heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Rapport Intermédiaire

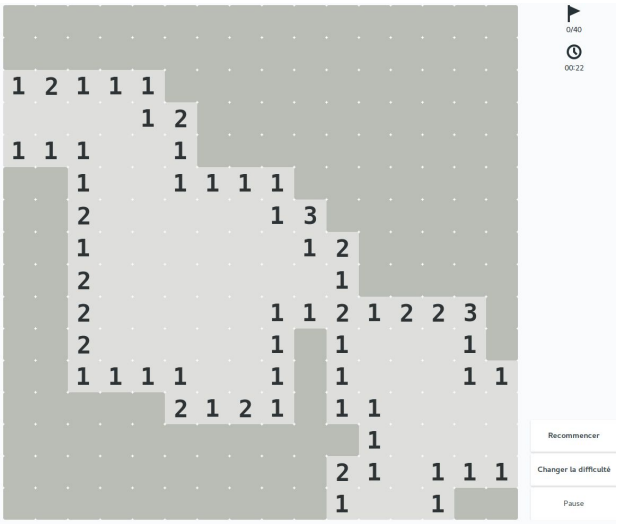
Démineur Multijoueur

Corentin Basler, Antonio Cusanelli, Marc Labie & Simon Jobin

Version 1.3, 26.05.2018

Etudes préliminaires

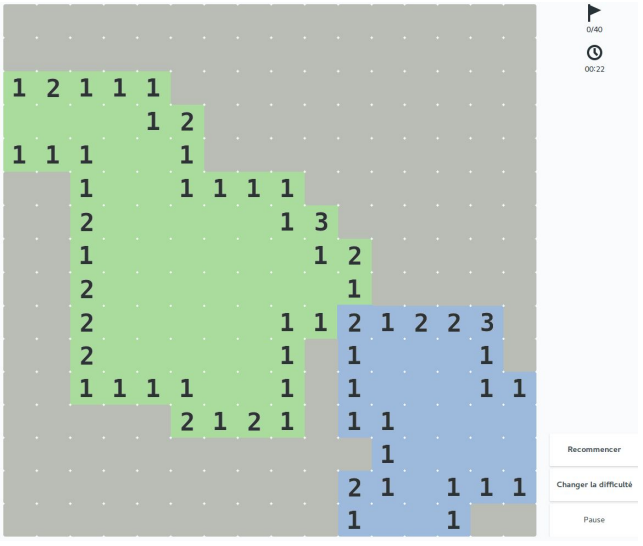
Description du projet[1]



Le démineur est un jeu qu'on ne présente plus. Il a parcouru les âges et est encore aujourd'hui présent sur beaucoup de plateforme, parfois même préinstallé avec un OS. La raison du succès constant de ce jeu de réflexion est peut-être due au fait qu'il a su rester simple et garder les mêmes règles au cours des années.
Nous souhaitons, dans le cadre de ce projet, créer notre propre démineur en Java, en lui apportant une modification : le multijoueur.

Règles

Le principe est le suivant : Il y a une machine hôte, qui héberge une partie, puis, un certain nombre de joueur peut rejoindre la partie (2 à 4 joueurs). La partie ne pourra commencer que lorsque le nombre de joueurs qui l'ont rejoint ait atteint le nombre de joueur minimum. Ensuite, les joueurs pourront commencer à chercher les cases sans mine, toutes sur la même grille. De ce fait, lorsqu'une case sera découverte par un joueur, les autres la verront aussi. Une couleur de case est attribuée à chaque joueur avant le début de la partie, comme le montre la figure à droite (case verte découverte par le joueur 1, case bleue par joueur 2).



[1] Image issue du démineur ubuntu

Gare cependant à ne pas tomber sur une mine. Comme dans le démineur classique, ces dernières sont fatales au joueur. Une partie est terminée si :

- il ne reste plus de joueurs (tous ont pris une bombe)
- il ne reste plus de case à découvrire
- si tous les joueurs quittent la partie.

Le gagnant est déterminé alors selon un système de score. Il y a trois choix de calcul de score disponible, il est défini dans la configuration de la partie.
Le premier calcul consiste à comptabiliser le nombre de cases découvertes par joueur.

Le deuxième calcul comptabilise la valeur inscrite sur la case. Si une case se trouve autour de 3 bombes, sa valeur sera de 3. Si elle est autour de 1 bombe, sa valeur de 1, et si autour d'aucune, sa valeur sera de 0. Cela incitera les joueurs à chercher la difficulté.

Le troisième calcul sera le premier et le second réunit. C'est à dire qu'on comptabilise à la fois le nombre de case et la valeur. Ce sera le mode par défaut.

Le système de drapeau est aussi conserver. Cependant, ils ne sont pas partagé. C'est à dire que si un joueur veut ajouter un drapeau sur une case où il sait qu'une mine s'y cache, sa découverte n'est pas partagée aux autres joueurs.

Bonus et Malus

On pourrait entrevoir aussi un système de bonus et de malus. Certaines cases pourrait offrir un bonus au joueur qui la découvre, comme par exemple une Vie supplémentaire, lui offrant la possibilité de se faire exploser pour le fun une fois durant la partie sans que cela ne le fasse perdre. Inversement, on pourrait aussi dissimuler des malus sur certaines cases, qui pourrait par exemple retirer une vie au joueur qui la découvre si ce dernier en possède plus d'une. Une autre idée serait aussi de leur indiquer la présence de 1-2 mines sur la grille en ajoutant un drapeau, ou alors d'en retirer 1-2 qu'il aurait placé jusque là.
D'autres idées pourraient venir s'additionner à celle-la.

Rôles

Joueur

Un joueur peut : rejoindre un lobby, quitter un lobby et jouer au jeu. Pour rejoindre un lobby, ce dernier doit avoir été créé par l'administrateur. Une fois dans ce lobby, il peut le quitter s'il souhaite. Lorsqu'il est dans le lobby, le joueur peut indiquer qu'il est prêt à jouer. Lors d'une partie, c'est un simple joueur qui peut effectuer toutes les actions nécessaires au jeu.
Mais un lobby, c'est quoi ?
Un lobby est une salle d'attente dans laquel se retrouve les joueurs et l'administrateur de la partie avant le début de la partie. Ils y configurent la partie, et attendent l'arrivée de tous les joueurs.

