Projet tutoré

Piccadilly Game

Étude préalable

Matéo Achterfeld
Tom Georgelin
Guillaume Keller
Martin Jossic

IUT Nancy-Charlemagne 2019-2020

Matéo Achterfeld - Tom Georgelin - Guillaume Keller - Martin Jossic
2019-2020

Table des matières :

Présentation du projet : 2

Liste des fonctionnalités : 2

Diagramme de cas d'utilisations : 3

Serveur et client 3

Diagrammes d'activité : 4

Point de vue du serveur : déroulement du jeu 4

Point de vue du serveur: déroulement du jeu 5

Point de vue du client : Avant la partie 6

Point de vue du client : Pendant la partie 7

Diagramme de séquence : 8

Interaction client serveur lors d'une partie 8

Recensement et évaluation des risques 10

Latence durant l'application : 10

Développement des cas d'utilisation développés 11

Maquettes de l'application : 12

Page d'accueil : 12

Page de l'application : 13

Résumé des scores d'une partie : 14

Présentation du projet :

Piccadilly Game est un concept d'application web proposant une interaction massive avec le public. Le panel de possibilité est infini, passant de la simple application de sondage pour un grand nombre, au jeu multijoueur temporairement accessible. Le principe est de proposer aux utilisateurs un support commun d'affichage (écran publicitaire, écran d'ordinateur, installation robotique, projection…) qui sera interactif au travers du périphérique de l'utilisateur (téléphone, tablette, ordinateur…). Ce dernier aura une vision personnalisée de l'application tandis que le support central sera un affichage global de l'état du jeu.

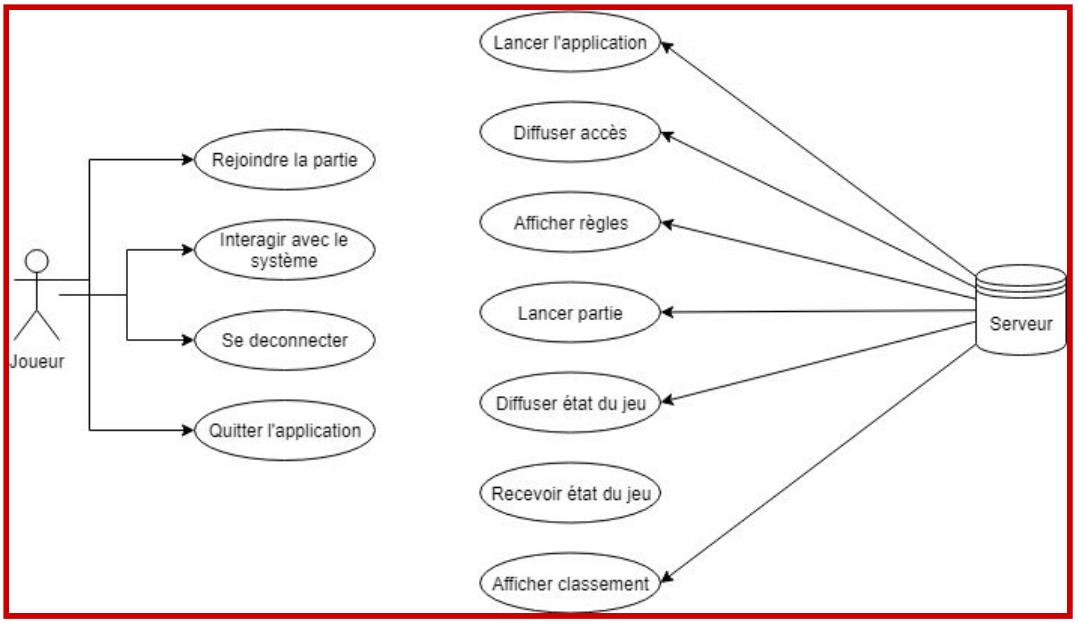
Ce projet sera basé sur une architecture client-serveur, pour avoir une vision globale et clair de notre jeu, on va présenter différents diagrammes.

