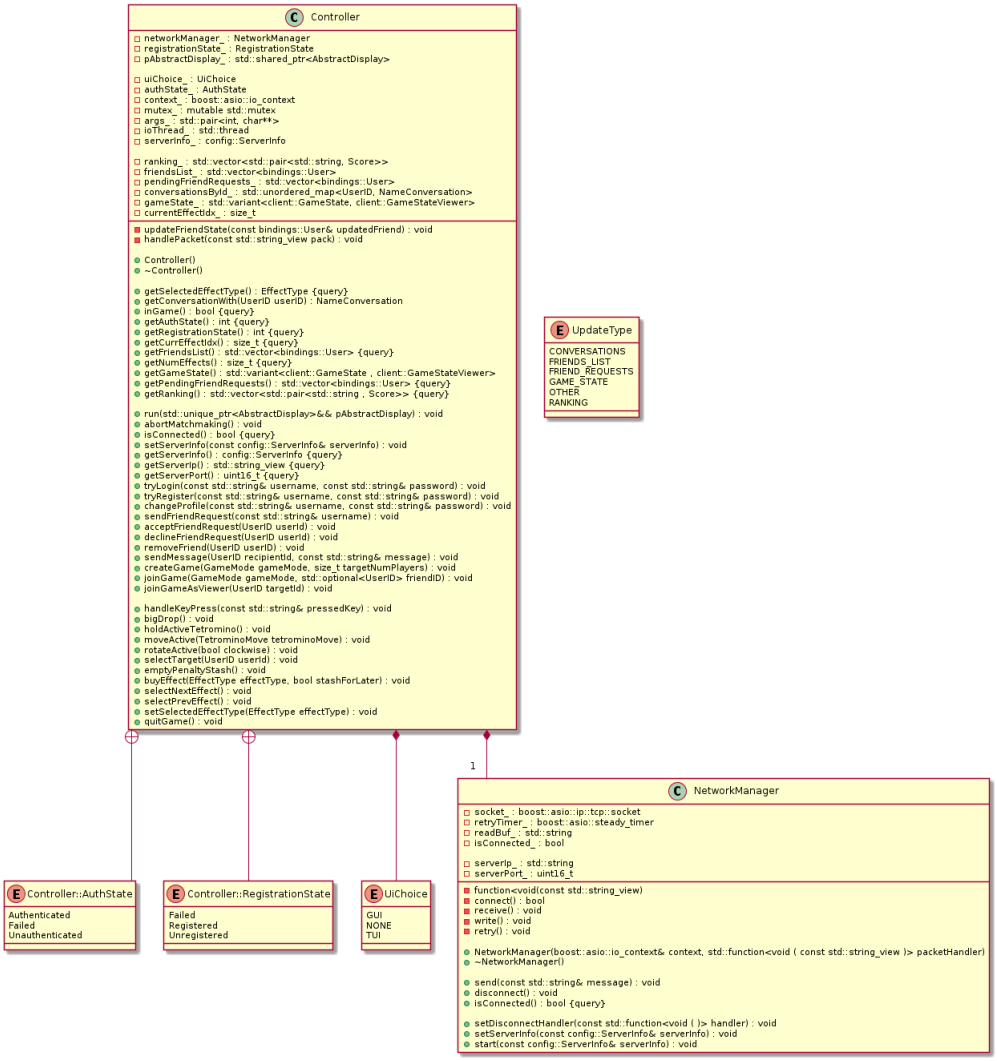



FIGURE 25 – Diagramme de classes du client

# 6 Fonctionnement du système

## 6.1 Création d'un compte

Lors du lancement de l'application, si l'utilisateur n'a pas de compte, il commence la procèdure d'inscription. Le Controller va se connecter au réseau du serveur. Network va créer une nouvelle instance de ClientLink. Ce nouvel objet le notifie à ClientManager qui appelle ensuite une méthode interne à elle-même. Ensuite l'utilisateur va entrer un username et un mot de passe sur l'interface et validera l'action. Le Controller enverra les données dans un paquet au ClientLink correspondant à l'utilisateur. ClientManager traite le paquet une fois signalé par ClientLink. La réponse au succès ou non de la nouvelle entrée dans la base de données est transmise de ClientManager à ClientLink qui écrira la réponse dans le socket. Le Controller traite la réponse et changera l'attribut nécessaire pour savoir le status de l'utilisateur.

