Touché Coulé: SRD

Projet d'informatique 2 (INFO-F209)

Islam BENAIM SAHTANE RGHIF, Romain CROUGHS, Noah DEBUS, Alessandro DORIGO, Gabriel GOLDSZTAJN, Manon GORTZ, Pawel JADCZUK, Philippe NYABYENDA, et Lucas VAN PRAAG

Département d'Informatique, Faculté des Sciences, Université Libre de Bruxelles

Année 2023-2024

Table des matières

Chapitre 1

Introduction

1.1 But du projet

Ce projet a pour objectif de créer une expérience immersive où deux joueurs s'affrontent dans le célèbre jeu de société "Bataille navale". Chacun des joueurs dispose d'un plateau de jeu unique, dissimulé à son adversaire, sur lequel il place stratégiquement ses navires. Le déroulement du jeu se fait tour à tour, avec les joueurs tentant de localiser et de couler les navires adverses en lançant des attaques sur des coordonnées spécifiques. Dans le jeu originel, lorsqu'une attaque atteint un navire, la case correspondante est signalée par un point rouge ; autrement, elle est marquée d'un point blanc. La victoire est remportée par le joueur qui parvient à couler tous les navires adverses.

Deux modes de jeu sont disponibles : le mode classique et le mode commandant.

Mode classique :
Les flottes des deux joueurs se composent de cinq navires de tailles variées : 2x1, 3x1, 3x1, 4x1, et 5x1. Ces navires peuvent être orientés de 90 degrés, mais leur placement en diagonale n'est pas autorisé. De plus, deux navires ne peuvent pas être adjacents l'un à l'autre. Chaque tour permet à un joueur de tirer un coup, mais tant que le tir touche un navire, le joueur peut continuer à tirer.

Mode commandant :
Dans le mode commandant, différentes factions sont introduites, chacune avec ses propres navires et compétences distinctes. Contrairement au mode classique, dans ce mode, deux navires peuvent être positionnés côte à côte. Au début de son tour, chaque joueur reçoit des points d'énergie. Chaque action, comme le tir, consomme des points d'énergie. De nouvelles actions sont également disponibles, telles que l'utilisation d'un sonar révélant une zone de 3x3, le déclenchement d'un bombardement aérien ciblant une zone de 4x1 et bien d'autres. Si un joueur n'utilise pas tous ses points d'énergie au cours d'un tour, il peut les conserver pour le tour suivant, offrant ainsi une flexibilité tactique pour planifier des actions stratégiques plus complexes.

Un mode "extra" supporter est également présent. Dans ce mode, l'utilisateur observe une partie jouée par un ami.

En dehors d'une partie, un joueur a la possibilité de discuter avec ses amis et de gérer sa liste d'amis.

1.2 Glossaire

— Matchmaking : Mécanisme de mise en relation des joueurs afin de créer une partie en ligne.
— Plateau (Board) : Deux matrices de jeu, qui sont assiociées à deux joueurs.
— Lobby : Interface/lieu où l'on regroupe des joueurs avant de démarrer une partie.
— Player (Joueur) : Membre de la partie qui communique avec le jeu suite aux actions (input) de l'utilisateur
— User (Utilisateur) : Client qui est connecté en reseau et lié à compte.

1.3 Historique

Version	Nom	Modifications	Date
0.1	Manon Gortz	Introduction	3/12/23
0.2	Manon Gortz	Besoins utilisateur et squelette besoins système	3/12/23
0.3	Manon Gortz	Ajout des besoins non fonctionnels système	7/12/23
0.4	Manon Gortz	Descrption du design du système	7/12/23
0.5	Manon Gortz	Version incomplète du diagramme de classe	7/12/23
0.6	Manon Gortz	Squelette fonctionnement système	7/12/23
1.1	Pawel Jadczuk	Modification de la structure du fichier	13/12/23
1.2	Pawel Jadczuk	Correction et retouche du fichier	13/12/23
1.3	Pawel Jadczuk / Philippe Nyabyenda	Ajout diagrammes de séquence (chp4)	14/12/23
1.4	Manon Gortz / Noah Debus	Modification finale du diagramme de classe (version soft et version full)	15/12/23
1.5	Pawel Jadczuk / Romain Croughs / Noah Debus	Restructuration du fichier et annotation des nouveaux sous-diagrammes	15/12/23
1.6	Pawel Jadczuk / Romain Croughs	Glossaire	15/12/23

Chapitre 2

Besoins utilisateurs

2.1 Connexion

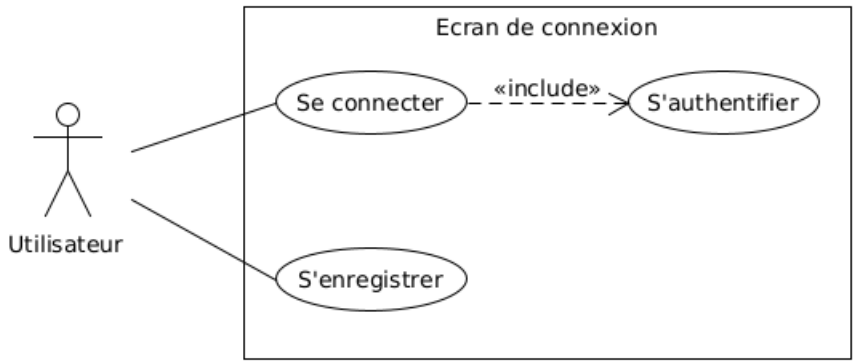

FIGURE 2.1 – Connexion

Lorsque le jeu est lancé, une fenêtre de connexion s'affiche, offrant à l'utilisateur la possibilité de se connecter à son compte. Si l'utilisateur ne dispose pas encore d'un compte, il a la possibilité d'en créer un. La creation d'un compte ne nécessite qu'un nom d'utilisateur valide, et un mot de passe. Une fois l'authentification réussie, l'utilisateur est directement dirigé vers le menu principal du jeu, sans avoir à se connecter une fois de plus.

2.2 Menu principal

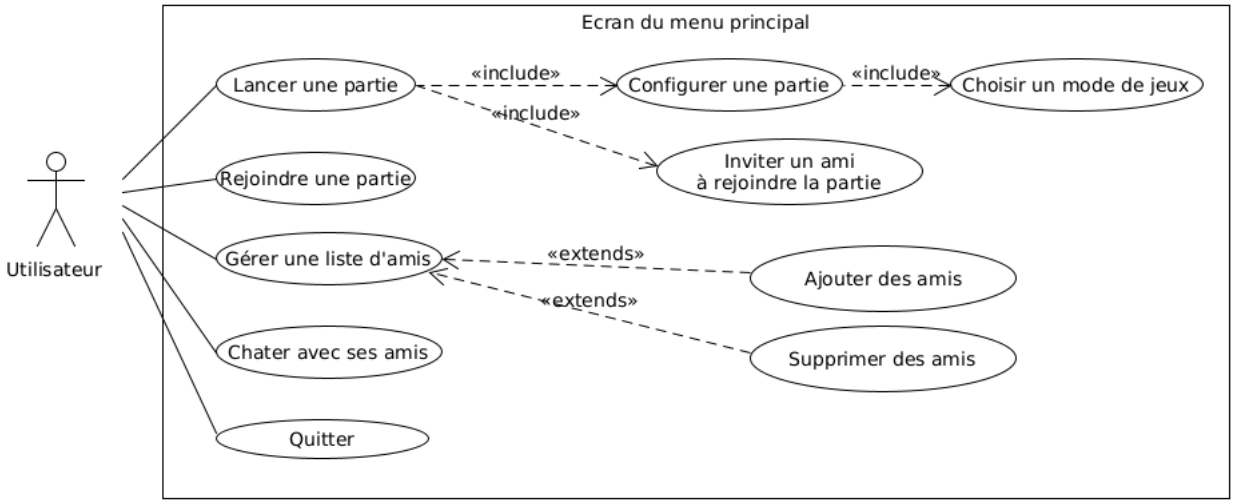

FIGURE 2.2 – Menu principal

Le menu principal est la première interface que l'utilisateur voit lorsqu'il lance le jeu. Elle reprend les informations suivantes :