Liste des fonctionnalités :

- Fonctionnement du jeu en temps réel comme si tout le monde jouait sur une même "console de jeu" (le serveur peut envoyer des données aux clients sans que celui ci ait à envoyer de requêtes d'actualisation)
- Connexion rapide grâce à un QR Code (pas de lien long à taper)
- Retransmission de l'état global de la partie sur un grand écran
- Adaptation aux connexions lentes (fonctionnalité à risque)
- Diffusion du score sur un site internet (pour que les joueurs puissent garder un "souvenir" de la partie)
- Différents jeux possibles : questions pour un champion, jeu de voiture, jeux de réflexe et rapidité...

Diagramme de cas d'utilisations :

Serveur et client



Dans ce diagramme, on voit que l'utilisateur doit pouvoir rejoindre une partie, celui-ci sera complété par la suite avec un diagramme d'activité permettant de se repérer de manière temporelle.

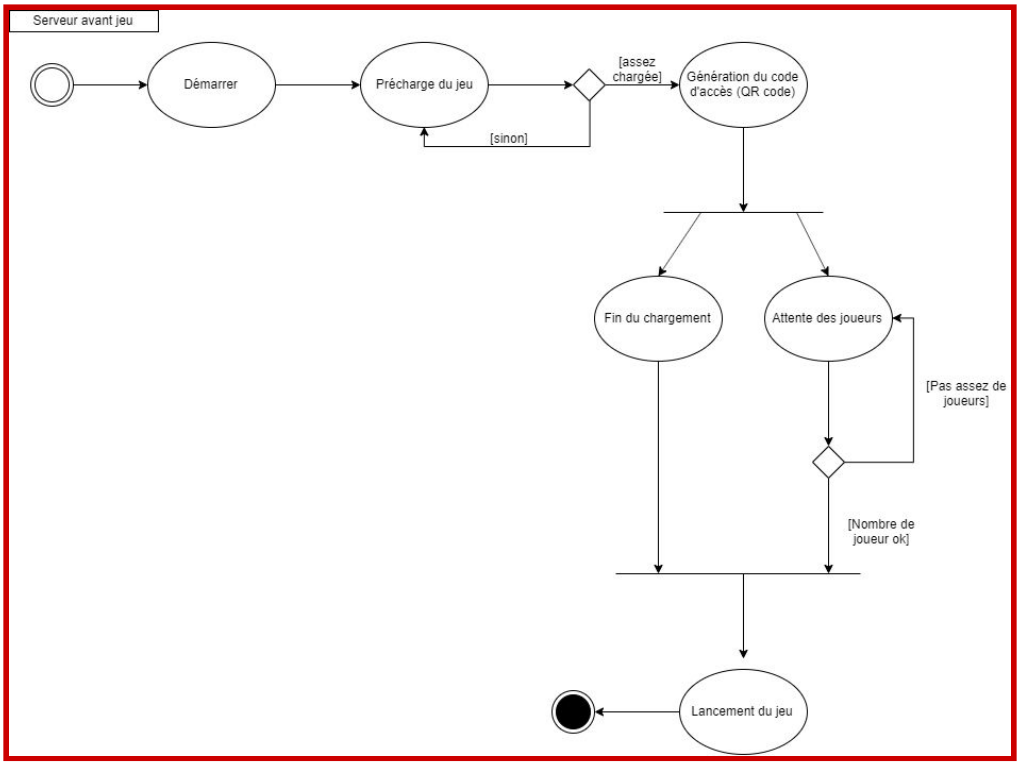
Nous y décrirons par la suite au travers de différents diagrammes les cas d'utilisations suivants :

- Coté client :
  - Déroulement du jeu

- Coté serveur :
  - Mise en place de la partie
  - Déroulement du jeu

Diagrammes d'activité :

