Développement et Difficultés
PC3R

Sofiane BELKHIR & khadija El-MAACHOUR & Oussama LAARAJ

Mai 2021

Table des matières

1 Développement et Difficultés

1.1 Points forts de notre application

Grace a notre application, on peut réserver un ticket de train pour une gare destinataire à partir d'une gare d'origine, ce ticket est valable tout le temps, et la personne qui a réservé ce ticket peut partir à la destination prévue quand elle le veut.

Les utilisateurs auront l'embarras du choix en terme de tarifs, en effet pour chaque gare destinataire, on propose deux tarifs, les tarifs, première classe et de deuxième classe, l'utilisation peut également visualiser les coordonnées de la ville sur la carte.

Quand l'utilisateur fait une recherche, on questionne d'abord notre base de données. En effet, si la gare d'origine existe dans notre base, on affiche les résultats, sinon on questionne l'API et les résultats obtenus seront rajoutés à notre base, on peut conclure que notre base continent le minimum afin d'alléger la mise à jour.

Le point le plus fort de notre application est l'utilisation de deux API, en effet, on a utilisé une API SNCF du coté serveur afin de récupérer les données telle que la gare d'origine, la gare destinataire et bien sur les tarifs, et du côté client, on a rajouté une API GOOGLE-MAP qui nous permet de voir les gares dans une map.

1.2 Difficultés rencontrées

Nous avons rencontré un certain nombre de difficultés dans le développent de notre application, et parmi eux, on trouve la mise à jour de notre base de données avec L'API SNCF.

Ça nous apprit beaucoup de temps pour comprendre la technologie CRON, car il y avait plusieurs primitives, même si à la fin ça nous as paru simple.

Le deuxième problème qu'on a rencontré, c'est le développement de la partie client avec un nouveau langage "reactjs", mais en fin de compte, on a pu maîtriser le fonctionnement de ce dernier.

Le troisième problème rencontré n'est pas lié à l'application elle-même, mais a notre situation, en effet à cause du manque de temps et de la situation actuel, on aurait bien aimé améliorer notre code, par exemple : on a fait une mise à jour brute, en gros on identifier toutes les données qui se trouvent dans notre base et on les supprime et après, on interroge l'API pour reconstruire notre base à nouveau. Si nous avons eu un peu plus de temps on aurais pu faire une mise à jour parfaite qui ne requière pas un formatage complet.

1.3 Choix d'implémentations

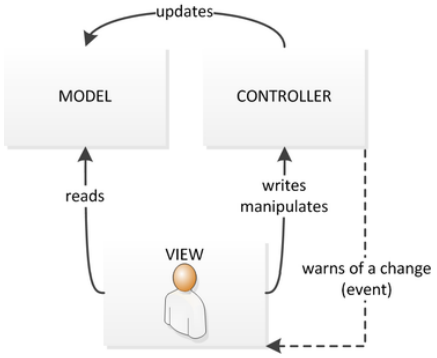
Dans notre projet, on a préféré utiliser l'approche service du coté serveur. En effet, cette approche nous a permis d'identifier les fonctionnalités nécessaires à la réalisation d'une application web rentable d'un côté et de l'autre nous n'avons pas pu concevoir notre application via une approche ressource, car elle nous a paru un peu difficile surtout en terme de maintien d'une session et tout ceci est cause par le manque d'état.

En plus de ça l'approche ressource n'est pas très sécuriser par rapport à une approche service SOAP, c'est la raison pour laquelle REST est approprié pour les URL publiques, mais il n'est pas bon pour le passage de données confidentielles entre le client et le serveur, et nous on aimerait garder cette confidentialité.

En ce qui concerne la partie cliente, on a préféré utiliser 4 pages indépendantes, une page d'accueil pour chercher un trajet, une page d'authentification pour s'identifier ou bien pour créer un compte, et une page pour choisir le tarif soit première classe ou deuxième et valider la réservation et la dernière page qui reflète les tickets déjà réservé qui peut être éventuellement téléchargé .

1.4 Choix de design

En terme de design, on a utilisé le design pattern Model-View-Controller (MVC) coté serveur, grâce a ce dernier, on a pu séparer la couche de présentation de la couche de gestion, voici un schéma qui récapitule ce design pattern :



Sachant que dans notre cas le controlleur, c'est les servlets, comme on n'a pas utilisé les pages JSP du coup notre vue, c'est le client et le modèle, ce sont toutes les classes qui sont en relation avec notre base de données.

Coté client, on a utilisé bootstrap, css et reactstrap pour concevoir nos pages.

On a choisi l'api SNCF, car vous avez dit que vous n'aimiez pas leur site de réservation, du coup, on voulait vous proposer quelque chose de meilleur.