

Génie Logiciel (GEN)

Projet – Chibre



Alexis Allemann / Alexandre Mottier

Du 01.04.2020 au 11.06.2020

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
4.2.	Front-end	5
4.3.	Infrastructure	5
4.4.	Schéma des cas d'utilisation	6
4.5.	Pistes d'extensions.....	7
5.	Modèle du domaine.....	8
6.	Interface homme-machine	9
6.1.	Conventions de nommage	9
7.	Environnement de développement et conception de l'application	10
7.1.	Authentification	10
7.2.	Réseau	11
7.3.	Annonces.....	12
8.	Interaction principale.....	13
9.	Conception globale des classes.....	15
9.1.	Front-end	15
9.2.	Exécutables	15
9.3.	Back-end.....	16
10.	Gestion de projet avec iceScrum	17
11.	Conclusion	18

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.

Le quatrième et dernier sprint (release) se déroule entre le 21 mai et le 11 juin 2020. Dans celui-ci, nous allons créer les schémas manquants du rapport et adapter ceux déjà présents. De plus nous allons corriger les derniers bugs de notre code et le mettre au propre afin de rendre le projet. Nous allons également consolider nos tests unitaires et le serveur.

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 sont décrites sous la forme de cas d'utilisation ou de stories.

4.1. BACK-END

La conception de l'application est de type client-serveur et est exécutée sur un seul PC dans un seul processus d'exécution. Chaque intervenant dispose de sa propre fenêtre graphique au sein du même environnement d'exécution et s'exécute dans son propre thread. Le serveur dispose également de son propre thread.

4.2. FRONT-END

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

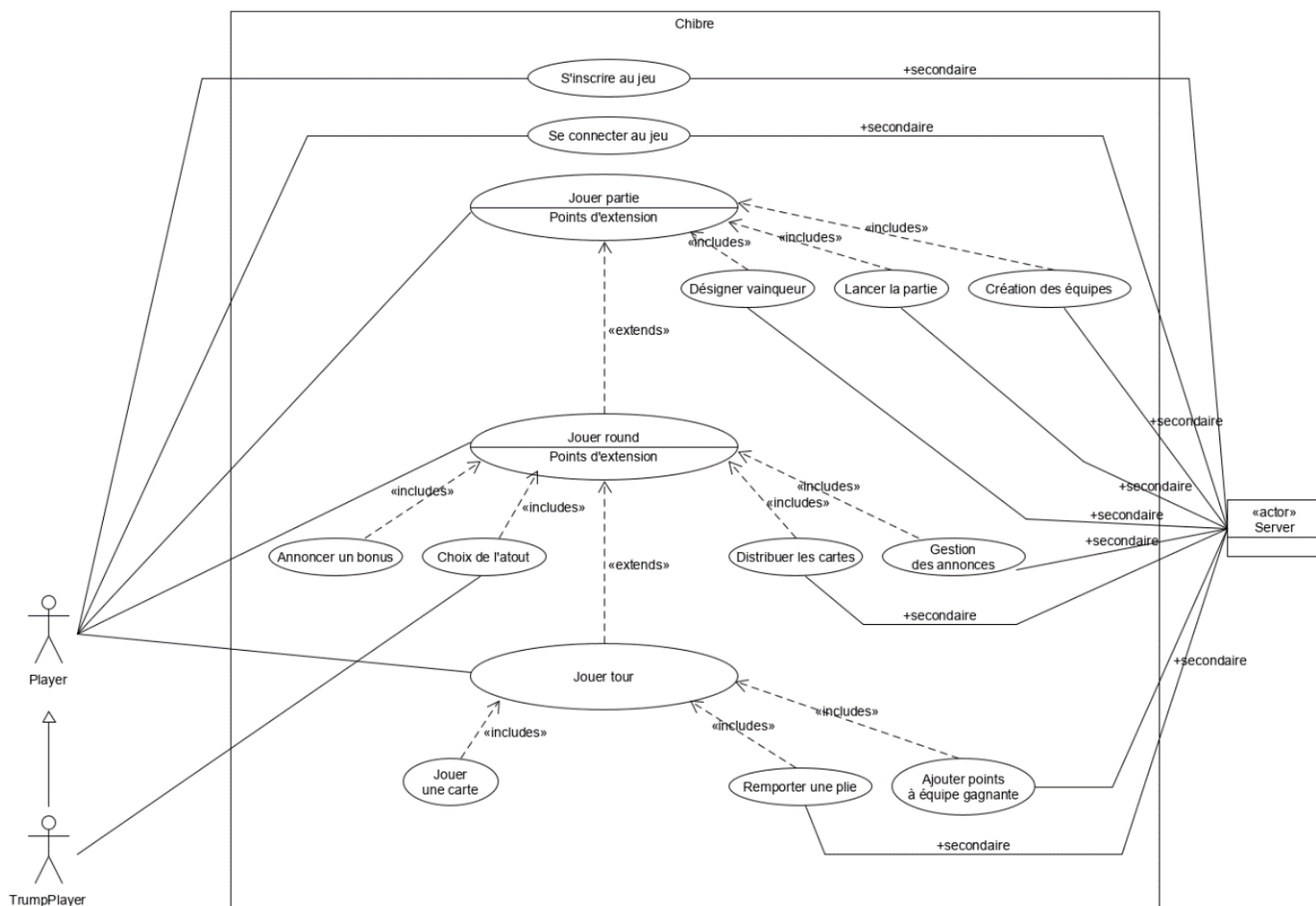
L'interface graphique est relativement simple en utilisant Java Swing.

4.3. INFRASTRUCTURE

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

L'ensemble de l'infrastructure présentée ci-dessus a été 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).

Les joueurs sont connectés au serveur lorsqu'ils lancent l'application et qu'ils se sont authentifiés ou qu'ils se sont inscrits. Ils sont alors mis dans une file d'attente par le serveur. Celui-ci lance une partie 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.

Lors de la création de la partie le serveur tire les équipes aléatoirement puis distribue les cartes aux joueurs. Le joueur ayant le 7 de carreau doit alors choisir l'atout (ou chibrer, son co-équipier choisit l'atout). Dès lors, les tours de jeux commencent.

