Points de ventes

Backend



const mongoose =require("mongoose")

const pointsventeSchema=mongoose.Schema({

    name:{ type: String, required: true},

    description:{ type: String, required: true },

    longitude:{ type: Number, required: true },

    latitude:{ type: Number, required: true },

    photo:{ type: String, required: false }

})

module.exports=mongoose.model('Pointsvente',pointsventeSchema)



const express=require("express")

const Pointsvente=require("../models/pointsvente")

const router=express.Router()

router.post("/",async(req,res)=>{

const loc1=new Pointsvente(req.body)

try {

    await loc1.save()

    res.status(200).json(loc1)

} catch (error) {

    res.status(404).json({message:error.message})

}

})

router.put("/:id",async(req,res)=>{

    try {

        const loc1 = await Pointsvente.findByIdAndUpdate(

            req.params.id,

            { $set: req.body },

            { new: true }

            );

            res.status(200).json(loc1);

    } catch (error) {

        res.status(404).json({message:error.message})

    }

})

router.get('/',async(req,res)=>{

    try {

        const loc= await Pointsvente.find({}, null, {sort: {'\_id': -1}})

        res.status(200).json(loc)

    } catch (error) {

        res.status(404).json({message:error.message})

    }

})

router.get("/:id",async(req,res)=>{

try {

    const loc=await Pointsvente.findById(req.params.id)

    res.status(200).json(loc)

} catch (error) {

    res.status(404).json({message:error.message})

}

})

router.delete("/:id",async(req,res)=>{

    try {

        await Pointsvente.findByIdAndDelete(req.params.id)

        res.status(200).json({messge:"Point vente supprimée"})

    } catch (error) {

        res.status(404).json({message:error.message})

    }

})

module.exports=router



const pointsventeRouter =require("./routes/pointsvente.route.js");

app.use('/api/pointsventes', pointsventeRouter);



const mongoose =require("mongoose")

const articleSchema=mongoose.Schema({

reference:{ type: String, required: true,unique:true },

designation:{ type: String, required: true,unique:true },

prix:{ type: Number, required: false },

marque:{ type: String, required: true },

qtestock:{ type: Number, required: false },

imageart:{ type: String, required: false },

scategorieID: {type:mongoose.Schema.Types.ObjectId,

ref:'Scategorie'},

depotID :  [{type:mongoose.Schema.Types.ObjectId,

    ref:'Pointsvente'}],

})

module.exports=mongoose.model('Article',articleSchema)



const express=require("express")

const Article=require("../models/article")

const router=express.Router()

router.post("/",async(req,res)=>{

const art1=new Article(req.body)

try {

    await art1.save()

    res.status(200).json(art1)

} catch (error) {

    res.status(404).json({message:error.message})

}

})

router.put("/:id",async(req,res)=>{

    try {

        const art1 = await Article.findByIdAndUpdate(

            req.params.id,

            { $set: req.body },

            { new: true }

            );

            res.status(200).json(art1);

    } catch (error) {

        res.status(404).json({message:error.message})

    }

})

router.get('/',async(req,res)=>{

    try {

        const art= await Article.find({}, null, {sort: {'\_id': -1}}).populate("scategorieID")

        res.status(200).json(art)

    } catch (error) {

        res.status(404).json({message:error.message})

    }

})

router.get("/:id",async(req,res)=>{

try {

    const art=await Article.findById(req.params.id).populate("scategorieID")

    res.status(200).json(art)

} catch (error) {

    res.status(404).json({message:error.message})

}

})

router.delete("/:id",async(req,res)=>{

    try {

        await Article.findByIdAndDelete(req.params.id)

        res.status(200).json({messge:"Article supprimée avec succées"})

    } catch (error) {

        res.status(404).json({message:error.message})

    }

})

router.get('/art/pagination', async(req, res) => {

    const filtre = req.query.filtre || "";

    const page = parseInt(req.query.page);

    const pageSize = parseInt(req.query.pageSize);

    // Calculate the start and end indexes for the requested page

    const startIndex = (page - 1) \* pageSize;

    const endIndex = page \* pageSize;

    const articles = await Article.find({ designation: { $regex: filtre, $options: "i"}}, null, {sort: {'\_id': -1}}).populate("scategorieID").exec()

    // Slice the products array based on the indexes

    const paginatedProducts = articles.slice(startIndex, endIndex);

