7wonders

Rapport COO #2

EL HAROUI Jassim – DRISSET Nicolas – GASMI Zakaria

Encadré par : Mme. CABRIO

Table des matières

I. Introduction

Au cours de notre année en L3 Miage à l'Université de Nice Sophia Antipolis, nous devions réaliser un projet JAVA Client/Server pour le Jeu de 7wonders qui a était proposé par M. RENEVIER.

Suite à des difficultés rencontrées qui ont fortement impacté le premier rendu du rapport COO, ce document a donc pour but de bien expliquer l'architecture et le fonctionnement de notre projet suite à des diagrammes de Use Case qui montrent les cas d'utilisation, ensuite le diagramme d'activité qui représente l'interaction entre le serveur et les clients et des diagrammes de séquences qui permettent de bien montrer les messages échangés entre les lignes de vie, présentés dans un ordre chronologique, et finalement les diagrammes de classes qui détaillent les packages, les classes, leurs relation et fonctionnement.

II. Glossaire

Socket côté client : permet de se connecter à une machine distante afin de communiquer avec elle.

Socket côté serveur : connexion qui attend qu'un client vienne se connecter afin de communiquer avec lui.

Diagramme de cas d'utilisation : permet d'identifier les possibilités d'interaction entre le système et les acteurs.

Diagramme d'activité : permet de modéliser le comportement du système, dont la séquence des actions et leurs conditions d'exécution.

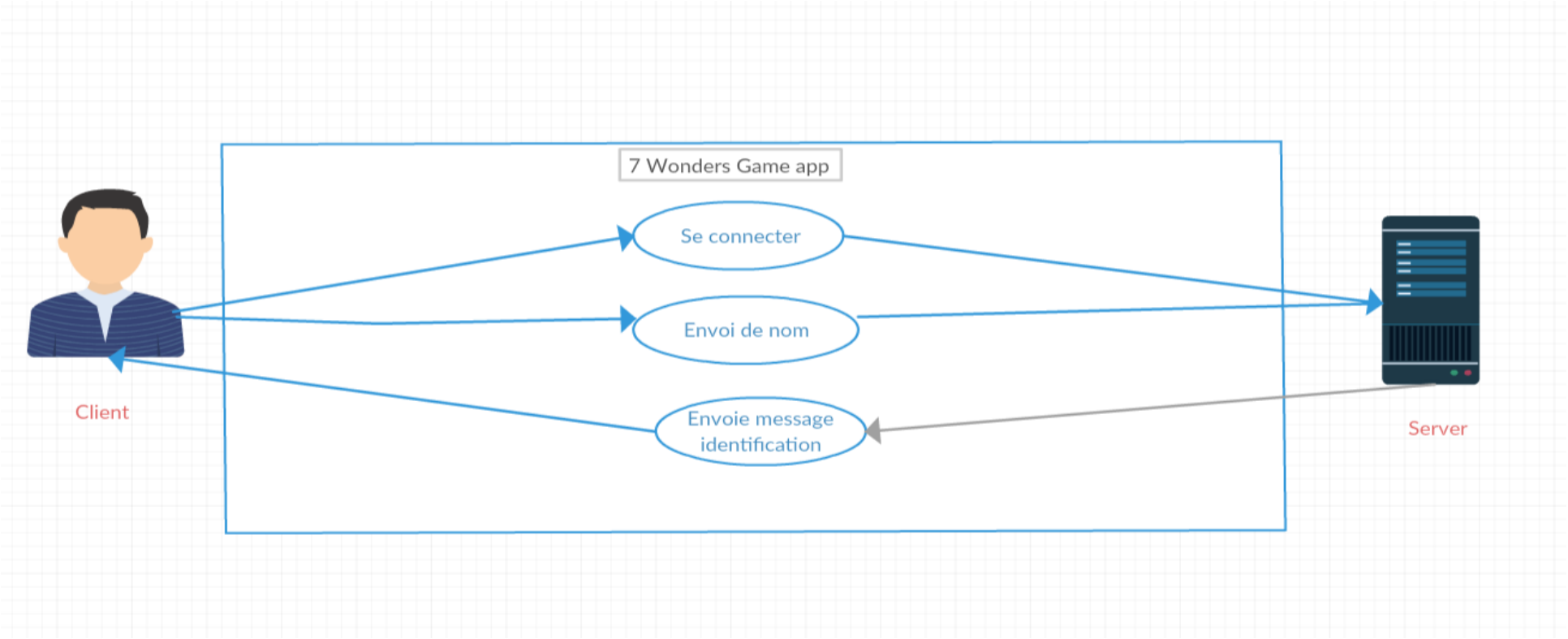
Diagramme de séquence : permet de bien montrer les messages échangés entre les lignes de vie, présentés dans un ordre chronologique

Diagramme de classe : permet de décrire clairement la structure en modélisant les classes, leurs attributs, leurs opérations et les relations entre eux.

