2018

Monopoly HEIG-VD

CAHIER DES CHARGES

BURGENER, CURCHOD, GONZALEZ LOPEZ, REYMOND

15 juin 2018 Version : 2.0

1 Table des matières

2 Introduction

Dans le cadre du module de Génie Logiciel (GEN), il nous a été demandé de développer un "mini-projet" implémentant une architecture client-serveur pouvant communiquer par le réseau. Après discussion nous sommes tombés d'accord sur l'idée d'un Monopoly sur le thème de la HEIG-VD.

Le client peut s'enregistrer et se connecter à l'aide d'un nom d'utilisateur et un mot de passe. Une fois authentifié, il a accès à la liste des parties en attente de joueurs. D'ici il peut rejoindre un "salon" de jeu ou en créer un. Une fois qu'il y a assez de joueurs dans le salon, et que ces derniers sont tous prêt, la partie commence.

Le serveur, lui, s'occupe de la gestion de ses clients lorsqu'ils ne sont pas en jeu, et de la gestion des parties en cours. C'est à dire que toute la logique du jeu, et l'évolution de la partie sont implémentées du côté serveur. Les clients se contentent d'interpréter les informations reçues et d'afficher l'état de la partie.

Derrière le serveur nous utilisons une base de données afin de pouvoir enregistrer les joueurs, fixer des limites de jeu (tel que le nombre de dés, ou encore le capital de départ), définir les objets inhérents au jeu (comme les cases et les cartes), et enfin nous avons également pensé à un suivi des statistiques de jeu pour les joueurs.

3 Analyse

3.1 Fonctionnement général de l'applicatif

3.1.1 Fonctionnalités de base

Pour notre jeu de société, voici les fonctionnalités de base que nous avons mises en place :

- **Authentification simple** : Nous demandons un nom d'utilisateur ainsi qu'un mot de passe à la connexion de l'utilisateur. Si l'utilisateur coche la case « Enregistrement », nous créons un compte utilisateur avec les informations renseignées.
- **Création d'un salon de jeu** : N'importe quel utilisateur peut créer son salon de jeu. Cela lui permet de devenir le gérant d'une partie, et donc de choisir les différents paramètres de la partie qui réglementeront le déroulement du jeu.
- **Possibilité de rejoindre un salon de jeu** : L'utilisateur a à sa disposition une liste des salons de jeu qui sont en attente de joueurs pour commencer la partie.
- **La phase de jeu** : Le joueur peut lancer les dés (lors de son tour), choisir d'acheter un terrain ou non, payer la caution pour la sortie de la salle d'examen ou tenter le lancer de dés. Sinon il doit se plier aux "effets" des cases sur lesquelles il arrive (CF règles du monopoly).
- **Espace administrateur** : L'administrateur peut, dans une interface différente accessible grâce au konami code, modifier certains paramètres concernant les limites du jeu. Par exemple il peut fixer les limites du nombre de dés utilisés et du capital de départ autorisé. Mais il a également la possibilité d'activer/désactiver certains paramètres de jeu, qui deviendraient ainsi visibles dans l'interface de création de partie des joueurs (exemple : permettre ou non la génération aléatoire du plateau de jeu).

Les fonctionnalités ci-dessus se concentraient principalement sur le point de vue du joueur. Nous avons considéré cette partie de notre projet comme étant primordiale. Celle-ci devait passer impérativement avant toute autre éventuelle implémentation de fonctionnalités optionnelles.

3.1.2 Règles du jeu

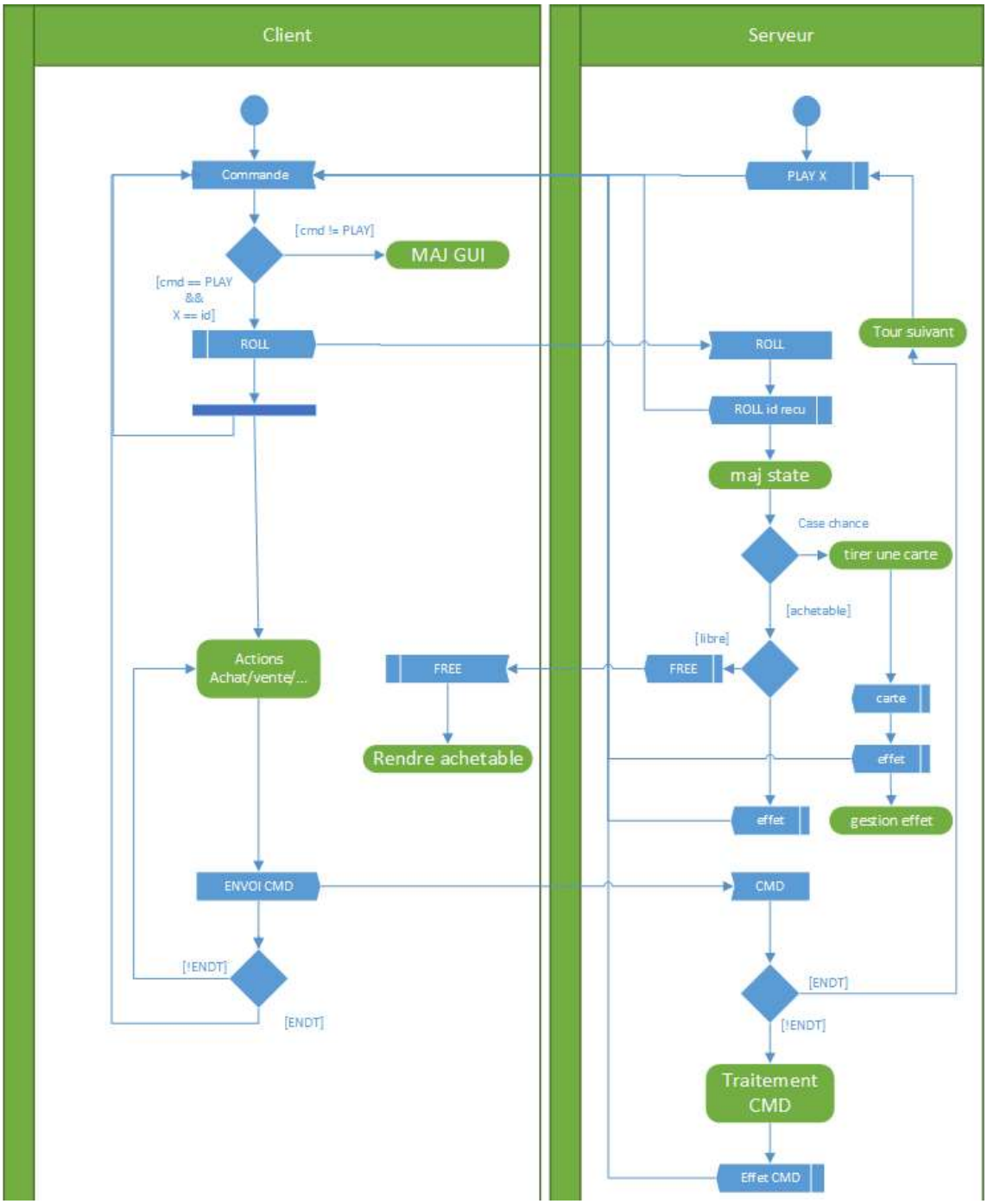
Pour notre projet, nous nous sommes basés sur les règles officielles du jeu Monopoly.

3.2 Partage des responsabilités entre le serveur et le client

Afin d'éviter une surcharge de communications vers le serveur, nous avons décidé d'implémenter toute la logique du jeu du côté serveur. C'est-à-dire que le serveur implémente tout ce qui touche aux transactions, aux échanges et accords entre les joueurs, au calcul des statistiques, à la génération du plateau de jeu et des decks de cartes communautaire/chance.

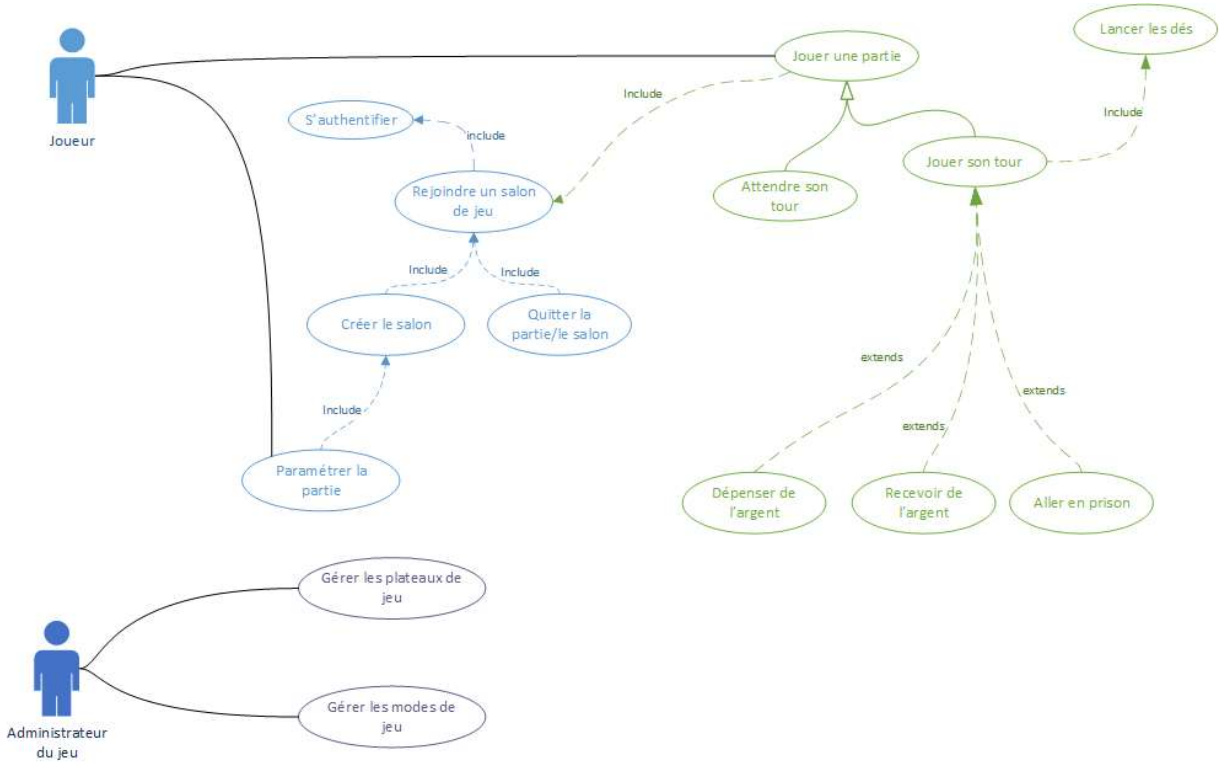
Le client, de son côté, s'occupe d'afficher le plateau de jeu, de maintenir à jour l'état de son joueur et de communiquer au serveur les actions du joueur (exemple : il dit au serveur que le joueur souhaite lancer les dés, acheter une case, etc.).

3.3 Diagramme d'activité général



3.4 Cas d'utilisation

3.4.1 Diagramme des cas d'utilisation



3.4.2 Description des acteurs

Nos scénarios comptent deux acteurs :

- L'administrateur du jeu, qui peut gérer les "modes" de jeu (autres que classique, des paramètres personnalisés).
- Le joueur est le principal acteur de notre projet. C'est lui qui participe à une partie de Cheseaux-Poly, mais pour cela il doit tout d'abord s'authentifier, puis choisir une partie. Enfin il attend que tous les joueurs soient prêts, pour que la partie se lance. En jeu, il doit attendre son tour pour lancer les dés et être actif dans la partie.

3.4.3 Scénario principal

Voici comment se déroule une partie de Cheseaux-Poly :

3.4.3.1 Scénario de préparation

Le scénario de préparation englobe les actions que le joueur doit entreprendre avant de pouvoir jouer.

L'utilisateur va **s'authentifier** à l'aide de son nom d'utilisateur et son mot de passe. Son compte est créé si l'utilisateur a au préalable sélectionné la case « Enregistrement ».

L'utilisateur authentifié peut parcourir la liste des parties en attente de joueurs dans le but de **rejoindre un salon de jeu**.

Il peut également **créer son propre salon de jeu** pour y accueillir d'autres joueurs, cela lui permet de **modifier les paramètres de la partie**.

Une fois que le joueur se sent d'attaque à commencer la partie, il peut se **déclarer « prêt »** et attendre que les autres joueurs fassent de même. Une fois tous les joueurs prêts à jouer, la partie se lance.

3.4.3.2 Scénario de jeu

Une fois la partie lancée, les participants passent en phase de jeu.

Avant de pouvoir jouer, le participant doit **attendre son tour**. Pendant ce temps il peut inspecter les cases du plateau.

Une fois son tour arrivé, il doit **lancer les dés**, pour avancer dans le plateau et ainsi participer à la partie. Si le joueur fait un double (les deux dés ont la même valeur), le joueur doit refaire un tour (relancer les dés). Au bout du troisième double de suite, il est contraint d'**aller en salle d'examen**.

Durant la partie, le joueur a de multiples occasions pour **dépenser son argent**, comme l'achat de propriétés ou encore payer un loyer/une facture, ou **en recevoir**, comme la réception d'un loyer ou la vente d'une propriété.

3.5 Modèle de domaine

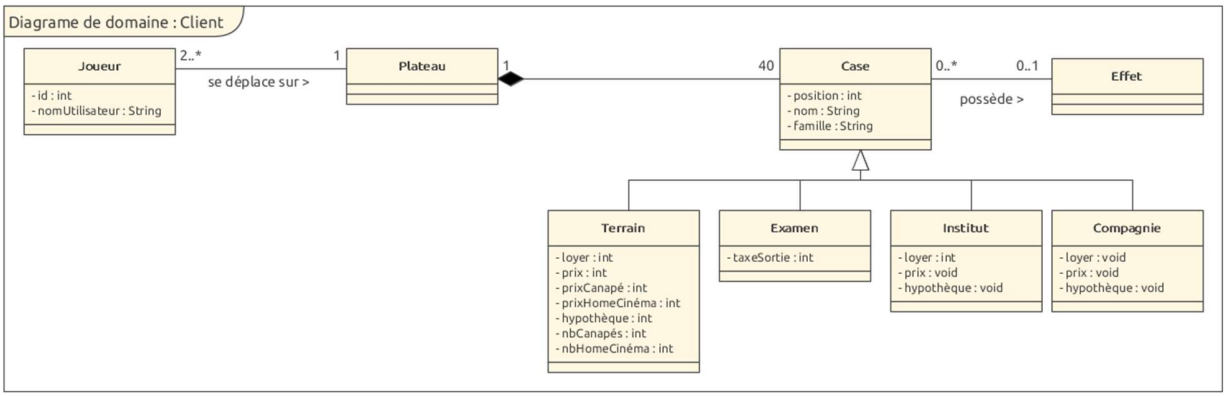


Figure 1 - Modèle de domaine client

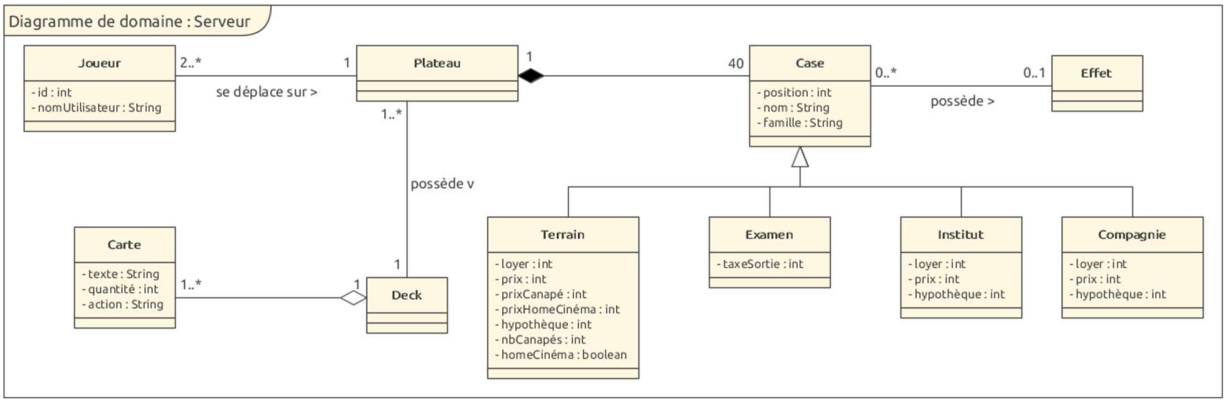
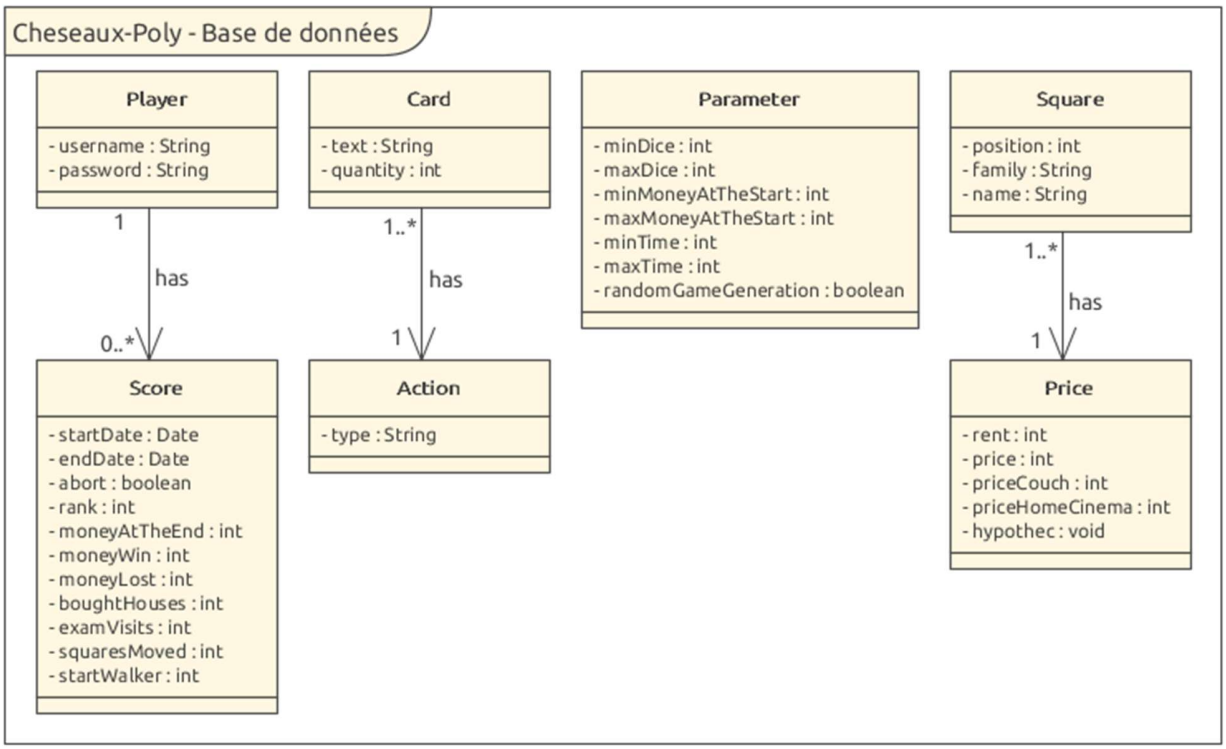


Figure 2 - Modèle de domaine serveur

Voici ci-dessus, les différentes classes à implémenter côté client et serveur. Les classes sont les mêmes des deux côtés. La différence réside dans le fait que seul le serveur doit générer des objets de type Deck et Carte provenant de la base de données, afin de pouvoir choisir aléatoirement une carte.

