Partie 3 : Interface utilisateur avec VueJS pour l’élaboration des CRUDs

Installer vue et vite vue-plugin

Dans l’invite de commande au niveau du projet créé taper :

npm i

npm install vue@next vue-loader@next

npm i @vitejs/plugin-vue

MAJ de vite.config.js



import { defineConfig } from 'vite';

import laravel from 'laravel-vite-plugin';

import vue from '@vitejs/plugin-vue'

export default defineConfig({

    plugins: [

        vue(),

        laravel({

            input: ['resources/css/app.css', 'resources/js/app.js'],

            refresh: true,

        }),

    ],

});

## Créer une route dynamique



Route::get('/{any}', function () {

    return view('layout.app');

})->where('any', '.\*');

Créer app.blade.php

Sous resources/views créer le dossier layout

Sous resources/views/layout créer le fichier app.blade.php



<!DOCTYPE html>

<html lang="{{ str\_replace('\_', '-', app()->getLocale()) }}">

<head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Application</title>

</head>

<body>

    <div id="app"></div>

    @vite('resources/js/app.js')

</body>

</html>

**Explication** :

<html lang="{{ str\_replace('\_', '-', app()->getLocale()) }}">

Cette ligne de code génère l'attribut **lang** de la balise **<html>** en fonction de la locale actuellement configurée pour l'application, en remplaçant les tirets bas par des tirets pour se conformer aux normes de codification des langues. Cela permet aux navigateurs et aux moteurs de recherche de connaître la langue principale de la page.

@vite('resources/js/app.js')

Cette ligne de code indique à Vite d'intégrer le fichier JavaScript **'resources/js/app.js'** dans le processus de construction de l'application web.

Installer les dépendances

npm i bootstrap jquery

npm install vue-router vue-axios

Créer un composant

Sous resources/js, créer le dossier components puis le fichier accueil.vue



<template>

    <div class="container">

        <div class="row justify-content-center">

            <div class="col-md-8">

                <div class="card">

                    <div class="card-header"  style="background-color: #8ac0db;">E-Books</div>

                    <div class="card-body"  style="background-color: #b6c0b6;">

                    HOME Page

                    </div>

                </div>

            </div>

        </div>

    </div>

</template>

<script>

    export default {

        mounted() {

            console.log('Component mounted.')

        }

    }

</script>

Créer routes.js

Sous resources/js créer le fichier routes.js



import Accueil from './components/accueil.vue';

export const routes = [

    {

        name: 'accueil',

        path: '/',

        component: Accueil

    }

];

Créer App.vue

Sous resources/js créer le fichier App.vue



<template>

    <div class="container">

        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

            <div class="collapse navbar-collapse">

                <div class="navbar-nav">

                    <router-link to="/" class="nav-item nav-link">Accueil</router-link>

                </div>

            </div>

        </nav>

        <router-view> </router-view>

    </div>

</template>

<script>

    export default {}

</script>

MAJ resources/js/app.js



import { createApp } from 'vue';

import { createRouter, createWebHashHistory } from "vue-router";

import { routes } from "./routes";

import 'bootstrap/dist/css/bootstrap.min.css'

import App from './App.vue'

let app = createApp(App)

const router = createRouter({

    history: createWebHashHistory(),

    routes: routes,

})

app.use(router);

app.mount("#app")

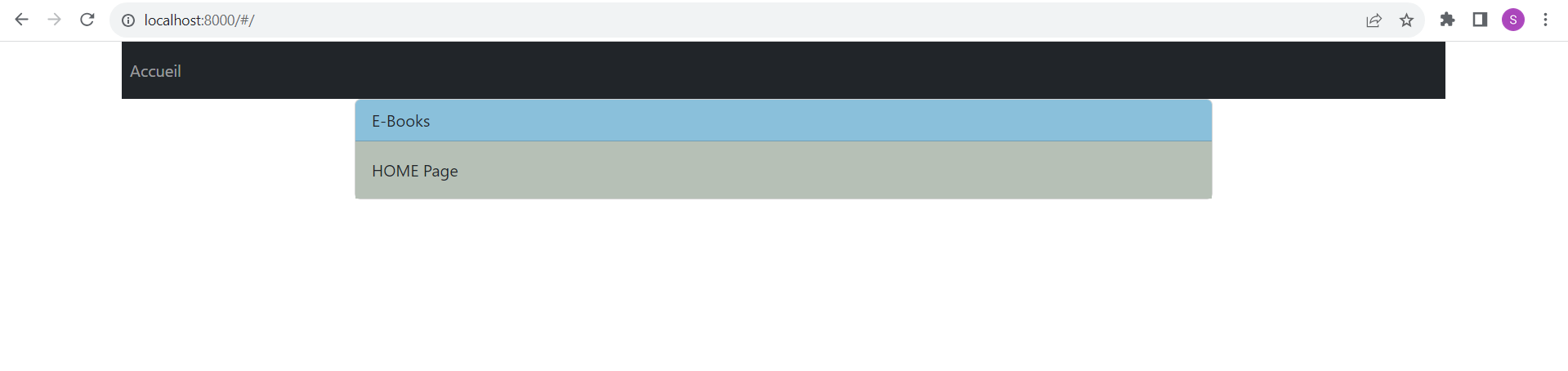
Exécuter

Dans 2 terminaux taper ;

npm run dev

php artisan serve

<http://localhost:8000>



Configuration

On va créer dans resources/js le fichier de configuration pour axios avec l’url du backend.

Créer les dossiers components sous resources/js

Puis le dossier config sous components.

Enfin créer le fichier axios.js



import axios from 'axios';

axios.defaults.baseURL = 'http://localhost:8000';

export default axios;

Affichage des livres

Pour l’affichage des données on va utiliser « DataTables for jQuery with styling for DataTables ».

Visiter :

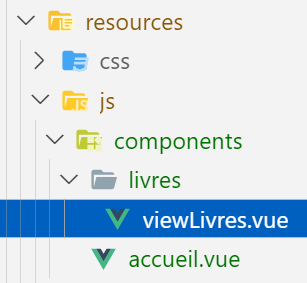
<https://www.npmjs.com/package/datatables.net-dt>

Installer le package :

npm i datatables.net-dt

Pour rappel jquery et bootstrap sont déjà installés.

Créer :



<template>

    <div>

        <h2 class="text-center"> Liste des livres </h2>

        <table class="table" id="example">

            <thead>

            <tr>

          <th>isbn</th>

          <th>Titre</th>

          <th>Année</th>

          <th>Prix</th>

          <th>Quantité</th>

          <th>Couverture</th>

          <th>Spécialité</th>

          <th>Editeur</th>

          <th>Auteurs</th>

          <th>Actions</th>

        </tr>

            </thead>

            <tbody>

            <tr v-for="l in Livres" :key="l.id">

                <td>{{l.isbn}}</td>

                <td>{{l.titre}}</td>

                <td>{{l.annedition}}</td>

                <td>{{l.prix}}</td>

                <td>{{l.qtestock}}</td>

                <td><img :src="l.couverture" :alt=l.titre width="120" /></td>

                <td>{{l.specialite.nomspecialite}}</td>

                <td>{{l.editeur.maisonedit}}</td>

                <td>

                    <div v-for="a in l.auteurs" :key="a.id">

                    {{a.nomauteur}}

                  </div>

                </td>

                <td>

                    <button @click.prevent="deleteLivre(l.id)" class="btn btn-danger">Supprimer</button></td>

            </tr>

            </tbody>

        </table>

    </div>

</template>

<script setup>

import axios from "../config/axios.js";

//Bootstrap and jQuery libraries

import 'bootstrap/dist/css/bootstrap.min.css';

import 'jquery/dist/jquery.min.js';

//Datatable Modules

import "datatables.net-dt/js/dataTables.dataTables"

import "datatables.net-dt/css/jquery.dataTables.min.css"

import $ from 'jquery'

import { ref, onMounted } from 'vue';

const Livres = ref([]);

 onMounted(() => {

            getLivres();

            }

);

const getLivres=()=>{

                  axios

                .get('/api/livres/')

                .then(response => {

                                 Livres.value = response.data;

                                 $(function() {$('#example').DataTable();});

                });

            }

const deleteLivre=(id)=> {

                 if (window.confirm("Etes-vous sûr de vouloir supprimer ?")) {

                   axios

                    .delete(`/api/livres/${id}`)

                    .then(() => {

                        let i = Livres.value.map(data => data.id).indexOf(id);

                        Livres.value.splice(i, 1)

                    })

                    .catch(error => {

                                console.log(error)

                            });

                 }

            }

</script>

Réglage des routes

Dans routes.js ajouter la route correspondante à la liste des articles.



import ViewLivres from './components/livres/viewlivres.vue';