    // Calculate the total number of pages

    const totalPages = Math.ceil(articles.length / pageSize);

    // Send the paginated products and total pages as the API response

    res.json({ products: paginatedProducts, totalPages });

  });

  // modifier quantité seulement

  router.put('/qty/:id', async (req, res) => { console.log(req.body)

    const qty = req.body.quantity||0;

    const articleId=req.params.id||null;

    const oldArticle=await Article.findById(articleId)

     try {

       const articleUpdated = await Article.findByIdAndUpdate(

         articleId,

         { qtestock: oldArticle.qtestock - qty},

         { new: true } // Return the updated document

       );

       if (!articleUpdated) {

         return res.status(404).json({ message: 'Product not found' });

       }

       const art = await Article.findById(articleId).populate("scategorieID").exec();

       res.status(200).json(art);

     } catch (error) {

       res.status(404).json({ message: error.message });

     }

   });

   //pagination avec use useSearchParams react

router.get('/art/paginationUSP', async (req, res) => {

    const page = parseInt(req.query.page) || 1;

    const limit = parseInt(req.query.limit) || 10;

    const searchQuery = req.query.searchQuery || "";

    const articles= await Article.find({ designation: { $regex: searchQuery, $options: "i"}}, null, {sort: {'\_id': -1}}).populate("scategorieID").exec()

    const tot = articles.length;

    const startIndex = (page - 1) \* limit;

    const endIndex = page \* limit;

    const results = {};

    if (endIndex < articles.length) {

      results.next = {

        page: page + 1,

        limit: limit

      };

    }

    if (startIndex > 0) {

      results.previous = {

        page: page - 1,

        limit: limit

      };

    }

    results.results = articles.slice(startIndex, endIndex);

    res.json({results,tot});

  });

  router.get('/art/points/:id', async (req, res) => {

    try {

      const art=await Article.findById(req.params.id).populate("depotID")

      res.status(200).json(art)

