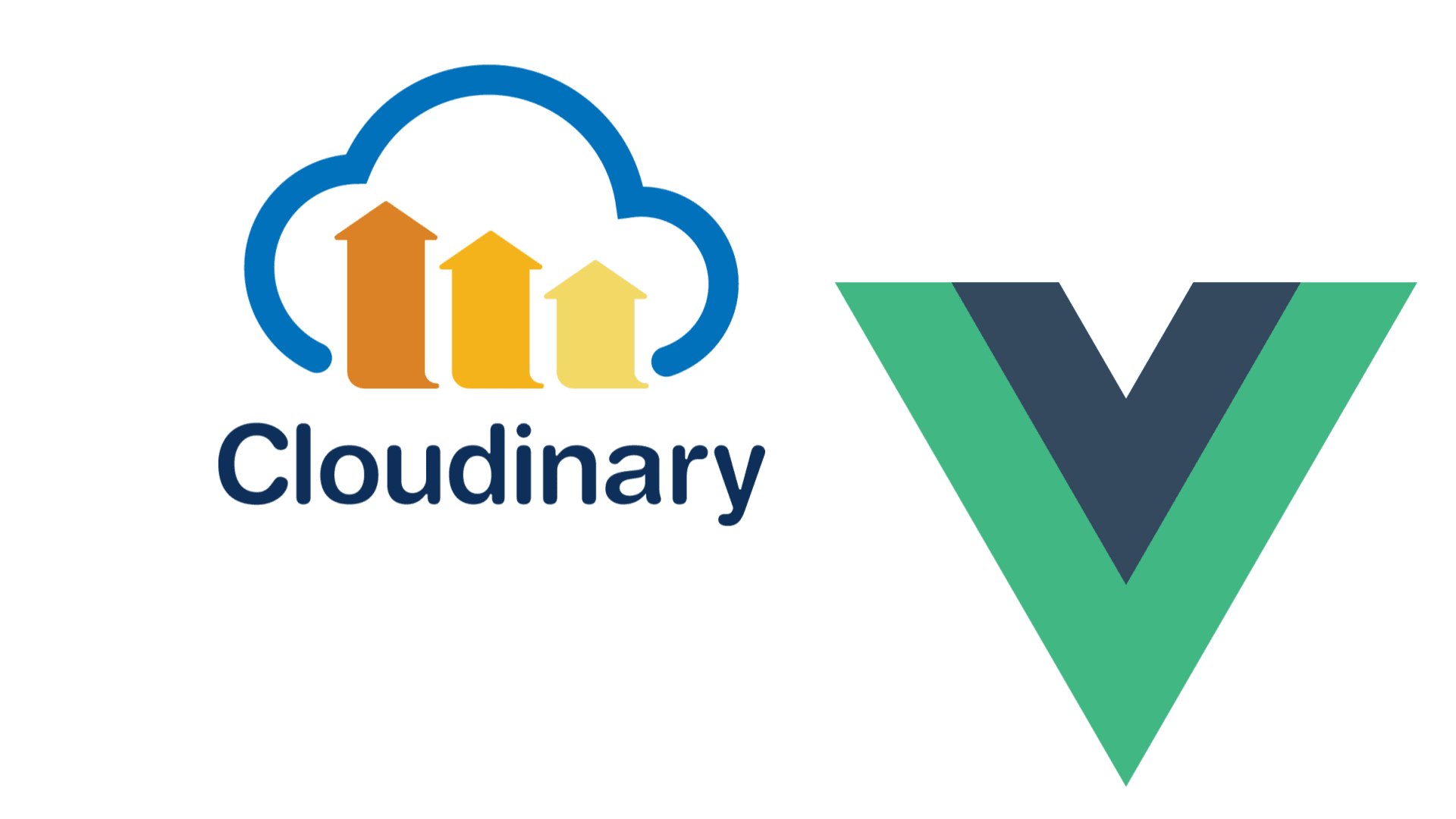
Atelier 4 : CRUD et Upload images



## Liste des articles

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

Visiter :

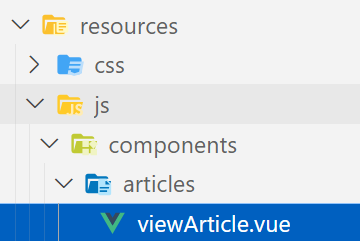
<https://www.npmjs.com/package/datatables.net-dt>

Installer le package :

npm i datatables.net-dt

Pour rappel jquery et bootstrap sont déjà installés.

Créer :



<template>

    <div>

        <div v-if="isLoading">

            <h2> Loading .... </h2>

        </div>

        <div v-else>

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

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

            <thead>

            <tr>

          <th>Référence</th>

          <th>Désignation</th>

          <th>Marque</th>

          <th>Prix</th>

          <th>Quantité</th>

          <th>Sous Catégorie</th>

          <th>Image</th>

          <th>Actions</th>

        </tr>

            </thead>

            <tbody>

            <tr v-for="p in Articles" :key="p.id">

                <td>{{p.reference}}</td>

                <td>{{p.designation}}</td>

                <td>{{p.marque}}</td>

                <td>{{p.prix}}</td>

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

                <td>{{p.scategories.nomscategorie}}</td>

                <td><img :src="p.imageart" :alt=p.designation width="100" /></td>

                <td>

                  <router-link :to="{name: 'editArticle', params: { id: p.id }}" class="btn btn-success">Modifier</router-link>

                  <button @click.prevent="deleteArticle(p.id)" class="btn btn-danger">Supprimer</button>

                </td>

            </tr>

            </tbody>

        </table>

    </div>

</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 Articles = ref([]);

const isLoading = ref(true);

 onMounted(() => {

            getArticles();

            }

);

const getArticles=()=>{

                  axios

                .get('/api/articles/')

                .then(response => {

                    isLoading.value = false;

                                 Articles.value = response.data;

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

                });

            }

const deleteArticle=(id)=> {

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

                   axios

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

                    .then(() => {

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

                        Articles.value.splice(i, 1)

                    })

                    .catch(error => {

                                console.log(error)

                            });

                 }

            }

</script>

On remarque l’utilisation de {{p.scategories.nomscategorie}} pour afficher la sous catégorie correspondante .Et c’est grâce à la relation One to Many qui a été mise en œuvre dans Laravel.

## Réglage des routes

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



import ViewArticle from "./components/articles/viewArticle.vue";

 {

        path: '/articles',

        name: 'articles',

        component: ViewArticle

    },

## Mise à jour du menu

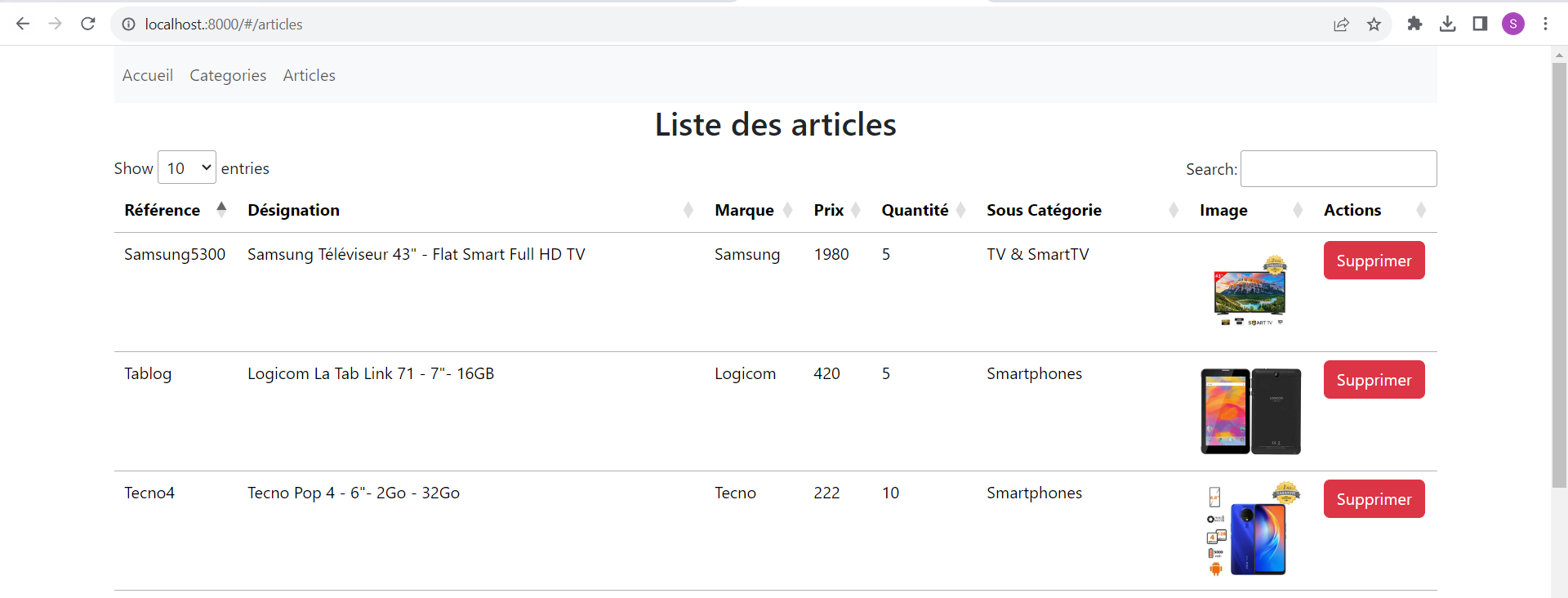
Ajouter le lien vers le composant viewArticle dans App.vue

