Micro-Projet Web
Cahier des Charges
PC3R

Sofiane BELKHIR & khadija El-MAACHOUR & Oussama LAARAJ

MAI 2021

Table des matières

1 Sujet de l'application et ses principales fonctionnalités

Notre application permet à un utilisateur de trouver toutes les gares destinataires possibles pour un voyage en entrant la gare de départ, et en plus, elle permet d'avoir les tarifs de premières et de deuxièmes classes sur chaque gare destinataire.

Après consultation des tarifs, l'utilisateur peut réserver un billet de train directement, en créant un compte ou en se connectant à l'aide de ces identifiants.

2 Description de l'API SNCF

Pour implémenter notre application, nous avons choisi l'API « Tarifs Intercités de jour », l'API contient plusieurs informations : gare de départ, gare destinataire, tarifs... Voici, le format des informations de chaque attribut :

* "origine" : "type" :"string", "title" :"Origine", "description" :" la ville de départ"
* "destination" : "type" :"string", "title" :"Destination ", "description" :" la ville destinataire"
* "prix d'appel 2nde" : "type" :"number", "title" :"Prix d'appel 2nde", "description" :""
* "plein tarif 1ere" : "type" :"number", "title" :"Plein Tarif 1ère", "description" :"Plein tarif échangeable et remboursable 1ère classe,"
* "plein tarif 2nde" : "type" :"number", "title" :"Plein Tarif 2nde", "description" :"Plein tarif échangeable et remboursable 2nde classe,"

cf. https ://ressources.data.sncf.com/explore/dataset/tarifs-intercites-de-jour/api

3 Fonctionnalités

* page d'accueil.
* rechercher un trajet.
* consultation des trajets possibles.
* consultation des tarifs.
* création d'un compte utilisateur / authentification.
* réservation d'un billet

4 Cas d'utilisation

4.1 Premier cas

Alice arrive sur la page d'accueil qui saisit une gare de départ, le système affiche toutes les gares destinataires possibles avec le tarif et le prix d'appel, elle sélectionne ensuite la destination souhaitée en cliquant sur le bouton "Réserver".

Le système lui demande sois une authentification, elle s'authentifie.

Après le système lui affiche une page "Booking", elle choisit le tarif correspondant.

4.2 Deuxième cas

Bob est un utilisateur, qui veut consulter les tarifs et les gares destinataires possibles afin de choisir le moins cher "sans avoir a réservé".

4.3 troisième cas

Sofiane est un utilisateur, qui veut partir en vacances après une année de confinement et de stress.

Il décide de partir loin de la Sorbonne, il accède à la page d'accueil et saisie la gare Paris Bercy, il choisit au hasard une gare destinataire, il créait un compte, mais encore une fois, il est redirigé vers la page d'accueil, il refait une recherche, et après avoir cliqué sur le bouton "Réserver", il est redirigé vers la page Booking, il choisit le tarif deuxiéme classe vu qu'il est le moins cher, et après, il clique sur le bouton "book".

Aprés il se déconnecte.

5 Base de données

5.1 Utilisateur

'nom‘ varchar(32) NOT NULL,
'prenom' varchar(32) NOT NULL,
' pseudo' varchar(32) NOT NULL,
'email' varchar(32) NOT NULL,
'motDePasse' blob NOT NULL,
'age' int(11) NOT NULL,
'telephone' varchar(10) NOT NULL

5.2 Réservation

'pseudo' varchar(32) NOT NULL,
‘ 'idTicket' varchar(10) NOT NULL

5.3 Session

'pseudo' varchar(32) NOT NULL,
'clef' varchar(32) NOT NULL,

'temps' timestamp NOT NULL DEFAULT current_timestamp()

5.4 tarifLine

‘origin‘ varchar(32) NOT NULL,
‘plein_tarif_1ere‘ int (3) NOT NULL,
‘destination‘ varchar(32) NOT NULL,
plein_tarif_2nde‘ varchar(32) NOT NULL, ‘
prix_d_appel_2nde‘ int(2) NOT NULL