III. Use cases

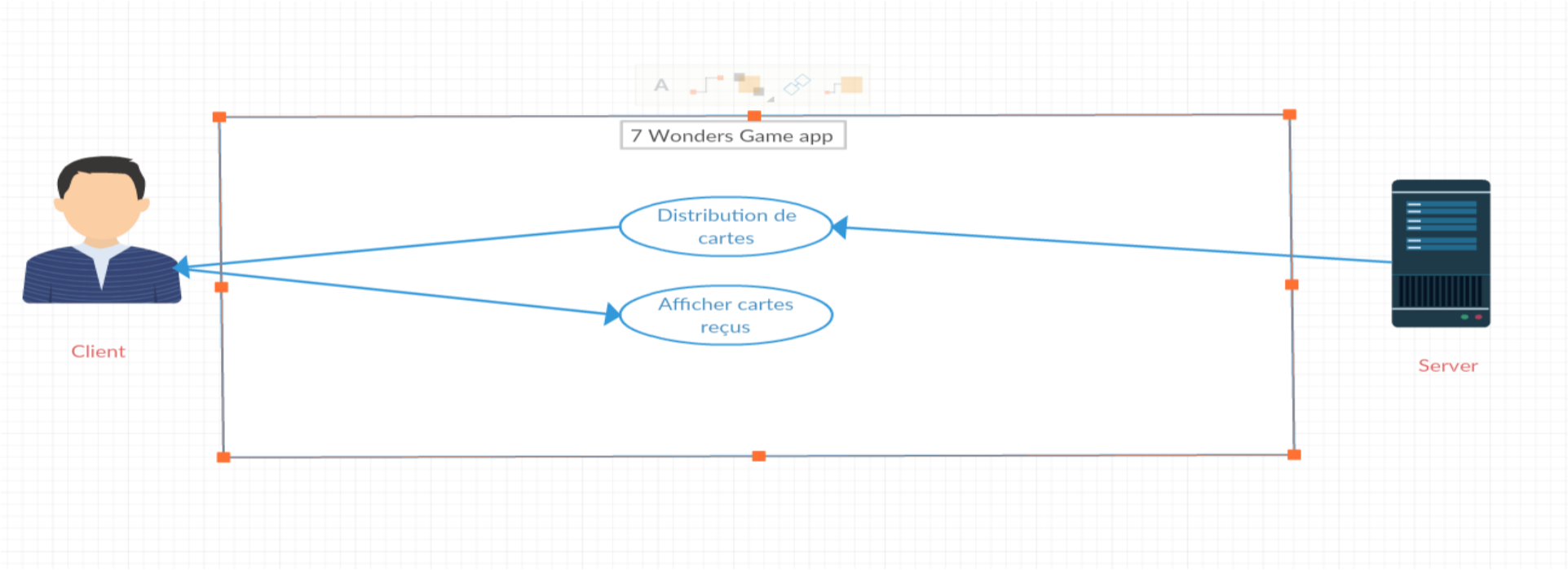
Dans cette partie on va parler de uses cases pour représenter les fonctionnalités de notre application jusque-là.

1. Identification



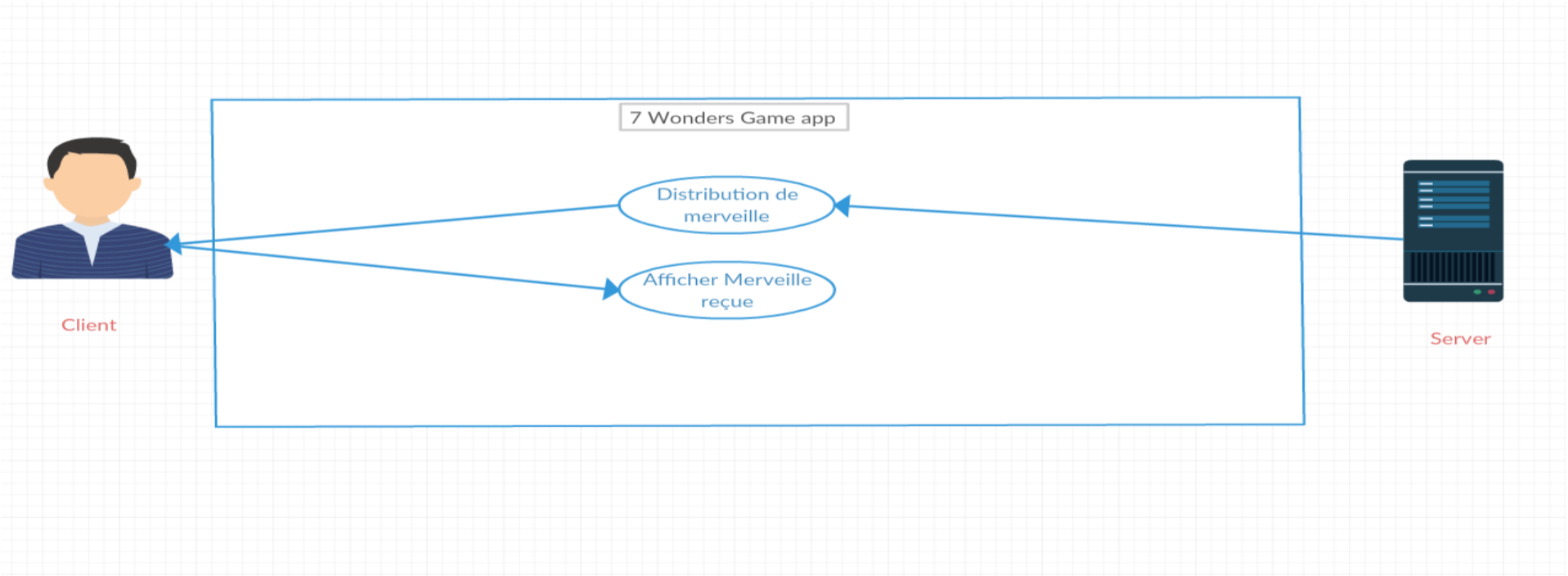
La Connexion Client qui consiste à la connexion du client et l'identification par le serveur via un listner coté serveur .