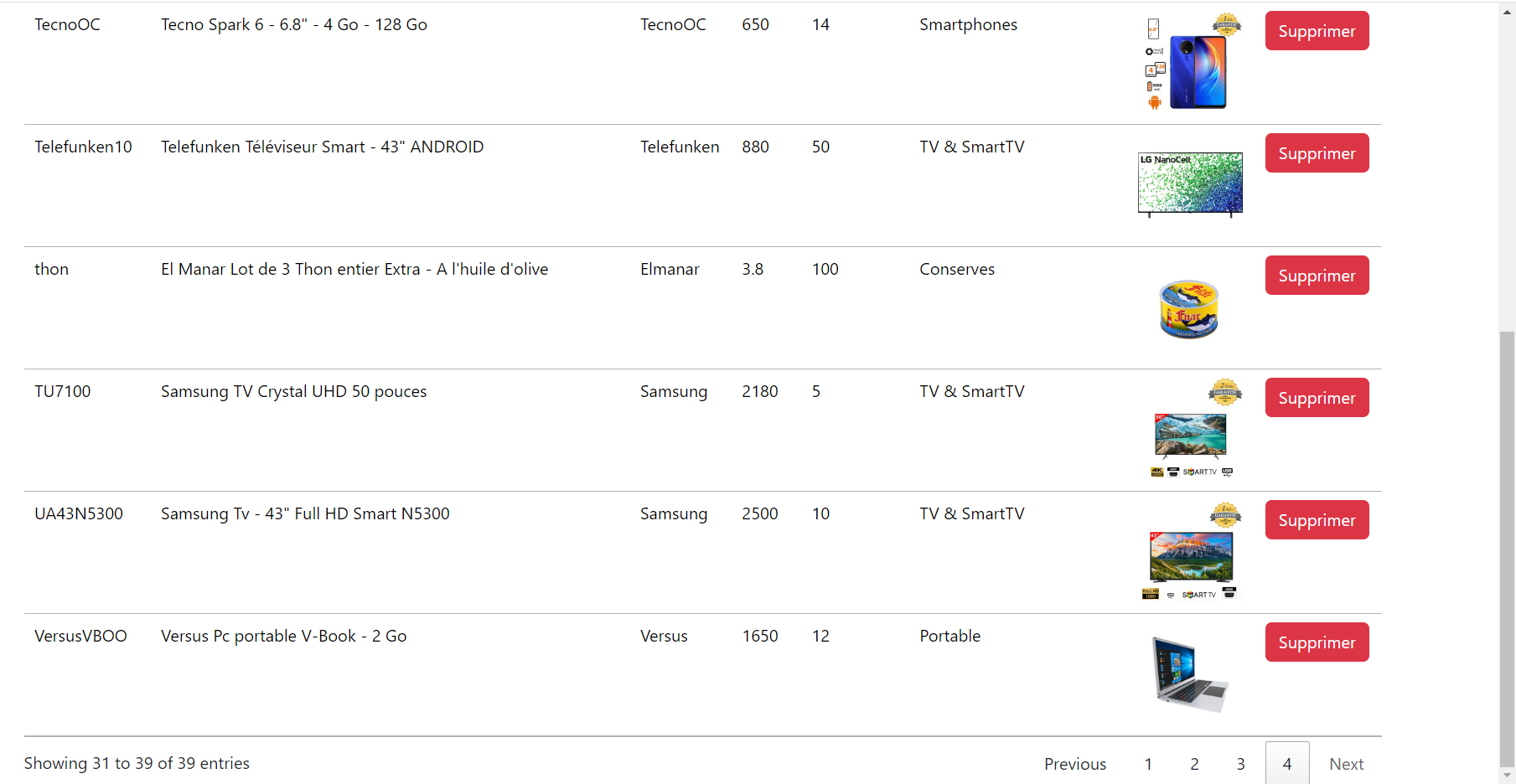


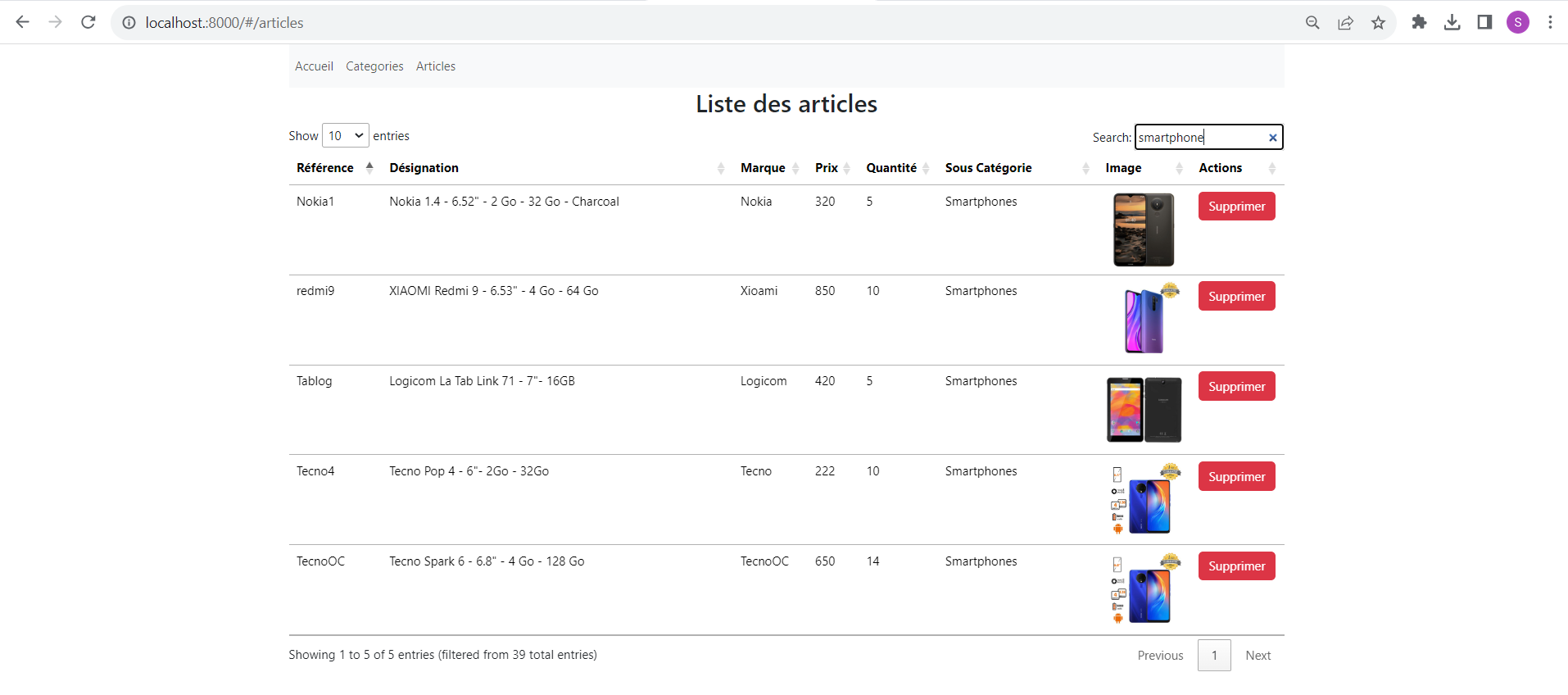
   <router-link to="/articles" class="nav-item nav-link">  Articles </router-link>



Puis affichage :







## Création de AddArticle

Maintenant, nous allons créer le formulaire d’ajout pour les articles en faisant appel à la requête axios.post().

On va commencer par créer le fichier addArticle.vue sous resources/js/components/articles.