Administrateur

Un administrateur peut effectuer les mêmes actions qu'un joueur. Il peut en plus initier un lobby, configurer une partie (taille de la carte, pourcentage de mines, nombres de joueurs, durée de la partie). Il peut également démarrer la partie une fois que le nombre de joueurs requis est suffisant, et que tous les joueurs sont prêts.

Serveur

Le rôle du serveur est d'héberger un lobby ainsi que la partie. C'est également lui qui va terminer la partie lorsque le temps est écoulé ou que toutes les cases ont étées découvertes.

Implémentation

les joueurs n'auront qu'une "Vue" sur leur ordinateur. Le contrôleur et le modèle du jeu serait sur le serveur, et ce dernier enverra quelle information à afficher sur la vue des joueurs, en fonctions des actions effectuées.
Cela empêchera ces dernier d'aller fouiller en mémoire pour savoir où se situe les mines.
L'emplacement des mines, bonus et malus sera généré aléatoirement par le serveur au début de la partie, en fonction des configurations définies par l'administrateur de la partie.
De plus, les actions effectué par les joueurs au cours de la partie seront traitée une à la suite de l'autre (FIFO).
Avec cette implémentation, nous éviterons de plus toute collision entre les actions des joueurs.

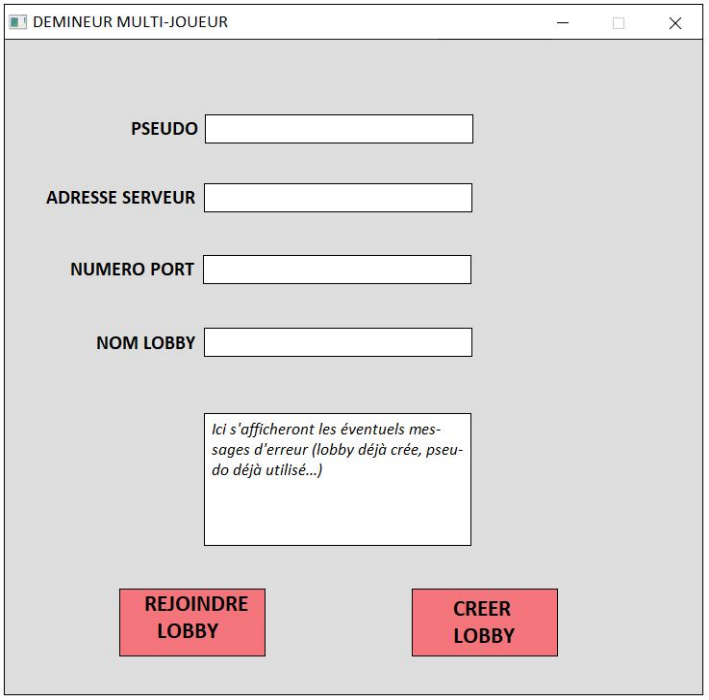
Stockage des données

Des configuration pré-enregistrée pourront être chargée pour configurer une partie. Ces dernières seront sauvegardée côté client, car chaque joueur pourra créer sa propre configuration.
Ce fichier sera sauvegardé au format XML. Il contiendra notamment des données telle que le mode de calcule de score, le nombre de joueur, la taille de la grille, la difficulté, etc…
Lors de la création d'un lobby, l'administrateur pourra chargé sa configuration s'il le souhaite, et la partie sera configurée en fonction.

Interfaces

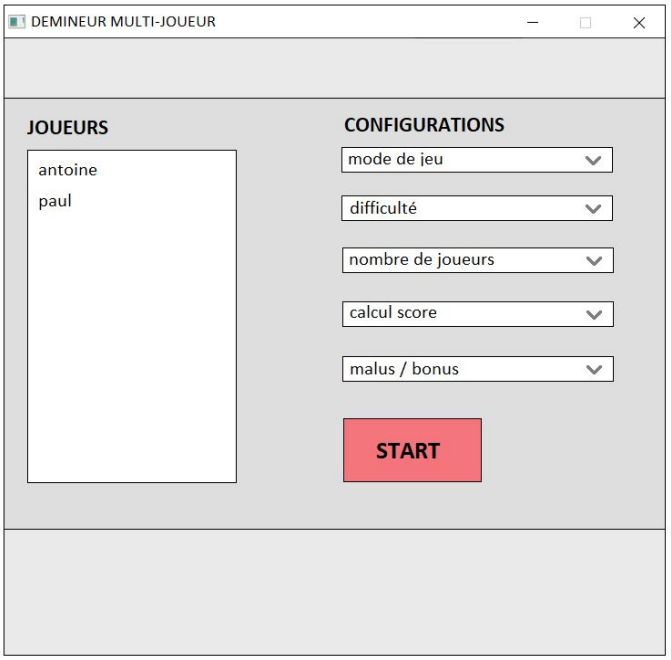
Fenêtre de login

La fenêtre principale permettra à un joueur d'entrer son pseudo, de spécifier une adresse IP et un numéro de port dans le but de créer un lobby ou de rejoindre un lobby déjà existant. Cette fenêtre se présentera sous la forme suivant (simplifiée):



