

Atelier 05 : MiniEcommerceApp

Le projet utilise **React 19**, **react-router-dom v7** et un **petit backend Express** pour enregistrer les commandes dans un fichier orders.json.

Les images **produits** sont en local (public/images/...) et la « base de données » des produits est un fichier JSON (src/data/products.json).

Étape 1 : Préparer l'environnement React

Objectif pédagogique :

Savoir créer un projet React fonctionnel et structurer les fichiers.

Instructions :

1. Dans le dossier Ressources de cet atelier, copier le dossier **MiniEcommerceApp** dans le dossier **C:\ReactProjects**
2. Ouvrir le dossier **C:\ReactProjects\MiniEcommerceApp** avec votre éditeur de code
3. Démarrer docker et vérifier que les containers ne sont pas en exécution
4. Avec l'invite de commande, entrez dans le dossier **C:\ReactProjects\MiniEcommerceApp** et lancez la commande suivante :
docker-compose up -d --build
5. Créez un projet React avec Vite ou Create React App.

1. Accéder au **MiniEcommerceApp_container** avec la commande suivante :

docker exec -it MiniEcommerceApp_container sh

2. **npm create vite@latest . --template**

Deux traits avant
template (optionnel)

```
/usr/src/app # npm create vite@latest . --template
> npx
> create-vite .
|
|   ▶ Select a framework:
|     React
|
|   ▶ Select a variant:
|     JavaScript
|
|   ▶ Use rollup-vite (Experimental)?:
|     No
|
|   ▶ Install with npm and start now?
|     Yes / No
|       Yes
```

3. Si le serveur Vite lancé, arrêtez-le (CTR + C)

4. Quitter le container

```
/usr/src/app # exit
```

5. Modifier le fichier vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react()],
  server: {
    host: true,           // équivaut à --host
    watch: {
      usePolling: true, // <== active le mode "polling"
      interval: 1000,   // <== vérifie les changements toutes les 1s
    },
  },
})
```

6. Puis arrêter et démarrer le container, directement avec le docker ou bien exécuter ces deux commandes

```
docker-compose down
docker-compose up -d --build
```

7. Entre dans le container

```
docker exec -it MiniEcommerceApp_container sh
```

8. Avant de commencer : Identifier la version actuelle

```
npm list react-router-dom
```

/usr/src/app # npm list react-router-dom

app@0.0.0 /usr/src/app

`-- (empty)

9. Installation

```
npm install react-router-dom@latest
```

10. Lancer cette commande :

npm run dev -- --host

11. Lancer l'application sur l'url <http://localhost:5173/>

Arborescence

C:\ReactProjects\MiniEcommerceApp

```
|-- public
|   |-- images
|   |   |-- product1.jpg
|   |   |-- product2.jpg
|   |   |-- product6.jpg
|-- src
|   |-- main.jsx
|   |-- App.jsx
|   |-- data
|   |   |-- products.json
|   |-- context
|   |   |-- CartContext.jsx
|   |-- components
|   |   |-- Header.jsx
|   |   |-- ProductCard.jsx
|   |   |-- ProductList.jsx
|   |-- pages
|   |   |-- Home.jsx
|   |   |-- ProductDetail.jsx
|   |   |-- Cart.jsx
|   |   |-- Checkout.jsx
|-- server
|   |-- package.json
|   |-- index.js
```

```
└─ orders.json
```

Backend (pour sauvegarder les commandes)

Ouvrez un nouveau dossier server :

```
cd server
```

```
npm init -y
```

```
npm install express body-parser cors
```

Créez un fichier **orders.json** initial contenant [].

Lancer backend :

```
node index.js
```

Lancer frontend (à la racine) :

```
npm run dev
```

Étape 1 — Page d'accueil : afficher la liste des produits

Objectif : afficher une liste de produits à partir d'un fichier JSON. Chaque carte produit montre image, titre, prix et bouton « Détails ».

Instructions aux participants :

- Créez src/data/products.json avec 6 produits.
- Créez des composants ProductList et ProductCard.
- Configurez routing basique (/ -> Home).

Fichiers / Code

src/data/products.json

