

Génie Logiciel (GEN)

Projet – Sprint 3 - Chibre



Alexis Allemann / Alexandre Mottier

Du 07.05.2020 au 20.05.2020

HEIG – VD / Travail à distance

Classe GEN-C

TABLE DES MATIÈRES

1.	Introduction	1
2.	Equipe de projet	2
3.	Vision	3
3.1.	Présentation du jeu	3
3.2.	Points techniques de l'implémentation	4
4.	Fonctionnalités, limites et pistes d'extension	5
4.1.	Back-end	5
4.1.1.	Premières stories	5
4.2.	Front-end	5
4.2.1.	Premières stories	5
4.3.	Infrastructure	5
4.4.	Schéma des cas d'utilisation	6
5.	Modèle du domaine	7
6.	Interface homme-machine	8
6.1.	Conventions de nommage	8
7.	Environnement de développement et conception de l'application	9
8.	Interaction principale	10
9.	Conception globale des classes	12
10.	Conclusion	14
11.	Annexes	15

1. INTRODUCTION

Dans le cadre du cours de génie logiciel (GEN) de l'HEIG-VD, nous devons effectuer un projet afin de pouvoir mettre en pratique ce qui est vu dans les périodes de théorie du cours.

Ce projet est effectué par un groupe composé de 2 personnes, durant les périodes de laboratoire à raison de 3 par semaine et durant le temps de travail personnel également estimé à 3 périodes par semaines. Le projet se déroule du 1er avril au 11 juin 2020. Celui-ci est séparé en 4 sprints puisque nous devons appliquer une méthode de développement agile avec Scrum.

Le premier sprint se déroule entre le 1er et le 23 avril 2020. Nous avons comme but de lancer le projet et livrer un premier produit selon les stories qui auront été définies et validées par le « product owner » M. Lachaize dans iceScrum.

Le second sprint se déroule entre le 23 avril et le 6 mai 2020. Nous avons comme but de continuer le développement du jeu et de son interface utilisateur selon les stories qui auront été définies et validées par le « product owner » M. Lachaize dans iceScrum.

Le troisième sprint se déroule entre le 7 et le 20 mai 2020. Dans celui-ci, nous allons principalement travailler sur l'implémentation du jeu et notamment les annonces, l'enregistrement des joueurs et la possibilité d'effectuer des parties complètes sur un jeu fonctionnel. Nous allons également travailler sur l'implémentation de DTO (data transfert objects) pour la communication réseau.

Cette documentation présente le projet que nous allons réaliser qui est un jeu du Chibre (également nommé Jass). Ce jeu de cartes est beaucoup joué en Suisse et nous avons trouvé intéressant de le développer pour ce projet car il permet d'aborder tous les points techniques abordés en cours théorique tout en gardant un produit qui soit lucratif à utiliser et à développer pour nous.

Nous allons nous répartir le travail entre les deux membres du groupe afin de pouvoir gagner en productivité. Nous ferons tout de même des séances afin que chacun puisse apporter ces idées et également pour assurer un suivi du travail de chacun.

Avant de commencer ce projet, nous avons hâte de pouvoir mettre en pratique les éléments appris afin de mieux comprendre ce qu'est le génie logiciel notamment en appliquant une méthode agile avec Scrum et en utilisant des outils tels que Git, Travis, iceScrum.

2. EQUIPE DE PROJET

ICESCRUM :

Le nom du projet iceScrum est **GEN_Allemann-Mottier** et sa clé unique est **GENAALAMO**. Voici le lien pour accéder à la gestion du projet : <https://cloud.icescrum.com/p/GENAALAMO>

GITHUB :

Le repository est nommé **GEN-Projet** et est disponible à l'adresse suivante : <https://github.com/alexis-allemann/GEN-Projet>

SCRUM MASTER :

Alexis Allemann : alexis-allemann, alexis.allemann@gmail.com, AAL

Alexandre Mottier : amottier, alexandre.mottier@heig-vd.ch, AMO

PRODUCT OWNER :

Patrick Lachaize : pogenc, pogenc@gmail.com, PLE

TEAM MEMBERS :

Alexis Allemann : alexis-allemann, alexis.allemann@gmail.com, AAL

Alexandre Mottier : amottier, alexandre.mottier@heig-vd.ch, AMO

3. VISION

Il s'agit d'implémenter un jeu du Chibre (Jass).

3.1. PRÉSENTATION DU JEU

GÉNÉRALITÉS

Ce jeu se joue à 4 joueurs, en 2 équipes de 2, avec un jeu de 36 cartes, allant de l'as au 6.

L'ORDRE DES CARTES

Dans la couleur d'atout : Valet - 9 - As - Roi - Dame - 10 - 8 - 7 - 6.

Le valet d'atout se nomme le Bauer, le 9 d'atout se nomme le Nell.

Dans toutes les autres couleurs c'est l'ordre habituel, soit : As - Roi - Dame - Valet - 10 - 9 - 8 - 7 - 6.

POINTS PAR CARTE

Carte	Atout	Normal
As	11	11
Roi	4	4
Dame	3	3
Valet	20	2
10	10	10
9	14	0
8-7-6	0	0

DISTRIBUTION

Le système mélange les cartes et les distribues aux joueurs (9 cartes par joueur).

DÉTERMINATION DE L'ATOUT

Au premier tour, le joueur ayant le 7 de carreau fait atout, et devient le donneur du tour suivant. Aux tours suivants, c'est la personne située après le donneur qui fera atout.

"Faire atout" signifie "choisir la couleur d'atout". Si le joueur ne peut pas ou ne veut pas choisir, il a la possibilité de "chibrer" et son partenaire doit obligatoirement choisir une couleur d'atout à sa place.

C'est le joueur à qui devait faire atout qui entame (même si c'est son partenaire qui a choisi l'atout).