3.6 Base de données : modèle conceptuel



Le modèle conceptuel de la base de données étant très similaire au schéma relationnel, le détail de chaque classe n'est pas détaillé ici. En revanche nous pouvons développer un peu les relations mises en place entre chaque entité, que nous ne pouvons voir que dans ce schéma.

Comme mentionné plus tard dans la planification des itérations, l'entité des scores n'a pas pu être implémentée par manque de temps et c'est d'ailleurs la seule parmi toutes ces entités. Les cartes ayant des actions associées nous avons fait en sorte, dans notre implémentation de la base de données de notre serveur, d'associer une constante en tant qu'action à chaque carte. Concernant les cases du plateau, elles sont presque toutes dotées d'un panel de prix. De ce fait chaque objet de type case possède ou non un objet de type prix, qui répertorie tous les prix associés à la case.

4 Conception du projet

4.1 Protocole d'échange entre le client et le serveur

Les échanges entre le client et le serveur sont principalement séparés en deux parties : une phase login-lobby qui consiste en la gestion de l'authentification de l'utilisateur, ainsi que la gestion des salons (et leur création) et une phase de jeu.

Pour ce qui est des outils que nous utilisons pour la communication entre le client et le serveur, nous avons choisis d'opter pour des sockets en ce qui concerne les communications sur le réseau. Enfin pour partager des informations sur des objets, nous utilisons la sérialisation à l'aide de JSON.

4.1.1 Protocole applicatif de la première phase

4.1.1.1 Protocole de connexion/gestion des salons de jeu

Message	Effet
Depuis le client	
RGSTR [username] [password_digest]	Demande au serveur s'il est possible de créer un compte avec ces identifiants
LOGIN [username] [password_digest]	Demande au serveur si un compte avec ces identifiants existe
NLOBBY [lobby_JSON]	Crée un salon de jeu, cette commande est suivie des informations du salon à créer (sérialisation de l'objet), puis de la commande JOIN
JOIN [lobby_id]	Tente de rejoindre un lobby s'il y a assez de place
QLOBBY	Quitte le lobby courant
READY	Se déclarer prêt à commencer la partie
BYE	Se déconnecte du serveur
SETP [parameter_JSON]	Envoie les nouveaux paramètres généraux du jeu au format JSON
Depuis le serveur	
WLCM [parameter_JSON]	Envoie les paramètres généraux du jeu à la connexion d'un client
OK	Réponse générale de confirmation après une requête du client
UKNW	Aucun compte utilisateur correspondant n'a été trouvé (réponse à LOGIN)
DENIED	Mot de passe erroné (réponse à LOGIN) ou identifiant déjà utilisé (réponse à RGSTR)
SEEYA	Réponse à BYE
ERR [error_message]	Signale une erreur au client

4.1.1.2 Protocole des salons de jeu

Message	Effet
Depuis le serveur	
LIST [listLobbies_JSON]	Envoie la liste des salons de jeu existants
NEW [lobby_JSON]	Signale la création d'un nouveau salon de jeu
UPDATED [lobby_JSON]	Signale la mise à jour d'un salon de jeu
DELETED [lobby_JSON]	Signale la suppression d'un salon de jeu
START [lobby_id]	Notifie les joueurs concernés que la partie peut démarrer

4.1.2 Protocole applicatif de la deuxième phase : phase de jeu

Message	Effet
Depuis le serveur	
PLYR [player-capital_JSON]	Envoie la liste des joueurs et leur capital
BOARD [squares_JSON]	Envoie la liste des cases du plateau de jeu
PLAY [player_id]	Annonce qui doit jouer
ROLL [player_id] [listDice]	Lance les dés
DRAW [player_id] [card_JSON]	Tire une carte
GAIN [player_id] [amount]	Gagne de l'argent
PAYS [player_id] [amount]	Perd de l'argent
MOVE [player_id] [square_position]	Se déplace sur une case
EXAM [player_id] [square_position]	Se fait envoyer en salle d'examen
FRDM [player_id]	Est libéré de la salle d'examen
FRDM_C [player_id]	Reçoit une carte de sortie de salle d'examen
FRDM_U [player_id]	Utilise une carte de sortie de salle d'examen
FRDM_T [player_id]	Paie la taxe de la salle d'examen pour sortir
FREE [square_position]	Signale que la case sur laquelle se trouve le joueur est libre
BUYS [player_id] [square_position]	Achète la case mentionnée
SELL [player_id] [square_position]	Vend la case mentionnée
BCOUCH [player_id] [square_position]	Achète un canapé pour la case mentionnée
SCOUCH [player_id] [square_position]	Vend un canapé pour la case mentionnée
BHCINE [player_id] [square_position]	Achète un home cinéma pour la case mentionnée
SHCINE [player_id] [square_position]	Vend un home cinéma pour la case mentionnée
HPTQ [player_id] [square_position]	Met en hypothèque la case mentionnée
NHPTQ [player_id] [square_position]	Retire l'hypothèque de la case mentionnée
ENDT	Termine le tour
BKRPT	Signale au joueur qu'il est en banqueroute
WIN [player_id]	Annonce le joueur gagnant
GOVR [player_id]	Annonce la banqueroute d'un joueur
RST [square_position]	Réinitialise la case mentionnée

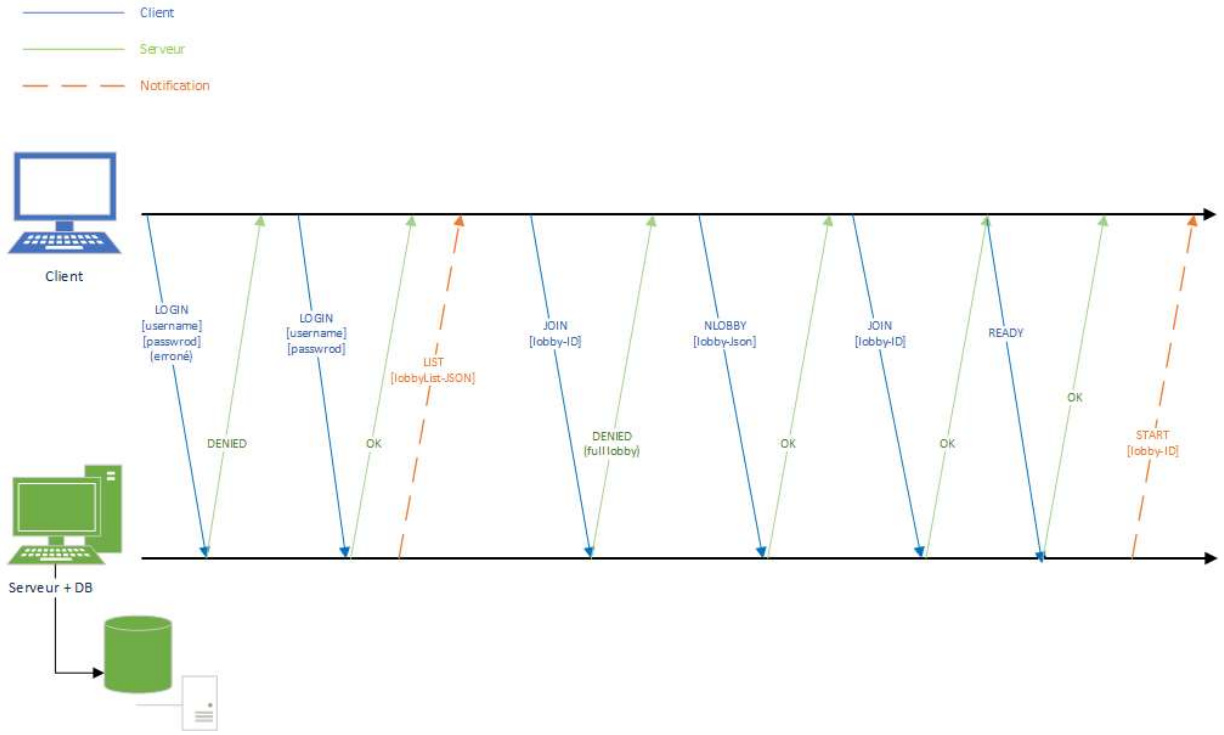


Figure 3 - Schéma de communication : phase login-lobby

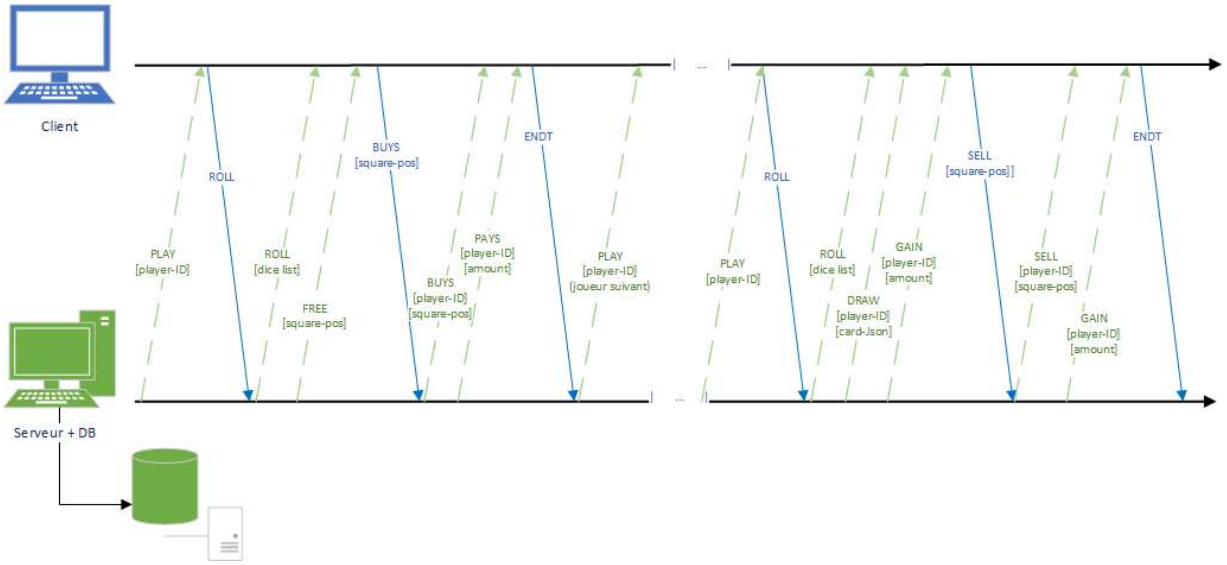
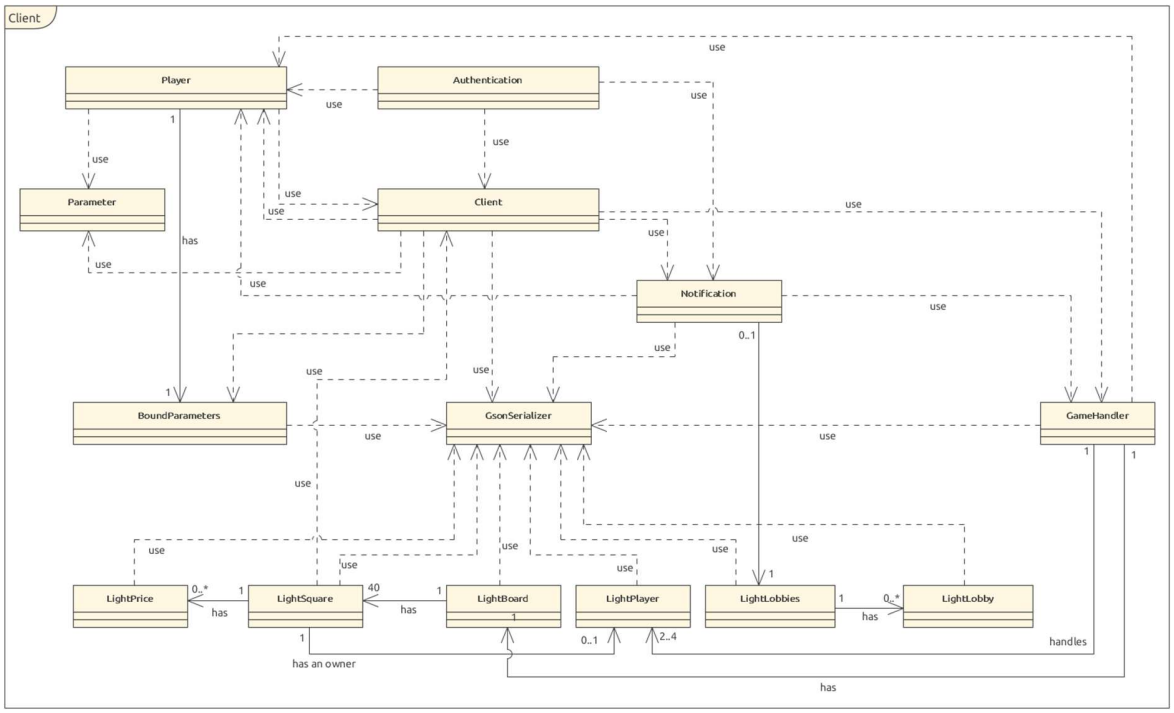
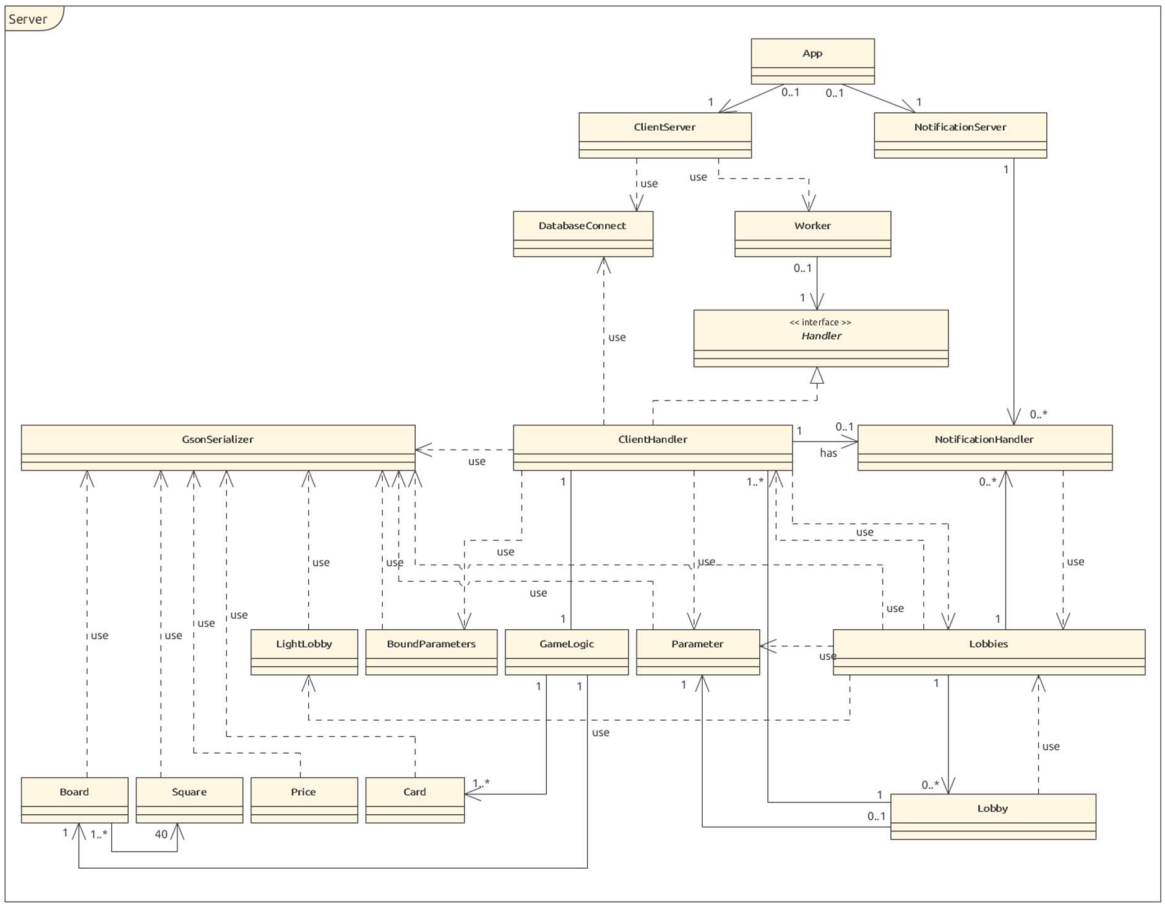


Figure 4 - Schéma de communication : phase de jeu

4.2 Diagrammes de classes





4.3 Base de données : modèle relationnel

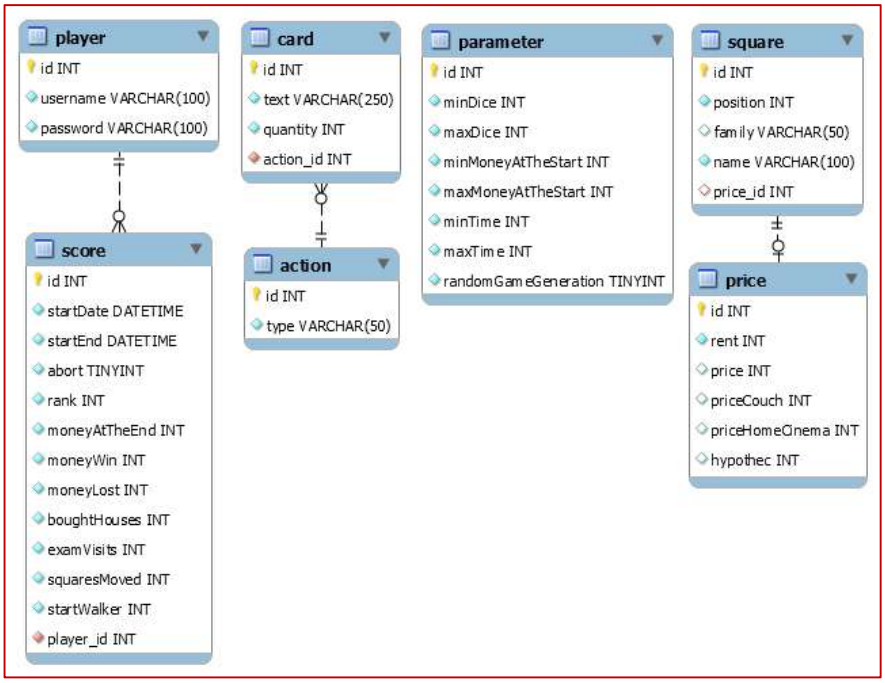


Figure 5 - Base de données de Cheseaux-poly

La base de données de notre projet est plutôt simpliste, car nous n'avons pas besoin de stocker d'autres informations que celles relatives aux joueurs, aux cartes, aux paramètres de jeu et aux cases.

