

TravelLog

Ce projet a été généré avec [Angular CLI](#) version 9.1.7.

Installation

Installez [git](#).

Installez [Node.js](#).

Ouvrez l'invite de commandes de votre choix.

Tappez `git clone` avec l'URL du dépôt pour récupérer le projet.

```
$ git clone https://github.com/CH-KevinDubois/travel-log
```

Serveur de développement

Lancer `ng serve --open` pour démarrer un serveur de développement, il ouvrira automatiquement votre navigateur à `http://localhost:4200/`. L'application se rechargera automatiquement si vous modifiez l'un des fichiers sources.

Build de l'application

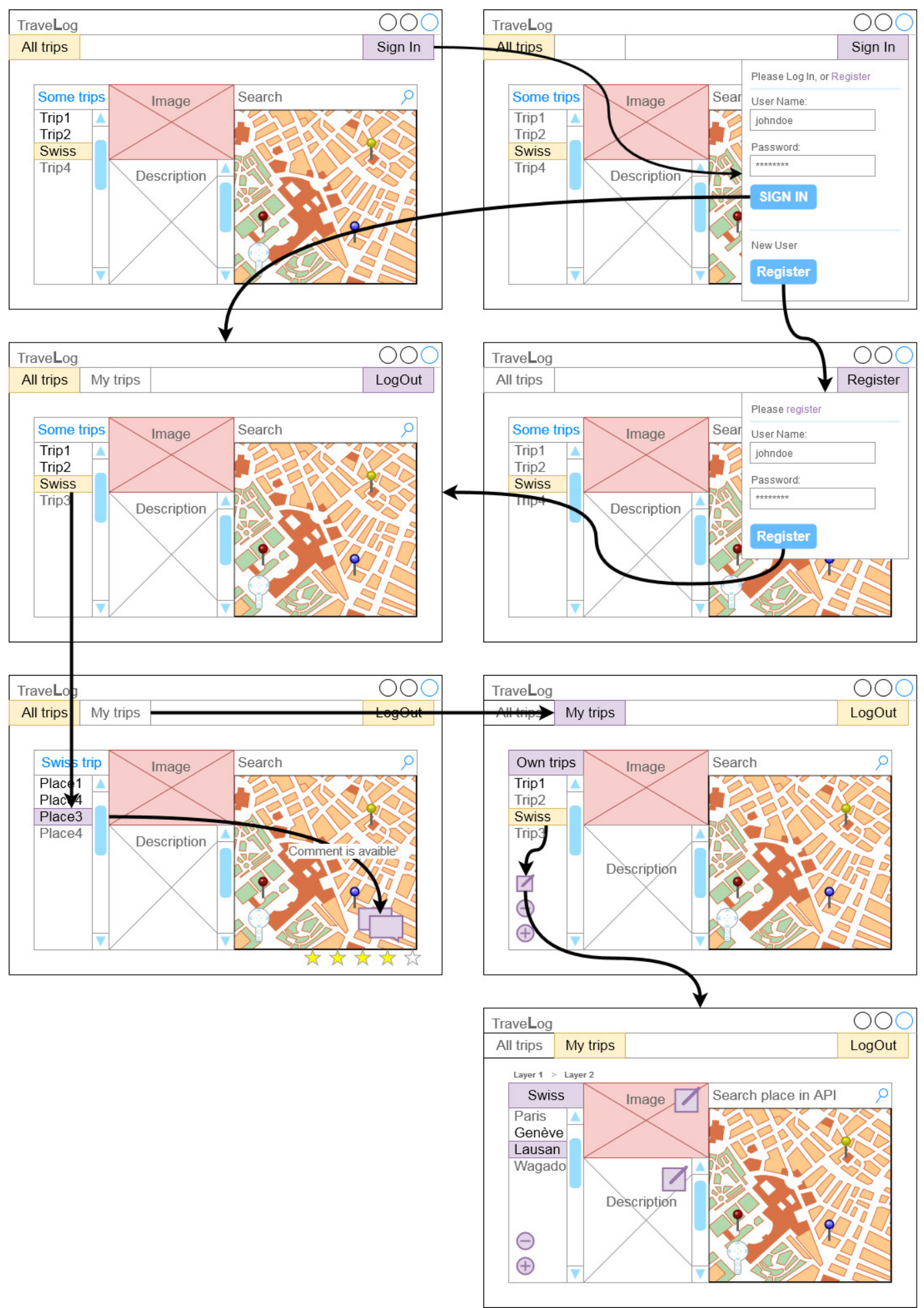
Pour générer l'application lancer la commande `ng build`. Les fichiers générés seront stockés dans le répertoire `dist/`. Utilisez le flag `--prod` pour un build de production.

