UNIVERSITÉ
DE REIMS
CHAMPAGNE-ARDENNE

Implémentation du Poker Texas Hold'em en Java avec l'archetecture client/serveur

Compte rendu TP3 INFO502

Élève :
Moussa TAYEB NEMICHE

Enseignants :
OLIVIER FLAUZAC
GEOFFREY WILHELM

Table des matières

1 Introduction

Ce rapport présente une analyse détaillée de l'implémentation d'un jeu de Poker Texas Hold'em en Java. Le code source fourni met en œuvre les mécaniques fondamentales du jeu, en reprenant le code du TP2 et en y apportant quelques modifications pour adopter une architecture client/serveur.

2 Architecture Globale

2.1 Structure des Classes

La classe principale dans ce code est PokerHoldem, qui contient toute la logique du jeu. Elle agit comme le coordinateur principal sur lequel repose l'architecture du jeu. Elle est complétée par la classe Joueur, qui gère les joueurs au sein d'une table, ainsi que la classe CartesCommunes, responsable de la gestion des cartes partagées sur la table

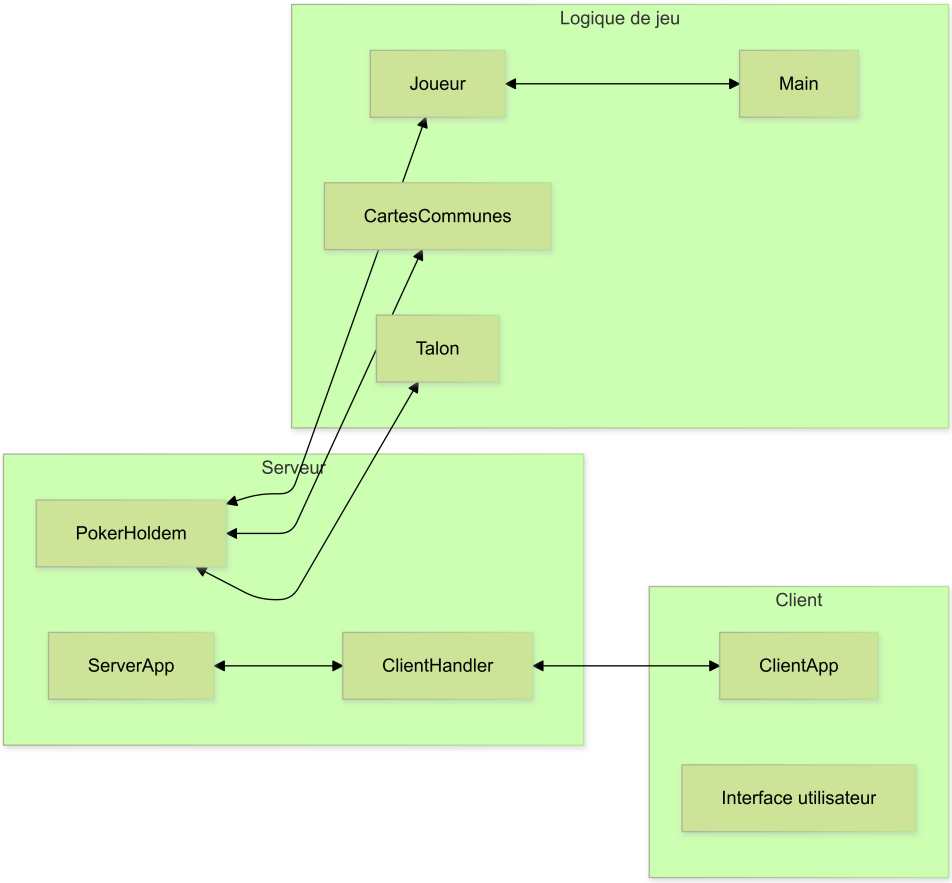

FIGURE 1 – Architecture du système

3 Analyse des Composants

3.1 Classe PokerHoldem

La classe PokerHoldem contient la logique principale du jeu. L'initialisation de cette classe s'effectue avec une validation du nombre de joueurs (entre 2 et 10 joueurs), conformément aux règles standard de Texas Hold'em. Elle gère également la création unique des éléments nécessaires au début de la partie, ainsi que l'initialisation des joueurs et des cartes communes. La gestion des parties respecte les règles du jeu, avec une séquence ordonnée d'actions, telles que la distribution des cartes privées et la gestion des différentes phases du jeu (Flop, Turn, River). Enfin, l'évaluation finale des combinaisons de mains permet de déterminer le gagnant de la partie.