Pour un joueur, nous gardons ses informations de connexion ainsi que les différents scores réalisés durant ses parties jouées. Un score est créé en début de partie et mis à jour en fin de partie, pour ne pas avoir d'interactions inutiles avec la base de données.

Toutes les cartes chance du jeu sont stockées dans la base de données. Pour chacune d'elles, est associé une action de type énumération (MOVE, BACK, EXAM, etc.).

Un seul lot de paramètres est enregistré dans la base de données. Ce sont les limites générales du jeu fixées par l'administrateur. Ces limites sont fournies au joueur uniquement quand il créée une partie.

Chaque case du plateau de jeu est déterminée par un type (START, EXAM, INSTITUTE, etc.) et un nom. Celles nécessitant le paiement d'un loyer et/ou pouvant être achetées, possèdent des attributs financiers en plus pour permettre le paiement d'un(e) loyer/taxe, l'achat d'un(e) hôtel/maison et la mise en hypothèque.

Concernant l'intégration de la base de données au sein de notre architecture, elle se situe du côté serveur. Ce dernier et le client s'échangent des messages protocolés. Les messages demandant une interaction avec la base de données sont générés et envoyés par le serveur uniquement. Le client n'a aucun lien direct avec la base de données. Il ne fait que transmettre une demande au serveur, qui analyse sa requête, génère une requête SQL associée, récupère la réponse, la formatte et la renvoie au client.

5 Implémentation du projet

5.1 Technologies utilisées

Pour développer ce projet nous avons choisi d'utiliser la version 10 de Java, et avons produit le code avec l'environnement de développement de JetBrains : IntelliJ IDEA. En ce qui concerne la partie graphique du client, nous avons opté pour la technologie JavaFX, tout d'abord car nous avions déjà des connaissances dans ce domaine, mais surtout car c'est une technologie moderne donnant plus de possibilité que Swing.

Pour la base de données nous avons choisi d'utiliser un serveur MySQL, car nous avons déjà tous travaillé avec cette technologie et qu'elle convenait parfaitement à la situation.

5.2 Problèmes rencontrés

1 - Notifications : Les notifications ont été un point noir relativement conséquent, dans la mesure où la mise en place s'est étalée sur 2 itérations de trop. L'implémentation que nous avions tout d'abord fournie était beaucoup trop gourmande en termes de ressources (le serveur s'essoufflait avec 3 clients connectés simultanément) et il a donc fallu revoir toute cette implémentation.

2 - Mauvaise estimation des tâches : Nous avons sur/sous-estimé beaucoup de tâches. Certaines étaient déclarée comme demandant une heure de développement lorsqu'en réalité le travail était réalisé en une vingtaine de minutes. D'autres tâches prévoyaient une certaine charge de travail du côté serveur, mais nous oubliions trop souvent le travail équivalent du côté client, ajoutant ainsi beaucoup d'heure de travail à effectuer qui n'étaient pas prises en compte. Un autre point concernant l'organisation des tâches : nous manquions de précisions dans la définition des tâches et du travail qu'il fallait effectuer, nous nous sommes trop de fois posé la question "mais qu'est-ce qu'il faut faire pour cette tâche ?".

3 - GUI : l'interface graphique a demandé beaucoup de temps à se mettre en place. Nous pensons que cela est dû à un manque de communication dans le groupe (quelle méthode utiliser, quand elles étaient disponibles, etc.).

4 - Reconnexion au canal de notification : lorsque le client termine une partie (ou s'il la quitte), il retourne à la liste de salons de jeu, cela implique de se reconnecter au canal de notifications afin d'être à jour dans l'état des différents salons de jeux. Ce problème est encore à résoudre, le client reçoit bien une liste de salons, mais ne reçoit jamais de mise à jour sur ces derniers. Aussi les boutons permettant de rejoindre un salon ne sont plus accessibles.

6 Gestion du projet

6.1 Rôle des participants au sein du groupe de développement

Tout d'abord, au niveau de la répartition des tâches, François Burgener gérait exclusivement la partie interface graphique (GUI). Héléna Reymond s'occupait de la partie concernant la base de données, ainsi que l'implémentation des fonctionnalités côté client/serveur. Daniel s'est chargé de la partie client/serveur et s'est principalement focalisé sur le système de notification entre le client et le serveur. Quant à Bryan il a principalement implémenté du côté serveur. Ce dernier était le chef de projet ainsi que notre scrum master.

6.2 Backlog de produit

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
3	Liste des salons de jeu			Technical story	en tant que A[1-Joueur] je veux obtenir la liste des salons de jeu afin de pouvoir en rejoindre un	transfert d'une liste de salon depuis le serveur ?			Burgener François	Lister salon de jeu
4	Création d'un salon de jeu			User story	en tant que A[1-Joueur] je veux créer mon propre salon de jeu Dans le but d'accueillir des joueurs et de jouer une partie avec des paramètres personnalisés	le paramétrage de la partie se ferait du côté client, à la confirmation le client envoie les informations du salon au serveur pour l'ajouter à la liste.			Burgener François	création salon de jeu
5	Paramétrage d'un salon de jeu			User story	en tant que A[1-Joueur] je veux utiliser des paramètres personnalisés dans le but de faire une partie amusante	récupération des limites des paramètres par le serveur, et communication au client			Burgener François	Paramétrer le salon de jeu
6	Rejoindre un salon de jeu			User story	en tant que A[1-Joueur] je veux rejoindre un salon de jeu pour jouer avec des personnes				Burgener François	rejoindre salon de jeu

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
8	Lancement de la partie			Technical story	le serveur envoie un signal à tous les joueurs pour les informer que tous les joueurs sont prêts et que la partie va commencer				Burgener François	lancer la partie
9	Préparer la partie			Technical story	Le serveur va préparer la partie cela inclut : - l'établissement de l'ordre de passage - génération du deck de carte chance - répartition de la fortune individuelle				Burgener François	"préparer" le jeu
12	Déplacement du pion			Technical story	Lancer les dés. génération de deux entiers entre 1 et 6. obligatoirement 2 pour gérer les doubles ! Une fois les dés lancés on déplace le pion du joueur d'autant de case que le résultat du lancer indique Lors d'un double, le joueur peut rejouer. Après 3 doubles d'affilée, le joueur est expédié en prison				Burgener François	déplacement du pion

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
13	Gestion de la case			Technical story	Le pion arrive à une case. Gérer les actions possibles liées à la case. - Acheter terrain / maison - Tirer carte chance - Payer taxe				Burgener François	détection de la case
14	Achat d'un terrain			User story	en tant que A[1-Joueur] j'aimerai acheter un terrain afin de pouvoir recevoir de l'argent par la suite	implique une dépense d'argent. si le joueur ne veut pas acheter la propriété ... enchère ou non ?			Burgener François	achat de terrain
15	Achat d'une maison			User story	en tant que A[1-Joueur] je veux acheter une maison afin d'augmenter le loyer	l'utilisateur doit avoir tous les terrains de la même couleur pour acheter une maison. il doit également être sur la case de la propriété ou il veut acheter sa maison			Burgener François	achat de "maison"
16	Vente d'une propriété			User story	en tant que A[1-Joueur] je veux vendre une de mes propriété afin de gagner de l'argent				Burgener François	Vente de propriété
17	Vente d'une maison			User story	En tant que A[1-Joueur] je veux vendre une maison afin de ne pas faire banqueroute				Burgener François	Vente de propriété

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
18	Hypothèque de propriété			User story	en tant que A[1-Joueur] je veux hypothéquer une propriété afin de gagner un peu d'argent sans la perdre				Burgener François	Hypothèque de propriété
19	Effet des cartes			Technical story	Gestion des effets des différentes cartes chance				Burgener François	effet des cartes "chances"
21	Détection banqueroute			Technical story	Lorsque le joueur ne possède plus assez d'argent, ni de possession pour redresser la barre, il fait banqueroute. Le joueur a perdu la partie et ne peut plus jouer. - Si possibilité de mettre terrains en hypothèque et continuez, proposer au joueur de mettre en hypothèque ou de déclarer forfait	on laisse la possibilité d'être spectateur			Burgener François	Détection banqueroute
23	Accès a la zone administrateur			User story	en tant que A[2-Administrateur] je veux accéder à la zone administrateur afin de calibrer les paramètres du jeu	accès par login spécial ?			Burgener François	Accès zone administrateur
24	Paramétrage général du jeu			User story	en tant que A[2-Administrateur] je veux modifier les paramètres du jeu pour calibrer l'expérience de jeu				Burgener François	modification des paramètres (admin)

ID	Name	Effort	Rank	Type	Description	Notes	Accepted date	Estimated date	Creator	Feature
25	Gestion d'inactivité			Technical story	Si le joueur ne fait aucune action durant la moitié du temps qu'il a à disposition, lui mettre un alerte avec un timer. S'il ne joue pas dans le temps imparti : - Malus (perte d'argent) S'il ne joue pas 3 tours d'affilée : - Exclure le joueur de la partie (forfait)	faite du côté client			Burgener François	Détection d'inactivité
26	gestion de prison			Technical story	déplace le joueur dans la case "prison". Il peut lancer les dés dans l'espoir de faire un double. S'il échoue il passe son tour. Il peut également sortir de prison à l'aide d'une carte chance ou en payant une taxe. Un joueur ne peut rester en prison au plus 3 tours.				Burgener François	Aller en prison
22	Information de la case			Technical story	à tout moment, l'utilisateur peut sélectionner une case pour en obtenir des informations. P. ex. : le prix (achat/loyer), les effets, le propriétaire				Burgener François	Informations case

6.3 Plan d'itérations

6.3.1 Sprint 1

BUT : Mise en place de l'enregistrement et de la connexion d'un joueur sur le jeu (base de données, serveur, client).

DURÉE : 19 heures (du 27 avril au 3 mai)

Figure 6 - Histoires du sprint 1

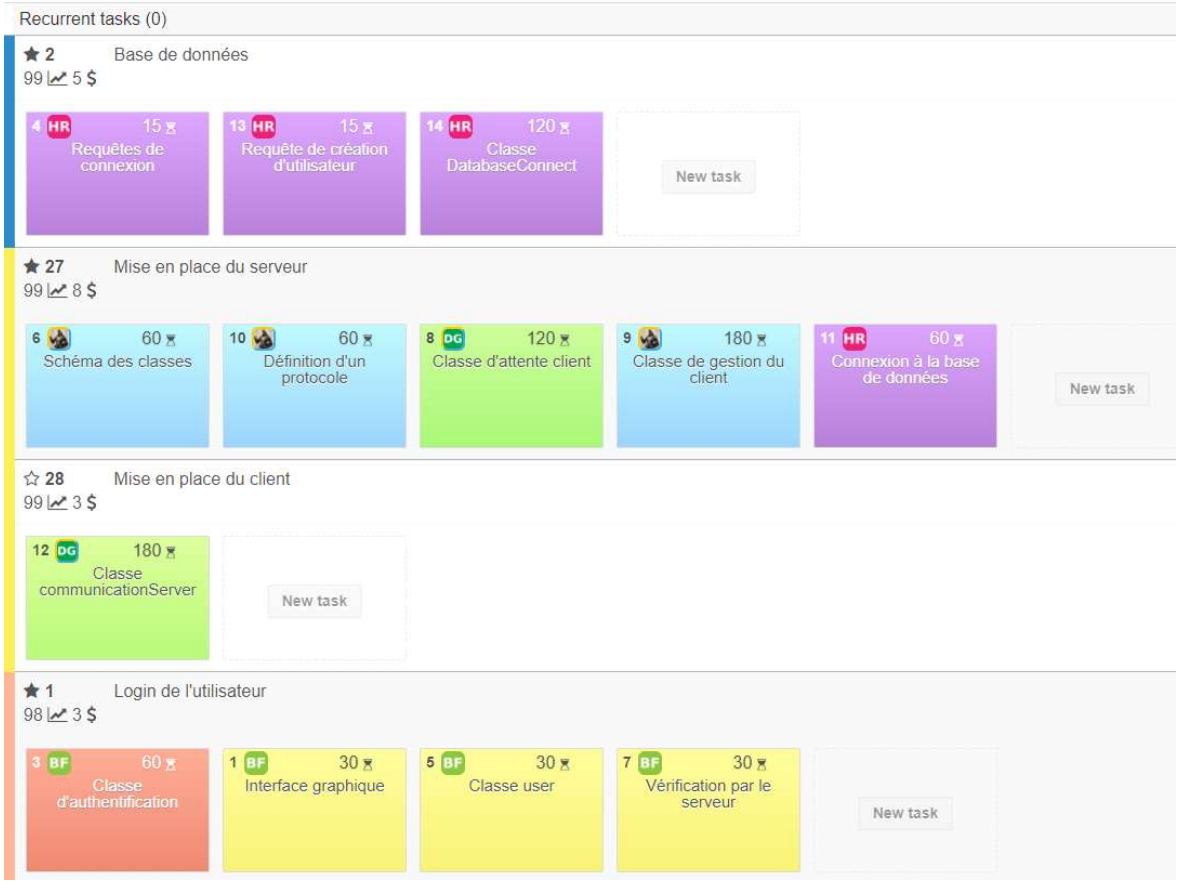


Figure 7 - Tâches du sprint 1

6.3.2 Sprint 2

BUT : Pouvoir, après s'être connecté au jeu, voir le salon de jeu avec les parties disponibles, créer sa propre partie avec des paramètres personnalisés et rejoindre une partie.

DURÉE : 19 heures (du 4 mai au 10 mai)

Figure 8 - Histoires du sprint 2

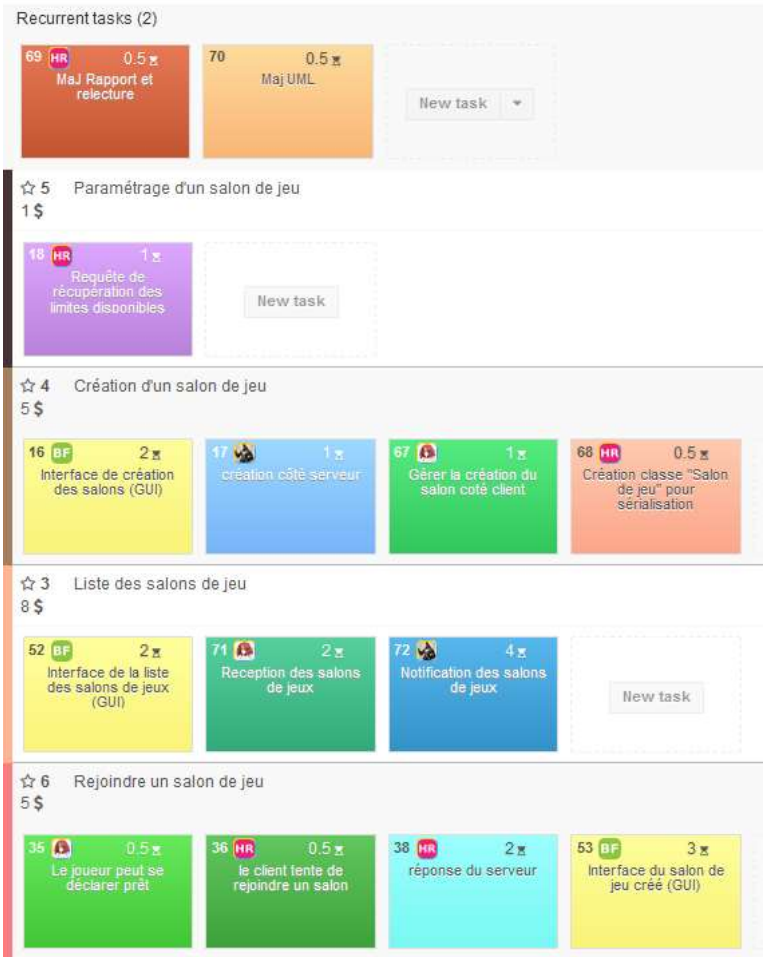


Figure 9 - Tâches du sprint 2

6.3.3 Sprint 3

BUT : Pouvoir, après avoir rejoint ou créé une partie, lancer cette dernière en se mettant prêt. Quand tout le monde est prêt, la partie s'initialise.