- La liste des amis
- Les notifications (si l'utilisateur a reçu un message)
- Les différents menus accessibles depuis l'écran principal

2.2.1 Lancement d'une partie

Lors du lancement d'une partie le joueur doit simplement fixer les règles en choisissant :

- Le type de partie : **classique** ou **commandant**
- Le temps d'une partie : au maximum **15 minutes**
- Le temps par tour d'un joueur : au maximum **30 secondes**

Après quoi il sera redirigé vers un lobby, auquel il pourra inviter des joueurs, qui seront tous en mode spectateur s'ils acceptent l'invitation. Un joueur peut aussi rejoindre le lobby sans recevoir d'invitation. Si au moins un joueur à rejoint la partie, le joueur principale peut le fixer en tant que second joueur et lancer la partie. Mais avant que la partie ne puisse commencer, il faudra placer tous les bateaux disponibles sur le board, après quoi la partie se lance.

2.2.2 Rejoindre une partie

Afin de rejoindre une partie, l'utilisateur doit simplement entrer un code de partie qu'il aura reçu de la part du joueur principal. Notons que le code peut être envoyé par le biais du chat, ou tout autre moyen de communication.

2.2.3 Gérer sa liste d'ami

Chaque utilisateur possède une liste d'amis qu'il peut modifier. Il a la possibilité de :
— ajouter un ami a sa liste
— retirer un ami de sa liste
La liste d'ami permet d'initier chat avec l'ami, ce qui permet de notamment communiquer avec lui, mais également de lui envoyer des codes de partie.

2.2.4 Chat

Si le joueur a des amis en ligne dans sa liste d'amis, il a la possibilité de communiquer avec eux via un chat. Si le joueur n'est pas en ligne, il peut tout de même envoyer un message, qui sera reçu par l'ami lorsqu'il se connectera. Cependant, nous ne pouvons pas assurer le système de notification lorsque l'ami n'est pas en ligne.

2.3 En partie

Le jeu doit être conçu pour offrir une interface intuitive, facilitant ainsi les actions du joueur. Dans un premier temps, les deux joueurs sont invités à placer les bateaux sur le board. Les bateaux qui sont disponibles varient du mode de jeu qui a été choisi. Un message d'erreur est affiché lorsque le bateau est placé sur une mauvaise case et auquel cas le joueur est invité à essayer une autre position. Le même procédé sera utilisé pour les tirs.

Une fois tout les bateaux placés sur le board, le timer commence et la game débute. Chacun des joueurs doit alors tirer sur une case du board de l'adversaire, et le jeu continue jusqu'à ce que l'un des joueurs ait coulé tous les bateaux de l'autre joueur. Si un joueur touche un bateau, il a le droit de tirer une autre fois, sinon c'est au tour de l'autre joueur. Pour rendre l'expérience en jeu plus agréable et plus interactive, le temps de jeu ne doit pas être trop long.

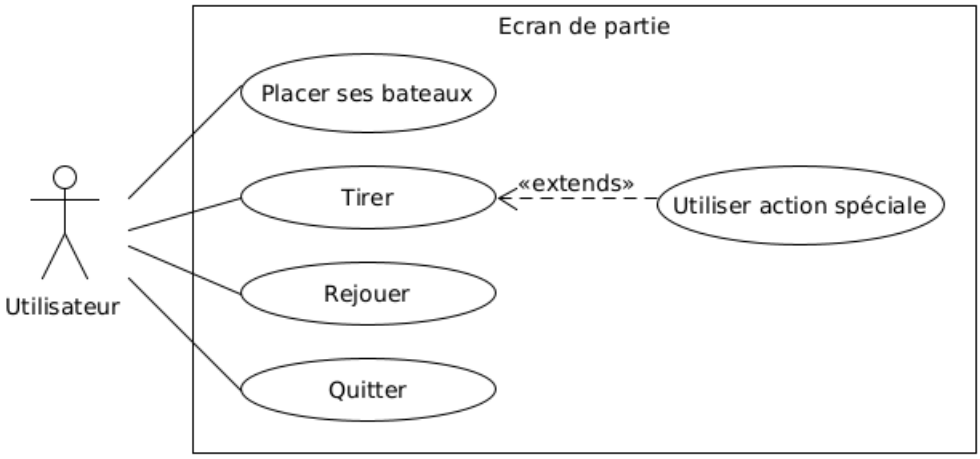

FIGURE 2.3 – En partie

2.3.1 Mode classique

Le mode classique est le mode jeu original de la bataille navale. Chaque joueur doit placer 5 bateaux, qui sont les suivants :

— 1 porte-avion (5 cases)
— 1 croiseur (4 cases)
— 2 sous-marin (3 cases)
— 1 torpilleur (2 cases)

Vous aurez remarqué que le contre-torpilleur n'est pas présent dans ce mode de jeu, mais que nous avons deux sous-marins. Cela s'explique car il n'est pas dissociable du sous-marin dans nos modes d'affichages, nous avons alors fait le choix de le fusionner avec le sous-marin.

2.3.2 Mode commandant

Le mode commandant est un mode de jeu plus complexe, qui offre une expérience de jeu plus intense. Chaque joueur peut choisir entre les trois factions disponibles, ce qui lui donnera des compétences spéciales, et une flotte de bateaux personnalisée. Cependant, toutes les factions commencent la partie avec une mine.

Tous les tours, les joueurs gagnent un point d'énergie. Avec cette énergie, le joueur peut utiliser une compétence spéciale. Si un joueur touche une mine, il perd 3 points d'énergies. Si le joueur a une énergie négative, le joueur en question ne peut pas tirer pour ce tour.

Les capacités qui sont diponibles pour toutes les classes sont donc les suivantes :

- **Tir normal 1x1** - *0 point d'énergie* : Le tir normal est le même que dans le mode classique. Il ne coûte pas d'énergie, mais il n'est pas disponible si l'énergie est négative.
- **Sonar 3x3** - *4 points d'énergie* : Le joueur peut utiliser un sonar pour savoir si une case est de l'eau ou non.
- **Bombardement en ligne 4x1** - *4 points d'énergie* : Le Bombardement en ligne est similiare au tir normal, mais il touche 4 cases en ligne.
- **Bombardement de périmètre 4x4** - *9 points d'énergie* : Le Bombardement de périmètre est similiare au tir normal, mais il touche les contours extérieurs d'un carré de 4x4.

Faction Mine

Cette faction est celle des spécialistes de la mine. Ils possèdent la flotte suivante :

— 2 bateaux de 2x1
— 2 bateau de 3 cases au choix
— 1 bateau de 4 cases au choix

La compétence spéciale de cette faction est la suivante :

- **Pose de mine** - *2 points d'énergie* : Le joueur peut poser une mine sur une case de son choix. Si un bateau adverse touche cette case, il perd 3 points d'énergie.

Cependant, cette faction **ne peut pas recevoir de bombardement de périmètre**.

Faction Sonar

Cette faction est celle des spécialistes du sonar. Ils possèdent la flotte suivante :

— 3 bateau de 2x1
— 4 bateau de 3 cases au choix

Les compétences spéciales de cette faction est la suivante :

- **Sonar 3x3** - *3 points d'énergie*

- **Sonar 10x1** - *4 points d'énergie*

Faction Bombardement

Cette faction est celle des spécialistes du bombardement. Ils possèdent la flotte suivante :
— 1 bateau de 2x1
— 1 bateau de 3 cases au choix
— 2 bateaux de 4 cases au choix
— 1 bateaux de 5 cases au choix
Les compétences spéciales de cette faction est la suivante :
- **Bombardement 2x2** - *4 points d'énergie*
- **Bombardement 4x4** - *7 points d'énergie*

Chapitre 3

Besoins système

3.1 Connexion

Le système doit pouvoir gérer la création de nouveaux comptes en verifiant au préalable les données fournies. Une fois validé, les données sont sauvegardées dans le serveur.