{

        name: 'viewLivres',

        path: '/livres',

        component: ViewLivres

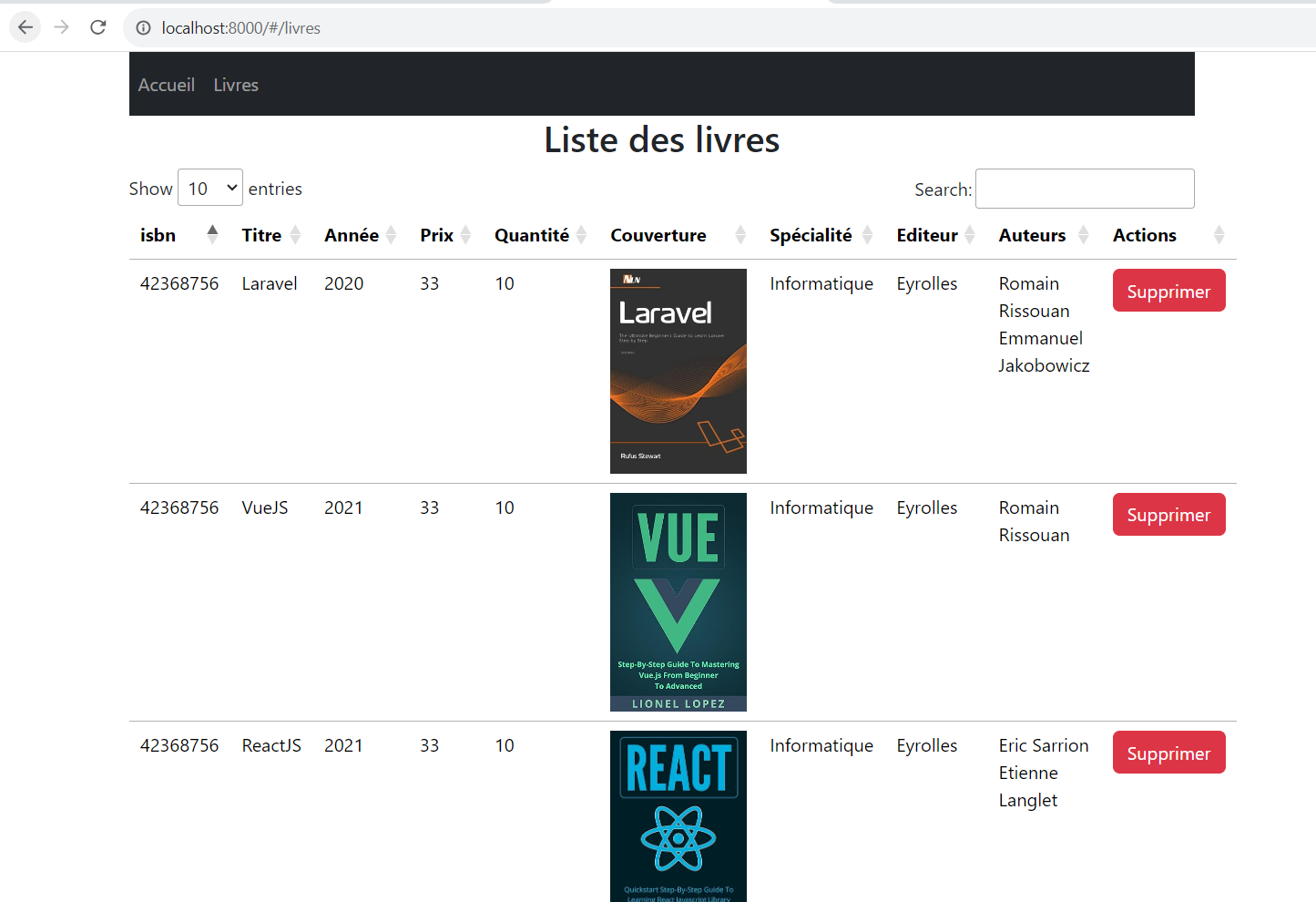
    },

Mise à jour du menu

Ajouter le lien vers le composant viewArticle dans App.vue



  <router-link to="/livres" class="nav-item nav-link">  Livres </router-link>

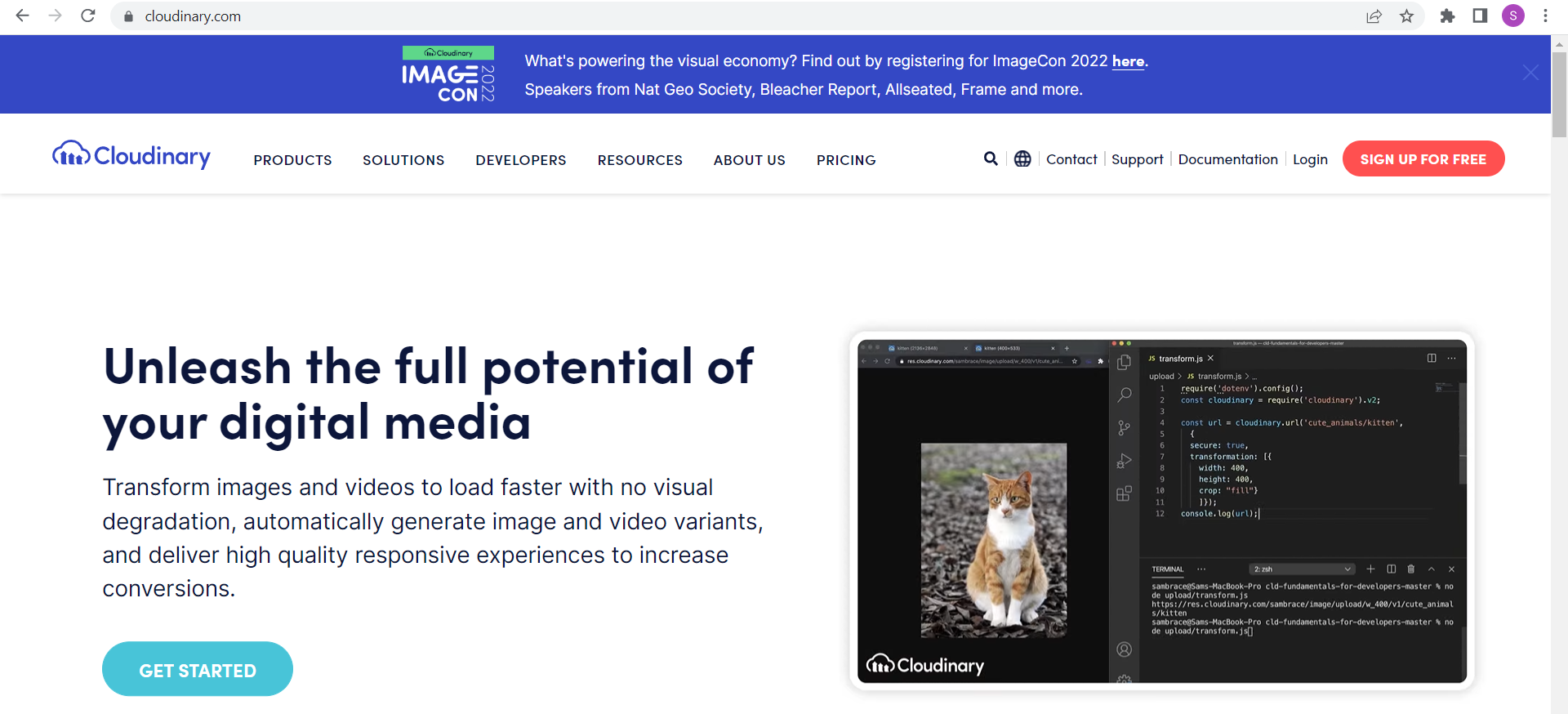


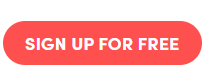
Cloudinary

Par la suite on va utiliser Cloudinary pour le storage des images.

