Atelier 7 NextJS : Shopping Cart

Dans cet atelier on va utiliser use-shopping-cart qui est une bibliothèque incluant l'état du panier. Use Shopping Cart est une bibliothèque de Hooks afin de fournir une première méthode API de gestion de l'état et de la logique du panier d'achat.

# Installation du package use-shopping-cart

Site : <https://useshoppingcart.com/>

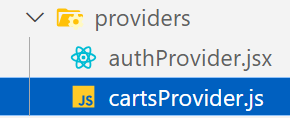
Dans le terminal taper :

npm install use-shopping-cart

# Configuration du CartProvider

Nous devons envelopper le composant client avec <CartProvider/> de use-shopping-cart. Cela nous permettra d'accéder à l'état et à la logique du panier à partir de n'importe quelle page ou composant de notre application.

Sous /src/providers créer le fichier cartsProvider.jsx



"use client";

import { CartProvider} from "use-shopping-cart";

export default function CartsProvider({ children }) {

  return (

    <CartProvider shouldPersist={true} >

      {children}

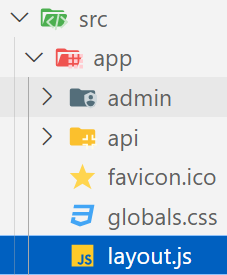
    </CartProvider>

  );

}

shouldPersist est mise à true pour stocker les articles du panier dans le localStorage.

Ensuite, intégrez CartProvider au composant layout principal (sous /src/app). Cela nécessite d'encapsuler les composants de corps du fichier layout, en connectant efficacement le CartProvider (marqué « use client ») avec le composant de mise en page côté serveur.



import { Inter } from "next/font/google";

import "./globals.css";

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

import Menu from '@/components/client/menu';

import connectDB from '@/lib/connectDB';

import AuthProvider from '@/providers/authProvider';

import CartsProvider from '@/providers/cartsProvider';

const inter = Inter({ subsets: ["latin"] });

export const metadata = {

  title: "Create Next App",

  description: "Generated by create next app",

};

export default function RootLayout({ children }) {

await connectDB();

  return (

    <html lang="en">

      <body className={inter.className}>

      <CartsProvider>

      <AuthProvider>

      <Menu/>

        {children}

        </AuthProvider>

        </CartsProvider>

      </body>

    </html>

  );

}

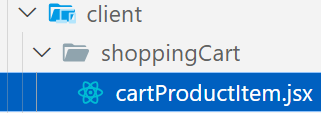
# Component cartProductItem

D’abord installer le package qui affichera le rating :

npm i @smastrom/react-rating

Référence : <https://www.npmjs.com/package/@smastrom/react-rating>

Sous le dossier /components/client/ créer le dossier shoppingCart puis y est le composant cartProductItem.jsx



"use client";

import React,{ useState } from "react";

import { Rating } from '@smastrom/react-rating'

import '@smastrom/react-rating/style.css'

import { useShoppingCart } from "use-shopping-cart" ;

const CartProductItem= ({livre})=> {

    const { addItem } = useShoppingCart();

    const [quantity, setQuantity] = useState(1);

    const decreaseQuantity = () => {

      if (quantity > 1) {

        setQuantity(quantity - 1);

      }

    };

    const increaseQuantity = () => {

      if (quantity < livre.qtestock) {

        setQuantity(quantity + 1);

      }

      else window.alert("Not Enough Stock")

    };

    const addToCart = (livre) => {

      const target = {

        id:livre.\_id,

        title:  livre.titre,

        image: livre.couverture,

        price : livre.prix

        };

        if(quantity<livre.qtestock){

       addItem(target, { count: quantity }).then((()=>{

        console.log('Item added to cart:', target);

        setQuantity(1)

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

      }

       else window.alert("Not Enough Stock")

      };

    return (

    <article className="col-sm-3 mt-5">

        <div className="card">

          <img

            src={livre?.couverture}

            className="card-img-top p-5"

            alt={livre.titre}

           />

        </div>

       <div className="text-center">

       <div>{livre.titre}</div>

       <div>Prix : {livre.prix} TND </div>

       <div className="flex flex-col items-center justify-center">

       <div className="flex"> <Rating style={{ maxWidth: 100 }} value={5} /> </div>

       </div>

       </div>

      <div className="text-center">

        <button className="mr-5"

          onClick={decreaseQuantity}

         >

          -

        </button>

        <span>{quantity}</span>

        <button className="ml-5"

          onClick={increaseQuantity}

        >

          +

        </button>

      </div>

      <div className="text-center">

        <button className="btn btn-warning" disabled={livre.qtestock <= 1} onClick={() => addToCart(livre)}>