Pour se connecter, le système vérifie l'authenticité des données dans le serveur.

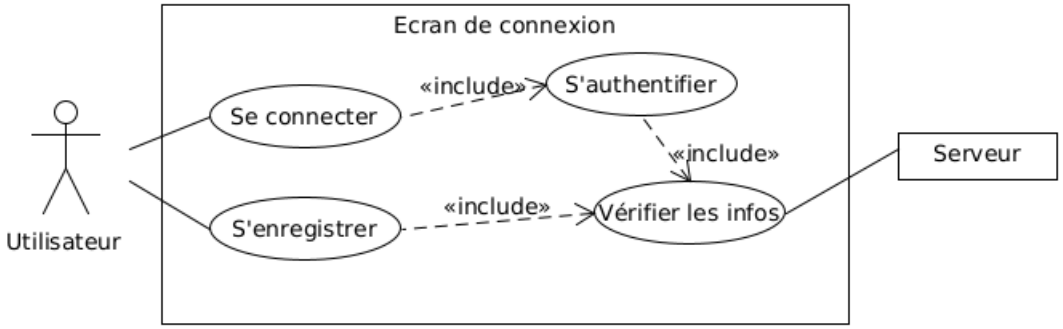

FIGURE 3.1 – Connexion

3.2 Lobby

Au lancement d'une partie, l'interface du lobby est affiché. On peut à ce moment inviter de nouveaux joueurs. Le lobby est joignable par d'autres joueurs sans nécessairement avoir envoyé d'invitations. Le système s'occupera de les ajouter au lobby et les définiras en tant que spectateur. Il faudra alors en définir un en tant que second joueur. Une fois cela fait, on peut lancer la partie. Le système s'occupera de vérifier cette contrainte et de passer au lancement de la game.

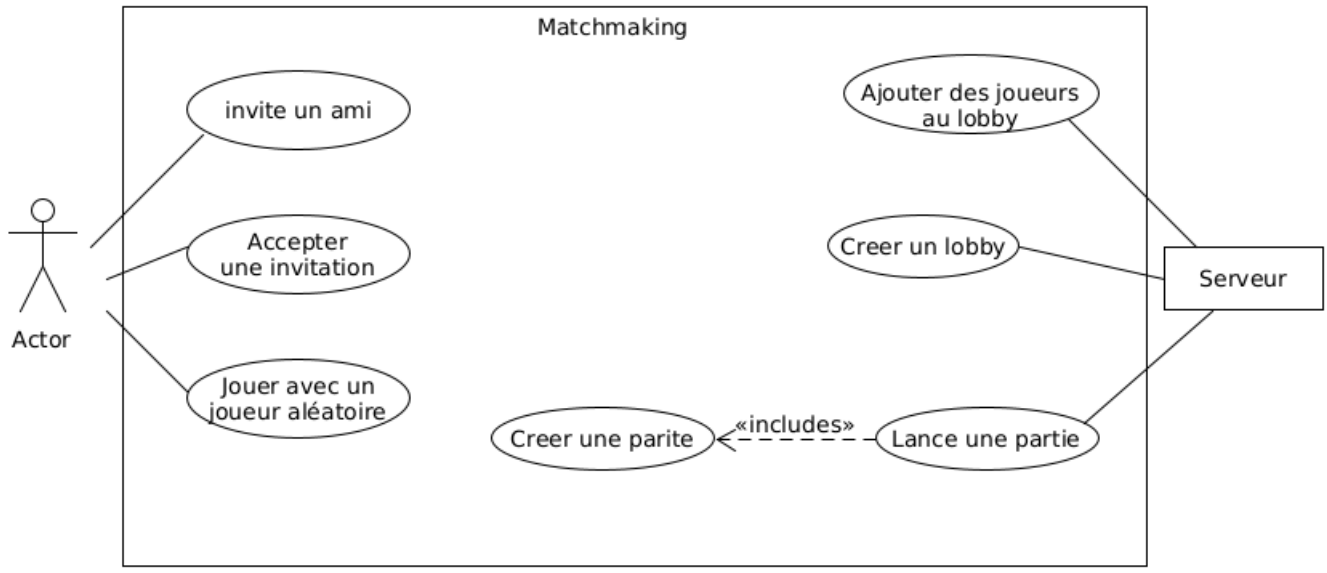

FIGURE 3.2 – Lobby

3.3 Partie

Le système doit pouvoir permettre une action au tour par tour pour chaque joueur mais aussi vérifier la validité de ces actions. Il doit laisser le même joueur jouer à chaque fois qu'il touche un bâteau ennemi.
Il permettera trois affichages distincts

- Le plateau avec les deux boards où le premier joueur ne voit pas les bateaux adverses.
- Le plateau avec les deux boards où le deuxième joueur ne voit pas les bateaux adverses.
- Le plateau avec les deux boards où le spectateur ne voit pas les bateaux adverses.

Chapitre 4

Design et fonctionnement du Système

4.1 Design du système

Le système est divisé en 7 entités :
— Client (en gris)
— Connexion et Inscription (en rose)
— Menu principal (en jaune)
— ModeCommande (en bleu ciel)
— Lobby (en orange)
— Chat (en vert)
— Jeu (en bleu foncé)
— Database (en rouge)

Nous allons les détailler dans différents diagrammes. A noter que toutes ces entités vont hériter des mêmes classes abstraites. Cela est du au fait d'utiliser le design MVC pour cette application. Chaque entité du système est donc composée de 3 composantes principales :

— La vue : Elle correspond à l'interface, elle permet d'afficher les informations et de reçevoir les input de l'utilisateur qu'elle communique au controller.
— Le controller : Il gère les requètes envoyées par la vue et les communique au serveur avec lequel il échange des informations. Il met le serveur à jour suite aux actions d'utilisateurs.
— Le serveur : il contient toutes les informations relatives aux utilisateur et à une partie. Il est responsable d'envoyer les informations requise par le controller et permet de mettre les joueurs en relation pour une partie.

En partie, chaque client est connecté au même plateau, et à son propre côté. Le game serveur fait le lien entre les deux. Le gameController permet de mettre à jour le plateau de chaque client après une action.

4.1.1 Client

Les classes abstraites Vue, Console et GUI jouent un rôle essentiel dans l'affichage des divers écrans de l'application, offrant ainsi une expérience utilisateur flexible soit en ligne de commande, soit à travers une interface graphique. La classe abstraite Controller sert à valider les saisies des utilisateurs avant de les transmettre au serveur, garantissant ainsi la cohérence et la sécurité des données échangées. Par ailleurs, la classe GameClient permet à l'application de se connecter au serveur et de lui soumettre des requêtes, assurant ainsi la communication fluide entre l'utilisateur et le système distant. Enfin, la classe Driver agit comme un pont entre toutes ces classes, orchestrant leur interaction harmonieuse pour offrir une expérience utilisateur complète et cohérente.