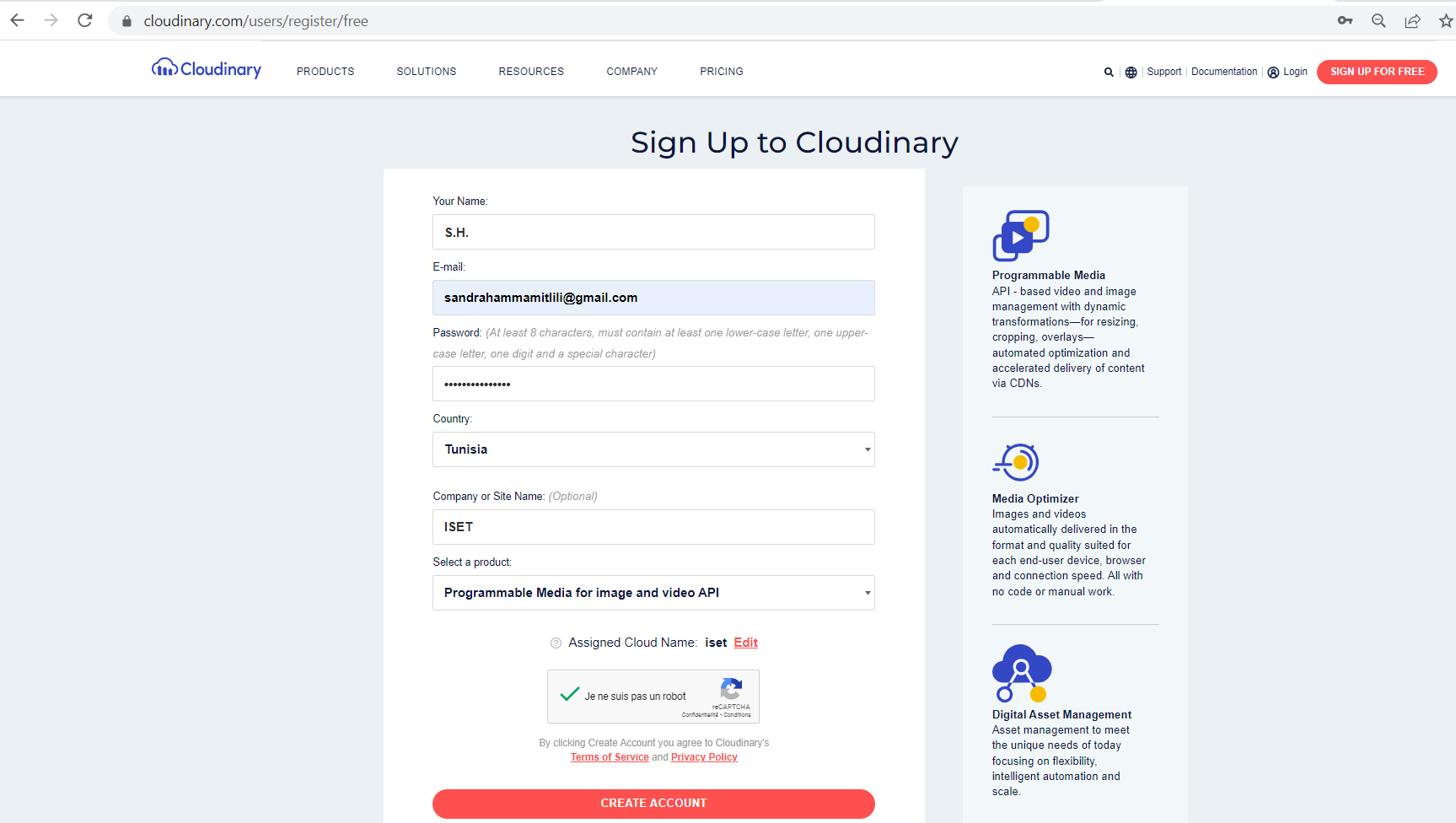
Site :