```
[  
 {  
   "id": 1,  
   "title": "Montre Classique",  
   "description": "Montre élégante en acier inoxydable.",  
   "price": 79.99,  
   "image": "/images/product1.jpg",  
   "category": "accessories"  
 },  
 {  
   "id": 2,  
   "title": "Casque Bluetooth",  
   "description": "Casque sans fil, autonomie 20h.",  
   "price": 59.99,  
   "image": "/images/product2.jpg",  
 }]
```

```
"category": "electronics"
},
{
  "id": 3,
  "title": "Sac à dos urbain",
  "description": "Sac avec compartiment pour ordinateur 15\".",
  "price": 49.99,
  "image": "/images/product3.jpg",
  "category": "fashion"
},
{
  "id": 4,
  "title": "Lunettes de soleil",
  "description": "Lunettes anti-UV, design moderne.",
  "price": 29.99,
  "image": "/images/product4.jpg",
  "category": "fashion"
},
{
  "id": 5,
  "title": "Clavier mécanique",
  "description": "Clavier LED, switches tactiles.",
  "price": 89.99,
  "image": "/images/product5.jpg",
  "category": "electronics"
},
{
  "id": 6,
  "title": "Tasse thermique",
  "description": "Garde votre boisson chaude pendant des heures.",
  "price": 19.99,
  "image": "/images/product6.jpg",
  "category": "home"
}
]
```

//src/components/ProductCard.jsx

```
import React from "react";
import { Link } from "react-router-dom";

export default function ProductCard({ product }) {
  return (
    <div className="product-card" style={{border:"1px solid #ddd", padding:12, borderRadius:8}}>
      <img src={product.image} alt={product.title} style={{width:"100%", height:180, objectFit:"cover", borderRadius:6}}/>
      <h3>{product.title}</h3>
      <p style={{minHeight:40}}>{product.description}</p>
      <p><strong>{product.price.toFixed(2)} €</strong></p>
      <div style={{display:"flex", gap:8}}>
        <Link to={`/product/${product.id}`}><button>Voir</button></Link>
        </div>
      </div>
    );
}
```

//src/components/ProductList.jsx

```
import React from "react";
import ProductCard from "./ProductCard";

export default function ProductList({ products }) {
  return (
    <div className="product-list" style={{display:"grid", gridTemplateColumns:"repeat(3,1fr)", gap:16}}>
      {products.map(p => <ProductCard key={p.id} product={p} />)}
    </div>
  );
}
```

```
//src/pages/Home.jsx
import React, { useEffect, useState } from "react";
import ProductList from "../components/ProductList";
import productsData from "../data/products.json";

export default function Home() {
  const [products, setProducts] = useState([]);

  useEffect(() => {
    // on charge depuis le fichier JSON local
    setProducts(productsData);
  }, []);

  return (
    <div style={{padding:20}}>
      <h1>Catalogue</h1>
      <ProductList products={products} />
    </div>
  );
}
```

```
//src/main.jsx
import React from "react";
import { createRoot } from "react-dom/client";
import { createBrowserRouter, RouterProvider } from "react-router-dom";
import Home from "./pages/Home";
import App from "./App";
import ProductDetail from "./pages/ProductDetail";
import Cart from "./pages/Cart";
import Checkout from "./pages/Checkout";

const router = createBrowserRouter([
  {
    path: "/",
    element: <App/>,
    children: [
      { index: true, element: <Home/> },
      { path: "product/:id", element: <ProductDetail/> },
      { path: "cart", element: <Cart/> },
      { path: "checkout", element: <Checkout/> }
    ]
  }
]);
```

```
createRoot(document.getElementById("root")).render(  
  <React.StrictMode>  
    <RouterProvider router={router} />  
  </React.StrictMode>  
)
```

```
//src/App.jsx  
import React from "react";  
import { Outlet } from "react-router-dom";  
import Header from "./components/Header";  
import { CartProvider } from "./context/CartContext";  
  
export default function App() {  
  return (  
    <CartProvider>  
      <Header />  
      <main style={{padding:20}}>  
        <Outlet />  
      </main>  
    </CartProvider>  
  );  
}
```

```
//src\components\Header.jsx  
import React from "react";  
import { Link } from "react-router-dom";  
import { useCart } from "../context/CartContext";  
  
export default function Header(){  
  const { cart } = useCart();  
  const totalItems = cart.reduce((s, i) => s + i.quantity, 0);  
  
  return (  
    <header style={{display:"flex", justifyContent:"space-between", alignItems:"center", padding:12, borderBottom:"1px solid #eee"}}>  
      <Link to="/"><h2>Mini E-commerce</h2></Link>  
      <nav>  
        <Link to="/cart">Panier ({totalItems})</Link>  
      </nav>  
    </header>  
  );  
}
```