          Add to cart

        </button>

        </div>

    </article>

     );

}

export default CartProductItem;

# Page cartProducts

Créer le dossier shoppingClient sous le répertoire /src/app/

Puis le dossier cartProducts

Dans ce dossier créer page.js

## 

import React from "react";

import CartLivreItem from '@/components/client/shoppingCart/cartProductItem';

import {fetchLivres} from "@/services/livreService";

async function getLivres(){

    const data=await fetchLivres()

    return data;

}

const CartLivresPage= async ()=> {

    const livres = await getLivres();

    return (

      <>

      <div className="row">

            {livres && livres?.map((livre) => (

              <CartLivreItem key={livre?.\_id} livre={livre} />

            ))}

          </div>

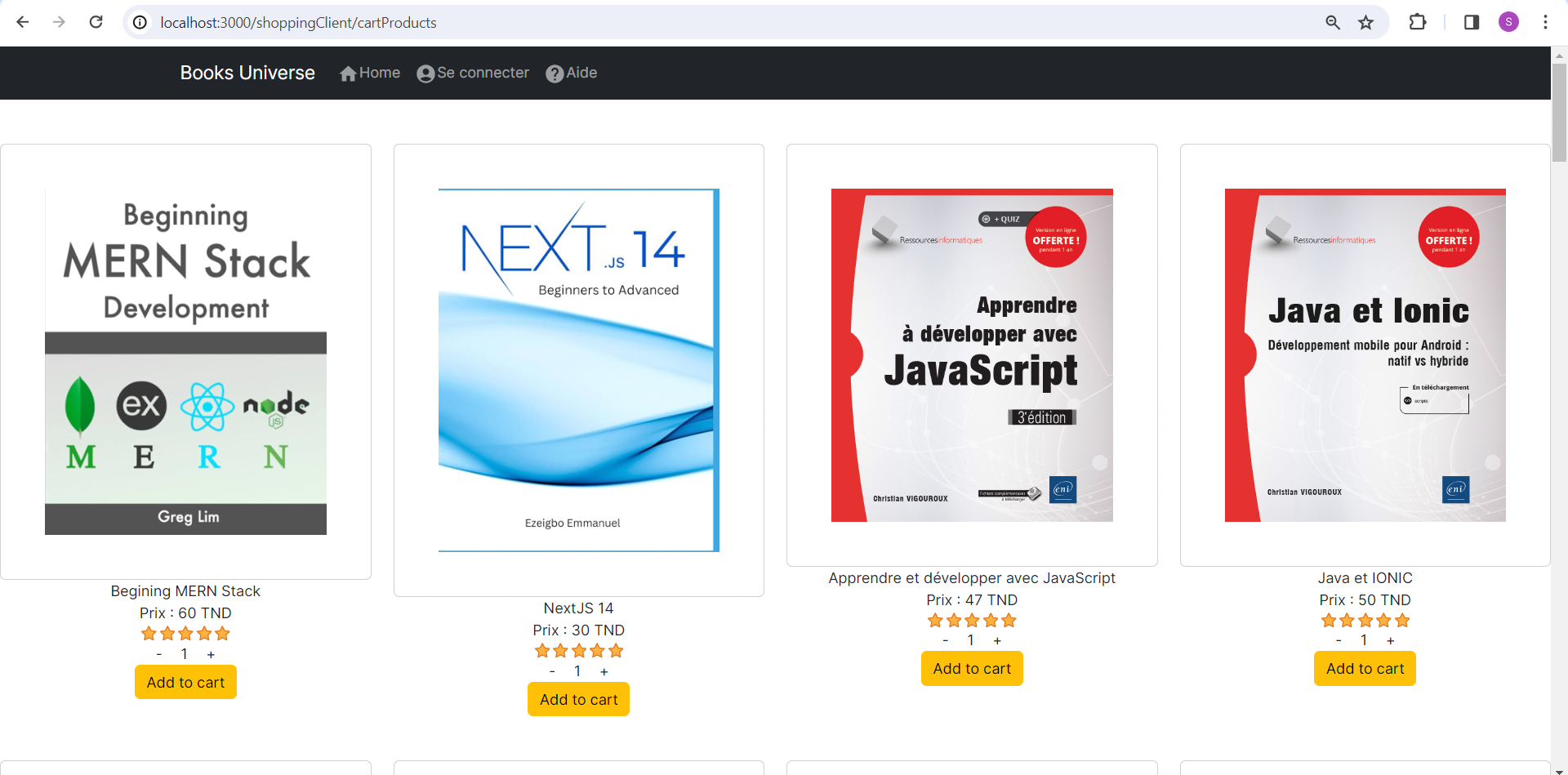
     </>

    )