3.2 Classe Joueur

La classe Joueur implémente une gestion des aspects individuels de chaque participant. Elle maintient l'état des cartes privées du joueur tout en fournissant des mécanismes pour l'évaluation de la meilleure main possible. Un aspect particulièrement notable est l'implémentation de l'algorithme de génération des combinaisons de cartes, qui permet de trouver toutes les possibilités de main pour déterminer la meilleure combinaison possible pour chaque joueur.

3.3 CartesCommunes

La gestion des cartes communes est assurée par une classe dédiée qui impose une limite stricte de cinq cartes, en réspectant les règles du Texas Hold'em. Cette classe offre des fonctionnalités de réinitialisation pour les nouvelles parties et maintient l'intégrité des données grâce à l'immutabilité des collections exposées. La protection des données est assurée par des mécanismes de contrôle, qui empêchent toute modification non autorisée des cartes communes.

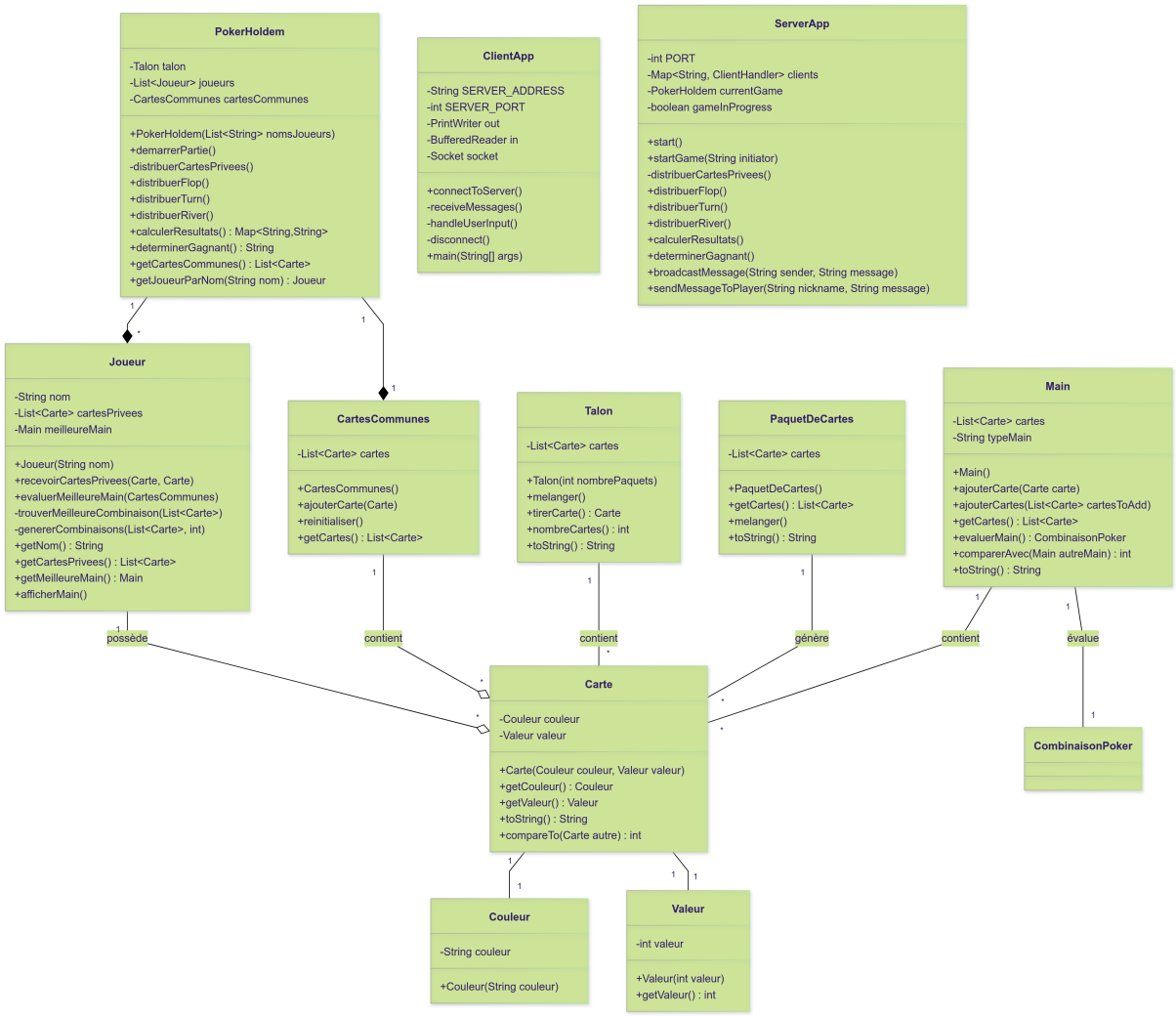

FIGURE 2 – Diagramme UML

4 Architecture Client-Serveur

Cette implémentation du jeu de Poker Texas Hold'em utilise une architecture client/-serveur. Cette séparation des rôles entre le serveur et les clients permet de centraliser la logique du jeu sur un serveur, tout en offrant aux utilisateurs une interface pour interagir avec celui-ci. Le modèle client/serveur est une approche idéale pour les jeux multijoueurs, comme ce jeu de Texas Hold'em, car il permet de centraliser le traitement et de synchroniser l'état des différents joueurs.

4.1 Le Serveur

Le serveur joue un rôle central dans la gestion de l'état du jeu. Il est responsable de la création des parties, en coordination avec les classes PokerHoldem et Joueur, pour gérer

la distribution des cartes et les différentes étapes du jeu (Flop, Turn, River). Une fois la partie initiée, le serveur reçoit les actions des joueurs et les traite conformément aux règles du Texas Hold'em. Il envoie également des mises à jour aux clients pour assurer la synchronisation et permettre l'affichage des informations pertinentes.

