# Technical Presentation

### Technologies utilisées

Notre projet étant une application web, il se décompose donc en deux : une partie serveur et une partie client. La partie serveur est entièrement écrite en NodeJS en utilisant le Framework Express, la partie cliente est gérée en Javascript. L’application utilise également une base de données SQL pour retenir les informations des utilisateurs et des parties, accessible par l’intermédiaire de PHPmyAdmin.

### La partie serveur/client

Le serveur est assez simple à mettre en place puisqu’il faut définir toutes les routes, c’est-à-dire les différentes adresses et les lier aux pages de rendu HTML. Il faut néanmoins plus de réflexion quant aux requêtes et données que l’on souhaite passer à chaque page, par exemple sur la page d’accueil : il faut chaque catégorie disponible ainsi que les points associés, dans le cas où le joueur à déjà testé le quizz correspondant.

Mise à part le routage, il faut gérer des sockets, ce sont des fonctions qui sont exécutées dynamiquement lorsqu’un joueur appuie sur un bouton ou réalise une action précise, par exemple appuyer sur « Play » génère une requête pour récupérer une première question à poser à l’utilisateur. Ces sockets sont spécifiques à chaque action et ils font le lien entre l’obtention de données et le rendu graphique, il faut donc faire les bonnes requêtes à la base de données selon le type de jeu et d’utilisateur.

### La base de données

La base de données est composée de 6 tables qui servent à sauvegarder de manière organisée les informations qui sont créées ; on a ainsi une table *user*, *question*, *category*, *level*, *match*, *game*. Ainsi lorsqu’un utilisateur s’enregistre sur l’application, la table user est remplie avec les informations qu’il nous fournit c’est-à-dire son pseudo (en vérifiant l’unicité de celui-ci) et son mot de passe. Pour des questions de sécurité, le mot de passe ne peut pas être stocké en clair dans la base, il nous a donc fallu trouver une solution pour le « hasher ».

C’est à partir d’un problème comme celui-là que l’on peut voir l’avantage d’utiliser un langage comme NodeJS. Il propose des modules à inclure qui permettent d’utiliser des fonctions pré-écrites par la communauté, ainsi nous avons utilisé un module « Password-hash » qui fait lui-même les transformations nécessaires avant l’insertion dans la base de données.

Le code est disponible à cette adresse : <https://github.com/Qu1lby/BigBen>

The main goal of this game is to get the more points and be on the podium!

**Points:** The points system is quite simple: for each category, a user can get a number of points depending on the level of questions he succeeded to access. For example, if the player failed to answer question 4, he will get an amount of points corresponding to this level. In addition, and in order to increase the challenge between players, there is some random points which are added to the score.

If a player wants more points on a category, he can play again and hope to win more random points but he can also decrease his score! In fact, if he succeeded the 10 levels and try again but this time can't reach all the questions, his score will not be the highest but the last one, in this example he will get less points than before the second game. That’s the trick!

You can see the podium (the top ten of the best player in BigBen) in live on the dedicated page reachable from the home page.