5.5 villes_france_free

‘ville_id‘ mediumint(8) unsigned NOT NULL AUTO_INCREMENT,
‘ville_departement‘ varchar(3) DEFAULT NULL,
‘ville_slug‘ varchar(255) DEFAULT NULL,
‘ville_nom‘ varchar(45) DEFAULT NULL,
‘ville_nom_simple‘ varchar(45) DEFAULT NULL,
‘ville_nom_reel‘ varchar(45) DEFAULT NULL,
‘ville_nom_soundex‘ varchar(20) DEFAULT NULL,
‘ville_nom_metaphone‘ varchar(22) DEFAULT NULL,
‘ville_code_postal‘ varchar(255) DEFAULT NULL,
‘ville_commune‘ varchar(3) DEFAULT NULL,
‘ville_code_commune‘ varchar(5) NOT NULL,
‘ville_arrondissement‘ smallint(3) unsigned DEFAULT NULL,
‘ville_canton‘ varchar(4) DEFAULT NULL,
‘ville_amdi‘ smallint(5) unsigned DEFAULT NULL,
‘ville_population_2010‘ mediumint(11) unsigned DEFAULT NULL,
‘ville_population_1999‘ mediumint(11) unsigned DEFAULT NULL,
‘ville_population_2012‘ mediumint(10) unsigned DEFAULT NULL COMMENT
'approximatif',
‘ville_densite_2010‘ int(11) DEFAULT NULL,
‘ville_surface‘ float DEFAULT NULL,
‘ville_longitude_deg‘ float DEFAULT NULL,
‘ville_latitude_deg‘ float DEFAULT NULL,
‘ville_longitude_grd‘ varchar(9) DEFAULT NULL,
‘ville_latitude_grd‘ varchar(8) DEFAULT NULL,
‘ville_longitude_dms‘ varchar(9) DEFAULT NULL,
‘ville_latitude_dms‘ varchar(8) DEFAULT NULL,
‘ville_zmin‘ mediumint(4) DEFAULT NULL,
‘ville_zmax‘ mediumint(4) DEFAULT NULL,

6 Mise a jours des données

Cette partie explique le fonctionnement logique de notre architecture, en effet, on doit mettre à jour notre base, et cela, dans plusieurs cas les voici :

- Modifications des tarifs.
- Ajouter des nouvelles lignes dans notre base de données.

On a utilisé la technologie CRON, en effet a chaque minuit on exécute le service "UPDATE" fournis par notre serveur.

7 Description du serveur

On a choisi l'approche service qui consiste à visualiser le serveur comme une bibliothèque de services. En effet, le serveur reçoit les requêtes de la part du client et les traites en se servant de la base de données à partir des paramètres données par le client.

La liste des services fournis par notre serveur est la suivante :

7.1 Service Réservation

Ce service, nous permet de réserver un billet à un certain tarif.

7.2 Service Search

Ce service, nous permet d'avoir toutes les gares destinataire à partir d'une gare origine, et bien sûr de consulter les tarifs d'un trajet spécifique que ce soit en première ou deuxième classe.

7.3 Service Authentification

Ce service, permet à un utilisateur d'ouvrir une session, on entrant ces identifiants pour réserver un billet.

7.4 Service Registration

Ce service, permet à un utilisateur de créer un compte.

7.5 Service ValidSession

Ce service, permet de savoir si une session est valide ou pas.

7.6 Service Coordinate

Ce service, permet de retourner les coordonnées d'une ville dans la map.