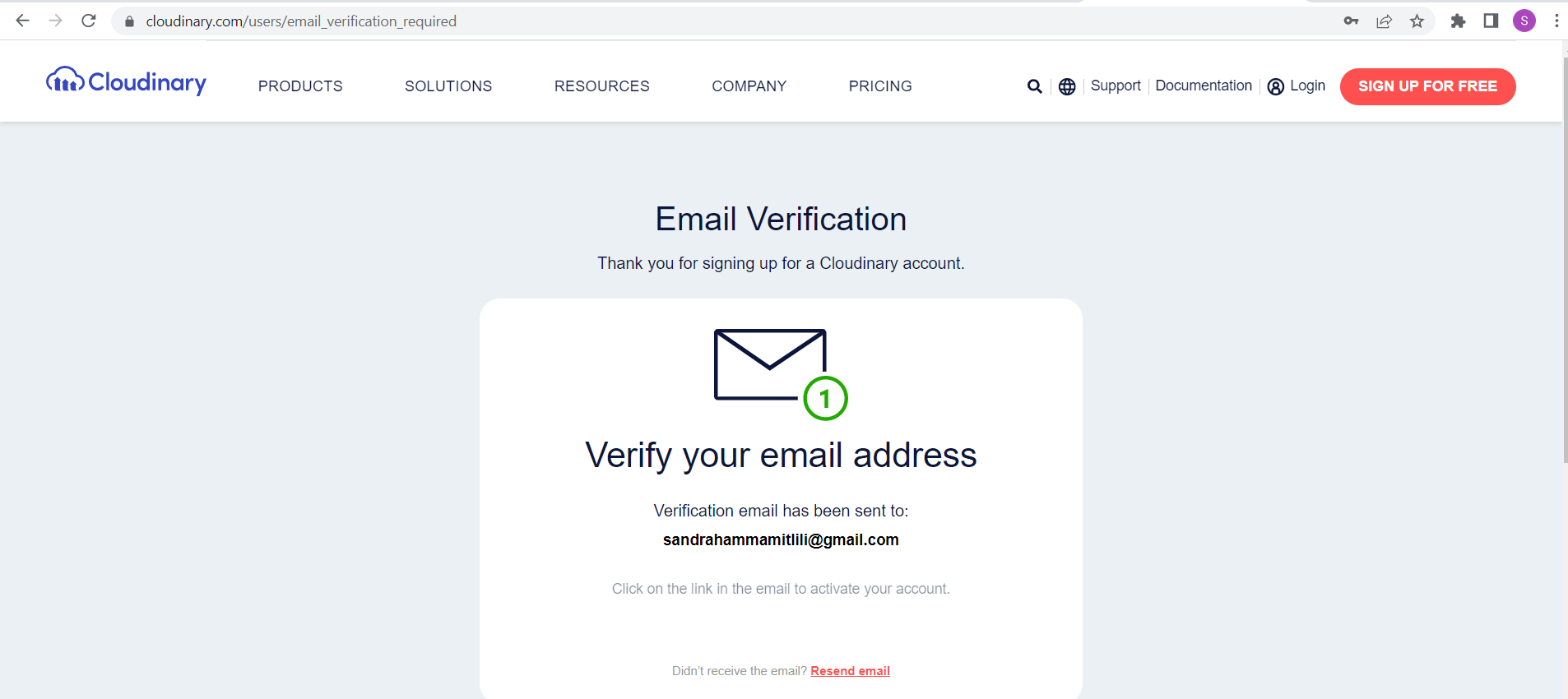
<https://cloudinary.com/>

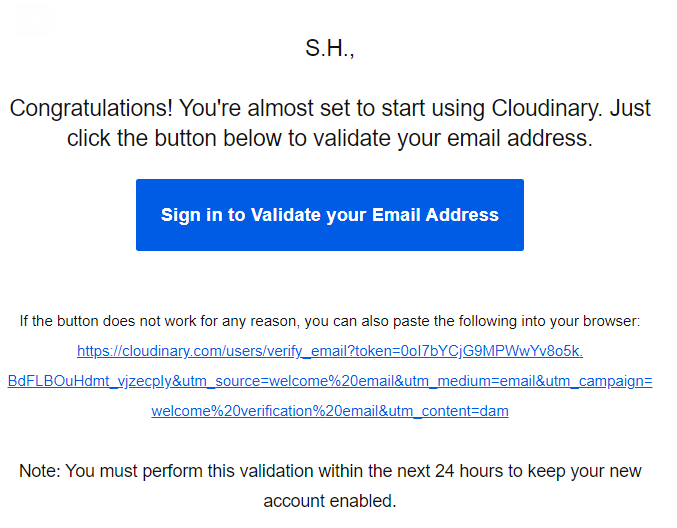


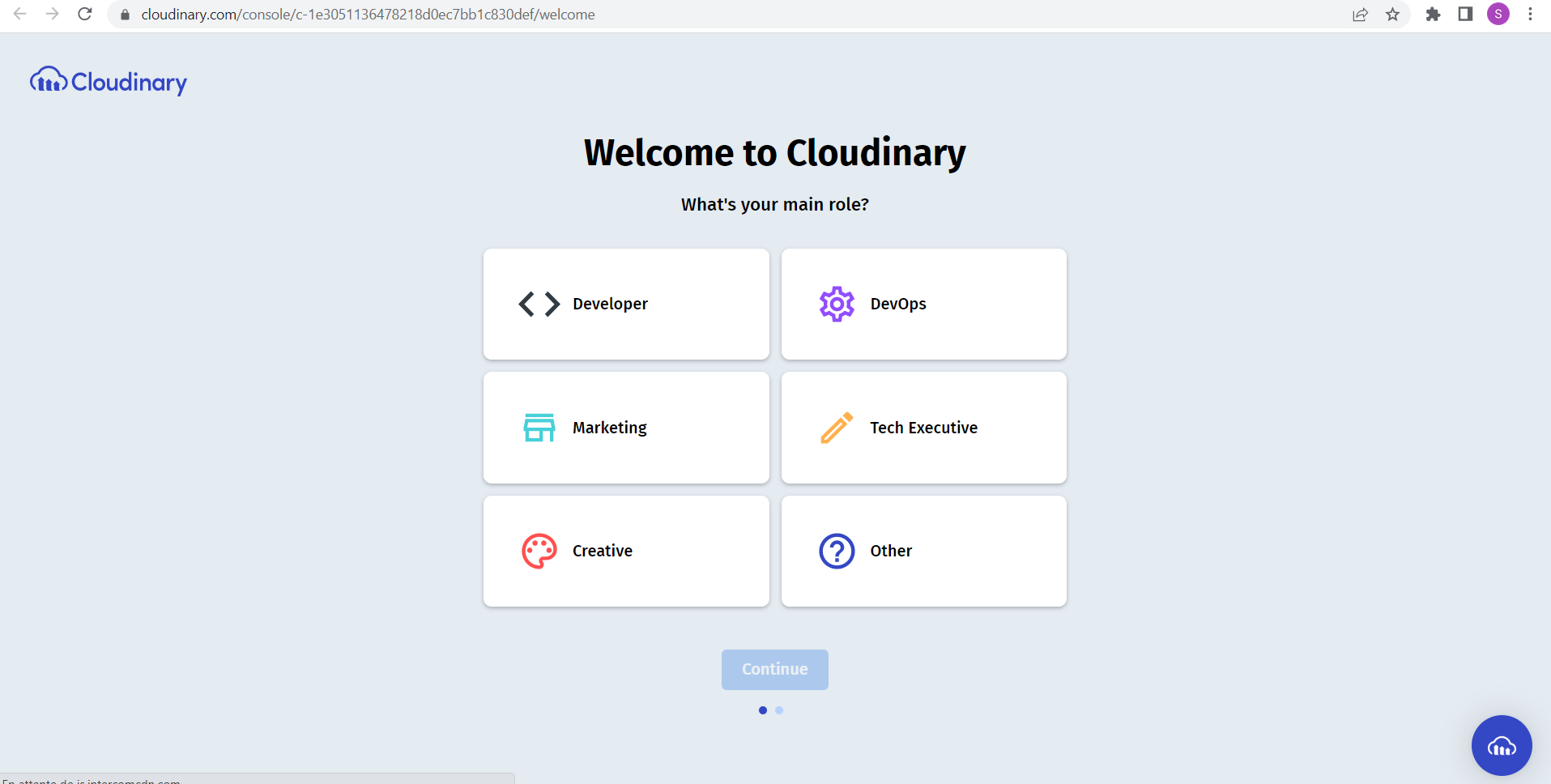


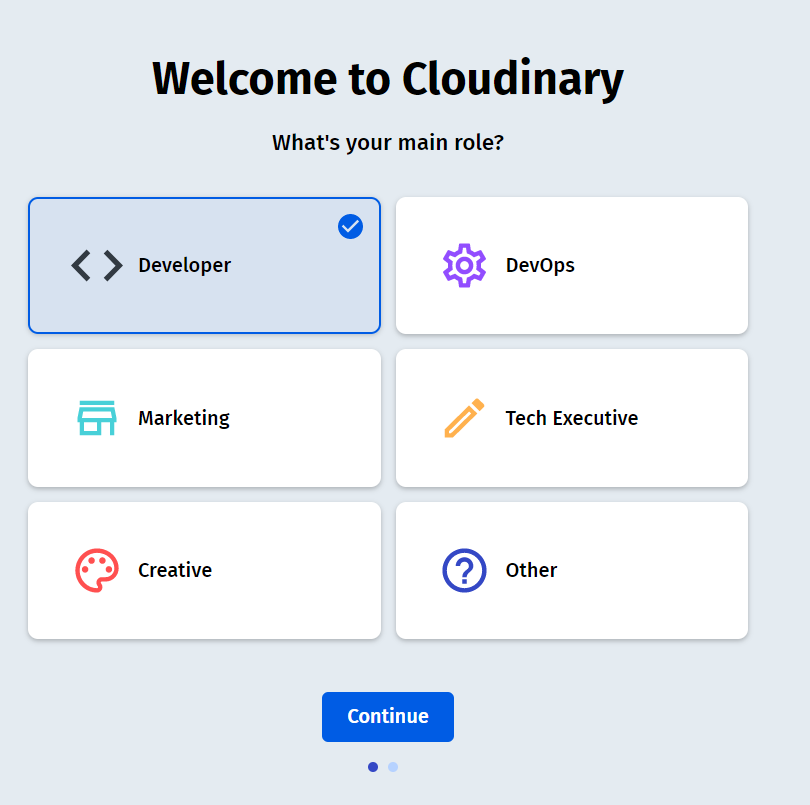
Mot de passe Cloudinary\_2022

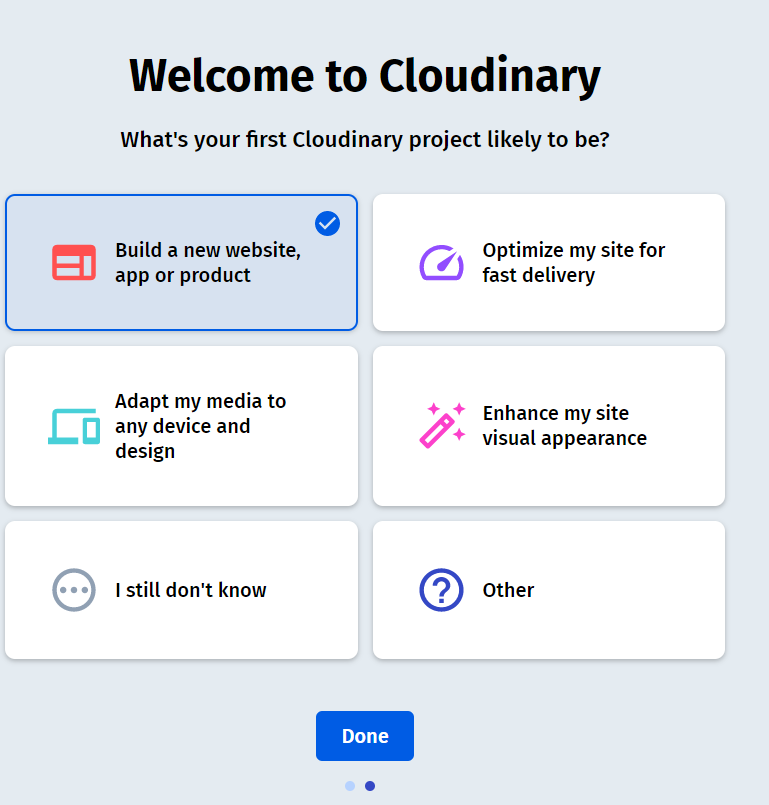


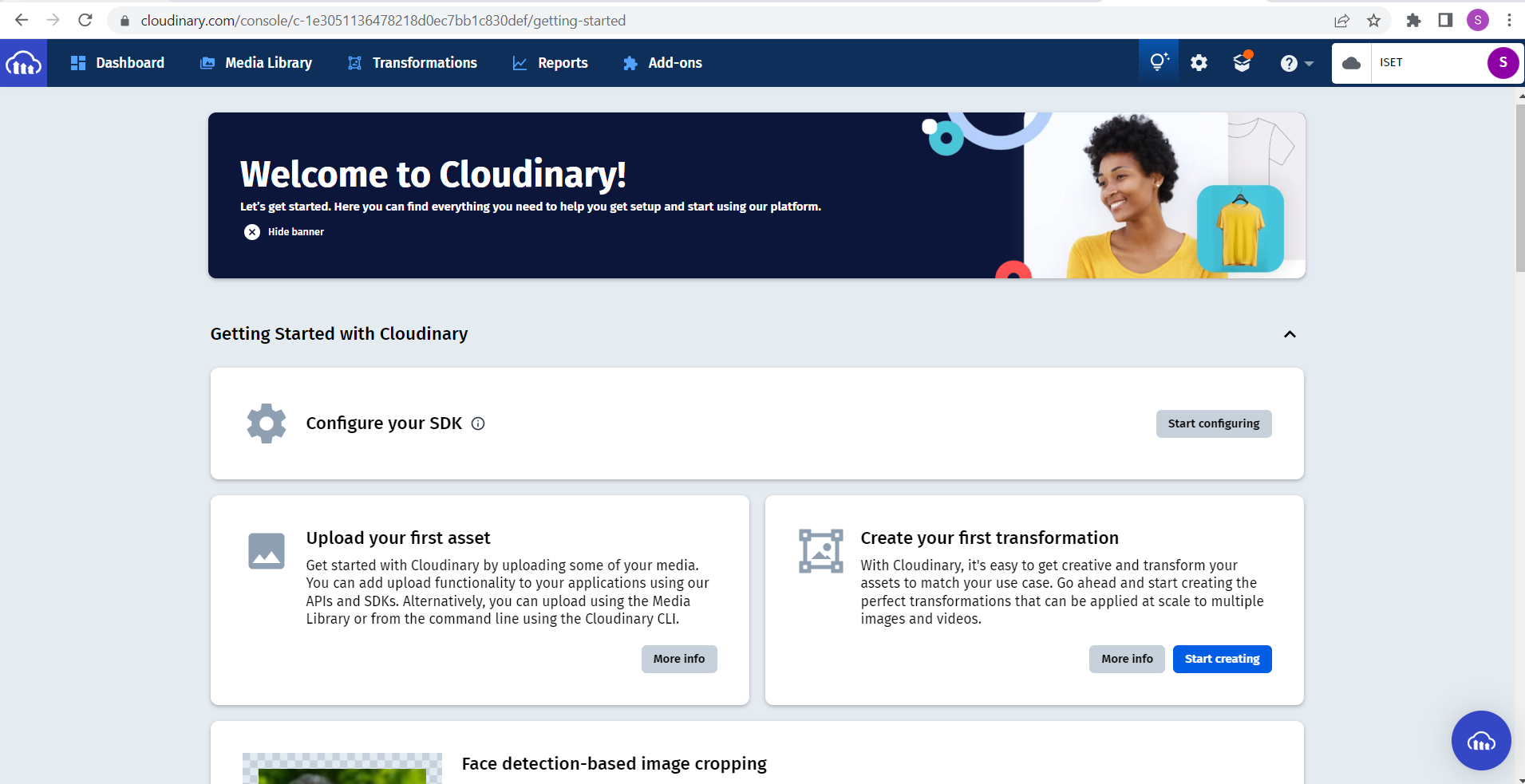


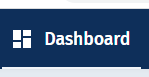


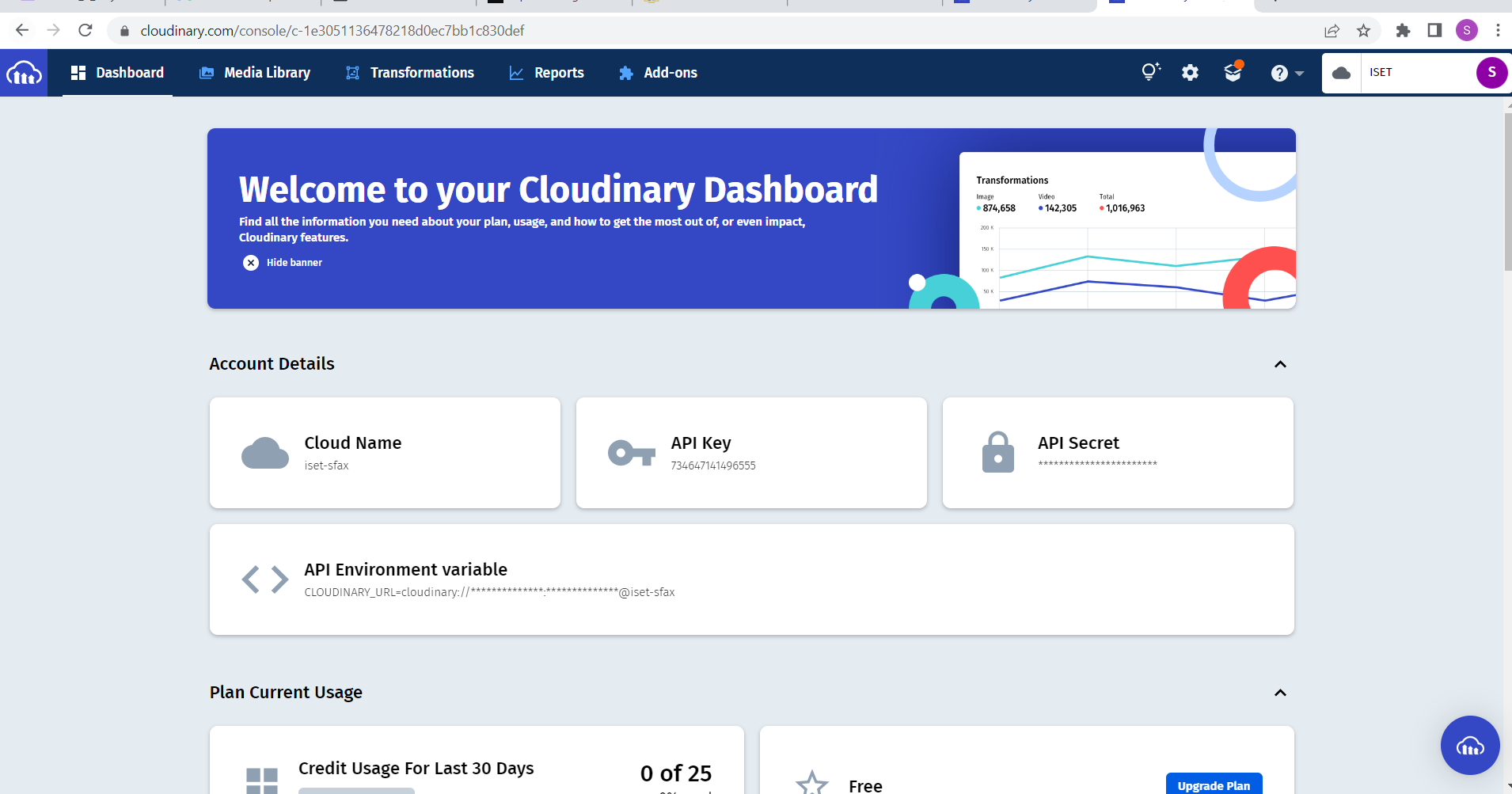


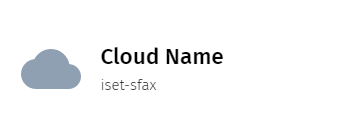






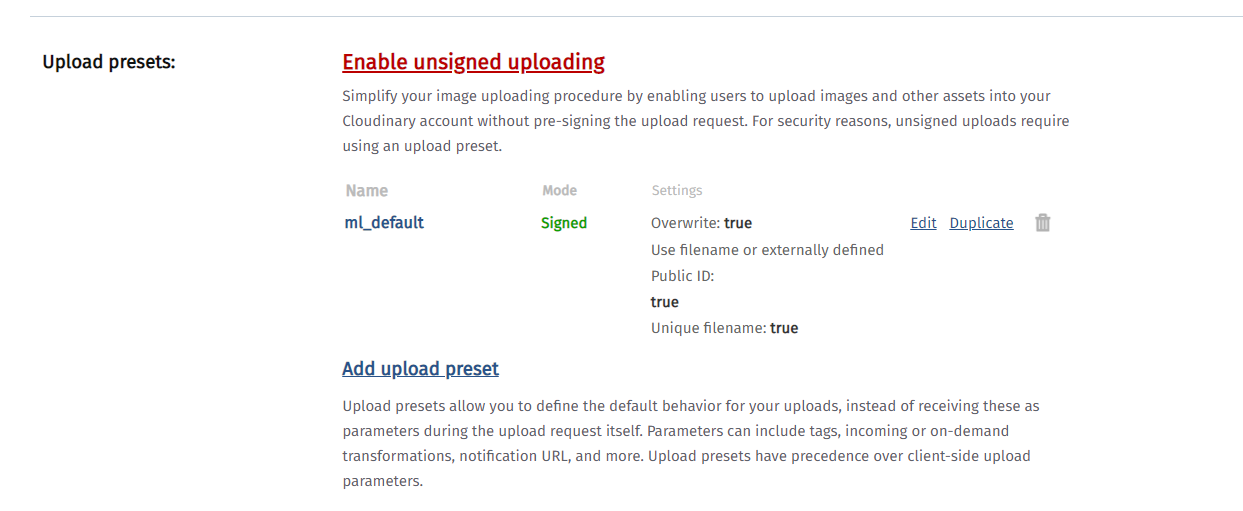


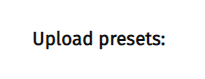




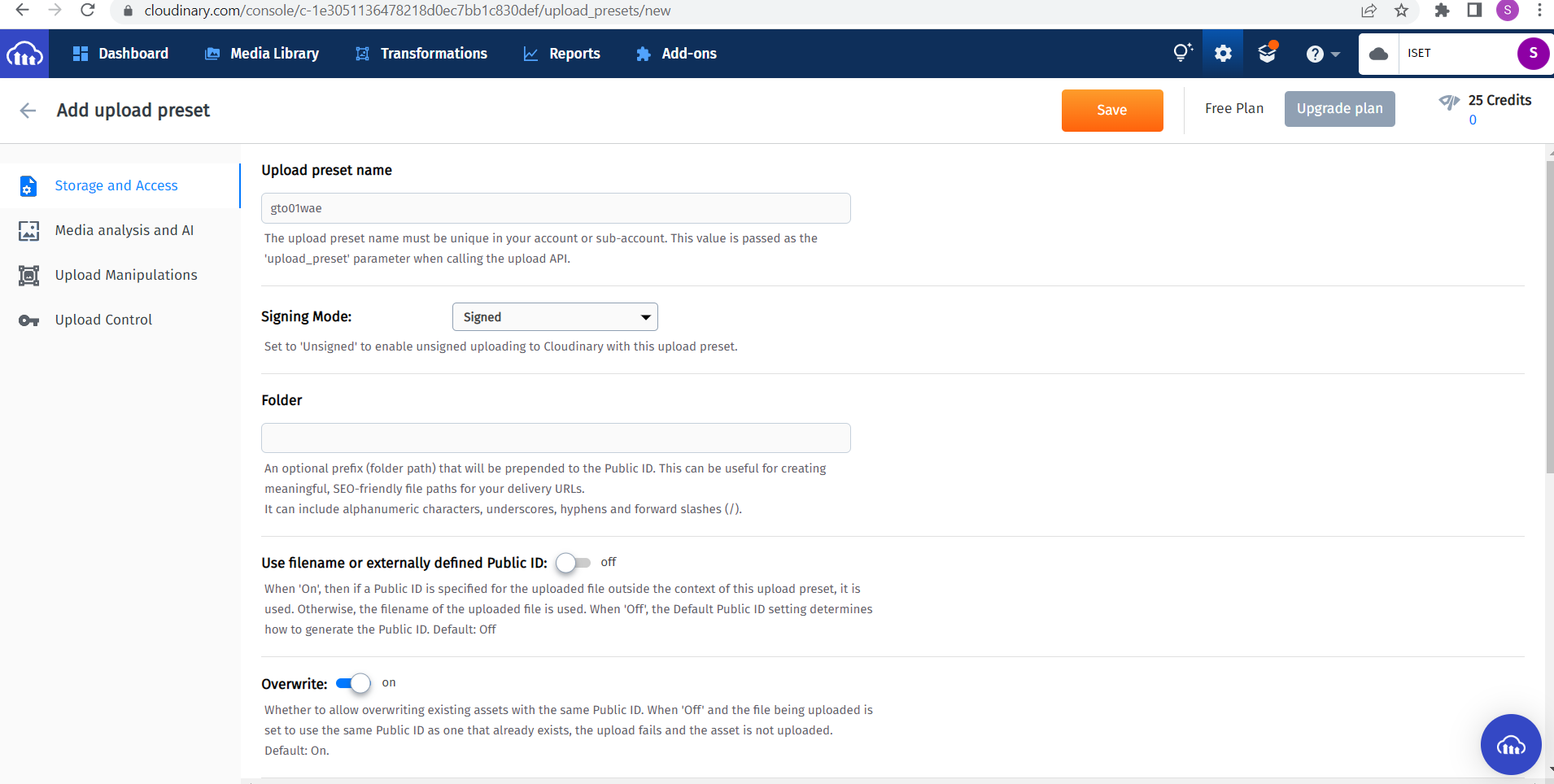


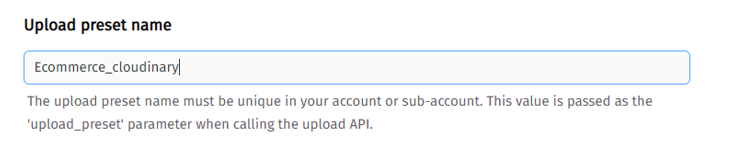


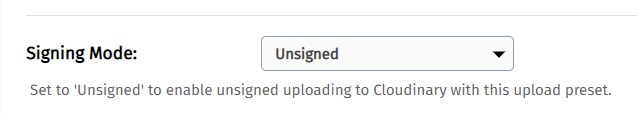


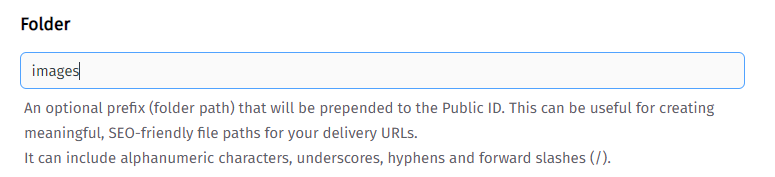


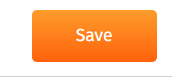




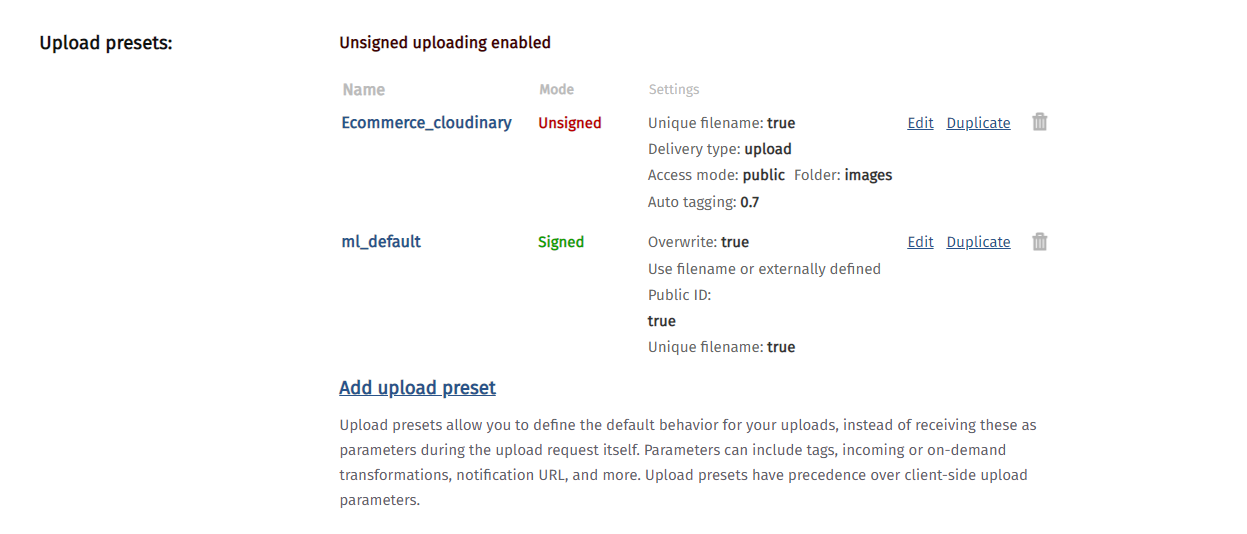


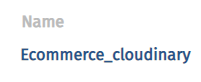












Filepond

Par la suite on va installer les packages relatifs à Filepond.

FilePond prend en charge les téléchargements de fichiers uniques et multiples. Ce package garde une trace de tous les fichiers téléchargés et fournit une interface plus simple pour que l'utilisateur puisse interagir avec les fichiers.

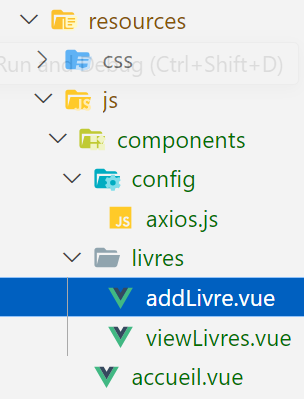
Vue FilePond est un composant adaptateur pratique pour FilePond, une bibliothèque JavaScript qui peut télécharger tout ce que vous lui lancez, optimise les images pour des téléchargements plus rapides et offre une expérience utilisateur exceptionnelle, accessible et fluide.