```
// src\context\CartContext.jsx  
import React, { createContext, useState, useContext } from "react";  
  
// Création du contexte  
export const CartContext = createContext();  
  
// Custom hook pour utiliser le panier  
export function useCart() {  
  const context = useContext(CartContext);  
  if (!context) {
```

```
throw new Error("useCart must be used within CartProvider");
}

return context;
}

// Provider pour englober l'app et partager le panier
export function CartProvider({ children }) {
  const [cart, setCart] = useState([]);

  // Fonction pour ajouter un produit
  const addToCart = (product) => {
    setCart((prevCart) => {
      const existing = prevCart.find((item) => item.id === product.id);
      if (existing) {
        // Incrémenter la quantité si déjà présent
        return prevCart.map((item) =>
          item.id === product.id
            ? { ...item, quantity: item.quantity + 1 }
            : item
        );
      } else {
        return [...prevCart, { ...product, quantity: 1 }];
      }
    });
  };

  // Fonction pour supprimer un produit
  const removeFromCart = (id) => {
    setCart((prevCart) => prevCart.filter((item) => item.id !== id));
  };

  // Fonction pour augmenter la quantité
  const increaseQty = (id) => {
    setCart((prevCart) =>
      prevCart.map((item) =>
        item.id === id
          ? { ...item, quantity: item.quantity + 1 }
          : item
      )
    );
  };

  // Fonction pour diminuer la quantité
  const decreaseQty = (id) => {
    setCart((prevCart) =>
      prevCart.map((item) =>
        item.id === id && item.quantity > 1
          ? { ...item, quantity: item.quantity - 1 }
          : item
      )
    );
  };

  // Calculer le total
  const total = cart.reduce((sum, item) => sum + item.price * item.quantity, 0);

  return (
    <CartContext.Provider value={{ cart, addToCart, removeFromCart, increaseQty, decreaseQty, total }}>
      {children}
    </CartContext.Provider>
  );
}
```

```
// src/pages/Cart.jsx

import { useCart } from "../context/CartContext";

export default function Cart() {
  const { cart, removeFromCart, increaseQty, decreaseQty, total } = useCart();

  return (
    <div style={{ padding: "20px" }}>
      <h1>🛒 Mon Panier</h1>

      {cart.length === 0 ? (
        <p>Votre panier est vide.</p>
      ) : (
        <div>
          <table border="1" cellPadding="8" style={{ width: "100%", borderCollapse: "collapse" }}>
            <thead>
              <tr>
                <th>Produit</th>
                <th>Pri</th>
                <th>Quantité</th>
                <th>Sous-total</th>
                <th>Action</th>
              </tr>
            </thead>

            <tbody>
              {cart.map((item) => (
                <tr key={item.id}>
                  <td>{item.title}</td>
                  <td>{item.price} TND</td>
                  <td>
                    <button onClick={() => decreaseQty(item.id)}>-</button>
                    <span style={{ margin: "0 10px" }}>{item.quantity}</span>
                    <button onClick={() => increaseQty(item.id)}>+</button>
                  </td>
                  <td>{(item.price * item.quantity).toFixed(2)} TND</td>
                  <td>
                    <button onClick={() => removeFromCart(item.id)}>❌ Supprimer</button>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>
        </div>
      )}
    </div>
  );
}
```

```
// src/components/Checkout.jsx
import { useState } from "react";
import { useCart } from "../context/CartContext";

export default function Checkout() {
  const { cart, totalPrice, clearCart } = useCart();
  const [form, setForm] = useState({
    fullname: "",
    email: "",
    address: ""
  });
  const [orderConfirmed, setOrderConfirmed] = useState(false);

  const handleChange = (e) => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = (e) => {
    e.preventDefault();

    if (!form.fullname || !form.email || !form.address) {
      alert("Veuillez remplir tous les champs.");
      return;
    }