Fenêtre du lobby

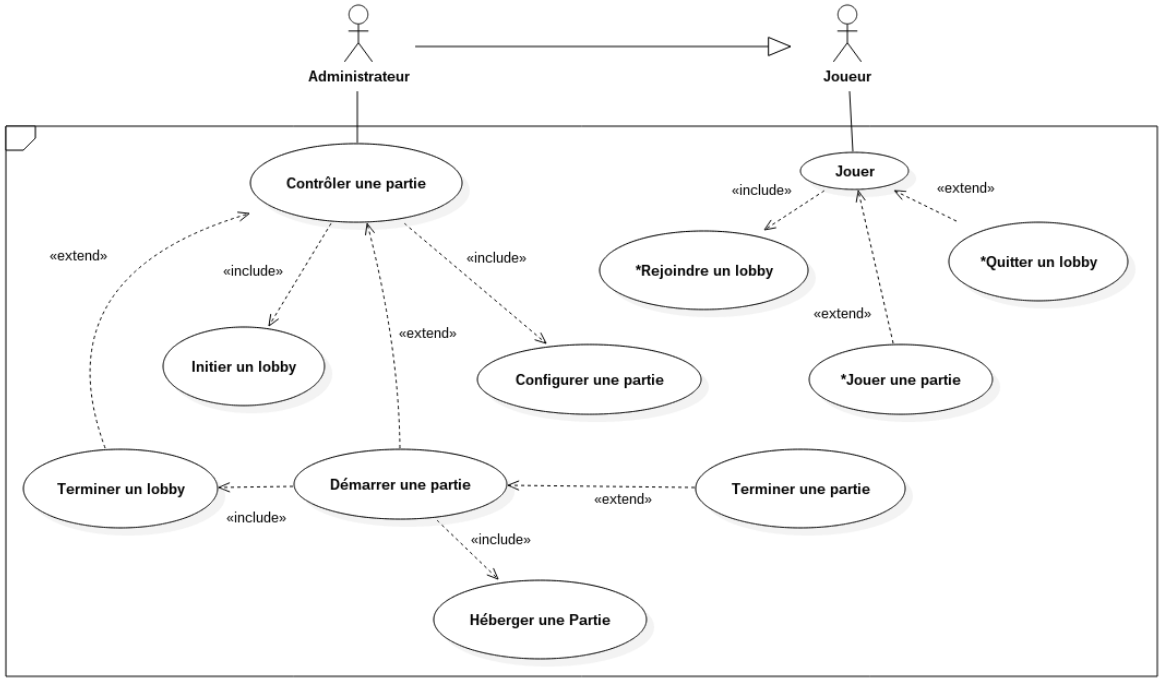
L'administrateur aura la possibilité de choisir les paramètres de la partie dans le menu de configuration. Il lui sera également possible d'importer une configuration prédéfinie enregistrée au préalable. Les joueurs qui auront rejoint la partie apparaîtront dans une liste. La fenêtre du lobby ressemblera à la figure suivante :



Interface du jeu

L'interface graphique se rapprocherait de celle présentée plus haut (description du projet). On y ajouterait encore un tableau affichant le score de chaque joueur trier, pour avoir lequel est en tête, et avoir un suivi de la partie pour chaque joueur.

Diagramme des cas



Cas d'utilisations

Jouer

Scénario principal :

1. Le joueur doit lancer le client
2. il doit rejoindre un lobby : include Rejoindre un lobby
   Prérequis : pour rejoindre un lobby, ce dernier doit être initié par l'administrateur.
3. Le joueur joue la partie

Rejoindre un lobby

Scénario principal :

1. Le joueur se connecte au serveur
2. Le joueur est placé dans un lobby ouvert sur le serveur.

Prérequis : Le lobby doit être initier.

Quitter un lobby

Scénario principal :

1. Le joueur peut quitter un lobby.
2. Le joueur reste connecté au serveur et peut rejoindre un autre lobby.

Jouer une partie

Scénario principal :

1. Le joueur peut cliquer sur une case de la carte
    a. Si c'est une mine il a perdu
    b. Si ce n'est pas une mine, la case est découverte et le score comptabilisé
2. Le joueur peut également marqué la case s'il pense qu'il y a une mine dessous.

Contrôler une partie

Scénario principal :

1. L'administrateur peut contrôler tous les aspects d'une partie.

Configurer une partie

Scénario principal :

1. L'administrateur peut configurer une partie.
    a. L'administrateur peut choisir la taille de la carte
    b. Il peut choisir le pourcentage de mines
    c. Il peut choisir le nombre de joueurs
    d. Il peut choisir le mode de jeu

Initier un lobby

Scénario principal :

1. L'administrateur peut initier un lobby.

Démarrer une partie

Scénario principal :

1. L'administrateur peut démarrer une partie
   Prérequis : la partie doit être configurer
2. Lorsque la partie est démarrée, le lobby est terminé.

Terminer un lobby

1. L'administrateur pour terminer un lobby
    a. Soit il le termine manuellement
    b. Soit il se termine automatiquement au lancement de la partie

Terminer une partie

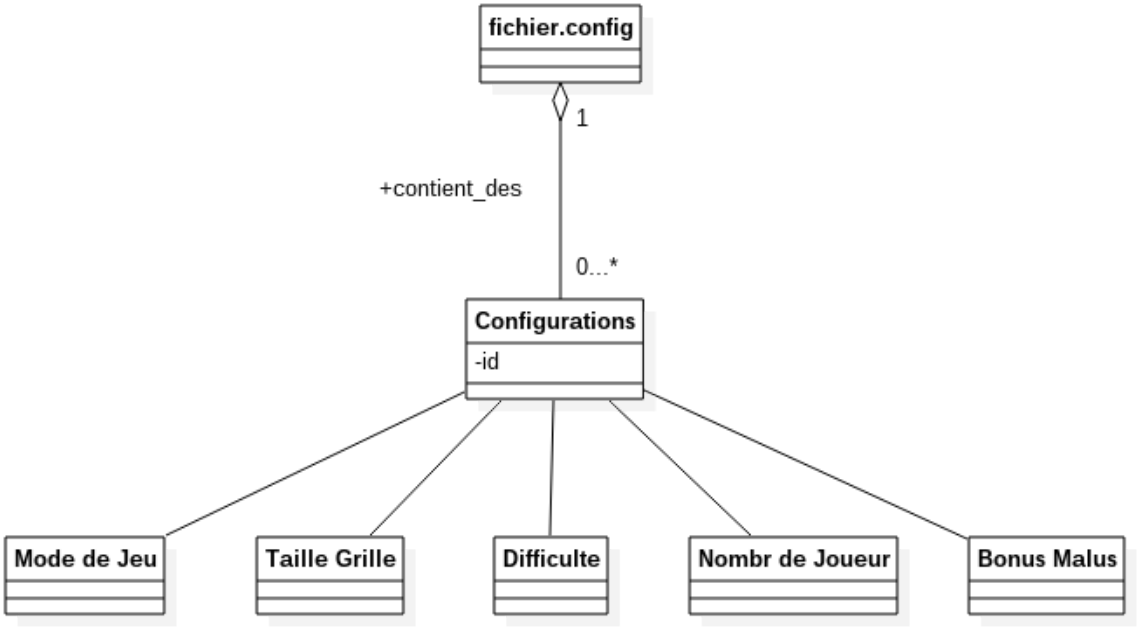
1. Lorsque la partie est finie (selon le mode de jeu choisi), la partie se termine

Héberger une partie

1. Au lancement de la partie par l'administrateur, la partie est automatiquement hébergé par le serveur

Modèle conceptuel

Comme présenté dans notre étude préliminaire, nous avons opté pour un stockage de configuration de partie pré-enregistrée dans un fichier XML côté client. De cette manière, ces configurations seront persistantes dans le temps, un administrateur pourra y accéder pour configurer sa partie.
Voici le modèle conceptuel du fichier XML :



Responsabilités client-serveur

Le client ainsi que le serveur auront tout deux des responsabilités à assumer pour que le déroulement d'une partie se fasse de la meilleur des manières.