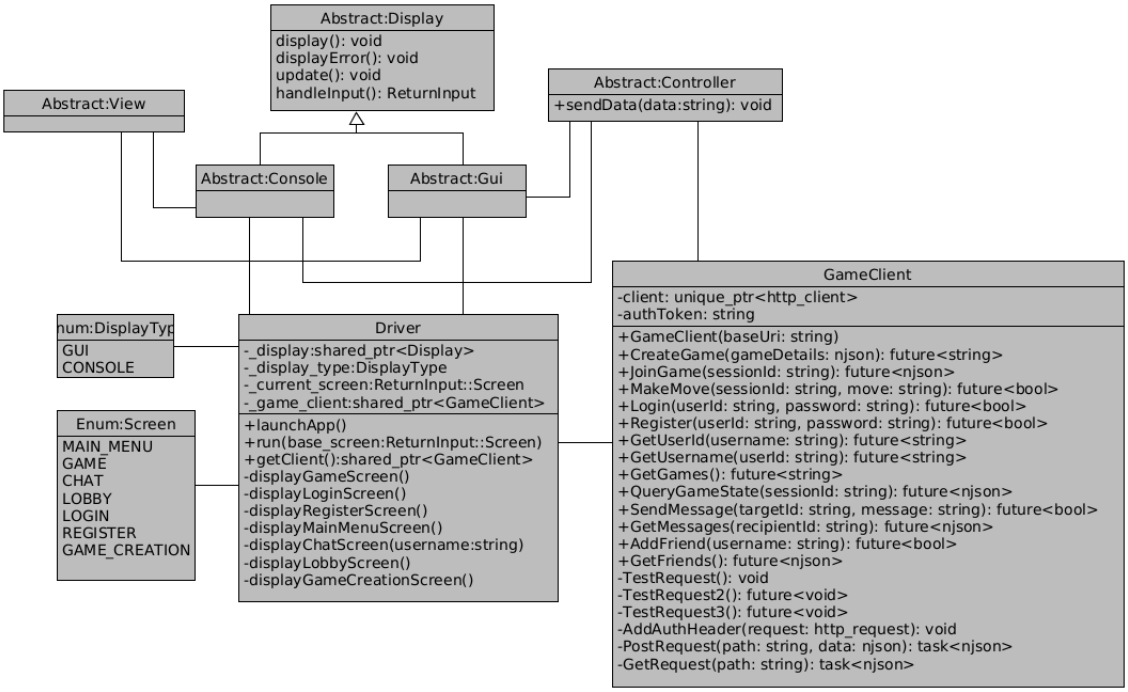

FIGURE 4.1 – Client

4.1.2 Connexion et Inscription

Dès que le joueur lance le jeu, il a la possibilité de soit créer un compte, soit se connecter au compte qu'il a déjà. Il aura donc deux interfaces différentes pour ces deux actions.

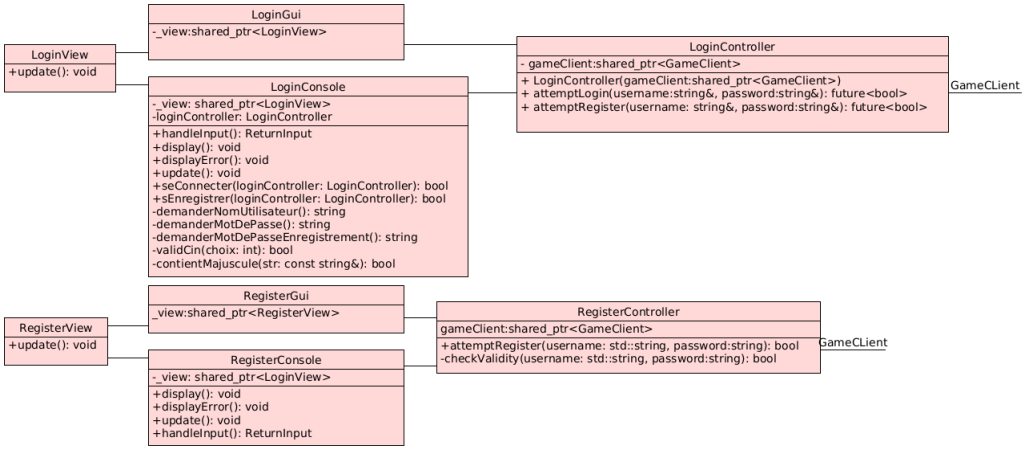

FIGURE 4.2 – Connexion et Inscription

4.1.3 Menu principal

Le menu principal est celui qui permettra d'accéder aux autres écrans du jeu, tels que la création de partie, ou la modification de profil.

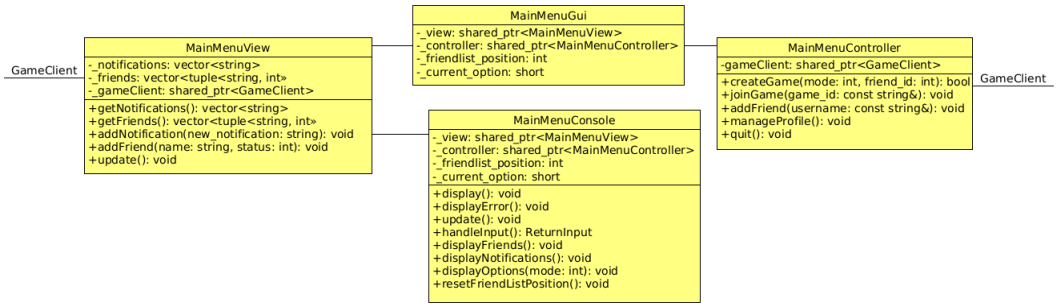

FIGURE 4.3 – Menu principal

4.1.4 Mode Commandant

Le mode Commandant est une option de jeu spéciale qui permet aux joueurs de choisir une faction de bateaux unique et d'accéder à des actions spéciales pour influencer les batailles navales.

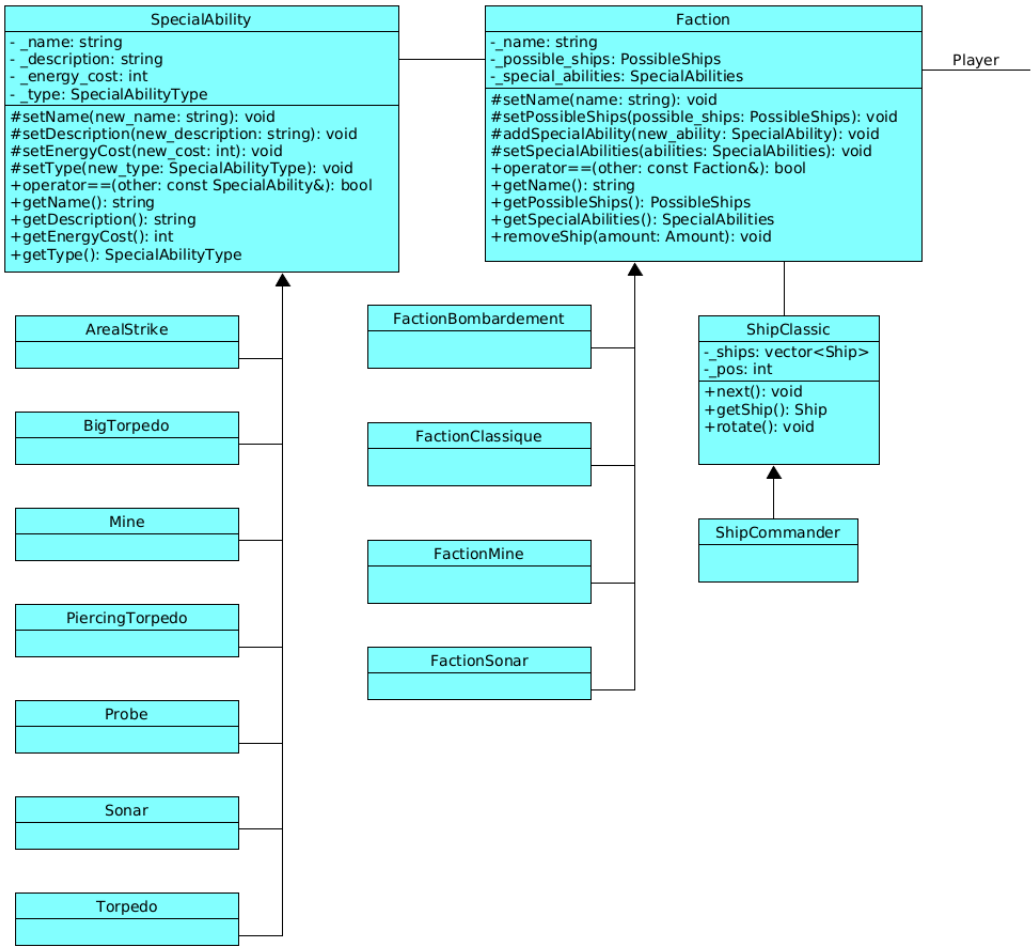

FIGURE 4.4 – Gestion des comptes

4.1.5 Loby

En lançant une partie, un joueur rejoint le lobby en attendant que les autres joueurs et spectateurs se joignent à celle-ci.