ANNONCES

Le système va rechercher si un joueur a une annonce dans son jeu, si oui il l'affiche lors du tour du joueur.

Les annonces possibles sont présentées ci-dessous :

ANNONCES POSSIBLES

Nom de l'annonce	Points	Description
Schtöckr	20	Le joueur possède le roi et la dame d'atout. Annoncé lorsque l'on pose la deuxième carte et peut être combiné avec une autre annonce).
3 cartes	20	Suite de 3 cartes dans la même couleur
4 cartes	50	Suite de 4 cartes dans la même couleur
Carré de 4 cartes	100	Quatre cartes de couleurs différentes mais de valeur identique
4 cartes	100	Suite de plus de 4 cartes dans la même couleur
Carré de neufs	150	Ce sont les 4 neufs
Carré de valets	200	Ce sont les 4 valets

Si les 2 équipes ont des annonces lors du premier tour, seuls les points des joueurs de l'équipe ayant la meilleure annonce sont enregistrés. En cas d'égalité entre les annonces, c'est l'annonce avec le plus de cartes qui prime, suivie de celle avec la plus haute carte (le bourg et le nell d'atout comptant comme une autre couleur, et correspond donc à un valet ou un neuf normaux), suivie de 2 celle en atout. En cas d'égalité, l'équipe du joueur ayant annoncé l'annonce la plus haute en premier note ses annonces.

DÉCOMPTE DES POINTS


A la fin de chaque tour, les points que chaque équipe a remportés sont enregistrés selon la valeur des cartes remportées.

Le dernier pli vaut 5 points supplémentaires.


FIN DE LA PARTIE


La première équipe à atteindre 1000 points remporte la partie.


3.2. POINTS TECHNIQUES DE L'IMPLÉMENTATION

 Les joueurs joueront en réseau en utilisant la programmation TCP. Il y aura un serveur et des utilisateurs.

↕ Les threads seront utilisés afin de gérer les parties, les actions des joueurs ainsi que la communication au travers du réseau TCP.

 Git sera utilisé afin de pouvoir gérer le code source et le développement d'une manière plus générale. Un repository GitHub en ligne est utilisé comme "remote repository".

 Nous utiliserons Travis relié à GitHub afin de faire le développement des tests unitaires pour suivre une méthodologie de développement TDD (Test Driven Development).

 Pour la gestion du projet, nous travaillons en mode agile selon Scrum à l'aide de IceScrum.

4. FONCTIONNALITÉS, LIMITES ET PISTES D'EXTENSION

Les fonctionnalités seront décrites sous la forme de cas d'utilisation ou de stories. Toutes les fonctionnalités ne sont pas forcément identifiées dans ce document car elles le seront en cours de projet. Néanmoins ce document présente les fonctionnalités qui seront livrées à la fin du premier sprint et quelques autres qui seront traitées dans les sprints suivants.

4.1. BACK-END

La conception de l'application devra être de type client-serveur mais dans un premier temps, l'application sera exécutée sur un seul PC dans un seul processus d'exécution. Chaque intervenant disposera de sa propre fenêtre graphique au sein du même environnement d'exécution et s'exécutera dans son propre thread. Le serveur disposera également de son propre thread.

Dans un deuxième temps, il sera envisagé de convertir l'application avec une technologie RPC (Remote Procedure Call) pour que chaque intervenant puisse utiliser son propre PC.

4.1.1. PREMIÈRES STORIES

- Mise en place client-serveur
- Mise en place MVC (Modèle-Vue-Contrôleur)
- Création des cartes du jeu

4.2. FRONT-END

L'interface graphique sera de type client lourd (pas web). Elle sera conçue avec une approche Modèle-Vue-Contrôleur pour permettre la réalisation de tests automatisés (de type JUnit) sur le modèle.

Dans un premier temps, nous allons réaliser une interface graphique relativement simple en utilisant Java Swing.

Dans un second temps, il sera envisagé d'utiliser une librairie Material-UI pour Swing afin de proposer une interface à l'utilisateur plus « User Friendly ». Nous avons trouvé une librairie sur GitHub que nous testerons avant de l'implémenter dans notre projet. Celle-ci est disponible à cette adresse : <https://github.com/atarw/material-ui-swing>

4.2.1. PREMIÈRES STORIES

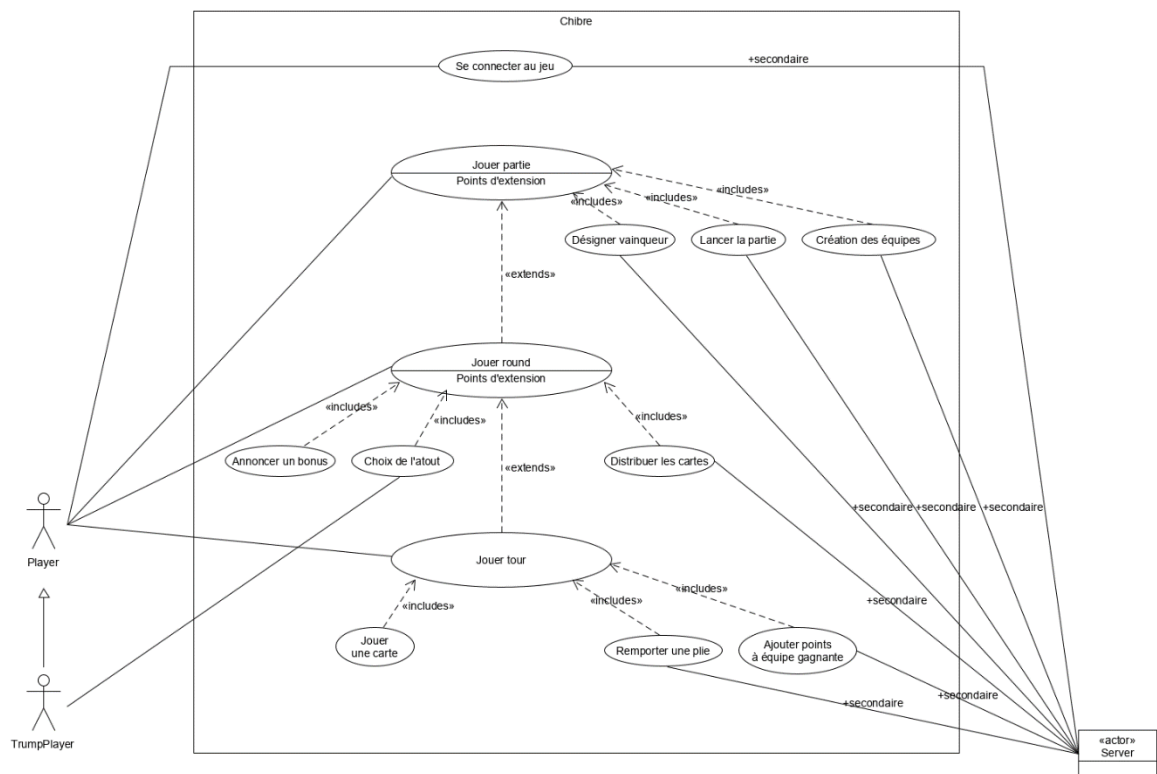
- Affichage des cartes
- Distribution des cartes par joueur

