Application Type SNCF

I. Cahier des charges

L'objectif de ce projet était de développer une application web faisant des requêtes à une base de données (lecture/écriture). Le sujet que l'on a choisi est une application web de gestion et de réservation de billet de train.

Le site web doit répondre aux problématiques suivantes :

- Les billets de train sont représentés avec leurs gares de départ/arrivées, la date et l'heure de départ/arrivée, le numéro du train et le prix de base
- Les clients sont identifiés par leur nom/prénom et ont accès ou non à une réduction.
- Les trains sont composés de voiture ayant un nombre de place maximum. Et les voitures sont composées de place ayant un numéro et un type (couloir/fenêtre).
- Les clients choisissent un billet, le prix après réduction est calculé et ils peuvent réserver s'ils le souhaitent

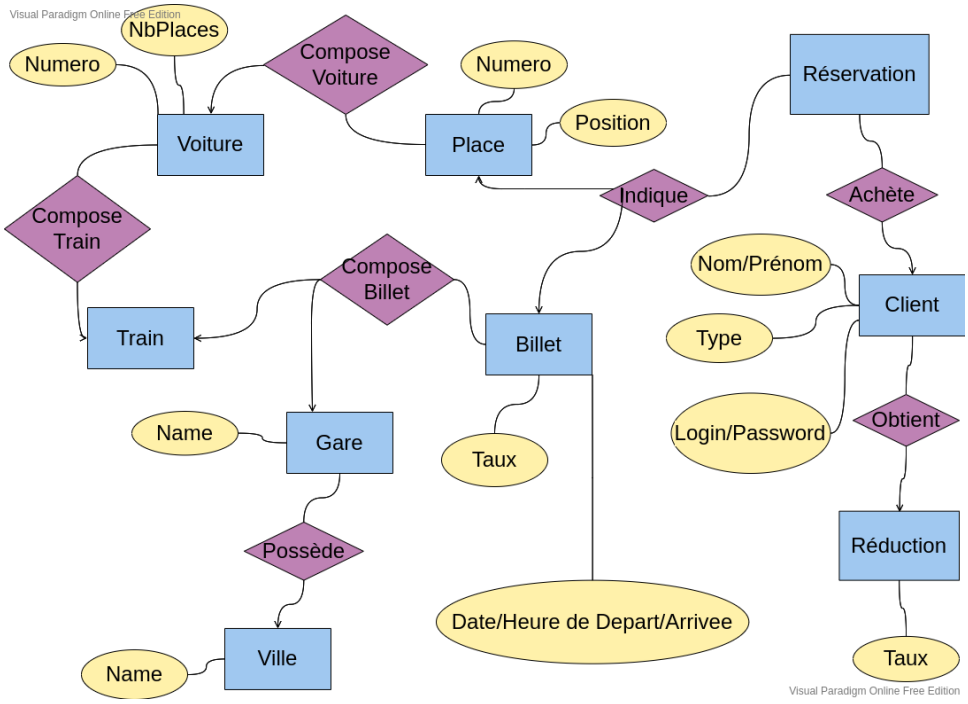
II. Solution proposée

La SNCF gère tous les trains et réservations de train à un niveau national. Ce n'est pas moins de 3000 gares ferroviaires, 15 000 départs de train par jour et 10 millions de voyageur par jour. source

Il faut donc un data center conséquent pour stocker toutes les données, un serveur puissant capable de gérer des millions de requête en simultanées, des moyens de contrôle afin que des personnes de prennent pas le même billet etc..

Modèle Entité-Association :

Voici notre modèle Entité Association :



Notre base de données est donc composée de 9 tables :

- Ville : Avec juste une colonne nom qui est sa PRIMARY KEY
- Gares : Qui est liée à une ville et qui possède un nom
- Reduction : Qui possède un nom qui est sa PRIMARY KEY et un taux de réduction
- Client : Qui possède un nom/prénom, un type (Adulte ou Enfant), un login qui est sa PRIMARY KEY et un password chiffré. Il est lié à une réduction.
- Train : Qui possède juste un id qui est sa PRIMARY KEY
- Voiture : Qui est liés à un train et possède un nombre de place ainsi qu'un numéro relatif à son train
- Place : Qui est liés à l'id d'une voiture (pas son numéro relatif) et possède un numéro relatif à sa voiture et une position (Couloir, Fenêtre).
- Billet : Qui est lié à un train et deux gares (arrivée/départ), qui possède un prix (l'image est fausse ce n'est pas un taux) et une date/heure de départ et d'arrivée
- Réservation : Qui est liés à un client et un billet. Elle possède aussi des numéros relatifs de place et de voiture

Les tables où nous n'avons pas déclaré la PRIMARY KEY ont un id arbitraire. Pour rendre la base de données robuste chaque fois où il y a écrit "Qui est lié" nous avons ajouté une clef étrangère vers la table citée.

Nous avons de plus ajouter d'autre contrainte pour rendre la base de données encore plus robuste.

- Gares : le couple (Name, Ville) doit être unique
- Voiture : le couple (Train, Numero) doit être unique
- Place : le couple (Voiture, Place) doit être unique
- Réservation : le triplet (Billet, Voiture, Place) doit être unique

-

Nous avons rajouté des triggers supplémentaires :

- Lorsque l'on rajoute une place à la table place, si le nombre de place associées à une même voiture devient plus grand que le nombre de place de la voiture, l'insertion est stoppée
- Lorsque l'on rajoute une réservation, si le numéro de place ne correspond à aucune place associée au numéro de voiture relatif associée au train du billet, alors l'insertion est stoppée

Toutes précautions ont été prise pour contrer les potentiels utilisateurs malveillants envoyant des requêtes sans utiliser le site web. En effet, si les utilisateurs utilisent le site web, beaucoup de choix ne leur sont même pas proposés.

Cette base de données répond entièrement aux attentes du cahier des charges.

Fonctionnement du logiciel :

Pour appliquer la réduction et avoir un suivi de qui fait les réservations nous nous sommes dirigés vers un système d'authentification. Pour concevoir le site web, nous avons utilisé le framework node.js permettant de simplifier la mise en place de routeur, et pour l'affichage des pages, nous utilisons ejs permettant de créer des pages html en envoyant des variables et qui nous permets de faire des conditions/boucles etc dans le html. Le site est censé être utilisé ainsi :

- Le client arrive sur la page, s'il n'est pas identifié il sera en permanence redirigé vers la page d'identification. Un client non identifié n'a accès qu'à deux pages, la page d'identification et la page d'inscription.
- Le client remplit le formulaire d'inscription et remplis le formulaire d'identification.
- Quand il est identifié il a enfin accès aux autres pages
- Un token stocké serveur permet à son authentification d'être sauvegardé, il n'a pas à se réidentifier en permanence.
- Il arrive sur la page d'accueil, il peut choisir une ville de départ, une ville d'arrivée et une date.
- La liste de tous les billets correspondant est envoyée.
- En haut à droite, il peut voir toutes les réservations effectuées et il peut se déconnecter (cela supprime le token coté serveur).
- Il peut choisir un billet affiché
- Puis sur une nouvelle page, il peut choisir sa voiture et sa place (une requête AJAX empêche une sélection (Voiture/Place) déjà prise.

On utilise une architecture MVC. Le contrôleur renvoie les requêtes au modèle qui fait des requêtes à la base de données et renvoie une vue.

III. Guide d’installation

Voir le README.md sur le dépôt github : https://github.com/jgoutal/Agence-SNCF