Les joueurs chacun leurs tours jouent une carte et effectuent leurs annonces. Lorsque chaque joueur a posé une carte, le serveur détecte la meilleure carte posée et attribue les points à l'équipe du joueur gagnant de la pli. Tant qu'une des deux équipes n'atteint pas les 1000 points, la partie continue.

4.5. PISTES D'EXTENSIONS

Amélioration de l'interface graphique : nous pourrions améliorer les interactions, les messages et design des fenêtres utilisateurs en utilisant des bibliothèques graphiques ou même un moteur de jeu.

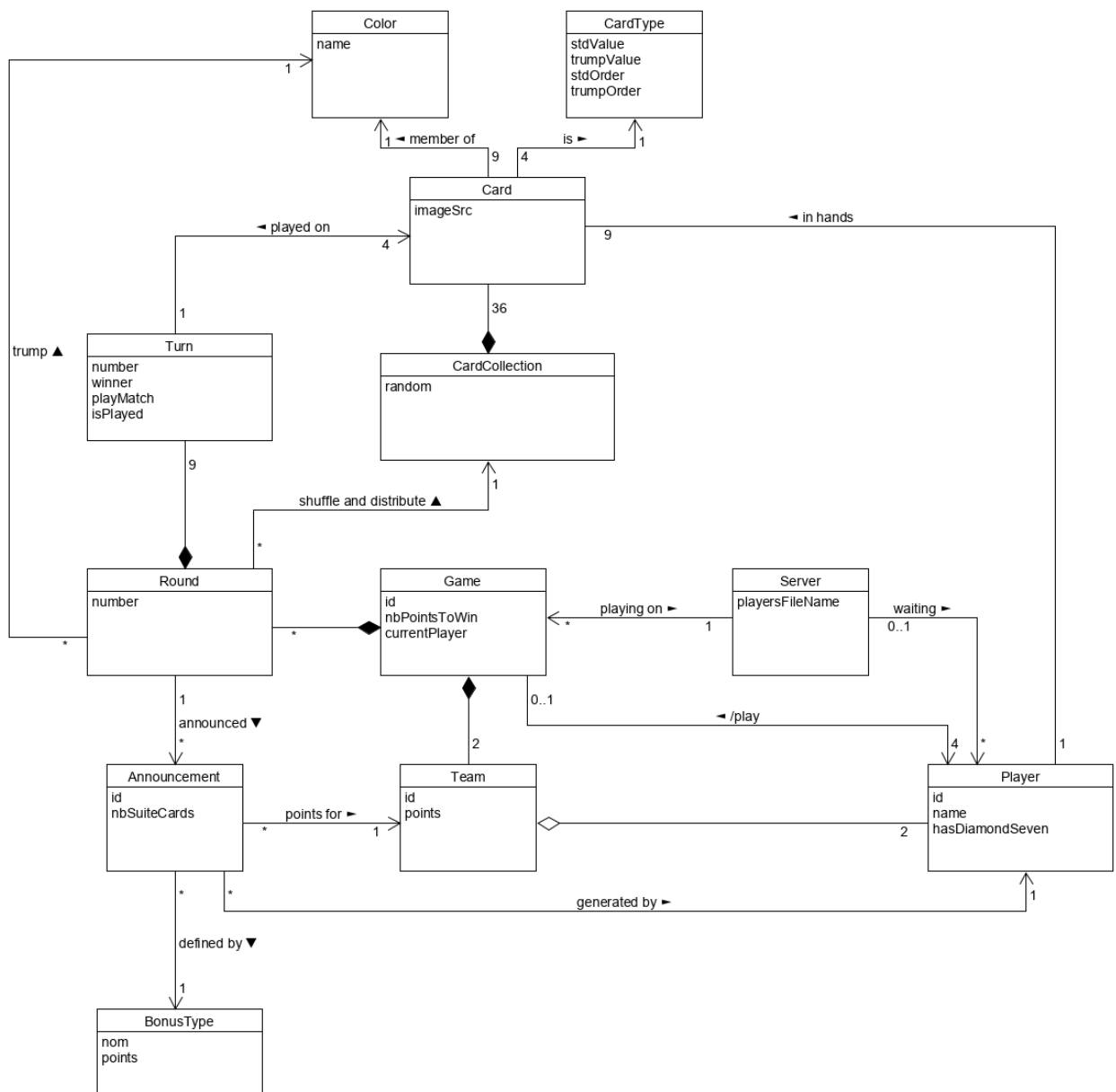
Génération des parties avec ID : il serait possible de personnaliser une partie en choisissant les membres qui la composent à l'aide d'un identifiant généré. Celui-ci permettrait aux utilisateurs de rejoindre la partie en question.

Remote Procedure Call (RPC) : mise en place d'une architecture réseau permettant à chaque joueur de jouer sur son propre ordinateur en se connectant sur un serveur distant.

Personnalisation de partie : pouvoir choisir les règles en vigueur et d'autres paramètres tels que le nombre de joueurs par exemple.

Timer pour réguler les temps de jeu : un timer empêchant les joueurs de prendre trop de temps lorsque c'est leur tour pourrait être mis en place.