Le serveur est aussi chargé de vérifier que les actions des joueurs sont valides et de résoudre les conflits, par exemple, si un joueur tente d'effectuer une action non permise ou d'agir de manière incohérente par rapport à l'état du jeu.

Le serveur utilise une architecture basée sur les **sockets réseau** pour communiquer avec les clients. Chaque client est gérée dans un thread différent, ce qui permet au serveur de traiter simultanément plusieurs clients de manière indépendante. Cette architecture permet de garantir une gestion centralisée et synchronisée du jeu, tout en réduisant la charge sur chaque client.

4.2 Le Client

Le client, quant à lui, est l'interface utilisateur qui permet aux joueurs de rejoindre une partie, d'effectuer des actions (START, HELP ; QUIT), et d'avoir une vision générale de l'état de la partie. Chaque joueur se connecte au serveur via une en ligne de commande. Le client communique avec le serveur en envoyant des messages avec les sockets TCP et en recevant des informations sur l'état du jeu.

Sur le plan technique, le client est conçu pour être léger. Il ne garde qu'une copie locale de l'état actuel du jeu, qu'il met à jour chaque fois qu'il reçoit de nouvelles informations du serveur. Cela permet de minimiser les ressources utilisé et de réduire la complexité du client.

4.3 Choix Techniques

- **Protocoles de Communication** : Le serveur et les clients communiquent via des **sockets TCP**, qui permettent une communication fiable et ordonnée. Ce protocole garantit que les messages envoyés entre le client et le serveur arrivent dans le bon ordre, ce qui est essentiel pour maintenir la cohérence de l'état du jeu.
- **Gestion de la Concurrence** : Pour permettre à plusieurs joueurs de se connecter et de jouer simultanément, le serveur utilise des **threads** (fils d'exécution) pour gérer chaque connexion de manière indépendante. pour optimiser les performances et d'assurer que le serveur peut gérer de multiples joueurs en même temps, sans que chaque connexion n'interfère avec les autres.
- **Scalabilité** : L'architecture client/serveur est conçue pour être scalable. Le serveur peut être déployé sur une machine puissante capable de gérer un grand nombre de joueurs en même temps, tout en réspectant les règles du jeu Texas Hold'em.

