FRONT END FULL STACK 201 - 202 -205 -206 IFMOTICA FES

***REACT TP API***

Objectif: utilisé API Locale –créer une application qui utilise un api local (exécution des requêtes get-post-put-délete)

Soit un api local suivant :

{

    "Articles": [

      {

        "id": 1,

        "nomA": "PC",

        "pu": 2500,

        "Qt":15

      },

      {

        "id": 2,

        "nomA": "TABLET",

        "pu": 4500,

        "Qt":150

      },

      {

        "id": 3,

        "nomA": "TV",

        "pu": 12500,

        "Qt":24

      },

      {

        "id": 4,

        "nomA": "SMART PHONE",

        "pu": 3500,

        "Qt":18

      }

    ]

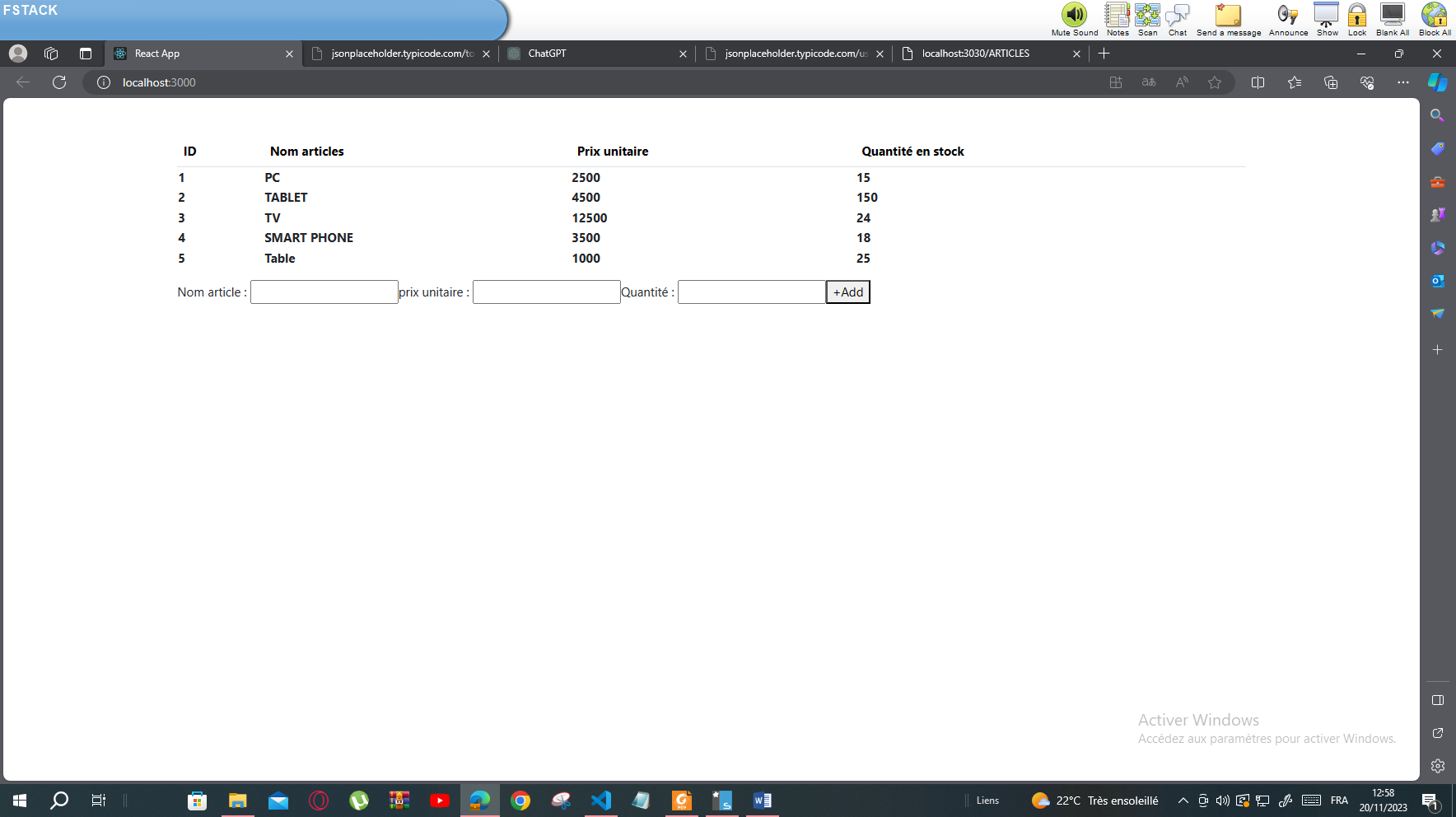
  }

Etape 1 : installer le serveur local json :

npm install json-server

Etape 1 : lancer server json local

npx json-server --watch db.json --port 3030



Article .js

import React,{useState,useEffect} from 'react'

import axios from 'axios'

export default function Article() {

    const [tarticles,setTarticles]=useState([])

    const [nomA,setNomA]=useState('')

    const [pu,setPu]=useState('')

    const [qt,setQt]=useState('')

    useEffect(()=>{

        axios.get('http://localhost:3030/articles')

        .then((res)=>{ console.log(res.data);setTarticles(res.data)})

        })

      const  ajout=()=>{

        axios.post('http://localhost:3030/articles',{nomA:nomA,pu:pu,Qt:qt})

        .then((res)=>{ alert('bien ajouter')

           }).catch(err=>alert('erreur'))

       setNomA('');

       setPu('');

       setQt('');

      }

        return(

        <div className='container mt-5'>

<table className='table'>

    <thead>

    <tr><th>ID</th><th>Nom articles</th><th>Prix unitaire</th><th>Quantité en stock</th></tr>

    </thead>

        {tarticles.map(Art=>{

        return(

            <tr><th>{Art.id} </th><th>{Art.nomA}</th><th>{Art.pu}</th><th>{Art.Qt}</th></tr>

        )

        })}

        </table>

        Nom article   : <input type='text' value={nomA} onChange={ev=>setNomA(ev.target.value)}></input>

        prix unitaire : <input type='text' value={pu} onChange={ev=>setPu(ev.target.value)}></input>

        Quantité      : <input type='text' value={qt} onChange={ev=>setQt(ev.target.value)}></input>

       <button onClick={ajout}> +Add </button>

     </div>

        )

        }

Index.html

<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8" />

    <link rel="icon" href="%PUBLIC\_URL%/favicon.ico" />

    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

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

    <meta name="theme-color" content="#000000" />

    <meta

      name="description"

      content="Web site created using create-react-app"

    />

    <link rel="apple-touch-icon" href="%PUBLIC\_URL%/logo192.png" />

    <!--

      manifest.json provides metadata used when your web app is installed on a

      user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/

    -->

    <link rel="manifest" href="%PUBLIC\_URL%/manifest.json" />

    <!--

      Notice the use of %PUBLIC\_URL% in the tags above.

      It will be replaced with the URL of the `public` folder during the build.

      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC\_URL%/favicon.ico" will

      work correctly both with client-side routing and a non-root public URL.

      Learn how to configure a non-root public URL by running `npm run build`.

    -->

    <title>React App</title>

  </head>

  <body>

    <noscript>You need to enable JavaScript to run this app.</noscript>

    <div id="root"></div>

    <!--

      This HTML file is a template.

      If you open it directly in the browser, you will see an empty page.

      You can add webfonts, meta tags, or analytics to this file.

      The build step will place the bundled scripts into the <body> tag.

      To begin the development, run `npm start` or `yarn start`.

      To create a production bundle, use `npm run build` or `yarn build`.

    -->

  </body>

</html>