Responsabilités côté client

Le client de son côté aura pour rôle de transmettre la plupart des actions de l'utilisateur au serveur de manière à ce que celui-ci puisse les traiter, pour autant qu'il en est l'autorisation. Il y aura aussi certaines actions qu'il sera en mesure de traiter lui-même, comme par exemple la création d'une configuration.
Il devra donc aussi être capable de traiter les réponses du serveur et de les transmettre à l'utilisateur de manière compréhensible pour que ce dernier comprenne quelles ont été les répercussions suite à son action. Cela se fera à l'aide d'une interface graphique, présenté plus haut.

Responsabilités côté serveur

Le serveur aura lui bien plus de responsabilité à assurer que le client. En effet, ce dernier devra faire en sorte de traiter plusieurs requête simultanément. Il sera donc multi-threadé. Cela signifie qu'il devra prendre garde à ce qu'une ressource ne soit pas modifiée simultanément par deux utilisateur, par exemple, si un utilisateur souhaite déminer une case alors que le serveur est déjà en train de la déminé pour un autre utilisateur, il devra s'assurer que le thread en pleine action termine son déminage avant que celui désirant lui aussi déminer la case ne puisse essayer de la déminer. Il devra donc s'assurer que la synchronisation des actions se fasse de manière à ne laisser aucun conflit dans la partie en cours.
De plus, il sera aussi responsable de mettre à jour tous les joueurs de la partie du statut actuel de cette dernière, ce qui implique :

- Le score actuel et l'état de chaque joueur
- L'état actuel de la grille
- signaler aux joueur qu'un autre joueur a quitté la partie

Ces derniers points cependant ne concerne qu'une partie en cours. Il devra transmettre d'autre informations avant le début d'une partie, comme :

- Indiquer qu'un joueur a rejoint un lobby
- indiquer qu'un joueur a quitter un lobby
- indiquer que lobby a été ouvert/fermé

- indiquer qu'une configuration a été modifiée

Il devra aussi assurer qu'aucune commande ne soit traiter à un moment où elle ne le devrait pas, comme modifier la taille d'une grille en pleine partie, ou déminer une case alors qu'une partie n'a pas encore commencer.
Il devra aussi faire attention à ce qu'uniquement l'administrateur de la partie puisse effectuer certaine action privilégiée, comme modifier la configuration d'une partie, expulser un joueur, ou démarrer la partie.
Il déterminera aussi quand une partie sera terminé, est sera responsable de fermer le lobby lorsqu'une commence.

Protocole

Code de confirmation

Lors de la réception d'une réponse du serveur, cette dernière sera précédée d'un code de confirmation pour indiquer à l'utilisateur le statut de sa commande. Ce code permettra au logiciel de simplifier les démarche à faire en fonction de celui-ci.
Nous nous sommes inspiré de ceux utilisé par le protocole SMTP.
Ces codes de confirmations peuvent encore subir des modifications, mais dans l'idée, ils correspondront à ceux-ci.

Code de confirmation	
Code	**Signification**
220	Code envoyé par le serveur lors de la connexion et de la déconnexion.
250	Confirmation de commande acceptée.
350	Information importante à transmettre au joueur.
450	La commande existe, mais le joueur n'a pas le droit de l'effectuer.
550	La commande existe, mais les paramètres envoyés ne sont pas autorisé, sans insuffisant, ou alors la commande ne peut pas encore être exécutée pour l'instant par le serveur.
650	La commande existe, les paramètres sont corrects, mais le lobby ou la partie n'ont pas accepté la requête.
750	La commande n'existe pas.

Commande

Les commandes ci-dessous seront celle envoyé par le client au serveur pour effectuer une action. Chacune des réponses du serveur sera précédée d'un code de confirmation spécifié plus haut. Pour savoir duquel il s'agira, veuillez vous référer au code couleur ci-dessus.
De plus, le serveur pourra répondre à la fois à la personne qui lance la commande, à tous les joueurs du lobby/de la partie, ou encore à un joueur bien spécifique.
Les réponses adressés à tous les joueurs apparaissent sur un fond bleu clair.

N'étant qu'en phase de conception, ces commandes peuvent encore subir des modifications au cours du développement.

<table>
<tr><th colspan="3">Commande Lobby</th></tr>
<tr><th>Commande</th><th>Action du serveur</th><th>Réponse du serveur</th></tr>
<tr><td rowspan="2">CREATE LOBBY

Paramètres :

<lobby_name>
<player_name></td><td rowspan="2">Crée un lobby avec pour nom <lobby_name>. Assimile le rôle d'administrateur de la partie au créateur du lobby.
Joint automatiquement l'administrateur au lobby, en lui donnant le nom de <player_name>.</td><td>Retourne
LOBBY CREATED: si le lobby a été créé
LOBBY NAME NOT AVAILABLE : si le nom du lobby est déjà utilisé
NOT ENOUGH ARGUMENTS: si l'utilisateur n'a pas passé assez d'argument en paramètre.
TOO MANY ARGUMENTS: si l'utilisateur a pas passé trop d'argument en paramètre.
ALREADY IN A LOBBY : Si l'utilisateur se trouve déjà dans un lobby.</td></tr>
<tr><td>Le joueur créant le lobby se verra informé de l'état actuel du lobby, à l'aide de plusieurs réponse ayant comme statut 350. Ces informations comprendront la configuration actuel du lobby.</td></tr>
<tr><td rowspan="2">OPEN LOBBY</td><td rowspan="2">Ouvre le Lobby aux autres joueurs. Ils pourront ainsi s'y connecter.</td><td>Retourne
OK: si le Lobby a été ouvert
NO LOBBY CREATED : Si vous n'avez pas créer de lobby
ACTION DENIED: si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie</td></tr>
<tr><td>LOBBY OPENED: pour indiquer aux joueurs que le lobby est ouvert</td></tr>
<tr><td rowspan="2">CLOSE LOBBY</td><td rowspan="2">Ferme le Lobby aux autres joueurs. Ils ne pourront ainsi plus s'y connecter.</td><td>Retourne
OK : si le Lobby a été fermé
NO LOBBY CREATED : Si vous n'avez pas créer de lobby
ACTION DENIED: si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie</td></tr>
<tr><td>LOBBY CLOSED: pour indiquer aux joueurs que le lobby est fermé</td></tr>
<tr><td>JOIN LOBBY

