SOFTWARE REQUIREMENT DOCUMENT

Capitali$t
(INFO-F-209)

Loïc Blommaert - Hugo Charels - Yacoub Lahdo - Rémy Ryckeboer
Bao Tran - Hà Uyên Tran - Joachim Violon

21 avril 2023

Table des matières

1 Introduction **3**
1.1 But du projet . . . 3
1.2 Glossaire . . . 3
1.3 Historique . . . 4

2 Besoins utilisateur : Fonctionnels **6**
2.1 Écran de connexion . . . 6
2.1.1 Se connecter à un compte . . . 6
2.1.2 Créer un compte . . . 6
2.2 Écran d'accueil . . . 7
2.2.1 Crée une partie . . . 7
2.2.2 Rejoindre une partie . . . 7
2.2.3 Classement des joueurs . . . 8
2.2.4 Discuter avec d'autres joueurs . . . 8
2.2.5 Gérer sa liste d'amis . . . 8
2.2.6 Déconnexion . . . 8
2.3 Lobby . . . 9
2.3.1 Discuter avec d'autre joueurs . . . 9
2.3.2 Lancer une partie . . . 9
2.4 Partie de Capitali$t . . . 9
2.4.1 Démarrage de la partie . . . 9
2.4.2 Déroulement d'un tour . . . 9
2.4.3 Déroulement d'un tour 'normal' . . . 9
2.4.4 Déroulement d'un tour où le joueur est en prison . . . 10
2.4.5 Déplacement d'un joueur . . . 11
2.4.6 Achat et vente de propriétés . . . 11
2.4.7 Faillite d'un joueur . . . 11
2.4.8 Déroulement d'une vente aux enchères . . . 12
2.4.9 Discuter avec d'autres joueurs . . . 12
2.4.10 Achat et vente de bâtiment . . . 12
2.4.11 Piocher une carte . . . 12

3 Besoins utilisateur : Non fonctionnels **13**
3.1 Interactivité, agréabilité et dynamisme . . . 13
3.1.1 Clics et boutons . . . 13
3.1.2 Interface de l'utilisateur . . . 13
3.1.3 Langue du jeu . . . 14
3.1.4 Création de compte, pseudo et mot de passe . . . 14

4 Besoins système : Fonctionnels **15**
4.1 Écran de connexion . . . 15
4.1.1 Connexion à un compte . . . 15
4.1.2 Création d'un compte . . . 15
4.2 Écran d'accueil . . . 15
4.2.1 Création d'une partie . . . 15
4.2.2 Connexion à une partie . . . 15
4.2.3 Gestion du classement . . . 15
4.2.4 Gestion de la liste d'amis . . . 16

4.2.5 Gestion du chat . . . 16
4.2.6 Destruction d'un lobby . . . 16
4.2.7 Déconnexion . . . 16
4.3 Gestion d'une partie . . . 16
4.3.1 Gestion du chat . . . 16
4.3.2 Gestion du démarrage de la partie . . . 16
4.3.3 Déroulement d'un tour . . . 16
4.3.4 Gestion des déplacements . . . 16
4.3.5 Gestion de l'argent . . . 16
4.3.6 Gestion des propriétés . . . 17
4.3.7 Gestion des actions . . . 17
4.3.8 Gestion des cartes . . . 18

5 Besoins système : Non fonctionnels 18
5.1 Système d'exploitation et langage de programmation . . . 18
5.2 Réseau . . . 18
5.3 Disponibilité . . . 18
5.4 Performances . . . 18
5.5 Capacité d'une partie . . . 18
5.6 Sécurité . . . 18
5.7 Versions . . . 18

6 Design et fonctionnement du système 19
6.1 Système de connexion . . . 19
6.1.1 Création d'un compte . . . 19
6.1.2 Connexion à un compte . . . 20
6.2 Fonctionnement de l'infrastructure . . . 21
6.2.1 Envoie d'une requête au serveur par le client . . . 22
6.2.2 Réception d'une requête par le serveur . . . 22
6.3 Client . . . 24
6.3.1 Diagramme de classe : client . . . 24
6.4 Fonctionnement du jeu . . . 25
6.4.1 Diagramme de classes : Modèle du jeu . . . 25

1 Introduction

Ce document s'adresse à l'ensemble des membres du projet et a pour vocation de regrouper de manière explicite les critères et les besoins du client pour la conception du jeu Capitali$t.

1.1 But du projet

L'objectif du projet est de réaliser une version électronique du jeu Capitali$t. C'est un jeu de plateau (formé de 40 cases contiguës et constituant un cycle) dans lequel les joueurs, chacun leur tour, sont amenés à s'enrichir et faire tomber en faillite leurs adversaires. Chaque tour, les joueurs peuvent réaliser certaines actions. Celles-ci sont soit obligatoires comme lancer les dés pour avancer, payer le loyer d'une propriété ou piocher une carte, soit facultatif comme acheter des propriétés ou construire des bâtiments. Un capital de départ est alloué à chaque joueur et il existe plusieurs moyens de gagner ou perdre de l'argent durant la partie. Lorsqu'un joueur ne peut plus payer son dû, il est éliminé, et le dernier joueur en lice sera le vainqueur. Pareillement, si la durée fixée de la partie est écoulée, le joueur le plus riche est le vainqueur.
Il existe 3 types de cases achetables : les propriétés (regroupées par couleur), les gares et la compagnies de service public.

Deux modes de jeu seront disponibles :

- Classique : Il n'y a pas de limites de temps, la partie dure jusqu'au moment où il reste un seul joueur.
- Rapide : Il y a une limite de temps et certaines règles spécifiques permettant d'accélérer la partie.

En-dehors d'une partie, le joueur a la possibilité de créer un compte et de discuter avec d'autres joueurs possédant un compte.

1.2 Glossaire

- **ASCII art** : réalisation d'un genre d'image à l'aide de caractères spéciaux.
- **CLI** : Command Line Interface, le programme en ligne de commande.
- **CUI** : Command User Interface, le programme dans une version graphique mais qui possède toujours un système de commande.
- **Database** : une base de données, un ensemble de données structurées
- **Faire un double** : avoir les deux dés qui indiquent le même nombre
- **GameCode** : code de 4 chiffres généré à la création d'une partie de Capitali$t afin de servir d'identifiant de la partie pour que d'autres utilisateurs puissent la rejoindre.
- **GUI** : Graphical User Interface, le programme dans une version graphique.
- **Lobby** : salon d'attente dans lequel se retrouve un joueur ayant rejoint une partie avant que celle-ci ne soit lancée.
- **MAJ** : Mise À Jour
- **Pseudo** : Pseudonyme, nom choisi pour désigner un compte par l'utilisateur
- **Salon de jeu** : sous-ensemble du serveur gérant les parties des joueurs
- **Terrain** : cases du plateau achetables

1.3 Historique

Version	Modifications	Auteurs	Date
0.1	Diagrammes de classes : classe du jeu	Hugo Charels Rémy Ryckeboer Tran Hà Uyên	12/11/2022
0.2	Use case écran d'accueil et connexion	Rémy Ryckeboer	18/11/2022
0.3	Diagrammes de classes : classe du jeu	Hugo Charels	18/11/2022
0.4	Diagramme de séquences : chat	Loïc Blommaert Yacoub Lahdo	20/11/2022
0.5	Use case : tour d'un joueur	Joachim Violon	21/11/2022
0.6	Besoin utilisateur fonctionnels : Écran de connexion et Écran d'accueil MAJ	Rémy Ryckeboer	22/11/2022
0.7	Diagrammes de classes : classes serveur et client	Hugo Charels	23/11/2022
0.8	Besoins utilisateur fonctionnels + Besoins système	Bao Tran	25/11/2022
0.9	Diagramme de séquences : enregistrement et connexion de l'utilisateur	Loïc Blommaert Yacoub Lahdo	26/11/2022
0.10	Diagrammes de classe : classes du jeu, serveur, client, interface graphique	Hugo Charels	26/11/2022
0.11	Besoins systèmes fonctionnels	Bao Tran	27/11/2022
0.12	Design et fonctionnement du système et déroulement d'un tour	Hà Uyên Tran	27/11/2022
0.13	Use case : déroulement d'un tour MAJ	Joachim Violon	27/11/2022
0.14	Diagrammes de séquences : gestion de comptes et en partie	Loïc Blommaert Yacoub Lahdo	27/11/2022
0.15	Corrections et améliorations des textes	Loïc Blommaert Hà Uyên Tran Yacoub Lahdo	28/11/2022
1.1	Diagrammes de séquences : chat, dice	Loïc Blommaert Yacoub Lahdo	08/12/2022
1.2	Diagrammes de séquences : negociation, drawCard, dice	Loïc Blommaert Yacoub Lahdo	09/12/2022
1.3	Class diagramme : Connexion System Class diagramme : Welcome screen System	Hugo Charels Rémy Ryckeboer	12/12/2022
1.4	Activity diagramme : register, login	Loïc Blommaert Rémy Ryckeboer	14/12/2022
1.5	Use case : Écran de connexion MAJ Use case : Écran d'accueil MAJ Tableau des règles de validité	Rémy Ryckeboer	15/12/2022
1.6	Besoins système	Bao Tran	16/12/2022
1.7	Ajout des différents UseCase en rapport avec le Gameplay	Joachim Violon	18/12/2022
1.8	Rédaction du SRD	Loïc Blommaert Rémy Ryckeboer Tran Hà Uyên	18/12/2022
1.9	Ajout diagramme d'état pour le gameplay	Joachim Violon	18/12/2022
1.10	Écriture des textes du point 2.4 et sous-points	Joachim Violon	19/12/2022
1.11	Diagramme de classe	Rémy Ryckeboer	19/12/2022
1.12	Diagramme de séquences	Loïc Blommaert Yacoub Lahdo	19/12/2022