2. Distribution de cartes



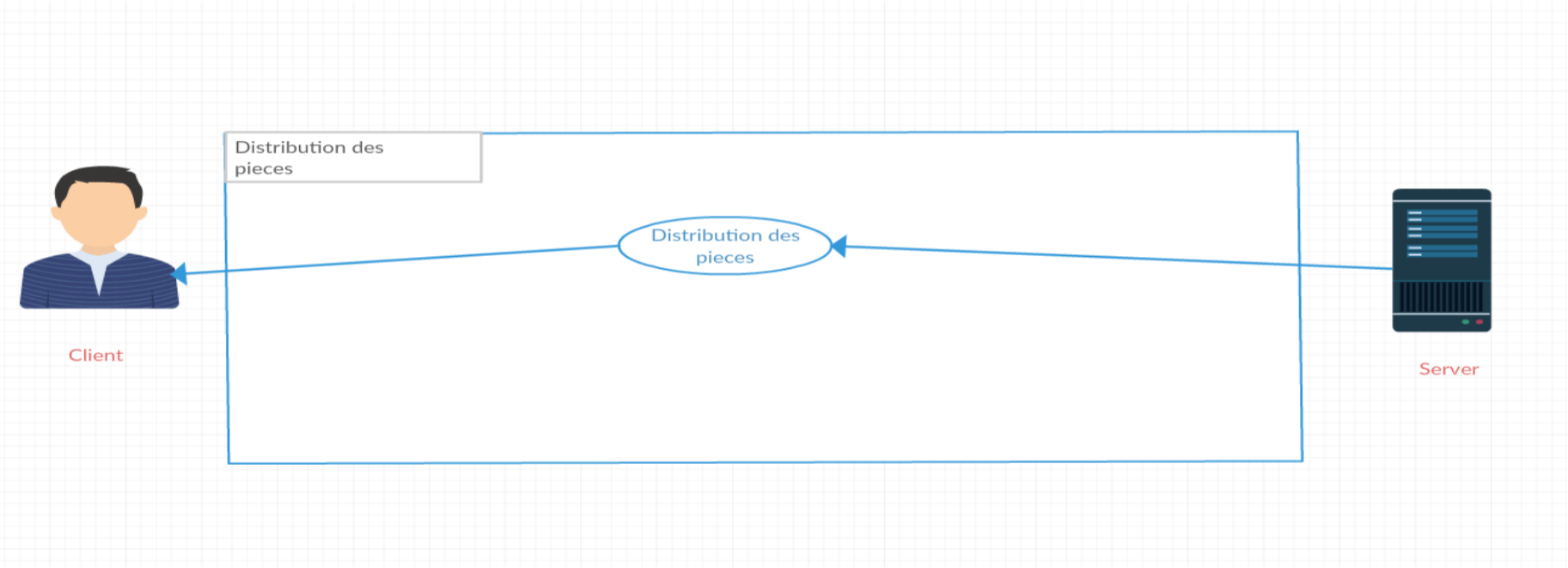
Distribution de cartes par le serveur d'une manière totalement aléatoire, le joueur affiche les carte reçus.

3. Distribution des merveilles



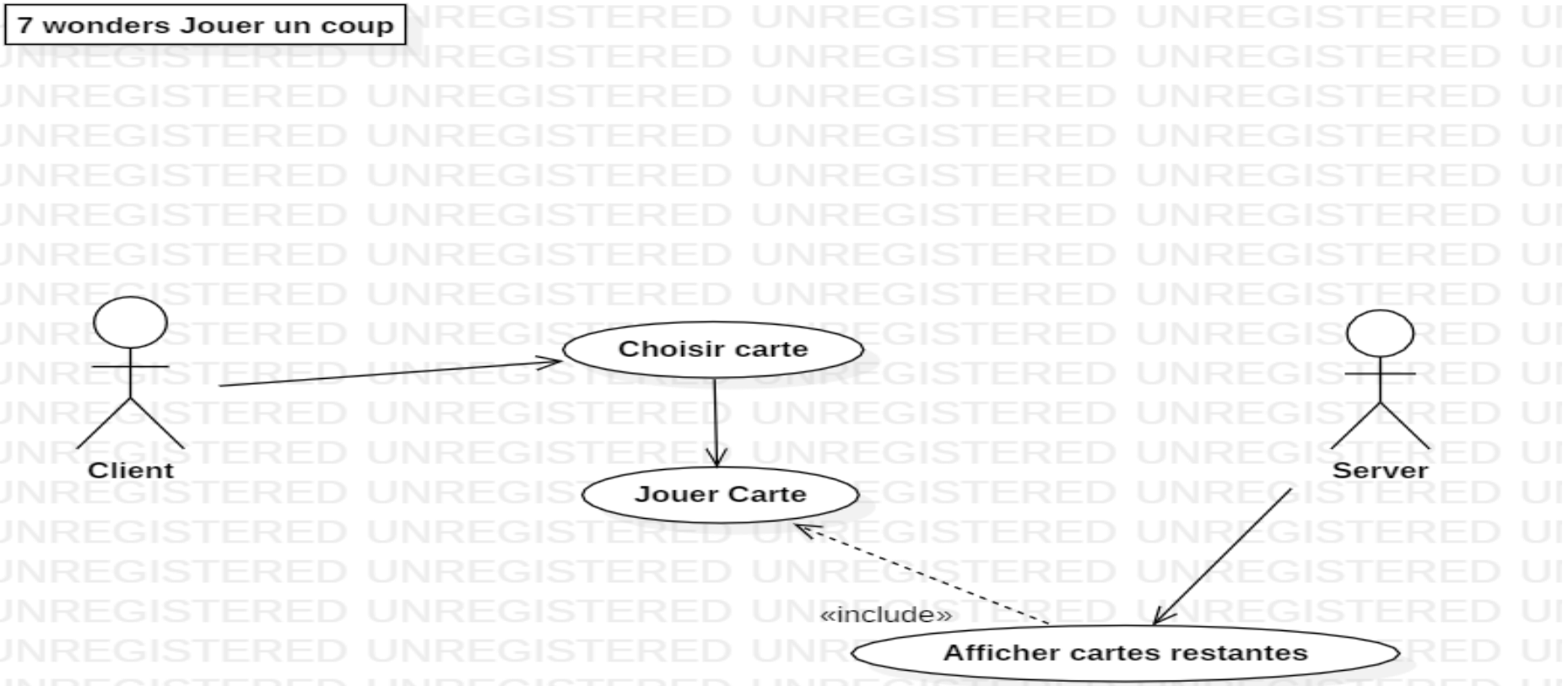
À l'instar de la distribution de cartes le serveur distribue pour chaque joueur une merveille aléatoirement, et le joueur affiche la merveille reçue.