<template lang="">

    <form @submit.prevent="ajouterproduit">

       <div class="form-group">

       <input type="text" class="form-control" placeholder="reference" v-model="article.reference">

     </div>

     <div class="form-group">

       <input type="text" class="form-control" placeholder="designation" v-model="article.designation">

     </div>

     <div class="form-group">

       <input type="text" class="form-control" placeholder="marque" v-model="article.marque">

     </div>

    <div class="form-group">

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

     </div>

    <div class="form-group">

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

     </div>

    <div class="form-group">

       Sous Catégories<select class="form-control" v-model="article.scategorieID">

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

       </select>

     </div>

     <div class="form-group">

        <input type="text" class="form-control" placeholder="image" v-model="article.imageart">

    </div>

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

   </form>

   </template>

   <script setup>

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

   import { useRouter } from 'vue-router';

   const router = useRouter()

import { ref, onMounted } from 'vue';

const Scategories = ref([]);

 onMounted(() => {

    getscategories();

            }

);

const getscategories=()=>{

    axios.get('/api/scategories').then(res => {

        Scategories.value = res.data;

            }).catch(error => {

                console.log(error)

            });

         }

let  article= {

               reference: "",

               designation: "",

               marque: "",

               prix: "",

               qtestock: "",

               imageart: "",

               scategorie:"",

       }

const  ajouterproduit=()=>{

          axios.post("/api/articles",article)

               .then(() => {

                     router.push('/articles')})

               .catch(error => {

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

                  }

   </script>

## Réglage des routes

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



import AddArticle from "./components/articles/addArticle.vue";

  {

        path: '/addArticle',

        name: 'addArticle',

        component: AddArticle

    },

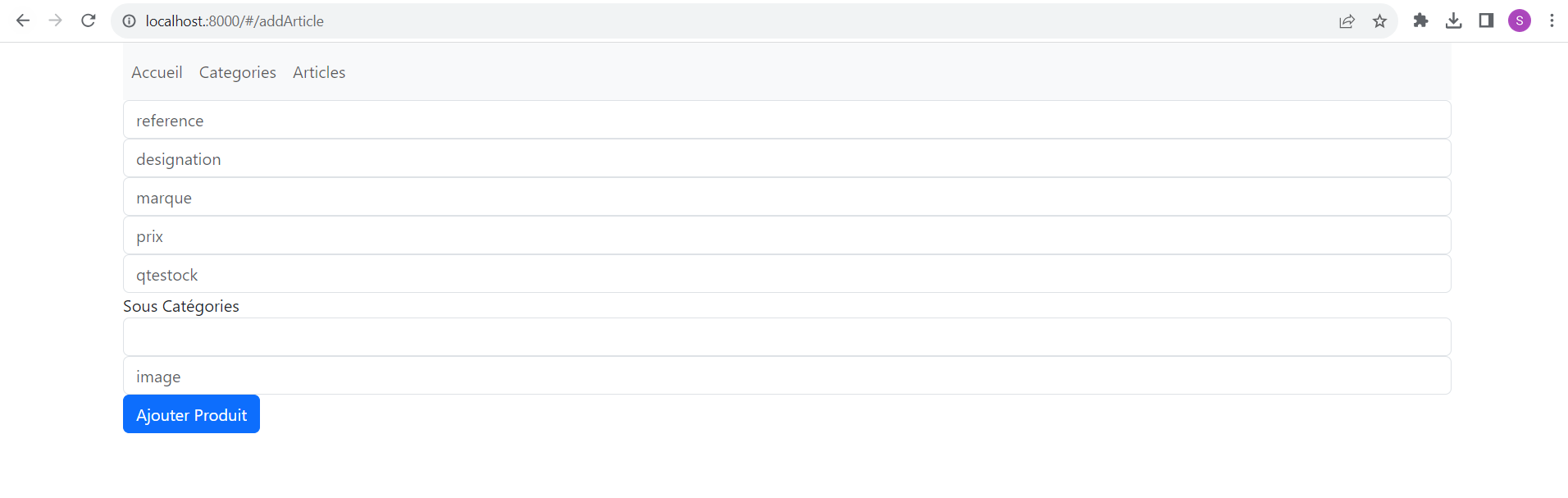
## Mise à jour de la liste

Ajouter le code :