Version	Modifications	Auteurs	Date
1.13	Relecture intégrale du SRD, rédaction supplémentaire, correction	Bao Tran Joahchim Violon	19/12/2022
1.14	Diagramme de classe : modèle du jeu	Loïc Blommaert Hà Uyên Tran	19/12/2022
1.15	Diagrammes de classe : Connexion System Class diagramme : Welcome screen System	Hugo Charels Rémy Ryckeboer	19/12/2022
1.16	Diagramme de classe : Modèle du serveur	Hugo Charels	19/12/2022
1.17	Diagrammes de séquence	Yacoub Lahdo	19/12/2022
1.18	Sauvetage du SRD (rip notre dodo...)	Tout le monde	19/12/2022+30min
2.0	Correction des typos	Loïc Blommaert	12/02/2023
3.0	Refonte du SRD	Tout le monde	10/04/2023
3.1	Use case remis à jour par rapport au modification apporter sur le code	Rémy Ryckeboer	17/04/2023
3.2	Mise à jour des textes des points 2, 3	Yacoub Lahdo	17/04/2023
3.3	Mise à jour des textes des points 4, 5	Tout le monde	18/04/2023
3.4	Mise à jour des diagrammes de séquences	Loïc Blommaert	20/04/2023
3.5	Refont de toute les diagrammes de classe	Tout le monde	20/04/2023
3.6	Mise à jour des textes du point 6	Tout le monde	21/04/2023
3.7	Mise à jour diagrammes de classe Client	Hugo Charels	21/04/2023

2 Besoins utilisateur : Fonctionnels

2.1 Écran de connexion

La figure 1 montre quels sont les choix de l'utilisateur lorsqu'il se trouve sur la page de connexion. Nous pouvons donc remarquer que 2 choix lui sont proposés : se connecter à son compte (Login) ou se créer un compte (Create an account). Si une de ces 2 actions est réussite (successfull), l'utilisateur sera redirigé vers le menu d'accueil du jeu.

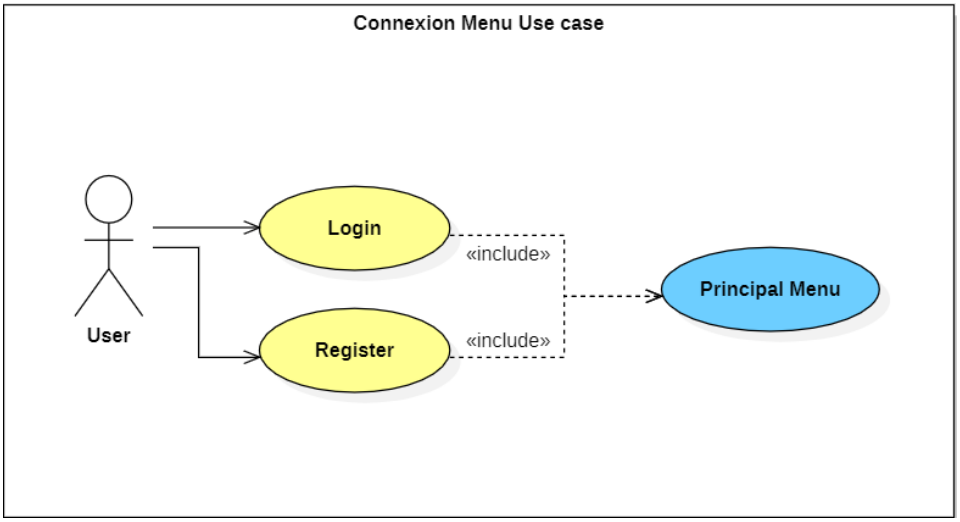

Figure 1 – Use Case : Connexion d'un utilisateur au jeu

2.1.1 Se connecter à un compte

Pour que l'action de se connecter à son compte soit qualifiée de réussie, le programme demandera à l'utilisateur de s'authentifier en entrant son nom d'utilisateur ainsi que son mot de passe. Si un de ses 2 paramètres n'est pas donné par l'utilisateur ou s'il ne respecte pas les règles de validité (figure 8), l'authentification échouera. Pareillement, si le programme ne trouve aucune correspondance entre le nom d'utilisateur entré et ceux présent dans la base de données, l'authentification n'aboutira pas. Pour finir, si le mot de passe entré n'est pas correct et ne correspond pas à celui associé au nom d'utilisateur, l'authentification échouera également. Si aucun des 3 cas cité précédemment ne se produit, la connexion sera qualifiée de réussie.

2.1.2 Créer un compte

Pour qu'une création de compte soit qualifiée de réussie, l'utilisateur devra impérativement entrer un nom d'utilisateur UNIQUE ainsi qu'un mot de passe. Le tout devant respecter les règles de validité (figure 8). Si le nom d'utilisateur n'est pas unique et qu'il existe déjà dans la base de données ou que les règles de validité ne sont pas respectées, la création du compte échouera. Dans le cas contraire, la création du compte réussira et la connexion à celui-ci se fera automatiquement.

2.2 Écran d'accueil

La figure 2 montre quelles sont les actions que l'utilisateur peut effectuer lorsqu'il se trouve dans le menu principal du jeu. Quatre catégories se démarquent : créer ou rejoindre une partie (*Create a game*, *Join a game*), consulter le classement des joueurs (*Show ranking*), gérer sa liste d'amis et discuter avec d'autres joueurs (*Show friend list*, *Send message*) ou se déconnecter (*Deconnection*).

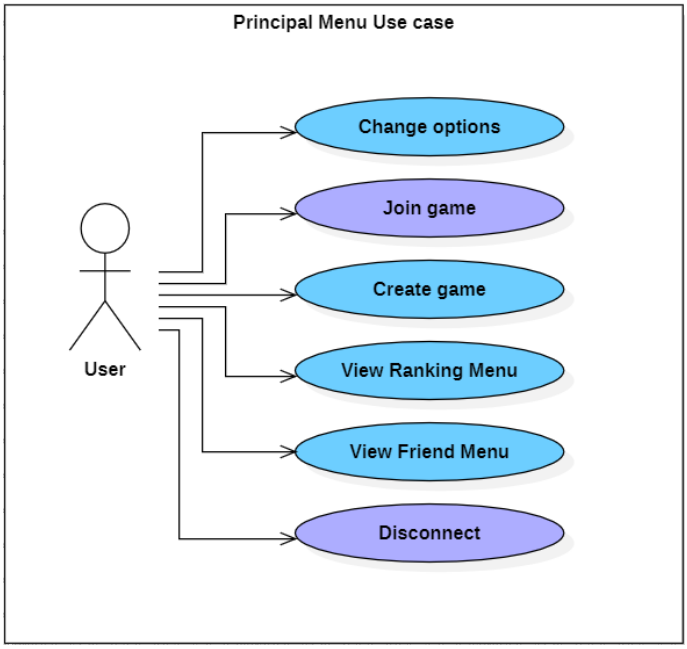

Figure 2 – Use Case : Menu d'accueil

2.2.1 Crée une partie

Chaque joueur a la possibilité de créer sa propre partie de Capitali$t. Lorsqu'il le fait, il peut décider ou non de changer les paramètres de base de la partie. Une fois ces choix faits, il est automatiquement connecté à la partie. Un GameCode s'affiche pour qu'il puisse le partager à d'autres utilisateurs (soit via un moyen externe au jeu, soit via le chat intégré (voir 2.2.4).