4.3. INFRASTRUCTURE

L'infrastructure du projet utilisera un repository GitHub avec Git pour la gestion du code source. Les tests unitaires seront réalisés avec Travis. Le projet Java utilisera Maven et l'environnement de développement utilisé sera IntelliJ IDEA de JetBrains.

L'ensemble de l'infrastructure présentée ci-dessus sera mise en place durant le premier sprint.

4.4. SCHÉMA DES CAS D'UTILISATION

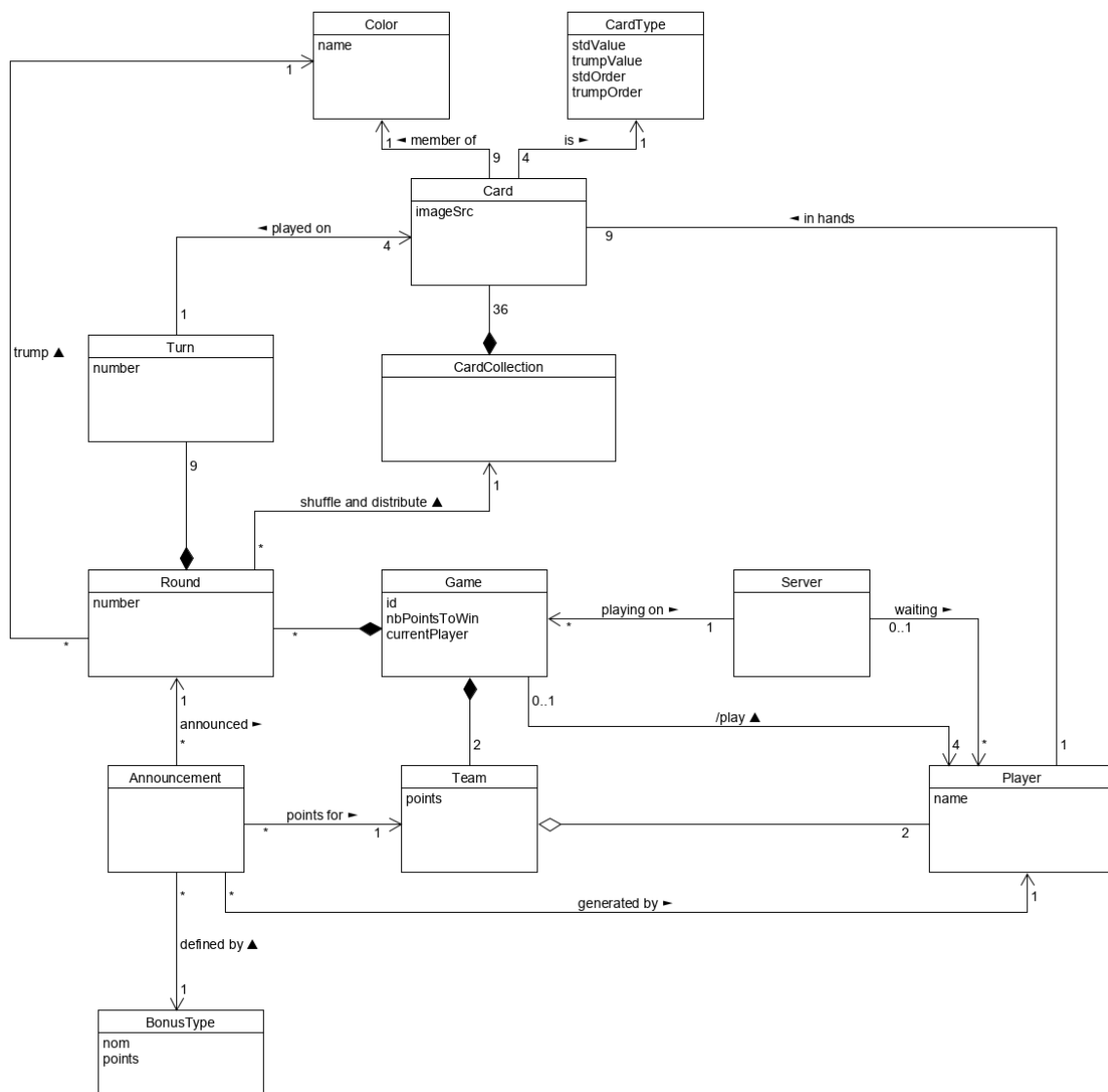


Note : l'image se trouve également en annexe du rapport (pour une meilleure lisibilité).