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





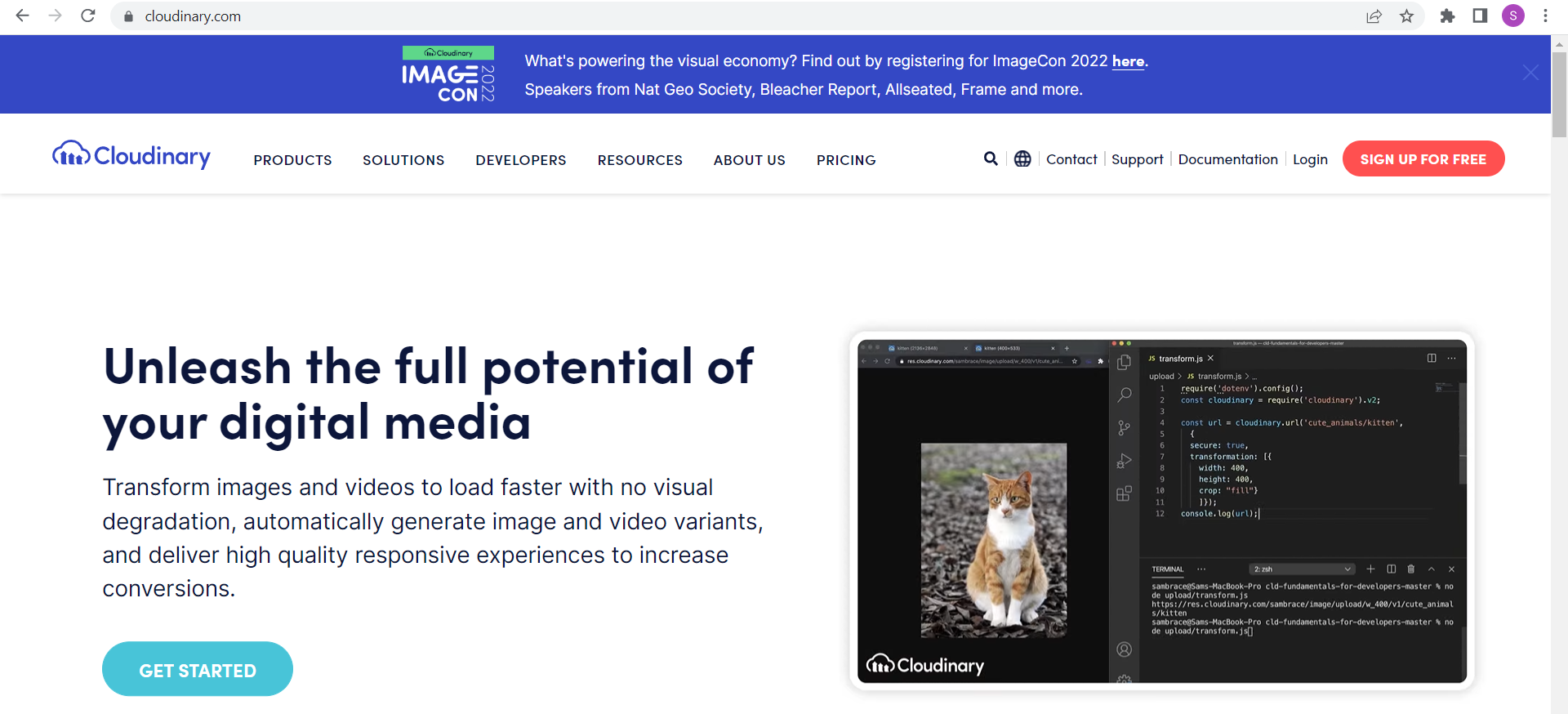
# Filepond et Cloudinary

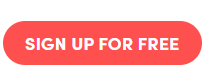
## Cloudinary

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

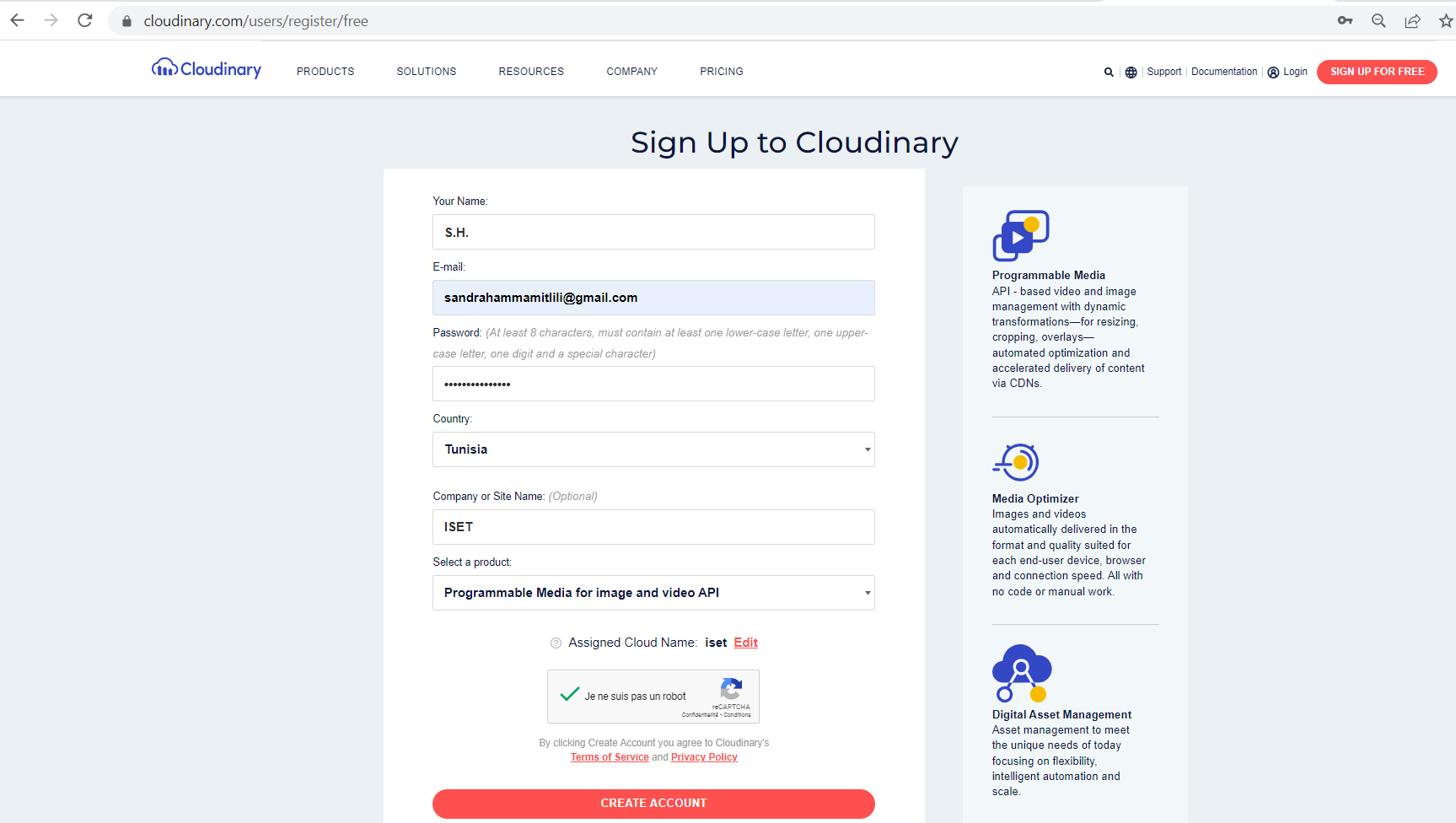
Site :

