ALBANET Xavier
RUSSO-PELOSI Louis
LABOUDI Hakim

classe FSI1

Rapport de Projet Programmation Web : 4 images

Membres du groupe de projet :

- ALBANET Xavier
- RUSSO-PELOSI Louis
- LABOUDI Hakim

Liens du Projet :

- url du site : https://xcuby-first-vue-app.glitch.me
- lien GitHub du projet : https://github.com/Xcuby/first-vue-app

Table des matières

ALBANET Xavier
RUSSO-PELOSI Louis
LABOUDI Hakim

I. Introduction

Les cours de programmation web avancés de 4ème année, venant approfondir et compléter les cours de programmation web de 3ème année, ont cette année pour finalité la création d'un site internet sous forme de projet. Ce projet se devait de respecter le cahier des charges suivant :

- Web interface built with HTML, CSS and Vue.js
- Web server built with Node.js and Express
- User authentication (and registration if your application requires it)
- CRUD interactions with the server
- Data persistence is optional: in-memory is good enough, but you can try to write to the file system (no database!)
- Deployed on Google Cloud Platform or Glitch

II. Le site : 4 images

a. Présentation du site

Notre site internet se présente sous forme de Quizz et se nomme 4 Images et permet d'évaluer le QI de l'utilisateur. Cependant ce n'est pas un réel test de QI donc l'évaluation final est aucun cas à prendre au sérieux.

La page d'accueil du site est simple, épuré et donne deux choix à l'utilisateur : se connecter ou s'inscrire. Si vous n'avez pas encore joué il faudra vous créer un compte. Sinon vous n'avez qu'à vous connecter.

Après vous êtes connecté vous arriverez sur une page de présentation et n'avez qu'à cliquer sur Accéder à mon profil en bas de la page pour arriver sur votre page Profile.

Arrivé sur votre page profile qui sera en quelque sorte un menu vous pourrez soit :

- Jouer : ce qui vous permettra ici soit de réaliser votre premier score ou alors de battre votre record
- Accéder à votre historique personnel : vous trouverez ici un tableau récapitulant l'ensemble des parties que vous avez réalisé sur votre profile avec le score obtenue.
- Accéder à au classement global des joueurs : vous trouverez ici le classement de tous les joueurs en fonctions de leur record

Lorsque vous jouez vous avez deux types de questions :

- Soit vous devez cliquer sur l'image correspondant à l'énoncé (ex : Quel est l'intrus)

- Soit vous devez entrer dans le champ de texte (qui apparaîtra si vous faites face à une question de ce type) la réponse correspondante à l'énoncé en fonction des images (ex : Quel est le point commun)

A noter que cela sera toujours spécifier à la fin de l'énoncer par « cliquer directement sur l'image correspondante » ou « entrez votre réponse dans le champ de texte »

b. Identifiants du site

Comme énoncé précédemment, il est tout à fait possible de se créer un compte avec l'identifiant et mot de passe de son choix afin de se connecter. Cependant, un compte administrateur existe déjà et est utilisable si vous n'avez pas envie de créer un compte. De plus, ce compte administrateur a le meilleur score possible 0 faute et un QI maximal! Les identifiant de ce compte sont :

- Identifiant : admin
- Mot de passe : Xav

III. Description du projet et difficultés rencontrées

a. Réponse aux contraintes demandées

Tout au long du projet, nous nous sommes efforcés de respecter le cahier des charges (qui a été rappelé dans l'introduction). Ainsi nous allons reprendre point par point ces exigences, il y a donc :

- ***Web interface built with HTML, CSS and Vue.js*** : Pour la création de l'interface Web, nous avons bien utilisé du HTML, CSS et Vue.js tout au long du projet et aucune autre.
- ***Web server built with Node.js and Express*** : Pour la création du serveur Web, nous avons bien utilisé Node.js et Express tout au long du projet et aucun autre.
- ***User authentication (and registration if your application requires it)*** : Nous avons bien mis en place un système d'authentification et d'inscription (par le biais de requête http vers notre serveur Web)
- ***CRUD interactions with the server*** : Effectivement nous avons bien réalisé des interactions :
    - de création avec le serveur, notamment à la création d'un nouvel utilisateur.
    - de lecture avec le serveur, notamment à la connexion d'un utilisateur déjà existant ou même à la vérification de la réponse choisit par l'utilisateur (car les réponses sont stockées sur le serveur.

  - de mise à jour avec le serveur, notamment pour mettre à jour le meilleur score de l'utilisateur sur le serveur.
  - de suppression avec le serveur, notamment pour supprimer le rang de chacun dans le classement global juste avant de le mettre à jour.
- ***Data persistence is optional : in-memory is good enough, but you can try to write to the file system (no database!)*** : Nous avons choisi de stocker des données en mémoire (cache) afin de les réutiliser sans avoir à réitérer le même appel serveur dans un autre fichier .vue. Nous avons notamment stocké de cette façon l'identifiant de l'utilisateur.
- ***Deployed on Google Cloud Platform or Glitch*** : Nous avons en effet déployé notre projet sur Glitch et notre site est actuellement en ligne, on rappelle que son url est : https://xcuby-first-vue-app.glitch.me

b. Etapes de conception, réalisation et déploiement

La première étape a été la prise en main de Vue.js et de Vuetify grâce aux cours qui nous ont été donné ou même grâce à différents tutoriels ou forums.