Dans un premier temps les joueurs sont connectés au serveur lorsqu'ils lancent l'application et ils sont mis dans une file d'attente par le serveur. Celui-ci lance une partie de cartes lorsque 4 joueurs sont en attente. Si un nouveau joueur se connecte à nouveau celui-ci rejoint la file d'attente puis démarrera lorsque le nombre de participants requis sera atteint.

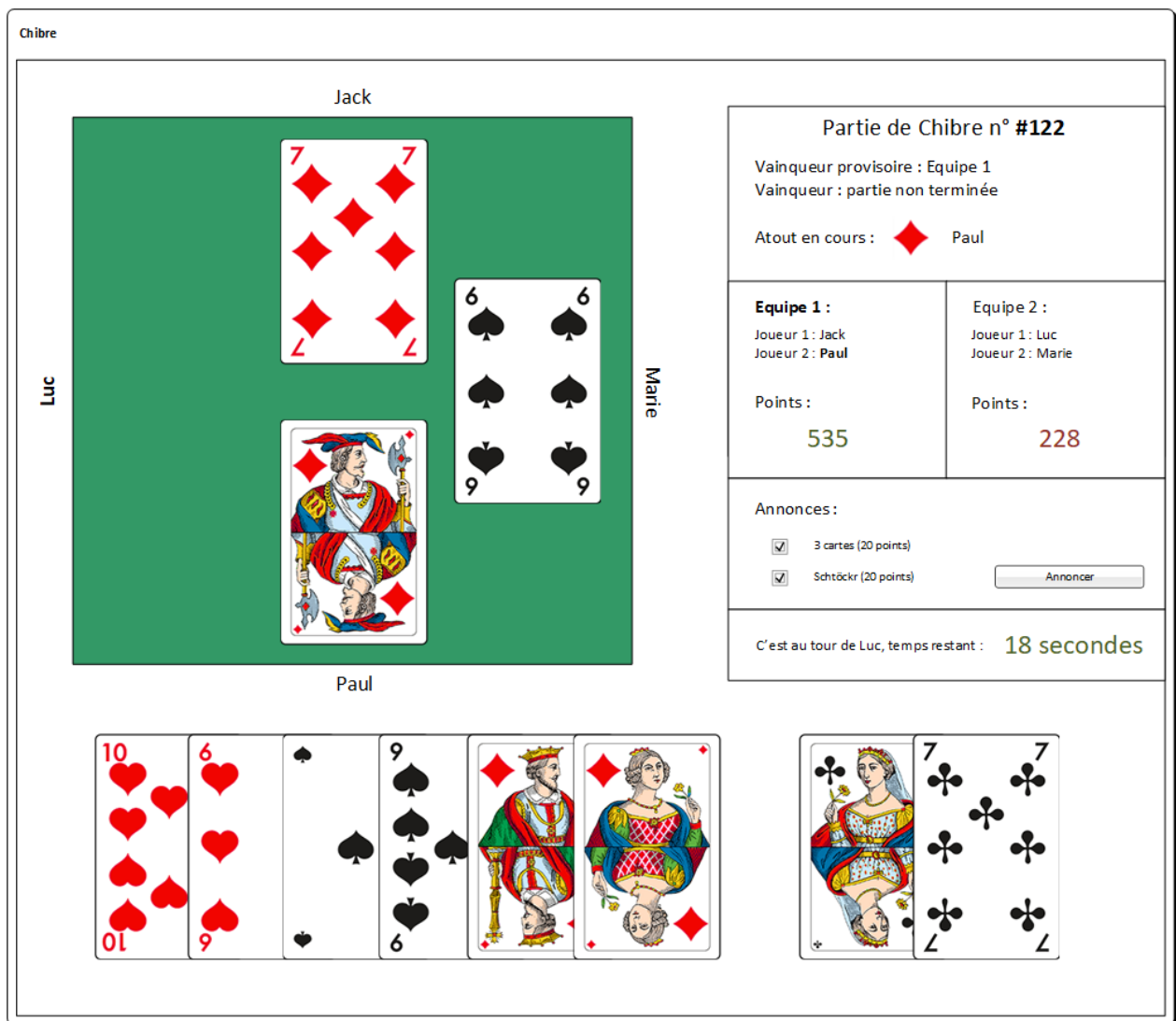
Dans un second temps, nous souhaitons permettre aux utilisateurs de générer un id de partie qu'ils peuvent diffuser à d'autres utilisateurs. Ainsi, les autres joueurs pourraient rejoindre la partie avec le code et celle-ci serait lancée lorsque 4 joueurs sont connectés.

5. MODÈLE DU DOMAINE



Note : l'image se trouve également en annexe du rapport (pour une meilleure lisibilité).

6. INTERFACE HOMME-MACHINE



Note : l'image se trouve également en annexe du rapport (pour une meilleure lisibilité).

Lors du premier tour le bloc « Annonces » contient les annonces possibles faites par le joueur. Ensuite, on affiche les annonces réalisées par les autres joueurs à titre indicatif dans ce bloc.

Ici, le joueur qui a cette interface est « Paul » car le tapis de jeu est orienté contre lui.

Pour jouer une carte, un glisser déposer sera effectué.