Paramètres :

<lobby_name>
<player_name></td><td>Ajoute le joueur souhaitant rejoindre le lobby au lobby <lobby_name> s'il y reste de la place et si <player_name> est encore disponible. Si c'est le cas, le joueur prendra la nom <player_name>.</td><td>Retourne
LOBBY JOINED: si le joueur a été ajouté au lobby
LOBBY FULL : Si le lobby est complet
LOBBY CLOSED : Si le lobby est fermé
NOT ENOUGH ARGUMENTS: si l'utilisateur n'a pas passé assez d'argument en paramètre.</td></tr>
</table>

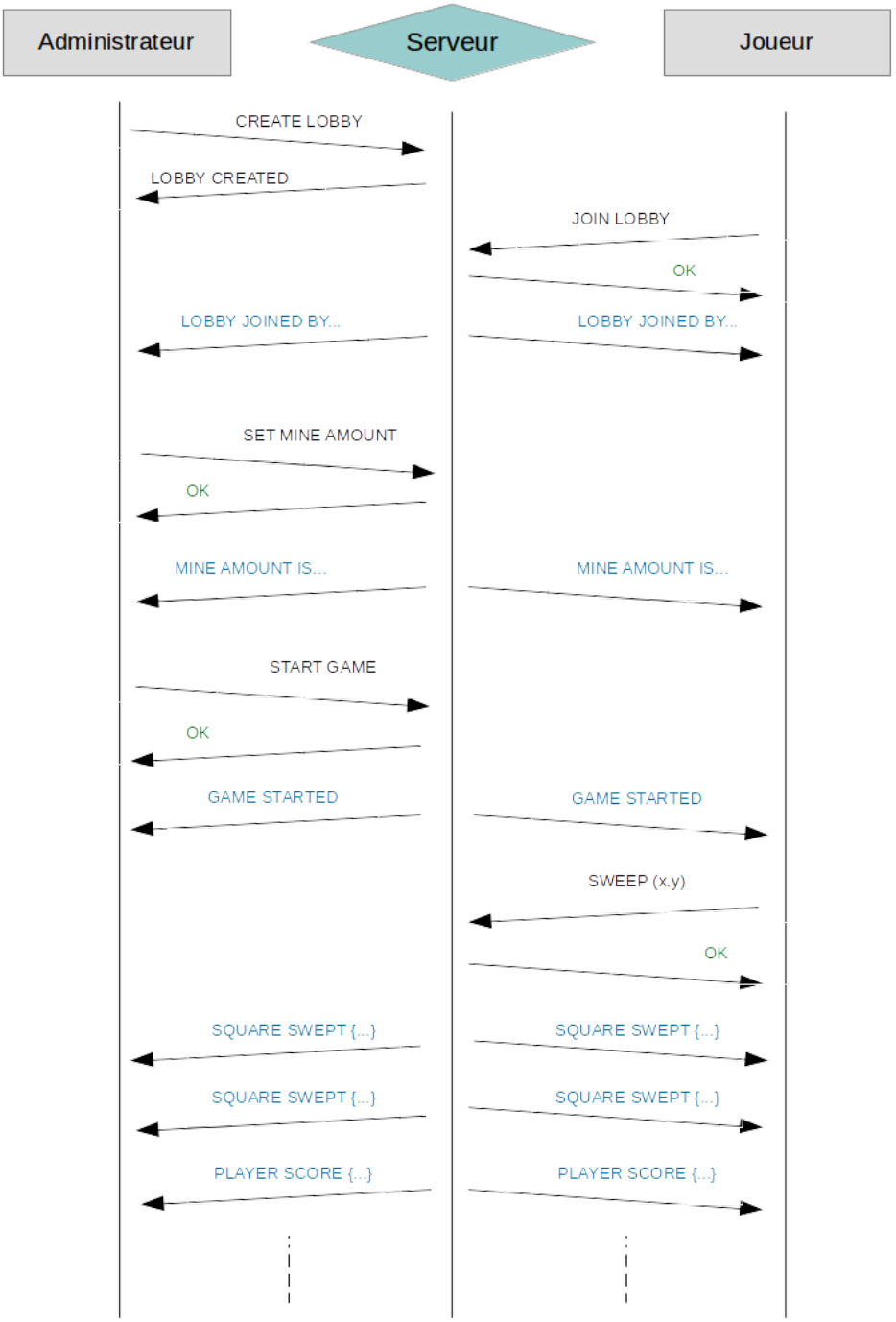
		TOO MANY ARGUMENTS: si l'utilisateur a pas passé trop d'argument en paramètre. **LOBBY NOT FOUND**: Si le lobby n'existe pas **PLAYER NAME NOT AVAILABLE** : si **<player_name>** est déjà utilisé par une autre joueur **ALREADY IN A LOBBY** : Si l'utilisateur se trouve déjà dans un lobby.
		LOBBY JOINED BY : <player_name>: pour indiquer aux joueurs que le joueur **<player_name>** a rejoint le lobby.
		Le joueur rejoignant le lobby se verra informé de l'état actuel du lobby, à l'aide de plusieurs réponse ayant comme statut 350. Ces informations comprendront les configuration actuel, ainsi que les joueurs actuellement dans le lobby.
QUIT LOBBY	Supprime le joueur du lobby. Si c'est l'administrateur qui quitte le lobby, cela supprimera le lobby.	Retourne **OK :** si le joueur a été supprimé du lobby. **NO LOBBY JOINED** : Si aucun lobby n'a été rejoints.
		LOBBY LEFT BY : <player_name>: pour indiquer aux joueurs que le joueur **<player_name>** a quitté le lobby.
EXPEL LOBBY Paramètre : <player>	Supprime le joueur **<player>** du lobby. Cette commande ne peut être envoyée que par l'administrateur.	Retourne **OK :** si le joueur a été supprimé du lobby **PLAYER NOT FOUND** : Si aucun joueur **<player>** n'a été trouvé dans le lobby. **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie.
		LOBBY LEFT BY : <player_name>: pour indiquer aux joueurs que le joueur **<player_name>** a quitté le lobby.
		YOU HAVE BEEN EXPELLED: pour indiquer au joueur concerné qu'il a été expulsé.
Commande de Configuration		
Commande	**Action du serveur**	**Réponse du serveur**
SET SCORE MODE Paramètre : <mode>	Configure le mode de calcule de score de la partie.	Retourne **OK :** si le mot a été configuré **MODE NOT FOUND** : Si le mode de calcul de score transmis n'est pas implémenté par le serveur **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est

