Projet IHM 2021 : Kelcarte

Cette application web permet de voir rapidement en effectuant une simulation si la carte jeune SNCF est intéressante ou pas.

Architecture de l'application

Dossier /components

> ButtonStep.vue StepArticle.vue BreadCrumb.vue

Nous avons voulu faire simple et efficace en concevant cette application. C'est pourquoi nos composant réutilisés, tels que :

- le "breadcrumb",
- la div utilisée tout au long du workflow dont le contenu seul change,
- ainsi que le bouton pour faire avancer le workflow.

Ces éléments sont présents dans le dossier **src/Components**.

Dossier /config

> config.js

Ce dossier contient une simple variable *apiURL*, qui permet de changer rapidement l'URL où est notre API.

Dossier /views

> Step0.vue Step1.vue Step2.vue Result.vue

Notre dossier views contient toutes nos vues propres au workflow de notre application. Nous avons utilisé le routeur pour gérer la navigation sur le site en veillant à ce que notre URL change lorsque l'on change de page.

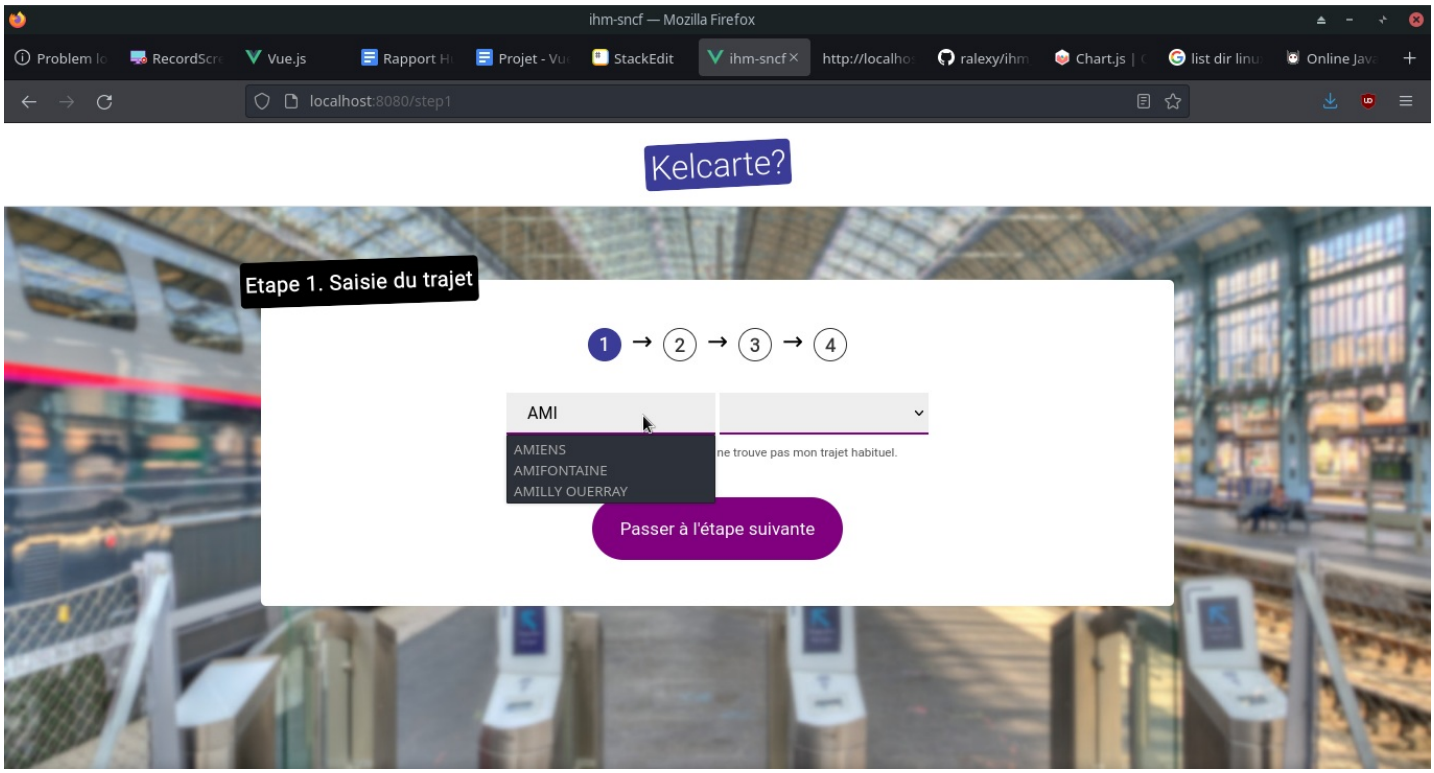
Informations intéressantes sur des fichiers en particulier

views/Step0.vue

Nous avons souhaité produire une illustration engageante, impactante et limpide. Elle vise à donner envie de faire la simulation, ainsi que comprendre rapidement son intérêt.