6.1. CONVENTIONS DE NOMMAGE

Composant	Préfixe	Exemple
Bouton	btn	btnName
Label	lbl	labelName
Panel	pnl	pnlName
Checkbox	cbx	cbx3Cards

7. ENVIRONNEMENT DE DÉVELOPPEMENT ET CONCEPTION DE L'APPLICATION

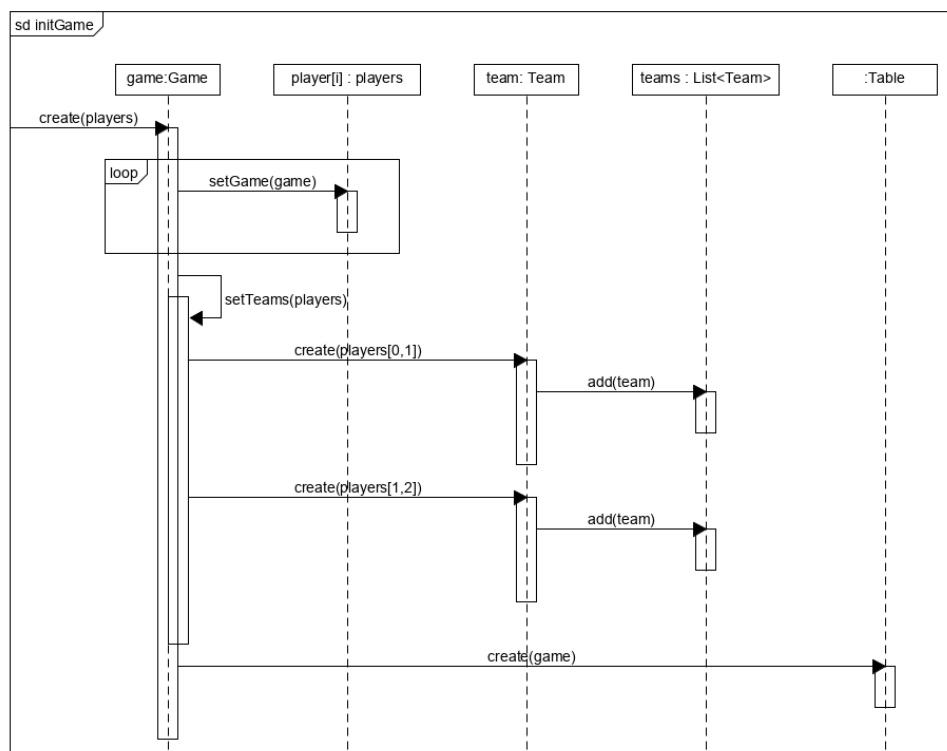
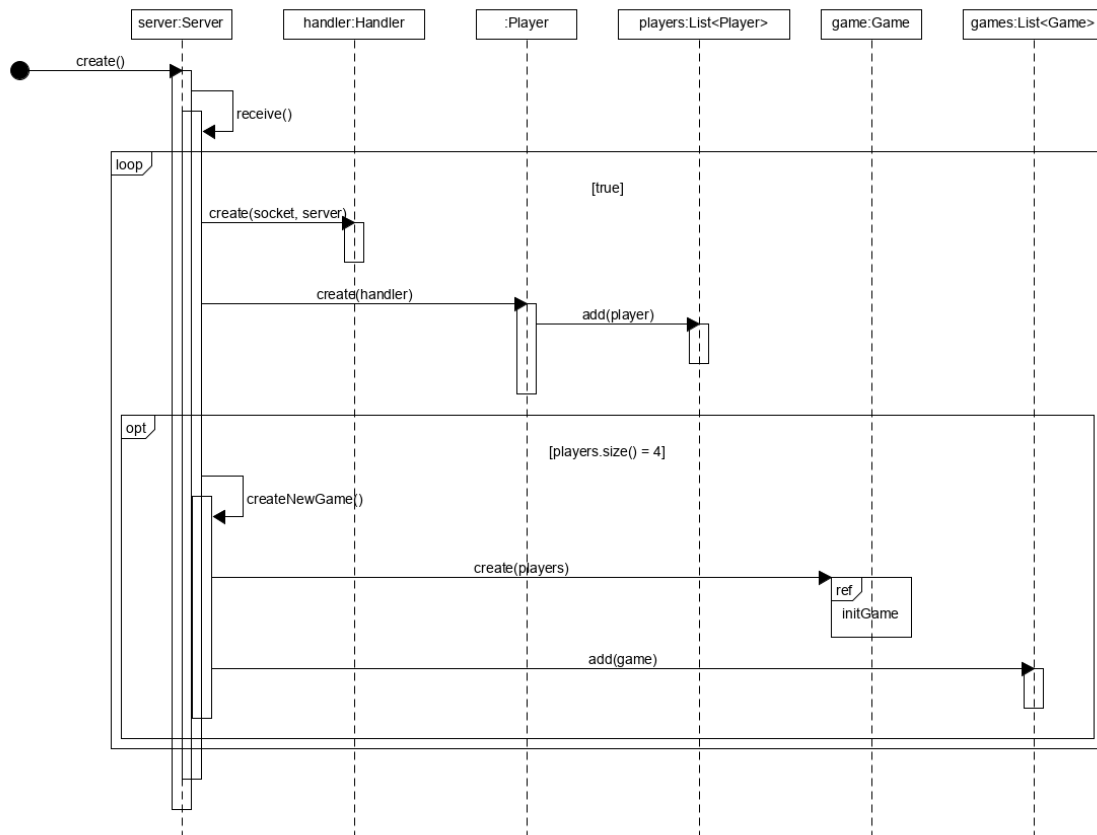
L'application sera développée en Java avec la bibliothèque graphique Swing, le Framework de test JUnit et les threads de java.lang. On la structurera selon le pattern Modèle-Vue-Contrôleur. L'environnement de développement utilisé sera IntelliJ IDEA de JetBrains. Concernant la gestion du code source, GitHub sera utilisé comme « remote repository ». Git sera utilisé en local sur les PC. L'architecture client-serveur utilisera la communication au travers du réseau TCP.

Ce choix est justifié par la volonté de mettre en pratique les connaissances acquises dans les cours logiciel du semestre précédent et de celui en cours. Il ne nécessite pas l'acquisition de technologies supplémentaires, l'objectif du projet devant être l'apprentissage du génie logiciel.