    // Simuler validation commande
    setOrderConfirmed(true);
    clearCart();
  };

  if (orderConfirmed) {
    return (
      <div className="checkout">
        <h2>🎉 Merci pour votre commande !</h2>
        <p>Votre commande a été validée avec succès.</p>
      </div>
    );
  }

  return (
    <div className="checkout">
      <h2>Validation de la commande</h2>

      {cart.length === 0 ? (
        <p>Aucun article dans le panier.</p>
      ) : (
        <>
          <h3>Total : {totalPrice.toFixed(2)} €</h3>

          <form onSubmit={handleSubmit} className="checkout-form">
            <label>Nom complet :</label>
            <input
              type="text"
              name="fullname"
              value={form.fullname}
              onChange={handleChange}
              placeholder="Votre nom"
            />

            <label>Email :</label>
            <input
              type="email"
              name="email"
            />
          </form>
        </>
      )}
    </div>
  );
}
```

```
        value={form.email}
        onChange={handleChange}
        placeholder="Votre e-mail"
      />

      <label>Adresse :</label>
      <textarea
        name="address"
        value={form.address}
        onChange={handleChange}
        placeholder="Adresse de livraison"
      ></textarea>

      <button type="submit" className="btn-pay">
        Confirmer et payer
      </button>
    </form>
  </>
}

);
}
```

```
// src\pages\ProductDetail.jsx
import React, { useContext } from "react";
import { useParams, Link } from "react-router-dom";
import productsData from "../data/products.json"; // fichier JSON des produits
import { CartContext } from "../context/CartContext";

function ProductDetail() {
  const { id } = useParams();
  const { addToCart } = useContext(CartContext);

  // Trouver le produit correspondant
  const product = productsData.find((p) => p.id === parseInt(id));

  if (!product) {
    return (
      <div>
        <h2>Produit non trouvé</h2>
        <Link to="/">Retour à l'accueil</Link>
      </div>
    );
  }

  return (
    <div className="product-detail">
      <h2>{product.name}</h2>
      <img
        src={product.image}
        alt={product.name}
        style={{ width: "300px", height: "300px" }}
      />
      <p>{product.description}</p>
      <p>Prix : {product.price} €</p>
      <button onClick={() => addToCart(product)}>Ajouter au panier</button>
      <br />
      <Link to="/">Retour à l'accueil</Link>
    </div>
  );
}
```

```
 );
}

export default ProductDetail;
```

Étape 2 — Page détail du produit + routing dynamique

Objectif : afficher la page détail pour un produit, possibilité d'ajouter au panier depuis la page détail.

Instructions :

- Créez src/pages/ProductDetail.jsx
- Récupérer l'id via useParams, charger depuis products.json, permettre réglage quantité et bouton Ajouter au panier.

```
//src/pages/ProductDetail.jsx
import React, { useState } from "react";
import { useParams } from "react-router-dom";
import productsData from "../data/products.json";
import { useCart } from "../context/CartContext";

export default function ProductDetail(){
  const { id } = useParams();
  const product = productsData.find(p => String(p.id) === String(id));
  const [qty, setQty] = useState(1);
  const { addToCart } = useCart();

  if (!product) return <div>Produit introuvable</div>

  return (
    <div style={{display:"flex", gap:20}}>
      <img src={product.image} alt={product.title} style={{width:360, height:360, objectFit:"cover"}}/>
      <div>
        <h2>{product.title}</h2>
        <p>{product.description}</p>
        <p><strong>{product.price.toFixed(2)} €</strong></p>
        <div style={{marginTop:12}}>
          <label>Quantité: </label>
          <input type="number" min="1" value={qty} onChange={e=>setQty(Math.max(1, Number(e.target.value)))}>
        </div>
        <button style={{marginTop:8}} onClick={() => addToCart(product, qty)}>Ajouter au panier</button>
      </div>
    </div>
  );
}
```

Étape 3 — Le panier (affichage, modification quantité, suppression)

Objectif : gérer le panier global avec Context API : voir items, modifier quantité, supprimer, calcul du total.

Instructions :

- Créez page Cart.jsx et affichez les actions possibles.
- Calculez total = \sum price * qty.