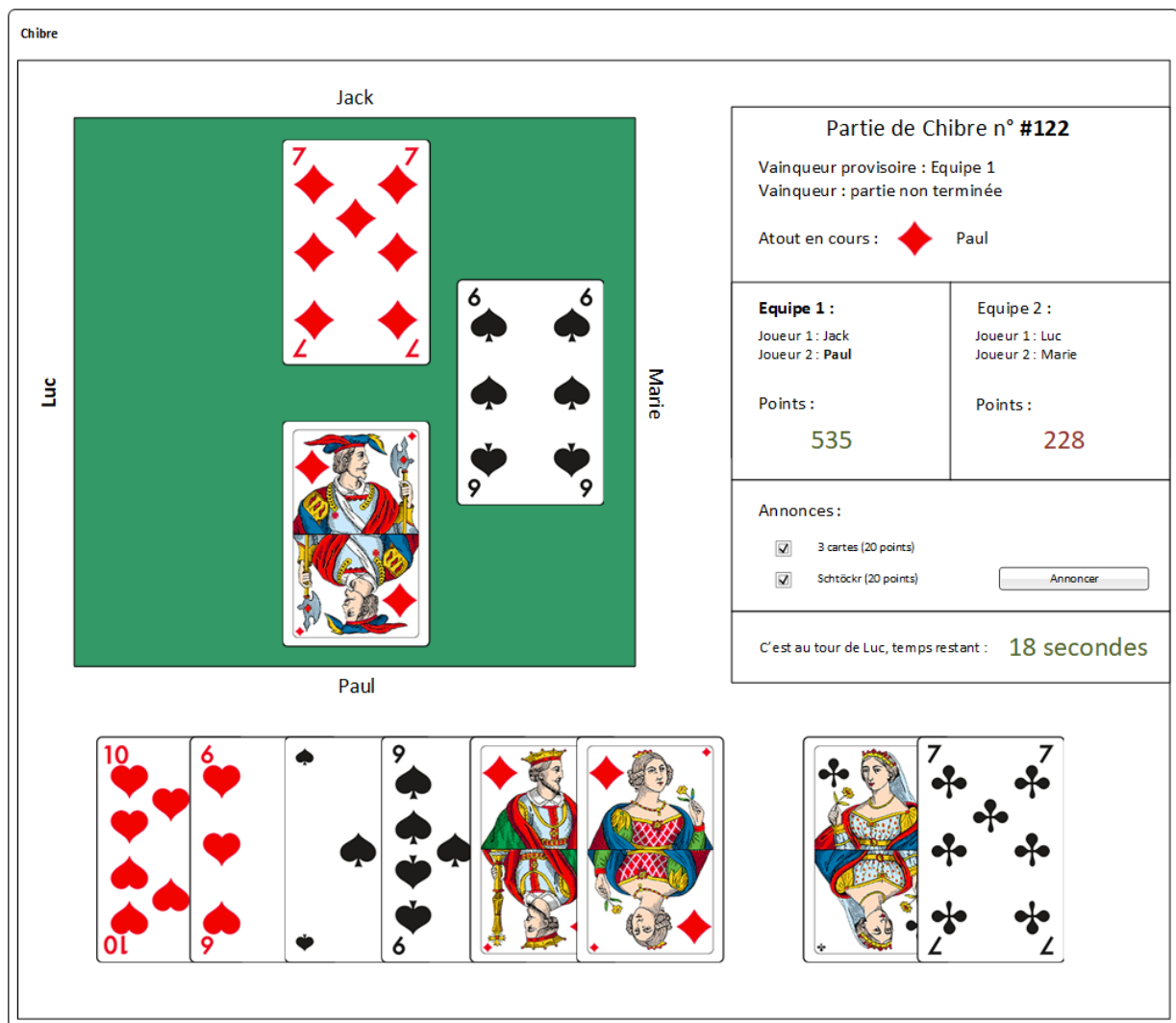
5. MODÈLE DU DOMAINE



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

Le modèle ci-dessus représente la partie backend ou métier de notre jeu. Il est donc directement tiré de la description du jeu au chapitre nommé « Vision » au-dessus.

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
Textbox	tbx	tbxUsername
Password	pwd	pwdPassword

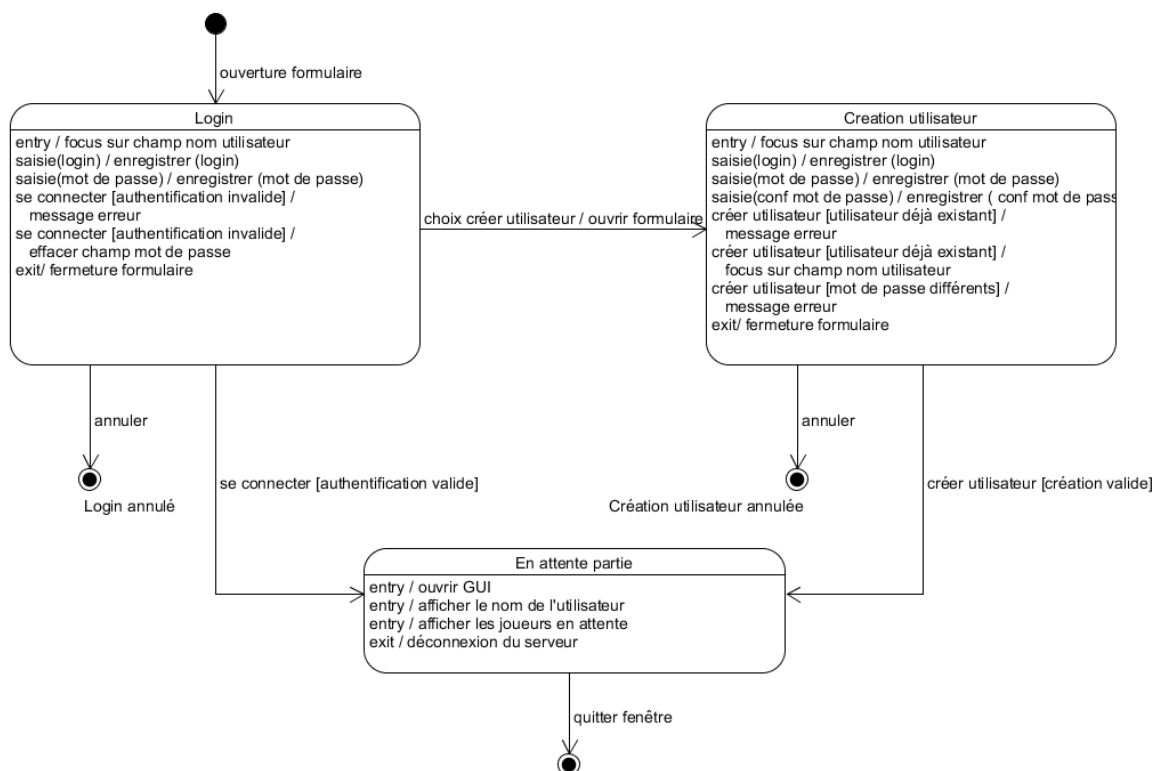
7. ENVIRONNEMENT DE DÉVELOPPEMENT ET CONCEPTION DE L'APPLICATION

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

7.1. AUTHENTIFICATION

Voici un diagramme d'états qui représente les états par lesquels l'utilisateur passe avant de se retrouver sur la GUI en attente qu'une partie commence :