Le schéma ci-dessus représente les requêtes entre le client et le serveur

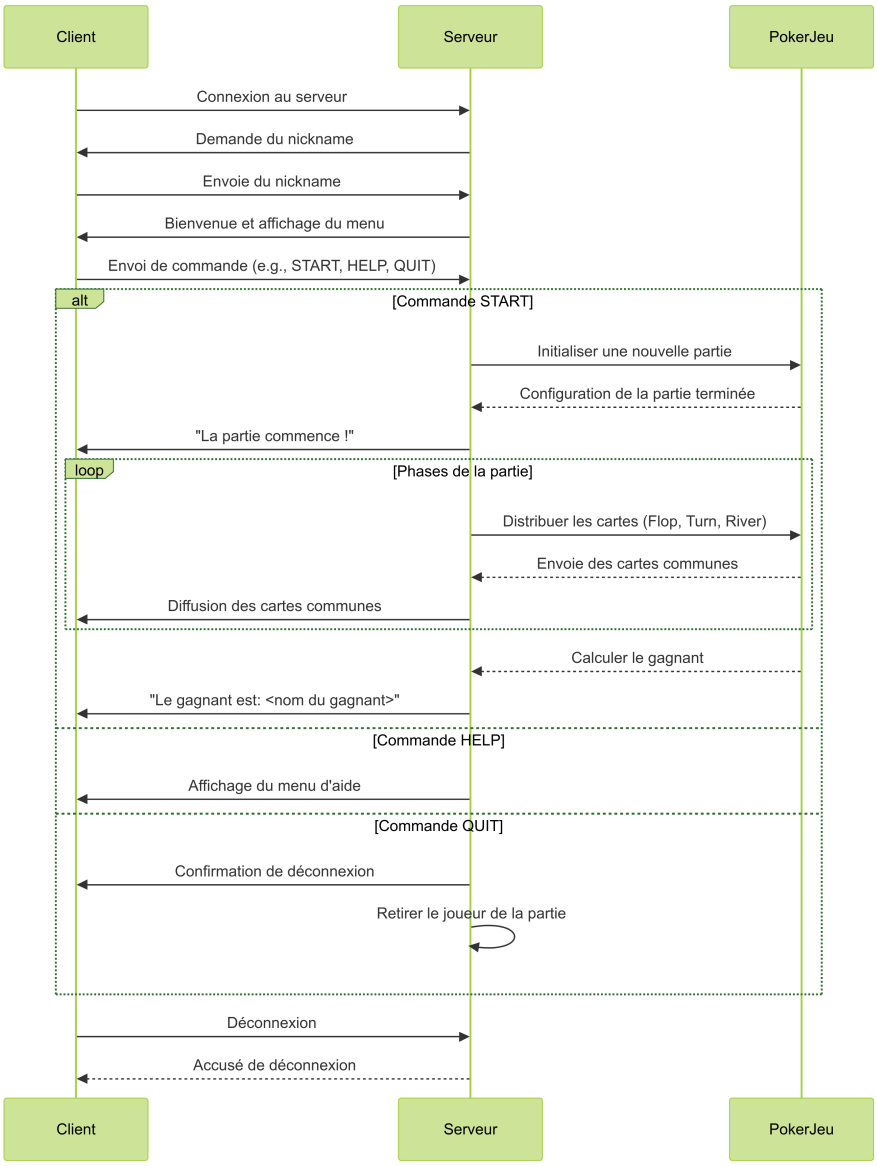

FIGURE 3 – Chronogramme des requêtes

4.4 Avantages

- **Centralisation du Contrôle** : Le serveur centralisé permet de s'assurer que toutes les règles sont appliquées correctement, en évitant toute modification frauduleuse de l'état du jeu par les joueurs.
- **Synchronisation** : Les informations sont envoyées en temps réel à tous les joueurs, pour synchroniser l'état de jeu entre plusieurs joueurs
- **Extensibilité** : L'architecture client/serveur permet d'ajouter facilement de nouvelles fonctionnalités côté serveur, comme la gestion de différents types de jeux ou l'intégration de nouveaux modes de jeu.

Présentation de l'application

Cette section présente l'application avec des images de différent acteur dans ce processus, donc 2 clients et un serveur.

Le réperetoire git du projet est structuré de manière à séparer les responsabilités entre deux branches principales :

- **Branche client** : Cette branche contient le code des acteurs **Acteur 1** et **Acteur 2**. ces deux acteurs représentent les clients donc exécutent le code client pour interagir avec le serveur.
- **Branche serveur** : Cette branche contient le code pour l'infrastructure, donc la partie serveur. Le serveur exécute le code serveur pour gérer les connexions et les échanges avec les clients, ou il attribue un thread à chaque client qui se connecte.