Faire les installations suivantes :

npm install vue-filepond filepond

npm install filepond-plugin-image-preview

addLivre.vue



<template>

    <div>

      <form @submit.prevent="ajouterproduit">

       <div class="form-group">

       <input type="text" class="form-control" placeholder="isbn" v-model="livre.isbn" require >

     </div>

     <div class="form-group">

       <input type="text" class="form-control" placeholder="titre" v-model="livre.titre" require>

     </div>

     <div class="form-group">

       <input type="number" class="form-control" placeholder="annedition" v-model="livre.annedition">

     </div>

    <div class="form-group">

       <input type="number" class="form-control" placeholder="prix" v-model="livre.prix">

     </div>

    <div class="form-group">

       <input type="number" class="form-control" placeholder="qtestock" v-model="livre.qtestock">

     </div>

    <div class="form-group">

       Spécialité<select class="form-control" v-model="livre.specialite\_id">

          <option v-for="sc in Specialites" :key="sc.id" :value=sc.id>{{sc.nomspecialite}}</option>

       </select>

       </div>

       <div class="form-group">

       Edition <select class="form-control" v-model="livre.editeur\_id">

          <option v-for="ed in Editeurs" :key="ed.id" :value=ed.id>{{ed.maisonedit}}</option>

       </select>