2.2.2 Rejoindre une partie

Chaque joueur a la possibilité de rejoindre une partie déjà créée au préalable en entrant le GameCode qui y correspond. Si ce dernier n'est pas reconnu, il ne se passe rien et l'utilisateur ne sera connecté à aucun lobby. Si au contraire le GameCode est reconnu alors l'utilisateur sera connecté à la partie et se retrouvera dans le lobby. Il n'est pas possible de voir l'ensemble des GameCodes actuellement valides. Il n'est également pas possible de rejoindre une partie qui est déjà en cours.

2.2.3 Classement des joueurs

L'utilisateur a la possibilité de regarder le classement des joueurs lorsqu'il se trouve sur l'écran d'accueil du jeu. Le classement est ordonné sur base des points qu'un joueur a acquis à la fin de ses parties de jeu. La manière dont un utilisateur obtient des points est décrite au point 4.2.3

2.2.4 Discuter avec d'autres joueurs

Il y a deux moyens de discuter avec d'autres utilisateurs à travers le jeu. Il y a un chat de partie, disponible lorsqu'on rejoint une partie, et des messages privés, disponible à tout moment. (voir 2.3.1)

Pour envoyer un message privé à quelqu'un, il faut au préalable être ami sur le jeu avec l'utilisateur à qui on tente d'envoyer le message. Une fois que cette condition est remplie, il suffit de taper le nom d'utilisateur de son ami, suivi de son message pour que seul l'ami ait accès à cette conversation.

2.2.5 Gérer sa liste d'amis

Si l'utilisateur choisit de gérer sa liste d'amis, il sera amené dans un autre menu où différentes actions lui sont proposées. La figure 3 montre les différentes actions possibles : ajouter un ami (*Add friend*), retirer un ami (*Remove friend*), voir ses demandes d'ami (*Show friend request*), gérer les demandes d'ami (*Accept friend request*, *Refuse friend request*), et voir les conversations qu'il a avec ses amis (*Show conversation*).

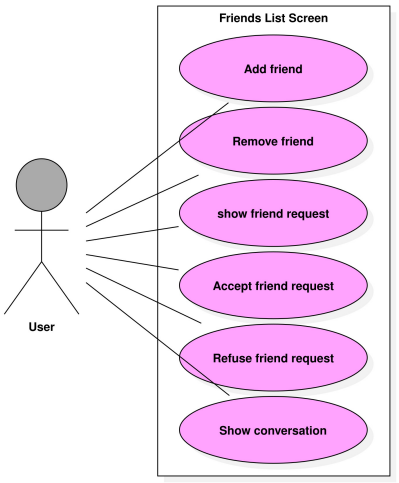

Figure 3 – Use Case : Menu d'amis

2.2.6 Déconnexion

Enfin, l'utilisateur a le droit de se déconnecter de son compte. S'il choisit cette option, il sera renvoyé à l'écran de connexion où l'on lui redemandera de s'identifier s'il souhaite se reconnecter.

2.3 Lobby

2.3.1 Discuter avec d'autre joueurs

Pour utiliser le chat de partie dans un lobby, il suffit de taper son message et celui-ci sera retransmis aux autres utilisateurs de la partie.

2.3.2 Lancer une partie

C'est le créateur de la partie qui décide de quand la lancer. Il est autorisé à lancer la partie à condition qu'il y ait minimum un autre joueur qui l'ait rejoint.

2.4 Partie de Capitali$t

2.4.1 Démarrage de la partie

Lors du démarrage de la partie, chaque joueur est positionné sur la case départ et commence avec la somme d'argent définie lors du paramétrage de la partie (par défaut 1500$). L'ordre dans lequel les joueurs pourront jouer est déterminé de façon aléatoire au démarrage de la partie.

Si nous sommes en mode "partie rapide", la valeur des deux propriétés que chaque joueur reçoit en début de partie est automatiquement prélevée sur leur réserve d'argent.

2.4.2 Déroulement d'un tour

Lors de son tour, chaque joueur pourra effectuer différents choix qui seront contraints par l'*état* du joueur, c'est-à-dire si le joueur est en prison, a fait faillite ou débute simplement son tour sans être dans aucun de ces cas. Nous appelons un début de tour où le joueur n'est dans aucun des cas spéciaux mentionnés ci-dessus le cas "normal". Il est important de savoir que certains événements comme la faillite d'un joueur peuvent provoquer le début de son tour même s'il n'était normalement pas le suivant dans l'ordre.

Les joueurs adverses sont majoritairement spectateurs lorsque les autres jouent. Seuls des négociations entre joueurs, les enchères ou certaines cartes peuvent mener à des interactions entre le jeu et des joueurs dont ce n'est normalement pas le tour. Évidemment, il est toujours possible de discuter dans le chat, que ce soit à votre tour ou non.

Nous discutons du déroulement d'un tour en fonction des différents états dans les points suivants.

2.4.3 Déroulement d'un tour 'normal'

Le joueur a 4 possibilités avant de jeter les dés. Il peut entamer des échanges avec les autres joueurs, tout comme il peut choisir d'hypothéquer ou de déshypothéquer une propriété, d'y ajouter des bâtiments ou de vendre des bâtiments déjà présents sur une de ses propriétés. Le joueur peut effectuer toutes ces actions autant de fois qu'il le souhaite, tant qu'il ne lance pas les dés. Une fois les dés lancés, le joueur n'aura plus accès a ces fonctionnalités excepté en cas de double ou il pourra a nouveau faire une de ces actions tant qu'il ne relance pas les dés.

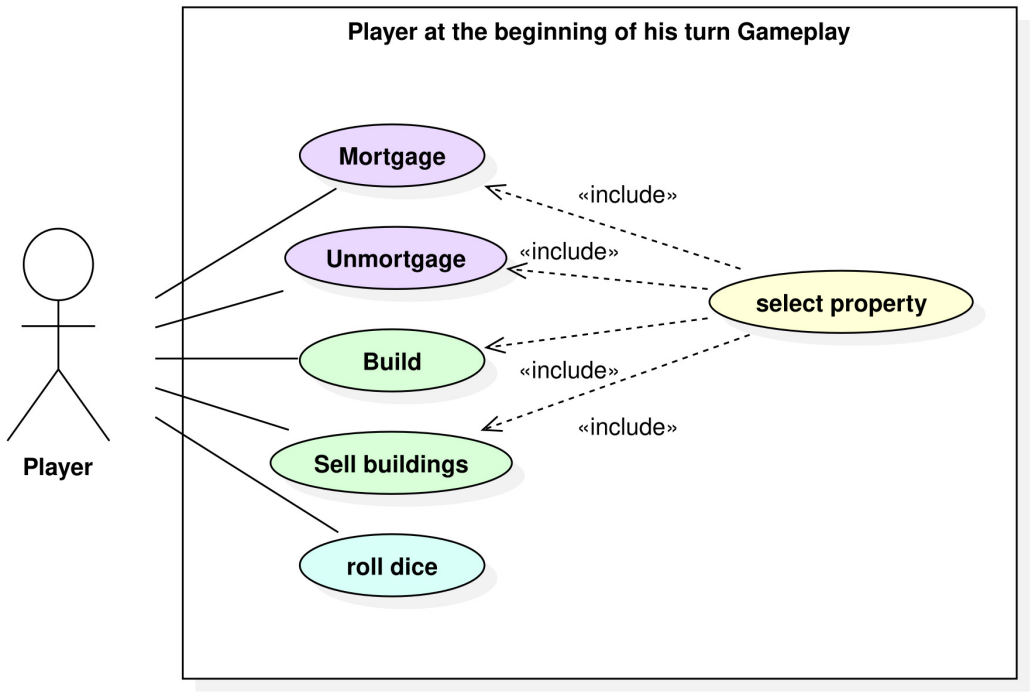

Figure 4 – Use Case : Joueur au début de son tour

2.4.4 Déroulement d'un tour où le joueur est en prison

Si un joueur est en prison, il est privé de toutes les actions qui sont habituellement disponibles au début du tour d'un joueur. Ici, le joueur peut soit tenter de faire un double (maximum 3 essais, un par tour), soit payer directement 50$, soit jouer une carte "Libérer de prison" si le joueur en possédait une préalablement à son arrivée en prison. Après 3 essais de doubles ratés, le joueur est contraint de payer. Une fois libéré, il redevient immédiatement un joueur normal et retombe dans le use case "tour normal". Le tour du joueur prend directement fin lorsqu'il sort de prison.