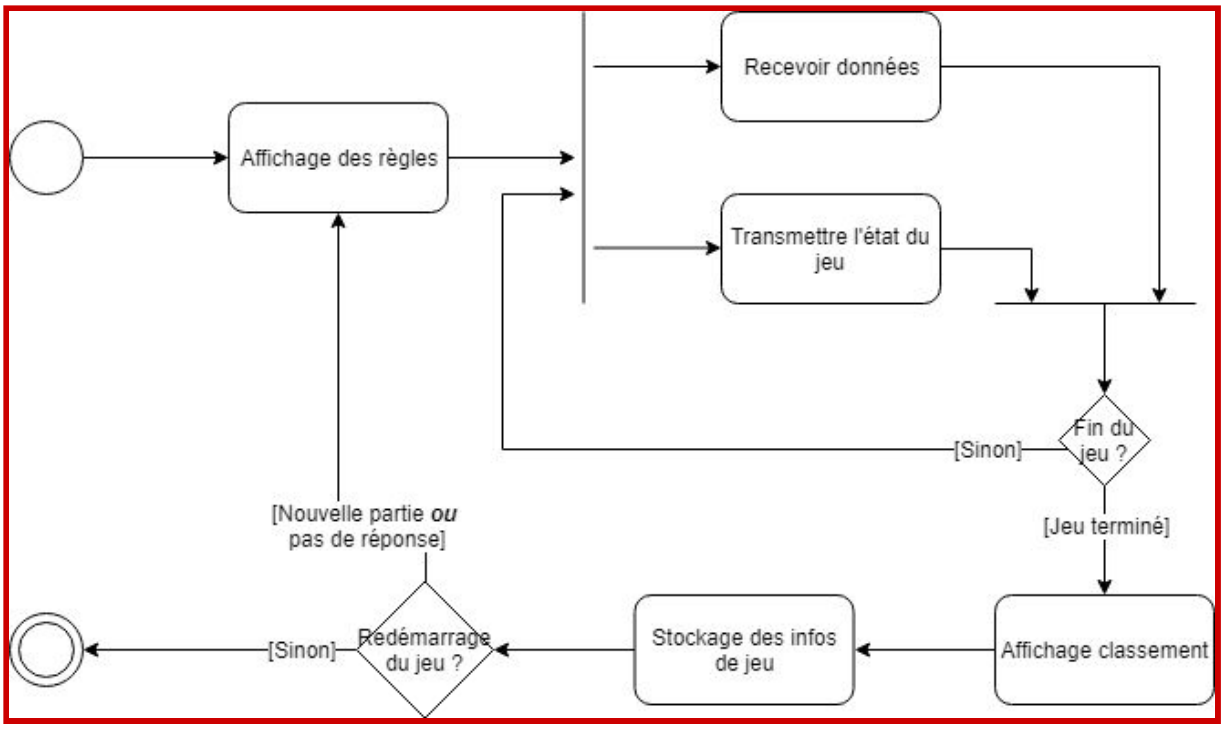
Point de vue du serveur : déroulement du jeu



Le serveur démarre puis s'occupe de précharger le jeu, il effectue cette tâche tant qu'il n'est pas assez chargé. Par la suite, il diffuse sur l'écran de diffusion le QR Code.

Le jeu va effectuer en parallèle le chargement (création) de l'application, ainsi que l'attente des joueurs, tant qu'il n'y a pas assez de joueurs connectés. Une fois ces opérations effectuées le jeu peut être lancé.