La seconde étape a été de concevoir notre quizz en stockant dans un premier temps les questions, réponses et résultats de QI dans un tableau à deux dimensions dans notre fichier .vue. Ce qui nous a permis par la suite de réaliser l'ensemble de l'application sur notre fichier .vue (sans page de login).

Ensuite, nous avons eu besoin de mettre en place et de prendre en main notre serveur en JavaScript. La prise a été plutôt rapide puisque le serveur est également en JavaScript et pas en PHP ou autre. Il nous a seulement suffit de prendre en main la mise en place des requêtes http et leur utilisation dans les fichiers .vue (fonction asynchrone, syntaxe …)

Cela nous a donc permis de réaliser la page de login (inscription et connexion) et de passer au Quizz (en utilisant un booléen dans un « v-show ») si l'utilisateur est connecté.

Nous avons ensuite pu nous pencher sur le stockage des réponses sur le serveur. Pour cela, nous avons choisi de créer un tableau sur notre server.js contenant les réponses dans l'ordre. Si l'utilisateur clique sur la bonne image ou rentre la bonne réponse le serveur renverra en réponse « true » sinon il renverra « false ». Nous avons choisi la méthode de comparer la réponse utilisateur avec la réponse dans le fichier server.js puis de renvoyer un booléen en fonction de la véracité de la réponse plutôt que le server répond en envoyant une requête contenant la réponse que l'on comparera dans le fichier .vue car l'utilisateur aura accès à cette réponse en analysant les requêtes http et connaîtra donc la bonne réponse lorsqu'il fera de nouveau le test.

Nous avons ensuite choisi de rajouter le stockage du meilleur score de l'utilisateur sur le serveur afin de lui rappeler lors de sa connexion.
Par la suite, nous ensuite rajouté un tableau servant d'historique personnel pour chaque utilisateur.
Enfin, nous avons rajouté un classement global prenant tous les joueurs et les classant en

fonction de leur meilleur score. Pour cela à chaque fois qu'un utilisateur fait un nouveau meilleur score, on retrie notre classement (grâce à la fonction .sort()).

L'architecture a été ensuite notre principale préoccupation et nous avons donc suivi le tutoriel de vue.js sur « vue-router ». Ainsi, après avoir suivi et compris le tutoriel, nous avons décidé de scindé notre projet en 3 fichier vue :

- PageAccueille.vue : ce fichier s'occupe de toute l'interface d'inscription et de connexion
- PageProfile : ce fichier sert comme un menu où l'utilisateur voit son identifiant, son meilleur score et peut choisir de voir son classement global, son historique personnel ou il peut cliquer sur jouer pour réaliser le quizz.
- Quizz.vue : ce fichier est l'ensemble du quizz

Nous avons ensuite choisi de stocker le nom d'utilisateur en mémoire cache afin d'y accéder de nouveau dans tous les autres fichiers .vue. Par exemple, on réutilise le nom d'utilisateur pour lui souhaiter la bienvenue de façon personnalisée sur la page de profile (qui est une autre page .vue) sans avoir à récupérer le nom d'un utilisateur à l'aide d'une autre requête GET.

Nous avons ensuite déployé notre projet sur Glitch en réalisant les prérequis nécessaire et notre site a été correctement mis en ligne sans problèmes.

c. Problèmes rencontrés

Nous avons perdu beaucoup de temps à comprendre et utiliser les tutoriels car l'utilisation d'un Framework était une nouvelle chose pour nous. Nous avons ensuite perdu beaucoup de temps à mettre en places nos connaissances même si les extensions ESLint et Vetur de Visual Studio Code nous ont bien aidé à nous corriger sur la synthaxe.

Un autre problème est que pendant plus d'une semaine nous n'arrivions pas à faire des requêtes HTTP d'un fichier .vue à notre server.js. Celle-ci nous renvoyé en effet toujours des erreurs de requêtes (notamment code 500). Enfin après avoir consulté de nombreux forums (surtout StackOverflow), nous avons réussis à nous corriger et utiliser GET et POST en fonction de leur nécessité, de correctement rédiger un envoie de requête grâce à axios, d'utiliser les fonctions asynchrones et de d'envoyer des réponses du serveur utile et utilisable.

L'architecture (routage) a également été une grosse difficulté pour nous puisque nous n'arrivions pas à appliquer le tutoriel à notre projet puisque le tutoriel était expliqué pour Vue.js or nous avions aussi Vuetify. Ce qui changé tout au niveau de la syntaxe…
Finalement nous avons réussi mais un problème persisté, même après avoir rajouté une meta permettant l'accès à des pages que si l'utilisateur est connecté, nous n'arrivions pas l'appliquer à nos pages. Nous avons réussi en bloquant la page et en mettant un message d'alerte à l'utilisateur lorsque celui-ci essaie d'accéder à une page nécessitant la connexion (en tapant directement l'url correspondant) alors que celui-ci ne l'est pas.

IV. Conclusion

Pour conclure, je dirai que ce projet nous été grandement bénéfique techniquement mais pas seulement. En effet, nous avions été grandement déçus des cours de programmation de Web de l'année dernière (même si ce n'était qu'une introduction). Mais cette année nous avons été grandement rassurés sur notre faisabilité à faire des sites internet, notamment grâce à la découverte de ce qu'est un Framework et son utilisation. De plus, notre gestion de projet sur Github (utilisation complète des branches) a grandement été amélioré et prend maintint son sens pour nous car avant, nous interprétions cela comme une perte de temps.

Toute notre équipe remercie donc Mr.CHEREL pour ces cours et l'opportunité d'apprentissage et praticité qu'a été ce projet pour nous.