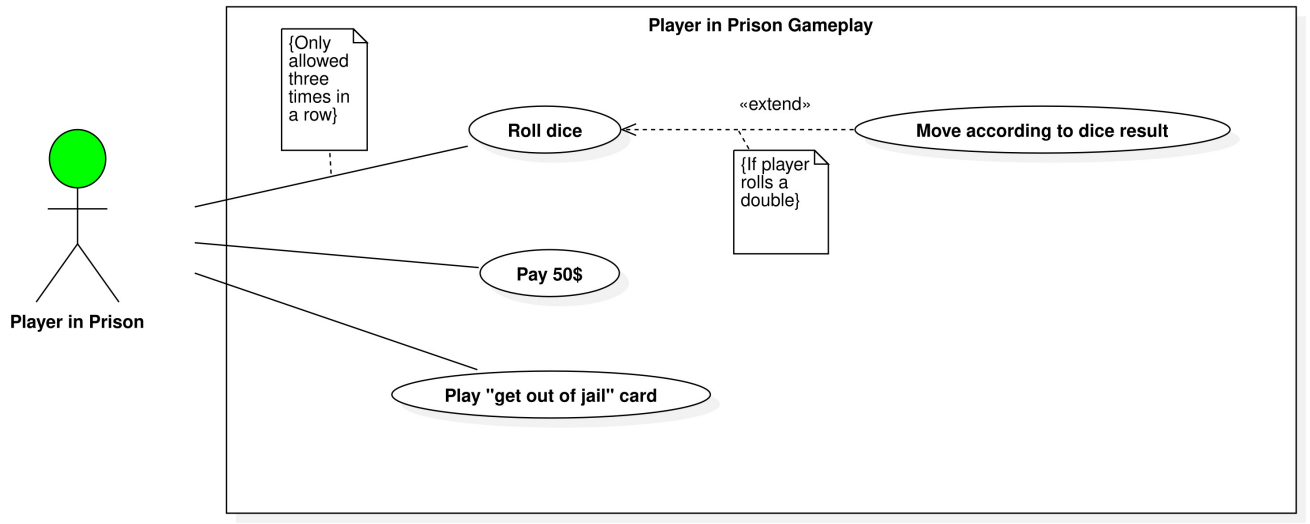

Figure 5 – Use Case : Joueur en prison

2.4.5 Déplacement d'un joueur

À moins d'être en prison, le joueur doit obligatoirement se déplacer chaque fois que c'est son tour. Pour ce faire, il interagit avec le programme pour lancer les dés, dont la somme indiquera au joueur le nombre de cases dont il avance. L'action de la case sur laquelle il arrive sera ensuite réalisée.

Si le joueur fait un double au lancé de dé, il peut les relancer immédiatement après (au même tour). Si le joueur jette des doubles trois fois d'affiler, il est envoyé en prison immédiatement.

2.4.6 Achat et vente de propriétés

Chaque fois qu'un joueur tombe sur une case propriété, gare ou compagnie de service public qui n'est pas possédée par un autre joueur, le joueur à la possibilité de l'acheter (pour peu qu'il en ait les moyens). S'il n'achète pas le terrain, celui-ci est automatiquement mis aux enchères.

2.4.7 Faillite d'un joueur

Une faillite surviendra la plupart du temps pendant le tour d'un joueur, mais elle pourrait aussi survenir pendant le tour d'un autre joueur. Si cela arrive, le tour de la personne dont c'est le tour est immédiatement interrompu jusqu'à la résolution (ou pas) de la situation de faillite. Deux options s'ouvrent au joueur se trouvant dans cette situation. Il peut soit vendre certains de ses bâtiments, soit hypothéquer des propriétés. Notons que pour hypothéquer des propriétés, il ne peut pas y avoir de bâtiments dessus. Évidemment, si le joueur n'arrive pas à couvrir ses dettes, il sera déclarer en faillite, ce qui provoque l'élimination du joueur de la partie.

Si la dette du joueur est envers la banque, toutes ses possessions seront vendues aux enchères. Sinon, toutes seront légués gratuitement au joueur qui a provoqué sa faillite.

2.4.8 Déroulement d'une vente aux enchères

Lorsqu'un joueur n'achète pas un terrain libre sur lequel il tombe, ou qu'un joueur se retrouve éliminé par la banque, des enchères débutent. Tous les joueurs, sauf celui ayant refusé l'achat de la propriété dans le premier cas, seront directement inscrits aux enchères. Dès lors, tour a tour il leur sera proposé dans un temps imparti, d'introduire une mise de valeur supérieur au prix précédent. Si le joueur ne possède pas assez d'argent ou prend trop de temps à répondre, il sera automatiquement retiré des enchères. Si celui-ci ne souhaite tout simplement pas participer, il pourra se désinscrire des enchères lors de son tour. Dès qu'il ne reste plus qu'un joueur inscrit aux enchères, si celui-ci a misé une somme correcte, il achetera la propriété pour la valeur entré, sinon le terrain restera sans propriétaire.

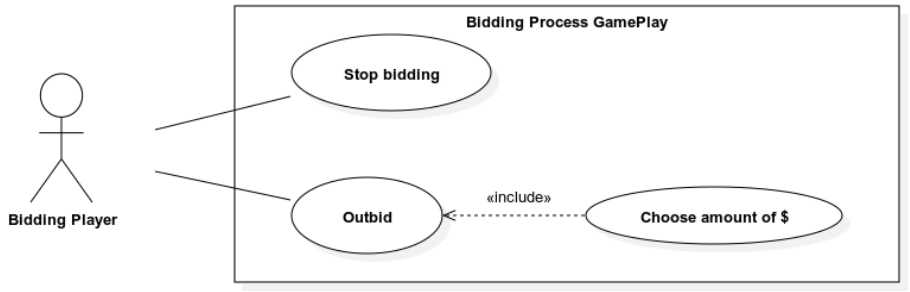

Figure 6 – Use Case : enchère

2.4.9 Discuter avec d'autres joueurs

L'utilisation du chat durant la partie se fait de la manière décrite au point 2.3.1.

2.4.10 Achat et vente de bâtiment

Lorsqu'un joueur possède tous les terrains partageant la même couleur. Il a la possibilité de construire durant son tour un ou plusieurs bâtiment(s) dessus (ce n'est pas possible au moins une propriété de la couleur est hypothéquée). Il est possible de construire jusqu'à 4 maisons sur un terrain. Ensuite, ces maisons peuvent être remplacées par un hôtel.
La différence entre le nombre de maisons sur des terrains d'une même couleur ne peut jamais dépasser 1.

2.4.11 Piocher une carte

Lorsqu'un joueur tombe sur une case chance ou caisse de communauté, il pioche une carte de la pile correspondante. Toutes les cartes ont un effet immédiat (et seront donc directement remises en dessous de la pile) sauf la carte *Vous êtes Libéré de Prison* dont l'utilisation sera proposée à chaque début de tour où le joueur est en prison.

3 Besoins utilisateur : Non fonctionnels

Le jeu doit proposer une interface agréable et intuitive permettant au joueur d'interagir facilement avec les différents aspects du jeu. Le joueur ne doit pas avoir un temps d'attente trop long au moment d'effectuer des actions pendant la partie (lancer les dés, acheter un bâtiment, ...) ou en dehors de la partie (créer un compte, créer une partie, ...).

3.1 Interactivité, agréabilité et dynamisme

3.1.1 Clics et boutons

Les utilisateurs interagissent plus facilement en cliquant sur des boutons avec une souris qu'en écrivant du texte sur leur clavier. Pour la version du jeu avec une interface graphique, la présence de boutons sera fortement employée pour rendre le jeu plus interactif et facile d'utilisation (chaque action possible sera associée à un bouton ou un autre élément interactif de la GUI). Dans le cas d'une version dans le terminal, l'utilisation de bouton se retrouve plus limité et l'utilisateur devra faire appel a l'utilisation de commande pouvant être interprétées par le jeu.

La liste des commandes disponibles pour l'utilisateur se trouve sur le GitHub dans le `ReadMe` à partir de `Connection`.