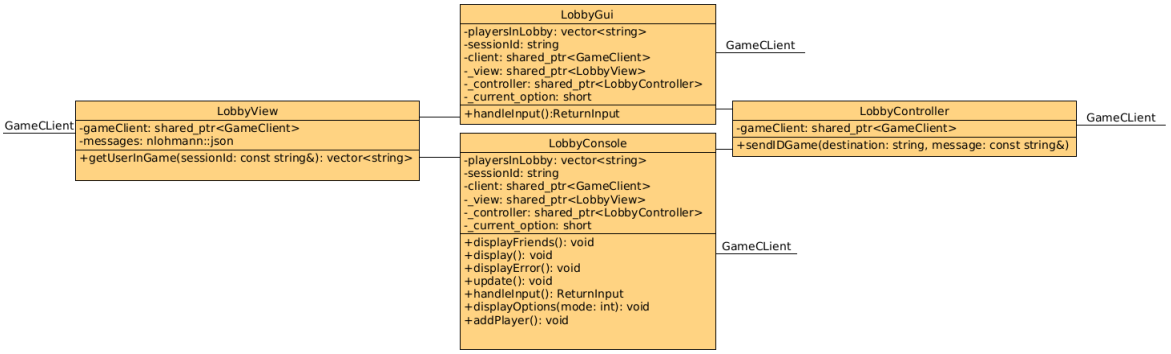

FIGURE 4.5 – Profil

4.1.6 Chat

Le jeu offre aux joueurs un chat qui leur permet d'envoyer une chaine de caractère qui sera envoyée à tous les autres joueurs.

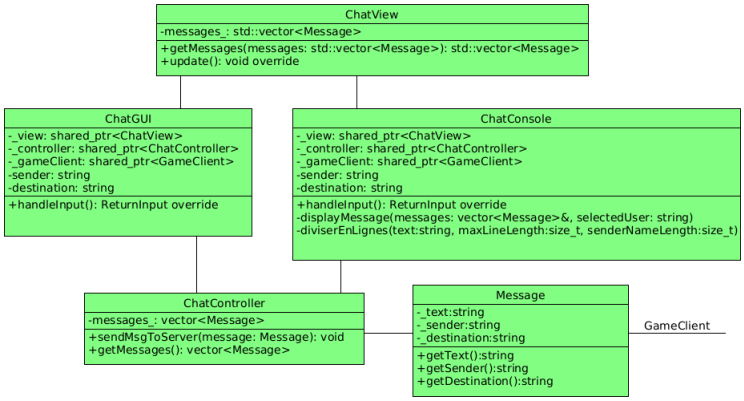

FIGURE 4.6 – Chat

4.1.7 Jeu

Affichage du jeu

La partie locale du jeu est concue de la manière suivante : le joueur a une board local qui est une copie du board du serveur (avec les cases de l'adversaire qui sont masquées), un controlleur qui permet de communiquer avec le serveur et un display qui permet d'afficher une partie à l'écran.

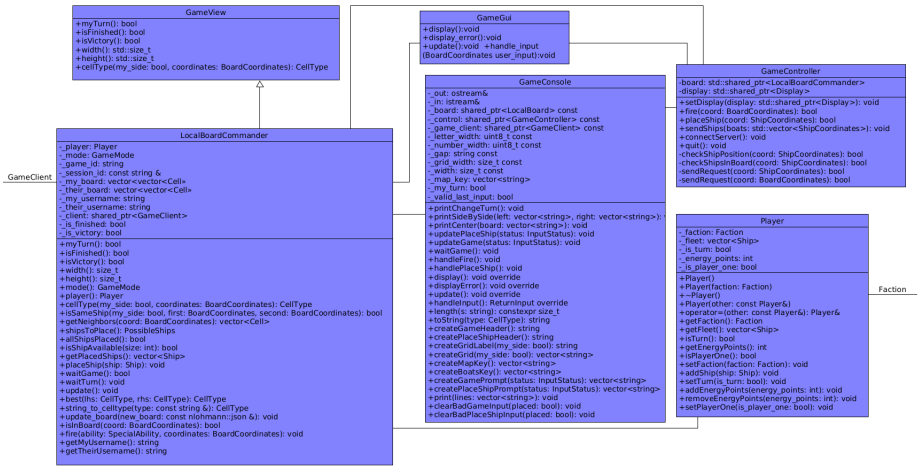

FIGURE 4.7 – Game affichage

Game server

La partie distante du jeu est gérée par une classe GameServer, chargée de réaliser les requêtes auprès de la base de données, de les communiquer au GameClient et de fournir des tokens d'identification. À chaque partie, le GameServer crée une session de jeu gérée par le SessionManager. La GameSession se charge de créer un état de jeu (gameState) ainsi qu'une instance de jeu (Game) contenant le plateau de jeu. Cette structure hiérarchique permet une gestion efficace et organisée des parties en ligne.

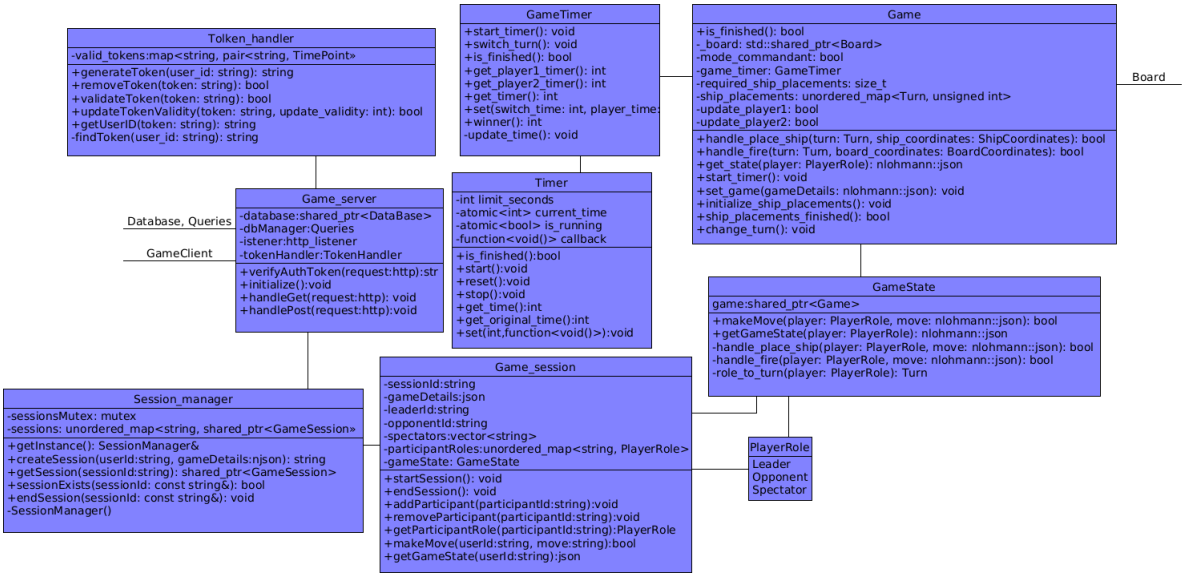

FIGURE 4.8 – Game Server

Plateau de jeu

En ce qui concerne la partie serveur d'une partie, nous avons une instance de Board qui contient les plateaux des deux joueurs, ainsi qu'une flotte qui appartient à ces deux joueurs. Le GameServeur est la classe qui s'occupe de la communication entre le Board et les joueurs.