		pas l'administrateur de la partie
		SCORE MODE IS : <mode>: pour indiquer aux joueurs que le mode de calcule de score est le mode **<mode>**.
SET MINE PROPORTION Paramètre : <%>	Configure la proportion de mine en **<%>** que comportera notre champ de mine lorsque la partie sera lancée.	Retourne **OK** : pour confirmer que la configuration a été prise en charge. **MINE PROPORTION NOT ALLOWED**: Si le % de mine n'est pas autorisé par le serveur, par exemple s'il est trop faible ou trop élevé. **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie
		MINE PROPORTION IS : <%>: pour indiquer aux joueurs que le nombre de mine a été fixé à **<%>**.
SET SIZE Pramètres : <width>X<height>	Configure la taille du champ de mine avec une largeur **<width>** et une hauteur **<height>**.	Retourne **OK :** si la taille a été configurée **SIZE NOT ALLOWED** : si la taille est trop grande ou trop petite, ou ne correspondant pas à celle préconfigurée par le serveur **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie
		SIZE IS : <width>X<heigth>: pour indiquer aux joueurs que la taille de la a été fixé à **<width>**X**<height>**.
SET PLAYER AMOUNT Paramètre : <amount>	Configure le nombre de joueur qu'il y aura dans la partie. Si le nombre de joueur est plus petit que le nombre de joueur dans le lobby, les joueurs en trop ayant rejoint la partie en dernier seront expulsés du lobby.	Retourne **OK :** si le nombre de joueur a été configuré **PLAYER AMOUNT NOT ALLOWED** : si le nombre de joueur est trop grande ou trop faible **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie
		PLAYER AMOUNT IS : <amount>: pour indiquer aux joueurs que le nombre de joueur est **<amount>.**
ENABLE BONUS-MALUS	Active ou désactive les bonus et malus.	Retourne **OK :** si les bonus et malus ont été activé/désactivé **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie
DISABLE BONUS-MALUS		**BONUS-MALUS ENABLED/DISABLED**: pour indiquer aux joueurs que les bonus et malus ont été activé/désactivé.
Commande Partie		

Commande	Action du serveur	Réponse du serveur
START GAME	Démarre la partie avec les configurations préconfigurée.	Retourne **OK :** si la partie peut commencer **ACTION DENIED** : si le joueur demandant à effectuer cette action n'est pas l'administrateur de la partie
		GAME STARTED: pour indiquer aux joueurs que la partie a commencé.
SWEEP Paramètre : (<x>,< y>)	Démine la case au coordonnée (x,y) du champ de mine et attribues la case au joueur. De plus, si certaines des cases voisines n'avoisines aucune mine, elle seront déminée elle aussi, ce qui peut démarrer une action en chaîne. Toute ces cases seront attribuée au joueur.	Retourne **OK :** si la case a étée déminée **SQUARE ALREADY SWEPT**: si la case a déjà étée déminée **YOU HAVE ALREADY LOST**: si le joueur est déjà discalifié **SQUARE NOT FOUND**: si la case ne se trouve pas sur le champ de mine.
		Le serveur répondra ici à tous le monde avec plusieurs réponses : **SQUARE SWEPT {...}**: pour indiquer aux joueurs qu'une case a été balayée. Cette réponse contiendra au format Json le contenu d'une case, soit : - ses coordonées - sa valeurs (0 à 8) - Le joueur qui l'a déminé - s'il s'agit d'une mine ou non - le bonus/malus qu'elle contient Donc si une case déminée démarre une action en chaîne, les joueurs vont recevoir plusieurs de ces réponses. **PLAYER SCORE {...}**: pour indiquer aux joueurs le score d'un joueur. Cette réponse sera au format Json, et contiendra : - Le nom du joueur - son score - son état (alive/dead). Après chaque déminage d'une case ou à la fin d'un déminage en chaîne, le score des joueurs sera à nouveau transmis aux joueurs.
QUIT GAME	Bloque le score du joueur sortant. Ce dernier aura perdu d'office. Le joueur sera de plus déconnecté du serveur. S'il ne reste plus aucun joueur dans la partie, le serveur termine la partie.	Retourne **OK :** si le joueur a été disqualifié **NOT IN A GAME**: si le joueur ne se toruve pas dans une partie.
		PLAYER LEFT <player_name>: pour indiquer aux joueurs que le joueur **<player_name>** a quitté la partie.
		GAME FINISHED <winner>: Ce message sera envoyé aux joueurs lorsqu'une partie sera terminée. Il indiquera qui est le gagnant **(<winner>).**

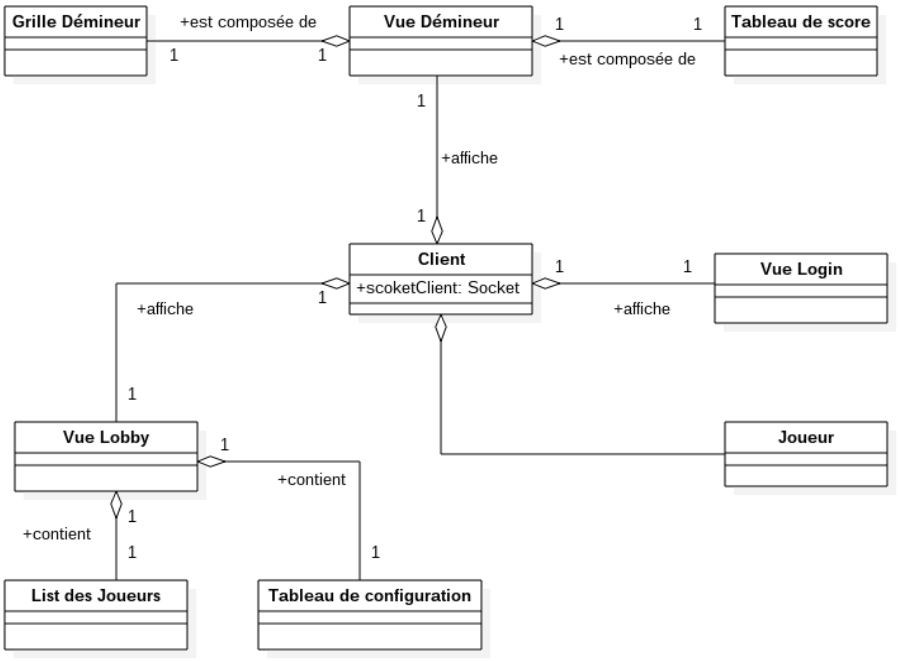
Exemple d'échange client-serveur

On peut voir sur la figure suivant un exemple d'échanger qui serait effectuée entre 2 joueurs, dont l'un serait l'administrateur, et le serveur, pour créer et jouer une partie. Toutes les commandes n'ont pas été utilisée, mais ce diagramme n'est là qu'à titre d'exemple.