4. Distribution de pièces



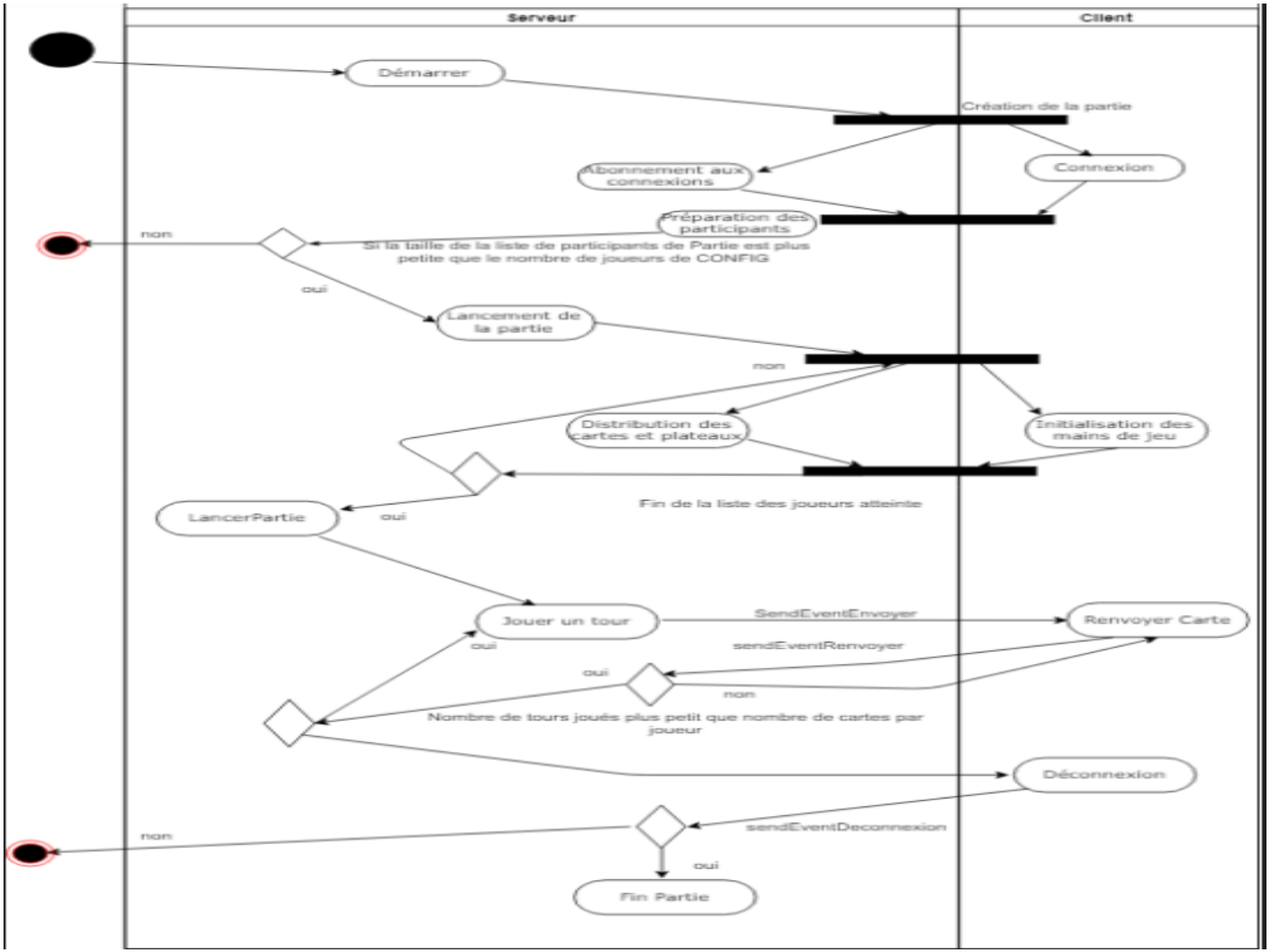
À cette étape le serveur distribue pour chaque joueur 3 pièces de valeur 1 de façon automatique et dès le lancement du jeu, vu que chacun dois systématiquement les posséder pour pouvoir débuter.

5. Jouer un coup



Le client choisis une carte, la met sur la table, et ainsi le serveur affiche les cartes restantes pour chaque joueur.