3.1.2 Interface de l'utilisateur

Le joueur doit pouvoir visualiser et se repérer facilement dans le jeu.
Pour la version du programme avec interface graphique, il y aura naturellement un ensemble de boutons et d'images (pouvant contenir du texte) pour naviguer dans l'interface.
Pour la version du programme dans le terminal, les limitations techniques obligeront l'utilisation d'entrées du clavier de l'utilisateur plutôt que de boutons. Certains éléments très importants comme le plateau de jeu seront représentés en ASCII art comme montre la figure 7.

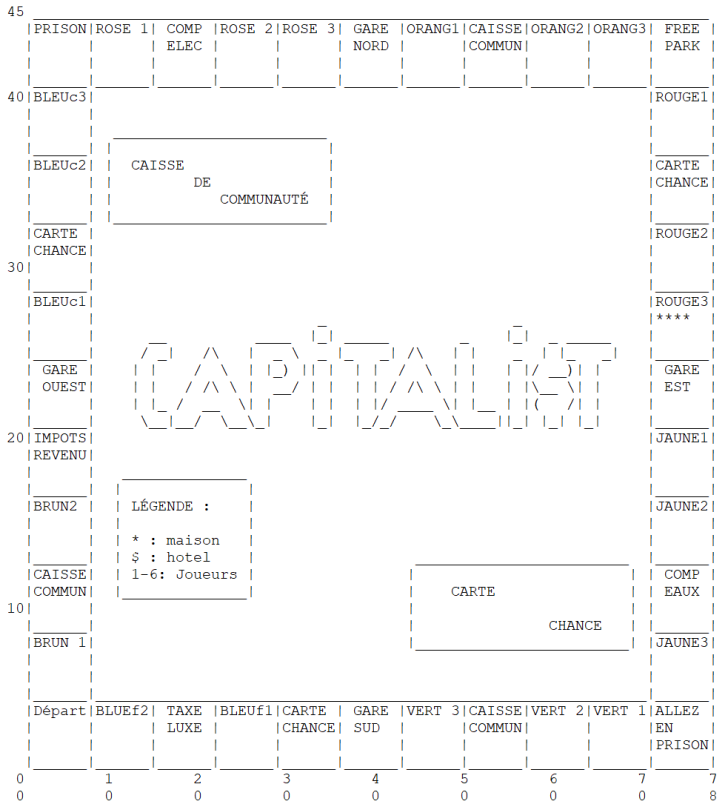

Figure 7 – Plateau de Capitali$t en ASCII art

3.1.3 Langue du jeu

L'ensemble des textes du jeu seront affichés en français.

3.1.4 Création de compte, pseudo et mot de passe

Les utilisateurs devront se créer un compte pour pouvoir jouer au jeu. Pour ce faire, ils devront choisir un nom d'utilisateur et un mot de passe qui leur permettra par la suite de se connecter. Le nom d'utilisateur et le mot de passe devront respecter le tableau de validité ci-dessous 8.

Règle	Nom d'utilisateur	Mot de passe	Message
Taille min	1 caractère	4 caractères	1 caractère
Taille max	32 caractère	32 caractères	1024 caractères
Caractères non-utilisable	spéciaux[1]	spéciaux[1]	aucun[2]

Figure 8 – Tableau : règles de validités

[1] Les caractères spéciaux autres que = - + * $ €§ £ @ & ? ! ne sont pas autorisés

[2] le message doit au moins contenir un caractère visible

4 Besoins système : Fonctionnels

4.1 Écran de connexion

Lorsqu'un utilisateur entre un pseudo et un mot de passe, aussi bien pour une connexion qu'une inscription, le système devra dans un premier temps vérifier que les textes entrés sont au format attendu.

4.1.1 Connexion à un compte

Si le format est valide dans le cadre d'une connexion, le système recherchera la présence des informations entrées (le pseudo et le mot de passe de l'utilisateur) dans la database.

- Si les informations entrées sont présentes dans la database, alors le joueur sera connecté au compte avec les informations correspondant.
- Sinon, le système continuera a affiché la page de connexion.

4.1.2 Création d'un compte

Si le format est valide dans le cadre d'une connexion, le système recherchera la présence des informations entrées (le pseudo et le mot de passe de l'utilisateur) dans la database.

- Si les informations entrées sont présentes dans la database, alors le joueur sera connecté au compte avec les informations correspondant.
- Sinon, le système continuera d'attendre un nouveau nom d'utilisateur et mot de passe.

4.2 Écran d'accueil

4.2.1 Création d'une partie

Le serveur doit gérer la création de lobby et permettre au joueur responsable de sa création de paramétrer la partie avant que celle-ci ne soit lancée. Le jeu sera initialisé avec des paramètres par défauts, le joueur pourra changer certaines choses comme la limite de temps, le mode de jeu, le nombre de joueurs ect...

Lors de la création d'un lobby, le système devra générer un GameCode (4 chiffres aléatoires qui identifient la partie créée). Ce code doit être stocké par le système de manière à ce qu'il puisse s'assurer qu'aucune autre partie ne possède le même identifiant.

4.2.2 Connexion à une partie

Chaque salon de jeu est représenté par un GameCode. Tout joueur rentrant un identifiant de partie valide, sera connecté à celle-ci, tant que la limite de joueur choisi par le créateur de la partie n'est pas atteinte. Si le lobby est plein ou qu'il correspond à une partie en cours, alors le système restera sur la page de connexion en ayant retiré le code entré et attend de mettre un nouveau code.

4.2.3 Gestion du classement

Le classement est calculé selon une formule très simple. A chaque fin de partie, chaque joueur gagne des points en fonction de sa position d'élimination (le premier éliminé ne gagne pas de points, le deuxième un point, le troisième deux points etc...). Les joueurs avec le plus de points se retrouvent en haut du classement tandis que ceux avec le moins de points se retrouvent en bas. Pour départager 2 joueurs qui auraient le même nombre de points, le système divisera le nombre de points gagnés par les joueurs par le nombre de parties jouées. Le joueur obtenant le nombre plus élevé à la fin de ce calcul sera classé devant l'autre joueur. Dans le cas où ce calcul donnerait un résultat encore identique, alors les deux joueurs seront affiché par ordre alphabétique de leur pseudo.

4.2.4 Gestion de la liste d'amis

Le serveur doit permettre aux utilisateurs de gérer leurs listes d'amis. Il doit permettre l'ajout et la suppression d'amis.
Le système considère que deux utilisateurs sont amis lorsque ceux-ci se sont respectivement ajoutés en tant qu'ami. Si l'un des deux utilisateurs supprime l'autre utilisateur de sa liste d'amis, alors ceux-ci ne sont plus considérés comme amis, et il ne leur est plus possible de communiquer via le chat privé.

4.2.5 Gestion du chat

Le système doit gérer, en parallèle de toutes ses activités, la vérification et la transmission des messages entre les amis. Il devra stocker l'historique des messages entre ceux-ci.

4.2.6 Destruction d'un lobby

Le serveur doit supprimer tout salon de jeu qui ne contient plus de joueur, il doit aussi fermer correctement tous les salons en cours si on vient à arrêter le serveur.

4.2.7 Déconnexion

Lorsqu'un client se déconnecte, le système devra le renvoyer sur la page d'authentification.

4.3 Gestion d'une partie

4.3.1 Gestion du chat

Le système doit gérer en parallèle de toutes ses activités la vérification et la transmission des messages entre les joueurs d'une partie. Ces messages ne doivent pas être enregistrés après leur envoi.

4.3.2 Gestion du démarrage de la partie

Au démarrage de la partie, le système doit afficher le plateau de jeu pour les joueurs et déterminer l'ordre de jouer des joueurs. Il affichera un ensemble d'image et de texte qui sont simples à comprendre de comment fonctionne une partie par les joueurs.

4.3.3 Déroulement d'un tour

Le système devra gérer le changement de tour d'un joueur à l'autre. Le système vérifiera que seul le joueur dont c'est le tour peut effectuer des actions réservées au joueur dont c'est le tour.

4.3.4 Gestion des déplacements

Le système génère aléatoirement un lancé de dés puis déplace le joueur en fonction du résultat du lancer. Si le résultat est un double, le jeu déplace le joueur puis lui demande de relancer les dés. Si au troisième lancé, le joueur refait un double le système doit mettre le joueur en prison.

4.3.5 Gestion de l'argent

Le jeu doit ajouter ou retirer de l'argent automatiquement aux joueurs lors d'événements entraînants un transfert d'argent.

