Site monopage (SPA) avec Laravel et Vue js

1. Une application monopage (ou SPA pour *Single-Page Application*) est une application web ou un site web qui se charge une seule fois, et où toutes les interactions avec l'utilisateur se déroulent sans recharger complètement la page.

2. **Vue.js** sera utilisé dans l'exemple présenté, il faut configurer les dépendances et les outils nécessaires.
   2.1 Vérification de Node.js et npm, sinon les installer :

```
PS C:\Ampps\www\4205H6_laravel\version\Blog5h6_v005mono> node -v
v20.10.0
PS C:\Ampps\www\4205H6_laravel\version\Blog5h6_v005mono> npm --v
10.2.3
```

   2.2 Installation des dépendances **Vue.js** nécessaires avec Laravel (voir les dépendances nécessaires pour react)

   a. Selon le cas installer **ui vue ou ui react** (vous avez choisis l'un des deux lors de l'installation du système d'authentification)

php artisan ui vue
php artisan ui react

   b. Installer les dépendances suivantes

npm install (pour la gestion des assets frontend (compilation CSS, JavaScript, etc.). Mix s'appuie sur Webpack et est configuré dans le fichier webpack.mix.js de votre projet.)

npm install vue (installer ou mettre à jour Vue.js manuellement en fonction des besoins de votre projet)
si il y a des erreurs d'installation exécuter dans l'ordre :

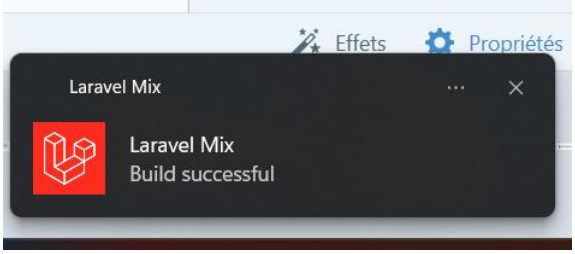
npm audit fix –force
npm install vue-router@next

npm install vue-axios –save (C'est un paquet qui lie Axios avec Vue.js, ce qui facilite l'utilisation d'Axios dans vos composants Vue. Ce paquet permet d'ajouter Axios globalement à votre application Vue.js pour faciliter les requêtes http vers l'API.)

npm install vue-loader (est un loader Webpack essentiel pour permettre la gestion et la compilation des fichiers .vue. Ces fichiers sont des composants monofichiers (Single File Components - SFC) spécifiques à Vue.js qui regroupent le code HTML (template), CSS (styles), et JavaScript (logique) dans un seul fichier.)

   c. Pour exécuter une application créée avec Laravel et Vue.js il faut exécuter les deux commandes suivantes après chaque modification

npm run dev ou watch (il faut s'assurer que le mix compile avec sucée pour pouvoir exécuter la prochaine commande, sinon il faut corriger toutes les erreurs)



Puis executer: php artisan serve

Note: pour ceux qui utilisent laravel9 et plus utilisez les étapes décrites dans le document pour installer et configurer vue et vite. Pour ceci commencez à l'étape 2.

3. Création de la page principale

3.1 Création de **monopage.blade.php**

a. Créer une page à la racine de **ressources/views,** cette page sera la porte d'entrée à l'application monopage.

Nomdelapage.blade.php

b. C'est une page HTML ordinaire avec quelques ajouts important pour l'application **monopage.**

c. Vérification si un utilisateur est connecté ou nom, nous allons voir les étapes numérotées dans la figure ci-dessous

```
<body>
    @if (Auth::check())
        @php
            $user_auth_data = [
                'isLoggedin' => true,
                'user' => Auth::user(),
            ];
        @endphp
    @else
        @php
            $user_auth_data = [
                'isLoggedin' => false,
            ];
        @endphp
    @endif
    <script>
        window.Laravel = JSON.parse(atob('{{ base64_encode(json_encode($user_auth_data) ) }}'));
    </script>
```