IV. Diagramme d'activité



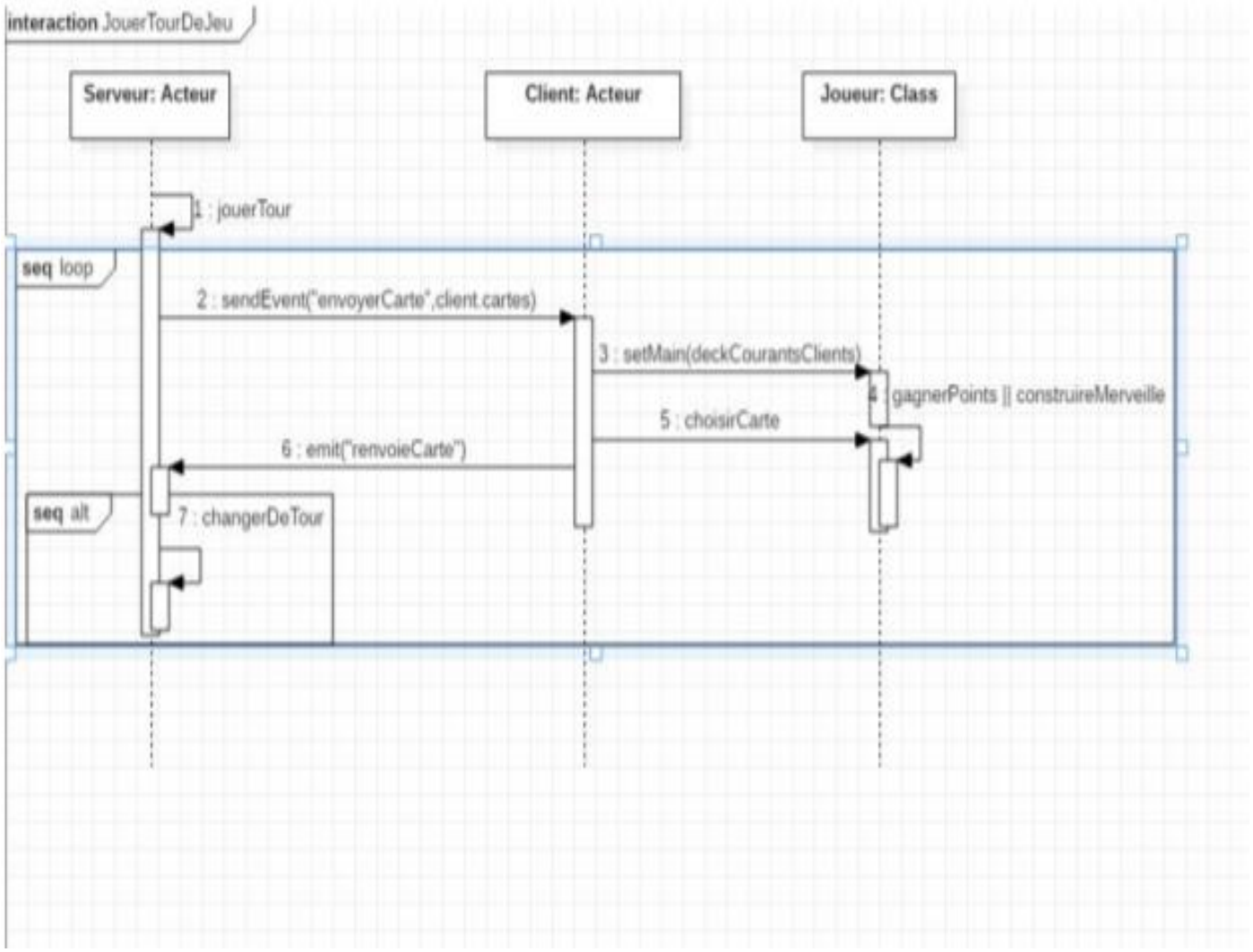
Ce diagramme représente l'interaction entre le serveur et les clients. Dans un premier temps, le serveur se lance et crée la partie. La création de la partie est alors lancée et le serveur s'abonne aux connexions des clients qui se déroulent en

parallèle. Le nombre de connexions représente le nombre de participants et ne doit pas dépasser le nombre de joueurs défini dans la configuration du jeu, sinon il y a sortie. Dans l'autre cas, le serveur lance la partie nouvellement créée et parcours la liste des joueurs pour jouer un tour. Tant que le nombre de tours est inférieur à 7 soit le nombre de cartes par joueur, on continue à rejouer un tour pour chacun des participants.Le decision node placé après le controleFlow : sendEventRenvoyer sert à indiquer que l'on rejoue une fois que tous les joueurs ont joué leurs cartes.

De même, le controlFlow : sentEventDeconnexion permet de fixer la condition selon laquelle il reste des joueurs connectés ou non pour déterminer si l'on quitte la partie ou non .Si tous les clients sont déconnectés, la partie peut alors être quittée.