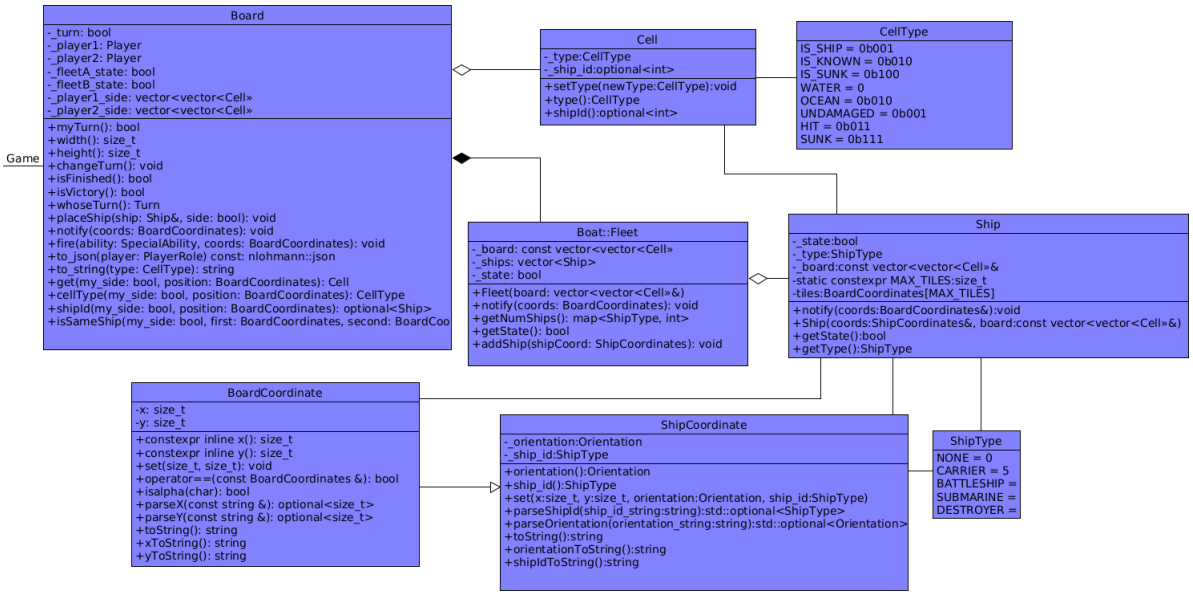

FIGURE 4.9 – Game Board

4.1.8 Database

La base de données stocke les informations utilisateur telles que le nom d'utilisateur, le mot de passe hashé, la liste d'amis et les messages. La base de données est composée d'une classe Database effectuant les opérations de base telles que l'insertion et la sélection de données, ainsi que d'une classe Queries qui manipule la classe Database pour soumettre des requêtes spécifiques à l'application. Ces classes renvoient des instances de QueryResult, permettant de gérer les erreurs liées à la base de données et le résultat des requêtes de manière efficace.

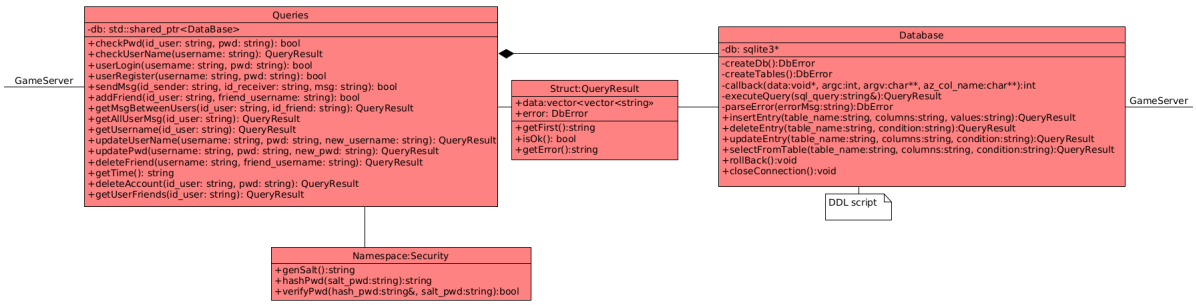

FIGURE 4.10 – Database

4.2 Fonctionnement du système

Cette section portera sur des diagrammes qui présenteront le fonctionnement de différentes parties du système.

4.2.1 Architecture de la communication client-serveur

Le client communique avec le serveur via une collection de requêtes HTTP. Techniquement, nous utilisons la bibliotèque `CppRestSDK` pour envoyer des requêtes HTTP.

Pour vérifier qu'un joueur soit bien connecté, le joueur dispose d'un token d'authentification qui est envoyé avec chaque requête. Ce token est généré lors de la connexion et est stocké dans le cache du client. Il n'est pas valide indéfiniment, et est régénéré à chaque nouvelle connexion. Chaque requête est interceptée par le serveur, qui vérifie la validité du token. Si le token est valide, la requête est traitée, sinon, une erreur est renvoyée. Ce token est envoyée pour toutes les requêtes qui nécessitent une authentification, comme par exemple, pour envoyer un message dans le chat, ou pour démarrer une partie.

De plus, bien que tout le code soit écrit en C++, nous avons décidé de nous inspiré des architectures REST utilisées dans les applications web. C'est pour cela que nous avons décidé que toutes les requêtes seraient envoyées en format JSON, et les réponses sont également envoyées avec ce format. Afin de pouvoir traiter le format JSON, nous utilisons la bibliotèque de Niels Lohmann, nlohmann/json.

4.2.2 Inscription

Si le joueur possède déjà un compte, il a la possibilité de se connecter directement. Le gestionnaire d'inscription (registerController) effectuera une première vérification des données saisies par l'utilisateur, notamment en s'assurant qu'aucun champ n'est laissé vide.

Une fois cette vérification effectuée, les données seront transmises au serveur via l'endpoint `api/register`. Ce dernier prendra comme argument la nom d'utilisateur et le mot de passe.

Ce dernier procédera ensuite à la vérification de la disponibilité du nom d'utilisateur. Afin de maximiser la sécurité, le mot de passe sera haché avant d'être stocké dans la base de données. La base de donnée contiendra alors le nom d'utilisateur et le mot de passe haché. Cependant, pour une recherche dans la base de donnée, il est bien plus efficace de donner un identifiant unique à chaque utilisateur. C'est pour cela que nous avons décidé de donner un identifiant unique à chaque utilisateur. Ce dernier sera généré automatiquement lors de l'inscription et est récupérable via l'endpoint `api/login/uid`.

Maintenant que tout s'est bien déroulé, l'utilisateur va automatiquement être connecté. C'est à dire que le token d'authentification sera généré et envoyé au client.

Remarque : Le token a deux fonctions : à savoir qui envoie le message, et savoir si la personne qui envoie le message est bien connecté. Par exemple, le token est particulièrment utile pour le chat, car il permet de savoir qui envoie le message, et si l'utilisateur est bien connecté (pour éviter que tout le monde puisse récupérer les message d'un autre utilisateur).