Il existe trois événement permettant de gagner de l'argent :

— Recevoir un loyer
— Passer par la case départ
— Piocher une carte chance ou caisse de communauté.
— Vendre des bâtiments
— Hypothéquer une propriété

Les événement provoquant la perte d'argent sont les suivants :

— Payer ses taxes ou un loyer
— Piocher une carte chance ou caisse de communauté
— Acheter des propriétés, construire des bâtiments ou payer pour sortir de prison.

Il est à noter que dans le mode rapide, à chaque début de tour le joueur perdra 20$.

La banque est une entité indépendante gérée par le système. Ayant de l'argent à l'infini, elle ne peut donc pas tomber en faillite. Son rôle est de prendre ou de donner de l'argent aux joueurs lorsqu'un événement ne requiert pas d'interaction avec un autre joueur.

4.3.6 Gestion des propriétés

Lorsqu'un joueur passe sur une propriété, le jeu lui donne la possibilité de l'acheter si elle est encore libre, si le joueur ne souhaite pas l'acheter, elle est mise aux enchères (voir point 2.4.8).

Lorsqu'un joueur tombe sur une propriété qui appartient à un autre joueur, le système l'oblige à payer son loyer. Si le joueur ne possède pas la somme demander après avoir vendu toutes ses possessions, il tombera en faillite et ses possessions restantes seront offert au joueur propriétaire de la case.

Lorsqu'un joueur tombe sur une propriété qui lui appartient, le système lui permet s'il possède toutes les propriétés d'une même couleur, d'y faire construire un ou plusieurs bâtiments, il lui est également possible de revendre ses bâtiments.

Il y a trois types de loyer à payer lorsqu'un joueur tombe sur une propriété. Le loyer est calculé automatiquement par le programme. Si c'est un terrain, le prix est calculé en fonction du nombre de bâtiments présent sur la propriété. Si c'est une gare, le prix est calculé en fonction du nombre de gares possédé par le propriétaire. Dans le dernier cas, le joueur doit relancer les dés, le prix à payer est multiplié par un nombre en fonction du nombre de compagnies possédé par le propriétaire.

4.3.7 Gestion des actions

A chaque tour, le jeu ne doit permettre qu'au joueur dont c'est le tour d'interagir avec le système. En plus du lancé les dés et se déplacer, le joueur peut faire plusieurs actions comme gérer ses propriétés (vendre, construire et hypothéquer). A tous moment de la partie, l'utilisateur aura accès au chat.

Lorsque le joueur se trouve en prison, le jeu ne lui laisse que trois possibilités d'action. Il peut soit tenter de faire un double, soit utiliser sa carte *Sortie de prison*, soit payer pour sortir. Au 3ème lancé de dé, si ce n'est toujours pas un double, le joueur sera obligé de payé pour sortir de prison.

4.3.8 Gestion des cartes

Chaque fois qu'un joueur tombe sur une case chance ou caisse de communauté (voir 2.4.11), le système choisi une carte aléatoirement et l'affiche au joueur. Le joueur est obligé d'effectuer l'action indiquée par la carte sauf si c'est une carte *Sortie de prison* dans ce cas le système lui permet de la garder pour l'utiliser en cas de besoin. Cette carte sera par la suite enlevée du paquet, les autres joueurs ne pourront pas utiliser cette carte utilisée.

5 Besoins système : Non fonctionnels

5.1 Système d'exploitation et langage de programmation

Le jeu doit être exécutable sur Linux, celui-ci devra être codé en C++.

5.2 Réseau

Le protocole utiliser est TCP IP.

5.3 Disponibilité

Le jeu est accessible tant que la connexion entre le client et le serveur est assurée. Les joueurs et le serveur doivent avoir accès à internet. Le serveur doit être manuellement lancé par un utilisateur, qui sera responsable de son exécution.

5.4 Performances

Le jeu doit être fluide avec un temps de réponse sous l'ordre des millisecondes en conditions normales.

5.5 Capacité d'une partie

Il y a 6 joueurs maximum par partie.

5.6 Sécurité

Pour pouvoir accéder à une partie, le joueur doit se connecter à un compte grâce à un mot de passe. Il faut que le joueur ne puisse pas tricher dans une partie, le joueur ne doit pas pouvoir faire une action qui enfreint les règles du jeu.

5.7 Versions

Nous proposerons trois versions du programme : La CLI, la CUI et la GUI.

— La CLI est version rudimentaire du programme, principalement destinée aux développeurs.

— La CUI est une version terminale du programme à l'aide la librairie Ncurses. Elle est destinée aux utilisateurs possédant une machine peu performante.

— La GUI est la version graphique du programme. C'est la version la plus élaborée que nous recommandons a chacun de nos utilisateurs.

Ces trois versions sont compatibles : ça signifie que que deux joueurs qui ne seraient pas sur la même version pourraient jouer sur la même partie.

6 Design et fonctionnement du système

6.1 Système de connexion

6.1.1 Création d'un compte

La création d'un compte se déroule selon le diagramme suivant :

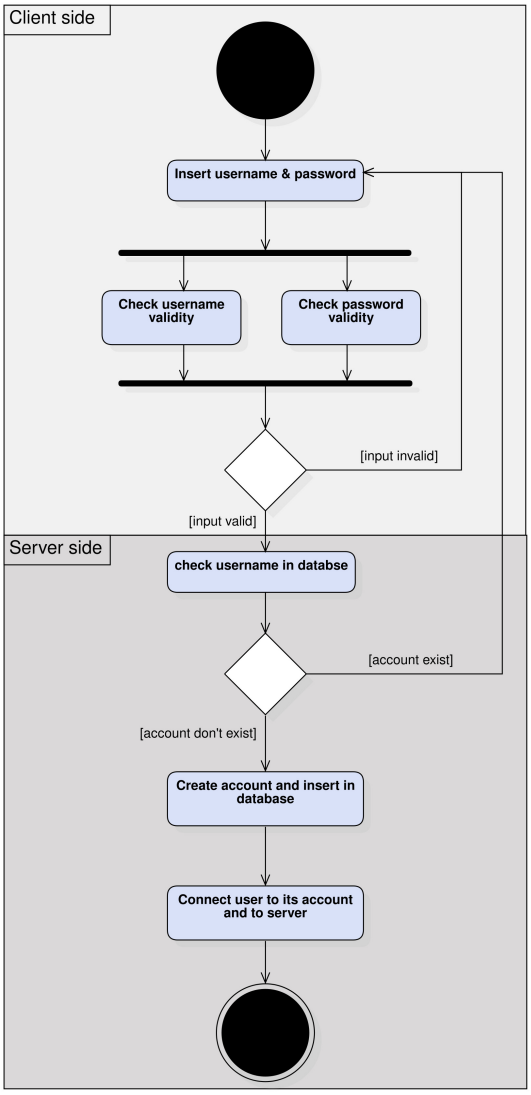

Figure 9 – Diagramme d'activité : création d'un compte

Le client entre son nom d'utilisateur ainsi que son mot de passe, qui se retrouvent envoyé au serveur qui si un compte existe déjà avec ce pseudonyme.

6.1.2 Connexion à un compte

La connexion a un compte se déroule quasiment de la même manière que la création d'un compte.