       </div>

       <div class="form-group">

       Auteurs <select class="form-control" v-model="livre.auteur\_ids" multiple>

          <option v-for="aut in Auteurs" :key="aut.id" :value=aut.id>{{aut.nomauteur}}</option>

       </select>

       </div>

        <!-- FilePond pour le téléchargement d'images -->

        <div class="form-group">

          <div>

            <file-pond

              name="test"

              ref="pond"

              class-name="my-pond"

              label-idle="Drop files here..."

              allow-multiple="false"

              accepted-file-types="image/jpeg, image/png"

              v-bind:files="myFiles"

              v-on:init="handleFilePondInit"

              :server="serverOptions()"

            />

          </div>

        </div>

        <button type="submit" class="btn btn-block btn-primary">Ajouter Produit</button>

      </form>

    </div>

  </template>

  <script setup>

  /\*

  En utilisant le préfixe : (liaison dynamique), vous indiquez à Vue.js

  d'évaluer la valeur de la propriété serverOptions comme une expression JavaScript plutôt que de considérer

  "serverOptions" comme une URL. Cela devrait résoudre l'erreur "405 (Method Not Allowed)"

  Avec () après serverOptions on appelle réellement la fonction serverOptions pour obtenir les options du serveur

  au lieu de simplement passer la référence de la fonction.