V. Diagramme de séquence

1. Modélisation d'un tour de jeu

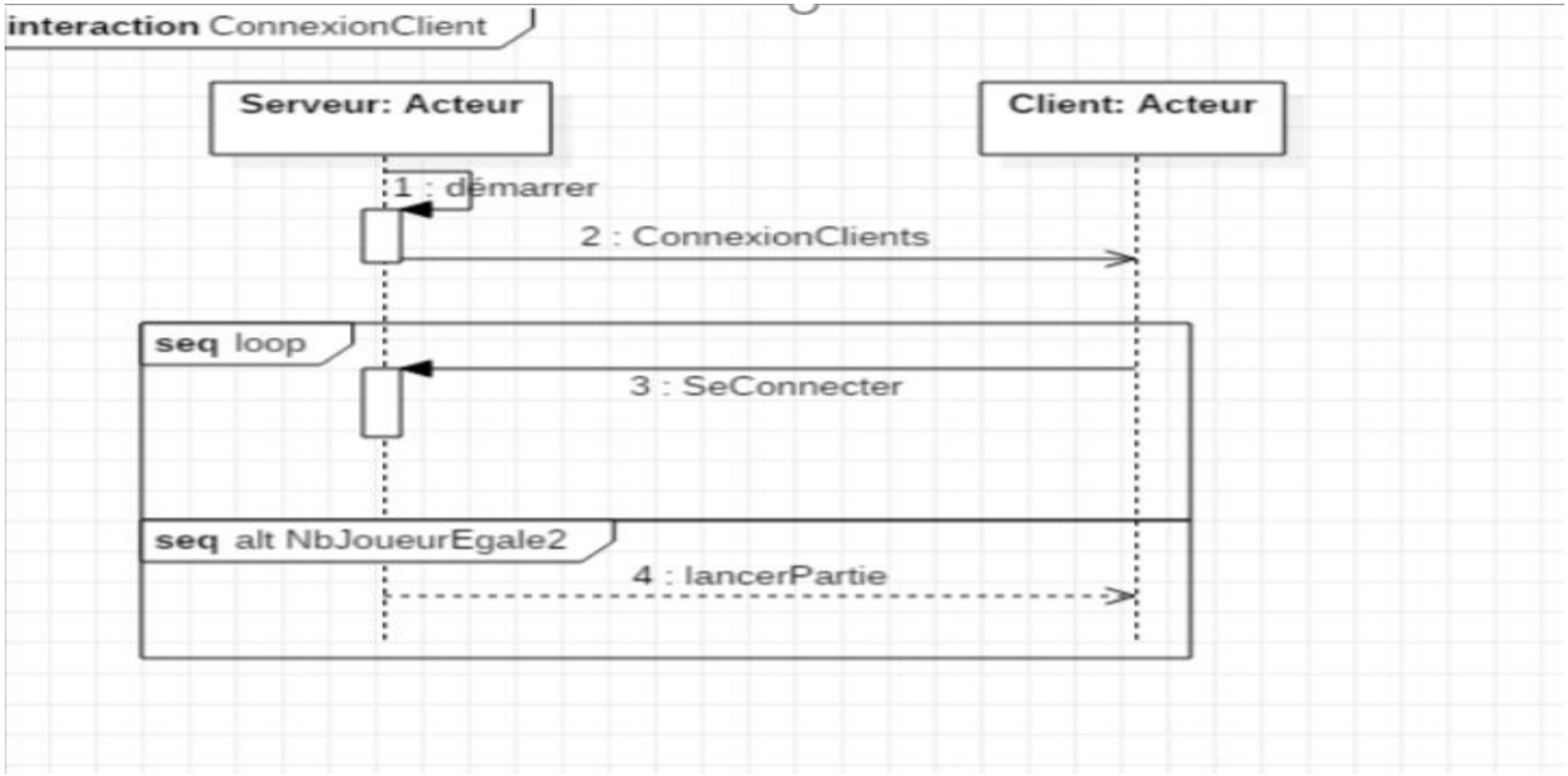


Ce diagramme de séquence illustre un tour de jeu mené par le serveur.

Le serveur joue un tour et se charge de distribuer les cartes aux clients. Le client a alors une réorganisation de sa main, avec la méthode setMain à partir de laquelle il le client choisit une carte et peut alors renvoyer un événement de type emit avec la carte choisie en argument.

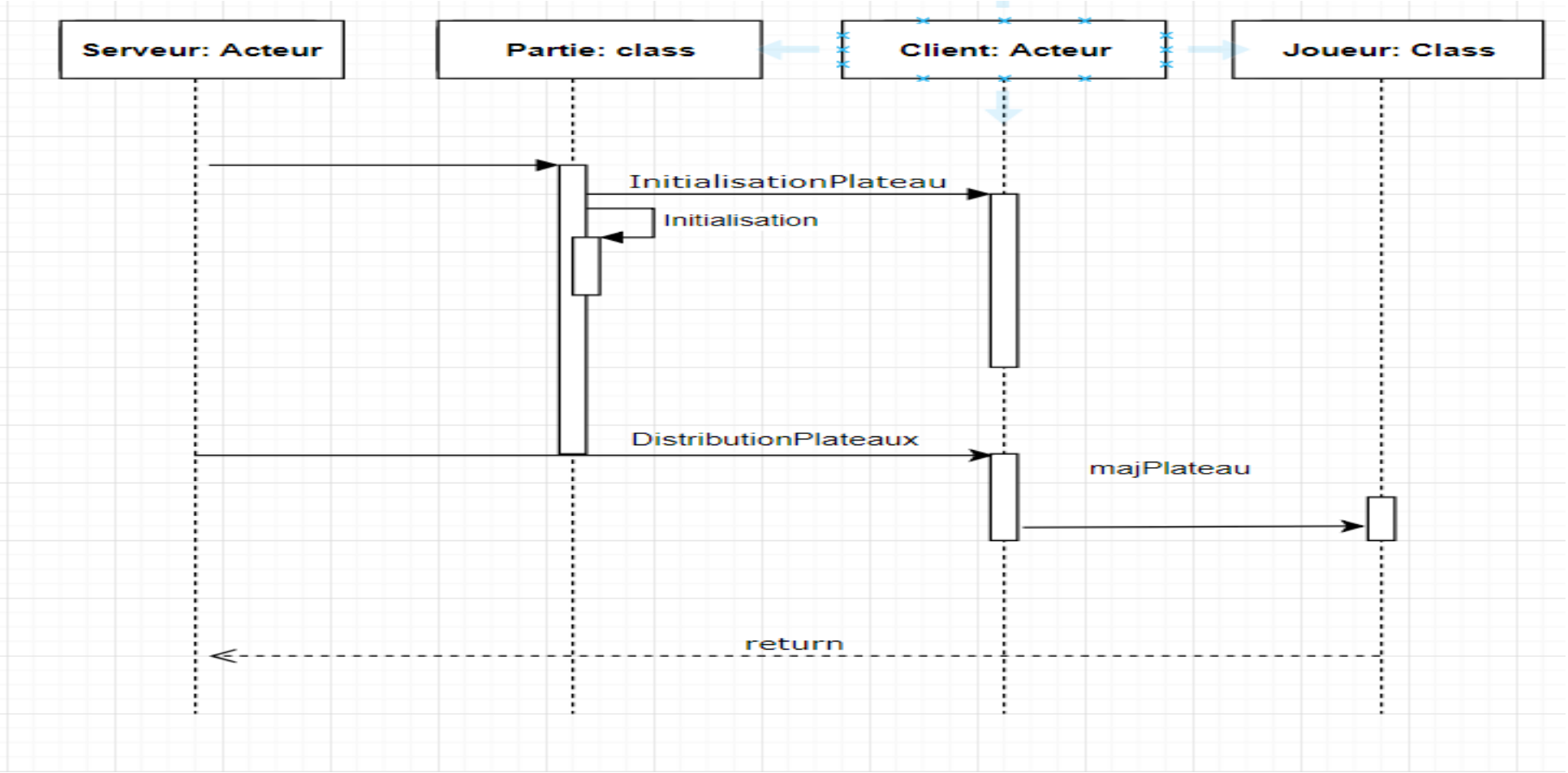
Une fois sur deux le joueur peut, s'il reste de la place sur le plateau qu'il a choisi construire une merveille, ou sinon il jouera une carte lui permettant de gagner des points