DURÉE : 23 heures (du 12 mai au 18 mai)

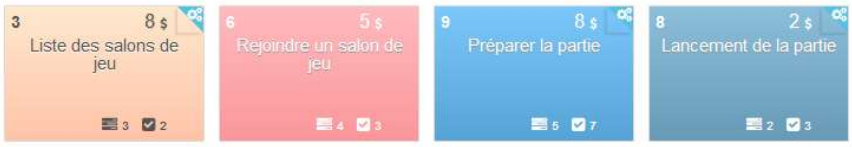


Figure 10 - Histoires du sprint 3

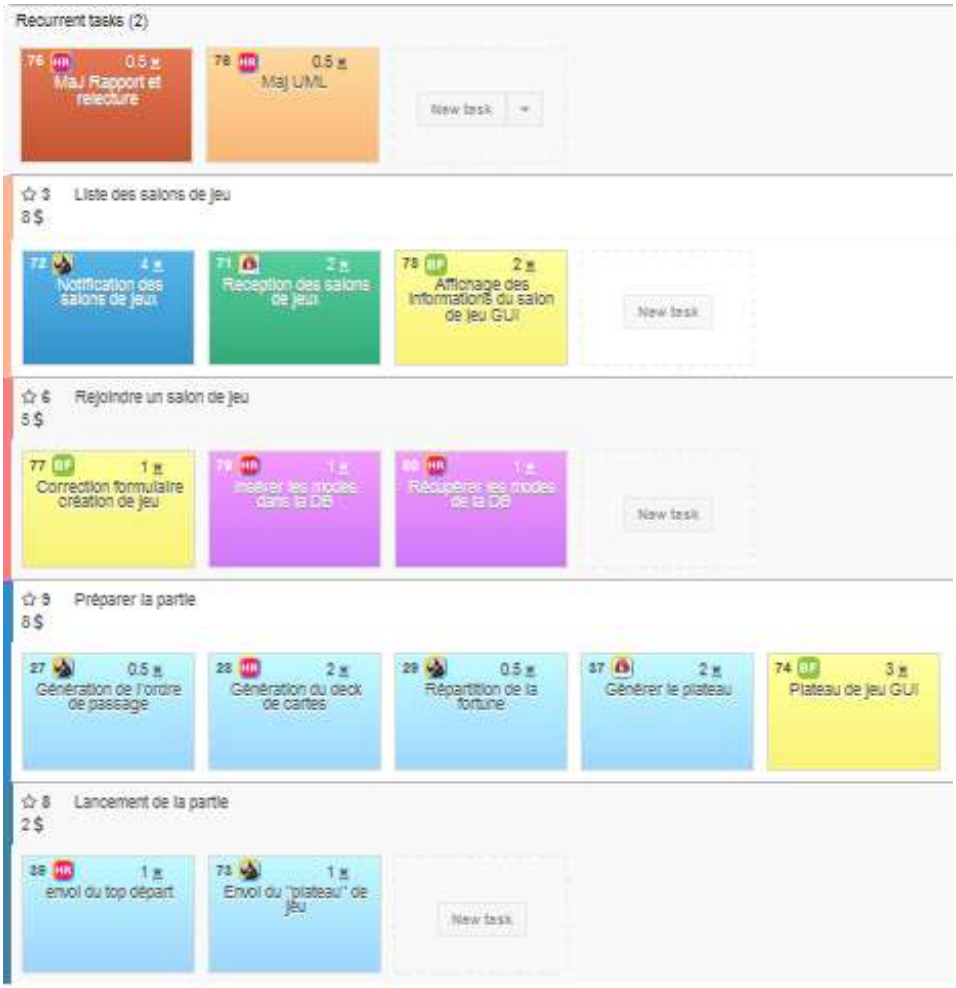
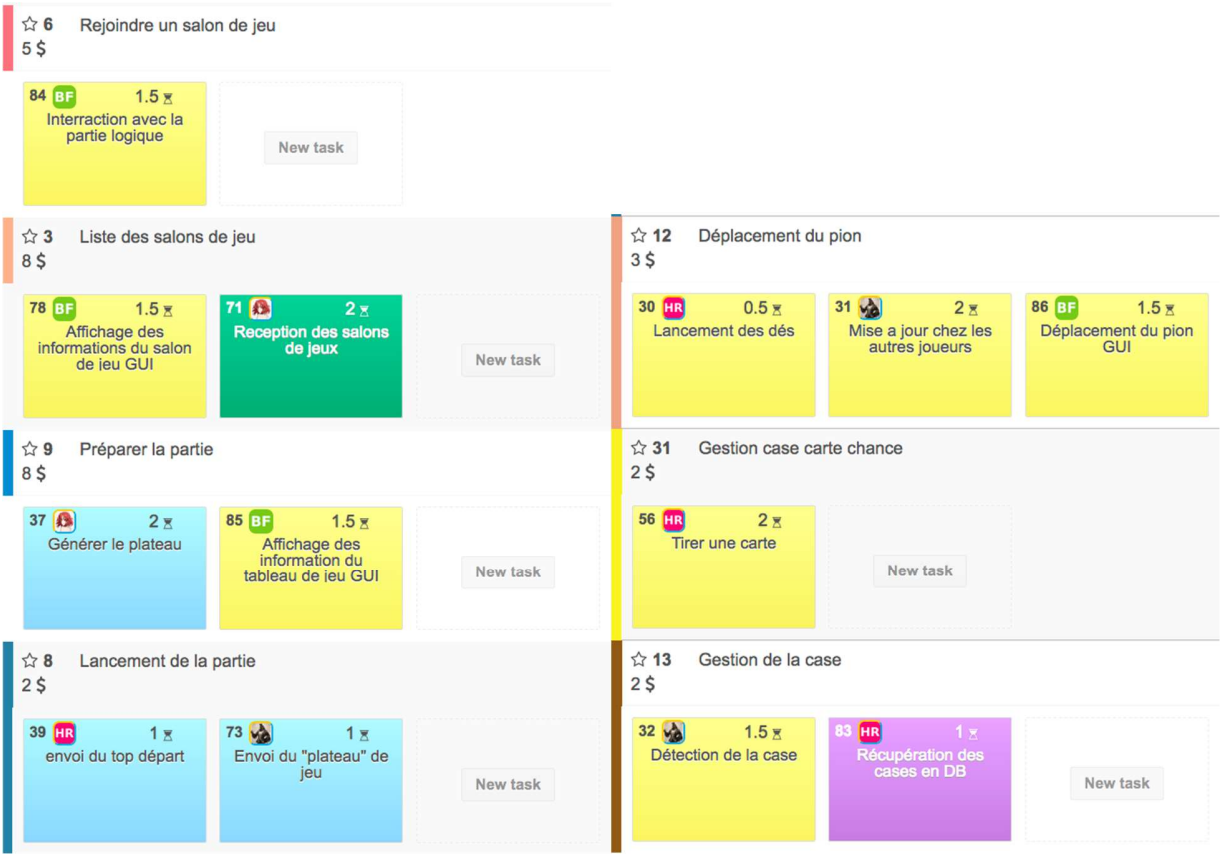


Figure 11 - Tâches du sprint 3

6.3.4 Sprint 4

BUT : Pouvoir, après avoir rejoint ou créé une partie, lancer cette dernière en se mettant prêt. Quand tout le monde est prêt, la partie s'initialise. Le médiateur du jeu (serveur) doit pouvoir déplacer les joueurs et gérer les cases « carte chance ».

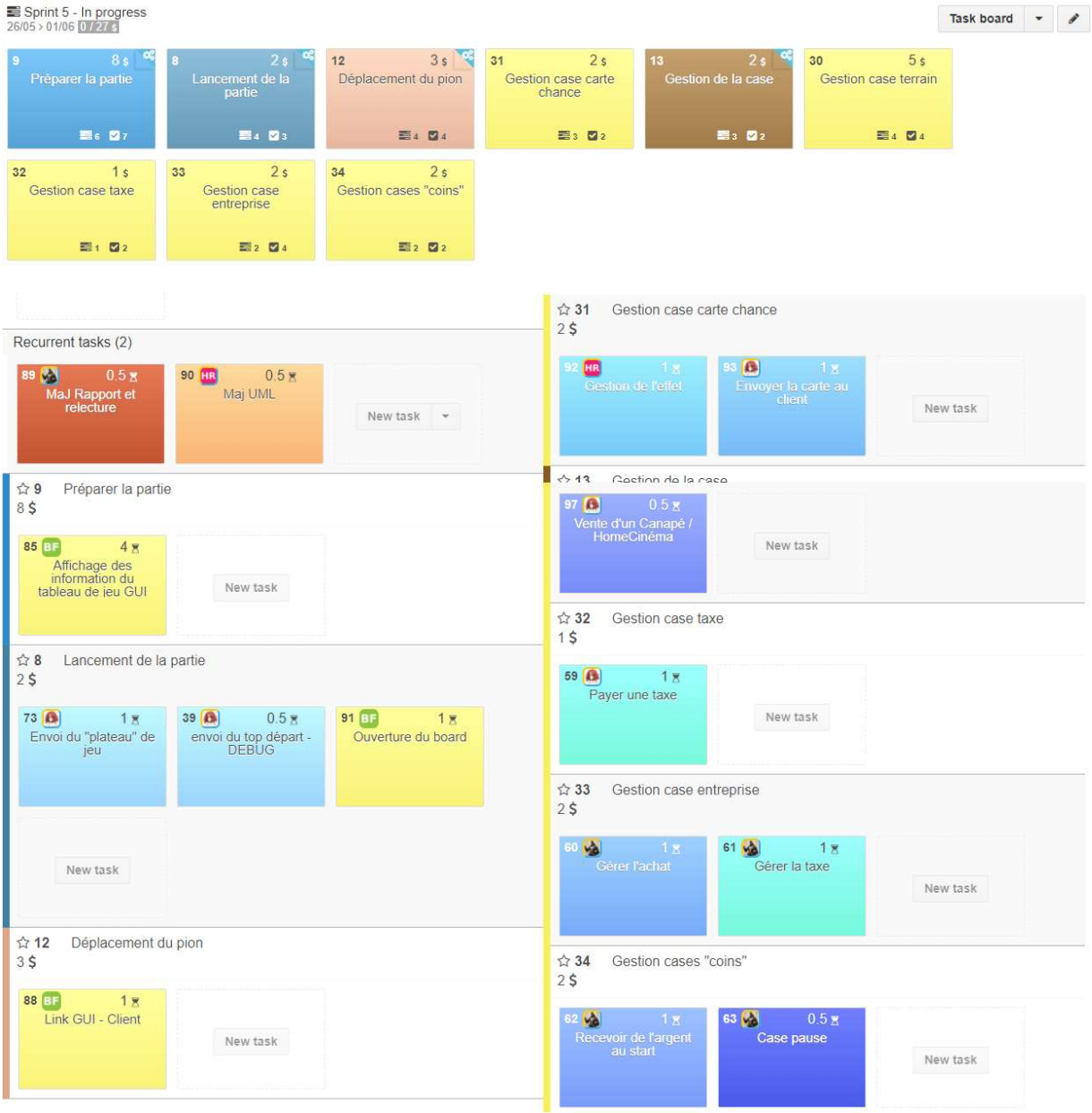
DURÉE : 20 heures (du 19 mai au 25 mai)



6.3.5 Sprint 5

BUT : Le serveur pourra générer le plateau de jeu et le transmettre au client. Il pourra également reconnaître certaines cases et gérer leurs actions respectives. Le client doit pouvoir accéder, lorsque tous les joueurs sont prêts, au plateau de jeu. Une fois dans la phase de jeu, ils pourront déplacer leur pion (chacun leur tour) sur le plateau et seront informés de la case sur laquelle ils atterrissent.

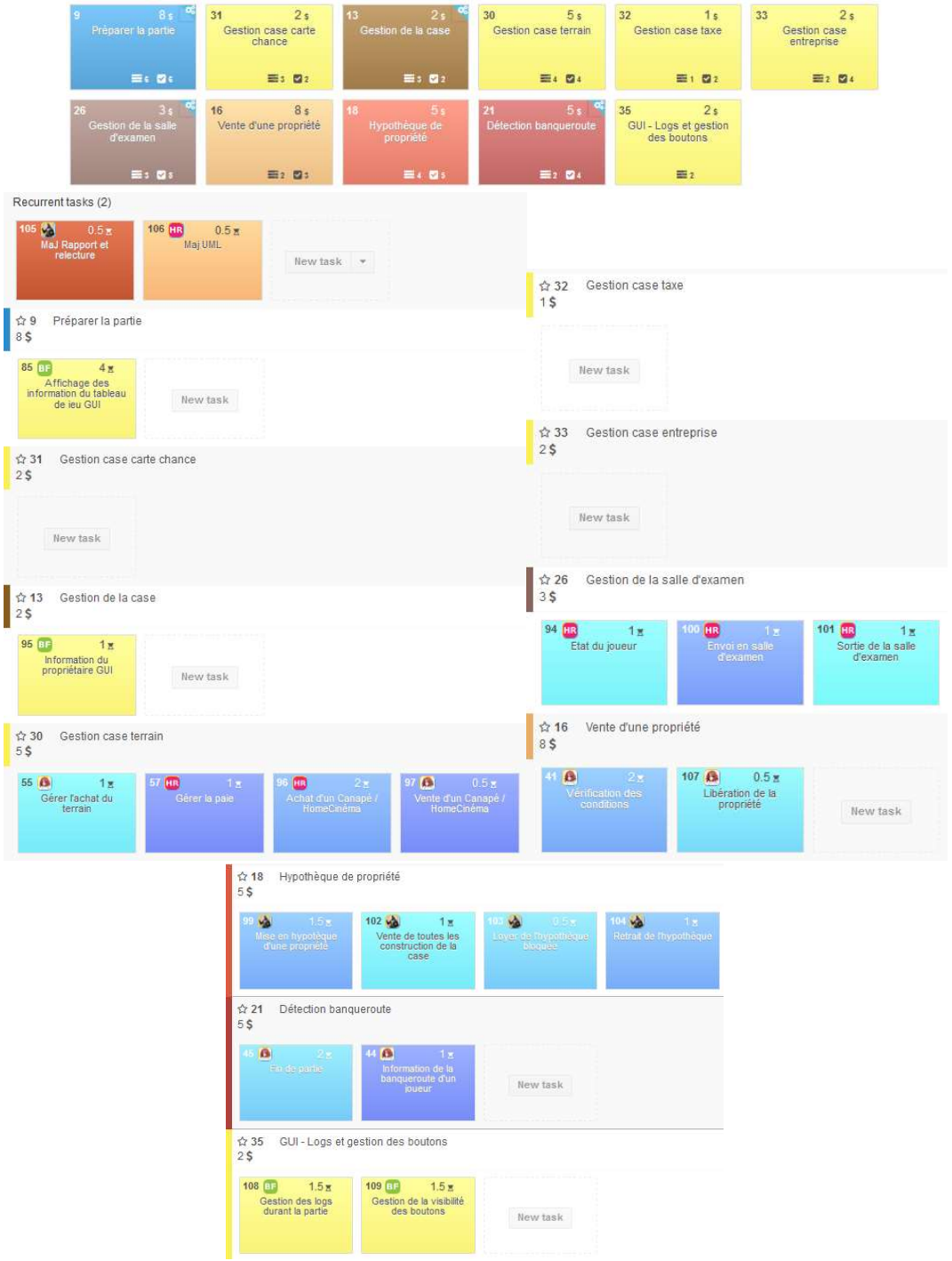
DURÉE : 21.5 heures (du 26 mai au 1er juin)



6.3.6 Sprint 6

BUT : Les joueurs sont informés de la case sur laquelle ils atterrissent et peuvent interagir avec elle. Ils ont la possibilité d'acheter/vendre/hypothéquer des propriétés, payer des taxes selon la case, aller en salle d'examen ou en sortir et faire banqueroute.

DURÉE : 26 heures (du 2 juin au 8 juin)



6.3.7 Sprint 7

BUT : Le joueur doit pouvoir jouer entièrement une partie et l'administrateur changer les paramètres généraux de jeu de Cheseaux-Poly.

DURÉE : 26 heures (du 9 juin au 15 juin)

6.4 Bilans d'itérations

Vous trouverez ci-après les bilans des itérations rédigés par nos soins. Les bilans d'itérations effectués en compagnie du professeur et de l'assistant sont fournis dans les annexes.

6.4.1 Itération 1

a) Bilan sur la terminaison des histoires

- Histoires planifiées : 1 – 2 – 27 – 28
- Histoires terminées : 1 – 2 – 27 – 28
- Histoires non-terminées : aucune

b) Vélocité du sprint

4 histoires réalisées/4 histoires planifiées → vélocité de 100%

c) Replanification

Le sprint ayant été complètement réalisé, aucune replanification n'est nécessaire.

d) Commentaire général

Tout s'est passé comme prévu et même plus rapidement que ce que nous pensions, avant d'entamer le sprint. Nous n'avons rencontré aucun problème durant le développement.

e) Autocritique

Le sprint s'est bien passé, néanmoins nous aurions pu faire une meilleure implémentation de certaines parties du code que nous allons devoir modifier de toute façon. Par exemple, nous aurions pu faire une meilleure gestion des erreurs, commenter le code de manière plus rigoureuse ou encore implémenter les messages d'erreurs lors de problèmes de connexion (interface graphique) plutôt que de faire des changements de couleur incompréhensibles pour un utilisateur lambda. Surtout que pour ce premier sprint, nous étions en avance et cela nous aurait fait gagner du temps pour la suite.

