esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

Programmation Web

Trouve ta recette !

Encadrant : M. CHEREL Louis

Elèves : Agnès Gaspard et Angélique Baille

Un projet qui répond à des contraintes

Notre projet consiste à créer un site web centralisant une banque de données de recettes, qu'il affiche selon certaines caractéristiques. Il est possible de classer par catégorie (entrée, plat ou dessert), par ingrédient, ou encore en fonction de leur entrée récente dans la banque de données. Cette action est réalisée par un utilisateur grâce à différents boutons du site.

Ce même utilisateur peut se connecter à notre application pour accéder à de nouvelles options. En effet, lors de la consultation d'une recette, il existe deux boutons : un pour accéder à ses favoris et l'autre pour sa liste de courses (cf. la partie des difficultés rencontrées). L'utilisateur connecté a aussi accès à un menu déroulant contenant ses favoris, sa liste de courses, un bouton permettant d'ajouter ses propres recettes au site, ainsi qu'une proposition de recette choisie aléatoirement parmi la liste des recettes. Il peut aussi voir ses informations personnelles et se déconnecter grâce à un autre menu déroulant.

Par ailleurs, il existe un compte administrateur qui n'a pas accès aux mêmes options qu'un utilisateur lambda une fois connecté. Il peut en effet voir la liste des utilisateurs ayant un compte sur notre site, ainsi que la liste des recettes. De plus, l'administrateur est en mesure d'ajouter, de modifier ou de supprimer une recette.

Le « *front-end* » du site est réalisé en HTML, CSS, et aussi avec le framework Vue.js, et l'application suit le modèle du « *single page website* ».

Le « *back-end* » du site est construit à partir du framework *Express* de Node.js et toutes nos données sont conservées sur un serveur. Elles sont récupérées à partir de celui-ci au démarrage du site et aussi lorsqu'un utilisateur se connecte, et chaque modification est prise en compte à la fois par le serveur et par le site en « *front-end* ». Les seules requêtes vers le serveur passent par des GET, POST, PUT ou DELETE.

Le site est accessible depuis un repository sur Github et grâce à une url depuis Glitch :

Depuis Github : https://github.com/Bisousnours/Trouve_ta_recette

Depuis Glitch :

- Code : https://glitch.com/edit/#!/serveur
- Application : https://serveur.glitch.me

De la conception au déploiement

Etape 1 : Choix du thème général de l'application

Après avoir eu connaissance des contraintes imposées pour le projet, nous avons dans un premier temps réfléchi au thème du site que nous allions réaliser, et nous avons choisi le sujet des recettes de cuisine.

Etape 2 : Brainstorming

Le thème général étant établi, nous avons réalisé un brainstorming afin de mettre en place une ligne directrice commune, en nous mettant d'accord sur les différentes fonctionnalités et sur le design global du site.

Etape 3 : Répartition des tâches

Nous nous sommes ensuite réparti les différentes tâches à réaliser.

Etape 4 : Méthode de mise en commun de nos codes

Etant donné que nous devions travailler sur le même code, et surtout sur les mêmes fichiers, il n'était pas possible de travailler localement. Aussi, nous avons mis notre code sur la plateforme de partage Glitch.com.

Etape 5 : Réalisation

Le cadre du projet étant ainsi posé, nous sommes donc passées à l'étape de réalisation.

a. Nous avons commencé par construire la structure visuelle du site en HTML et CSS. Pour modifier le visuel sans changer de page, nous avons créé des sections à afficher ou à cacher lors de la navigation sur notre site.

b. La découverte de Vue.js nous a amenées à repenser plusieurs parties de notre code notamment à quel moment et comment afficher nos différentes sections.

c. Nous avons créé les fonctions d'affichage, de filtre et d'authentification.

d. Une fois le « *front-end* » fonctionnel, nous avons attaqué la partie serveur avec des exemples et un simple *GET* et des tests côté client.

e. Enfin, nous avons implémenté toutes les fonctions liées au serveur.

Etape 6 : Déploiement

Pour le déploiement de notre application, nous avons utilisé Glitch qui permet aussi d'obtenir une adresse URL.

Les difficultés rencontrées

En plus de petits problèmes rencontrés (comme dans toute programmation d'application) que nous avons réglés assez rapidement, nous avons été confrontées à des difficultés plus importantes qui nous ont demandé plus de temps et de réflexion.

Stockage des données

L'une des premières difficultés a été de faire la distinction entre les données stockées au niveau du « *front-end* » et celles stockées en mémoire dans le « *back-end* ». Nous mettions en effet initialement toutes nos données dans notre vue. Mais après quelques recherches et réflexions, nous avons compris que la mémoire du « *front-end* » était une mémoire directement liée à l'interface visuelle, et pouvant être qualifiée de « mémoire à court terme » (réinitialisation à chaque mise à jour de la page), tandis que la mémoire du « *back-end* » correspondait à une « mémoire à long terme » qui n'est réinitialisée que lorsque le serveur est arrêté.