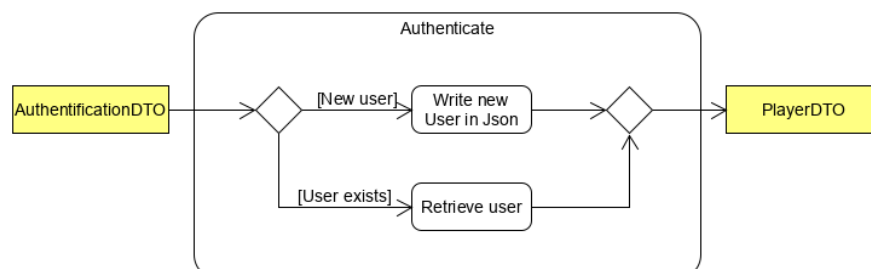
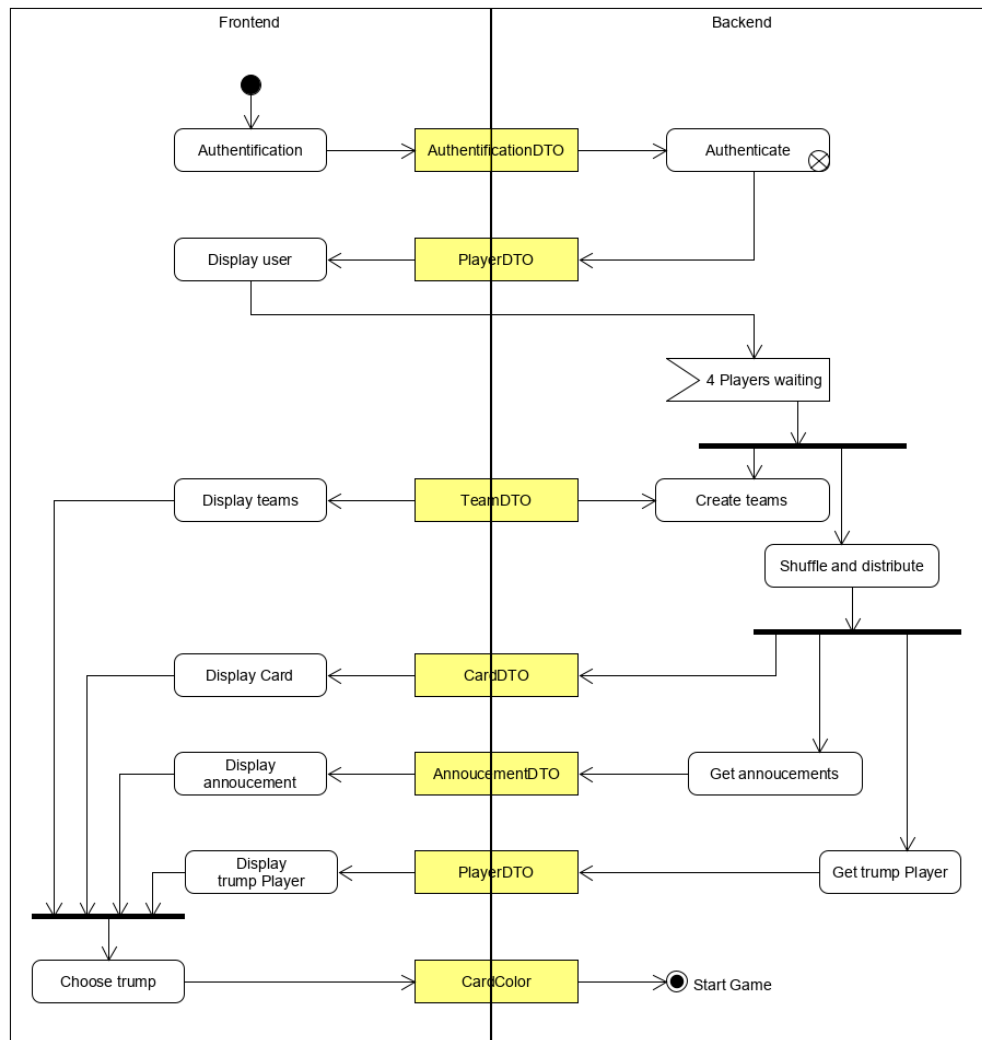
Si un joueur est créé, le serveur l'enregistre dans un fichier JSON pour que lorsque ce dernier redémarre, les joueurs puissent être rechargés. Voici un aperçu du contenu du fichier JSON :

```
[
  {
    "password": "d6a7304069f99c5a5bfe62f1bf9134fb10e109eb5042210045ac422e90b61e6e",
    "username": "Alexis"
  },
  {
    "password": "13e15721c9d4ad58d34983344dfba265a90d80f63db77c2eb3804379d9608889",
    "username": "Alexandre"
  }
]
```

Note : les mots de passes sont hachés en SHA256.