2. Modélisation de la connexion Client Serveur



Ce diagramme représente le serveur qui de son démarrage va demander la connexion des clients de façon asynchrone. Le serveur peut ainsi lancer la partie une fois que le nombre attendu de joueurs connectés figurant dans la classe config, est atteint.

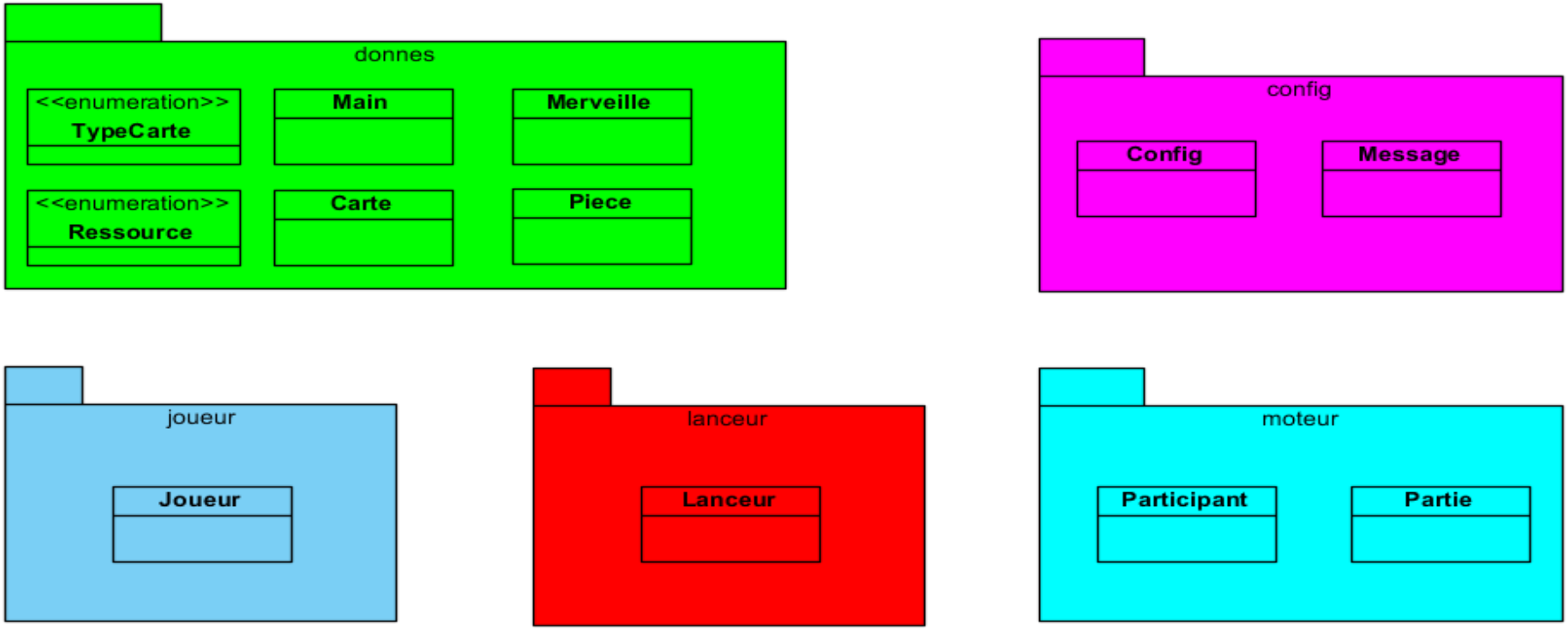
3. Modélisation de la distribution des plateaux par le serveur



Le serveur initialise tout d'abord les plateaux. Puis pour chaque joueur connecté le serveur distribue un plateau de jeu représentant une merveille.

VI. Diagramme de classe

1. Les packages



Le package données:

Ce package contient tous les éléments de jeu qui seront utilisés dans les autres packages pour lancer une fonction et la faire fonctionner.

Le package Config :

Ce package contient des informations qui se changent pas en static final (la configuration du serveur ainsi que les messages à afficher dans la console).

Le package joueur :

Ce package contient le joueur avec tous ses caractéristiques (la merveille, les cartes, les pièces..) et il s'occupe de la partie réseau d'un client.