Point de vue du serveur: déroulement du jeu



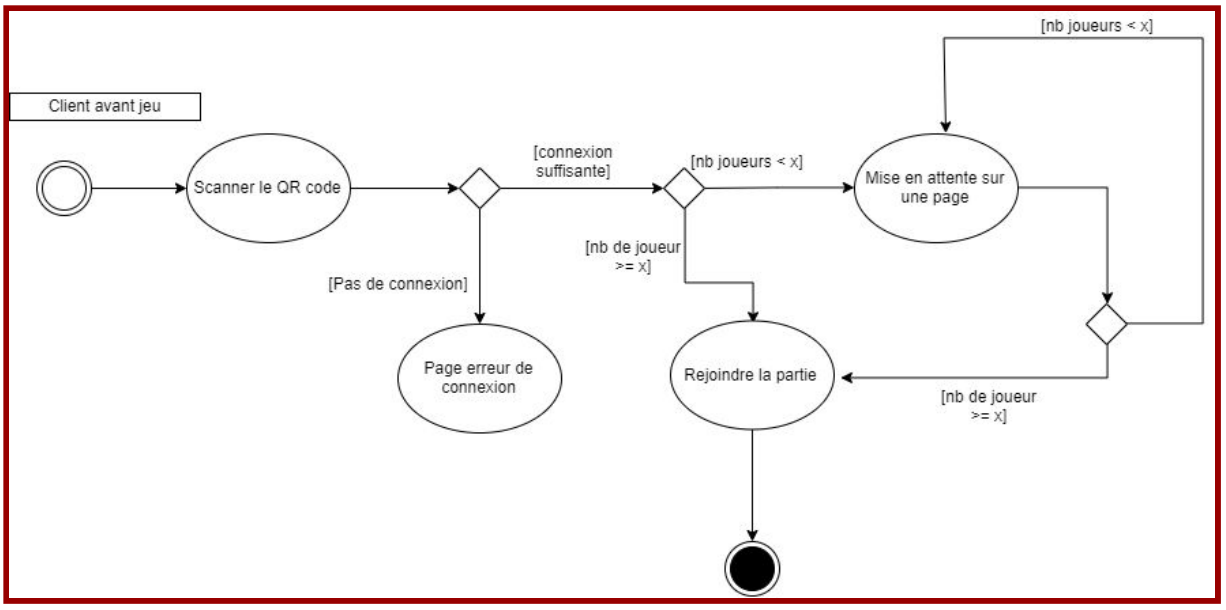
Une fois l'application lancée, le serveur s'occupe de diffuser les règles, attends un certain laps de temps, puis la partie démarre.

La boucle de jeu est alors lancée, avec une réception constante des données des joueurs puis une retransmission de ce qui a été reçu vers l'ensemble des joueurs afin d'actualiser l'intégralité des clients (y compris l'écran de diffusion).

Une fois l'application terminée, le classement est diffusé aux joueurs par leur propre terminal ainsi que par l'écran général. Les résultats seront consultables après la partie grâce à un lien donné à chaque utilisateur.

Après la consultation des résultats, la possibilité est donné à l'administrateur de l'application de relancer le jeu, en cas d'une attente trop longue de celui-ci, l'application est relancée par défaut avec malgré tout une possibilité de la couper à tout moment.