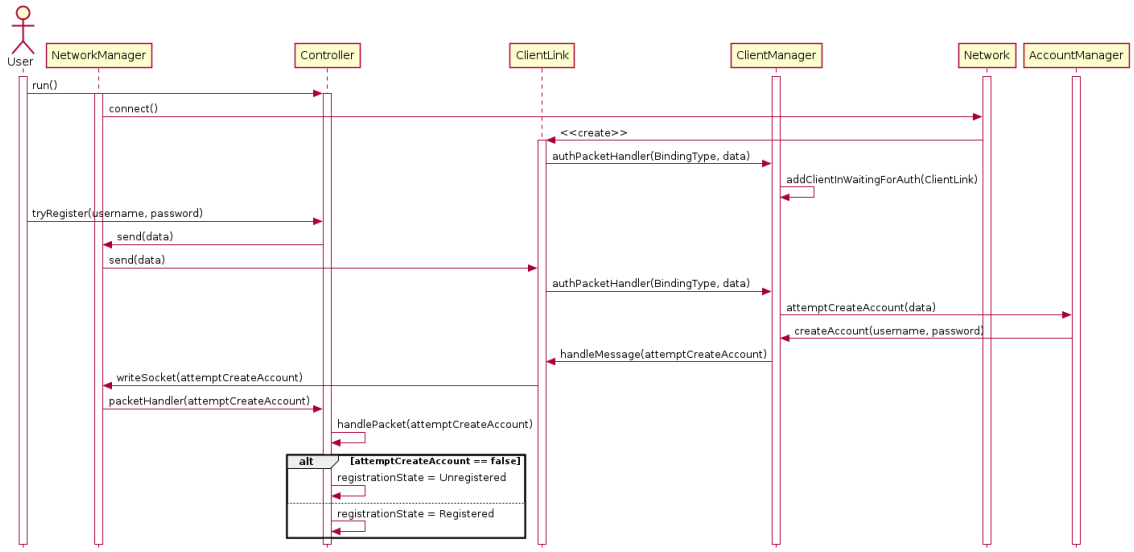



FIGURE 26 – Diagramme de séquence d'inscription

## 6.2 Connexion

Lors du lancement de l'application, si l'utilisateur possède déjà un compte, il commence la procédure d'authentification. Le Controller va se connecter au réseau du serveur. Network va créer une nouvelle instance de ClientLink. Ce nouvel objet le notifie à ClientManager qui appelle ensuite une méthode interne à elle-même. Ensuite l'utilisateur va entrer un username et un mot de passe sur l'interface et validera l'action. Le Controller enverra les données dans un paquet au ClientLink correspondant à l'utilisateur. ClientManager traite le paquet une fois signalé par ClientLink. ClientManager envoie un message à la base de données pour vérifier si le compte existe et si le mot de passe est correct avec le compte associé. La réponse sera récupérée par ClientManager. Ce dernier envoie un paquet à écrire pour le ClientLink de l'utilisateur. Le Controller traite la réponse et changera l'attribut nécessaire pour savoir le status de l'utilisateur.

Si l'authentification est un succès, le ClientManager va rajouter le ClientLink associé à l'utilisateur dans le vecteur d'utilisateurs connectés.