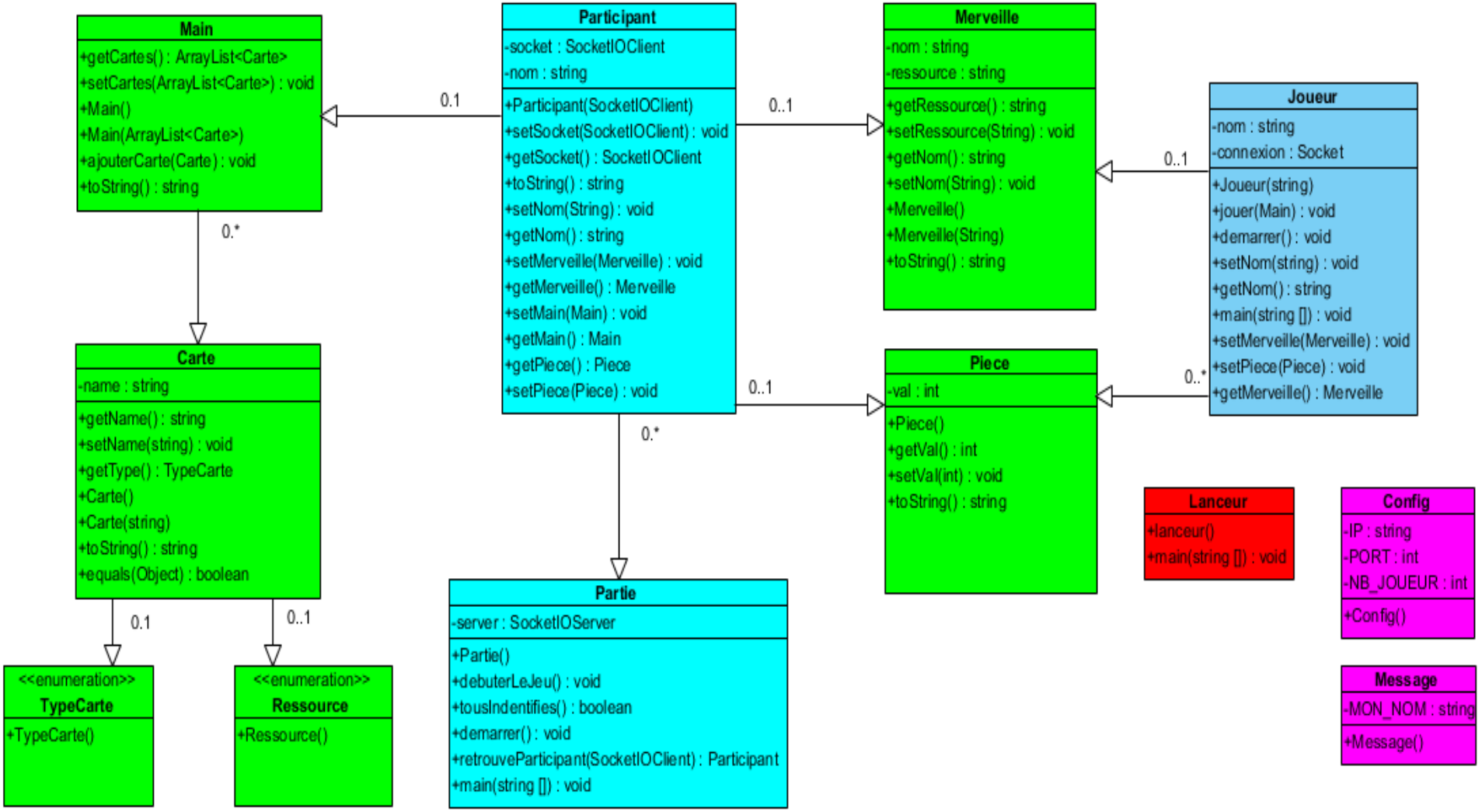
Le package moteur:

Ce package gère la partie réseau du serveur qui est lancé à partir de la classe Partie
Avec l'aide des informations du client qui se trouve dans Participant.
C'est le principal package qui lance les échanges Serveur-Client.

Le package lanceur:

C'est le package qui contient l'exécutable qui lance l'application.

2. Diagramme de classe détaillé



La partie verte : elle contient les classes (Merveille, Carte et Pièce) et les énumérations (Typecarte et Ressources) qui composent le package ‘donnees’ qui contient lui-même tous les éléments de jeu qui seront utilisés dans les autres packages surtout pour la classe Joueur.

La partie bleue foncée : elle contient que la classe Joueur qui dispose de Merveille et Piece et la socket connexion et elle sert à identifier un joueur après la connexion au serveur et lance la distribution de merveilles, des pièces et des cartes en utilisant du Json JsonObject et JsonArray.

La partie bleue claire : cette partie gère les échanges entre Client et Serveur, la classe Participant contienne les informations du client et s’occupe de la partie client alors que la classe Partie lance le coté Server.

La partie violette : elle contienne deux classe pour le stockage des informations non changeable ; la classe Config pour les données de config et la classe Massages pour les données affichées sur la console

La partie rouge : c’est la partie principale qui lance l’application et contienne l’exécutable.

VII. Conclusion

1. Analyse de notre solution (points forts et points faibles)

Points faibles :

Retard considérable pour le lancement de la production du projet pour des causes personnelles qui ont fortement impacté sur ce dernier.

Problèmes techniques entre la compréhension du MAVEN en synchronisation Avec GitHub.

Difficulté de compréhension du client serveur à cause de gros problèmes personnels qui a résulter d'absences lors des séances des cours du client serveur.

Points fort :

Evolution lors des dernières itérations grâce aux conseils et remarques de Mme. CABRIO.

2. Evolution prévue

Mettre en place un bon nombre d'améliorations et gérer bien le temps.