Mise à part ce petit point, tout le reste s'est bien passé, le sprint est concluant et opérationnel (tous les tests sont passés), nous n'avons donc pas eu d'adaptation à faire pour la suite et nous pouvons continuer sur cette bonne voie.

f) Bilans personnels

- Bryan (3.5 heures réalisées VS 4.5 heures planifiées)
  Une première itération qui s'est plutôt bien passée. L'implémentation d'un serveur est intéressante.
- Daniel (5 heures réalisées VS 5 heures planifiées)
  Fondamentalement, tout s'est bien passé. Le premier sprint était assez critique, car il met en place la connexion entre client et serveur (qui va être utilisée par la suite). Cela dit, nous avons bien implémenté cela, sans trop de problèmes, ce qui va nous permettre de bien poursuivre notre travail. L'effort fourni était conséquent, mais pas exagéré.
- François (1.5 heures réalisées VS 1.5 heures planifiées)
  Tout s'est bien passé et dans les temps.
- Héléna (2.5 heures réalisées VS 3.5 heures planifiées)
  Pour ma part, tout s'est bien passé. Je n'ai rencontré aucun problème au niveau de la base de données. Le point le plus délicat était de faire en sorte que mes collègues accèdent au serveur de base de données stocké sur mon ordinateur. Pour cela j'ai dû créer un utilisateur lié à la base de données provenant de n'importe quelle adresse IP, et ajouter une règle dans mon pare-feu pour autoriser des connexions venant de l'extérieur sur le port de MySQL.

6.4.2 Itération 2

a) Bilan sur la terminaison des histoires

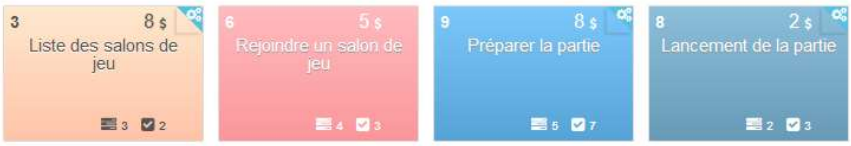
- Histoires planifiées : 3 – 4 – 5 – 6
- Histoires terminées : 4 – 5
- Histoires non-terminées : 3 (réalisée à 50%) – 6 (réalisée à 95%)

b) Vélocité du sprint

2 histoires réalisées/4 histoires planifiées → vélocité de 50%

c) Replanification

1) Les histoires 3 et 6, non terminées, sont reportées sur le sprint suivant (sprint n°3). Les raisons de la replanification sont la charge de travail trop importante des autres cours (surtout de PRO), ainsi que la complexité de l'histoire 3 à mettre en place. La non vérification du dernier test de l'histoire 6 dépend de l'histoire 3 non terminée. Cette dernière a été implémentée, mais n'a pas fini de l'être. Elle doit être encore développée, corrigée et testée pour valider le sprint 2 complet.
2) Les sprints concernés par la replanification sont probablement les sprints 3 et 4. 3 parce que nous devons y ajouter les deux histoires non terminées et 4 car le sprint 3 étant plus grand que prévu, nous n'arriverons certainement pas au bout de tout. Néanmoins notre retard sera assez vite rattrapé, le sprint 2 ayant été le plus compliqué à mettre en œuvre.
3) Printscreen du sprint 3 replanifié



d) Commentaire général

Pour cette itération, tout ne s'est malheureusement pas passé comme prévu. Nous n'avons pas réussi à tout finir. Globalement, nous avons quand même réussi à faire la grosse partie du boulot (2 histoires sur 4 et les histoires non terminées sont effectuées à plus de 50% chacune).

e) Autocritique

Les difficultés rencontrées pour ce sprint, raisons pour lesquelles nous n'avons pas réussi à terminer toutes les histoires, sont les suivantes :

1. La charge de travail des autres branches était plus conséquente que lors de la première itération. En effet, le rendu du projet de semestre s'approchant à grand pas, nous avons été pas mal focalisés dessus.

2. En partie dû à la charge de travail mais également au weekend de l'ascension, nous n'avons pas pu nous "retrouver" (que ce soit physiquement ou par chat) pour se coordonner. Il y a un groupe qui a donc travaillé une journée, soigneusement, avec des tests unitaires pour valider le code implémenté. Et le lendemain, l'autre groupe a été malheureusement moins rigoureux et nous avons dû tous repasser dans le code pour corriger et valider les deux histoires.

Pour résumé, nous n'avons pas pu nous focaliser autant que voulu pour cette itération et par manque d'organisation, nous avons perdu le temps que nous n'avions déjà pas...

f) Bilans personnels

- Bryan (5 heures réalisées VS 5 heures planifiées)
  Un second sprint qui s'est moins bien passé que le précédent. Cela est principalement dû à la quantité de travail demandée par les autres branches (PRO, pour n'en citer qu'une), et aussi peut-être un manque d'organisation. Mais nous avons tout de même pu progresser dans le développement du projet et terminer une bonne partie des histoires qui étaient prévues.
- Daniel (8 heures réalisées VS 3 heures planifiées)
  Pour cette itération, j'estime avoir travaillé très méthodiquement, pour chaque fonctionnalité implémentée, j'ai testé grâce aux tests unitaire (JUnit) son bon fonctionnement et ainsi pu validé la presque totalité de mes tâches. Pour les tâches non terminées, je pense que nous les avons probablement sous-estimées et notre manque de méthodologie "général" a fait que nous n'avons pas pu les terminer. Mais je vais en garder l'enseignement qu'il se doit et éviter, à l'avenir, de reproduire les mêmes erreurs.
- François (3.5 heures réalisées VS 7 heures planifiées)
  Fondamentalement, pour ma part tout s'est bien passé en ce qui concerne les interfaces graphiques. Nous n'avons pas pu faire toutes les interactions avec le client dû à la quantité de travail que l'on avait à faire pour les autres branches. Mais nous avons tout de même fait une grande majorité des tâches demandées pour le sprint2.
- Héléna (9 heures réalisées VS 5 heures planifiées)
  Du côté base de données et sérialisation, tout s'est bien passé. Je n'ai rencontré aucun problème et ai pu avancer comme c'était prévu. En revanche, je pense que ce sprint était un peu trop chargé au niveau des histoires, car il tombait la même semaine que le dernier sprint du cours de PRO. De ce fait nous avons principalement concentré nos efforts sur PRO, délaissant au départ GEN, pour finir par tout faire en une journée, ce qui n'était pas suffisant pour remplir ce sprint à 100%.

6.4.3 Itération 3

a) Bilan sur la terminaison des histoires

- Histoires planifiées : 6 – 3 – 9 - 8
- Histoires terminées : 6 (1 test restant)
- Histoires non-terminées : 3 (réalisée à 50%) – 9 (réalisée à 70%) – 8 (réalisée à 0%)

b) Vélocité du sprint

1 histoires réalisées/4 histoires planifiées → vélocité de 25%

c) Replanification

Vu la progression de cette itération, nous envisageons de faire une replanification globale des itérations. Beaucoup d'histoires vont être décalées pour donner des itérations équilibrées.

d) Commentaire général

Comme l'itération précédente, nous n'avons pas pu terminer toutes les tâches planifiées. Principalement car le manque d'organisation de notre part lors de la seconde itération a conduit à une correction du travail effectué. Nous avons tout de même pu progresser dans le développement du projet.

e) Autocritique

Pour cette itération, à nouveau, nous avons consacré plus de temps au projet de semestre car c'était la dernière semaine pour boucler notre travail. Une autre erreur de notre part serait une mauvaise estimation de l'investissement requis pour cette itération. Nous nous étions basés sur la progression que nous avions au terme de la seconde itération, mais il a finalement fallu repasser sur une partie du travail effectué.

f) Bilans personnels

- Bryan (3 heures réalisées VS 6 heures planifiées)
  Cette troisième itération a également subi l'influence de la charge de travail des autres cours (encore une fois, principalement PRO). De plus notre manque d'organisation lors de l'itération 2 a causé une correction du travail effectué, retardant donc à nouveau l'avancement de l'itération. Malgré ces contre-temps, nous avons pu progresser dans le développement du projet, ce n'est pas comme si on stagnait. De plus je pense que la fin du module PRO est un signe de bon augure pour la suite du Cheseaux-Poly.
- Daniel (5 heures réalisées VS 4 heures planifiées)
  Comme pour l'itération 2, nous avons manqué de temps à cause de la charge de travail pour les autres cours. Nous avons néanmoins pu avancer une partie des tâches et une grosse partie de l'aspect des notifications des parties est en place, ce qui est plutôt bon étant donné qu'il s'agit d'un aspect critique de notre application que de bien gérer cela.
  Je trouve qu'on a mal estimé la charge de travail concernant la partie notifications et que cela a eu un effet négatif dans nos itérations. Néanmoins, je garde bon espoir de rattraper ce retard avec l'équipe et de pouvoir profiter d'une petite partie de Cheseaux-Poly une fois le projet terminé.
- François (3 heures réalisées VS 6 heures planifiées)
  Pour les mêmes raisons que l'itération 2, nous avons eu du retard dû à la charge de travail des cours, surtout dû à PRO. Nous avons quand même réussi à faire pas mal de tâches qui était prévu. Surtout que ces tâches, comme la notification, sont des tâches principales pour notre

application. En ce qui concerne la GUI, nous avons les fenêtres qui sont faites mais nous n'avons aucun lien avec la partie logique de l'application.

- Héléna (4 heures réalisées VS 6 heures planifiées)
  L'itération 3 fut malheureusement pauvre en développement, du fait que nous étions dans la dernière semaine critique pour le rendu du projet de PRO. De ce fait, je n'ai pas pu bosser avant jeudi soir (veille de la présentation de l'itération 3). Néanmoins j'ai quand même pu terminer le 80% des tâches qui m'étaient assignées. Je n'ai pas pu gérer l'envoi du top départ de la partie, parce que le code gérant cette interaction n'était pas terminé (Notification).

6.4.4 Itération 4

a) Bilan sur la terminaison des histoires

- Histoires planifiées : 3 – 6 – 8 – 9 – 12 – 13 – 31
- Histoires terminées : 3 – 6
- Histoires non-terminées : 8 (réalisée à 25%) – 9 (réalisée à 80%) – 12 (réalisée à 75%) – 13 (réalisée à 50%) – 31 (réalisée à 33%)

b) Vélocité du sprint

2 histoires réalisées / 7 histoires planifiée ≈ 29%

c) Replanification

Les histoires qui n'ont pas été terminées (ni testées) ont été reportées dans le sprint 5. Étant donné que la plupart de ces histoires sont bien entamées, nous pensons ne pas devoir passer beaucoup de temps sur celles-ci.

d) Commentaire général

La fin du module de PRO a effectivement été bénéfique au déroulement du projet. Nous avons pu nous concentrer sur le développement de Cheseaux-Poly, et rattraper une bonne partie de notre retard. Nous avons terminé la mise en place des notifications pour les salons de jeux, et commencé la logique de jeu. La GUI a également bien progressé, mais il reste encore à la raccorder au logiciel.

e) Autocritique

Bien que l'itération se soit mieux déroulée que les deux dernières nous y sommes pris au dernier moment pour travailler sur le projet. En nous organisant mieux (s'y prendre un peu plus à l'avance), nous aurions pu tester les tâches que nous estimions terminées et régler les problèmes que nous avons trouvé peu de temps avant la démonstration.

f) Bilans personnels

- Bryan (4 heures réalisées VS 4.5 heures planifiées)
  Une itération qui s'est bien mieux déroulée que les deux précédentes. Nous nous étions organisés, et nous avons pu grandement avancer dans le projet. Le retard n'est peut-être pas encore rattrapé, mais nous sommes convaincus de pouvoir le rattraper prochainement (peut-être à l'itération suivante ?). L'implémentation de la logique de jeu est un défi intéressant, qui nécessite de prendre en considération les éventuelles actions de tous les joueurs. Je prends un certain plaisir à implémenter cette partie.
- Daniel (10 heures réalisées VS 4 heures planifiées)
  Pour ce sprint, Nous devions absolument terminer la partie qui s'occupe des notifications des changements au niveau des lobby entre le serveur et le client. Nous avions eu beaucoup de problèmes à ce niveau-là et avions déjà trop de retard. Je me suis donc attelé à la tâche et j'y ai passé plus de temps qu'initialement prévu, mais j'ai au moins réussi à terminer la tâche (et débloquer le reste). Comme cela m'a pris plus de temps que prévu, je n'ai pu aider que pour une autre tâche et nous avons donc toujours du retard à la fin du sprint. Mais la grosse partie est derrière nous et j'ai bon espoir qu'on récupère la moitié de notre retard dans le prochain sprint. Mis à part le retard, le sprint c'est bien passé en ce qui concerne l'équipe et l'engagement personnel, donc rien à dire là-dessus.
- François (5 heures réalisées VS 4.5 heures planifiées)

Cette itération s'est bien mieux passée que les précédentes. Nous avons pu commencer à intégrer la logique dans la GUI. Nous avons pu implémenter la majorité des taches prévues mais nous n'avons pas encore rattrapé totalement notre retard.

- Héléna (5 heures réalisées VS 3 heures planifiées)
  Pour ma part le sprint s'est bien passé. Mes tâches se sont effectuées sans embûches. Nous nous sommes enfin débloqués avec la mise en place des notifications, de ce fait nous pouvons dès à présent rattraper notre retard et continuer dans le projet. De plus la charge de travail s'est un peu allégée, ce qui nous permet de nous concentrer un peu plus sur nos tâches respectives de Cheseaux-Poly.

6.4.5 Itération 5

a) Bilan sur la terminaison des histoires

- Histoires planifiées : 8 – 12 – 13 – 30 – 31 – 32 – 33 – 34
- Histoires terminées : 8 – 12 – 34
- Histoires non-terminées : 13– 31 – 32 – 33 (toutes réalisées à 80%) – 30 (réalisée à 0%)

b) Vélocité du sprint

3 histoires réalisées / 8 histoires planifiée ≈ 37.5%

c) Replanification

Les histoires qui ne sont pas complètement terminées ont été reportées dans l'itération 5. Dans la plupart de ces dernières c'est la partie Interface Utilisateur (GUI) qui n'est pas encore implémentée, ou simplement des tests qui n'ont pas été validés. Dans le cas de l'histoire 30 (Gestion des cases terrain) bien que nous n'ayons pas trouvé le temps de l'implémenter, nous voyons comment l'implémenter et la mettre en place.

d) Commentaire général

Une itération qui s'est globalement bien passée, nous avons pu mieux nous organiser, et donc travailler de façon plus efficace. Malgré la charge de travail supplémentaire pour l'itération 6, nous sommes confiants et pensons que nous pourrons assumer ces histoires tout en prenant en charge le travail initialement prévu dans l'itération suivante.

e) Autocritique

Globalement nous pensons que nous nous sommes mieux organisés que le sprint précédent. Nous pourrions encore améliorer le travail montré en nous y prenant encore un peu plus en avance, mais le travail demandé dans les autres branches limite cette possibilité.

f) Bilans personnels

- Bryan (5 heures réalisées VS 5.5 heures planifiées)
  Une itération qui s'est, selon moi, très bien déroulée. Nous n'avons pas terminé toutes les histoires (on en a même délaissé une) mais nous avons fait d'importantes progressions. La partie GUI peine un peu à se mettre en place mais c'est une partie importante du projet alors il faut prendre le temps nécessaire pour fournir un résultat convenable. En ce qui concerne le serveur, nous progressons à un rythme convenable et les fonctionnalités se mettent en place sans trop de problème. Le travail sur le serveur est toujours aussi intéressant, et l'équipe est toujours aussi motivée à terminer le projet. Un point qui, d'après moi, s'est bien passé durant cette itération est que nous avons pu bien départager les tâches : chacun travaillait "de son côté" tout en communiquant avec les autres, nous avons rapidement avancé grâce à ça.
- Daniel (5 heures réalisées VS 6 heures planifiées)
  Le sprint s'est plutôt bien passé, le retard diminue de plus en plus, le lien entre GUI et infos reçues du serveur étant le point critique souvent le plus difficile à gérer. Nous avons enfin un plateau de jeu avec quelques informations (les couleurs) et les pions qui se déplacent dessus, en activant certaines actions de cases (comme la case start et le gain de départ ou encore les cases chances). Je pense que pour ce sprint, si les autres tâches externes au projet nous le permettent (autres branches et projet), nous allons pouvoir combler encore un peu de notre retard et avoir une bonne partie du plateau fonctionnelle. Pour ma part, le travail est toujours intéressant et le travail au sein de l'équipe est agréable et motivant. J'ai hâte de voir le produit final et j'espère que nous pourrons implémenter tout ce que nous comptions avoir dans notre Cheseaux-Poly.

- François (5 heures réalisées VS 6 heures planifiées)
  Pour ce sprint 5, nous n'avons pas eu trop de difficultés à réaliser les tâches prévues. Nous avons tout de même encore un peu de retard mais cela est minime. Nous avons eu quelques problèmes au niveau de la synchronisation de thread que nous avons résolu assez rapidement. Nous avons pu intégrer une bonne partie de la logique de l'application dans la GUI.
- Héléna (7 heures réalisées VS 4.5 heures planifiées)
  Ce sprint a été le plus intéressant, car nous avons enfin eu accès au plateau de jeu après la création de la partie. De ce fait, tout s'est débloqué et les tâches peuvent être enchaînées jusqu'à la fin du projet, normalement, sans encombre. Il ne nous reste plus qu'à mettre en place les différentes règles du Monopoly et le jeu sera fin prêt.