7.7 Service UPDATE

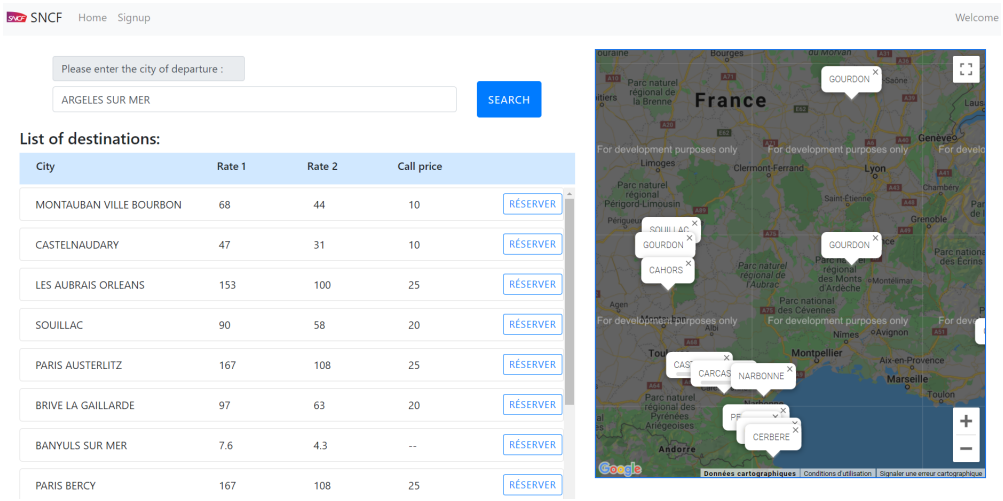
Ce service, permet de mettre à jour notre base de données, en effet ce service est appeler par CRON chaque jour à minuit.

8 Description du client

Le plan du site se compose en 3 pages :

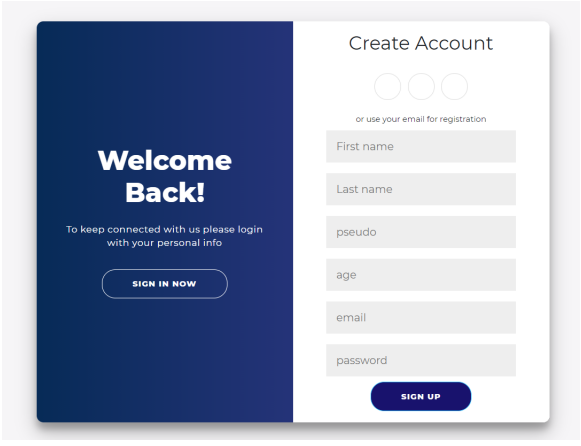
8.1 Page d'accueil

La page contient une barre de recherche afin de trouver un trajet et pour cela en faisant appel au service "Search" du serveur avec la méthode GET.



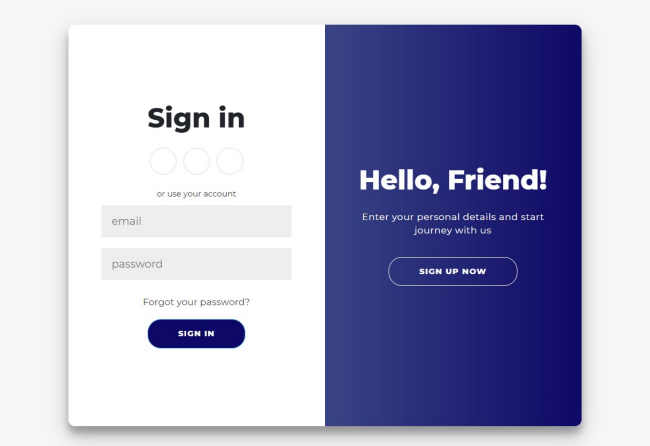
8.2 Page d'authentification

Cette page contient un formulaire pour l'authentification et un bouton "Sign in" qui va faire appel au service "Authentification" du serveur avec la méthode POST d'un coté, et de l'autre un formulaire d'inscription dans le cas où l'utilisateur n'a pas réussi a s'authentifie. Ce dernier fera appel au service "Registration" du serveur avec la méthode PUT.