Approche

Idée initiale

Je suis parti de l'idée de me baser entièrement sur deux vues principales. La première permettant de visualiser l'ensemble de tous les voyages et d'ajouter des commentaires, et la seconde permettant seulement de gérer ses propres voyages. Pour la saisie des informations (se logger, s'inscrire, ajouter un nouveau voyage, ...) je souhaitais utiliser des boîtes de dialogues. Et finalement pour afficher les voyages et les places, j'imaginais utiliser deux listes superposées (une sélection dans la première permettant de passer dans la seconde) et d'utiliser un sort de breadcrumb. Pour ce qu'est de la mise en page et des composants « préfabriqués », je me suis dit pourquoi pas partir sur un petit Angular Material sachant que je connaissais déjà un peu de Bootstrap ! Feu gaz pour la nouveauté, nous sommes là pour le plaisir de découvrir et pour apprendre, aren't we?

Comme conseillé, j'ai créé un petit draft en début de projet :



C'était peu être un peu optimiste comme objectif, mais pour ma décharge, je commençais juste à découvrir ce que permet le Framework, n'ayant aucune connaissance préalable ni d'Angular, de React ou de Vue. Je ne savais pas vraiment ce qu'il m'attendait. Avec un peu de recul, je peux dire que je ne me suis pas franchement simplifié la vie. Premièrement du point de vue de la structure des pages, une approche plus linéaire mais très certainement plus simple aurait été de faire une page par "action", ni plus ni moins. Deuxièmement, du point de vue de l'utilisation de Angular Material, qui n'est mine de rien pas si facile à prendre en main avec tout juste deux mois de cours sur Angular.

Principaux problèmes rencontrés

Le problème principal rencontré avec la structure pour laquelle j'ai opté est le **state** et **data management**. Comment garder une cohérence dans l'application et synchroniser les actions de l'utilisateur et les données entre les divers composants ? J'ai passé par toutes les étapes, en commençant par utiliser des '@Input' et '@Output'. Forcément cela devient vite une usine à gaz et c'est presque impossible de s'y retrouver une fois que les composants sont dans la configuration parent-enfant (en plus chaîner des inputs c'est moche). J'ai testé le '@ChildView' (je ne sais pas si c'est vraiment propre comme approche), mais cela ne règle en rien le problème de "unrelated components". Au final, comme rien ne fonctionnait simplement, je me suis tourné vers **RxJs** et des 'Subjects' injectés au travers de 'Services'.

RxJs c'est beau, c'est "relativement" simple, mais attention aux satanées suscriptions. J'ai cherché des heures durant d'où provenaient certaines duplications d'actions, j'ai trouvé la réponse dans le subscribe(). Je ne me rappelle pas avoir été avertis sur le fait que les suscriptions peuvent foutre un bordel total si elles ne sont pas unsubscribed. Pour régler une fois pour toute ce problème (le faire manuellement est très lourd), j'ai simplement créer un tableau de suscription et à la destruction du composant j'itère simplement sur le tableau et unsubscribe toutes les suscriptions qui y sont stockées.