Illustrations

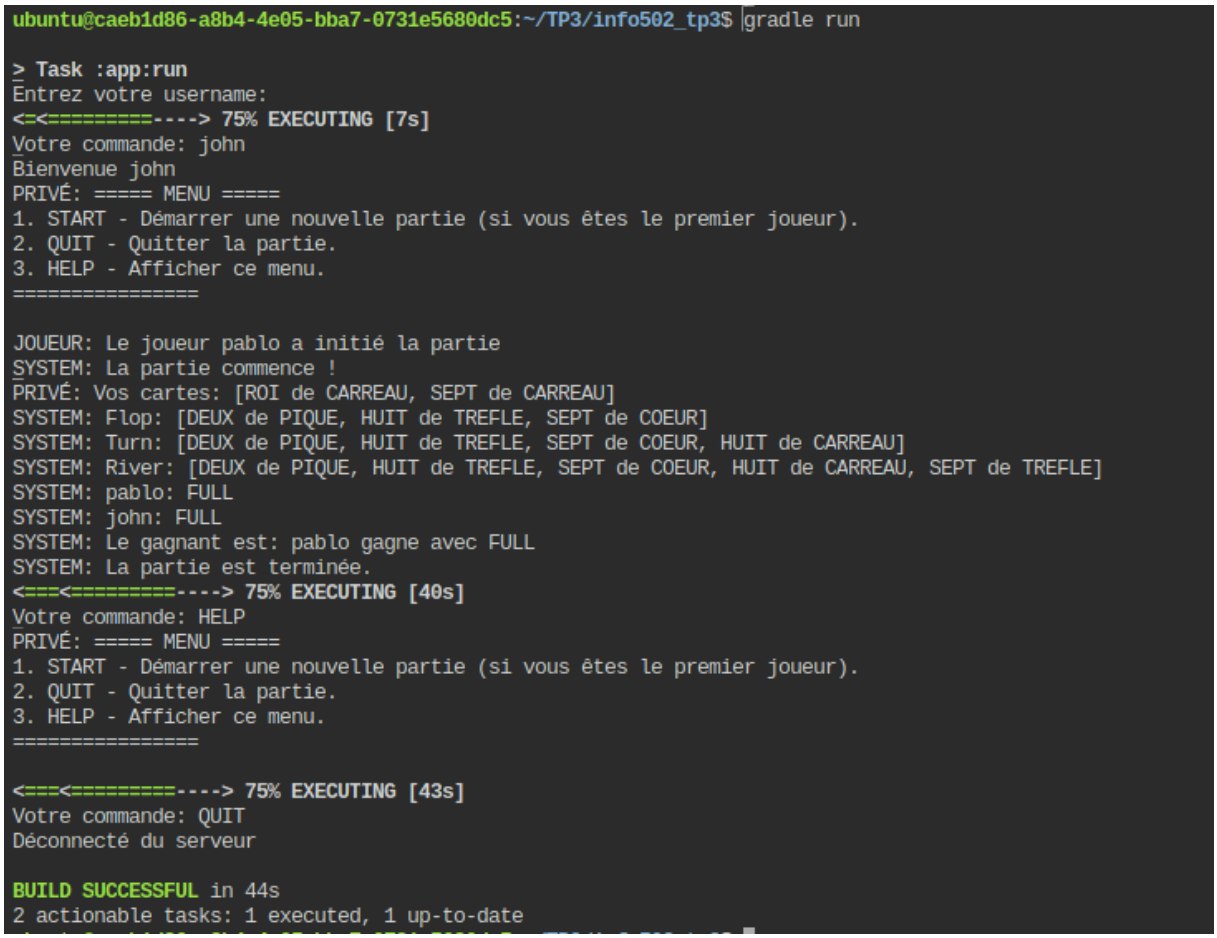

FIGURE 4 – Interface de l'application pour Acteur 1 (branche client).