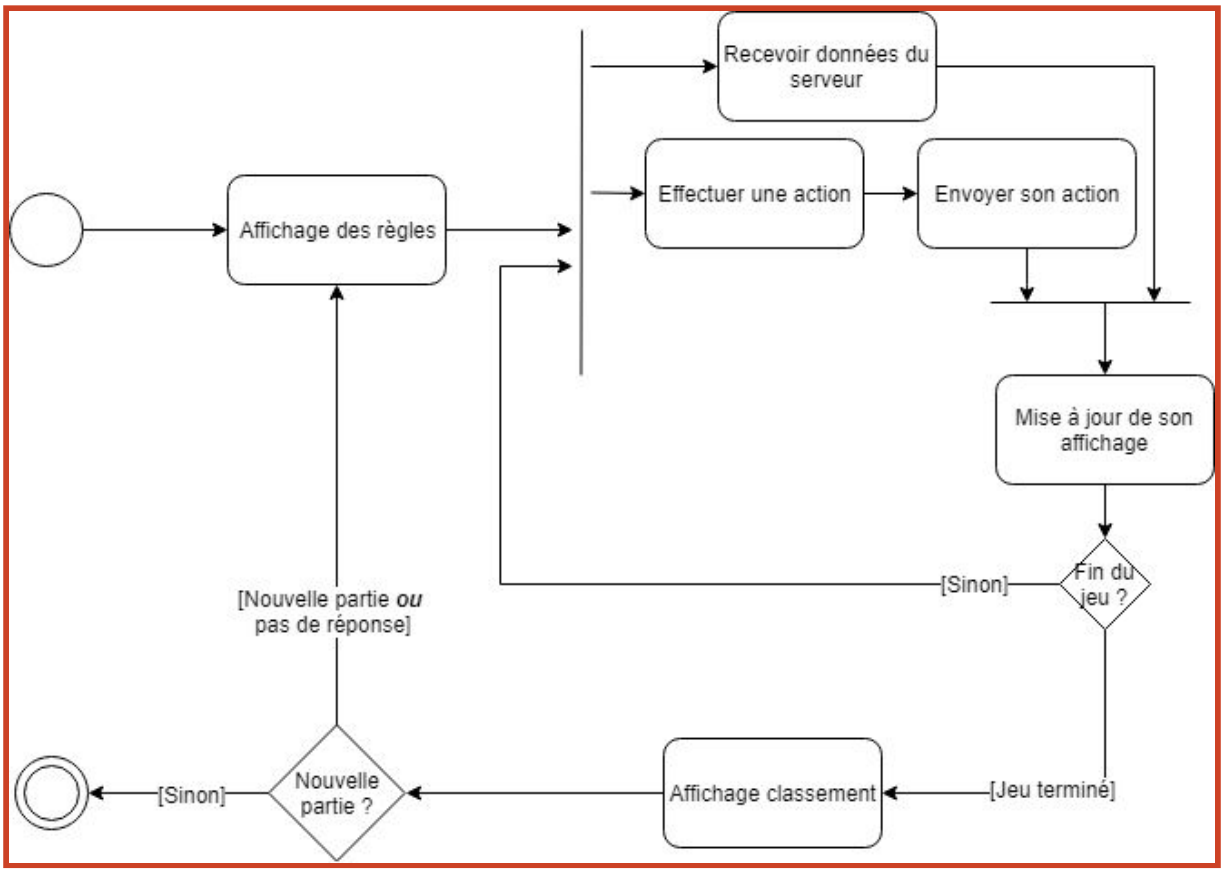
Point de vue du client : Avant la partie



Une partie débute par le scan d'un QR Code. Si la connexion du joueur est suffisante, il est mis en attente sur une page dédiée avant la partie.

Commence alors une boucle qui va attendre que le nombre minimum de joueur soit atteint ou que le temps soit écoulé. Alors, la partie commence.