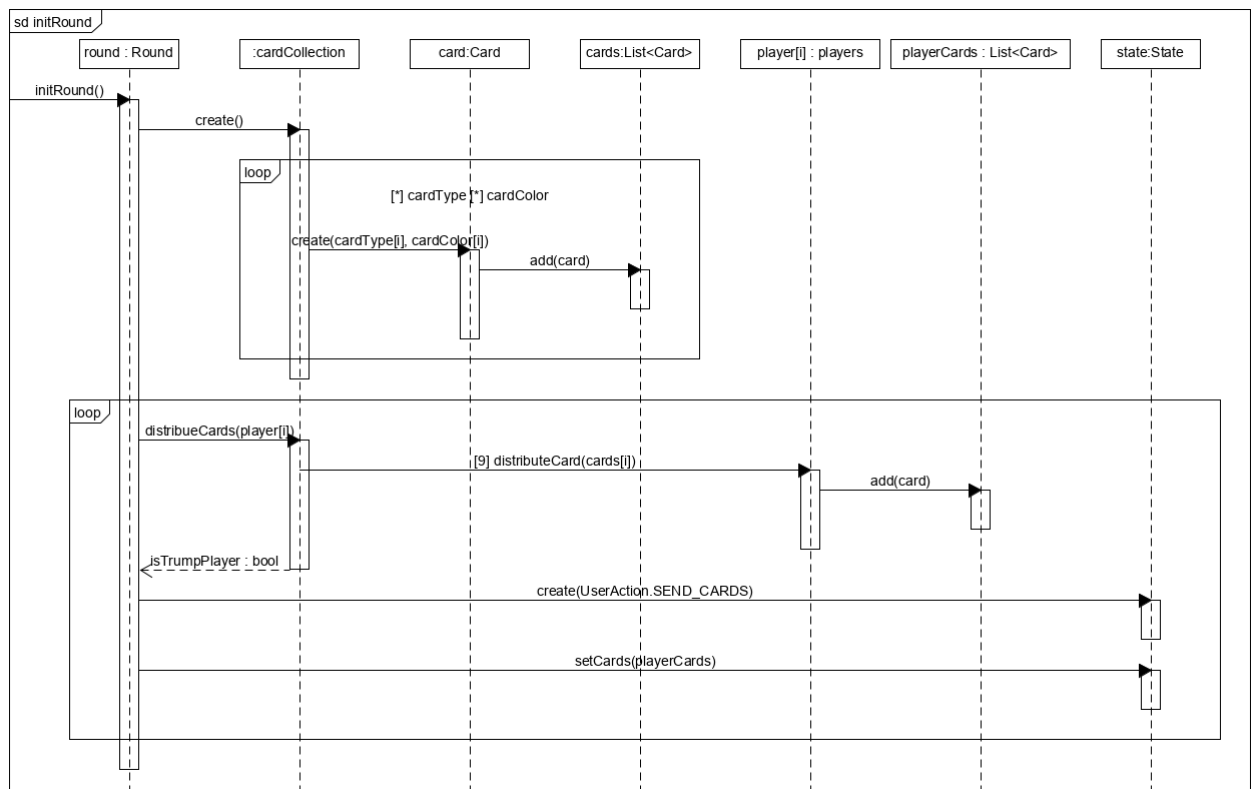
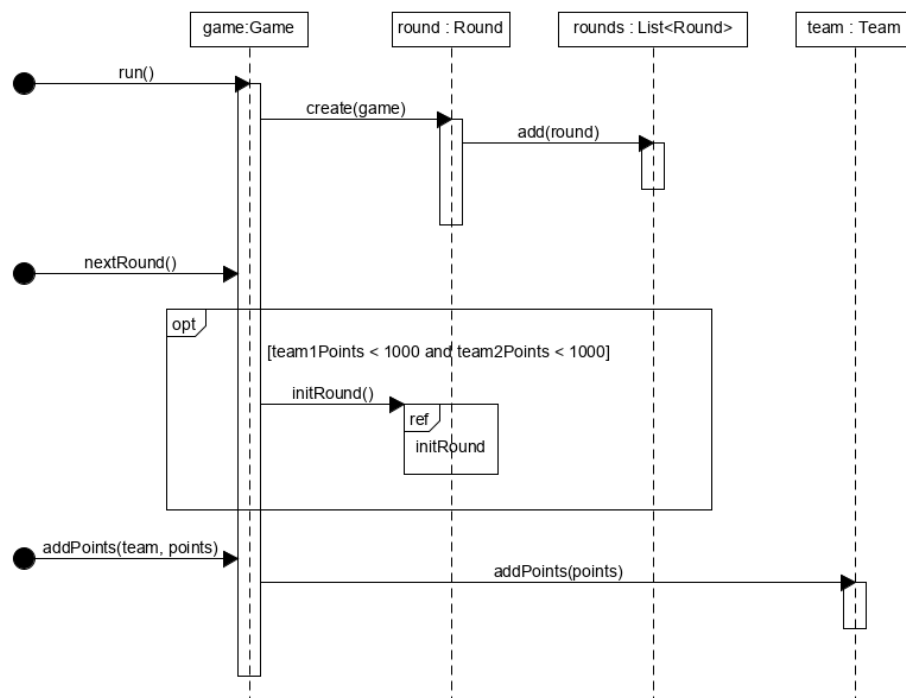
Dans un second temps, nous verrons s'il est possible d'intégrer un libraire Material-UI à Swing comme présenté dans les fonctionnalités du projet au-dessus. Nous verrons également s'il est possible de convertir l'application avec une technologie RPC (Remote Procedure Call) pour que chaque intervenant puisse utiliser son propre PC.