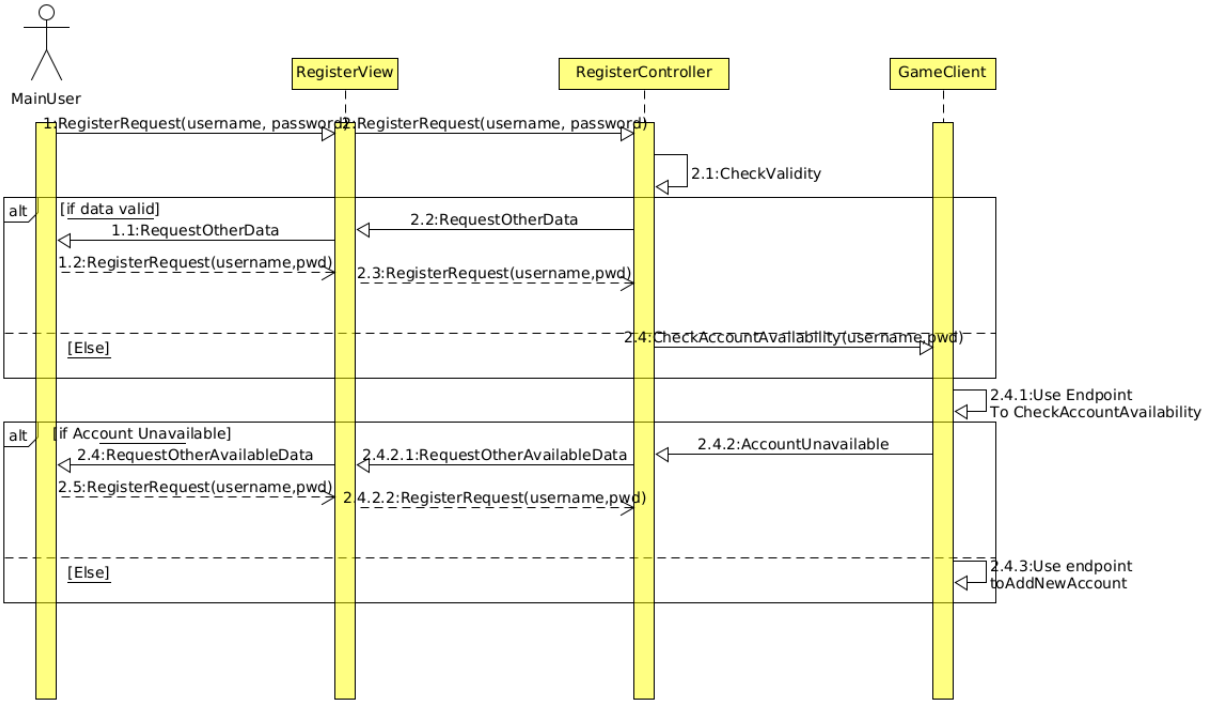

FIGURE 4.11 – Inscription

4.2.3 Connexion

La connexion est en quelques sortes similaire à l'inscription. L'utilisateur doit saisir son nom d'utilisateur et son mot de passe, et le gestionnaire de compte va vérifier si le compte est bien valide. Si c'est le cas, le serveur va générer un token d'authentification et l'envoyer au client. Dans le cas contraire, le client va recevoir un code d'erreur.

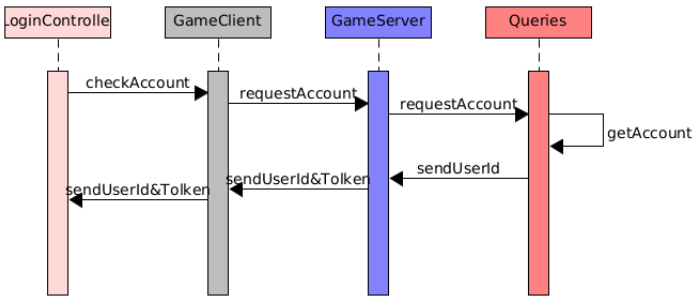

FIGURE 4.12 – Connexion

4.2.4 Chat

Le chat permet aux utilisateurs de communiquer les uns entre les autres. Il s'agit uniquement de *whispers* (messages privés), qui ne comprennent que des messages textuels. Le chat est complètement géré par le serveur, c'est-à-dire qu'aucun message n'est stocké par l'utilisateur.

L'utilisateur a alors simplement deux endpoint pour chatter : `api/chat/send` pour envoyer un message, et `api/chat/get` pour recevoir les messages. Lors de chaque requête, l'utilisateur envoie son token et le nom de son ami. Le serveur vérifie alors si l'utilisateur est bien connecté, et si l'ami existe bien. Si c'est le cas, le serveur peut alors effectuer toutes les manipulations nécessaires pour envoyer ou recevoir un message.

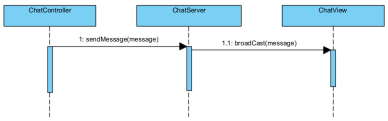

FIGURE 4.13 – Envoie d'un message

4.2.5 Création et configuration d'une partie

Ce diagramme décrit la séquence d'actions réalisées lorsqu'un utilisateur veut créer/configurer une partie.

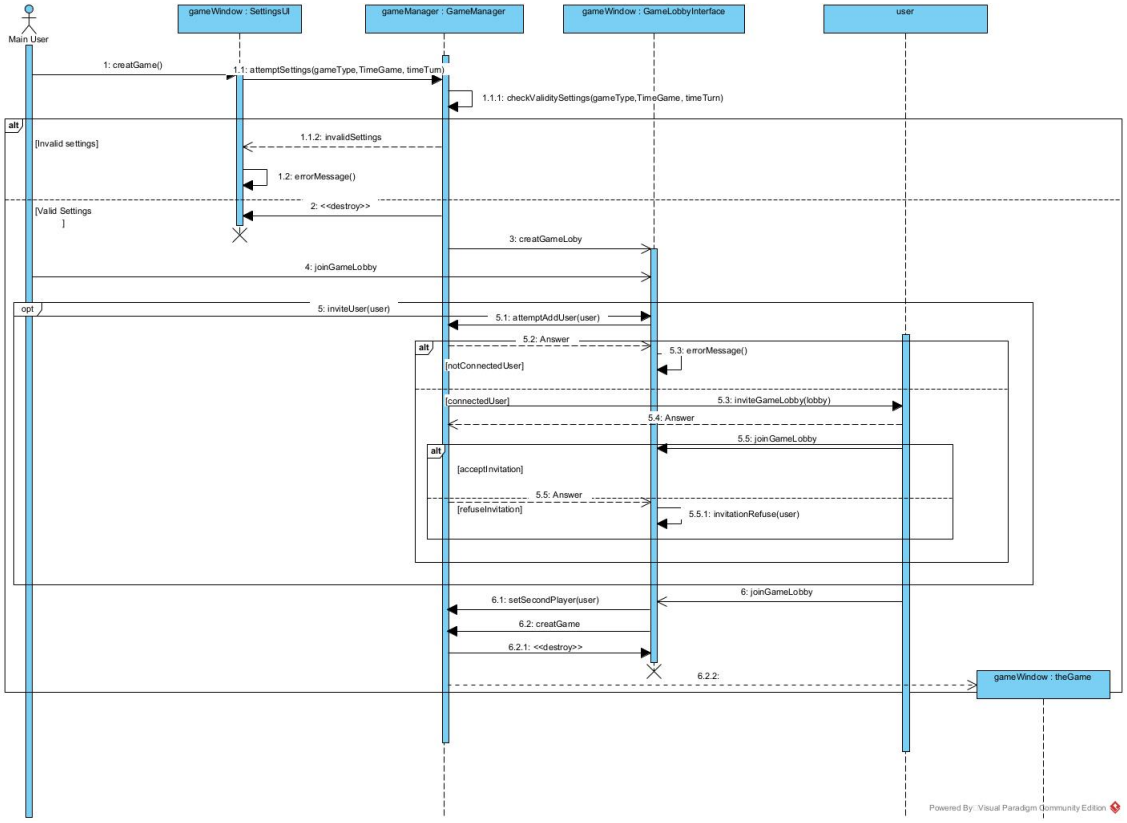

FIGURE 4.14 – Création d'une partie

4.2.6 Lobby

Ce diagramme décrit la séquence d'actions réalisés lorsqu'un utilisateur se trouve au stade de lobby et qu'il veut démarrer une partie. Le deuxième joueur a déjà été fixé, et les bateaux doivent être placés sur le plateau pour que la partie puisse se lancer. Mais avant de placer les bateaux, le plateau avec les deux boards est créé mais aussi 2 players. Un player est assigné à chaque user.

