RAPPORT DE PROGRAMMATION WEB

4A CFA

BOURGAIN Rudy

CLAUSSE Klyve

2018

SOMMAIRE

ACCÈS AU SITE

DESCRIPTION DU PROJET

Le projet consiste à créer un site web faisant office d'interface de visualisation de notes entre professeurs et élèves. Cette interface web a été développée en HTML/CSS avec l'utilisation de la technologie Vue.js pour la partie front-end et le serveur web stockant les données en PostgreSQL avec l'utilisation de Node.js et Express.

L'interface permet la création et modification de sessions professeurs pouvant modifier les notes des élèves dans différentes matières et de sessions élèves pouvant uniquement observer leurs résultats. Les données des sessions sont stockées sur un serveur web dans une VM.

ÉTAPES DE CONCEPTION

En premier lieu, nous avons réfléchi à la façon dont nous allions afficher les données aux élèves. Celles-ci sont disposées sous formes de tableaux répertoriant leur nom, leur classe, les matières et leurs notes. Rudy s'est principalement chargé de la partie back-end et Klyve de la partie front-end.

Pour l'interface, nous avons, d'abord créé un modèle de page admin pouvant gérer les classes, les utilisateurs et les matières.
Puis, nous avons lié les données renseignées par l'admin à un modèle de page élève via des fichiers json.
Nous nous sommes servis de Vue.component pour la création des différents paramètres comme les classes ou les utilisateurs.
Nous avons également créé plusieurs méthodes javascript pour implémenter différentes fonctionnalités.
Nous avons aussi eu recours à différentes fonctionnalités Vue.js. Par exemple, sur la page admin, v-on:click affecte à une variable la section de la page sur laquelle on

veut naviguer via un clic sur un bouton et v-if/v-else-if redirige l'utilisateur vers la section voulue.

Côté back-end, nous avons assurément géré l'authentification manuellement (hash et salage des mots de passes, vérification de l'authentification, gestion de cookie de session). Nous avons parfaitement conscience que cette pratique est vouée à l'échec tant au niveau de la sécurité que de la stabilité de l'application. En situation de production, nous opterons pour l'usage de librairies. Ce projet nous a, en revanche, permis de comprendre de manière plus approfondie le fonctionnement de la gestion de session ainsi que l'authentification sur les applications web.
Cette partie fut réellement formatrice pour nous, nous avons découvert à quel point la gestion de cet aspect est complexe.

Toujours côté back-end, nous avons tenu à respecter le schéma classique d'une application web moderne :

- Le routeur traite les requêtes SQL.
- Le contrôleur s'assure de la légitimité et la conformité de la requête.
- Les modèles traitent les données.
- La couche database interagit avec la base de données.

Nous avons donc commencé par gérer l'authentification. Ensuite nous avons commencé à créer des données persistantes. Cela fut d'abord sous forme de fichier JSON (le fichier "users.json" à la racine de notre dépôt en est un reliquat).

Malgré vos mises en garde, nous avons choisi de stocker nos données dans une base PostGreSQL. Le paragraphe ci-dessous a pour but de vous expliquer pourquoi.

Au vu du type d'application que nous avons conçu, les données saisies ont d'importantes interliaisons. Il nous apparaissait alors plus compliqué de faire des fichiers JSON qu'une base de données. A noter que des association complexes de fichiers JSON est une source importante de bugs que l'on a préféré éviter.

Une autre décision importante a été de choisir PostGreSQL au lieu de SQLite. Il est motivé par la présence de moteur transactionnel (COMMIT/ROLLBACK). Dans le cas de notre application, la cohérence des données enregistrées est primordiale. L'utilisation systématique du moteur transactionnel pour toute insertion ou modification de données a eu un coût assez important en temps de développement mais fait office de garde-fou face aux mauvaises requêtes SQL soumises à la base de données (et il y en a eu beaucoup durant le développement).

La mise en oeuvre d'une base de données PostGreSQL est donc une solution plus compliquée et coûteuse que des fichiers JSON. Cependant, elle nous a semblé nécessaire au vu de notre type d'application (bien que les bases de données SQL sont loin d'être la seule solution pour avoir des données persistantes). Ensuite, ensuite nous avons décidé de recourir à une base de données pour héberger les données.

DIFFICULTÉS RENCONTRÉES

L'une des plus grosses difficultés rencontrées est incontestablement la découverte de la programmation asynchrone. Comme vous pourrez le voir dans notre code côté back-end, tout cela n'est pas toujours géré idéalement. Nous avons complètement découvert ce concept (et donc les promesses). Par manque de temps, nous n'avons pas pu remettre tout cela au propre.

Qui dit programmation asynchrone dit beaucoup de bugs associés lorsque votre application n'était originellement pas prévue pour. Nous avons essayé de nous adapter au mieux et estimons avoir compris le principe des promesses (et des callbacks bien que nous ayons préféré directement programmer en promesses).

AMÉLIORATIONS FUTURES

Conformément au projet AGILE, nous cherchons dans un premier temps à mettre en oeuvre un produit répondant au strict minimum pour ensuite l'améliorer.
Nous sommes précisément rendus à cette phase. L'application répond au strict minimum :

- un administrateur peut créer et gérer des utilisateur (renommage, réinitialisation de mot de passe, gestion des profils (admin, prof, élève)...), des classes composées d'élèves et des matières associées à une classe.
- Les professeurs peuvent ensuite saisir les notes de leurs élèves (ils n'ont accès qu'à leurs matières respectives)

- Les élèves peuvent finalement consulter leurs notes (et ne voit évidemment que ses notes).

La suite du processus voudrait qu'on améliore cette application. Les axes sont assez nombreux, l'ergonomie de l'interface est perfectible. On peut aussi calculer la moyenne de chaque élève, la moyenne de classe…

Certains bugs sont également à corriger (le plus ennuyeux est la modification d'une note par un professeur, le problème semble être côté base de données). Nous avons également constaté en rédigeant ce rapport que l'application ne gérait pas deux connexions provenant de deux adresses IP différentes. Cela semble être dû à la gestion de session elle-même (et pourrait se corriger avec l'usage d'une librairie).

Seule certitude : il est plus facile de corriger des bugs que de re-traiter des données dont la cohérence a été altérée.

Accéder à notre site web

URL : http://139.59.147.233:3000
login : admin
mot de passe : admin

La liste des utilisateur est accessible dans la console d'administration. Leur mot de passe est généralement le même que le nom de compte et réinitialisable.

Si vous souhaitez avoir accès en SSH au serveur, merci de me communiquer votre clé RSA.

Il faut se déconnecter (menu > se déconnecter) pour changer d'utilisateur.