7.2. RÉSEAU

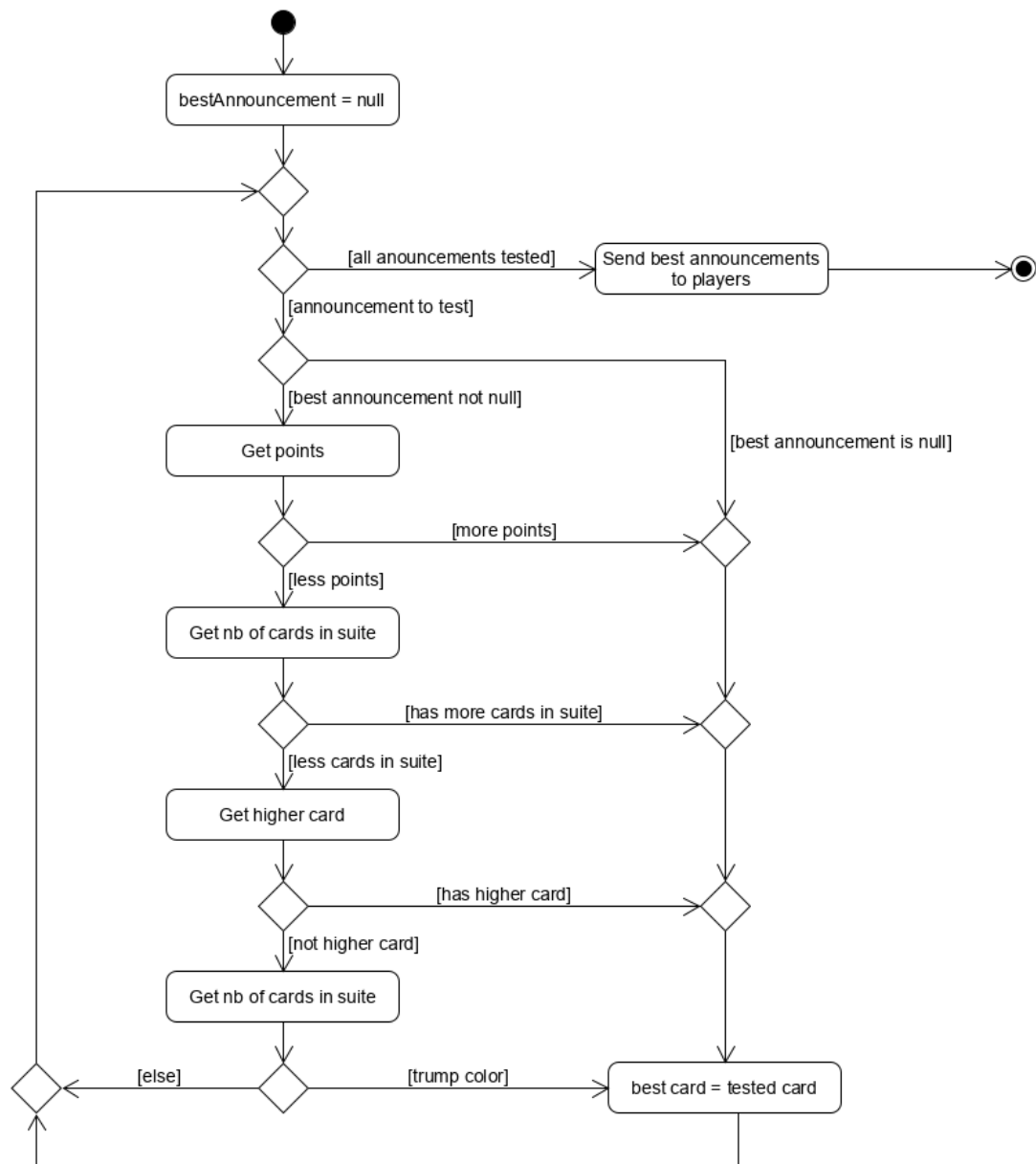
Nous avons choisi d'utiliser des DTO (data transfert objects) afin de pouvoir librement choisir quelles informations communiquent de manière sérialisée entre le serveur et les clients connectés. L'avantage d'utiliser des DTO est de séparer les responsabilités entre les parties et cela permet d'avoir un code plus maintenable et évolutif.



La DTO d'authentification contient les données tels que le nom d'utilisateur et le mot de passe. La DTO des joueurs contient les informations tels que son nom et son identifiant. Il en va de même pour la DTO équipe qui a un nom. Une DTO peut contenir une liste d'autres DTO dans ses attributs. Par exemple, un joueur « playerDTO » contient une liste de cartes (CardDTO).

7.3. ANNONCES

Voici le diagramme d'activité de l'algorithme de détection de la meilleure annonce :



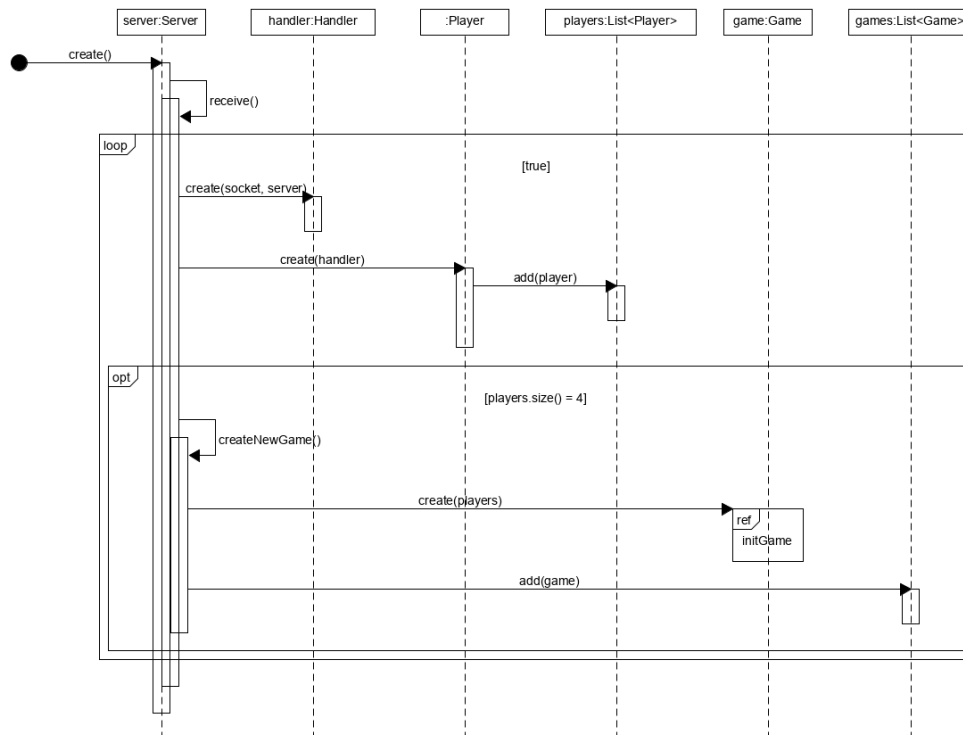
Comme on peut le voir, les critères d'évaluation d'un bonus sont les suivants :

- Valeur du bonus en nombre de points
- Nombre de cartes à la suite
- Plus haute carte
- Est-ce de la couleur atout