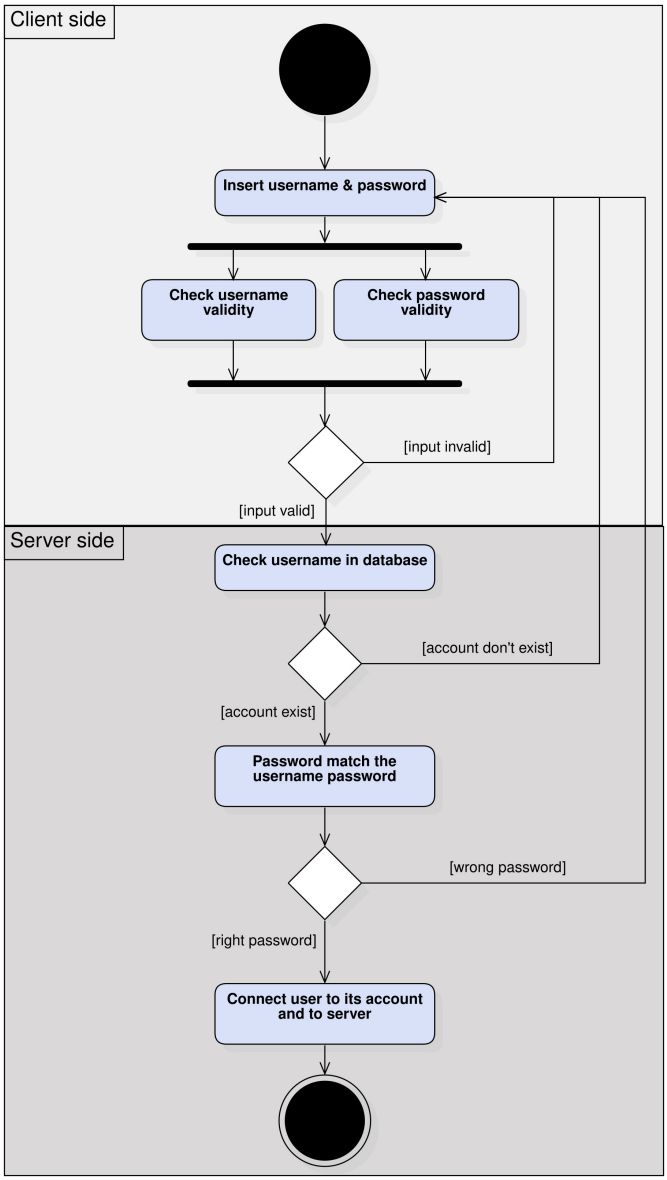

Figure 10 – Diagramme d'activité : connexion

Le client entre son nom d'utilisateur ainsi que son mot de passe, qui seront encore une fois envoyé au serveur qui vérifiera si l'utilisateur existe dans la database.

6.2 Fonctionnement de l'infrastructure

L'infrastructure du serveur se décompose comme le diagramme suivant nous le montre :

FIGURE 11 – Diagramme de classes : Système du serveur et base de donnée

Il existe une classe principale Server, qui se charge de créer des GameServer (lobby de jeu) en cas de requête par l'utilisateur. Le serveur charge a son lancement la database et stocke son contenu dans une classe Database,

qui contiendra l'ensemble des informations concernant les joueurs. Le serveur communique avec le client grâce a l'énumération QUERY_TYPE. Les deux classes couleur saumon (List et AccessMonitor) servent à gérer les problèmes de synchronisations.

6.2.1 Envoie d'une requête au serveur par le client

Le diagramme suivant représente le fonctionnement du système lorsqu'un utilisateur effectue une requête auprès du serveur :

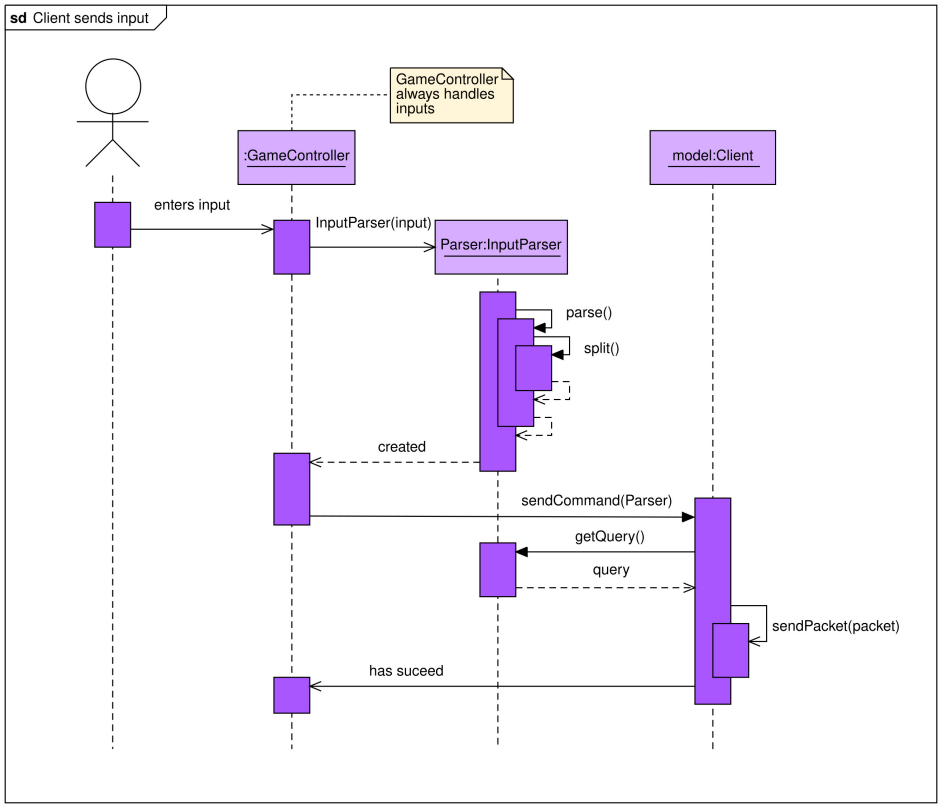

Figure 12 – Diagramme de séquence : Envoie d'une requête au serveur par le client

Un objet de la classe GameController écoute les entrées du joueur. Lorsque celui-ci en soumet une, le string issu de l'entrée sera traité et analysé afin que le GameController puisse envoyer par la suite, la requête associée au model. La requête est envoyée par packet.

6.2.2 Réception d'une requête par le serveur

Le diagramme suivant représente la manière dont le serveur reçoit les requêtes du joueur :

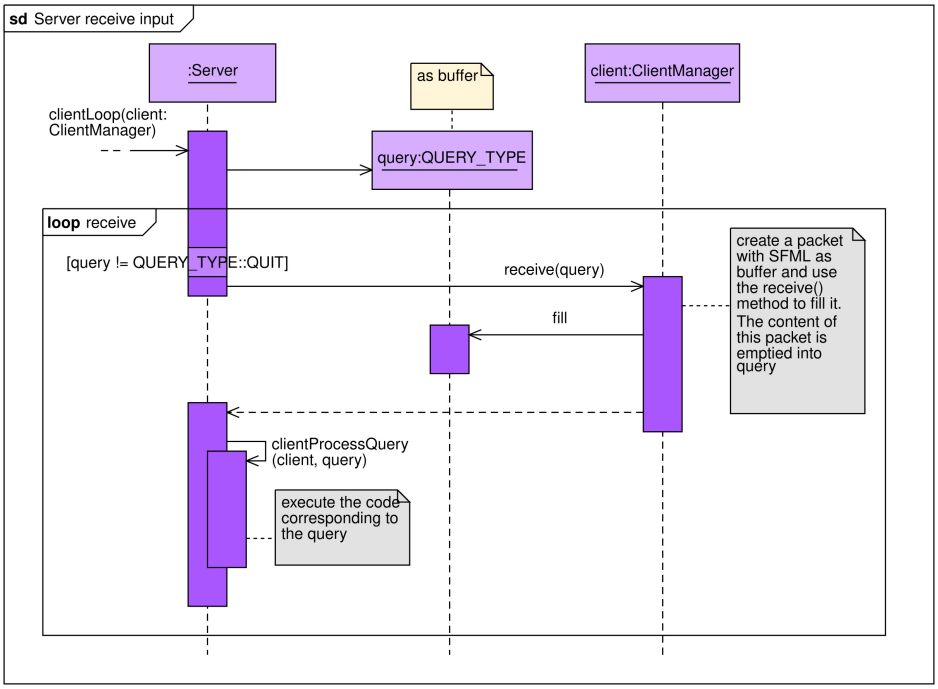

Figure 13 – Diagramme de séquence : Réception d'une requête par le serveur

Le serveur crée un thread par client, lorsqu'un client se connecte. Ce thread exécute une fonction clientLoop qui attends de recevoir le packet envoyé précédemment par le client. Le contenu du packet (la valeur de la requête) se retrouve stocké dans la variable query. une fois le packet vidé, le serveur appelle la fonction clientProcessQuery qui exécutera, en fonction de la query, le code correspondant.