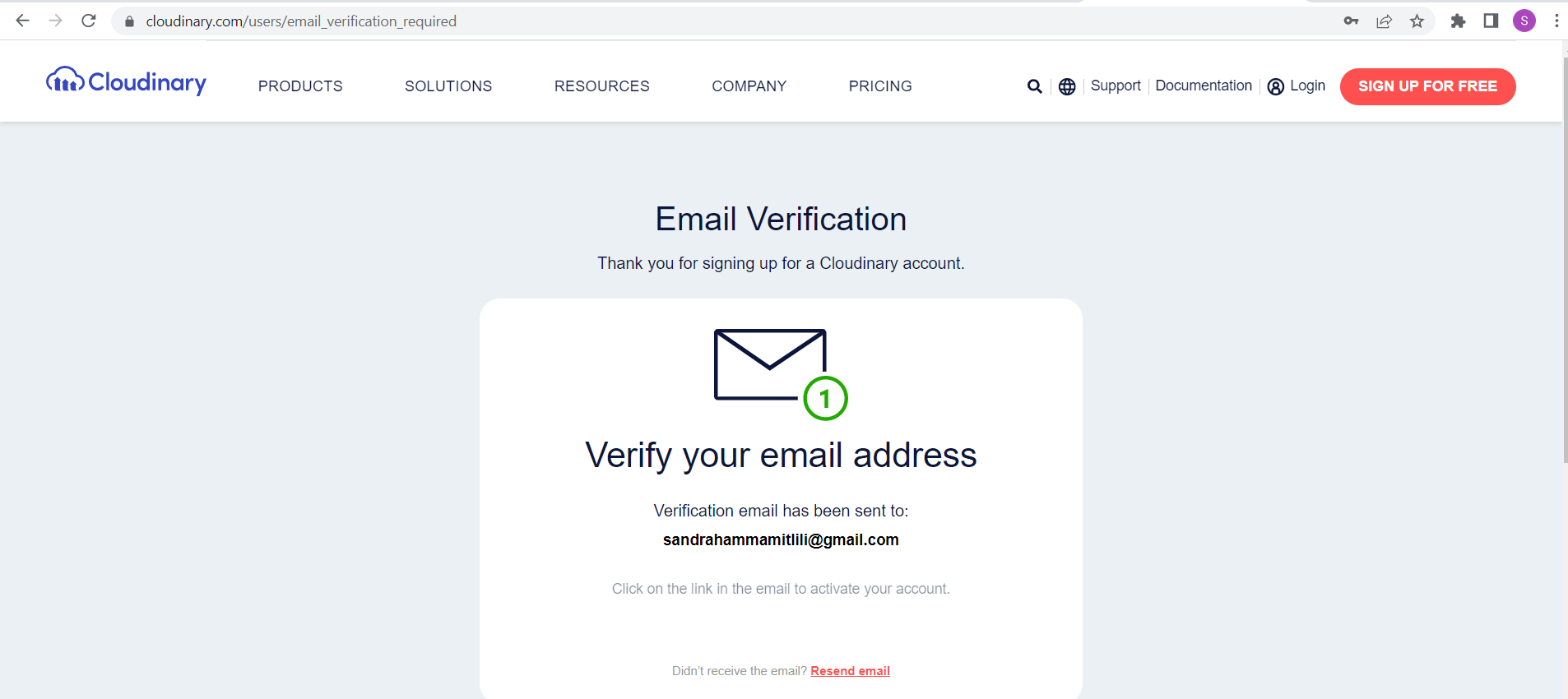
<https://cloudinary.com/>



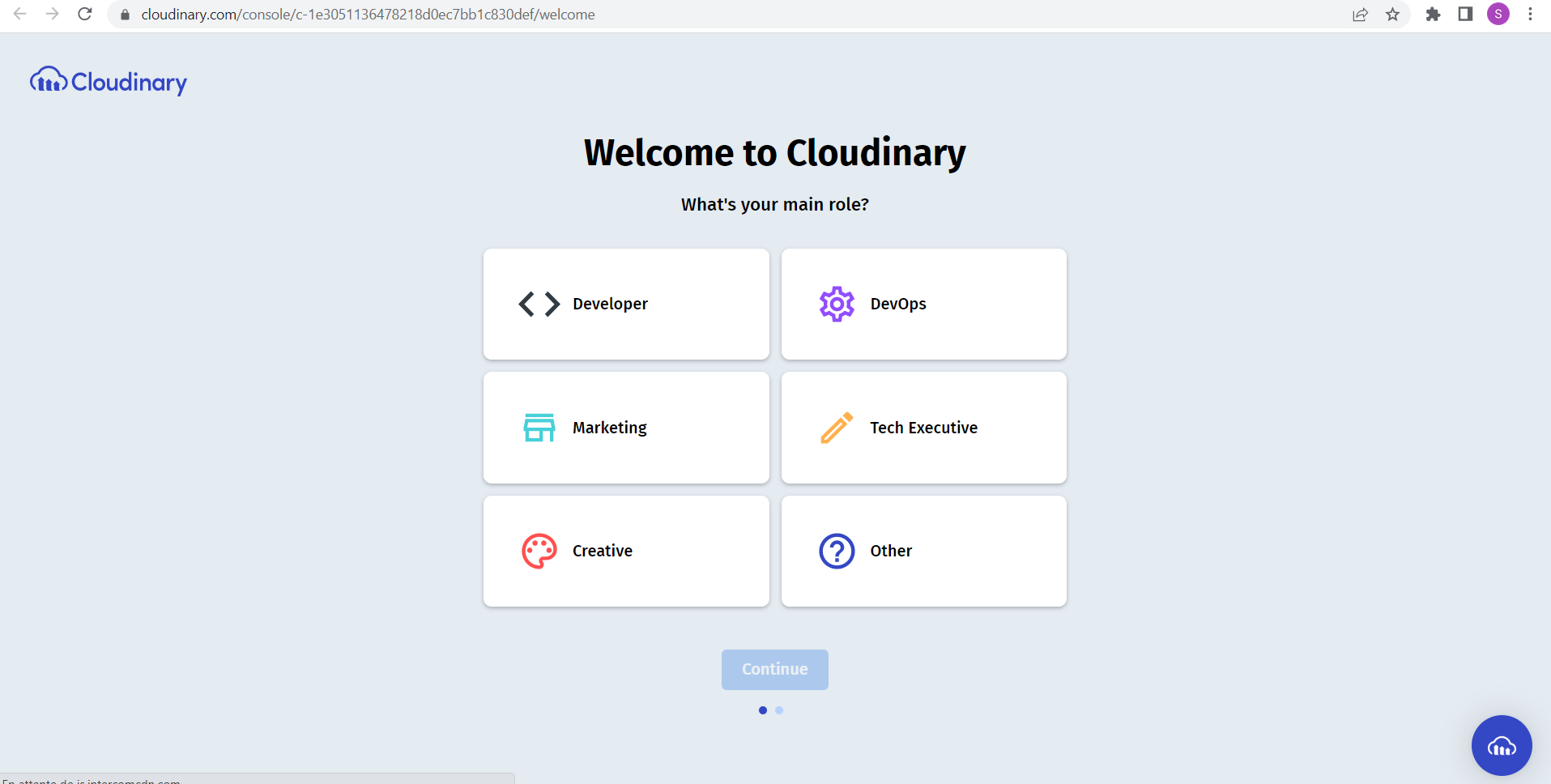


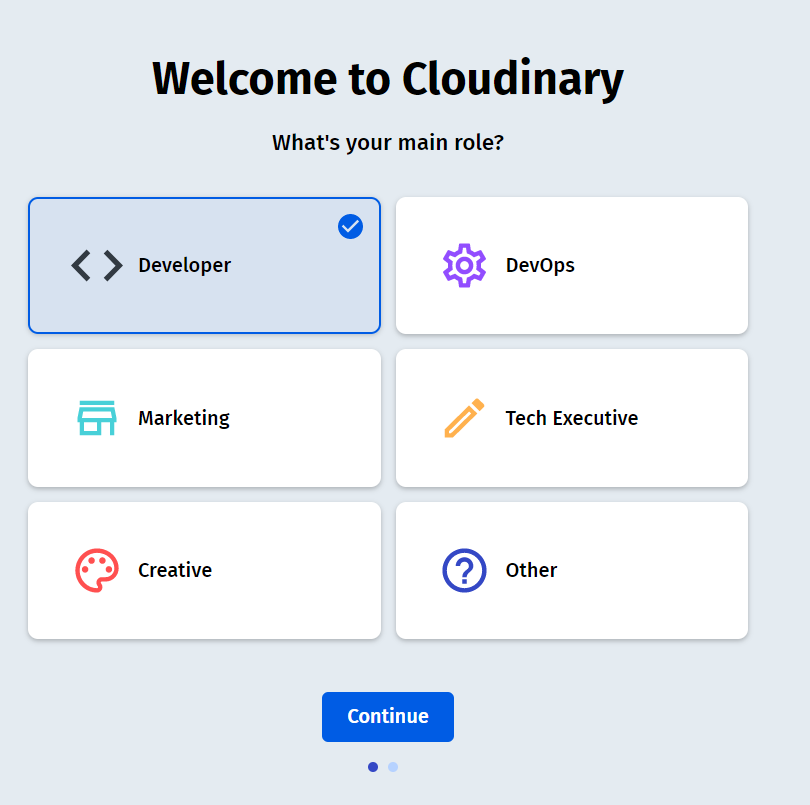
Mot de passe Cloudinary\_2022

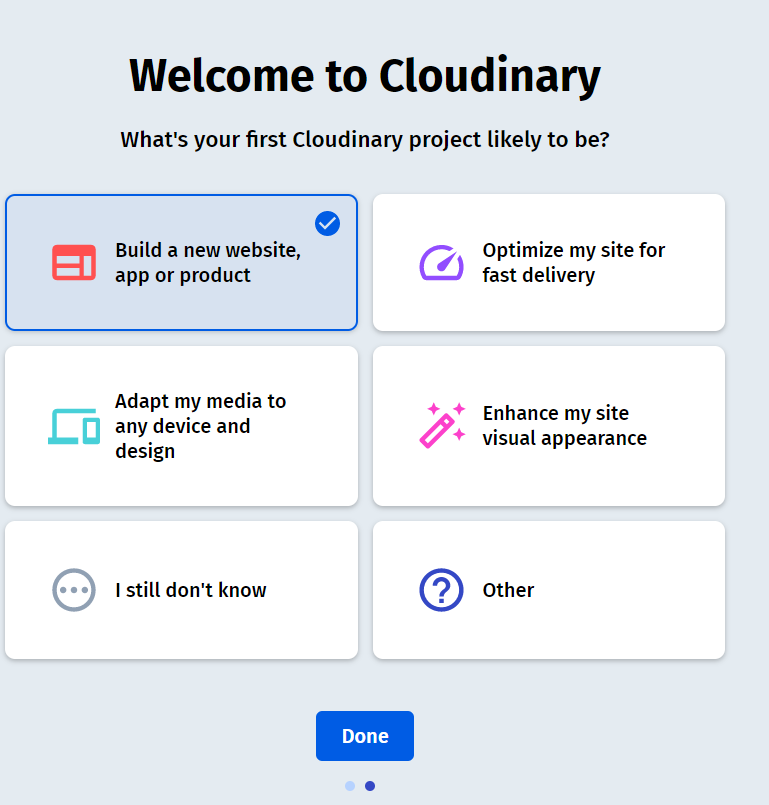


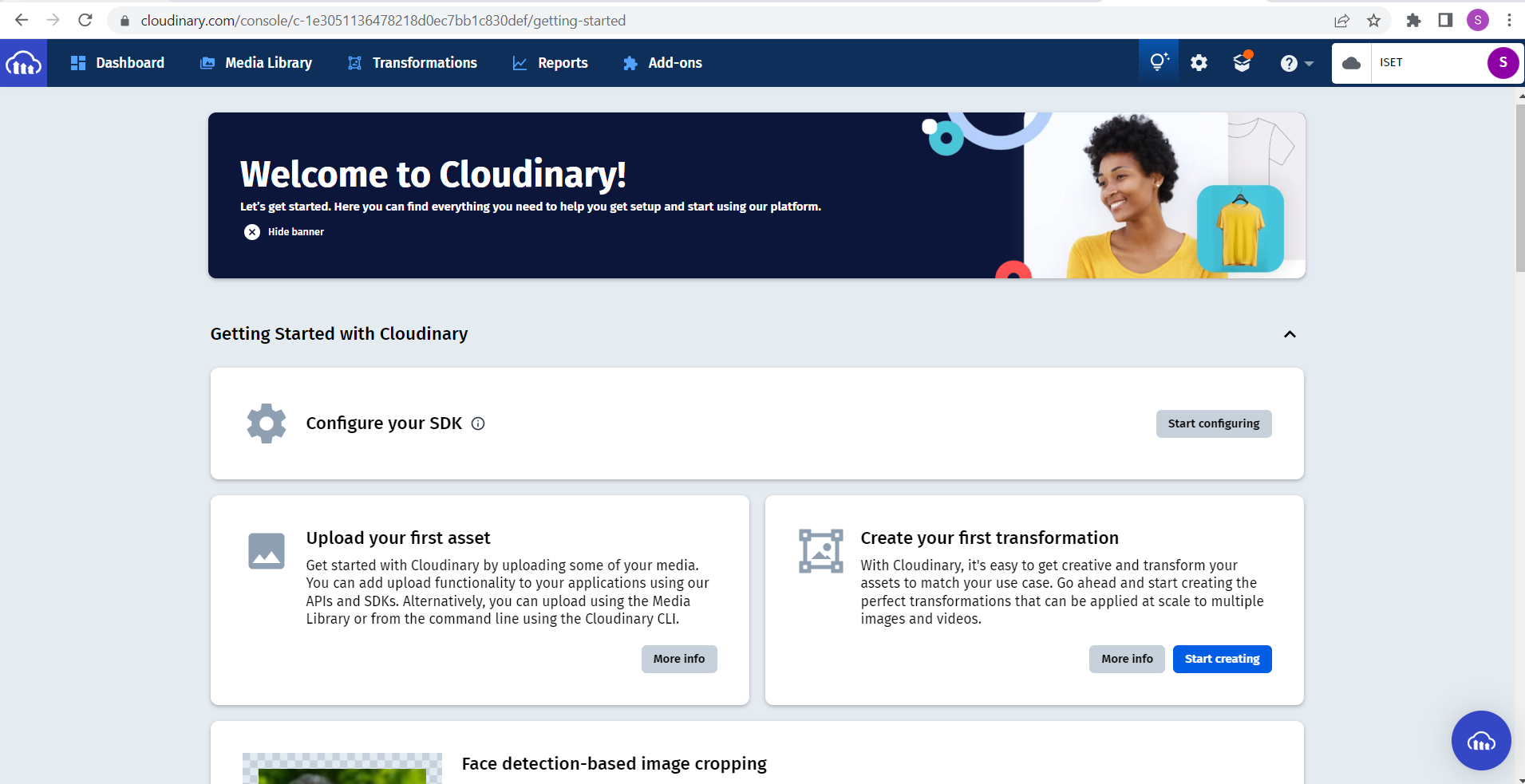


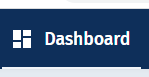


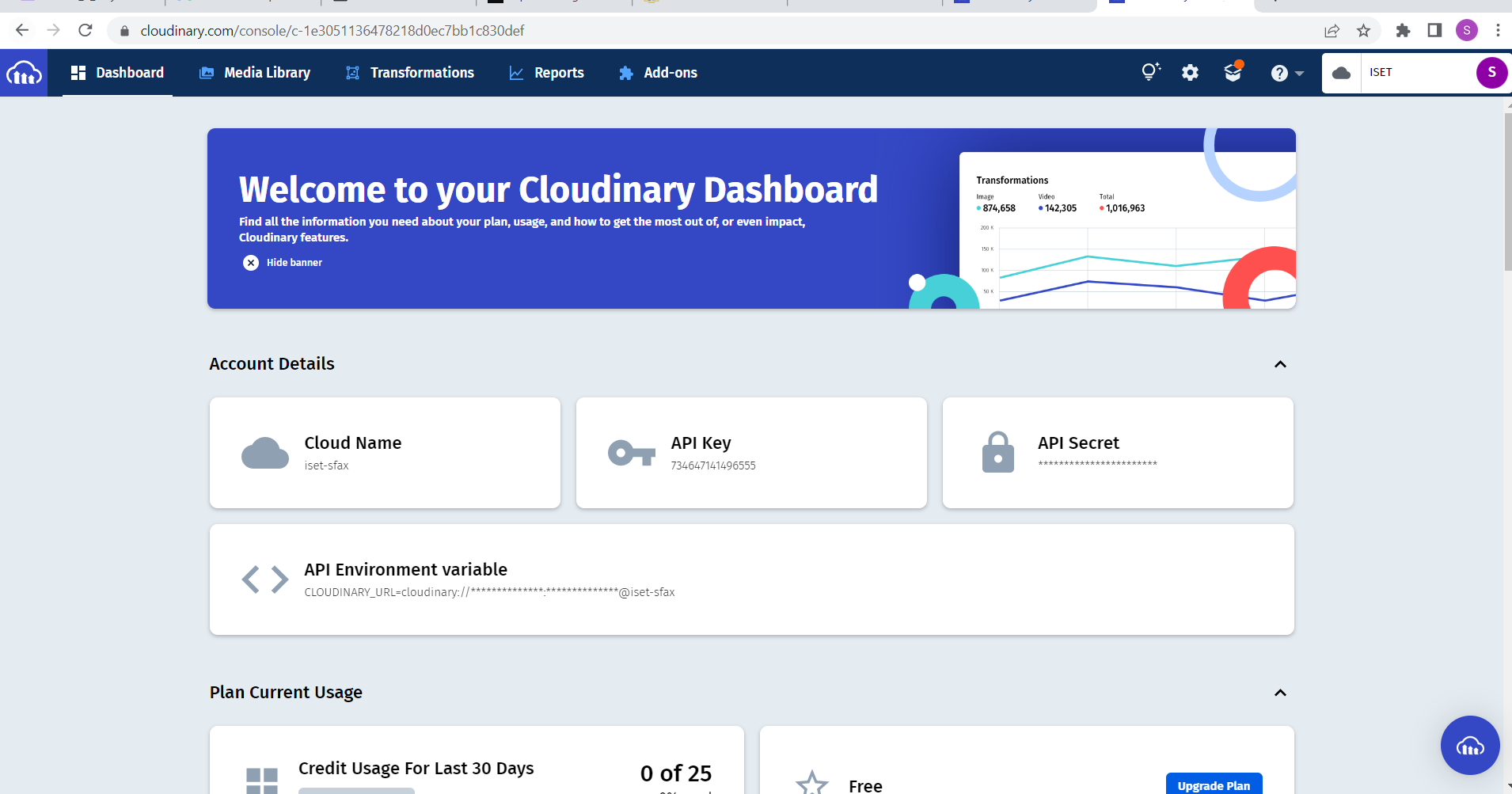


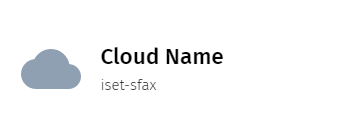






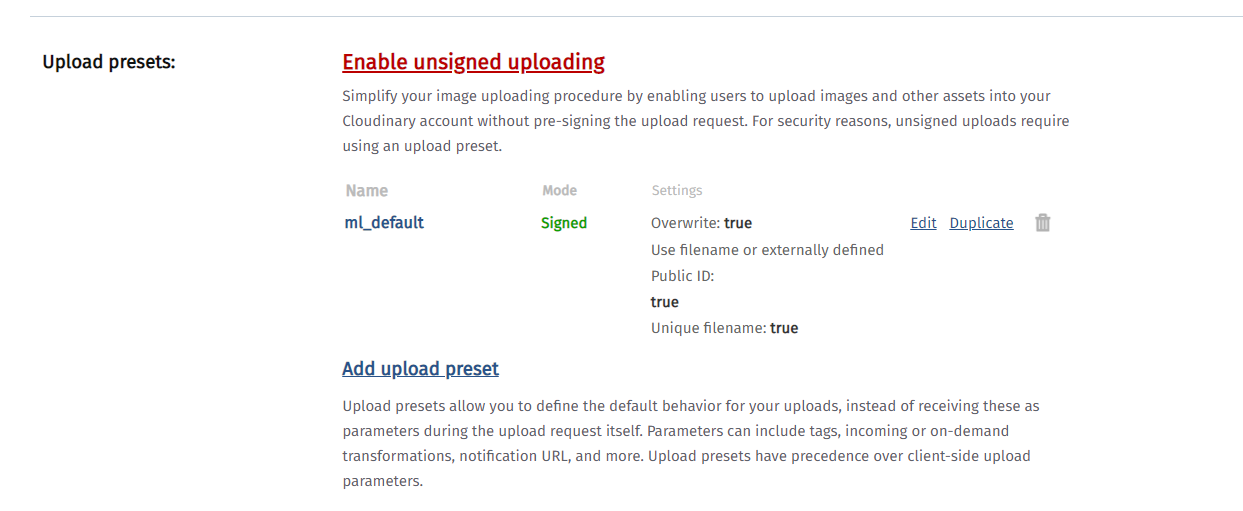


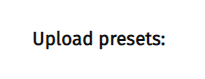




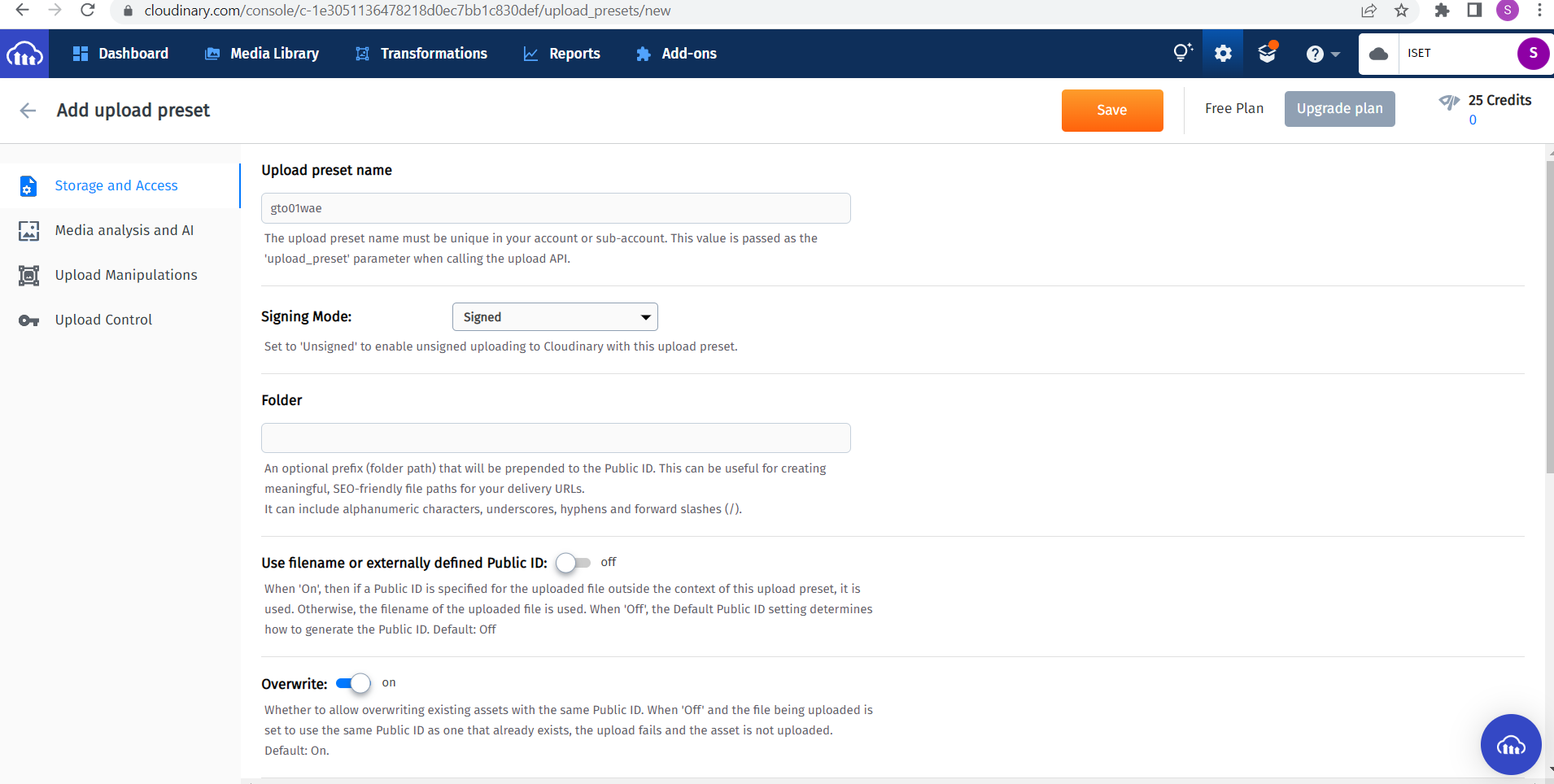


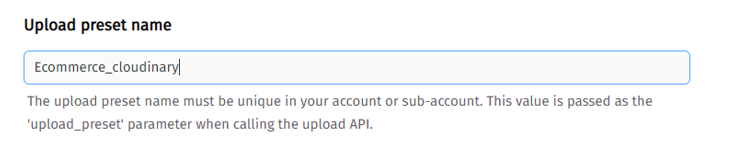


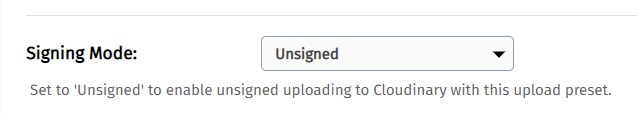


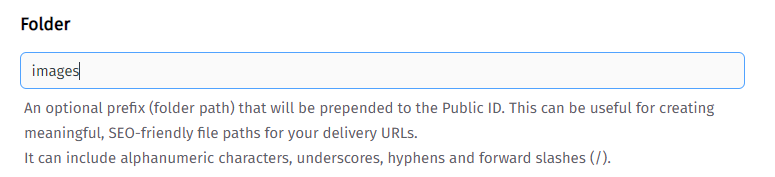


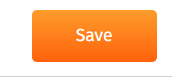




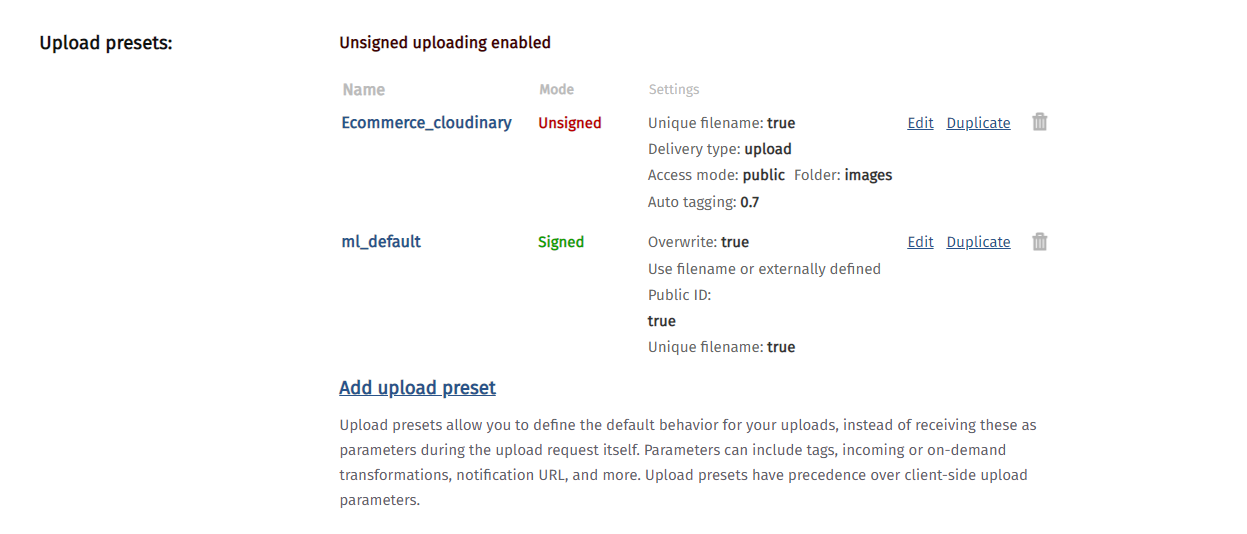


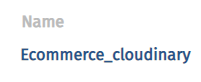












## Filepond

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

Puis modifier le code :