  \*/

  import { ref, onMounted } from 'vue';

  import vueFilePond from 'vue-filepond';

  import 'filepond/dist/filepond.min.css';

  import FilePondPluginImagePreview from 'filepond-plugin-image-preview';

  import 'filepond-plugin-image-preview/dist/filepond-plugin-image-preview.min.css';

  // Create FilePond component

  const FilePond = vueFilePond(FilePondPluginImagePreview);

  const myFiles = ref([]);

  import axios from "../config/axios.js";

  import { useRouter } from 'vue-router';

  const router = useRouter();

  const Specialites = ref([]);

  const getSpecialites = () => {

    axios.get('/api/specialites')

      .then(res => {

        Specialites.value = res.data;

      })

      .catch(error => {

        console.log(error);

      });

  }

  const Editeurs = ref([]);

  const getEditeurs = () => {

    axios.get('/api/editeurs')

      .then(res => {

        Editeurs.value = res.data;

      })

      .catch(error => {

        console.log(error);

      });

  }

  const Auteurs = ref([]);

  const getAuteurs = () => {

    axios.get('/api/auteurs')

      .then(res => {

        Auteurs.value = res.data;

      })

      .catch(error => {

        console.log(error);

      });

  }

  onMounted(() => {

    getSpecialites();

    getEditeurs();

    getAuteurs();

  });

  const livre = {

    isbn: "",

    titre: "",

    annedition: "",

    prix: "",

    qtestock: "",

    couverture: "",

    specialite\_id: "",

    editeur\_id : "",

    auteur\_ids:[]

  };

  const  ajouterproduit=()=>{

     console.log(livre)

     axios.post("/api/livres",livre)

          .then(() => {

                router.push('/livres')})

          .catch(error => {

             console.error("There was an error!", error); alert("There was an error!")})

             }

  const handleFilePondInit = () => {

    console.log('FilePond has initialized');

  }

  const serverOptions = () => { console.log('server pond');

    return {

      process: (fieldName, file, metadata, load, error, progress, abort) => {

        const data = new FormData();

        data.append('file', file);

        data.append('upload\_preset', 'Ecommerce\_cloudinary');

        data.append('cloud\_name', 'iset-sfax');

        data.append('public\_id', file.name);

        axios.post('https://api.cloudinary.com/v1\_1/iset-sfax/image/upload', data)

          .then((response) => response.data)

          .then((data) => {

            console.log(data);

            livre.couverture = data.url;

            load(data);

          })

          .catch((error) => {

            console.error('Error uploading file:', error);

            error('Upload failed');

            abort();

          });

      },

    };

  };

  </script>

  Réglage des routes

Dans routes.js ajouter la route correspondante au formulaire d’ajout.



import AddLivre from "./components/livres/addLivre.vue";