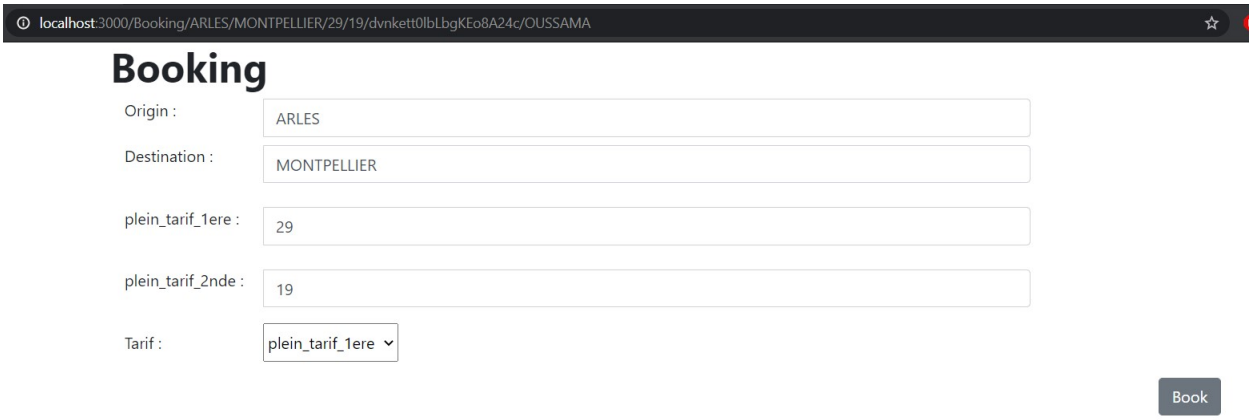


8.3 Page de Booking

La page contient les informations relative à la réservation, il y a un bouton qui permet de choisir le tarif, mais sans indiquer ni la date ni le train à prendre, cette fonctionnalité

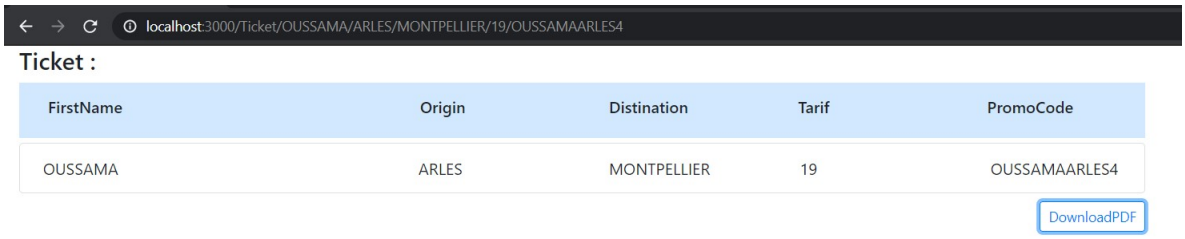


nous permet juste d'avoir un code identifiant comme quoi il a réserver ce dernier à un certain prix. Et à la fin un bouton "book" qui fait appel à la méthode "Réservation" du serveur par la méthode POST.



8.4 Page Ticket

La page contient les tickets qu'on a réservés, et un bouton "DownlandPDF".



9 Déscription des requêtes

Bouton 'Réserver' appelle la méthode 'booking' qui :

* Envoie au serveur une requête GET https/..../'/ValidSession ?key=" pour vérifier si le client est déjà connecté, il se redirige vers la page 'book'
* Sinon vers la page register 'register'

Bouton 'Sign Up' appelle la méthode 'signup' qui :

* Envoie au serveur une requête GET '/Registration ?nom='+this.state.firstname+'&prenom=' +this.state.lastname+'&pseudo='+this.state.pseudo+'&motDePasse ='+this.state.password+ '&email='+this.state.email+'age='+this.state.age+'&telephone='+this.state.telephone ;
* Se redirige vers la page d'accueil.

Bouton 'Sign in' appelle la méthode 'signin' qui :

* Envoie au serveur une requête GET '/Authentification ?pseudo='+this.state.pseudoLogin+ '&motDePasse='+this.state.passwordLogin ;
* se rediriger vers la page ticket.
* se rediriger vers la page ticket.

Bouton 'Book' appelle la méthode 'handleSubmit' qui :

* GET /PromoCodeGen ?userName='+this.props.match.params.Name+'&destination=' +this.props.match .params.destination ;

Bouton 'DownloadPDF' appelle la méthode 'generatePDF' qui :

* genere le fichier pdf.
* rediriger vers la page d'accueil.

10 Schéma du système

Le schéma est disponible en pièce jointe dans le dossier : "systeme.png".