8. INTERACTION PRINCIPALE

Voici l'interaction principale du jeu qui explique comment la partie est créée avec les joueurs et détaille les actions jusqu'à la distribution des cartes aux joueurs :



La suite du flux est déclenchée par la méthode run du thread.



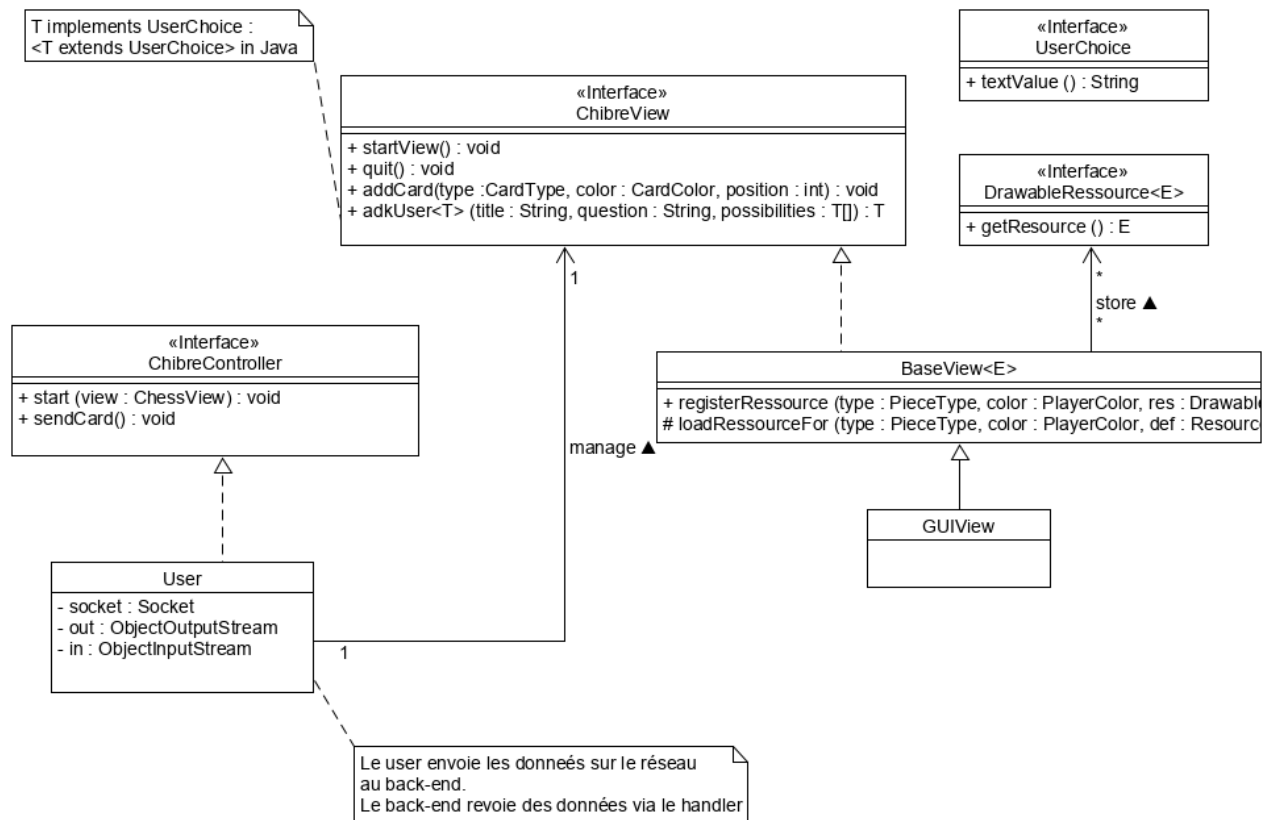
Note : les schémas sont disponibles en images en annexe du rapport pour une meilleure lisibilité.

9. CONCEPTION GLOBALE DES CLASSES

Note : les images des UMLs se trouvent également en annexe du rapport (pour une meilleure lisibilité).