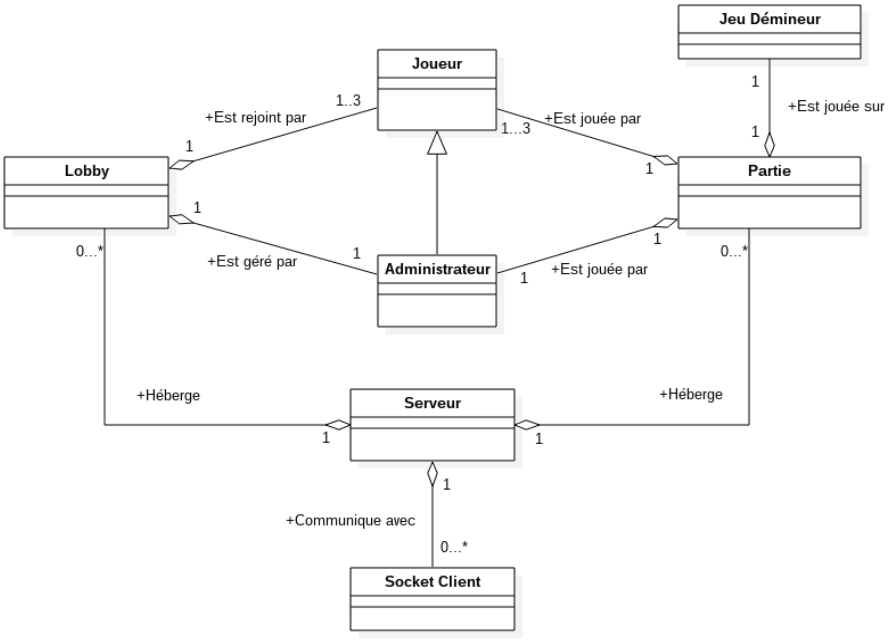


Modèles de domaine Client & Serveur

Client



Serveur



Backlog de produit & Plan d'itération

Plan d'itération :

Itération 1:

A la fin de la première itération, plusieurs application client devrait pouvoir être lancé et communiqué avec un serveur. Ce dernier devrait pouvoir différencier les différentes commandes du protocole, sans pour autant exécuter une action.

Itération 2:

A la fin de cette itération, l'implémentation du code nécessaire pour représenter un lobby sera mis en place, ainsi que celle du jeu du démineur (les deux côtés serveur).

Itération 3:

A la fin de cette itération, les applications clients doivent pouvoir être en mesure de créer un lobby sur le serveur, ce qui aura pour effet d'ouvrir une interface graphique représentant le client. L'interface graphique de connexion de base sera donc aussi disponible.

Itération 4:

A la fin de cette itération, nous aurons implémenté la possibilité à un utilisateur de rejoindre et quitter un lobby, dans son cas d'utilisation "jouer".

Itération 5:

A la fin de cette itération, il sera possible aux joueurs de débuter une partie, accompagnée de son interface graphique, avec un mode de calcule de score par défaut.

Itération 6:

A la fin de cette itération, le serveur gérera automatiquement la terminaison d'une partie et d'un lobby. De plus, si nous n'avons pas de retard, nous devrions être en mesure d'ajouter des bonus et malus.

Itération 7:

Pour la dernière itération, l'administrateur sera en mesure de configurer sa partie manuellement, ou à partir d'un fichier xml charger sur son ordinateur.
Nous terminerons nos rapport, manuel, et documentation.

Le backlog est disponible en page suivante.

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
3	Connexion client-serveur			Technical story	Réalisation d'un client et d'un serveur. réalisation du la communication client-serveur : - établissement du protocole - définition des ports et adresse ip - gestion de l'intelligence du programme				Antonio Cusanelli	Développement général
7	Démarrer et héberger une partie			User story	Administrateur peut démarrer une partie, elle est automatiquement hébergée par le serveur.				Antonio Cusanelli	Développement du serveur
11	Interface graphique rejoindre lobby			User story	Partie de la GUI qui permet au A[1-Joueur] de joindre ou créer un lobby				Antonio Cusanelli	Développement du client

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
16	Jeu			Technical story	Implémentation du jeu-même : * représentation de la grille (tableau) * action sur la grille * algo permettant de remplir les grilles d'éléments et le répartir.				Antonio Cusanelli	Développement général
1	Jouer et accéder à un lobby			User story	As a A[1-Joueur] A[2-Administrateur] I want to démarrer le programme et accéder un lobby In order to jouer				Antonio Cusanelli	Développement du client
17	Rapport, documentation, manuel			Technical story	Rédaction du rapport, des différentes documentations : manuel utilisation, déploiement.				Antonio Cusanelli	Rédaction du rapport
9	Terminer une partie			User story	La partie est terminé lorsque c'est la fin du jeu ou lorsque l'A[2-Administrateur] décide de quitter le lobby.				Antonio Cusanelli	Développement du client

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
5	Création de configurations de partie			User story	Conception permettant au A[2-Administrateur] de créer une configuration de partie selon les options possible : * taille de la grille * difficulté (pourcentage de mine) * calcul du score * activation de l'option bonus/malus				Antonio Cusanelli	Développement du client
20	Gestion bonus/malus			User story	Intégration du système bonus/malus au jeu				Antonio Cusanelli	Développement général
13	Implémentation du lobby			Technical story	Ajout de la partie lobby au serveur pour la gestion d'attente des joueurs				Antonio Cusanelli	Développement du serveur
6	Initier (créer) un lobby			User story	As a A[1-Joueur], qui deviendra A[2-Administrateur] de la partie : celui qui crée un lobby est A[2-Administrateur], I want to créer un lobby In order to choisir une configuration et lancer une partie				Antonio Cusanelli	Développement du client
19	Interface graphique de la partie			User story	Les A[1-Joueur], A[2-Administrateur] peuvent interagir avec l'interface : cliquer sur une case, quitter, etc.... afin de jouer				Antonio Cusanelli	Développement du client

