

CENTRALE LYON

APPLICATION WEB
RAPPORT DE PROJET

Connect 4

Élèves :

Ulysse DURAND
Oumaima LAKLOUCH

Enseignants :

René CHALON
Romain VUILLEMOT

1^{er} avril 2025

Table des matières

Introduction	2
1 Principe du jeu :	2
2 Fonctionnement du code	3
2.1 structures de données	3
2.1.1 rooms	3
2.1.2 Hall of Fame	3
2.2 Communication entre client et serveur	3
2.2.1 serveur -> client	3
2.2.2 client -> serveur	4
3 Exemple d'utilisation	4

Introduction

1 Principe du jeu :

Connect 4 est une application web multijoueur qui permet à plusieurs utilisateurs de s'affronter en ligne . Le système de jeu repose sur des salles virtuelles, au nombre de dix disponibles en simultan  , chacune pouvant accueillir jusqu'   deux joueurs.

Le plateau de jeu propos   dans l'application est une grille de 6 lignes par 7 colonnes . Les joueurs d  posent    tour de r  le un pion dans la colonne de leur choix, le pion tombant automatiquement    la position la plus basse disponible dans cette colonne.

Le but du jeu est simple : les joueurs doivent aligner quatre pions de leur couleur, que ce soit horizontalement, verticalement ou en diagonale, avant leur adversaire. Le premier joueur    r  ussir cet alignement remporte la partie.

   chaque victoire, le joueur obtient un point suppl  mentaire qui est ajout      son score global. Ce score est ensuite visible dans le Hall of Fame, un tableau de classement affichant les meilleurs joueurs. Ce classement est mis    jour automatiquement au fil des victoires, permettant de suivre la progression et la performance de chaque joueur.

Voici les fonctionnalit  s attendues de l'application

Cahier des charges

- Utiliser l'  l  ment canvas
- Doit   tre multijoueur (≥ 2 joueurs)
- Utiliser socket.io pour une connexion permanente au serveur
- Faire un hall of fame
- Ajouter un chat qui utilise aussi socket.io

Design

- Un nombre fini de salles de jeu (playroom) : 10

Page principale

url : root

- Une liste de salles (playroom) avec le nombre de joueurs    l'int  rieur
- Un hall of fame affichant ceux qui ont le plus souvent gagn  
- Un champ pour rentrer son pseudo

Pour rejoindre une playroom il faut un pseudo, sinon redirection vers root/ avec message d'erreur. Si la game est pleine, pareil, redirection vers root/ avec message d'erreur.

Salle

url : root/roomX (room0 - room9)

- Un canvas avec une grille de jeu
- Un chat avec l'autre joueur
- Un bouton "Import a game" -> Demande de choisir un fichier    importer, si valide, demande    l'autre joueur si il veut bien accepter son fichier
- Un bouton "Export the game" -> Redirige vers root/roomX/download qui fait t  l  charger

Tout sauf les fonctionnalit  s "Import a game" et "Export a game" a   t   fait.

2 Fonctionnement du code

2.1 structures de données

2.1.1 rooms

Les salles contiennent :

- id :
L'identifiant de la salle (room0, ..., room9)
- players :
La liste des joueurs dans la salle
- gameState :
Le plateau dans la salle (sous forme d'un tableau de null | 0 | 1)
- currentPlayer :
L'id (0 ou 1) du joueur dont c'est le tour
- moves :
Pas utilisé, est supposé stocker la liste des coups pour pouvoir enregistrer la partie.

2.1.2 Hall of Fame

le hall of fame est simplement un dictionnaire <playerName,winCount>, c'est comme ça qu'il est représenté dans le fichier json.

2.2 Communication entre client et serveur

Le code utilise un WebSocket pour faire la communication entre client et serveur.
Voici la liste des messages possibles via WebSocket :

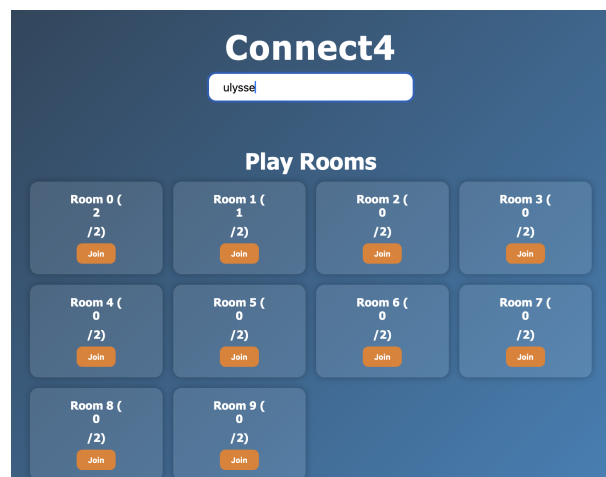
2.2.1 serveur -> client

- updateRooms
donne la nouvelle composition des salles de jeu (pour mettre à jour le nombre de joueurs dans chaque salle)
- updateHOF
donne le nouveau hall of fame (pour le mettre à jour dès que quelqu'un gagne)
- startGame
donne l'ordre de lancer la partie, spécifie aussi l'état du plateau de jeu
- stopGame
donne l'ordre de terminer la partie, spécifie aussi l'état du plateau de jeu
- error
signale qu'il y a une erreur, comme que la salle à rejoindre est pleine ou que la salle à rejoindre n'existe pas.
- updateGame
met à jour l'état du plateau de jeu (lorsque que quelqu'un joue par exemple)
- message
s'envoie quand un message est envoyé dans le chat de la salle de jeu, il contient le contenu du nouveau message