/views/Step1.vue

L'autocomplétion, couplée au select dans le deuxième champ a été, après réflexion la méthode la plus efficace et la plus naturelle pour saisir une gare de départ et d'arrivée sans déconvenues. Les sites de réservation utilisent ce type de menus qui on été éprouvés depuis des années...



- Nous avons utilisé le localStorage pour conserver les préférences de l'utilisateur tout au long de sa saisie d'informations, dans le but de conserver ses données lorsqu'il navigue entre différentes routes.
- Nous avons eu recours à la librairie Axios pour effectuer des requêtes HTTP vers une API maison, retournant au format JSON :
  - Toutes les gares ayant un suffixe précisé (ici AMI) : http://kelcarw.cluster029.hosting.ovh.net/?action=suggestOriginStation&nameGare=AMI


```
[{"idGare":"1722","nameGare":"AMIENS","idRegion":"7"},{"idGare":"2390","nameGare":"AMIFONTAINE","idRegion":"7"},{"idGare":"2627","nameGare":"AMILLY OUERRAY","idRegion":"10"}]
```


- Toutes les gares desservies par un départ précisé (ici AMIENS) : http://kelcarw.cluster029.hosting.ovh.net/?action=suggestDestinationStation&nameGare=AMIENS


```
[{"nameGareDestination":"LILLE EUROPE"},{"nameGareDestination":"ETAPLES LE TOUQUE"},{"nameGareDestination":"ARRAS"},{"nameGareDestination":"WIMILLE WIMEREUX"},{"nameGareDestination":"MARQUISE RINXENT"},{"
```

- Toutes nos infos nécessaires à la visualisation de données, une fois notre workflow rempli :


```
{"normalPrice":"20.3","nameRegion":"NOUVELLE AQUITAINE","pricePromo":10.15,"periode":"bleue","simulation":[{"normal":0,"discount":49,"interesting":false},{"normal":20.3,"discount":59.15,"interesting":fals
```


Nous avons directement injecté notre HTML dans notre component, pour éviter de le saturer en props, et lui passer directement du HTML au passage.

```
<template>
    <StepArticle id='step2' title='Etape 2. Saisie fréquence de voyage' breadCrumbStep=2>
        <form method="post" @submit="checkForm" autocomplete="off">
            <p class="center">Je compte voyager environ <input type="number" name="frequence" id="frequence" v-model="frequence" step="1" min="1" max="364"> fois par an</p>
            <ButtonStep msg='Passer à l&#39;étape suivante' link='step3' />
        </form>
    </StepArticle>
</template>
```

/views/Step3.vue

Lorsque toutes nos données sont saisies, nous faisons un appel final à notre API pour obtenir nos données pouvant être affichées dans nos graphes :

- Exemple pour un utilisateur partant de Nantes, arrivant à "La Menitre", 1 seule fois par an, sans connaître son jour précis, ni son heure de départ :
    - http://kelcarw.cluster029.hosting.ovh.net/?action=getPrices&origin=NANTES&destination=LA%20MENITRE&day=0&hour=0&frequence=1

/components/Breadcrumb.vue

Le code du breadcrumb a été assez complexe à implémenter, il différait au niveau CSS du contexte où l'on se trouvait (le workflow devait être remonté via un lien, mais on ne devait pas sauter d'étapes) :