```
//src/pages/Cart.jsx
import React from "react";
import { Link, useNavigate } from "react-router-dom";
import { useCart } from "../context/CartContext";

export default function Cart(){
  const { cart, updateQuantity, removeFromCart, clearCart } = useCart();
  const navigate = useNavigate();

  const total = cart.reduce((s, i) => s + i.price * i.quantity, 0);

  if (cart.length === 0) return (
    <div>
      <h2>Votre panier est vide</h2>
      <Link to="/">Retour au catalogue</Link>
    </div>
  );

  return (
    <div>
      <h2>Panier</h2>
      <table style={{width:"100%", borderCollapse:"collapse"}}>
        <thead>
          <tr>
            <th>Produit</th><th>Prix</th><th>Quantité</th><th>Sous-total</th><th></th>
          </tr>
        </thead>
        <tbody>
          {cart.map(item => (
            <tr key={item.id}>
              <td style={{padding:8}}>{item.title}</td>
              <td style={{padding:8}}>{item.price.toFixed(2)} €</td>
              <td style={{padding:8}}>
                <input type="number" min="1" value={item.quantity} onChange={e => updateQuantity(item.id, Math.max(1, Number(e.target.value)))} style={{width:60}}/>
              </td>
              <td style={{padding:8}}>{(item.price * item.quantity).toFixed(2)} €</td>
              <td style={{padding:8}}><button onClick={() => removeFromCart(item.id)}>Supprimer</button></td>
            </tr>
          ))}
        </tbody>
      </table>
      <h3>Total: {total.toFixed(2)} €</h3>
      <div style={{display:"flex", gap:8}}>
        <button onClick={() => { clearCart(); }}>Vider le panier</button>
        <button onClick={() => navigate("/checkout")}>Passer à la commande</button>
      </div>
    </div>
  );
}
```

Étape 4 — Checkout et enregistrement des commandes dans orders.json

Objectif : collecter les infos client, envoyer la commande au backend Express qui stocke les commandes dans server/orders.json.

Instructions :

- Créez src/pages/Checkout.jsx avec un formulaire simple (nom, email, adresse).
- POST /api/orders vers le backend (localhost:5000 par défaut).
- Backend : server/index.js — accepte POST et écrit dans orders.json.

```
//src/pages/Checkout.jsx
import React, { useState } from "react";
import { useNavigate } from "react-router-dom";
import { useCart } from "../context/CartContext";

export default function Checkout(){
  const { cart, clearCart } = useCart();
  const [form, setForm] = useState({ name: "", email: "", address: "" });
  const [loading, setLoading] = useState(false);
  const navigate = useNavigate();

  const total = cart.reduce((s,i) => s + i.price * i.quantity, 0);

  async function handleSubmit(e){
    e.preventDefault();
    if (cart.length === 0) return alert("Panier vide");
    setLoading(true);
    const order = {
      customer: form,
      items: cart,
      total,
      createdAt: new Date().toISOString()
    };

    try {
      const res = await fetch("http://localhost:5000/api/orders", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(order)
      });
      if (!res.ok) throw new Error("Erreur sauvegarde commande");
      clearCart();
      alert("Commande enregistrée !");
      navigate("/");
    } catch (err) {
      console.error(err);
      alert("Impossible d'enregistrer la commande. Vérifiez le backend.");
    } finally {
      setLoading(false);
    }
  }

  return (
    <div style={{maxWidth:600}>
      <h2>Checkout</h2>
      <div>
        <h4>Résumé</h4>
        {cart.map(i => <div key={i.id}>{i.title} x {i.quantity} — {(i.price * i.quantity).toFixed(2)} €</div>)}
        <p>Total: {total.toFixed(2)} €</p>
      </div>
    </div>
  )
}
```

```
<form onSubmit={handleSubmit} style={{display:"grid", gap:8, marginTop:12}}>
  <input placeholder="Nom complet" required value={form.name} onChange={e=>setForm({...form,
name:e.target.value})}/>
  <input placeholder="Email" type="email" required value={form.email} onChange={e=>setForm({...form,
email:e.target.value})}/>
  <textarea placeholder="Adresse" required value={form.address} onChange={e=>setForm({...form,
address:e.target.value})}/>
  <button type="submit" disabled={loading}>{loading ? "Enregistrement..." : "Valider la commande"}</button>
</form>
</div>
);
}
```

Backend — server/index.js