Point de vue du client : Pendant la partie

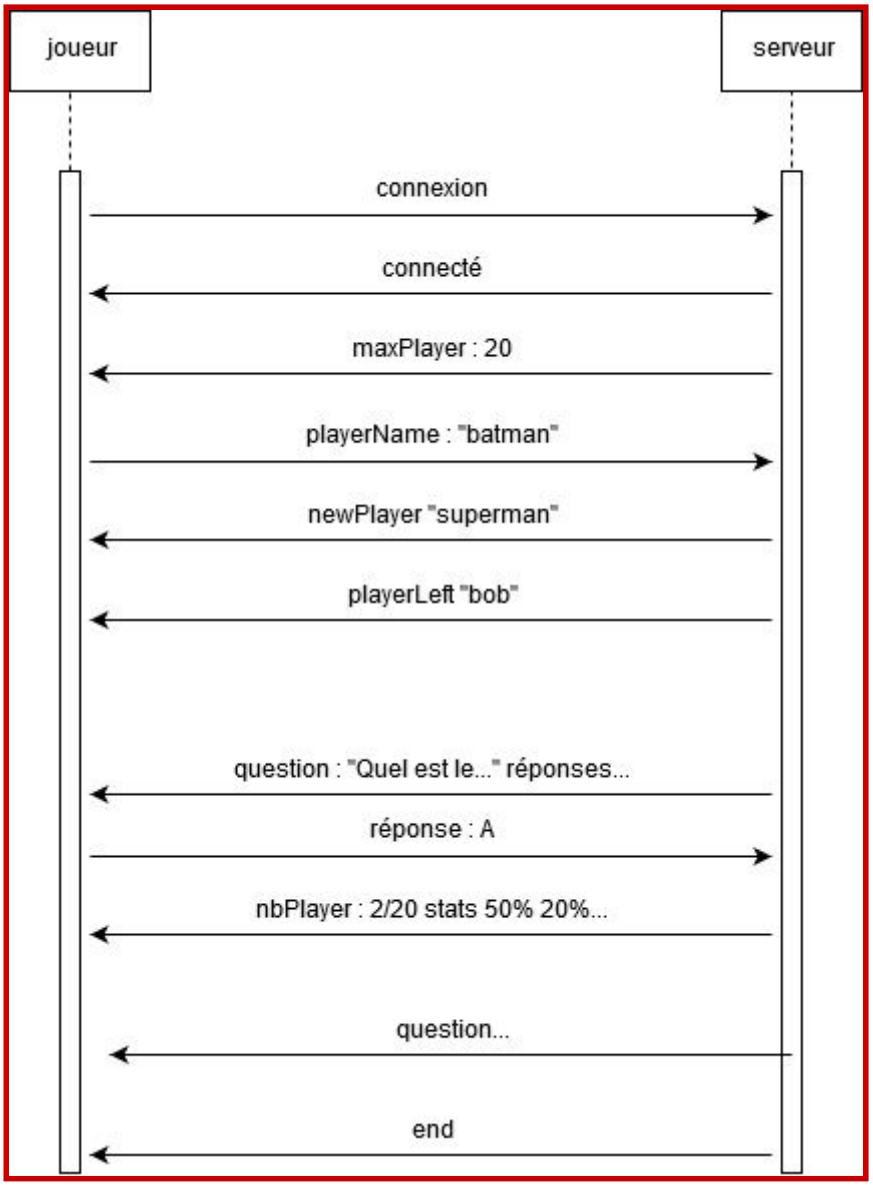


Durant une partie, le joueur est d'abord informé des règles avec lesquelles il va jouer. Par la suite, la boucle de jeu démarre. A chaque tour de boucle, le joueur recevra les données que lui envois le serveur pour mettre à jour son affichage spécifique, et il enverra l'action effectuée durant son tour de jeu.

A la fin de la partie, le joueur a accès à son classement, puis une nouvelle partie peut être lancée

Diagramme de séquence :

Interaction client serveur lors d'une partie



Ce diagramme montre les messages circulant entre le client et le serveur pendant une partie avec un jeu de type QCM (comme dans l'émission "Qui veut gagner des millions"). Une série de question est préparée sur le serveur.

Les joueurs ont un temps limité pour répondre à chaque question en choisissant la réponse A, B, C… Une fois qu'un joueur a répondu à une question, il se met en attente de la question suivante.

Pendant ce temps, il reçoit en temps réel les statistiques de réponse et le nombre de joueurs ayant répondu à la question.
Sur chaque réponse, il voit le taux de clic en pourcentage (50% d'utilisateurs ont répondu B par exemple). Quand le temps de réponse est écoulé ou que tous les utilisateurs ont répondu à la question avant la fin du temps, on passe à la question suivante.

Explication de la séquence de communication

- "***connexion***" et "***connecté***" : étape de connexion WebSocket.