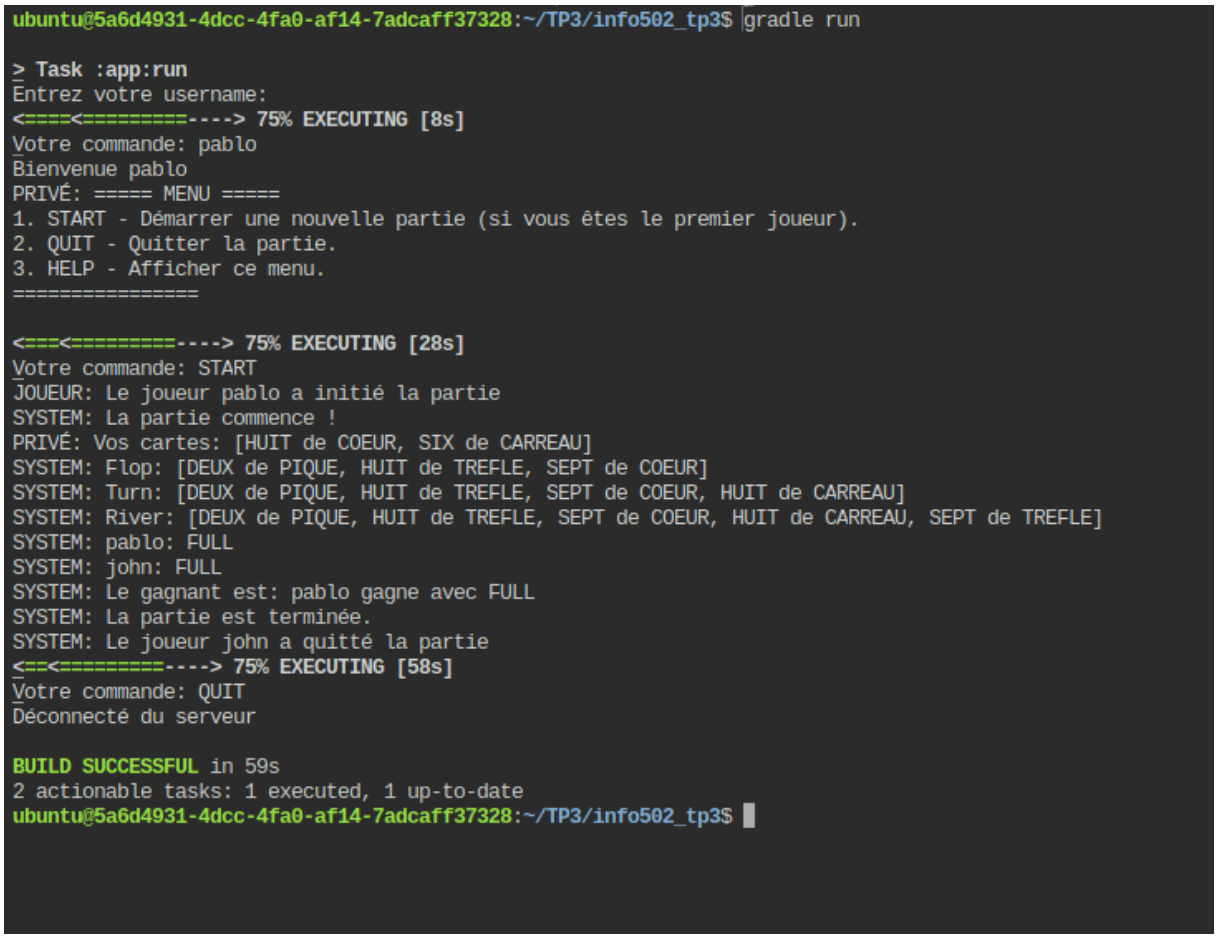

FIGURE 5 – Interface de l'application pour Acteur 2 (branche client).

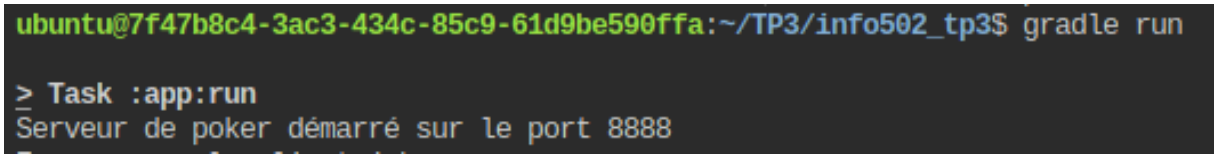

FIGURE 6 – Infrastructure de l'application côté serveur (branche serveur).

Les images ci-dessus illustrent les interfaces pour les deux acteurs ainsi que l'infrastructure du serveur. ce qui rend le déploiment de l'application, plus facile et plus simple a utiliser :

- Pour **Acteur 1** et **Acteur 2**, il suffit d'utiliser la branche client. - Pour **Infrastructure**, il suffit d'utiliser la branche serveur.