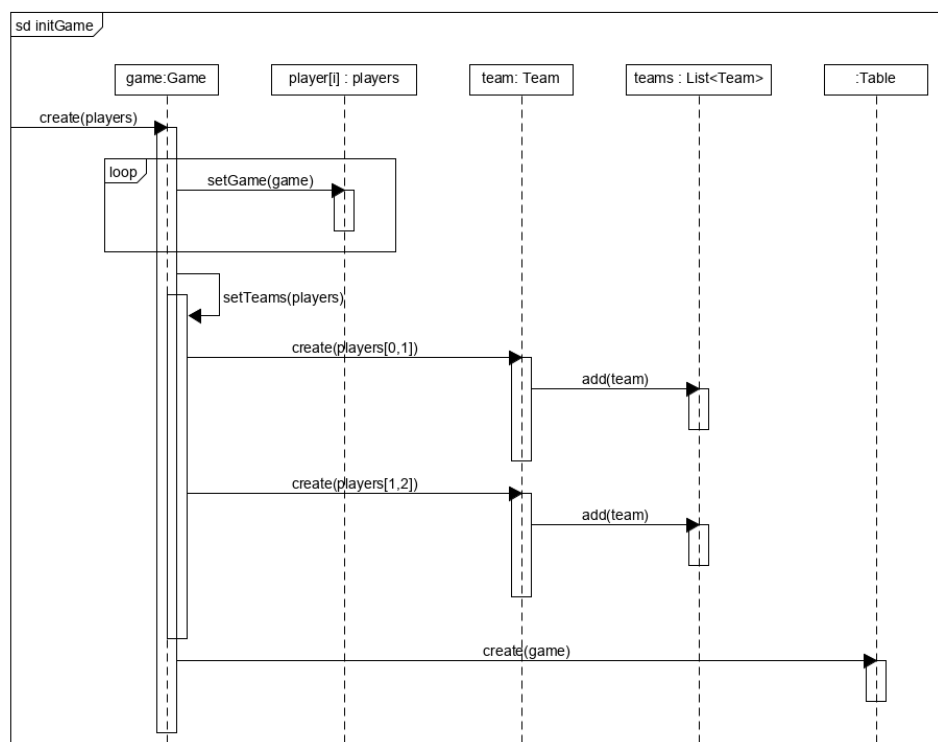
Ces critères d'évaluations sont croissants. On regarde d'abord la valeur en points, puis le nombre de cartes, etc.

8. INTERACTION PRINCIPALE

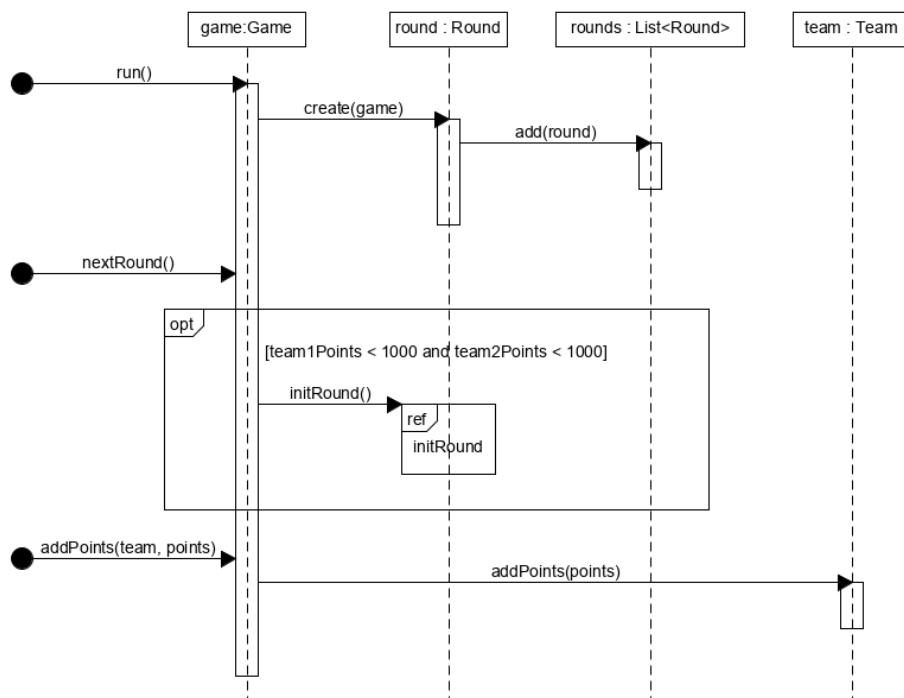
Voici l'interaction principale du jeu qui explique comment la partie est créée avec les joueurs et détaille les actions jusqu'à l'initialisation d'un round :



Ici, lorsque les joueurs sont connectés et authentifiés, ils sont ajoutés à une file d'attente. Lorsque 4 joueurs sont en file d'attente, une partie est créée avec ceux-ci :



La suite du flux est déclenchée par la méthode run du thread lorsque la partie commence :



On observe que l'on joue des rounds tant qu'une équipe n'atteint pas les 1000 points. Chaque tour de jeu, on ajoute les points à l'équipe qui a gagné la plie.

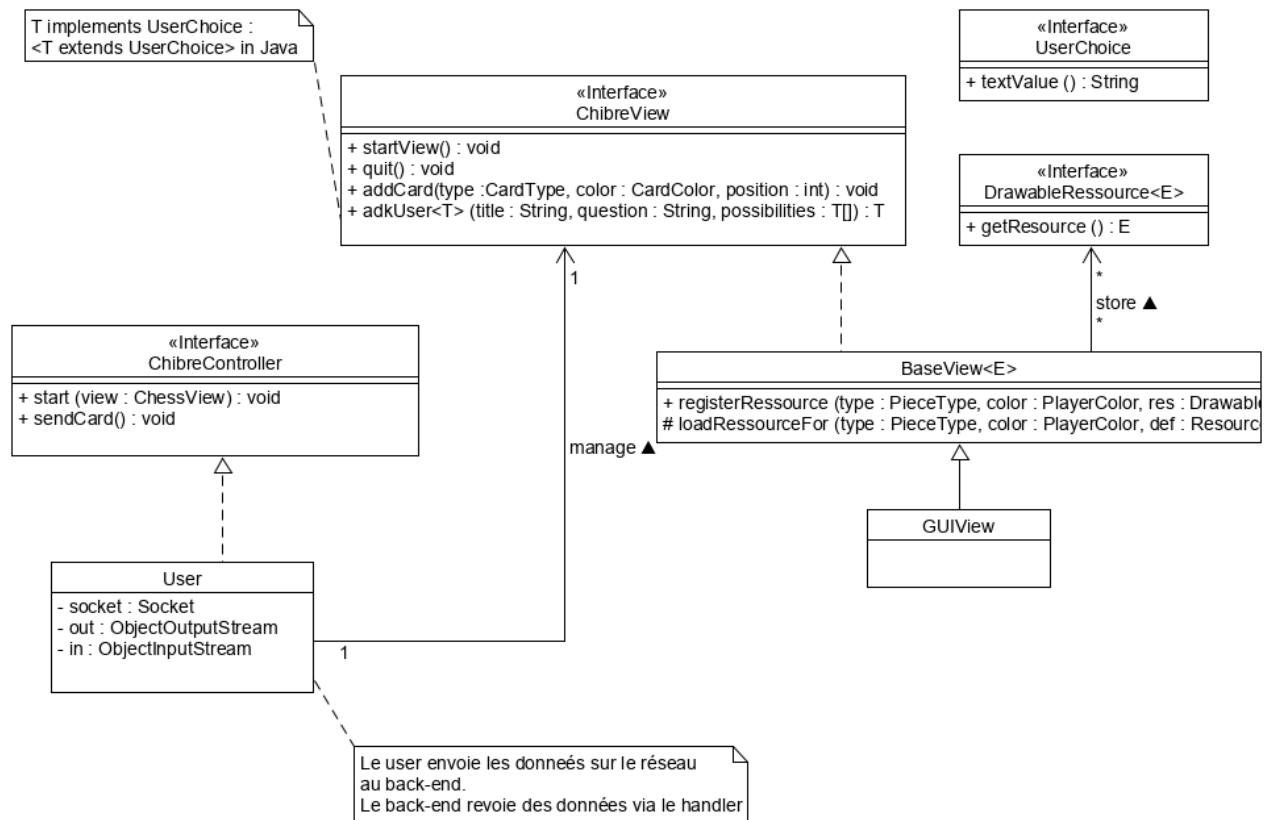
Note : les schémas sont disponibles en images en annexe du rapport pour une meilleure lisibilité.