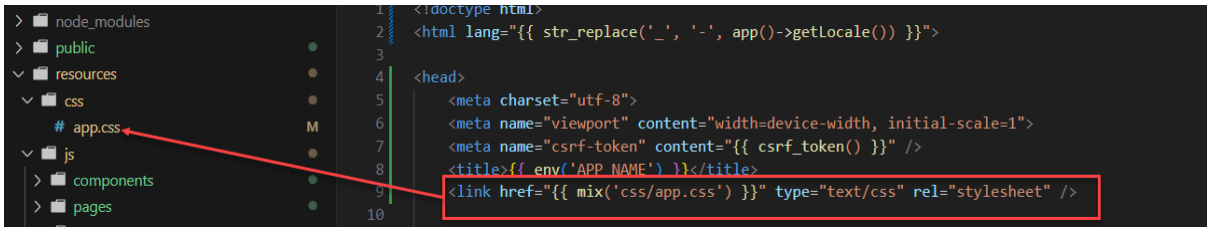
c.1 Vérification si un utilisateur est connecté ou non revoie de **true** ou **false** selon le cas et préparation des données sur l'utilisateur dans une variable JavaScript qui sera ensuite utilisée dans la page web.

c.2 Si l'utilisateur est connecté, une variable ***$user_auth_data*** est définie. Elle contient un tableau associatif avec deux clés :

a. ***'isLoggedin' => true*** : Cela indique que l'utilisateur est connecté.
b. ***'user' => Auth::user()*** : Cela récupère les informations de l'utilisateur authentifié, telles que son nom, son email, son identifiant, etc., et les place dans la clé ***'user'***.

c.3 La ligne de code suivante est utilisée pour passer des données du côté serveur (dans Laravel) vers le côté client (dans JavaScript) de manière sécurisée et compacte en utilisant **Base64**.

d. Dans **head**, inclure le fichier CSS compilé dans votre application web, en utilisant **Laravel Mix**.



e. Dans **body**, inclure le fichier JavaScript compilé dans votre application web en utilisant **Laravel Mix**.

4. Création de la route qui mène vers la page **monopage.blade.php** dans web.php

4.1 Laravel permet de capturer toutes les requêtes et de rediriger l'utilisateur vers la vue **monopage**, ce qui est typique dans les **applications monopages (SPA)**.
4.2 Laravel retourne la même vue pour toutes les URLs, et c'est ensuite le JavaScript côté client (par exemple Vue.js ou React) qui prend en charge le routage et le rendu du contenu en fonction de l'URL demandée.

a. **Route::get('{any}', function () { ... })**
a.1 Cette ligne crée une **route de type GET** qui capte toutes les requêtes HTTP GET. {any}' est un paramètre de route dynamique qui capture **n'importe quelle partie de l'URL** après la racine de l'application. Cela signifie que, peu importe l'URL que l'utilisateur entre, Laravel redirigera toujours vers cette route.
b. **return view('monopage');**
Lorsque la route est correspondue, elle retourne la vue **monopage**. monopage fait généralement référence à un fichier de vue Blade, comme monopage.blade.php. C'est cette vue qui sera envoyée à l'utilisateur.
c. **->where('any', '.*');**
Cette partie permet de définir une **expression régulière** qui spécifie ce qui peut correspondre au paramètre {any}. 'any' est le nom du paramètre de la route et '.*' est l'expression régulière qui signifie "tous les caractères, zéro ou plusieurs". Autrement dit, cette partie de la route peut correspondre à n'importe quelle URL, qu'il s'agisse de /home, /about, /dashboard, ou même /blog/123.

5. Configuration de la partie js
5.1 S'assurer que **.vue** et **.sass** sont configurés dans **webpack.mix.js** qui se trouve à la racine de votre projet

```
13
14  mix.js('resources/js/app.js', 'public/js')
15      .vue()
16      .sass('resources/sass/app.scss', 'public/css');
17
```

5.2 Configurer le fichier **ressources/css/app.css** (est joint sur Moodle)
5.3 Tout les reste se déroulera dans le répertoire **ressources/js.**

a. Vérifier la configuration de **ressources/js/bootstrap.js**
b. Création du gabarit **App.vue** des composantes vue.js dans
c. Création des différentes composantes
d. Création des routes vers les composantes
e. Importer : le gabarit **App.vue**, le répertoire des routes ainsi que les dépendances nécessaires dans **app.js**

```
import { createApp } from "vue"

require('./bootstrap')
import App from "./App.vue"
import axios from 'axios'
import router from './router'

const app = createApp(App)
app.config.globalProperties.$axios = axios;
app.use(router)
app.mount('#app')
```