6.3 Client

6.3.1 Diagramme de classe : client

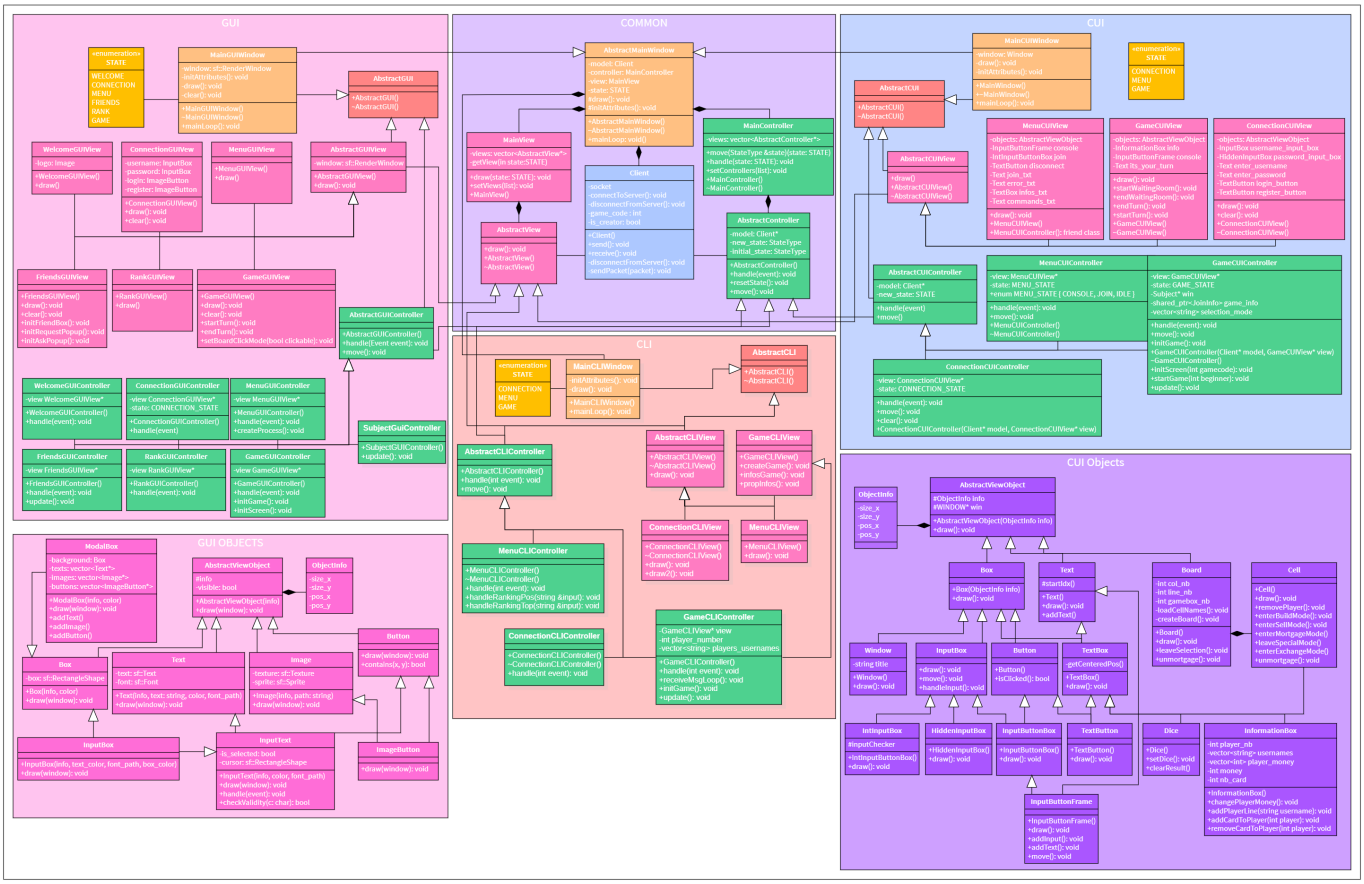

FIGURE 14 – Diagramme de classes : client

Le diagramme représente l'intégralité des classes, avec leurs attributs et méthodes essentielles selon les trois versions existantes du programme : La CLI, la CUI et la GUI. Ces trois versions distinctes partagent cependant la même structure d'origine représentée dans COMMON, ainsi que le même modèle, ce qui implique que les trois versions sont compatibles.

Les deux versions graphiques (CUI et GUI) possèdent un ensemble d'objet préconçu pour remplir les fenêtres, ces objets sont représentés dans l'ensemble CUI Objects et GUI objects.

6.4 Fonctionnement du jeu

6.4.1 Diagramme de classes : Modèle du jeu

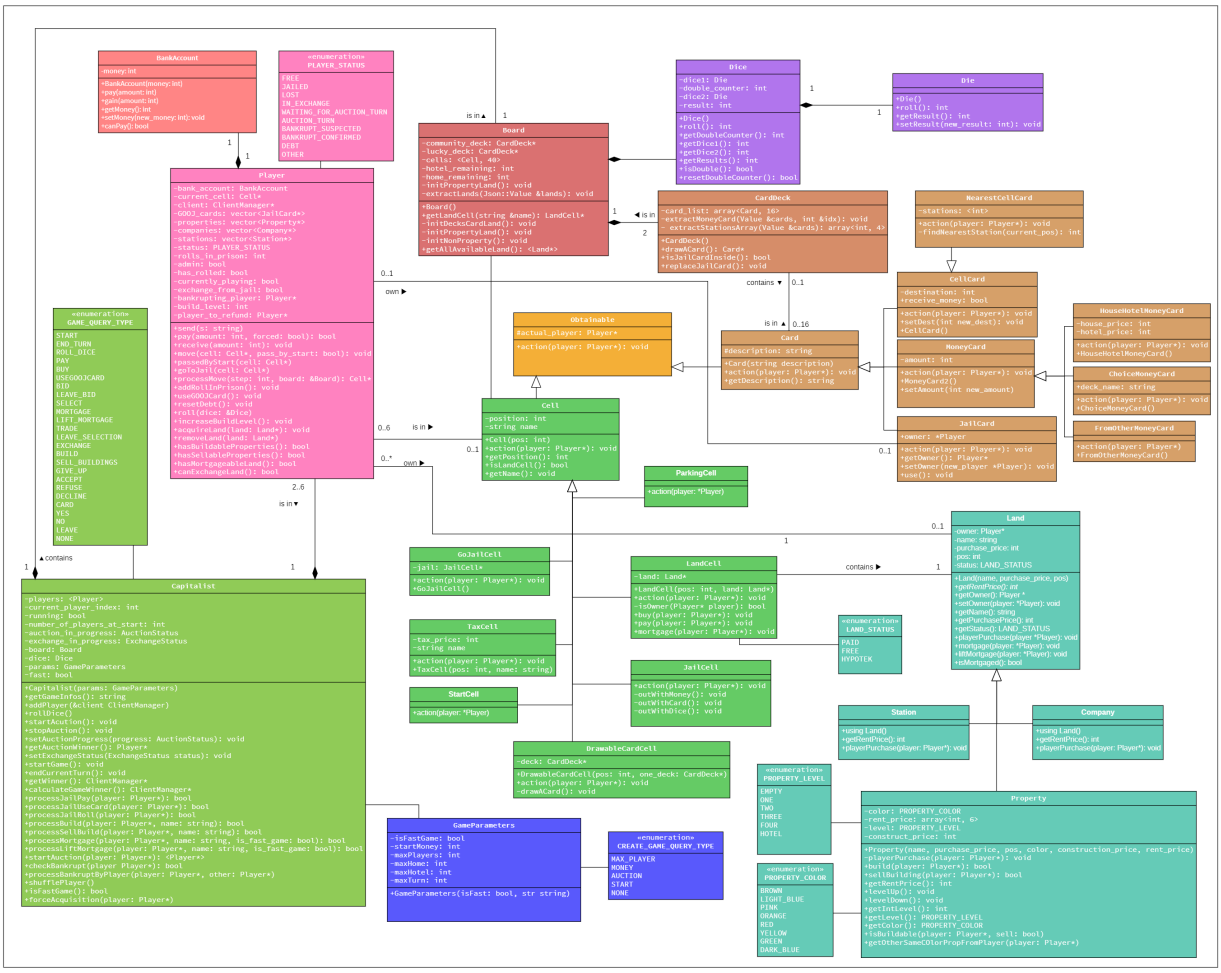

FIGURE 15 – Diagramme de classes : Modèle du jeu

La figure 15 représente le modèle du jeu. Une instance de Capitalist est créée à chaque début de partie initialisant Board, l'ensemble des Player et lançant la boucle de jeu. La classe Board représente le plateau contenant le reste des éléments du jeu : les cases (Cell), les cartes (Card), les propriétés (Property) qui héritent de la classe Obtainable représentant tous les éléments que les joueurs peuvent obtenir. Chacune de ces classes ont également des classes enfants qui représentent une spécification de ces classes.