

Mini Projet – Quiz multi-utilisateurs

L'objectif de ce projet est de réaliser la version multi-utilisateurs d'un jeu de quiz dont les sources vous sont fournies. Vous devrez vous appuyer sur la librairie React pour la partie cliente.

Principe du jeu :

Le jeu consiste, pour chaque joueur à répondre correctement aux questions affichées en choisissant une réponse parmi celles-proposées.

Au début du jeu, chaque utilisateur doit saisir un pseudo (unique) pour jouer. Un compteur indique le nombre de joueurs connectés et une table affiche la liste les joueurs connectés.

Un bouton permet de démarrer le jeu pour l'ensemble des joueurs connectés (afficher le quiz).

Lorsqu'un joueur choisit une réponse il voit les réponses des autres joueurs pour la même question (si ceux-ci ont répondu). Lorsque tous les joueurs ont répondu à une même question, il devient impossible de la modifier et la réponse correcte est affichée. La figure ci-dessous présente les différentes possibilités :

Pseudo : Léon-Claude Nombre de joueurs : 3

Quelle est la particularité du sphénodon, un reptile de Nouvelle Zélande ?

2/3 ☒ Il a trois pattes arrières

1/3 ☐ Il a un troisième oeil

0/3 ☐ Il peut planer plus de 10 secondes

De quelle nationalité est Lou Bega qui connut un grand succès avec Mambo Number 5 en 1999 ?

0/3 ☐ Allemande

2/3 ☒ Anglaise

0/3 ☐ Espagnole

Que craint un coulrophobe ?

? ☐ une couleur

? ☐ un clown

? ☐ une couleuvre

Figure 1 : En haut, question terminée par tous les joueurs., Au milieu, question répondu par le joueur mais toujours modifiable car certains joueurs n'ont pas répondu. En bas, question non répondue par le joueur.

A la fin du jeu, c'est à dire quand tous les joueurs ont répondu à toutes les questions, les scores sont affichés dans la table des joueurs.

Pseudo	Score
Léon-Claude	1
Isa-Belle	2
Lou-Zer	0

Figure 2 : Table des score à la fin de la partie.

Travail attendu :

Vous devez ajouter les fonctionnalités suivantes :

- Demander un pseudo (unique) au joueur
- Afficher combien d'autres joueurs sont connectés dans un compteur et quels sont leurs pseudo dans une table.
- Lancer la partie lorsqu'un des joueurs clique sur le bouton démarrer la partie.
- Afficher leurs réponses aux différentes questions pour respecter le principe du jeu.
- Calculer les scores et identifier le gagnant à la fin du jeu

Il n'est pas nécessaire de gérer les cas où un ou plusieurs joueurs s'ajoutent ou se retirent en cours de partie mais il faut le prendre en compte avant le début de la partie.

Contraintes techniques :

Le projet sera réalisé avec des technologies JavaScript et en particulier NodeJS associé aux Framework Express et Socket.IO pour la partie côté serveur. Pour le code côté client vous devez utiliser React et les composants Bootstrap.

Code fourni :

L'archive contient du code pour démarrer le projet. Le code est constitué de deux dossier :

- **server** : contenant l'ensemble des sources pour le serveur (dont les dépendances et des cripts de lancement sont données dans le fichier package.json). En particulier Vous trouverez un module nommé **kwiz_module** contenant plusieurs fonctionnalités pour récupérer les questions, gérer les clients et leurs réponses. Vous n'êtes pas obligé de l'utiliser mais cela peut accélérer votre travail.
- **client** : contenant l'ensemble des sources pour le client React (dont les dépendances et des cripts de lancement sont données dans le fichier package.json).

Évaluation des fonctionnalités /15 :

Voici un barème utilisé pour noter le projet :

- Demander un pseudo au joueur et l'afficher / 2
- Afficher le nombre de joueurs connectés / 1
- Afficher la liste de tous les clients connectés dans une table / 2

Lancer le jeu lorsqu'un des joueurs clique sur démarrer / 1

- Afficher le nombre de fois qu'une réponse a été choisie par les autres joueurs lorsque le joueur choisi une réponse / 4
- Afficher la solution d'une question lorsque tous les joueurs ont répondu à cette question / 2

- Empêcher la modification lorsque tous les joueurs ont répondu à une question / 1
- Afficher les scores et le gagnant dans la table des joueurs à la fin de la partie / 2

Qualité du code /5	Excellent	ok	Nok
Html/CSS/JS Bootstrap et Nodejs	Le code est bien structuré et utilise correctement les fonctionnalités du serveur et du client. Le code est modulaire, factorisé et pertinemment commenté.	Le code est globalement bien structuré et utilise les fonctionnalités du serveur et du client mais avec quelques erreurs. Il y a quelques parties ad-hoc et/ou peu de commentaires pertinents.	Le code n'est pas bien organisé et les fonctionnalités du client et du serveur sont mal ou pas exploitées. Le code est très ad-hoc et peu ou mal commenté.

Conseils :

Il est important de bien concevoir les échanges entre le serveur et les clients. Il vous faut donc définir un protocole de messages simples pour les évènements. Par exemple, (*'nouveau_client', nom du client*) ou encore (*nom_clients_connectés, {'clients': ['pseudo1', 'pseudo2', 'pseudo3']}*).

Pour identifier et stocker les données provenant des clients sur le serveur, il est pratique de créer un objet qui stocke en mémoire l'ID (unique) du client attribué par la socket. Cet ID s'obtient grâce à la méthode *var clientID = socket.id* ; Le module kwiz fournit est conçu pour fonctionner avec ce principe.

Améliorations possibles :

Utilisez une animation lors du changement du nombre de joueur +0.5

Vos propositions (ajout de sons, nouvelles règles) à discuter. + ?