Quelques mots sur mes choix

Je ne me suis pas attaqué au téléchargement d'une image au sein d'une api externe pour gérer les images de l'application. J'ai pensé pouvoir détourner le champ **pictureUrl** pour stocker une image en base64, mais le champ n'est pas assez long pour des images de grande taille. Au final, je me suis simplement contenté de faire en sorte que l'utilisateur puisse indiquer une url et j'affiche l'image si l'url est donnée.

Je n'ai pas créé de page de recherche à proprement parler. J'ai simplement ajouté au-dessus de la table principale des voyages un champ de recherche et un champ filtre qui sont accessibles un fois l'utilisateur loggé. Je les ai implémentés en me basant sur les composants "chips" d'Angular Material. Pour ce qui concerne la table des places, je n'ai implémenté que la fonction de recherche. Ces deux fonctionnalités ne possèdent pas la même implémentation. La fonction de recherche est basée sur l'API Travel Log en utilisant le query parameter 'search', alors que la fonction de filtre est implémentée directement en RxJs sur le flux de données retournées. La différence de comportement s'observe lorsque l'utilisateur insert plusieurs mots dans le champ filtre ou le champ recherche. La recherche a un comportement de type OR, soit "mot1" ou "mot2" alors que le filtre a un comportement de type AND, soit "mot1" et "mot2".

Lorsqu'un voyage et/ou une place est sélectionnée, une query string indiquant son ID apparaît dans l'url. L'idée était de pouvoir bookmarquer le lien, puis à la saisie d'un tel lien, de directement sélectionner les éléments dont le ou les ID correspondraient. Lors de la sélection d'une place sans avoir préalablement sélectionné un voyage à un comportement différent, on peut retrouver le voyage père qui est notifié. Je

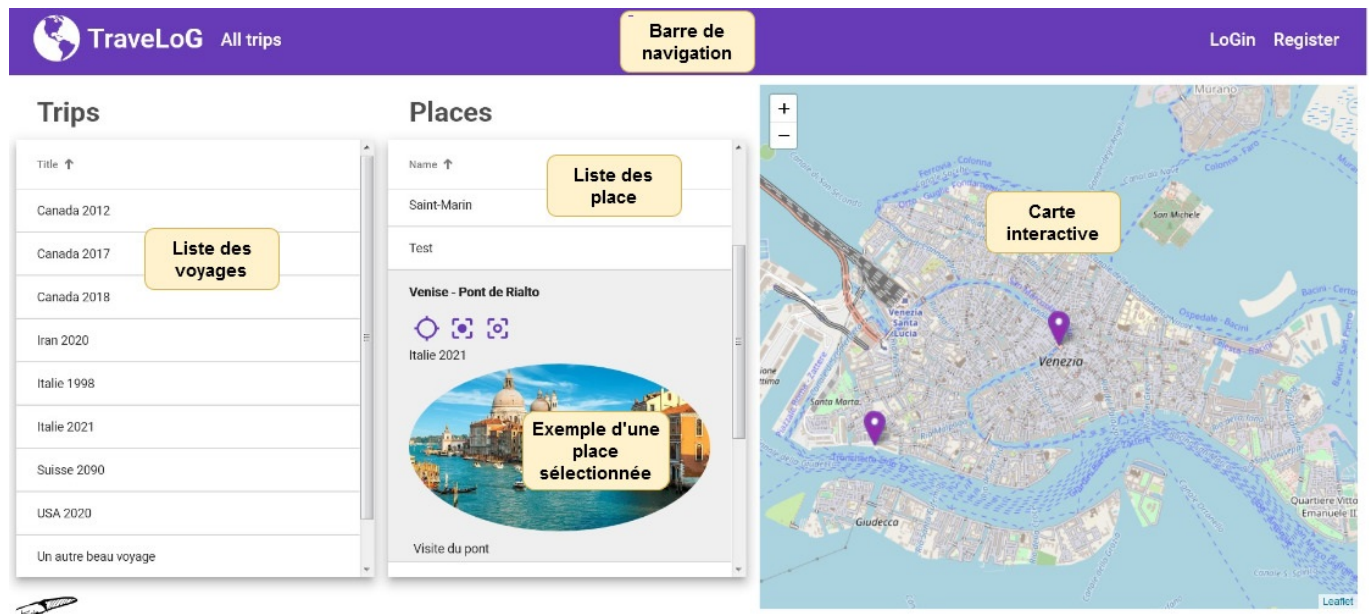
pensais utiliser le même principe pour sélection l'élément père avec un clic sur le nom. Faute de temps, je ne suis malheureusement pas allé jusqu'à l'implémentation de cette fonctionnalité.