- "***minPlayer***" et "***maxPlayer***" : avant que la partie se lance, le joueur est mis dans une "salle d'attente" où il voit le nombre de joueurs ayant rejoint la partie, le message "***minPlayer***" permet simplement au client de connaître le nombre de joueur minimal pour afficher sur la page "encore x joueur manquant". Le message "maxPlayer" permet de connaître le nombre maximum de joueurs dans la partie et d'afficher "x/20 joueurs connectés".

- "***playerName***" est envoyé au serveur pour indiquer le pseudo choisi par le joueur.

- "***newPlayer***" est reçu quand un nouveau joueur rejoint la partie. On donne avec ce message le pseudo du joueur s'étant connecté. Le client maintient en local la liste des joueurs connectés et en déduit le nombre de joueurs présents à afficher dans "x/20 joueurs connectés".

- "***playerLeft***" est reçu quand un joueur a quitté la partie. On reçoit avec ce message le nom du joueur s'étant déconnecté pour que le client l'efface de sa liste de joueurs interne.

- "***question***" est reçu quand une nouvelle question est lancée et est accompagné des réponses possibles à la question parmi lesquels l'utilisateur fera son choix. Ce message met fin à la question précédente (il peut être envoyé avant la fin du temps si tous les joueurs ont répondu avant l'échéance). Quand on est en "salle d'attente", ce message lance aussi la partie en plus de donner la première question (avec les réponses possibles).

- **"*réponse*"** est envoyé pour indiquer au serveur la réponse choisie par l'utilisateur (A, B, C…). Une fois qu'il a répondu, l'utilisateur ne peut pas modifier sa réponse.

- "***nbPlayer…***" contient les données sur les réponses données par les autres joueurs à savoir le nombre de joueurs ayant répondu et combien de joueurs (en pourcentage) ont cliqué sur les réponses A, B, C... Ces données ne sont reçues qu'une fois que l'utilisateur a lui-même répondu à la question en cours.

On retourne à "***question***" jusqu'à qu'il n'y ait plus de questions.

- "***end***" est reçu quand toutes les questions ont été jouées. La partie est alors terminée.

Recensement et évaluation des risques

Connexion des utilisateurs :

Des utilisateurs avec des connexions différentes doivent avoir accès au jeu sans qu'on ait de problème.

Surcharge du serveur :

Le serveur doit être proportionné à la charge qu'on lui impose (Limitation du nombre de joueur).

Latence durant l'application :

Si le serveur éprouve des difficultés à répondre à temps, il faut prévoir une méthode pour diminuer ce temps.

Risque abordé durant la première itération :

Le point "risque" développé durant la première itération sera la connexion des utilisateurs, l'application ne traitant pas encore une utilisation du temps réel très indispensable, ce point fera parti de la suite des itérations.

Développement des cas d'utilisation développés

Résumé des cas d'utilisations

(Utilisateur)	***(Serveur)***
Rejoindre la partie Interagir avec le système Se déconnecter Quitter l'application	Lancer l'application Diffuser l'application Afficher les règles Lancer la partie Diffuser l'état du jeu Recevoir l'état du jeu Afficher classement

Au cours de la première itération, nous développerons les fonctionnalités suivantes :

Martin :

Serveur :
- Diffusion de l'application
- Réception de l'état du jeu
- Diffusion de l'état du jeu

Utilisateur :
- -

Matéo:

Serveur :
- Lancer l'application

Utilisateur :
- Se déconnecter
- Quitter l'application

Tom :

Serveur :
- Afficher les règles
- Lancer la partie
- Afficher classement

Utilisateur :
- -

Guillaume :

Serveur :