6.4.6 Itération 6

a) Bilan sur la terminaison des histoires

- Histoires planifiées : 9 - 13 – 18 – 26 – 30 – 31 - 32
- Histoires terminées : 9 – 26 – 30 – 31 – 32
- Histoires non-terminées : 13 (réalisée à 90%) – 18 (réalisée à 95% - tests manquants)

b) Vélocité du sprint

histoires réalisées / 7 histoires planifiée ≈ 71%

c) Replanification

Les histoires qui ne sont pas complètement terminées ont été reportées dans l'itération 7. Le sprint a été presque entièrement réalisé. Pour la gestion de la case (histoire 13), il ne reste plus qu'à mettre en place une information visuelle sur chacune des cases pour informer le client des possessions de chacun. Pour l'hypothèque de propriété (histoire 18), elle a été complétement implémentée, mais n'a pas pu être testée et debuggée.

d) Commentaire général

Cette semaine aussi, nous avons pu donner le meilleur de nous-mêmes pour avancer au maximum ce projet. De ce fait, nous avons encore une fois pu rattraper notre retard et liquider le plus d'histoires possibles pour ne pas être débordés lors de la dernière itération. Nous sommes confiants pour le dernier sprint.

e) Autocritique

La cohésion de groupe a été très sollicitée, afin de terminer ces différentes histoires. Etant plus coordonnés que jamais, les histoires sont réalisées en chaîne, passant d'abord par les développeurs, puis par la GUI qui s'occupe de lier le tout dans un temps relativement rapide et acceptable. Nous sommes tous très contents d'avoir pu avancer autant et d'avoir rattraper le retard accumulé lors des premières semaines. Nous nous réjouissons de la fin du projet, afin de pouvoir contempler le fruit de notre dur labeur.

f) Bilans personnels

- Bryan (5 heures réalisées VS 4 heures planifiées)
  Une itération qui s'est bien passée de mon point de vue, Le serveur a continué à progresser, les problèmes étaient réglés relativement rapidement, et la mise en place des hypothèques s'est faite sans trop de soucis. Certes nous n'avons pas pu clore toutes les histoires, mais pour la plus grande partie de ces dernières il ne reste uniquement que les tests à valider, ou les liens avec la GUI à terminer. Je pense que nous n'aurons aucun problème à fournir un résultat fonctionnel et complet à la fin de l'itération suivante.
- Daniel (7 heures réalisées VS 7 heures planifiées)
  Pour ce sprint, j'ai énormément travaillé sur des tâches non prévues. En effet, j'étais censé m'occuper de plusieurs points coté serveur, comme par exemple l'achat de canapés (maison du monopoly) ou encore la détection de banqueroute et la vente de propriétés. Mais finalement, je n'ai fait que ce qui concernait les canapés / home cinémas (maisons et hôtels), puis le reste du temps je me suis occupé du lien entre les informations envoyées du serveur et l'affichage en GUI (la partie GUI pure étant effectuée par François Burgener). J'ai donc passé beaucoup de temps sur plusieurs parties précédemment implémentées sur le serveur mais pas encore implémentées du coté client. Et bien sûr, avec cela, tous les bugs liés à cette mise en commun du code et assemblage.

Globalement, cela s'est très bien passé. Bryan Curchod s'est occupé de certaines de mes tâches pendant que je travaillais sur cette partie imprévue et cela ne nous a pas retardé d'avantage, ce qui est une très bonne chose pour un avant dernier sprint. De ce fait, le programme tourne presque à la perfection, la plupart des histoires sont implémentées et quelques petits détails restent sur le feu. Je suis très heureux d'être arrivé jusqu'à ce point et de ne devoir faire l'impasse que sur un minimum d'histoires non indispensables au bon déroulement du jeu. J'ai hâte de terminer le dernier sprint et d'avoir une version totalement jouable, qui sera une belle récompense pour tout ce temps investi, ainsi que la note maximale pour ce magnifique projet.

- François (5 heures réalisées VS 5 heures planifiées)
  Cette itération s'est bien passée. Nous avons pu avoir la majorité des fonctionnalités de notre jeu. Nous avons pu les intégrer à la GUI. Pour ma part j'ai pas eu trop de difficultés à faire les tâches qui m'étaient assignées.
- Héléna (12 heures réalisées VS 4.5 heures planifiées)
  Ici encore, je n'ai pas eu de problème particulier lors du développement de mes histoires. Toutefois la gestion de la salle d'examen (envoi, sortie) était un peu compliquée, dans le sens où il fallait penser à beaucoup de cas et surtout ne pas oublier certains détails d'implémentation, ou encore il fallait bien implémenter la succession d'étapes à réaliser pour interagir correctement avec la salle d'examen. Je suis satisfaite de notre avancement, car nous arrivons enfin au bout avec un produit fonctionnel et agréable à utiliser.

6.4.7 Itération 7

a) Bilan sur la terminaison des histoires

- Histoires planifiées : 13 – 16 – 18 – 21 – 23 – 25 – 29 - 33 – 35
- Histoires terminées : 13 – 16 – 18 – 21 – 23 – 33 – 35
- Histoires non-terminées : 25 – 29 (réalisées à 0% - abandonnées)

b) Vélocité du sprint

7 histoires réalisées / 9 histoires planifiée ≈ 78%

c) Commentaire général

Ce sprint s'est terminé avec succès. Les deux histoires non commencées ont été un choix stratégique et sans conséquences pour le rendu final du projet, puisque nous fournissons quand même un produit abouti et fonctionnel. Les histoires 25 et 29 tiennent plus de l'amélioration que d'une fonctionnalité essentielle au projet. C'est pour cela que nous avons choisi de ne pas les implémenter, mais aussi parce que nous devions nous concentrer sur les autres histoires à terminer absolument.

d) Autocritique

Nous terminons ce projet sur une très bonne note, puisque nous avons implémenté toutes les histoires prévues au départ. Nous sommes très contents d'avoir mené à terme ce monopoly façon HEIG. Nous nous réjouissons de pouvoir l'utiliser pour nous amuser entre nous. Nous sommes contents de notre implémentation de ce jeu en réseau, qui est pour nous une première dans le domaine et pour un projet aussi conséquent. Nous sommes conscients que nous aurions pu faire mieux certaines choses, et nous en tiendrons compte pour les projets futurs.

e) Bilans personnels

- Bryan (9.5 heures réalisées VS 5 heures planifiées)

  Je pense que ce sprint était celui qui s'est le mieux passé, avec le tout premier. Nous nous sommes retrouvés pour travailler tous ensemble, on a bien progressé, beaucoup de problèmes ont été résolus (comme la fin d'une partie, ou alors la sortie d'une partie en cours). Une fois toute cette partie réglée, nous avons pu nous concentrer sur les autres branches sans se soucier de Cheseaux-Poly et ça nous a été grandement bénéfique. Bien que la finalisation du projet (documentation, rapport, présentation, ...) se soit faite un peu au dernier moment, je pense que nous nous sommes très bien débrouillés lors de cette itération. Le projet se termine donc sur une note très positive, notre jeu fonctionne et surtout on peut faire une partie sans trop de problème !

- Daniel (> 10 heures réalisées VS 6 heures planifiées)

  Pour ce dernier sprint, j'ai terminé les tâches que je n'avais pas pu terminer le plus rapidement possible, puis je me suis penché sur tous les petits bugs liés au gameplay (les problèmes dans le jeu directement, mauvais paiements, ...). Ensuite, je me suis penché sur deux plus gros problèmes liés au retour de l'utilisateur d'une partie au lobby. En effet, les notifications ne se relançaient pas correctement et cela est fort embêtant. Il y avait d'autres problèmes à ce niveau, que j'ai pu corriger sans trop de souci. Mais il me reste celui des notifications, que j'espère résoudre avant la présentation du projet !

  A part cela, le sprint s'est plutôt bien passé. Ce fût un dernier sprint plutôt "tranquille", nous pouvions directement voir ce qui changeait dans le jeu lors des modifications et globalement notre programme fonctionne correctement. Nous avons un serveur stable qui ne s'arrête que dans des cas extrêmement peu courants (les exceptions ne sont pas encore totalement et

proprement bien gérées). Notre client est aussi très stable et ne possède, à notre connaissance, que le problème évoqué. Tout le reste, mis à part quelques petits détails graphiques, fonctionne aisément.

Pour revenir sur ce projet de manière globale, je l'ai trouvé très formateur, la méthodologie mise en pratique est fort intéressante et me sera sans aucun doute bien utile pour mon émancipation professionnelle. Cela m'a fait plaisir de travailler avec cette équipe, nous avons eu des moments de stress et de désaccords, mais surtout beaucoup de moments de rires et de joie. En ce qui concerne le logiciel pour l'organisation et la mise en pratique de la méthodologie, IceScrum, je ne l'ai pas particulièrement apprécié. Je trouve qu'il a énormément de potentiel et d'idée, mais il n'est pas très intuitif et il manque un peu de réactivité ainsi que de certaines fonctionnalités basiques. Après, je suppose que la version payante propose ces fonctionnalités manquantes, mais je doute qu'elle soit plus optimisée et intuitive. Dommage. Peut-être essayerais-je de concevoir personnellement un tel outil, moins puissant mais plus réactif (challenge intéressant !). Je trouvais IceScrum un peu lourd pour ce que c'est.

- François (10 heures réalisées VS 6 heures planifiées)
  Cette itération finale s'est très bien déroulée. Pour ma part l'itération qui s'est le mieux passé, car on était réuni les quatre pendant toute une après-midi. Nous avons pu rapidement finir toutes les fonctionnalités de notre application pour ensuite procéder aux tests. Nous avons eu pas mal de petits bugs qui ont été corrigés assez rapidement.
- Héléna (> 10 heures réalisées VS 4.5 heures planifiées)
  Très bonne itération puisque nous avons pu boucler toutes les histoires initialement planifiées. Notre jeu est fonctionnel, abouti et agréable à utiliser. Nous avons d'ailleurs pu profiter d'y jouer lors de plusieurs tests de fin d'histoire. Je suis très contente d'avoir fait cette aventure avec mes camarades et d'avoir participé à un projet d'une telle envergure.

6.5 Stratégie de tests

La totalité des tests de notre projet a été effectuées par le duo formé de Héléna Reymond et Daniel Gonzalez Lopez. Ils se tournaient tout de même vers François et moi-même s'ils en avaient besoin (question d'implémentation, d'interface graphique, ...). Les premiers tests, consistant simplement à connecter le client au serveur et effectuer quelques communications, ont été réalisés à l'aide de tests JUnit. Puis nous avons utilisé l'invite de commande (notamment l'outil Telnet) pour tester les réponses et les communications du serveur. Enfin une fois que ces éléments étaient validés, les derniers tests nécessitaient de passer par l'interface graphique. Généralement les tests étaient effectués entre le jeudi et le vendredi.

Nous avions testé la classe Client qui permet d'établir la connexion au serveur, ainsi que de s'enregistrer et se connecter à un compte existant. Tous les tests passent.

6.6 Stratégie d'intégration du code de chaque participant (GIT)

Afin d'intégrer et de gérer le code de chaque participant du projet, nous avons choisi d'utiliser l'outil de versionning Git. Tout d'abord car c'est un outil que nous avions déjà tous utilisé par le passé, mais surtout car il gère automatiquement l'intégration de chaque ligne de code des fichiers sources. Enfin, pour nous faciliter la visualisation et la gestion, nous utilisions l'outil GitKraken qui fournit une interface graphique pour l'utilisation de Git.

7 Etats des lieux

7.1 Ce qui fonctionne

Connexion	L'utilisateur entre un nom d'utilisateur et un mot de passe pour se connecter	Réussi
Enregistrement	L'utilisateur entre un nom d'utilisateur et un mot de passe pour s'enregistrer	Réussi
Communication avec le serveur	Le client doit être capable de se connecter au serveur, lui transmettre des commandes/messages, lire et interpréter les réponses de connexion	Réussi
Création d'un salon de jeu	L'utilisateur peut créer un salon de jeu et s'affiche dans la liste des salons de jeu. Les bonnes commandes sont envoyées au serveur	Réussi
Vérifier les notifications serveur	Vérifier que l'interface se met bien à jour lorsque le serveur envoie des notifications pour le salon de jeux.	Réussi
Rejoindre un salon de jeu	Vérifier que le client envoie les bonnes commandes pour rejoindre un salon de jeu et se déclarer prêt. Vérifier que l'utilisateur rejoint le salon de jeu(gui).	Réussi
Lancer les dés	Le client informe le serveur qu'il est actif, et que le serveur peut lancer les dés. Le serveur renvoie X nombre aléatoire.	Réussi
Déplacement du pion	Le serveur doit déplacer le pion du joueur conformément au lancer de dés.	Réussi
Gestion des cases coins	Lorsque le joueur arrive sur la case start il reçoit de l'argent	Réussi
Lancement de la partie	Lorsque tous les joueurs sont prêts (2 joueurs minimum), on envoie le plateau de jeu du coté serveur et on affiche le plateau de jeu (GUI)	Réussi
Gestion des case	En tombant sur les cases, le bon effet est appliqué, taxe à payer, loyer à payer, recevoir de l'argent ou tirer une carte.	Réussi
Achat terrain	A l'achat d'un terrain : - L'acheteur voit son argent diminuer	Réussi
Déplacement du pion GUI	Lorsque le joueur lancer les dés, le pion se déplace bien de la somme de tous les dés, ou alors avec un effet d'une carte qui lui demande de se déplacer de x case ou d'aller directement à une certaine case.	Réussi
Gestion des case coin	Lorsque le joueur arrive sur la case Envoie en salle d'examen le joueur va en examen.	Réussi
Gestion salle d'examen	Lorsque le joueur fait trois double de suite il est directement déplacé en salle d'examen. Lorsqu'il est en examen est qu'il fait un double il peut sortir d'examen. Après trois tours de suite le joueur peut sortir de prison	Réussi
Affichage des informations de la partie	La GUI récupère toutes les informations de la partie et les affiches. Lorsque je clique sur une case du jeu	Réussi

	Je peux voir toutes ses informations (nom, propriétaire, prix, nombre de bâtiments, hypothèque, etc.)	
Gestion de Banque route	Lorsque c'est mon tour et que je suis en banqueroute Un message m'informe que je n'ai plus d'argent et que je dois me séparer de biens pour ne pas perdre la partie. Si le joueur	Réussi
Gestion des hypothèque	Le joueur possédant une propriété, peut la mettre en hypothèque, par conséquent il gagnera l'argent de cette dernière. Ou la déshypothéquer et perdre de l'argent. Le loyer est bloqué si un joueur tombe sur une propriété hypothéquée	Réussi
Zone administrateur	Lorsque j'effectue la suite de raccourcis clavier dans l'application. La fenêtre de modification des paramètres généraux se présente à moi. Lorsque je change les paramètres généraux. Ceux-ci sont correctement modifiés en base de données et dans le jeu	Réussi
Fin de partie	Lorsqu'il reste plus qu'un joueur, la partie se termine. Lorsque je quitte la partie je suis redirigée sur la listes des salons de jeu	Réussi

7.2 Ce qu'il resterait à développer

7.2.1 Ce qu'il reste à développer

1 - Génération aléatoire du plateau : Dans les paramètres accessibles par la zone administrateur réside une option de "génération aléatoire". Cette option permettrait aux joueurs de faire une partie sur un plateau "chamboulé", c'est à dire que les cases ne seraient pas à leur place habituelle. C'est une fonctionnalité à laquelle nous avons rapidement pensé au début du projet, mais que l'on a également rapidement laissé de côté à cause de la charge de travail que cela impliquerait.

2 - Gestion des scores : La base de données possède déjà les prérequis pour le stockage des scores et autres exploits des joueurs (par exemple, le total des gains et des pertes). Il ne reste qu'à implémenter l'enregistrement de ces statistiques dans le serveur et de donner un accès à ces données aux clients.

3 - Gestion d'inactivité : Encore une fonctionnalité à laquelle nous avions pensé dès le début du projet. Malheureusement le retard engrangé nous a forcé à mettre cette option de côté. Le but de la gestion d'inactivité d'un joueur est d'empêcher de bloquer une partie dans le cas où un joueur est absent. Nous laisserions un délai d'une minute pour lancer les dés, puis encore une minute (ou plus) pour qu'il puisse effectuer ses actions libres (achat, vente, hypothèque). Au bout de ce délai son tour serait simplement passé, au bout de la deuxième fois il recevrait une pénalité sous la forme d'une taxe. Enfin à la troisième fois, le joueur serait purement et simplement expulsé de la partie.

4 - Interface graphique : Une amélioration qui rendrait l'expérience de jeu plus agréable, ajouter des images sur les cases, mettre des pions avec des petites icônes etc. En soit cela ne changerait rien au "gameplay", mais quand ça plait à l'œil, c'est tout de suite plus attrayant.

5 - Mise aux enchères : Officiellement, dans les règles du monopoly, lorsqu'un joueur atterrit sur une case achetable (propriété, gare, entreprise), il a le choix de l'acheter ou non. Dans la négative, la case est l'objet d'une enchère à laquelle tous les joueurs participent. Chacun mettrait un prix et aurait la possibilité de surenchérir, le gagnant de l'enchère remporte la propriété. Il serait intéressant d'implémenter cette fonctionnalité dans notre projet, avec évidemment un temps limite pour enchérir, en cas de temps écoulé le joueur ne pourrait plus participer aux enchères.

6 - Échanges entre joueurs : Une fonctionnalité qui est presque essentielle au monopoly. Un joueur pourrait entamer des négociations avec un autre pour discuter de la possession d'une propriété (ou d'une carte "libération de prison"), moyennant de l'argent ou une autre propriété. A nouveau, un système de compte à rebours permettrait d'éviter de bloquer tous les joueurs pour une transaction.

7 - Boite de discussion : dans le cas où les joueurs ne sont pas dans la même pièce, il peut être très intéressant de communiquer par le biais d'une boite de chat. Elle faciliterait également des opérations telles que les échanges entre joueurs.

7.2.2 Eventuelle planification

Afin de mettre en place les fonctionnalités citées, voici une planification qui nous semble raisonnable :

Itération 8-9 : Gestion d'inactivité (No 3) - Cette fonctionnalité requiert beaucoup d'efforts et permettra d'enchainer sur plusieurs autres options. c'est pourquoi nous devrions passer un temps considérable à développer et tester cette amélioration.

Itération 10 : Mise aux enchères (No 5) - le mécanisme de compte à rebours étant déjà implémenté, la mise aux enchères devrait prendre un certain temps mais nous pensons que nous sommes capables, avec une certaine organisation, de le faire en une itération.

Itération 11 : Boite de discussion (No 7 & 4) : la boite de chat ne représente pas vraiment un défi d'implémentation, nous pensons donc ne pas avoir à passer beaucoup de temps à sa réalisation. Nous en profitons donc pour investir le temps restant à l'amélioration de l'interface graphique.