<template>

  <div>

    <form @submit.prevent="ajouterproduit">

     <div class="form-group">

     <input type="text" class="form-control" placeholder="reference" v-model="article.reference">

   </div>

   <div class="form-group">

     <input type="text" class="form-control" placeholder="designation" v-model="article.designation">

   </div>

   <div class="form-group">

     <input type="text" class="form-control" placeholder="marque" v-model="article.marque">

   </div>

  <div class="form-group">

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

   </div>

  <div class="form-group">

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

   </div>

  <div class="form-group">

     Sous Catégories<select class="form-control" v-model="article.scategorieID">

        <option v-for="sc in Scategories" :key="sc.id" :value=sc.id>{{sc.nomscategorie}}</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 Scategories = ref([]);

const getScategories = () => {

  axios.get('/api/scategories')

    .then(res => {

      Scategories.value = res.data;

    })

    .catch(error => {

      console.log(error);

    });

}

onMounted(() => {

  getScategories();

});

const article = {

  reference: "",

  designation: "",

  marque: "",

  prix: "",

  qtestock: "",

  imageart: "",

  scategorieID: "",

};

const  ajouterproduit=()=>{

   console.log(article)

   axios.post("/api/articles",article)

        .then(() => {

              router.push('/articles')})

        .catch(error => {

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

          article.imageart = data.url;

          load(data);

        })

        .catch((error) => {

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

          error('Upload failed');

          abort();