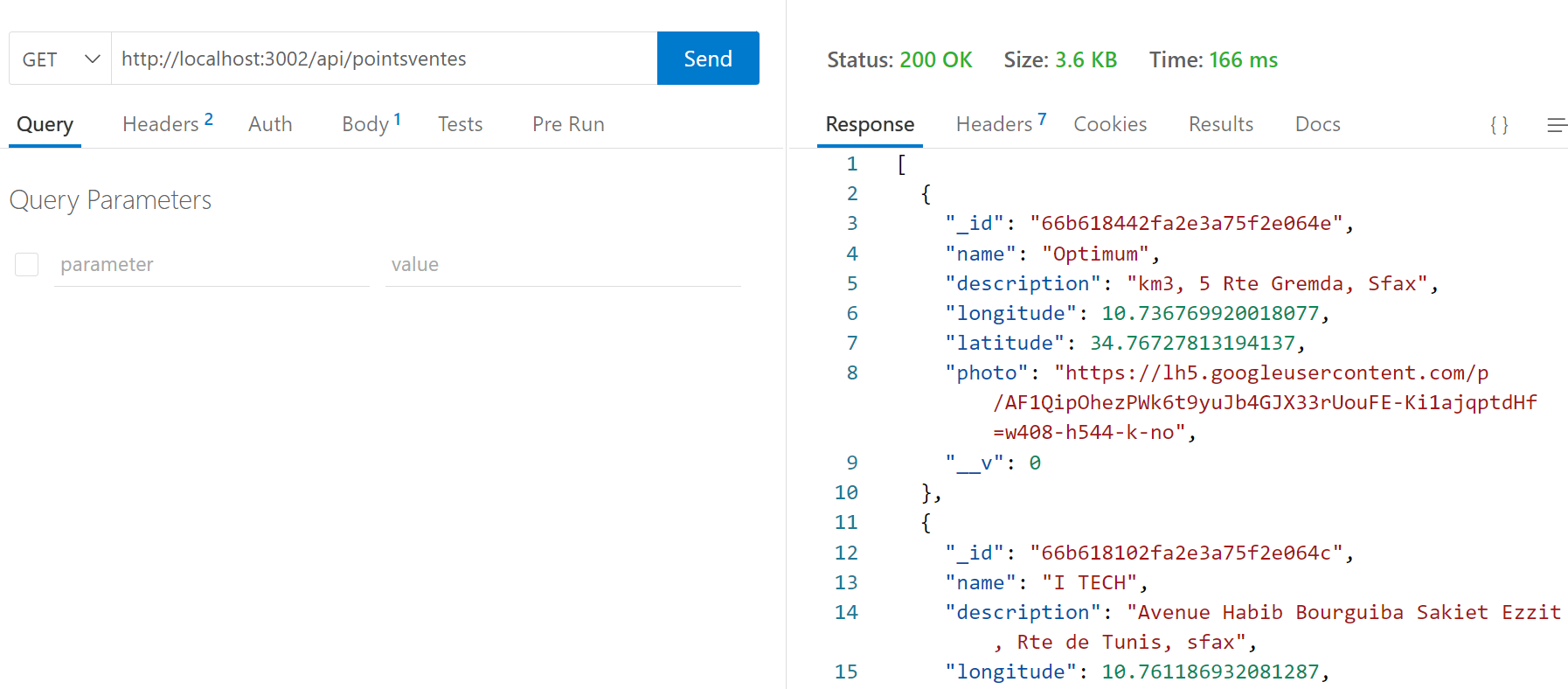
  } catch (error) {

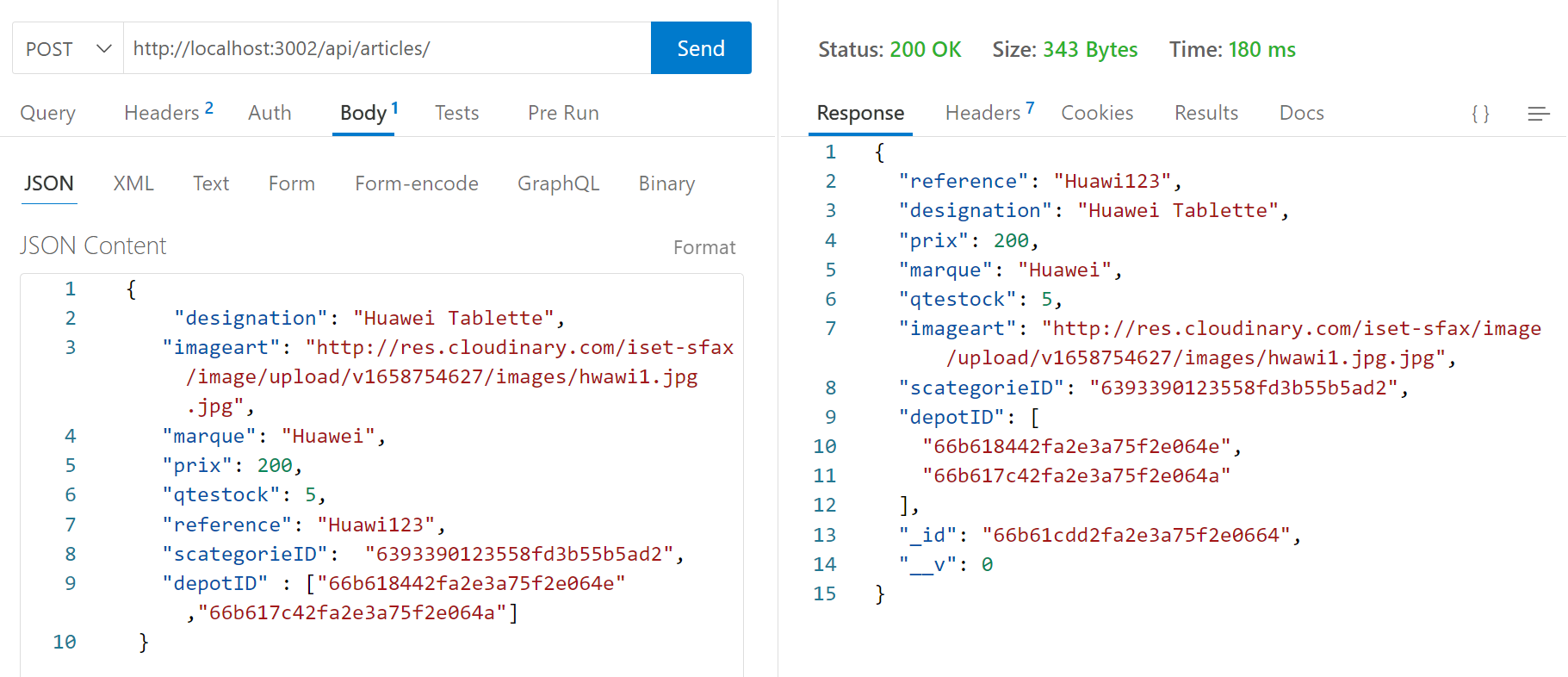
      res.status(404).json({message:error.message})