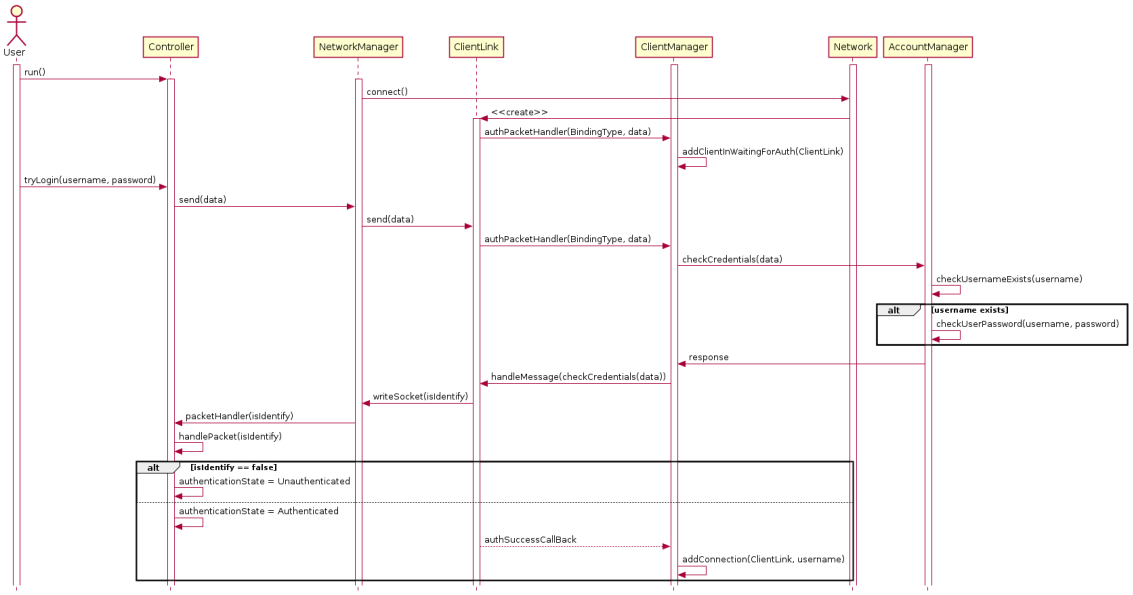



FIGURE 27 – Diagramme de séquence de connexion

## 6.3 Matchmaking côté serveur

Quand l'utilisateur veut créer une partie, ClientLink reçoit un paquet correspondant à sa demande. Ce dernier est transmis à ClientManager avec l'userID du client. ClientManager traite la demande et va envoyer un message à Matchmaking pour créer une partie. Une instance de GameCandidate sera créée par Matchmaking en conséquent.

Quand l'utilisateur veut rejoindre une partie existante, ClientLink reçoit un paquet correspondant à sa demande. Ce dernier est transmis à ClientManager avec l'userID du client. ClientManager traite sa demande et va envoyer un message à Matchmaking. Il va essayer de trouver selon le mode de jeu et la demande ou non de la présence d'un autre userID spécifique. Si la partie est trouvée, Matchmaking va essayer de rajouter l'utilisateur dans l'instance de GameCandidate. Si cela n'est pas possible, MatchMaking créera alors une nouvelle instance de GameCandidate.

Puis quand le nombre de joueurs maximum dans la partie sera atteint, Matchmaking enverra un message à GamesManager pour initier une nouvelle instance de GameServer.

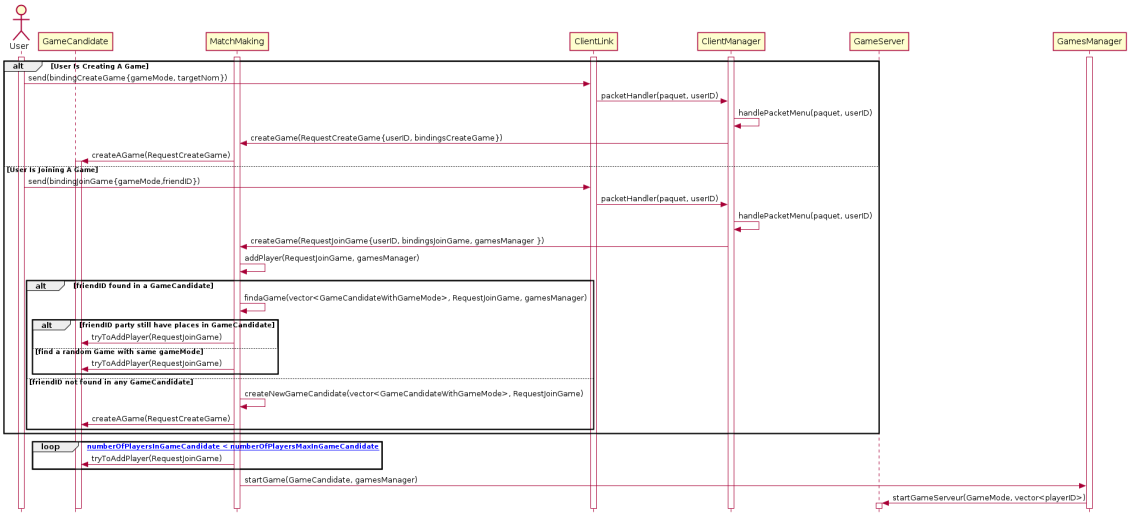



FIGURE 28 – Diagramme de séquence de Matchmaking côté serveur

## 6.4 Matchmaking côté client

Quand un utilisateur sélectionne dans le menu pour commencer une partie, il commence la procédure de Matchmaking. Le GameMenu s'affiche sur l'écran.

L'utilisateur peut créer une nouvelle partie de jeu. Il sélectionne le mode de jeu. Puis si ce n'est pas le mode Dual ou Endless, il doit remplir le champs concernant le nombre maximum de joueurs dans la nouvelle partie. Une fois que le GameMenu ait toutes ces informations, il envoit une requête à Controller, elle sera mise dans un paquet pour que NetworkManager l'écrive dans le socket du réseau.

L'utilisateur peut également choisir de rejoindre une partie existante avec le mode de jeu souhaité. Il sélectionne une partie créée par un de ses amis en ligne ou rejoint une partie existante aléatoirement. Une requête est envoyée de GameMenu à Controller pour qu'un paquet soit transféré au serveur via le NetworkManager.

