Mini Projet – Quiz multi-utilisateurs

L'objectif de ce projet est de réaliser la version multi-utilisateurs d'un quiz. Vous devez ajouter des fonctionnalités qui permettent de savoir combien d'autres joueurs sont connectés, quels sont leurs pseudos et quelles sont leurs réponses aux différentes questions. Le jeu sera réalisé avec des technologies JavaScript et en particulier NodeJS associé aux Framework Express et Socket.IO pour la partie serveur.

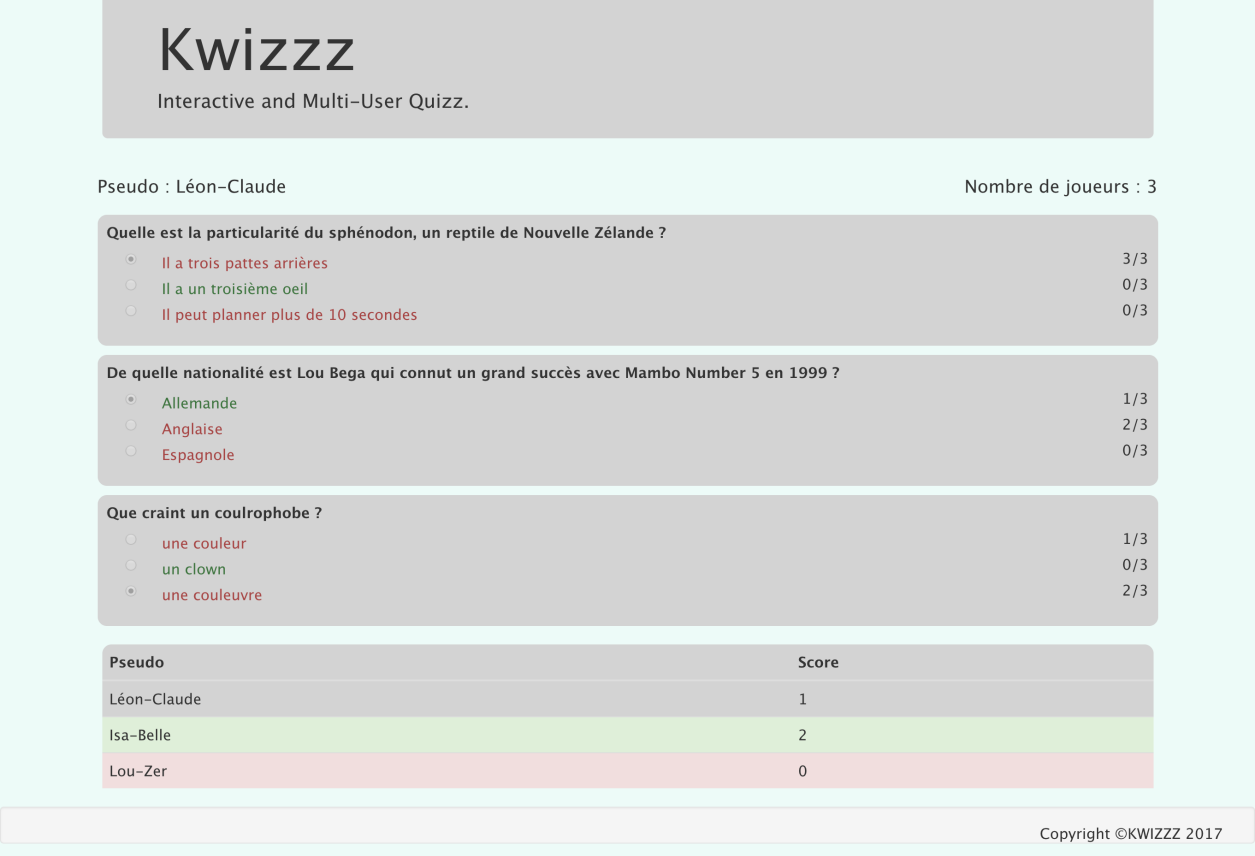

Figure 1 : Capture d'écran du jeu. Un compteur indique le nombre de joueurs connectés. Une fois qu'une réponse a été choisie par tous les joueurs, la bonne réponse s'affiche en vert et les mauvaises en rouge. Sur la droite apparaissent les réponses des autres joueurs. Celles-ci ne sont visibles que pour les questions auxquelles le joueur à déjà répondu.

Code fourni :

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

- package.json avec une description basique du projet et des dépendances
- server.js avec du code permettant d'afficher une page web contenant le quiz
- /kwiz_module qui contient des fonctionnalités pour récupérer les questions, gérer les clients et leurs réponses.
- /public contenant les sources des fichiers de l'interface du client (html / css / javascript)

Travail attendu /15 :
Vous devez réaliser une version multijoueur du code fourni. On considèrera que tous les joueurs doivent commencer la partie en même temps. Il n'est pas nécessaire de gérer les cas où un ou plusieurs joueurs s'ajoutent ou se retirent en cours de partie.

En particulier, voici la liste des fonctionnalités à réaliser :

- Demander un pseudo au joueur et l'afficher / 2
- Afficher le nombre de joueurs connectés / 2
- Afficher la liste de tous les clients connectés dans une table / 4
- 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

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','pseudi3']})*.
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 :
Ajouter un score aux joueurs, l'afficher sur la page et dans le tableau des joueurs +1
Demander un nouveau pseudo au joueur si celui-ci est déjà utilisé +0,5
Utilisez une animation lors du changement du nombre de joueur +0,5