Itération 12 : Echange entre joueurs (No 6) - Cette fonctionnalité peut représenter un certain effort en termes de travail, même si le mécanisme de compte à rebours est déjà en place. Nous y consacrerions donc toute une itération afin de pouvoir fournir un résultat plus que satisfaisant.

Itération 13 : Gestion des scores & Génération aléatoire du plateau (début) (No 2 & 1) - la gestion des scores est certes une étape relativement conséquente, mais pas assez pour remplir toute une itération. Nous pourrions donc commencer l'implémentation de la génération aléatoire du plateau.

Itération 14 : Génération aléatoire du plateau (fin) (No 1) - Finalisation de la génération commencée lors de l'itération 13.

8 Auto-critique

En ce qui concerne notre solution technologique, elle est globalement bien implémentée. Nous avons fait un gros travail de conception en amont de sorte qu'il y ait le moins de retouches à faire. Et effet, nous avons réussi à bien concevoir et comprendre ce que nous voulions avant de commencer l'implémentation, ce qui fait que nous sommes souvent partis directement dans la bonne direction. De plus, la conception étant bien faite, cela a beaucoup facilité les retouches nécessaires lorsque nous nous sommes retrouvés dans des situations difficiles.

Pour donner un exemple, notre application possède une communication client – serveur sur deux canaux, un canal pour les commandes du client et les réponses du serveur et un canal pour les notifications (asynchrones) du serveur au client. Au début, nous avions conçu cette partie avec deux threads différents pour chaque canal, que ce soit du côté serveur comme du coté client. Mais nous nous sommes vite rendu compte que cela demandait une bien trop grande ressource à cause des multiples threads, deux par client. Nous avons donc repensé le serveur pour qu'il n'utilise pas de thread actif pour envoyer les notifications, mais simplement une classe qui notifiait à chaque fois qu'il y avait un changement. Ce changement n'a demandé qu'une adaptation de la classe en question et rien d'autre.

Notre solution est assez performante et ne demande donc pas excessivement de ressources. Par contre, nous pourrions améliorer le design de celle-ci, notamment pour la rendre un peu plus intuitive (avec des messages d'aide, en rajoutant des raccourcis ou encore en rajoutant la possibilité de nommer les parties ou de pouvoir rechercher un utilisateur). En ce qui concerne le code et l'implémentation de notre solution, nous pourrions clairement en améliorer la clarté, ce qui permettrait un maintien plus facile de ce dernier. Nous pourrions mieux gérer les exceptions, nettoyer le projet et généraliser le code, ce dernier étant pour le moment rempli de bouts de codes se ressemblant, de méthodes servant simplement de redirection, car nous avons fait une hiérarchie de classes. Par exemple, la GUI ne s'adresse pas directement à la classe qui communique avec le serveur, mais passe par une autre classe. Et du coup, certains noms de méthodes se retrouvent sur plusieurs classes alors que nous pourrions clairement simplifier un peu les interactions au niveau du client. En ce qui concerne le serveur, celui-ci est plus optimisé en termes de liaisons entre les classes et est plus difficilement améliorable.

Pour en finir avec la partie technologique, nous sommes assez fiers de notre conception du produit et de sa réalisation finale. Bien évidemment qu'il y a beaucoup d'améliorations qui peuvent être apportées, mais pour un projet sur 7 semaines et d'une telle envergure, sa conception et le résultat sont plutôt encourageants.

Concernant la gestion du projet et la planification, nous avons suivi au maximum la méthodologie souhaitée. Nous avons donc fait un gros travail de planification au début pour préparer les histoires et leurs tâches correspondantes, nous avons aussi fait un maximum d'estimations pour celles-ci et pour celles qui n'avaient pas totalement été estimées avant le début du sprint 1, nous l'avons fait avant chaque début de sprint concerné.

Notre plus gros souci, de manière globale, c'est que nous avons énormément axé notre travail sur la méthodologie au début puis, plus le temps passait, moins on prenait de temps pour la partie gestion. Cela vient bien évidemment en partie du fait qu'un grosse partie de la gestion est faite en amont, mais cela est probablement aussi dû au fait que ce projet était un peu trop conséquent pour être réalisé en si peu de

temps et nous avons donc concentré nos efforts dans l'implémentation de la solution plutôt que dans les bilans d'itérations.

De ce fait, et c'est une amélioration importante à apporter à de futurs projets, il faudrait que nous prenions vraiment le temps de faire toute la partie de débriefing, d'analyse du sprint précédent et de la planification du suivant, et cela même si nous avons un retard important. Nous trouvons dommage d'avoir laissé cette partie un peu de côté, car ce genre d'exercice est toujours excellent pour apprendre de ses erreurs et éviter de retomber dans les pièges. C'est donc un point que nous mettrons en place au prochain gros travail, que ce soit au sein de cette même équipe ou d'une autre.

Un autre problème, lié au précédent, est que du fait de ne pas faire les séances de débriefing, nous faisions le planifications et changements plus au feeling qu'autre chose. Bien sûr, si nous avons agi comme cela c'est aussi en partie dû au fait que le projet avançait globalement bien et que nous nous voyions tous les jours, ce qui nous permettait d'en discuter à tout moment. Mais nous sommes persuadés que de se forcer à faire une petite séance de débriefing à la fin de chaque sprint peut être extrêmement bénéfique.

Un dernier gros problème, concernant les planifications de chaque sprint, c'est que nous avions tendance à écrire les tâches pour la partie serveur, car c'était la partie logique des tâches, et d'en oublier totalement la répercussion du côté client. Nous avons donc presque tout le temps dû rajouter des tâches concernant la partie client en milieu de sprint ou alors pour le sprint suivant. C'est aussi un point pour lequel nous serons tous beaucoup plus méticuleux, cela nous évitera d'avoir de mauvaises surprises et de se retrouver avec une charge de travail extraordinaire. Nous sommes bien évidemment conscients qu'il y a toujours des tâches qui surviennent dans de tels projets, mais en ce qui concernait les tâches côté client, c'était clairement une erreur d'inattention de notre part.

Pour en conclure avec la méthodologie et le déroulement du projet, nous pouvons dire que nous sommes conscients des erreurs que nous avons commises ainsi que de l'importance d'améliorer ces points-là, l'apport bénéfique étant considérable. Nous serons bien plus vigilants pour les prochains projets pour ne pas retomber dans les mêmes pièges et mieux gérer notre équipe et nos ressources.

9 Conclusion

Pour conclure ce rapport, nous soulignerons que ce projet nous a été très utile pour deux grandes raisons. Tout d'abord, cela nous a permis de nous exercer au niveau de la planification et de la conception d'un projet assez conséquent au sein de nos études. Nous avons eu un autre grand projet ce semestre que nous avons réalisé sans connaître les techniques de génie logiciel pour lequel cela nous aurait été très utile. Ce deuxième projet conséquent nous exerce donc une nouvelle fois à toutes ces tâches de planification et de gestion d'équipe, ce qui est excellent pour nous habituer et nous faire gagner les l'expérience. Ensuite, ce projet nous a permis de mettre en œuvre une méthodologie de travail que nous avions récemment apprise. Certes, nous ne la maîtrisions pas encore et certains concepts nous ont échappés. Mais c'est également cela qui est intéressant. En effet, certains aspects de la méthodologie que nous avons utilisée nous ont montré leur utilité et efficacité. Et même pour ceux que nous n'avons pas utilisés, nous en comprenons désormais la nécessité et tâcherons de les mettre en place pour les prochains projets.

Plus globalement, ce projet fût très enrichissant. Nous ne cessons d'apprendre des choses grâce à ce genre de projet et le travail d'équipe. C'est toujours agréable de travailler en groupe, d'apprendre à communiquer avec les autres, à utiliser des outils nouveaux nous permettant d'être plus efficaces dans nos tâches (git/github, IceScrum, maven, …), s'adapter aux contraintes et à l'équipe, mettre en place des technologies et suivre des méthodologies. Toutes ces choses nous font gagner en expérience et nous permettront, dans nos vies professionnelles, d'être efficaces et de trouver des solutions pour chaque problème rencontré. Nous sommes donc sincèrement reconnaissant pour ce cours et ce projet et sommes fier du résultat obtenu.

10 Annexes

10.1 Manuel d'installation

10.1.1 Prérequis

Afin de pouvoir jouer à Cheseaux-Poly, voici ce dont vous avez besoin :

- Cheseaux-Poly_Server.jar
- Cheseaux-Poly_Client.jar
- server_config.properties
- client_config.properties
- script_db.sql
- Avoir installé le SDK de Java 10

Soyez attentifs au fait que, si vous voulez déployer le jeu sur plusieurs ordinateurs au lieu d'un seul, il vous faudra ouvrir les ports nécessaires que ce soit pour la base de données ou le serveur, afin que les connexions se fassent correctement entre les différents réseaux.

En revanche si vous souhaitez tester le jeu sur un même ordinateur, veillez-vous à utiliser un ordinateur pouvant supporter tous ces acteurs en même temps car le jeu, utilisant des threads, peut parfois demander beaucoup de ressources à la machine.

10.1.2 Mise en place de la base de données

Rien de bien compliqué pour mettre en place la base de données. Il suffit d'exécuter le script SQL dans le SGBD de votre choix. Celui-ci va se charger de créer les différentes tables requises pour le jeu et de créer des données telles que les paramètres généraux du jeu, les cartes et les cases du plateau.

Comme mentionné précédemment, n'oubliez pas d'ouvrir un port pour MySQL si le serveur ne se situe pas sur le même ordinateur que la base de données. De plus, dans ce cas-ci, il vous faudra créer un utilisateur, pouvant provenir de n'importe quelle adresse extérieure (0.0.0.0), ayant tous les droits sur la base de données générée.

Que le jeu soit sur plusieurs ordinateurs ou un seul, il vous faudra renseigner un utilisateur de votre SGBD ayant accès à la base de données, dans le fichier de configuration du serveur que nous verrons après.

10.1.3 Mise en place du serveur de jeu

Le serveur n'ayant pas d'interface graphique, nous vous recommandons de le lancer depuis un terminal, afin d'avoir un aperçu des différents logs générés des différentes interactions entre les clients et lui-même (cette remarque est aussi valable pour le lancement du client, si vous voulez voir les logs).

Mais avant de faire cela, il vous faudra configurer le serveur. Ouvrez le fichier de configuration « server_config.properties » et renseignez l'adresse du serveur (localhost suffit), ainsi que l'adresse IP de la base de données, son nom, son port et l'utilisateur qui permet de s'y connecter. **Ne séparez jamais l'exécutable du serveur de son fichier de configuration**. Le serveur et la base de données n'ayant pas pu être mis sur un serveur distant, ils changent d'adresse à chaque déploiement du jeu. De ce fait il faut le reconfigurer à chaque fois (cette remarque est aussi applicable au cas du client).

D'une fois que cela a été fait, vous pouvez démarrer le serveur comme suit :

1) Ouvrir un terminal
2) Se déplacer dans le dossier contenant le .jar et le .properties
3) Lancer la commande « java -jar Cheseaux-Poly_Server.jar »

10.1.4 Mise en place du client

Comme pour le serveur, avant de lancer votre client, il vous faudra configurer le fichier « client_config.properties ». Il vous suffit juste de renseigner les ports et l'adresse IP utilisés par le serveur.

Une fois cela effectué, double-cliquez sur l'exécutable et c'est parti pour une super partie de Cheseaux-Poly !

10.2 Manuel d'utilisation

Pour des raisons de présentation, nous avons joint le manuel d'utilisation à la fin de ce rapport.

10.3 Retours sur les itérations par l'enseignant

10.3.1 Itération 1

10.3.1.1 No 1 - Login de l'utilisateur

en tant que Joueur
Je veux me connecter
afin de pouvoir rejoindre un salon

Démo1: Enregistrement
*Given*
L'utilisateur entre un nom d'utilisateur et un mot de passe pour s'enregistrer
*When*
Lorsque que l'utilisateur clique sur le bouton
*Then*
☑ L'utilisateur arrive dans le menu des salons de jeu, si le nom d'utilisateur est utilisé, on affiche un message d'erreur

Observé:
Ca fonctionne, avec les messages d'erreur côté client et signalisation en console dans le serveur

Démo2: Connexion
*Given*
L'utilisateur entre un nom d'utilisateur et un mot de passe pour se connecter
*WHEN*
Lorsque que l'utilisateur clique sur le bouton
*Then*
☑ L'utilisateur arrive dans le menu des salons de jeu, si le nom d'utilisateur et le mot de passe ne correspond pas, on affiche un message d'erreur

Observé:
Ca fonctionne, avec les messages d'erreur côté client et signalisation en console dans le serveur

10.3.1.2 No 28 - Mise en place du client

Démo1: Communication avec le serveur
☑ Le client doit être capable de se connecter au serveur,
☑ lui transmettre des commandes/messages,
☑ lire et interpréter les réponses de connexion

Observé:
Interface avec login et mot de passe et un bouton, le tout en couleur, dont le look change selon le mode enregistrement ou connexion.

Dès que le client est lancé, ce dernier se connecte au serveur.

10.3.1.3 No 2 - Base de données - Technical story

La base de données doit être crée, les tables initialisées, peut-être des données tests, requête pour créer les utilisateurs, et autres UPDATE

Démo1: Requête insertion utilisateur
☑ On teste manuellement la requête d'insertion d'un utilisateur.
☑ On vérifie qu'un utilisateur a été inséré dans la base de donnée

Observé:

Fonctionnel
Toute la base de données est déjà mise en place
Les tests "manuels" se font via le terminal.

Démo2: Requête de correspondance du mot de passe avec le nom d'utilisateur
- ☑ On teste manuellement la requête qui vérifie qu'un mot de passe correspond à un nom d'utilisateur.
- ☑ On regarde que cette requête nous retourne un seul ID.

Observé:
Fonctionnel

10.3.1.4 No 27 - Mise en place du serveur

Démo1: Démonstration
- ☑ Le serveur doit répondre correctement aux réponses
- ☑ il doit se connecter à la base de données
- ☑ il doit récupérer les données de la base de données

Observé:
Les messages de connexion, d'enregistrement sont signalés en console

Démo2: Enregistrement manuel d'un utilisateur
Lancer le serveur
Ouvrir une fenêtre du terminal
Se connecter au serveur (telnet [address] [port])
Écrire la commande :
RGSTR [username] [password_digest]
- ☑ Si username disponible, le serveur doit répondre :
  - ☑ OK
  - ☑ Aller voir dans la base de données si le nouvel utilisateur a été créé.
- ☑ Sinon :
  - ☑ DENIED

Observé:

Démo3: Connexion manuel d'un utilisateur
Lancer le serveur
Ouvrir une fenêtre du terminal
Se connecter au serveur (telnet [address] [port])
Écrire la commande :
LOGIN [username] [password_digest]
- ☑ Si username existant et password correspondant, le serveur doit répondre :
  - ☑ OK
- ☑ Sinon :
  - ☑ Si aucun username correspondant :
    - ☑ UNKNOWN
  - ☑ Si mot de passe erroné :
    - ☑ DENIED

Observé:

Démo4: Vérifier le protocole

☑ Tester toutes les commandes du protocole et vérifier les réponses envoyées par le serveur.

Observé:
Testé en console, message par message.

Remarque: un test unitaire aurait pu être mis en place.

Démo4: 2 requêtes de connexion en //
Ouvrir 2 terminaux
Connecter au serveur le premier terminal
☑ Tester d'envoyer une requête login par exemple
Connecter au serveur le deuxième terminal
☑ Vérifier si les deux terminaux sont gérés par le serveur en parallèle (en envoyant des commandes depuis les deux)

Observé:
Testé avec telnet alors qu'un autre client est connecté

10.3.1.5 Bilan général

Tout ok.
Des tests unitaires ont été mis en place, notamment pour le client.
Base client-serveur "fait main" en se basant sur les labos RES.

10.3.2 Itération 2

10.3.2.1 No 4- Création d'un salon de jeu - User story

en tant que Joueur

je veux créer mon propre salon de jeu

dans le but d'accueillir des joueurs et de jouer une partie avec des paramètres personnalisés

Démo1: Le client doit bien envoyer la commande de création

- [x] Faire un test unitaire qui crée manuellement une partie

  Puis qui utilise le client pour envoyer sa commande.

- [x] Vérifier que la commande reçue par le serveur soit correcte (LOG)

  Observé:

Démo2: La GUI doit bien utiliser les méthodes disponibles

Faire un test en créant une partie depuis l'interface graphique.

- [x] Le serveur doit retourner une confirmation

  Observé:

Démo3: Le serveur doit interpréter et créer la partie

- [x] Lors de la réception d'une commande de création, le serveur doit correctement l'interpêter et créer la partie correspondante, puis retourner une confirmation au client.

- [x] Vérifier la création de la partie directement sur le serveur (debug mode) et l'envoi de la confirmation.

  Pour ce faire, créer une partie depuis un terminal directement (pour pouvoir voir les réponses du serveur et éviter les problèmes coté client)

  Observé:

Démo4: GUI utilise les bons paramètres

- [x] Vérifier que la GUI affiche les bons paramètres récupérés de la base de données

- [x] La GUI ne doit pas pouvoir dépasser les limites imposées.

  Observé:

10.3.2.2 No 5- Paramétrage d'un salon de jeu - User story

en tant que Joueur

je veux utiliser des paramètres personnalisés

dans le but de faire une partie amusante

Démo1: Les limites de paramètres sont parfaitement récupérées

☑ Les limites de paramètres stockées dans la base de données sont bien récupérées et reportées dans l'interface de création de partie

Observé:

Démo2: Les paramètres sont correctement modifiables

☑ Le créateur de la partie peut choisir ses propres paramètres et les enregistrer pour la partie créée

Observé:

10.3.2.3 Bilan général

Tout réalisé comme prévu.

10.3.2.4 Bilan itération précédente

Très bon bilan, très complet!

Note: Héléna et Daniel on travaillé de nombreuses heures (resp. 9 et 8) --> Vacances prochaines?