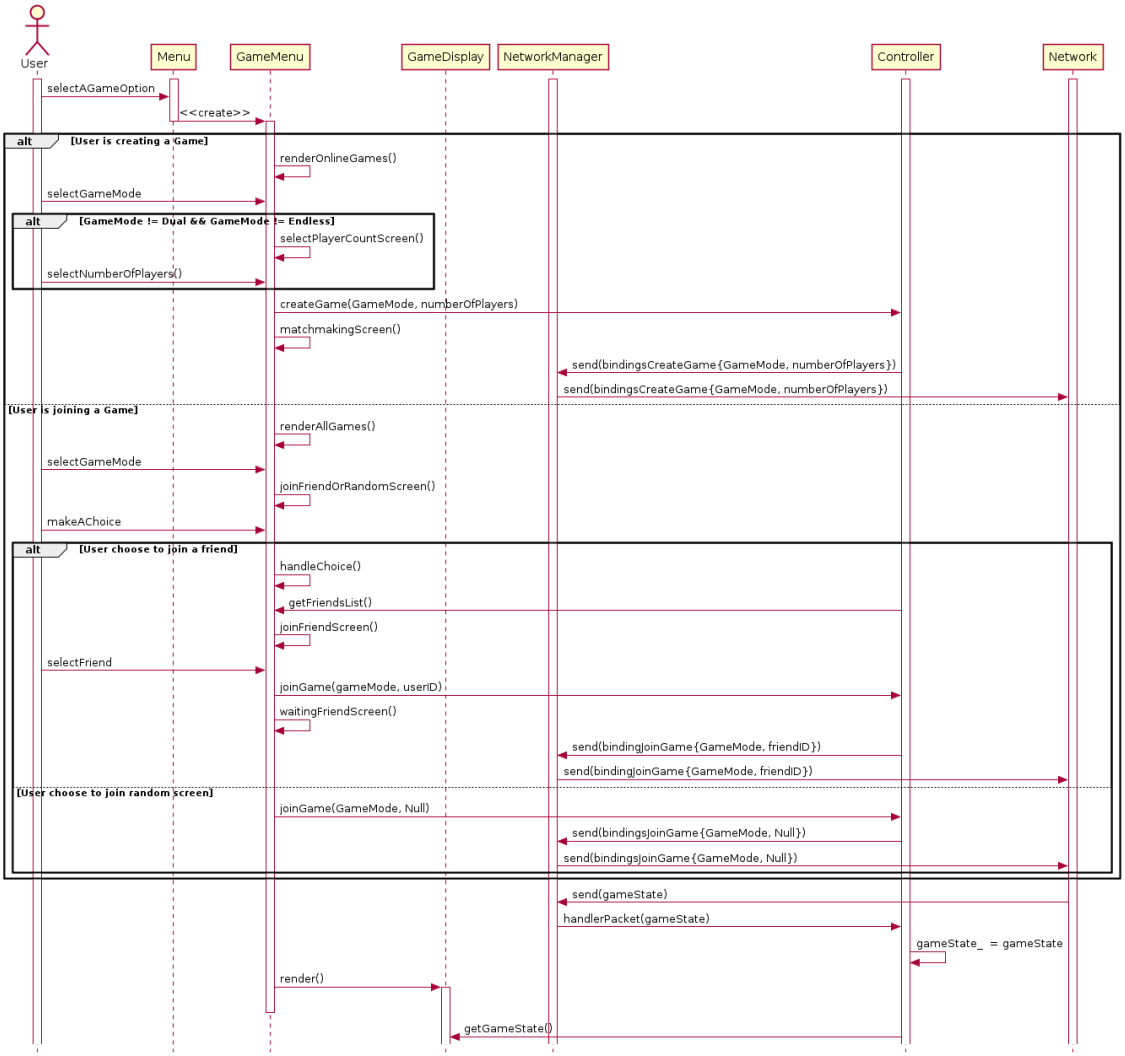



FIGURE 29 – Diagramme de séquence de Matchmaking côté client

## 6.5 Game côté serveur

Au début d'une partie, GamesManager commencera une instance de GameServer avec le mode de jeu indiqué et les playerID des utilisateurs présents. Durant la partie, le joueur enverra des paquets de jeu à son ClientLink correspondant. Ensuite, le paquet est reçue dans ClientManager puis transmis à GamesManager qui rajoutera à la queue de paquets que le GameServer doit encore

traitée. Le GameServer traite le paquet et l'incorpore dans son GameEngine. Une fois tous les paquets reçsu traités, le GameServer envera le GameState de la partie mis à jour à ClientManager. Puis il est passé à ClientLink pour être écrit dans le socket afin que l'utilisateur le reçoive. A la fin d'une partie, le GameServer le signalera à GamesManager puis se détruit.