Aperçu de l'utilisation de l'application

Page d'accueil - All trips - Etat non loggé





La page d'accueil fait office de page qui permet d'afficher tous les voyages et toutes les places. Les fonctions de recherche/filtre sont désactivées car l'utilisateur n'est pas loggé. Sur cette page l'utilisateur peut :

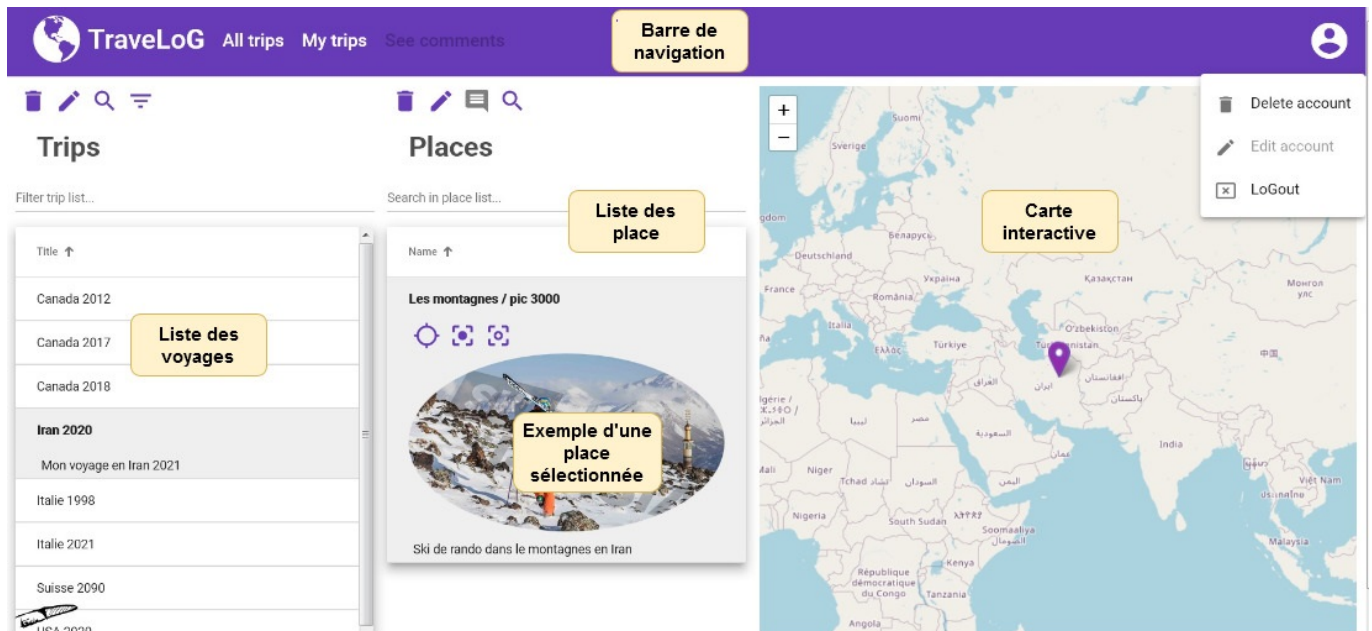
- S'enregistrer (s'enregistrer ne logge pas l'utilisateur automatiquement, il faut le faire en deux étapes)
- Se logger
- Sélectionner une place
- Sélectionner une place
- Se déplacer sur la carte