```
// C:\ReactProjects\MiniEcommerceApp\server\index.js

const express = require("express");
const fs = require("fs");
const path = require("path");
const cors = require("cors");
const bodyParser = require("body-parser");

const app = express();
app.use(cors());
app.use(bodyParser.json());

const ORDERS_FILE = path.join(__dirname, "orders.json");

function readOrders(){
try {
  const s = fs.readFileSync(ORDERS_FILE, "utf8");
  return JSON.parse(s || "[]");
} catch (e) {
  return [];
}
```

```
}
```

```
}
```

```
function writeOrders(orders){  
    fs.writeFileSync(ORDERS_FILE, JSON.stringify(orders, null, 2));  
}  
  
app.post("/api/orders", (req, res) => {  
    const order = req.body;  
    if (!order || !order.items) return res.status(400).json({ error: "Données invalides" });  
  
    const orders = readOrders();  
    // ajouter un id simple  
    const newOrder = { id: orders.length + 1, ...order };  
    orders.push(newOrder);  
    writeOrders(orders);  
  
    res.status(201).json({ message: "Commande enregistrée", order: newOrder });  
});
```

```
const PORT = process.env.PORT || 5000;
```

```
app.listen(PORT, () => console.log(`Server orders running on  
http://localhost:${PORT}`));
```

Remarque pédagogique : Le backend est volontairement minimal — il écrit dans un fichier orders.json. En production, on utiliserait une base de données. Pour l'atelier, montrez le contenu de orders.json après chaque commande.

Étape 5 — Recherche, filtrage et bonnes pratiques

Objectif : ajouter recherche et filtres (par catégorie), et appliquer quelques optimisations (keys, destructuring).

Instructions :

- Dans Home.jsx, ajoutez un champ search et un select catégorie.
- Filtrez le tableau productsData en appliquant toLowerCase().

Extrait modifié de src/pages/Home.jsx (ajout recherche / filter):

```
// ... même chemin que précédemment

import React, { useEffect, useState, useMemo } from "react";
import ProductList from "../components/ProductList";
import productsData from "../data/products.json";

export default function Home() {

  const [products, setProducts] = useState([]);
  const [q, setQ] = useState("");
  const [cat, setCat] = useState("");

  useEffect(() => setProducts(productsData), []);

  const categories = useMemo(() => {
    return [...new Set(productsData.map(p => p.category))];
  }, []);

  const filtered = products.filter(p => {
    const matchesQ = p.title.toLowerCase().includes(q.toLowerCase()) ||
      p.description.toLowerCase().includes(q.toLowerCase());
    const matchesCat = cat ? p.category === cat : true;
    return matchesQ && matchesCat;
  });

}
```

```
return (

  <div style={{padding:20}>

    <h1>Catalogue</h1>

    <div style={{display:"flex", gap:8, marginBottom:12}}>

      <input placeholder="Rechercher..." value={q} onChange={e=>setQ(e.target.value)} />

      <select value={cat} onChange={e=>setCat(e.target.value)}>

        <option value="">Toutes catégories</option>

        {categories.map(c => <option key={c} value={c}>{c}</option>)}

      </select>

    </div>

    <ProductList products={filtered} />

  </div>

);

}
```

Étape 6 — Persistance, sauvegarde des commandes et test

Objectif : tester le flux complet : ajouter au panier, passer commande, puis vérifier server/orders.json.

Instructions :

1. Démarrez le backend : node server/index.js (ou npm start si script défini).
2. Démarrez le frontend : npm run dev.
3. Ouvrez <http://localhost:5173/> (ou port Vite) — voir catalogue.
4. Allez sur un produit → ajouter au panier → panier → passer à la commande → remplir le formulaire → valider.
5. Ouvrez server/orders.json : une nouvelle commande doit être ajoutée.

Mini E-commerce / Catalogue de produits

Objectif pédagogique : Combiner tous les concepts et préparer à des projets réels.

Fonctionnalités principales :

- Affichage d'une liste de produits depuis un fichier JSON ou une API.
- Filtrage et recherche de produits.
- Ajout au panier et calcul du total.
- Affichage du détail d'un produit via routing (react-router-dom).
- Gestion du panier dans un contexte global (Context API).

Concepts React couverts :

- Composants imbriqués et communication parent-enfant.
- Props et state avancé.
- Routing avec react-router-dom.
- Context API pour état global (panier).
- Gestion d'événements complexes.
- Optimisation et bonnes pratiques (key, map, destructuring).