Un autre problème que nous n'avons pas résolu est l'ajout d'une recette aux favoris en « *back-end* », nous l'avons fait en « *front-end* ».

Vue.js

Un deuxième obstacle important à surmonter a été de comprendre les finesses de Vue.js. En effet, nous avions commencé notre code en créant une vue pour chaque base de données que nous utilisions : liste de recettes, de courses, de favoris, d'ingrédients…. Ce qui nous a rapidement posé problème pour utiliser ces données. Il nous a alors fallu ajouter des variables à notre vue initiale et corriger tout notre code, ce qui a représenté une certaine perte de temps.

De plus, des subtilités comme « *filter* » ou encore la fonction fléchée ne nous apparaissaient pas clairement, la fonction de filtrage a alors été compliquée à réaliser. Nous avions du mal à atteindre les listes incluses dans d'autres listes (ingrédients dans recette). Et à l'issue de nombreux tests et de longues recherches, nous avons réussi à obtenir le résultat souhaité.

La gestion des « *v-model »* fut aussi laborieuse, notamment pour la variable « *contenuActuel* », car nous la changions naïvement dans une fonction qui se rappelait elle-même pour vérifier l'état de cette variable. Il a donc fallu changer notre logique de départ, et nous n'avons pas trouvé d'autre solution que de recréer une partie de la fonction ailleurs.

Serveur

Le principe de *route* fut complexe à appréhender car elle semblait contredire le côté « *single page application* » de notre site. Ce n'est que lorsque nous avons considéré cela comme un stockage de données que nous avons pu mieux aborder le problème. Cependant, au fur et à mesure que nous avancions dans le code (et rencontrions des difficultés), nous nous sommes aperçues que chacune d'entre nous avait une compréhension différente du serveur. Le fait de réaliser deux fois les mêmes actions, une côté client, une côté serveur, nous semblait étrange. Mais finalement, à l'issue de discussions et nombreuses recherches, nous avons fini par éclaircir ces « mystères ».

Glitch

Nous avons aussi rencontré plusieurs problèmes avec la plateforme Glitch, notamment lors de la création d'un projet *Express* avec la notion de serveur. En effet, l'arborescence d'un tel projet ne correspondait pas à celle que nous avions en local (et qui était demandée), et il nous a alors fallu ajouter et modifier certains fichiers du projet.

De plus, la méthode *POST* nous a envoyé une erreur de type 500 lors d'un test que nous avons effectué alors que nous n'avions pas modifié le code, et que le code présent sur Glitch fonctionnait parfaitement sur un serveur Node.js local. Malgré de nombreuses recherches, nous ne comprenons toujours pas d'où vient ce problème, mais nous l'avons contourné en important notre projet à partir d'un dépôt Github.

Rafraichissement de la page

Nous avons aussi rencontré des problèmes de moindre envergure mais qui nous ont tout de même pris du temps. Par exemple, la validation d'un formulaire via notamment notre bouton « Recherche » entrainait parfois un clignotement. Nous ne savions alors pas que cette action rechargeait la page, et cela rendait parfois notre programme inopérant.

Pour aller plus loin

Nous avions pensé à de nombreuses options à intégrer au site, mais nous avons manqué de temps pour les réaliser, en raison notamment de nos diverses difficultés précédemment évoquées.

- Permettre à un utilisateur de s'inscrire et que ses informations soient enregistrées dans notre base de données utilisateur. Cela impliquerait de vérifier identifiant et mot de passe lors de la connexion pour que seuls ceux ayant un compte aient le droit de se connecter. Chaque identifiant serait alors unique.
- Trier les recettes alphabétiquement quel que soit le filtre d'affichage.
- Créer un système d'avis où chaque utilisateur connecté pourrait voter entre 1 et 5 pour les recettes. Une moyenne de tous les avis serait calculée pour attribuer une note globale à la recette.
- Changer l'icône cœur des favoris en noir lorsque la recette est dans les favoris de l'utilisateur connecté.
- Faire de même pour l'icône du caddy et ajouter des ingrédients d'une recette à la liste de courses.
- Permettre un retour en arrière sur nos sections grâce à un bouton.
- Créer un modèle de message que l'administrateur pourrait utiliser pour communiquer des annonces aux utilisateurs possédant un compte sur notre site.
- Mettre une alerte pour les utilisateurs lorsqu'une nouvelle recette est ajoutée.
- Faire valider par l'administrateur tout ajout de recette par un utilisateur avant de l'afficher et de « l'enregistrer » dans notre base de données de recettes.

Conclusion

La création de cette application nous a donc appris à manipuler de nombreux outils et, même si nous avons rencontré plusieurs difficultés, nous avons réussi à rendre un site relativement fonctionnel qui répond aux contraintes demandées, dans le temps imparti.

Nous avons découvert différents aspects de la programmation Web auxquels nous ne nous attendions pas, ce qui nous a permis de laisser libre cours à notre imagination et de réaliser un site assez complet.