Réalisation d'une application web VueJS et Vuetify

1 Introduction

On va utiliser ici les différentes notions abordées dans le sujet précédent. Le but ici est de développer une application web permettant de naviguer dans une liste de films. Pour cela, vous trouverez un fichier json sur Moodle, contenant une liste de catégories et une liste de films. Chaque film est représenté par un titre, un synopsis, le lien vers l'affiche du film (**attention :** certaines images n'existent plus). Il vous faudra à la fois développer le côté client en proposant une interface responsive exploitant la bibliothèque Vuetify, mais également le côté serveur qui répondra aux requêtes de client (obtenir la liste de catégories et obtenir la liste des films appartenant à une catégorie données).

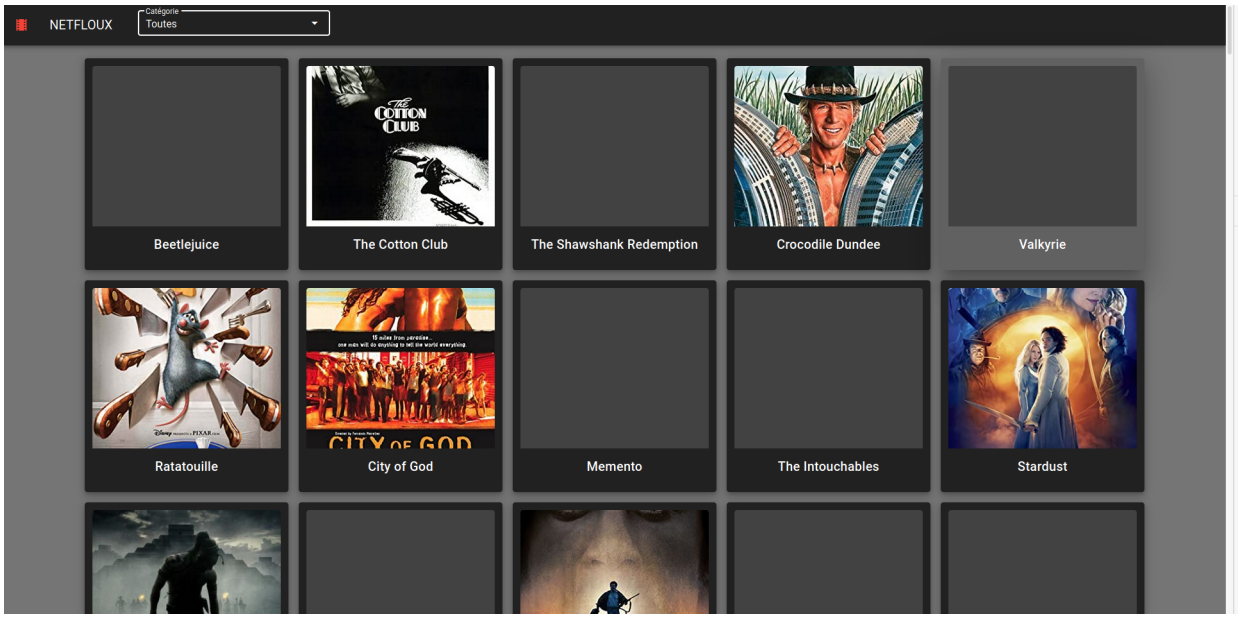

FIGURE 1 – Un exemple de rendu de l'application

2 Mise en place du serveur

Commençons par mettre en place un serveur NodeJS qui servira les fichiers du client tout en proposant une API gérant les données du fichiers JSON. Pour cela, créez un serveur express classique servant les fichiers en statique, auquel on ajoutera les fonctionnalités suivantes : le chargement de la base de données, et la mise en place de l'API pour notre application.

Charger la base de données se fait très simplement, on lit le fichier de manière synchrone, puis on parse le résultat pour obtenir l'objet JSON correspondant :

```
let database = JSON.parse(fs.readFileSync('database.json'));
```

En ce qui concerne la mise en place de l'API se fera via le déploiement de deux méthodes `app.get`. La première permettra à l'utilisateur de récupérer la liste des catégories :

```
  app.get('/categories', function (req, res) {
    res.send(database.genres);
  });
```

La seconde devra permettre à l'utilisateur de récupérer la liste des films répondant à une catégorie, ou tous les films si la catégorie passée en paramètres est 'all'.

3 Création de l'interface

On développera l'interface dans un fichier "index.html", entre les balises `v-app`. On commencera par mettre en place un `v-container` qui contiendra les différents éléments. Celui-ci aura les attributs fluid et fill-height qui ordonnera au composant de prendre toute la largeur et toute la hauteur de la fenêtre. On définira également les marges à zéro :

```
<v-container fluid fill-height class="ma-0 pa-0" >
   <v-layout column align-stretch>
     <!--TODO-->
   </v-layout>
</v-container>
```

On définit ensuite un conteneur flex via un `v-layout`, organisant ses enfants en colonne, avec la propriété "align-stretch", qui forcera les enfants à prendre toute la largeur disponible.