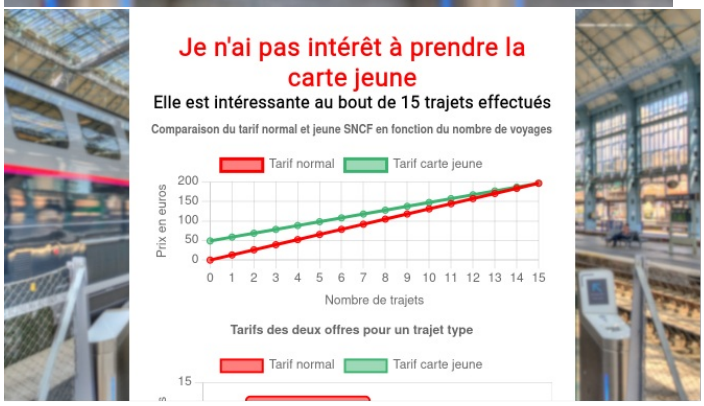
```
<template>
    <!-- On switche l'état de notre breadcrumb selon l'étape, on ne peut pas faire avancer le workflow sans compléter le formulaire (ça n'aurait pas de sens sinon) -->

    <nav aria-label="breadcrumb" class="breadcrumb">

        <ul>

            <li v-if="step == 1"><span class="step active">1</span> <span aria-current="page" class="stepTextActive hide">Saisie du trajet</span></li>

            <li v-else>
                <router-link to="step1"><span class="step">1</span> <span class="hide">Saisie du trajet</span></router-link>
            </li>



            <li v-if="step == 2"><span class="step active">2</span> <span aria-current="page" class="stepTextActive hide">Saisie fréquence de voyage</span></li>

            <li v-if="step < 2"><span class="step">2</span> <span class="hide">Saisie fréquence de voyage</span></li>

            <li v-if="step > 2">
                <router-link to="step2"><span class="step">2</span> <span class="hide">Saisie fréquence de voyage</span></router-link>
            </li>

            <li v-if="step == 3"><span class="step active">3</span> <span aria-current="page" class="stepTextActive hide">Saisie horaire de voyage</span></li>

            <li v-if="step < 3"><span class="step">3</span> <span class="hide">Saisie horaire de voyage</span></li>

            <li v-if="step > 3">
                <router-link to="step3"><span class="step">3</span> <span class="hide">Saisie horaire de voyage</span></router-link>
            </li>

            <!-- Cas particulier, on a accès au résultat qu'en validant l'étape 3 -->

            <li v-if="step == 4"><span class="step active">4</span> <span aria-current="page" class="stepTextActive hide">Résultat</span></li>

            <li v-else><span class="step">4</span> <span class="hide">Résultat</span></li>

        </ul>
    </nav>
</template>
```

/views/Result.vue

Nous affichons aussi bien nos graphes sur un ordinateur, que sur un Smartphone.

Pour améliorer le confort de lecture de nos graphes, nous avons demandé au mobinaute de basculer en mode portrait pour cette dernière étape.

Kelcarte?



Guide de démarrage

Ces instructions vous permettront de récupérer ce projet et de le lancer sur une machine locale de développement à des fins de tests.

Prérequis

Vous devez posséder un serveur web et y installer

```
PHP 7 & MySQL, nodeJS, npm
```

Installation de l'API PHP

> Il n'est pas nécessaire d'installer la partie API du projet, en effet une API déployée et déjà configurée a été livrée avec le projet.

Toutefois si vous souhaitez installer le serveur d'API sur votre machine, il suffira d'importer la base de données fournie dans le dossier sur votre machine **/db/kelcarte.sql** Et d'éditer la configuration SQL du fichier, en précisant vos identifiants **/api/PdoKelCarte.php**

La configuration locale la plus commune pour PdoKelCarte.php est :

```
'localhost' pour le serveur
'root' pour l'utilisateur
'' OU 'root' pour mot de passe
```

L'application web sera consultable via le dossier **/www/**, il est vivement conseillé de mettre en place un VirtualHost ou bien un .htaccess pour empêcher de remonter l'arborescence du serveur (particulièrement si celui-ci est accessible via Internet).

Installation de l'app VueJS

Il suffira de vous placer dans le dossier ihm-sncf et de taper la commande pour tester l'application

```
npm run serve
```

Son adresse locale vous sera communiquée directement dans votre terminal, c'est en général http://localhost:8080

> Pensez également à modifier la constante apiURL dans le fichier **src/config/config.js** par l'adresse de votre API si vous l'avez modifiée.

Tester l'application

L'application est directement consultable ici : http://159.65.56.183

Conçu avec

- VueJS - Framework Javascript réactif nous ayant permis de réaliser l'app
- ChartJS - Librairie JS pour afficher des graphiques
- PHP - Langage de programmation permettant de créer des pages web générées dynamiquement
- PHPStorm IDE spécialisé pour PHP, édité par la société JetBrains également co-autrice de AndroidStudio

Versioning

GitHub a été utilisé pour maintenir un versionning du projet.

Vous pouvez le consulter à cette adresse : https://github.com/ralexy/ihm_sncf

Auteurs

- **Alexy ROUSSEAU** - Etudiant Master 2 IDC - contact@alexy-rousseau.com
- **Manil KESOURI** - Etudiant Master 2 IDC - 21914480@etu.unicaen.fr