Listing 1 – Commandes Git pour les branches

```
# Pour accéder à la branche client
git checkout client

# Pour accéder à la branche serveur
git checkout serveur
```

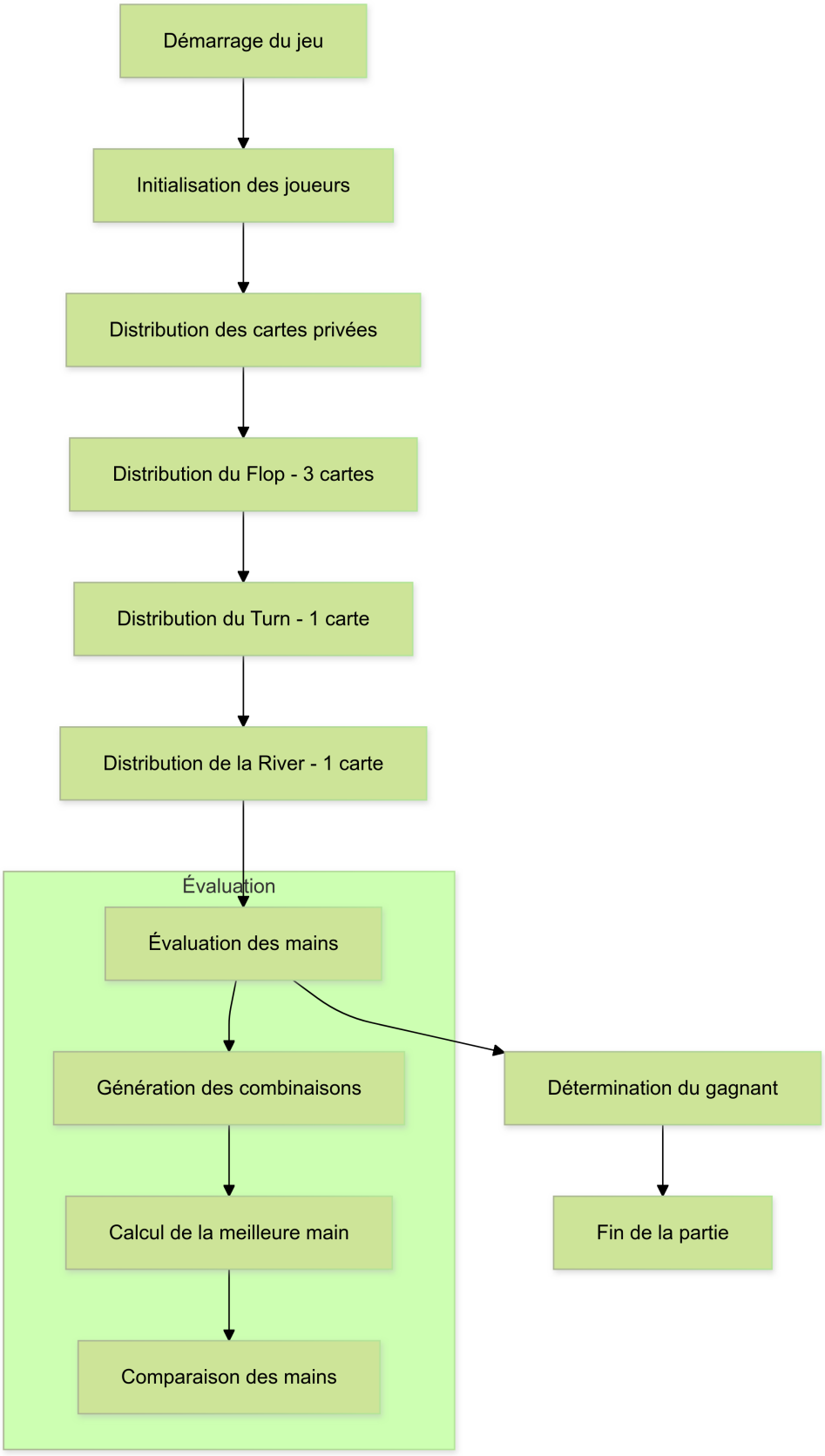

FIGURE 7 – Diagramme de flux

4.5 Conclusion

L'architecture client/serveur utilisée pour ce jeu de Poker Texas Hold'em permet une gestion centralisée et fluide des parties, tout en offrant une interface utilisateur simple et interactive. Les choix techniques, comme l'utilisation des sockets pour la communication et des threads pour la gestion concurrente, assurent une expérience de jeu stable et extensible. Ce modèle permet d'ajouter des fonctionnalités supplémentaires et d'améliorer les performances au fur et à mesure des besoins.