- -

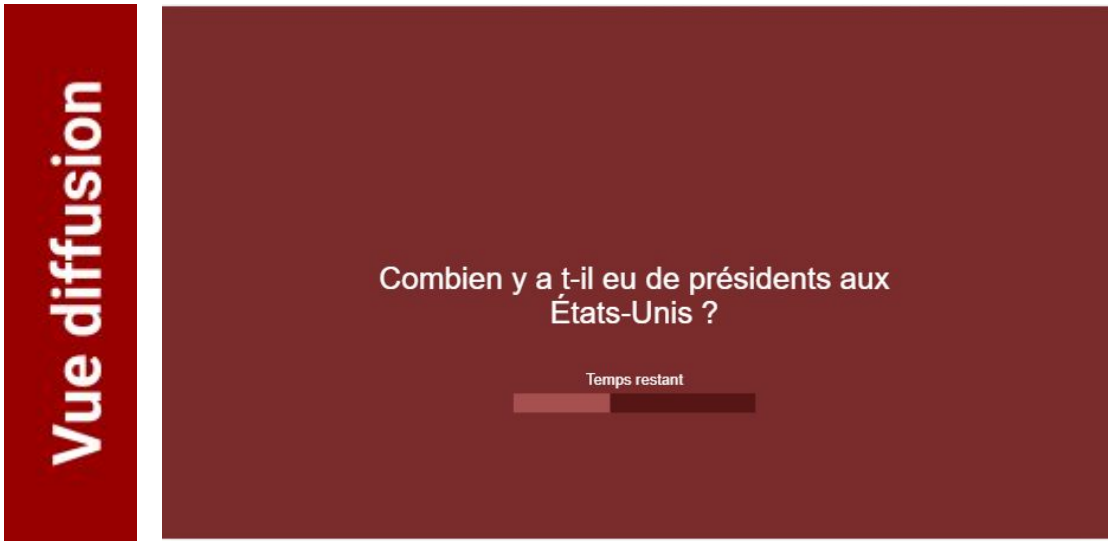
Utilisateur :

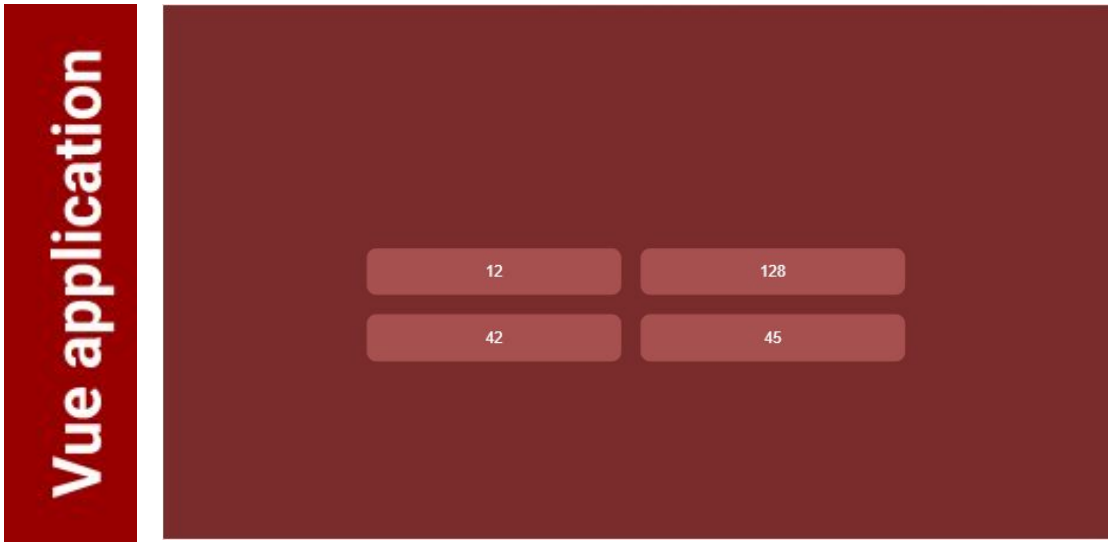
- Rejoindre la partie
- Interagir avec le système

Maquettes de l'application :

Page d'accueil :

Page de l'application :





Résumé des scores d'une partie :

Vue diffusion

Vue application