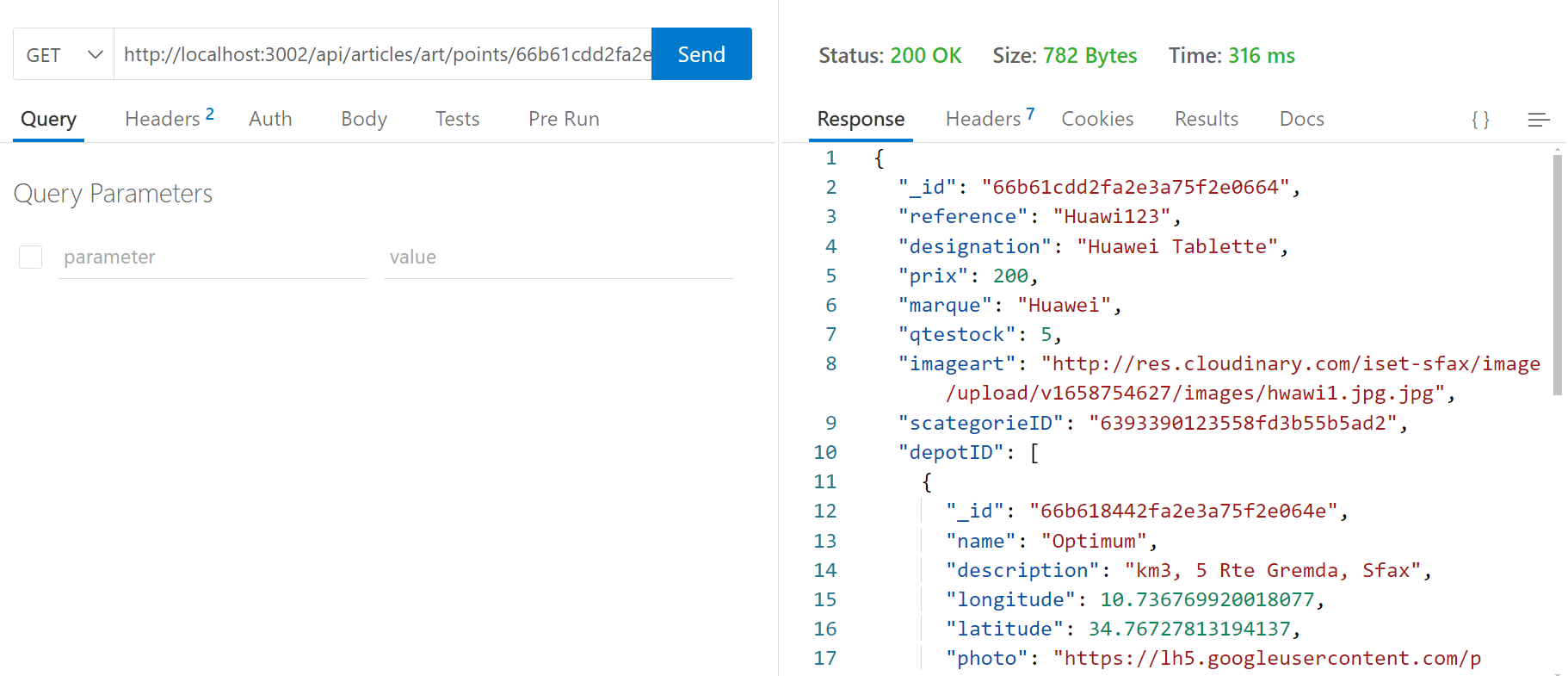
  }

  })

module.exports=router







Front end

Dépendances :

npm install react-leaflet leaflet

npm install leaflet-routing-machine



<link

rel="stylesheet"

href="https://unpkg.com/leaflet@1.6.0/dist/leaflet.css"

integrity="sha512-xwE/Az9zrjBIphAcBb3F6JVqxf46+CDLwfLMHloNu6KEQCAWi6HcDUbeOfBIptF7tcCzusKFjFw2yuvEpDL9wQ=="

crossorigin=""