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

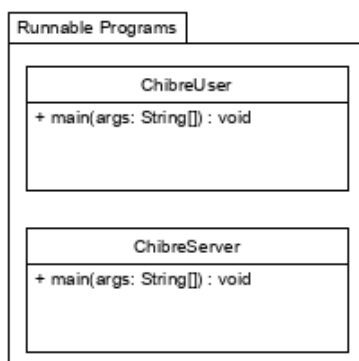
Frontend avec MVC



Les ressources de l'interface « DrawableRessource » permettent de charger les images des cartes.

9.2. EXÉCUTABLES

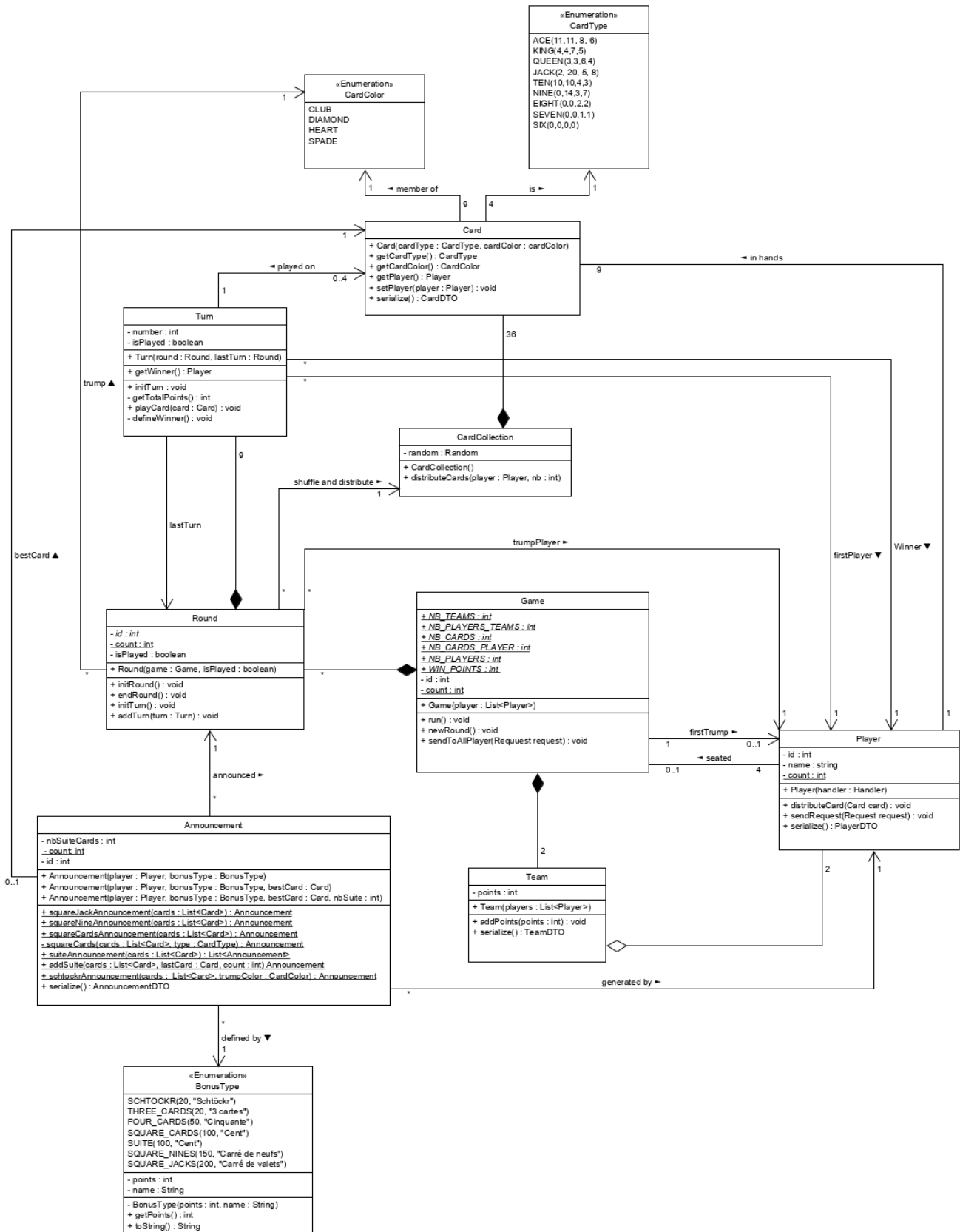
Voici les programmes exécutables qui peuvent être lancés. Le serveur n'est lancé qu'une fois en premier et ensuite les utilisateurs seront automatiquement connectés à celui-ci lors de lancement du programme.