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
14	Interface graphique lobby			User story	Partie de la GUI qui permet au A[1-Joueur] d'attendre le nombre de joueur demandé				Antonio Cusanelli	Développement du client
4	Quitter un lobby			User story	Le A[1-Joueur] peut quitter le lobby mais la partie continue. l'A[2-Administrateur] peut quitter le lobby et la partie se termine.				Antonio Cusanelli	Développement du client
18	Implémentation fichier de configuration xml -- base de données			Technical story	Système lecture écriture permettant d'interagir entre le programme et le fichier				Antonio Cusanelli	Base de données
8	Terminer un lobby			User story	L'A[2-Administrateur] peut une fois la partie terminée, terminer un lobby ce qui engendra sa suppression.				Antonio Cusanelli	Développement du client

Bilans d'itération

Première itération

ScrumMaster :

1. Terminaison des histoires
   Histoires "Connexion clients-serveur" et "implémentation de l'interface graphique rejoindre lobby" planifiées, toutes les deux sont terminées correctement.

2. vélocité du sprint : 18 (selon iceScrum)

3. Replanification : Il n'y pas de replanification prévue étant donné que les tâches prévues sont terminées.

4. Commentaire général : Tout s'est passé comme prévu.

Bilans personnels :

Marc Labie : La mise en place du serveur s'est faite assez facilement. J'ai repris en grande ligne ce que nous avons appris dans le cours RES.

Corentin Basler : La mise en place de la classe représentant le protocole fut assez rapide. C'est pourquoi j'étais disponible pour aider mes camarades.

Simon Jobin : Malgré la simplicité graphique de la fenêtre d'accueil, le prise en main de Gradle et JavaFX m'a pris un certain temps.

Deuxième itération

ScrumMaster :

1. Terminaison des histoires
   Histoires "jeu" et "implémentation du lobby" planifiées, toutes les deux sont terminées correctement.

2. vélocité du sprint : 18 (selon iceScrum)

3. Replanification : Il n'y pas de replanification prévue étant donné que les tâches prévues sont terminées. Il y aura des tâches en moins pour l'histoire "interface graphique de la partie" étant donné que Simon a déjà commencé à implémenter l'interface.

4. Commentaire général : Tout s'est passé comme prévu. Simon a pris de l'avance en créant un interface pour la partie jeu en partie fonctionnel. on peut y jouer individuellement.

Bilans personnels :

Marc Labie : L'itération a pris plus de temps que prévu. J'avais omis la complexité pour gérer les ressources partagées. En effet, lorsqu'on modifie une valeur que chaque joueur peut modifier, il faut prendre garde à ce qu'aucun autre joueur ne puisse la modifier en même temps, par exemple, si deux joueur souhaite rejoindre un lobby alors que le lobby est pratiquement complet.

Corentin Basler : Cette deuxième itération a été un peu plus longue que la première. Pour cette itération j'ai travaillé sur les classes internes du serveur, à savoir la classe de configuration. C'est un travail agréable qui c'est bien passé. J'ai cependant dû effectuer quelques recherches pour faciliter mon implémentation.

Simon Jobin : Une fois la prise en main de JavaFX effectuée, j'ai pu assez facilement prendre un peu d'avance pour représenter graphiquement le plateau de jeu ainsi que la validation de l'algorithme de répartition aléatoire des mines et celui de la découverte de la zone autour de la case cliquée.

Troisième itération

ScrumMaster :

1. Terminaison des histoires : Ce sprint a été peu productif dû au rendu du projet PRO pour la semaine prochaine. Du coup, aucune des deux histoires prévues ("Initier (créer) un lobby" et "interface graphique lobby") n'ont pas été terminée.

2. vélocité du sprint : 0 (selon scrum)

3. Replanification : les tâches restantes des histoires ont été replanifiées pour le sprint n°4.

4. Commentaire général : On a essayé d'avancer un maximum pour ce sprint, mais, malheureusement, nous avons eu peu de temps pour nous consacrer à ce sprint.

Bilans personnels :

Marc Labie :

La semaine fut plutôt chargée. Nous avons donc pas pu consacrer autant de temps que nous le voulions pour le projet. Nous avons terminer une histoire, mais nous avons dû reporter les autres pour la semaine prochaine.

Corentin Basler :
La semaine était très chargée. En effet, nous avons énormément travaillé sur le projet de semestre. C'est pourquoi, nous n'avons pas pu consacrer beaucoup de temps à ce projet.

Simon Jobin :
Devant faire face à une très grosse charge de travail liée au rendu du projet de semestre, je n'ai malheureusement pas pu effectuer ma tâche.

Quatrième itération

ScrumMaster :

1. Terminaison des histoires : L'interface graphique de lobby a été terminée. Par contre, nous avons rencontré des problèmes d'implémentation de traitement de la communication entre le client et le serveur au niveau du client. De ce fait, nous n'avons pas pu finaliser les histoires "initier (créer) un lobby", "jouer et accéder à un lobby" et "quitter un lobby".

2. vélocité du sprint : 3 selon iceScrum

3. Replanification : les méthodes citées, ci-dessus, sont replanifiées pour le sprint 5.

4. Commentaire général : Ce sprint est tombé dans la semaine du rendu du projet PRO auquel on a dû y consacrer l'entier de notre temps, et par conséquent, les problèmes rencontrés n'ont pas pu être résolu.

Bilans personnels :

Marc Labie :
Nous avions le rendu du projet de semestre cette semaine. Du coup, nous n'avons pas pu avancer autant que nous le voulions. De plus, nous avions quelque souci de référence dans notre code ce qui nous a fait perdre du temps pour le débugger.

Corentin Basler :
La semaine était encore chargée avec le rendu du projet de semestre. Nous avons donc repoussé les tâches incomplètes pour le sprint suivant.

Simon Jobin :
Ayant pris du retard au sprint 3 et la semaine étant encore chargée, nous avons repoussé les tâches incomplètes au sprint suivant. Cependant, ayant terminé l'implémentation des algorithmes du jeu au sprint 2, j'ai pu prendre un peu d'avance et les concrétiser dans une interface graphique, ce qui me rendra la suite plus aisée.