Barre de navigation - Etat loggé

Sous la barre de navigation, l'utilisateur peut :






- Aller à la page qui permet d'afficher tous les voyages (All trips)
- Aller à la page de ses propres voyages (My trips)
- Aller au commentaires (pas implémenté - retiré du cdc)
- Gérer son compte utilisateur 
 - Supprimer le compte 
 - Modifier le compte (grisé car pas implémenté - ne figure pas dans cdc) 
 - Se délogger 



Page My trips - Etat loggé





Une fois loggé, l'utilisateur est redirigé automatiquement vers la page de ses voyages. Les actions possibles sont :

En entête de la liste des voyages (Trips)


- Si aucun voyage sélectionné, ajouter un voyage 
- Voyage sélectionné, supprimer le voyage 
- Voyage sélectionné, éditer le voyage 
- Chercher dans la liste 
- Filtrer dans la liste 

En entête de la liste des place (Places)

Pour ajouter/éditer/supprimer une place il est nécessaire d'avoir préalablement sélectionné un voyage. Si une place est sélectionnée sans voyage, le nom du voyage associé est affiché au-dessus de la description de la place.

- Si aucune place sélectionnés, ajouter une place 
 - Si la géolocalisation est activée, l'emplacement approximatif de l'utilisateur sera affiché au moyen d'un cercle (prend quelques bonnes secondes).
 - L'utilisateur peut entrer les coordonnées de son choix, ou alors cliquer une localisation directement sur la carte.
- Place sélectionnée, supprimer la place 
- Place sélectionnée, éditer la place 
- Chercher dans la liste 

Page All trips - Etat loggé




Être loggé ajoute la possibilité de filtrer/chercher les voyages et de rechercher les places. L'icône pour l'ajout de commentaires  est grisé car pas implémentés (retrait du cdc).

Sélection d'une entrée de la liste des voyages

Lorsque l'utilisateur clique sur une entrée de la liste des voyages, la cellule active est déroulée pour inclure la description du voyage et, en parallèle, la liste des places et la carte est mise à jour. Seules les places associées au voyage sélectionné figureront dans la liste des places et sur la carte.

Sélection d'une entrée de la liste des places

Lorsque l'utilisteur clique sur une entrée de la liste de places, la cellule active est déroulée pour inclure :

- Icône de localisation  qui permet de recentrer le lieu sur la carte.
- Icône de zoom rapproché 
- Icône de zoom moyen 
- Le nom du voyage père (est actif si le voyage père n'est pas sélectionné).
- La photographie associée (est active si elle est renseignée).
- La description de la place.

Quelques mots pour conclure

Difficile parfois de marier boulot et études. Après des journées 9-10h à gérer des projets, il n'est pas toujours agréable de se plonger dans de la programmation ... j'ai fait ce que j'ai pu avec la motivation et le temps libre à ma disposition afin de rendre quelque chose de potable !

Je n'ai pas précisément calculé mais j'estime avoir passé entre 100 et 150 heures sur ce projet. Ça peut sembler beaucoup oui, mais au final il y a autant d'heures passées à éplucher de la doc ou à regarder des vidéos pour comprendre certaines spécificités/fonctionnalités que d'heures de codages effectives.

Un tout grand merci pour ce cours super intéressant dispensé d'une manière très pro. C'est allé va très très vite (normal vu ta maîtrise du sujet), et je suis sûr que je n'ai pas retenu plus de 30% de ce qui a été montré, mais je me suis fait plaisir en apprenant de nouvelles technologies. Au final, élargir sa vision c'est tout ce qui compte.

KD