  {

        path: '/addLivre',

        name: 'addLivre',

        component: AddLivre

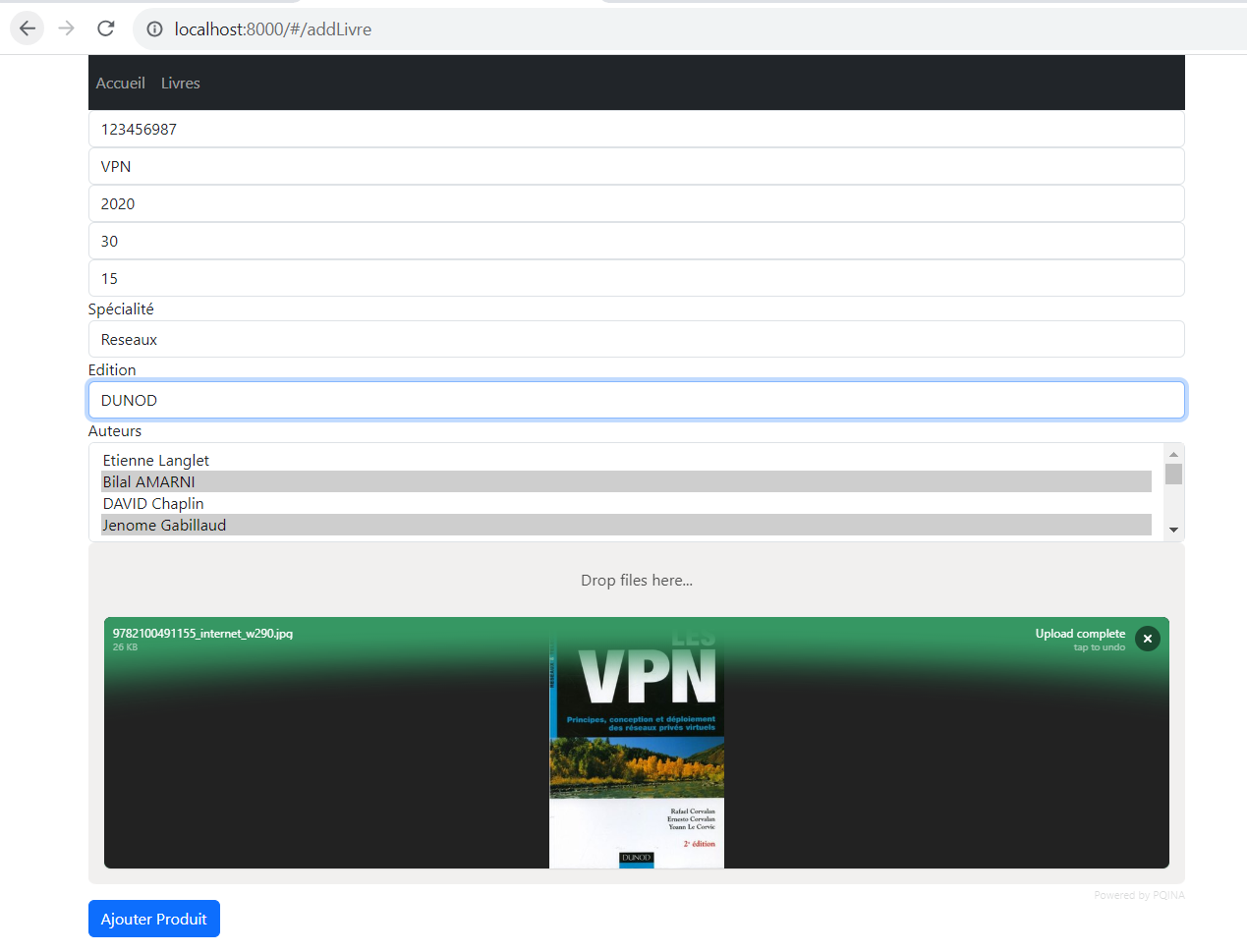
    },

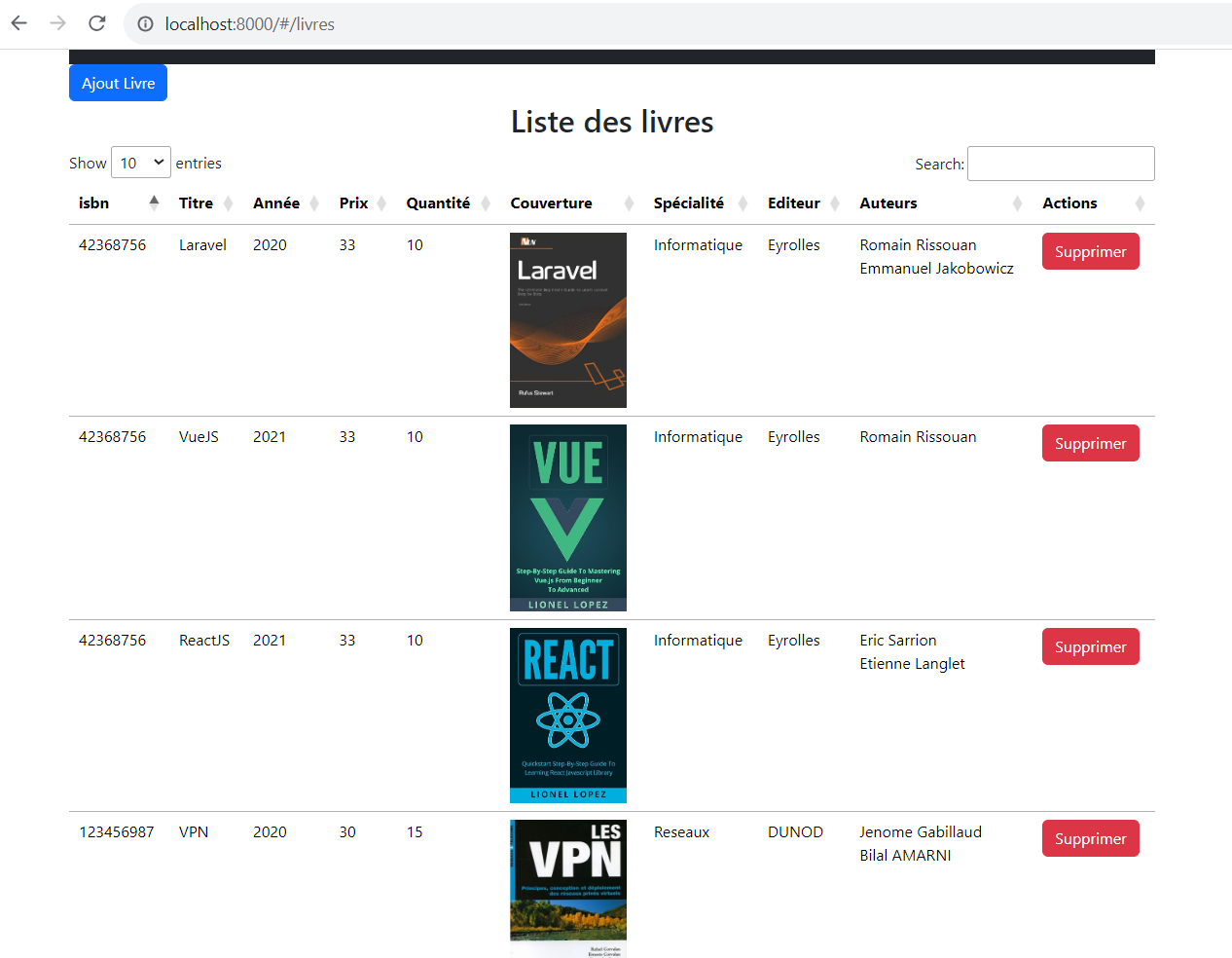
Mise à jour de la liste

Ajouter le code :



  <router-link :to="{name: 'addLivre'}" class="btn btn-primary">Ajout Livre</router-link>





Modification de l’article

Créer le fichier components/livres/editLivre.vue



<template>

    <div>

      <form @submit.prevent="modifierlivre">

       <div class="form-group">

       <input type="text" class="form-control" placeholder="isbn" v-model="livre.isbn" require >

     </div>

     <div class="form-group">

       <input type="text" class="form-control" placeholder="titre" v-model="livre.titre" require>

     </div>

     <div class="form-group">

       <input type="number" class="form-control" placeholder="annedition" v-model="livre.annedition">

     </div>

    <div class="form-group">

       <input type="number" class="form-control" placeholder="prix" v-model="livre.prix">

     </div>

    <div class="form-group">

       <input type="number" class="form-control" placeholder="qtestock" v-model="livre.qtestock">

     </div>

    <div class="form-group">

       Spécialité<select class="form-control" v-model="livre.specialite\_id">

          <option v-for="sc in Specialites" :key="sc.id" :value=sc.id>{{sc.nomspecialite}}</option>

       </select>

       </div>

       <div class="form-group">

       Edition <select class="form-control" v-model="livre.editeur\_id">

          <option v-for="ed in Editeurs" :key="ed.id" :value=ed.id>{{ed.maisonedit}}</option>

       </select>

       </div>

       <div class="form-group">

       Auteurs <select class="form-control" v-model="livre.auteur\_ids" multiple>

          <option v-for="aut in Auteurs" :key="aut.id" :value=aut.id :class="{ optionExist: selectOne(aut.id) }">{{aut.nomauteur}}</option>

       </select>

       </div>

        <!-- FilePond pour le téléchargement d'images -->

        <div class="form-group">

          <div>

            <file-pond

              name="test"

              ref="pond"

              class-name="my-pond"

              label-idle="Drop files here..."

              allow-multiple="false"

              accepted-file-types="image/jpeg, image/png"

              v-bind:files="myFiles"

              :server="serverOptions()"

            />

          </div>

        </div>

        <button type="submit" class="btn btn-block btn-primary">Modifier Produit</button>

      </form>

    </div>

  </template>

  <script setup>

  import { ref, onMounted } from 'vue';

  import vueFilePond from 'vue-filepond';

  import 'filepond/dist/filepond.min.css';

  import FilePondPluginImagePreview from 'filepond-plugin-image-preview';

  import 'filepond-plugin-image-preview/dist/filepond-plugin-image-preview.min.css';

  const FilePond = vueFilePond(FilePondPluginImagePreview);

  import axios from "../config/axios.js";

  import { useRouter,useRoute } from 'vue-router';

    const router = useRouter() ;

    const route = useRoute();

    const Specialites = ref([]);

  const getSpecialites = async() => {

    await axios.get('/api/specialites')

      .then(res => {

        Specialites.value = res.data;

         })

      .catch(error => {

        console.log(error);

      });

  }

  const Editeurs = ref([]);

  const getEditeurs = async () => {

    await axios.get('/api/editeurs')

      .then(res => {

        Editeurs.value = res.data;

      })

      .catch(error => {

        console.log(error);

      });

  }

  const Auteurs = ref([]);

  const getAuteurs = async () => {

    await axios.get('/api/auteurs')

      .then(res => {

        Auteurs.value = res.data;

      })

      .catch(error => {

        console.log(error);

      });

  }

const livre = ref({});

const fetchLivre= async()=> {

        await axios

                .get(`/api/livres/${route.params.id}`)

                .then((res) => {

                    livre.value = res.data;

                    console.log(res.data)

                       console.log("livre.value.auteurs:", livre.value.auteurs);

                })

                .catch((err) => {console.error(err)})

    }

let  myFiles = ref([]);

  onMounted(async() => {

   await getSpecialites();

   await  getEditeurs();

   await  getAuteurs();

   await fetchLivre();

   await handleFilePondInit();

  });

  const handleFilePondInit = async() => {

  if (livre.value.couverture) {

    myFiles.value = [

  {

    source: livre.value.couverture,

    options: { type: 'local' }

  }

  ]

 }

  }

  const serverOptions = () => {

    return {

        load: (source, load, error, progress, abort, headers) => {

                    var myRequest = new Request(source);

                    fetch(myRequest).then(function(response) {

                      response.blob().then(function(myBlob) {

                        load(myBlob);

                      });

                    });

                  },

      process: (fieldName, file, metadata, load, error, progress, abort) => {

        const data = new FormData();

        data.append('file', file);

        data.append('upload\_preset', 'Ecommerce\_cloudinary');

        data.append('cloud\_name', 'iset-sfax');

        data.append('public\_id', file.name);

        axios.post('https://api.cloudinary.com/v1\_1/iset-sfax/image/upload', data)

          .then((response) => response.data)

          .then((data) => {

             livre.value.couverture = data.url;

            load(data);

          })

          .catch((error) => {

            console.error('Error uploading file:', error);

            error('Upload failed');

            abort();

          });

      },

    };

  };

  const  modifierlivre=()=>{

     console.log(livre.value)

     axios.put(`/api/livres/${route.params.id}`,livre.value)

          .then(() => {

                router.push('/livres')})

          .catch(error => {

             console.error("There was an error!", error);

             alert("There was an error!")

            })

}

const selectOne = (autId) => {

  if (livre.value.auteurs) {

    /\* some renvoie vrai si, dans le tableau,

    il trouve un élément pour lequel la fonction fournie renvoie vrai ;

    sinon, il renvoie faux.

    \*/

    return livre.value.auteurs.some((auteur) => auteur.id === autId);

  } else {

    return false;

  }

};

  </script>

<style scoped>

.optionExist {

  background-color: yellow;

}

</style>

**Explication** :

Les données du composant sont gérées à l'aide de l'API Composition de Vue.js (Vue 3). Les propriétés sont déclarées à l'aide de ref. Par exemple, article est une référence à un objet qui stocke les données du produit, et myFiles est une référence à un tableau qui stocke les fichiers téléchargés.

Leurs utilisations requiert .value

La fonction handleFilePondInit est utilisée pour initialiser FilePond. Si le produit a déjà une image (dans la base de données), elle ajoute cette image à la liste myFiles, ce qui permet à FilePond de l'afficher. A la place de l’appeler dans init de filepond il faut l’appeler dans onMounted après fetchLivre pour pouvoir récupérer la valeur de l’image.

La fonction serverOptions contient les options de configuration pour FilePond. Elle spécifie comment les fichiers doivent être traités lors du téléchargement. Les options comprennent la fonction load (qui n’existait pas dans l’ajout de l’article et ajoutée ici dans l’edit), qui permet de charger une image depuis une URL (récupérée de la BD et mise dans myFiles), et la fonction process, qui gère le téléchargement réel vers le service Cloudinary.



Ajouter :

import EditLivre from "./components/livres/editLivre.vue";

{

        name: 'editLivre',

        path: '/editLivre/:id',

        component: EditLivre

    },



Ajouter :

<router-link :to="{name: 'editLivre', params: { id: l.id }}" class="btn btn-success">Edit</router-link>