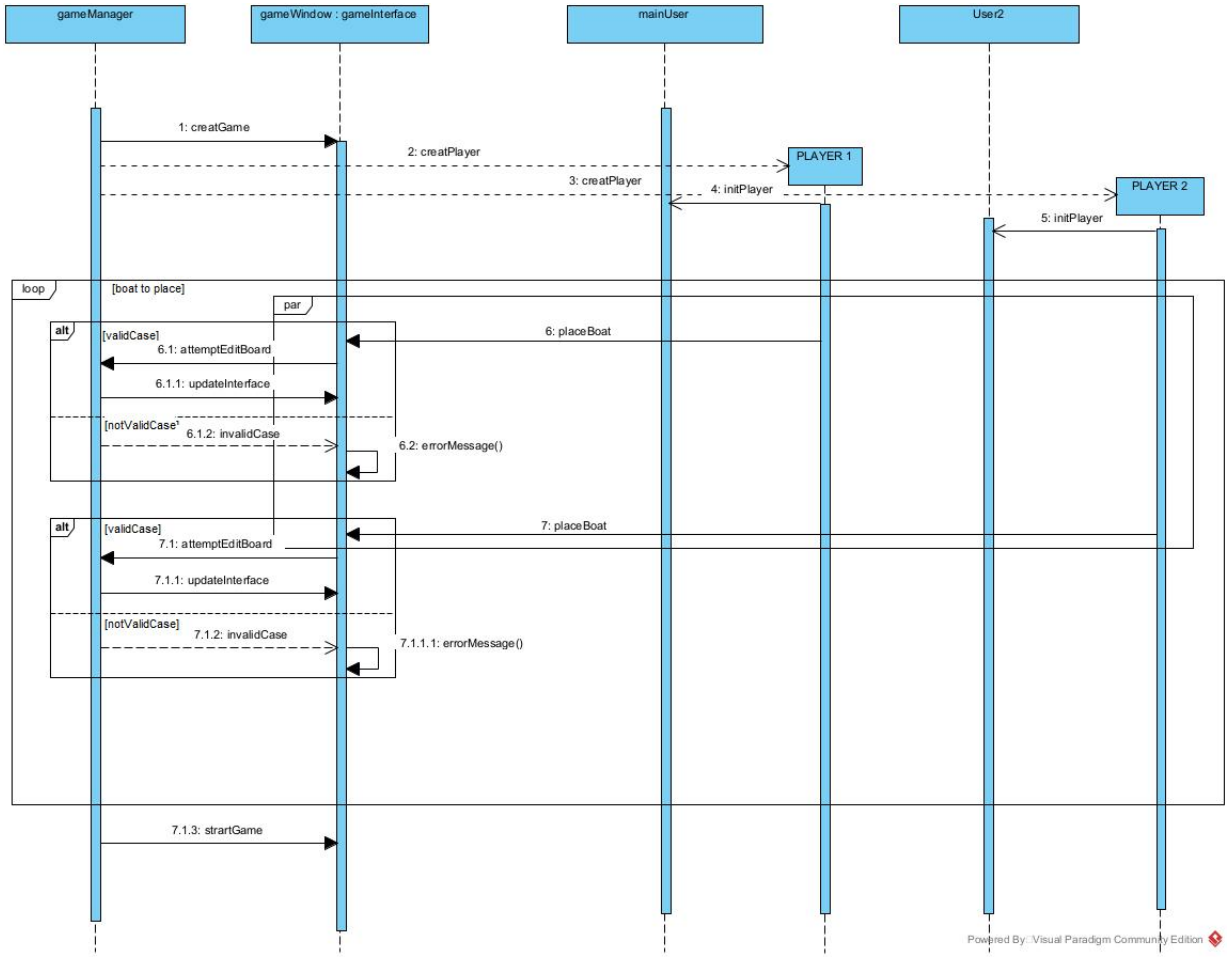

FIGURE 4.15 – Avant que la partie commence

4.2.7 Boucle de jeu

La déroulement de la partie se passe au tour par tour. A tour de rôle le user transmet son action mais l'action est gérée par player. Cela permet à une classe player d'avoir des points d'énergie, des actions et faire partie d'une faction. L'action est d'abbord vérifiée pour savoir si la case ciblée est autorisée. Si c'est le cas le plateau est modifié et l'interface est mise à jour.

Vu que nous utilisons une architecture RESTful pour la communication client-serveur, le client doit faire du polling pour savoir si c'est à son tour de jouer. C'est à dire que le client doit envoyer une requête pour savoir si c'est à son tour de jouer. Si c'est le cas, le serveur renvoie une réponse positive, sinon, une réponse négative.

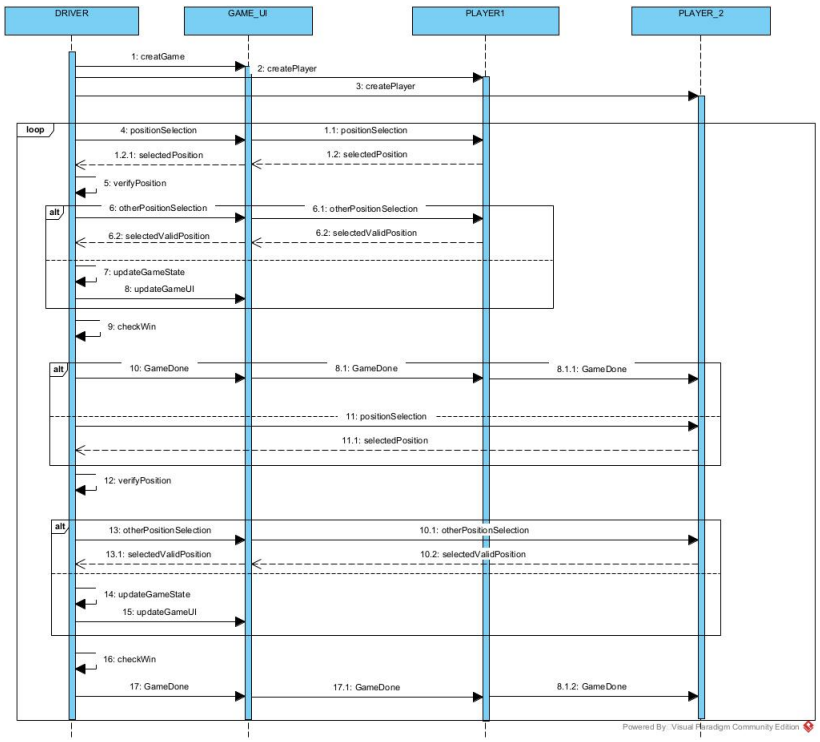

FIGURE 4.16 – Boucle de jeu

4.2.8 Gestion des comptes

Afin de gérer tous les comptes utilisateur, leurs liste d'ami, et les messages privés, nous utilisons un moteur de base de donnée. Cela nous permet de stocker toutes les informations de manière efficace. Dans notre cas, nous utilisons une base de donnée relationnelle, à savoir SQLite. La plupart des informations qui sont fournies par le client sont stockées dans cette dernière.