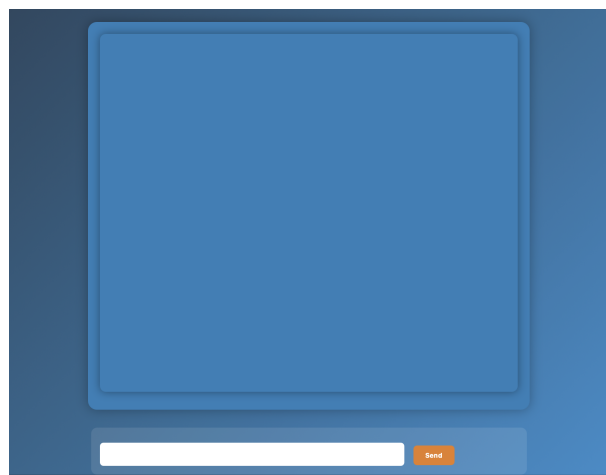
2.2.2 client -> serveur

- askRooms
demande l'état des salles pour pouvoir les afficher (alors updateRooms est retourné par le serveur)
- move
est envoyé lorsqu'un coup est joué, il contient alors l'id de la colonne jouée (entre 0 et 6)
- chatMessage
envoi un message dans le chat, il contient le contenu du message
- joinRoom
pour rejoindre une salle, il contient le nom du joueur et le numéro de la salle
- disconnect
automatique pas besoin d'être émis, notifie que le joueur a arrêté la liaison socket

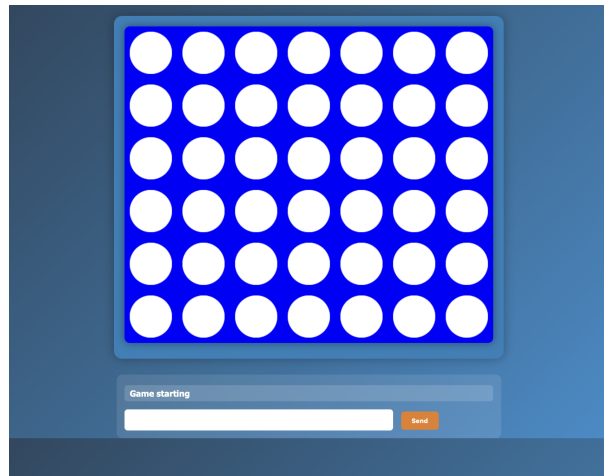
3 Exemple d'utilisation



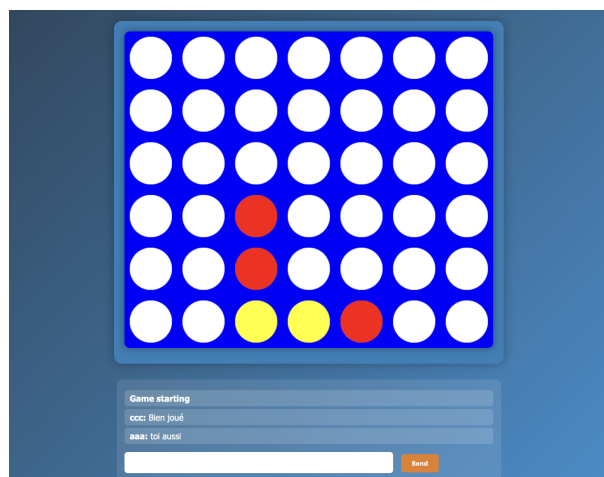
Ici la salle 0 a déjà deux joueurs qui s'affrontent, la salle 1 en a un qui attend son adversaire, les autres sont vides



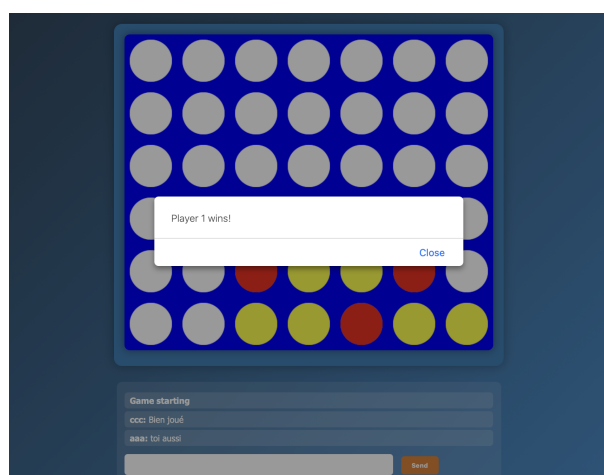
Dans la salle 1, le joueur est seul il ne se passe rien



Dans la salle 0, les deux joueurs commencent leur partie, ils peuvent communiquer dans le chat



Ils peuvent ajouter un pion chacun leur tour de leur couleur.



Jusqu'à la victoire