9.3. BACK-END

Voici le schéma UML de classes de notre logique métier backend. Celui-ci est partiel afin de montrer les méthodes principales utilisées :

Note : le schéma se trouve également en annexe du rapport (pour une meilleure lisibilité).



10. GESTION DE PROJET AVEC ICESCRUM

Tout au long du projet, nous avons organisé notre travail et nos tâches avec iceScrum afin de travailler en mode agile. Voici les principales fonctionnalités que nous avons exploitées de ce logiciel de gestion de projet :

Tout d'abord nos idées de fonctionnalités, d'améliorations et de corrections de bugs étaient saisies en tant que stories dans le backlog sandbox. Nous répartissions les tâches à l'aide des tags suivants :

Back-End / Front-End / Bug / Documentation / Infrastructure / Fonctionnalité

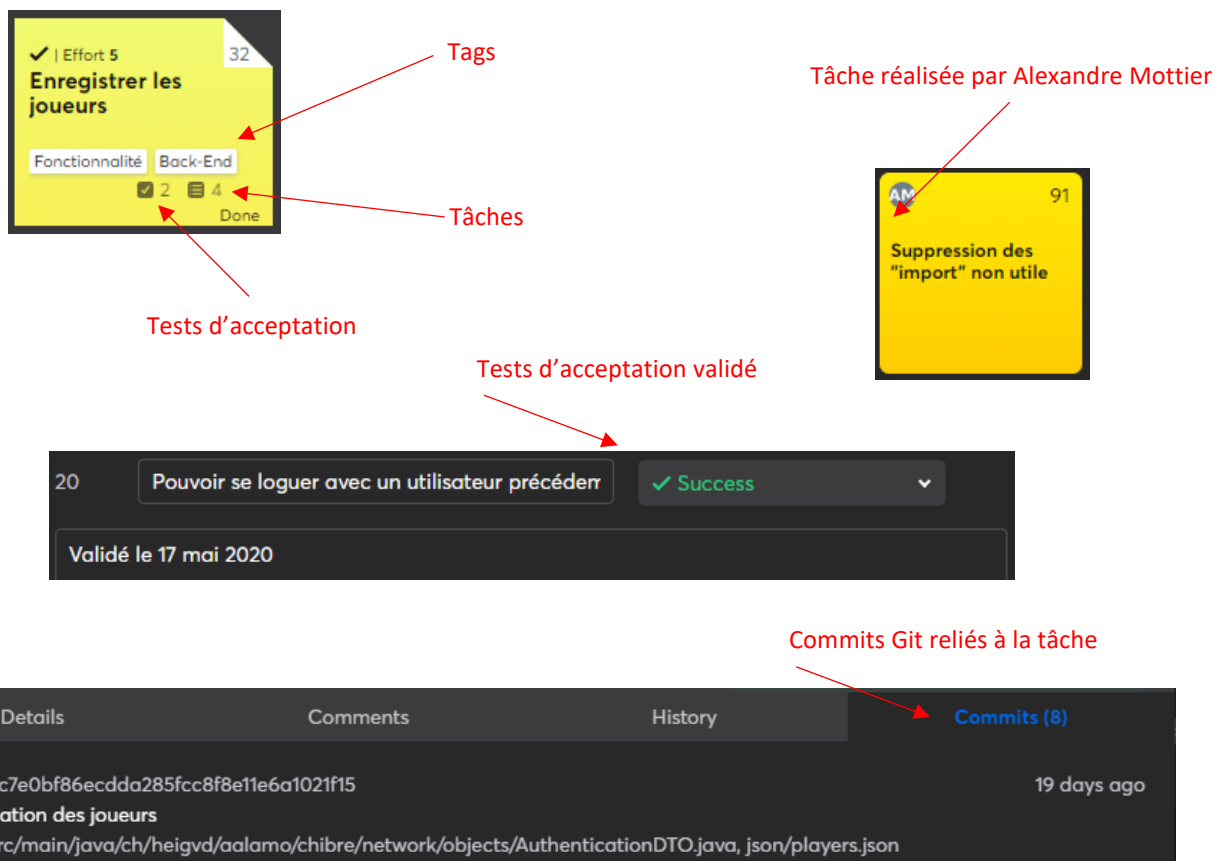
Ensuite, notre product owner, M. Lachaize organisait une séance pour valider des stories à réaliser. Nous devions alors planifier celles-ci et les ajouter à un sprint depuis le backlog.

Nous avons utilisé les options des stories qui permettaient de définir des tâches à réaliser pour parvenir au résultat d'implémentation de la story. Ces mêmes tâches nous permettaient d'avoir un aperçu en temps direct dans l'onglet « Task board » du travail à réaliser. Elles ont aussi permis de pouvoir répartir le travail entre les membres du groupe. Chacun était responsable de mettre à jour l'avancement de son travail en variant les heures restantes sur les tâches et de changer les statuts de ces dernières.

Afin de valider une story, nous avons autant que possible ajouté des tests d'acceptation. Un tel test permet de valider que l'implémentation réalisée correspond au résultat attendu de la story.

Nous avons choisi d'utiliser le plugin GitHub relié à iceScrum. Celui-ci nous a permis de lier des commits à des tâches et de pouvoir faire varier le temps restant d'une tâche depuis Git.

Voici quelques images présentant les fonctionnalités utilisées :



11. 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 le 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 créé 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.

Lors du dernier sprint, nous avons commencé par corriger les différents bugs d'interface graphique et de traitement backend. Nous avons ensuite amélioré et simplifier le code pour avoir une release la plus propre possible. Nous avons aussi beaucoup travaillé sur l'adaptation du rapport final et les schémas qu'il contient.

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 la globalité du projet. Nous avons passé beaucoup d'heures de travail sur ce dernier et nous pensons avoir atteints les objectifs de celui-ci. De plus, nous avons su travailler en groupe et pu profiter des avantages que cela apporte.

12. ANNEXES

Diagramme des cas d'utilisations

Diagramme du modèle de domaine

Diagramme de l'interface homme-machine

Diagramme de séquence

- newGame
- initGame
- runGame
- initRound

Diagrammes d'activité

- Représentation de l'utilisation des DTO
- Algorithme de détection de la meilleure annonce

Diagramme d'états

- Etats lors de l'authentification

Diagramme de conception globale des classes

- Programmes exécutables
- Classes Front-End
- Classes Back-End