CookChef

BIRON Thomas & TCHASSEM Mike | Projet Web | 19/11/2018

Table des matières

Description du projet

CookChef est un site de recette de cuisine où chaque utilisateur a accès à de nombreuses recettes de dessert facilement.

Chaque utilisateur peut afficher le contenu d'une recette avec les ingrédients nécessaires à sa réalisation et les étapes à suivre en cliquant dessus. Il peut également **filtrer** les recettes par titre en utilisant la barre de recherche.

L'idée de notre site n'est certes pas révolutionnaire mais elle nous permet de respecter tous les points techniques demandés pour ce projet :

- Nous avons fait le choix de ne pas laisser l'utilisateur se créer un compte. Seul l'administrateur du site peut utiliser la fonction « *Se connecter* » pour se connecter avec **test@test.com** / **password**.
- A chaque fois que l'administrateur se connecte, une nouvelle session est créée côté serveur grâce au module **express-session.**
- L'interaction CRUD est respectée car l'administrateur une fois connecté peut ajouter une recette, supprimer n'importe quelle recette en cliquant sur la croix rouge ou modifier les étapes de la recette. Tous les utilisateurs voient les recettes.
- Nous utilisons une liste dans */routes/Articles.js* pour stocker nos recettes en mémoire ce qui signifie que les données ne sont pas persistantes.

Conception

Pour réaliser cette application web, nous nous sommes d'abord focalisé sur la partie cliente et l'architecture de notre site. Nous voulions des éléments précis comme une barre de navigation, avec la possibilité de voir la page principale et la page « *Qui sommes-nous ?* » ainsi qu'un bouton « *Se connecter* » à droite. Nous voulions une barre de recherche et un bas de page classique.

Nous sommes par la suite montés en compétence sur NodeJs et VueJs et avons continué le projet en gérant en parallèle la partie serveur et cliente. Nous voulions rendre cette application intuitive, simple et épurée pour que l'utilisateur ait une expérience satisfaisante sans avoir à chercher longtemps une fonctionnalité.

Réalisation

Côté client :

Nous avons une page principale */public/index.html* qui n'est appelée que lorsque l'on fait un **GET /**. L'application est une application de page unique (SPA) et tout est rendu dynamiquement à l'aide du framework **Vuejs**. Le contenu est affiché ou caché à l'aide de variables booléennes et de v-show / v-if / v-else. Nous avons deux « *pages* » : la

page principale où l'on peut consulter les recettes et une page « *Qui sommes-nous ?* » classique.

Notre unique page */public/index.html* est composée de plusieurs composants (***vuejs components***) avec notamment une **barre de navigation**, nos **recettes** (barre de recherche, création, modification, ajout et affichage), un **système de login** classique et un **bas de page**. Ces différents composants communiquent entre eux à l'aide d'un **bus principal** */javascripts/bus.js* et chaque composant est stocké dans un fichier */public/javascripts/components/*.vue*. Les composants et leurs éléments respectifs se cachent et s'affichent en fonction des actions de l'utilisateur, s'il est connecté ou non en tant qu'administrateur par exemple. Le point de départ de la logique de tous ces composants est le fichier */public/javascripts/main.js*.

Nous avons également créé toutes nos feuilles de styles « à la main » et avons tenu à **ne pas utiliser de bibliothèques tierces** comme bootstrap. Nous avons notamment utilisé **flexbox** pour afficher tous nos éléments dynamiquement.

Pour toutes nos requêtes (GET et POST), nous avons utilisé le module **axios** pour sa simplicité.

Côté serveur :

Nous avons utilisé **NodeJs** et **Express** pour réaliser le site côté serveur. Pour gérer l'authentification nous avons utilisé le modules **express-session** couplé au module **bcrypt**. Le **hash** du mot de passe administrateur est stocké dans une variable côté serveur et non pas en clair grâce à ce module. Le hash est comparé à chaque authentification. Lorsque l'administrateur se déconnecte en cliquant sur « Se déconnecter », la session est détruite.

Toutes nos routes se situe dans le fichier */routes/index.js* et sont gérées avec le middleware **express.Router()**. Pour **protéger** les routes **sensibles** (création, suppression ou modification de recette), nous avons une fonction permettant de **vérifier** que la **session en cours** corresponde bien à celle de l'administrateur préalablement connecté (en vérifiant la session) avant d'exécuter ces différentes actions. Cette vérification côté serveur est cruciale car ne faire qu'une vérification côté client ne vaut pas grand-chose puisque ce dernier peut **altérer facilement** ce qu'il reçoit ou ce qu'il envoie.

Déploiement

Nous avons déployé assez facilement notre application sur *Glitch* en la synchronisant à notre répertoire *GitHub*.

Problèmes rencontrés

La difficulté majeure dans ce projet était notre manque d'expérience dans ce domaine. Nous avons dû nous mettre techniquement à niveau rapidement, suivre des tutoriels plus ou moins bons en ligne, ne répondant souvent pas à nos problématiques.

L'appréhension de *NodeJs* et *VueJs* ne fut pas évidente, on avait tendance à mélanger la partie cliente et serveur. Le fait que les deux soient en JavaScript ne nous a évidemment pas aidé au début ...

Lors du déploiement nous avons dû adapter quelques éléments de notre code. Nous avons également modifié légèrement le */package.json* pour que *Glitch* puisse démarrer notre serveur à partir de */app.js* et non pas de */server.js* qui est le point d'entrée par défaut.

Conclusion

Avec un peu de recul nous aurions pu économiser de nombreuses heures en évitant de suivre certains tutoriels complètement hors-sujet avec ce que l'on recherchait.

Le fait d'écrire tout en JavaScript nous a dans un premier temps embrouillé puis nous nous sommes rendu compte de sa puissance. Tout écrire dans un seul langage est rapide, simple et cohérent.

Cette application est notre première application web réaliste utilisant des outils modernes et nous a permis de mieux connaître le concept de front-end / back-end. Nous avons pu comprendre les enjeux d'une application web, qu'ils soient à la fois au niveau de l'expérience utilisateur côté client ou côté serveur avec des problématiques de stockage de données et de sécurité.