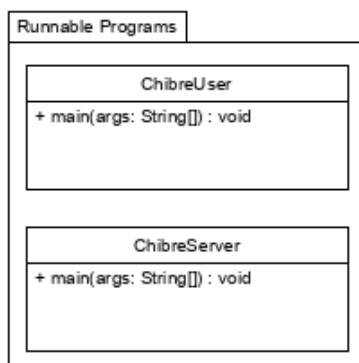
9.1. FRONT-END

Frontend avec MVC

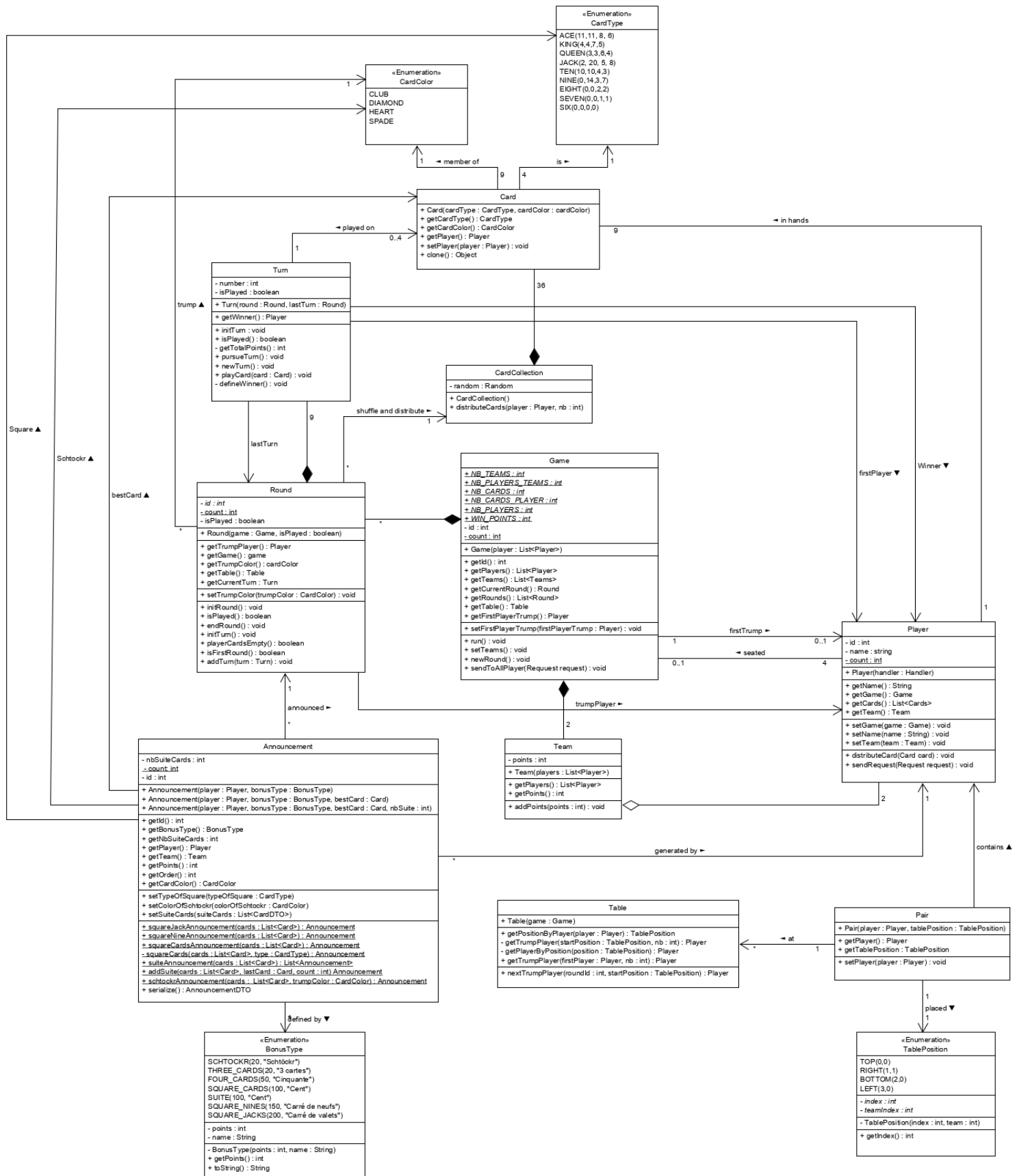


9.2. EXÉCUTABLES

Voici les programmes exécutables qui peuvent être lancés. Le serveur n'est lancé qu'une fois en premier et ensuite les utilisateurs seront automatiquement connectés à celui-ci lors de lancement du programme.



9.3. BACK-END



10. CONCLUSION

Dans le premier sprint, nous avons commencé par réaliser le chapitre « Vision » de ce rapport. C'est dans celle-ci que l'on a présenté le jeu que l'on souhaite réaliser. M. Lachaize à valider notre idée de jeu le jeudi 2 avril.

Ensuite, nous avons réalisés la mise en place de l'architecture en utilisant GitHub, Travis, IntelliJ. Nous avons ensuite réalisé les différents diagrammes demandés dans ce rapport afin de mettre en commun nos idées et notre vision de l'implémentation du jeu du Chibre.

Les diagrammes utilisent le format UML et nous permettent de bien conceptualiser notre projet en adoptant une approche orienté objet.

Ensuite, une fois ceux-ci réalisés nous avons passé à l'implémentation du modèle MVC et de l'architecture client-serveur. Cela nous a permis d'ensuite travailler sur le code de l'application dans un environnement bien défini. Nous avons repris un modèle MVC utilisé dans un laboratoire du cours de programmation orientée objet 1 de l'HEIG-VD.

En parallèle du travail d'implémentation des stories du sprint 1 concernant l'affichage et la distribution de cartes aux joueurs, nous avons rédigé ce rapport.

Lors du second sprint nous avons remis à jours les différents schémas du sprint 1 et nous avons ensuite beaucoup travaillé sur l'architecture backend du jeu. Dans un second temps, nous avons créé tout le frontend et plus précisément la GUI du jeu.

Nous avons ensuite réalisé les schémas d'interactions qui mettent en avant le comportement du jeu et des classes de l'ajout de joueurs au serveur à la distribution des cartes dans une partie.

Dans ce troisième sprint, nous avons premièrement réaliser le travail de login des joueurs et l'enregistrement de ceux-ci de manière permanente dans un fichier JSON dans le projet. Ensuite, nous avons travaillé sur la gestion des annonces autant au niveau backend que frontend. Nous avons passé un certain temps à réaliser cela car les algorithmes de détection des différents cas d'annonces ne sont pas évidents. Nous avons aussi créer une architecture de communication réseau avec des DTO pour permettre de sérialiser nos objets selon notre utilisation. Finalement, nous avons débogger les tours pour que le jeu puisse être fonctionnel et que des parties puissent être jouées correctement.

Tout au long du projet, nous avons utilisé les capacités de iceScrum pour gérer les tâches et la répartition du travail entre les deux membres du groupe.

Nous sommes contents du travail qui a été réalisé durant ce premier sprint. Heureusement que nous avons une semaine de vacance où nous avons pu avancer ce projet car le charge de travail était élevée pour ce premier sprint. Néanmoins, nous avons su travailler en groupe et nous avons pu profiter des avantages que cela apporte.

11. ANNEXES

Diagramme des cas d'utilisations

Diagramme du modèle de domaine

Diagramme de l'interface homme-machine

Diagramme de séquence

- newGame
- initGame
- runGame
- initRound

Diagramme de conception globale des classes

- RunnablePrograms
- Front-End
- Back-End