/>



import React, { useEffect } from 'react'

import { useDispatch, useSelector } from "react-redux";

import {setPage, setLimit, getArticlesPagination} from "../../../features/articleSlice";

import AfficheArticles from './AfficheArticles';

import Pagination from './Pagination';

import "../../../style.css";

import Headerarticle from './Headerarticle';

const Listarticles = () => {

  let {page,limit,searchTerm} = useSelector((state)=>state.storearticles);

    const getProducts=async()=>{

           dispatch(getArticlesPagination())

          }

    const dispatch=useDispatch()

    useEffect(() => {

      getProducts()

    }, [dispatch,page,limit,searchTerm])

   const handleLimitChange = React.useCallback((event) => {

      dispatch(setLimit(parseInt(event.target.value, 10)));

      dispatch(setPage(1)); // Réinitialiser la page lorsque le nombre d'éléments par page change

    }, [dispatch]);

  return (

    <div>

     <div className="table-container-header">

     <Headerarticle />

    </div>

    <AfficheArticles/>

      <div style={{ "display": "flex", "justifyContent": "right"}}>

      <div className="limit-selector-container">

                <label>

                    Afficher &nbsp;

                    <select

                      value={limit}

                      onChange={(event) =>  {handleLimitChange(event)}}

                    >

                      <option value={5}>5</option>

                      <option value={10}>10</option>

                      <option value={20}>20</option>

                      <option value={100}>100</option>

                    </select>

                  </label>

                  </div>

     <Pagination />

      </div>

    </div>

  )

}

export default Listarticles



import React from 'react'

import { useDispatch, useSelector } from "react-redux";

import {setPage,setSearchTerm} from "../../../features/articleSlice";

import "../../../style.css";

const Headerarticle = () => {

  const dispatch=useDispatch()

  let {searchTerm} = useSelector((state)=>state.storearticles);

  return (

  <div className="search-container">

            <i className="fa-solid fa-search"></i>

            <input

              type="text"

              value={searchTerm}

              onChange={(event)=>{ dispatch(setSearchTerm(event.target.value));dispatch(setPage(1))}}

              placeholder="Rechercher des articles..."

              className="search-input"

            />

          </div>

  )

}

export default Headerarticle



import React from 'react'

import { useDispatch, useSelector } from "react-redux";

import { addToCart } from "../../../features/cartSlice";

import { useNavigate, Link } from "react-router-dom";

import "./styleCard.css"

const AfficheArticles = () => {

const {articles,isLoading,error} = useSelector((state)=>state.storearticles);

const paginatedArticles = React.useMemo(() => articles, [articles]);

const dispatch = useDispatch();

let navigate=useNavigate();

const handleAddToCart = (art) => {

  dispatch(addToCart(art));

  navigate("/cart");

};

return (

  <div className="container">

    <div className="row">

      {isLoading ? <center>Loading ....</center> : null}

      {error ? <center>Error ....</center> : null}

      {!isLoading && paginatedArticles && paginatedArticles.map((article, ind) => {

        return (

          <div className="col-xl-3 col-lg-6 col-md-2 col-sm-6 col-12 mb-5" key={ind}>

            <div className='card'>

            <Link to={`/locationsPV/${article.\_id}`} >

            <button className='card-button'>

            <i className="fas fa-map-marker-alt"></i> Voir points de ventes

            </button>

            </Link>

              {article.imageart && <img src={article.imageart} alt={article.reference} />}

              <div className='card-content'>

                <h1 className='card-title'>{article.reference}</h1>

                <p className='card-description'>{article.designation.substr(0, 20)}</p>

                <h1 className='card-title'>Prix : {article.prix} TND</h1>

                <button className='card-button' disabled={article.qtestock <= 1} onClick={() => handleAddToCart(article)}>

                  <i className="fa-solid fa-cart-shopping"></i> Add to cart

                </button>

              </div>

            </div>

          </div>

        );

      })}

    </div>

  </div>

);