En parallèle, il vous faudra développer le modèle avec les données (liste de films, liste de catégories, variables pour l'interface, ...) mais également les méthodes permettant d'appeler l'API de notre serveur. De plus, il faudra que ces méthodes soient appelées lors de la création de la page. Pour cela, vous pourrez procéder de la sorte :

```
new Vue({
  el: '#app',
  vuetify: new Vuetify(),
  data:{
    ...
  },
  created:function(){
    this.getCategories();
    this.getMovies();
  },
  ...
```

Ainsi, les méthodes `getCategories` et `getMovies` seront appelées dès la création de notre instance de Vue. Il faudra bien entendu créer ces méthodes, chacune d'entre elles faisant appel à axios pour effectuer une requête GET.

3.1 La barre d'application

La barre d'application est le premier élément à afficher dans la page, en partant du haut. Pour cela, on va utiliser le composant Vuetify `v-app-bar`, de la manière suivante :

```
<v-app-bar max-height="150">
  <v-app-bar-nav-icon>
    <v-icon>mdi-filmstrip</v-icon>
  </v-app-bar-nav-icon>
  <v-toolbar-title>
    Ma barre d'application
  </v-toolbar-title>
</v-app-bar>
```

Les deux enfants directs sont l'icône visible à gauche de la barre, ainsi que le titre, qui pourra accueillir par la suite plusieurs composants (un titre et une liste pour sélectionner la catégorie).

1. définir un seul enfant pour `v-toolbar-title`, qui sera un `v-layout` permettant d'organiser ses enfants en ligne ;
2. ajouter comme enfant au `v-layout` une division qui contiendra la titre de l'application, avec une taille de texte appropriée ;
3. ajouter comme enfant au `v-layout` une liste déroulante, dont les objets seront les titres des catégories ;
4. régler les marges et les padding ;
5. modifier les couleurs des différents éléments selon votre envie.

Symétriquement, vous pourrez définir un pied de page avec l'élément `v-footer`.

3.2 La zone centrale

3.2.a Le conteneur

Le conteneur de la zone centrale est un bloc prenant toute la largeur, et toute la hauteur restante (une fois celles de la barre d'application et du pied de page retirées). Pour cela, on met en place un `v-layout` avec les propriétés row, wrap, align-center et justify-center qui permettront respectivement de :

- placer les éléments en ligne ;
- autoriser les éléments à passer à la ligne suivante si la place vient à manquer ;
- aligner les éléments de manière centrée sur l'axe secondaire ;
- aligner les éléments de manière centrée sur l'axe primaire.

```
<v-layout row wrap align-center justify-center
  class="grey darken-1 white--text flex-grow-1">
</v-layout>
```

On lui ajoute également quelques classes concernant les couleurs, mais également la classe "flex-grow-1" (ne fonctionne que si le parent est de la classe d-flex) l'obligeant à prendre toute la hauteur restante du parent.

3.2.b Les cartes de film

La carte d'un film est un composant `v-card` hébergeant une image (`v-img`) ainsi que le titre de celui-ci :

- on veut que la hauteur de l'image soit de 250 ;
- on définit une couleur de fond à celle-ci au cas où ;
- le `v-card-title` contient une division qui contient le titre du film : cette division nous permettra d'afficher le texte sur une seule ligne, en cachant le texte qui dépasse, via le style :

```
    white-space: nowrap;
    overflow: hidden;
    text-overflow: ellipsis;
```

On voudrait également que la couleur de fond de la carte change lorsque la souris survole celle-ci. Pour cela, on utilise le composant `v-hover` :

```
<v-hover v-for="(movie,index) in movies" :key="index" v-slot="{
 ↪ hover }">
  <v-card :class="hover?'grey darken-2':'grey darken-4
   ↪ white--text'" >
    <v-card-text>
      {{movie.title}}
    </v-card-text>
  </v-card>
</v-hover>
```

3.3 L'overlay

Lorsque l'utilisateur clique sur un film, il s'attend à voir les informations concernant ce dernier apparaître à l'écran. Nous allons utiliser un `v-dialog` pour afficher une carte au dessus de l'interface, en fonction d'une variable booléenne `montrerInfos`. Vous placerez la balise `v-dialog` comme étant un enfant du conteneur principal, en sachant que cet élément n'agira pas sur la taille de son parent et l'organisation des autres éléments :

```
1 <v-dialog v-model="montrerInfos" width="1000" class="ma-0 pa-0">
2   <!--contenu de l'overlay -->
3 </v-dialog>
```