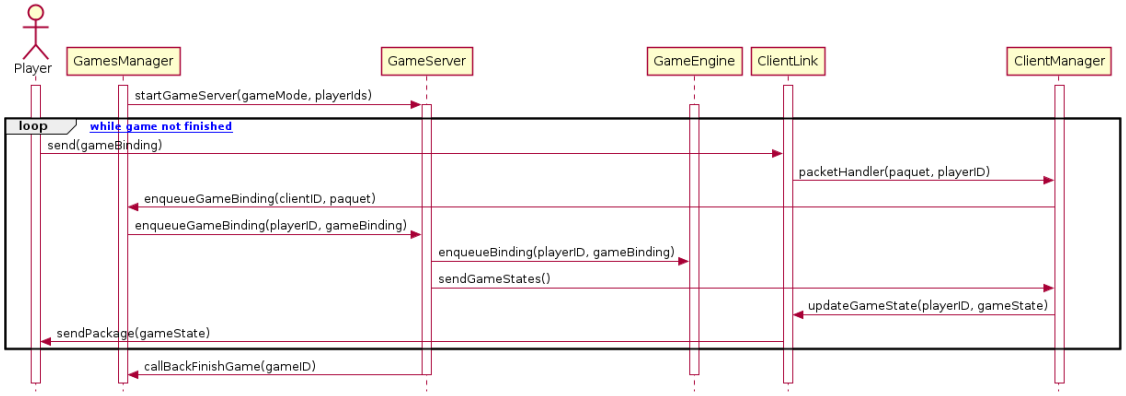



FIGURE 30 – Diagramme de séquence du jeu côté serveur

## 6.6 Game côté client

Au début d'une partie, l'objet GameDisplay est activé. Durant la partie, le joueur presse une touche de son clavier qui sera signalé à GameDisplay. Ce dernier enverra un message à Controller qui sera transformé en paquet et écrit dans le socket à NetworkManager. Le paquet sera reçu au serveur. Une fois le paquet traité, le serveur enverra à NetworkManager le nouveau gameState de la partie dans un paquet. Ce dernier sera traité dans le Controller qui forcera l'écran de l'utilisateur de se refaire.

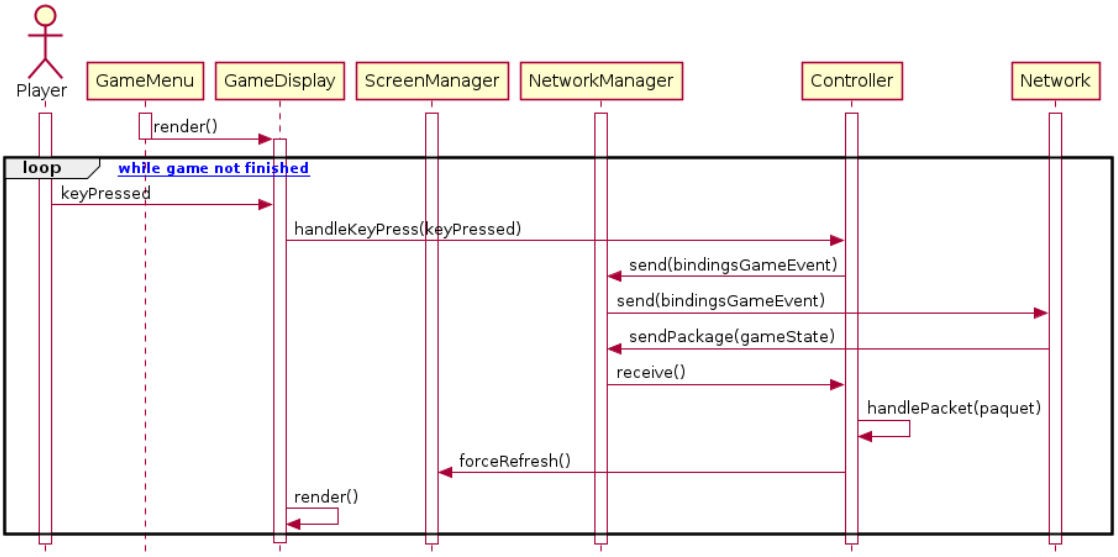



FIGURE 31 – Diagramme de séquence du jeu côté client

# 7 Annexes

## 7.1 Liens

https ://tetris.fandom.com/wiki/SRS