}

export default AfficheArticles

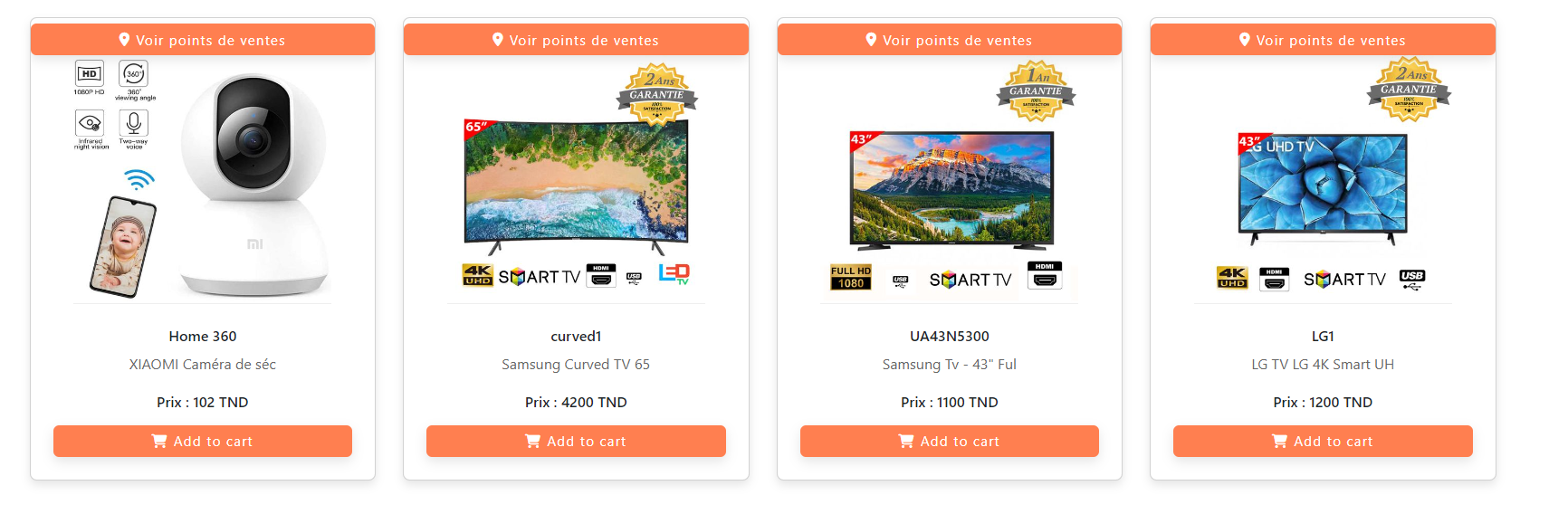


import ListarticlesPV from './components/client/articlesPointsVentes/Listarticles'

import Locations from './components/client/articlesPointsVentes/geolocation/Location'

<Route path="/listarticlesPV" element={<ListarticlesPV/>}/>

<Route path='/locationsPV/:id' element={<Locations/>}/>





.leaflet-container {

width: 100%;

height: 100vh;

}



import { useEffect, useState } from 'react';

import './styleLocation.css';

import MapLeaflet from './MapLeaflet';

import axios from "axios";

import {  useParams} from 'react-router-dom';

function Location() {

  const {id} = useParams();

  const [locations, setLocations] = useState([]);

  useEffect(() => {

    axios.get('http://localhost:3002/api/articles/art/points/'+ id)

       .then((res) => {

              setLocations(res.data.depotID);

      })

      .catch((error) => {

        console.error("Error fetching locations:", error);

      });

  }, []);

  return (

    <div className="leaflet-container">

      {locations.length > 0 ? (

        <MapLeaflet locations={locations} />

      ) : (

        <p>Loading locations...</p>

      )}

    </div>

  );

}

export default Location;



import React from "react";

import { MapContainer, TileLayer, Marker , Popup } from 'react-leaflet';

function MapLeaflet({locations}) {

  return (

    <MapContainer center={[locations[0].latitude, locations[0].longitude]} zoom={13} scrollWheelZoom={false}>

    <TileLayer

      url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"

      attribution='&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'

    />

    {

        locations && locations.map((item,ind)=>{

        return  <Marker position={[item.latitude, item.longitude]} key={ind}>

        <Popup>

        <img src={item.photo} width={50} height={50} alt={item.name}/>

        <br/>

        {item.name} <br/> {item.description}

        </Popup>

      </Marker>

        })

    }

  </MapContainer>

  );

}

export default MapLeaflet;