Revoir toutefois la valeur de la vélocité: il s'agit simplement de signaler le nombre de points d'histoires terminées (par un %)

10.3.2.5 Echange avec Christophe

10.3.2.5.1 7 Mai

Hello,

Tout est ok pour votre sprint (temps, descriptif, etc.)

Juste peut-être préciser

- *Création d'un salon de jeu*
  - *Test : GUI utilise les bons paramètres → quels paramètres devons-nous voir ? Il s'agit du choix de mode et du nombre de dés ?*

Idem sur

- *Paramétrage d'un salon de jeu → peut-être préciser ce qui est paramétrable à ce niveau. Il est indiqué « les limites de jeu fixées par l'administrateur ». Quelles sont ces limites ?*

Sinon pour le reste c'est très bien et bien précis, top.

Merci bien

10.3.3 Itération 3

10.3.3.1 No 3- Liste des salons de jeu - Technical story

en tant que Joueur

je veux obtenir la liste des salons de jeu afin de pouvoir en rejoindre un

Démo1: Vérifier les notifications serveur

Créer, à l'aide de plusieurs terminaux, des parties sur le serveur.

- ☐ Puis, à l'aide d'un autre terminal, qu'on connecte au port de notification, vérifier que les bonnes infos sont envoyées, au bon format.
- ☐ Une fois connecté, créer une autre partie et vérifier que le terminal connecté aux notifications reçoit la mise à jour.

Observé:

Démo2: Vérifier la mise à jour niveau GUI

- ☐ Vérifier que l'interface se met bien à jour lorsque le serveur envoie des notifications pour le salon de jeux.
- ☐ Cela vérifie également que le Client interpète bien les informations reçues du serveur.

Observé:

10.3.3.2 No 6- Rejoindre un salon de jeu - User story

en tant que Joueur

je veux rejoindre un salon de jeu

pour jouer avec des personnes

Démo1: Vérifier la commande client

Vérifier que le client envoie la bonne commande

- ☑ Pour rejoindre une partie
- ☑ Pour se déclarer prêt

Observé:

Démo2: Vérifier l'interprétation du serveur

- ☑ Vérifier que lorsque le serveur reçoit une commande d'un client pour rejoindre une partie, celui-ci vérifie bien que la partie soit joignable et qu'il ajoute bien l'utilisateur à la partie.

Observé:

Démo3: Vérifier la GUI

- ☐ Vérifier que, lorsqu'un utilisateur rejoint un salon de jeu, ce dernier s'affiche bien dans l'interface pour le bon salon.
- ☐ Vérifier que les utilisateur du salon sont bien notifié et que l'affichage se fait correctement

Observé:

10.3.3.3 No 9- Préparer la partie - Technical story

Le serveur va préparer la partie cela inclut :

- l'établissement de l'ordre de passage

- génération du deck de carte chance

- répartition de la fortune individuelle

- génération du plateau

Démo1: Le serveur informe le premier joueur

- ☐ une fois la préparation terminée, le serveur informe le premier joueur que c'est à son tour de jouer

Réception de la commande PLAY de la part du serveur.

Observé:

Démo2: Génération d'une "file" de joueurs

création d'une file qui indique l'ordre des joueurs.

- ☐ Chaque joueur ne doit y apparaître qu'une seule fois.
- ☐ La taille de la file doit être égale au nombre de joueurs.

Observé:

Démo3: Répartition de la fortune

- ☐ Tous les joueurs possèdent la même somme.
- ☐ La somme est conforme aux paramètres saisis à la création du salon.

Observé:

Démo4: Génération du plateau

créer le conteneur des cases du plateau.

- ☐ Toutes les cases doivent apparaître une et une seule fois.
- ☐ L'ordre des cases doit respecter la disposition d'un monopoly classique

Observé:

Démo5: Génération du deck de carte

Création d'une pile de "carte".

- ☐ Les cartes utilisées doivent être celles de la DB.
- ☐ La présence de plusieurs exemplaires d'une même carte est acceptée dans le deck.

Observé:

Démo6: Affichage du plateau

lors du début de la partie la GUI affiche le plateau de jeu.

☐ Le plateau affiché correspond au plateau généré

Observé:

Démo7: Affichage du pateau de jeu

☐ Affichage du plateau de jeu avec les données récupérer du serveur

Observé:

10.3.3.4 No 8- Lancement de la partie - Technical story

Le serveur envoie un signal à tous les joueurs pour les informer que tous les joueurs sont prêts et que la partie va commencer

Démo1: Lancement de la partie (GUI)

☐ Lorsque tout les joueurs sont prêt (2 joueurs minimum) , on ouvre une nouvelle fenêtre qui sera notre partie (avec le plateau reçu du serveur)

Observé:

Démo2: Bon envoi du plateau

☐ Lorsque les joueurs sont prêts, vérifier que le serveur envoie bien le plateau à tous les joueurs de la partie.

☐ Vérifier du coté serveur grâce aux logs et du coté client avec le plateau généré (breakpoint après la réception et la sérialisation pour voir si c'est le bon et le même entre serveur et joueurs)

Observé:

Démo3: Lancer la partie au bon moment

Vérifier que le serveur lance bien la partie d'après les conditions définies.

☐ S'il y a deux personnes au minimum prêt, lancer la partie quoiqu'il arrive après 2 minutes.

☐ S'il y a plus de 2 joueurs et qu'ils sont tous prêts, lancer la partie dès que le dernier se déclare prêt.

Observé:

10.3.3.5 Bilan général

Histoire no 3: Réalisée partiellement à 50% (envoi ok, réception à faire)

Histoire no 6: Réalisée partiellement à 80% (manque la GUI)

Histoire no 9: Réalisée partiellement à 70% (manque la génération de plateau)

Histoire no 8: Réalisée à 0%

Replanification globale en vue

Difficultés rencontrées: mise en œuvre des notifications.

Bon espoir que le retard rencontré puisse être rattrapé, l'équipe est confiante

10.3.3.6 Bilan itération précédente

Idem sprint précédent: Très bon bilan, très complet!

Note: Héléna et Daniel ont travaillé de nombreuses heures (resp. 9 et 8) --> Vacances prochaines?

10.3.3.7 Echange avec Christophe

10.3.3.7.1 15 Mai

Hello,

C'est parfait pour le sprint 3, tout est ok. Faites quand-même attention au nombre d'heures (vous êtes à 22h pour 4 quand-même)

Bonne continuation

++

10.3.4 Itération 4

10.3.4.1 No 3- Liste des salons de jeu

Le serveur va préparer la partie cela inclut :

- l'établissement de l'ordre de passage
- génération du deck de carte chance
- répartition de la fortune individuelle
- génération du plateau

Démo1: Vérifier les notifications serveur

Créer, à l'aide de plusieurs terminaux, des parties sur le serveur.

Puis, à l'aide d'un autre terminal, qu'on connecte au port de notification, vérifier que les bonnes infos sont envoyées, au bon format.

☑ Une fois connecté, créer une autre partie et vérifier que le terminal connecté aux notifications reçoit la mise à jour.

Observé: Fonctionnel

Démo2: Vérifier la mise à jour niveau GUI

☑ Vérifier que l'interface se met bien à jour lorsque le serveur envoie des notifications pour le salon de jeux.

☑ Cela vérifie également que le Client interprète bien les informations reçues du serveur.

Observé: Fonctionnel

10.3.4.2 No 9- Préparer la partie

Le serveur va préparer la partie cela inclut :

- l'établissement de l'ordre de passage

- génération du deck de carte chance

- répartition de la fortune individuelle

- génération du plateau

Démo1: Le serveur informe le premier joueur

une fois la préparation terminée, le serveur informe le premier joueur que c'est à son tour de jouer

☐ Réception de la commande PLAY de la part du serveur.

Observé:

Démo2: Génération d'une "file" de joueurs

création d'une file qui indique l'ordre des joueurs.

Chaque joueur ne doit y apparaître qu'une seule fois.

☐ La taille de la file doit être égale au nombre de joueurs.

Observé:

Démo3: Répartition de la fortune

Tous les joueurs possèdent la même somme.

☐ La somme est conforme aux paramètres saisis à la création du salon.

Observé:

Démo4: Génération du plateau

créer le conteneur des cases du plateau.

Toutes les cases doivent apparaître une et une seule fois.

☐ L'ordre des cases doit respecter la disposition d'un monopoly classique

Observé:

Démo5: Génération du deck de carte

Création d'une pile de "carte".

Les cartes utilisées doivent être celles de la DB.

☐ La présence de plusieurs exemplaires d'une même carte est acceptée dans le deck.

Observé:

Démo6: Affichage du plateau GUI

lors du début de la partie la GUI affiche le plateau de jeu.

☐ Le plateau affiché correspond au plateau généré

Observé:

Démo7: Affichage du pateau de jeu

☐ Affichage du plateau de jeu avec les données récupérer du serveur

Observé:

10.3.4.3 No 8- Lancement de la partie

le serveur envoie un signal à tous les joueurs pour les informer que tous les joueurs sont prêts et que la partie va commencer

Démo1: Lancement de la partie (GUI)

☐ Lorsque tout les joueurs sont prêt (2 joueurs minimum) , on ouvre une nouvelle fenêtre qui sera notre partie (avec le plateau reçu du serveur)

Observé:

Démo2: Bon envoi du plateau

☐ Lorsque les joueurs sont prêts, vérifier que le serveur envoie bien le plateau à tous les joueurs de la partie.

- [ ] Vérifier du coté serveur grâce aux logs et du coté client avec le plateau généré (breakpoint après la réception et la sérialisation pour voir si c'est le bon et le même entre serveur et joueurs)

Observé:

Démo3: Lancer la partie au bon moment

- [ ] Vérifier que le serveur lance bien la partie d'après les conditions définies.
- [ ] S'il y a deux personnes au minimum prêt, lancer la partie quoiqu'il arrive après 2 minutes.
- [ ] S'il y a plus de 2 joueurs et qu'ils sont tous prêts, lancer la partie dès que le dernier se déclare prêt.

Observé:

Histoire non terminée

10.3.4.4 No 12- Déplacement du pion

Lancer les dés.

(obligatoirement 2 pour gérer les doubles !)

génération de deux entiers entre 1 et 6.

Une fois les dés lancés on déplace le pion du joueur d'autant de cases que le résultat du lancer indique

Lors d'un double, le joueur peut rejouer.

Après 3 doubles d'affilée, le joueur est expédié en prison

Démo1: Le client demande à lancer les dés

Le client informe le serveur qu'il est actif, et que le serveur peut lancer les dés

Observé: Fonctionnel

Démo2: Le serveur lance les dés

Gestion du nom de dés

Génération de X nombres aléatoires, X étant le nombre de dés

Observé:Fonctionnel

Démo3: Déplacement du pion GUI

Dans la fenêtre du client lorsque je clique sur un bouton, un pion se déplace d'une valeur aléatoire

Observé:Fonctionnel

Démo4: Déplacement du pion Serveur

Le serveur doit déplacer le pion du joueur conformément au lancer de dés

Observé: Non fonctionnel

L'interface est fonctionnelle : Le pion se déplace de manière aléatoire entre 2 et 12

Mais pas encore de communication au serveur

10.3.4.5 No 31- Gestion case carte chance

en tant que Joueur

je veux pouvoir tirer une carte chance quand je tombe sur la case associée

dans le but de bénéficier de ses effets

Démo1: Réception d'une carte aléatoire

la carte doit venir du sommet du deck.

☐ une fois l'effet "consommé", la carte doit retourner au fond du deck

Observé:

Démo2: Envoi d'une carte

☐ Le serveur doit envoyer les informations de la carte tirée au client

Observé:

10.3.4.6 No 13- Gestion de la case

Le pion arrive à une case.

Gérer les actions possibles liées à la case.

- Acheter terrain / maison

- Tirer carte chance

- Payer taxe

Démo1: Les cases sont correctement récupérées de la DB

En tant que Serveur

Lorsque je souhaite récupérer les cases du jeu

☐ Je reçois toutes les cases avec leurs informations

Observé:

10.3.4.7 No 6- Rejoindre un salon de jeu

en tant que Joueur

je veux rejoindre un salon de jeu

pour jouer avec des personnes

Démo1: Vérifier la commande client

Vérifier que le client envoie la bonne commande

☑ - Pour rejoindre une partie

☑ - Pour se déclarer prêt

Observé: Fonctionnel

Démo2: Vérifier l'interprétation du serveur

☑ Vérifier que lorsque le serveur reçoit une commande d'un client pour rejoindre une partie, celui-ci vérifie bien que la partie soit joignable et qu'il ajoute bien l'utilisateur à la partie.

Observé:Fonctionnel

Démo3: Vérifier la GUI

☑ -Vérifier que, lorsqu'un utilisateur rejoint un salon de jeu, ce dernier s'affiche bien dans l'interface pour le bon salon.

☑ - Vérifier que les utilisateur du salon sont bien notifié et que l'affichage se fait correctement

Observé:Fonctionnel

10.3.4.8 Bilan général itération no 4

- Histoire 12: Réalisée mais pas encore de communication avec le serveur
- Histoire 3: Terminée
- Histoire 6: Terminée
- Histoire 9: Non terminée
- Histoire 8: Non terminée
- Histoire 31: Non terminée
- Histoire 13: Non terminée

Au final, beaucoup d'avance mais blocage au niveau de la création de partie, entrainant la non terminaison de plusieurs histoires

Environ 70% de code réalisé.

Replanification souhaitée: Reporter sur le sprint suivant

10.3.5 Itération 5

10.3.5.1 Histoires (15 tâches done, 2 in progress, 4 to do)

9 Préparer la partie
La création de la partie se fait côté serveur/client sans soucis. Quand tout le monde est prêt, la partie commence, les cases sont générées. Il manque la répercussion sur la GUI. Fait à 80%. La fin sera faite sprint suivant.
8 Lancement de la partie
Done. Les personnes quittent le lobby et se retrouvent dans la partie lancée. Le premier joueur à commencer est notifié.
12 Déplacement du pion
Le serveur reçoit bien le lancement du dé et transmets le résultat (état de la case et nombre de cases à se déplacer). Le client GUI est alors mis à jour. Il peut ensuite prévenir le serveur qu'il a fini son tour.
31 Gestion case carte chance
Les cartes sont toutes gérées côté serveur. Le client reçoit bien la carte tirée mais n'applique pas le résultat. Fait à 80%. La fin sera faite sprint suivant.
13 Gestion de la case
Gestion de la case : joueur présent et informations sur la case (propriétaire, etc.) géré par le serveur mais pas encore visible sur GUI. Fait à 80%. La fin sera faite sprint suivant.
30 Gestion case terrain
Pas fait. Reporté au sprint suivant
32 Gestion case taxe
Géré au niveau du serveur pas GUI du client. Fait à 80%. La fin sera faite sprint suivant.
33 Gestion case entreprise
Géré au niveau du serveur pas GUI du client. Fait à 80%. La fin sera faite sprint suivant.
34 Gestion cases "coins"
Done. Start et pause sont gérés. La prison est prévue au sprint d'après.

10.3.5.2 Bilan général

Beaucoup de fonctions implémentées au niveau du serveur et retransmis aux clients mais pas encore visible sous forme de GUI. La majorité des histoires seront terminées au sprint suivant car il manque que 20%. Le sprint suivant peut accepter cette charge supplémentaire.

10.3.6 Itération 6

10.3.6.1 No 13- Gestion de la case

Le pion arrive à une case.

Gérer les actions possibles liées à la case.

- Acheter terrain / maison

- Tirer carte chance

- Payer taxe

10.3.6.2 No 30- Gestion case terrain

En arrivant sur une case terrain,

Je peux soit l'acheter,

Soit payer la personne qui la possède.

10.3.6.3 No 33- Gestion case entreprise

Lorsque j'arrive sur une case entreprise,

Je veux pouvoir l'acheter si elle est libre,

ou payer la taxe si elle possède un propriétaire.

10.3.6.4 ☑ No 16- Vente d'une propriété

en tant que Joueur

je veux vendre une de mes propriétés

afin de gagner de l'argent

10.3.6.5 No 18- Hypothèque de propriété

en tant que Joueur

je veux hypothéquer une propriété

afin de gagner un peu d'argent sans la perdre

10.3.6.6 No 21- Détection banqueroute

Lorsque le joueur ne possède plus assez d'argent, ni de possession pour redresser la barre, il fait banqueroute. Le joueur a perdu la partie et ne peut plus jouer.

- Si possibilité de mettre terrains en hypothèque et continuer, proposer au joueur de mettre en hypothèque ou de déclarer forfait

10.3.6.7 No 35- GUI - Logs et gestion des bouton

Gestion de log. Lorsqu'un joueur lance les dés, cela s'affichage dans les logs du plateau de jeu. Idem lorsque le joueur tire une carte, passe par la case départ ou passe par la case qui amène en prison

10.3.6.8 ☑ No 31- Gestion case carte chance

en tant que Joueur

je veux pouvoir tirer une carte chance quand je tombe sur la case associée dans le but de bénéficier de ses effets

10.3.6.9 No 9- Préparer la partie

Le serveur va préparer la partie cela inclut :

- l'établissement de l'ordre de passage

- génération du deck de carte chance

- répartition de la fortune individuelle

- génération du plateau

10.3.6.10 No 32- Gestion case taxe

en tant que Joueur

je veux pouvoir payer une taxe en tombant sur une case nécessitant un paiement

dans le but de ne pas avoir de problèmes avec la justice

10.3.6.11 Bilan général itération no 6

Complètement terminées:

9 - 26 - 31 - 32

En voie de terminaison

30 - 13 - 18 - 33

Non réalisées

16 - 21

10.3.6.11.1Pour le sprint 7

Les histoires 21 et 16 sont reportées

Certaines histoires du sprint 7 (gestion d'inactivité et les scores : Histoires 29 et 25) risquent d'être lâchées, car non importantes pour le jeu.

Un test unitaire a été mis en place pour la génération des tests unitaires.

Sinon : groupe en confiance

Observé

Jeu en réseau avec 3 joueurs, déjà grandement fonctionnel.

En gros seuls les achats et les ventes ne sont pas encore fonctionnels, sinon besoin de peaufiner un certain nombre de points.