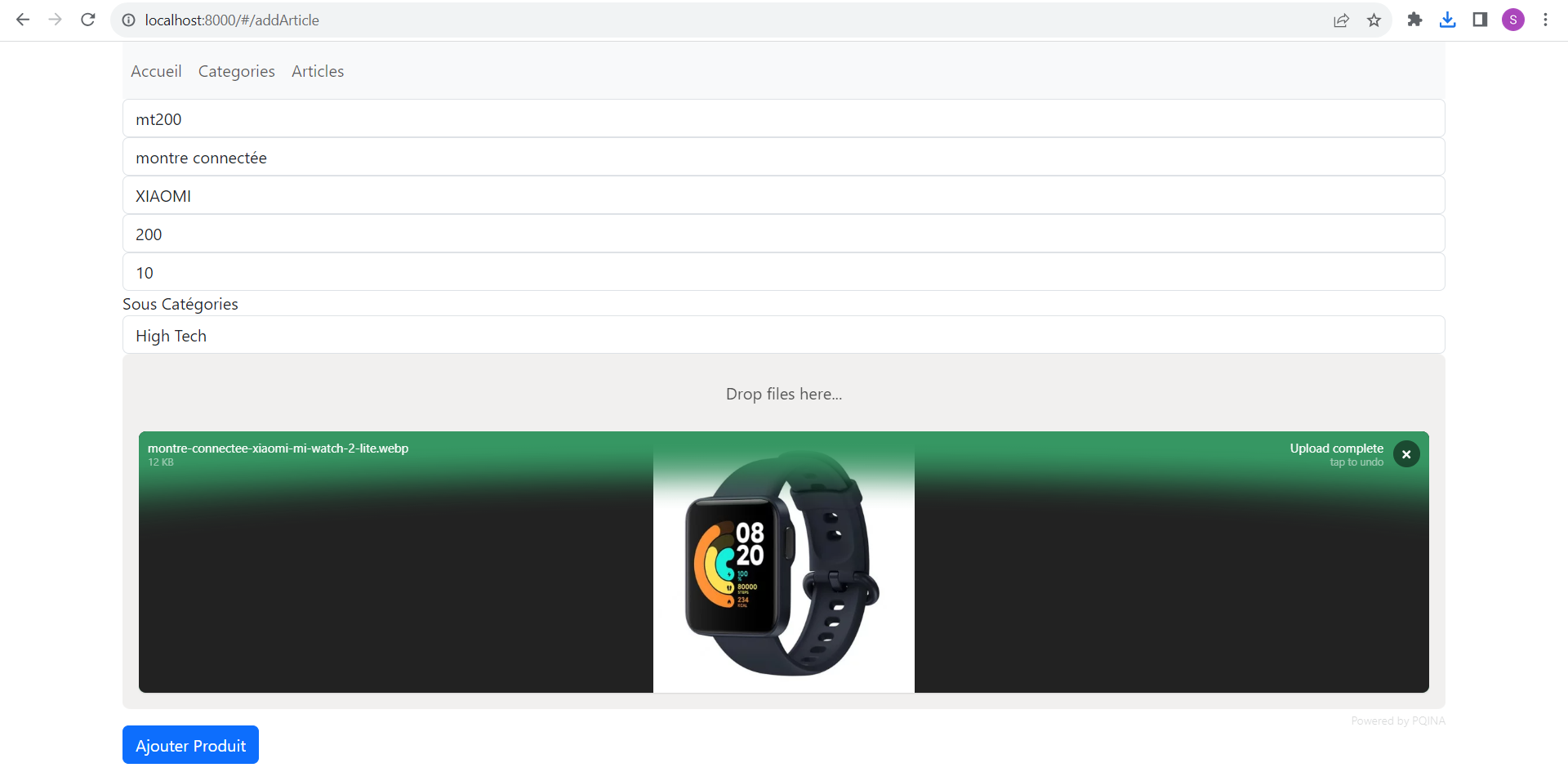
        });

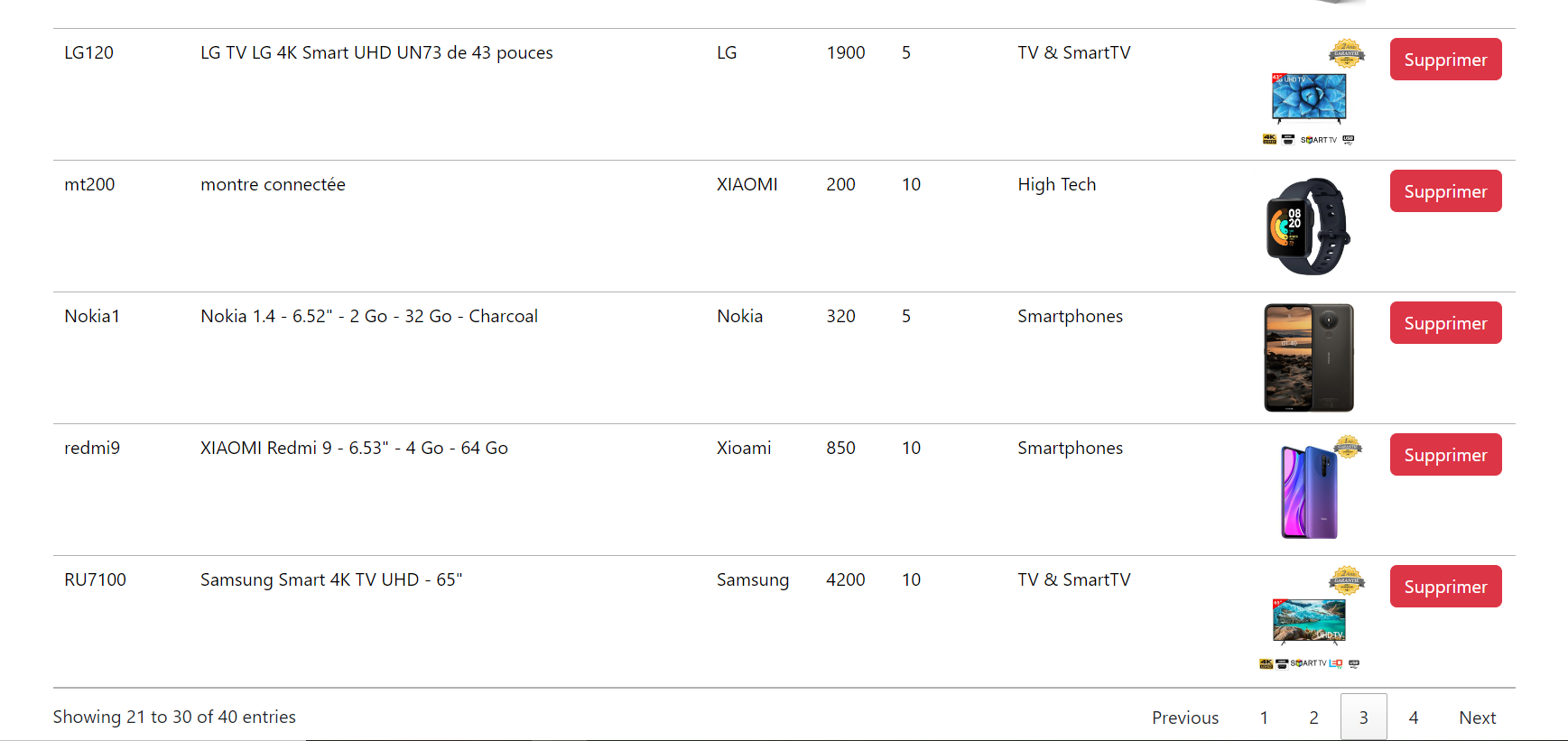
    },

  };

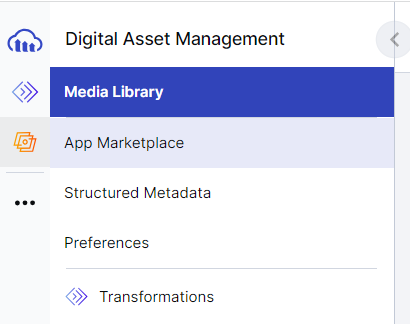
};

</script>





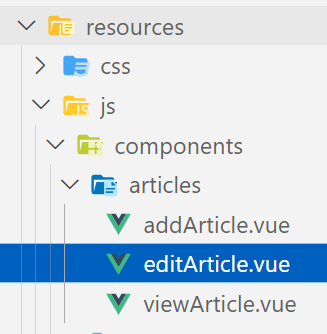
<https://console.cloudinary.com/console/c-1e3051136478218d0ec7bb1c830def/media_library/homepage>





# Modification de l’article

Créer le fichier components/articles/editArticle.vue





<template>

    <div>

      <form @submit.prevent="modifierproduit">

       <div class="form-group">

       <input type="text" class="form-control" placeholder="reference" v-model="article.reference">

     </div>

     <div class="form-group">

       <input type="text" class="form-control" placeholder="designation" v-model="article.designation">

     </div>

     <div class="form-group">

       <input type="text" class="form-control" placeholder="marque" v-model="article.marque">

     </div>

    <div class="form-group">

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

     </div>

    <div class="form-group">

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

     </div>

    <div class="form-group">

       Sous Catégories<select class="form-control" v-model="article.scategorieID">

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

       </select>

       </div>

        <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 Scategories = ref([]);

  const article = ref({});

const fetchArticle= async()=> {

        await axios

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

                .then((res) => {

                    article**.value** = res.data;

                    })

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

    }

let  myFiles = ref([]);

  const getScategories = async() => {

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

      .then(res => {

        Scategories.value = res.data;

      })

      .catch(error => {

        console.log(error);

      });

  }

  onMounted(async() => {

   await getScategories();

   await fetchArticle();

   await handleFilePondInit();

  });

  const handleFilePondInit = async() => {

  if (article.value.imageart) {

    myFiles.value = [

  {

    source: article.value.imageart,

    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) => {

            console.log(data);

            article.value.imageart = data.url;

            console.log(article.value.imageart)

            load(data);

          })

          .catch((error) => {

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

            error('Upload failed');

            abort();

          });

      },

    };

  };

  const  modifierproduit=()=>{

     console.log(article.value)

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

          .then(() => {

                router.push('/articles')})

          .catch(error => {

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

}

  </script>

**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 fetchArticle 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 EditArticle from "./components/articles/editArticle.vue";

{

        name: 'editArticle',

        path: '/editArticle/:id',

        component: EditArticle

    },



Ajouter :

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