}

export default CartLivresPage;

<http://localhost:3000/shoppingClient/cartProducts>



# Ajouter l’icône dans le menu



'use client';

import Container from 'react-bootstrap/Container';

import Nav from 'react-bootstrap/Nav';

import Navbar from 'react-bootstrap/Navbar';

import Link from 'next/link';

import HomeIcon from '@mui/icons-material/Home';

import AccountCircleIcon from '@mui/icons-material/AccountCircle';

import HelpIcon from '@mui/icons-material/Help';

import LogoutIcon from '@mui/icons-material/Logout';

import { useSession, signIn, signOut } from 'next-auth/react'

import { useRouter } from 'next/navigation'

import ShoppingBasketIcon from '@mui/icons-material/ShoppingBasket';

import ShoppingCartIcon from '@mui/icons-material/ShoppingCart';

import { useShoppingCart } from "use-shopping-cart" ;

function Menu() {

  const router = useRouter()

  const { handleCartClick, cartCount } = useShoppingCart();

  const { data: session } = useSession()

  return (

    <>

      <Navbar bg="dark" data-bs-theme="dark">

        <Container>

          <Navbar.Brand href="#home">Books Universe</Navbar.Brand>

          <Nav className="me-auto">

            <Nav.Link as={Link} href="/"><HomeIcon/>Home</Nav.Link>

                        {session? <Nav.Link onClick={() => signOut()}><LogoutIcon/>Se déconnecter</Nav.Link>

            : <Nav.Link onClick={() => signIn()}><AccountCircleIcon/>Se connecter</Nav.Link>

            }

   <Nav.Link onClick={() => router.push('/shoppingClient/cartProducts')}>

            <ShoppingBasketIcon style={{ color: 'aqua' }}/> Shopping Cart </Nav.Link>

         <Nav.Link as={Link} href="/"><HelpIcon/>Aide</Nav.Link>

          </Nav>

          <button className="relative" onClick={() => handleCartClick()}>

                <ShoppingCartIcon color="warning"/>

                <div className="rounded-full flex justify-center items-center bg-red-500 text-xs text-white absolute w-6 h-5 bottom-6 -right-1">

                {cartCount}

                </div>

          </button>

        </Container>

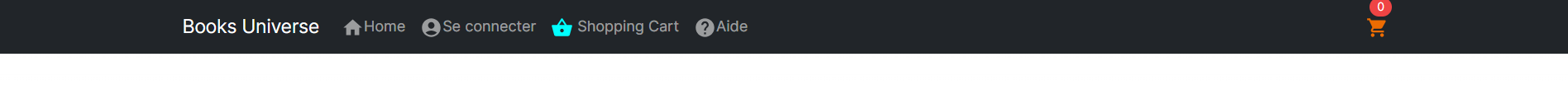
      </Navbar>

     </>

  );

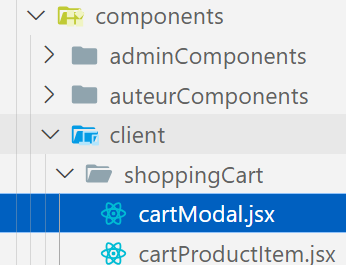
}

export default Menu;



# cartModal

Créer le fichier cartModal.jsx sous /src/components/client/shoppingCart



Nous pouvons utiliser la propriété shouldDisplayCart de use-shopping-cart pour contrôler la visibilité de notre panier. Cette propriété booléenne détermine si le panier doit être affiché ou masqué.

"use client";

import { useState } from "react";

import { useShoppingCart } from "use-shopping-cart";

import DeleteIcon from '@mui/icons-material/Delete';

import ClearIcon from '@mui/icons-material/Clear';

export default function cartModal() {

    const { shouldDisplayCart,cartDetails,cartCount , removeItem , clearCart, totalPrice,} = useShoppingCart();

    const [status, setStatus] = useState("idle");

  return (

    <div

      className={`bg-white flex flex-col absolute right-3 md:right-9 top-14 w-80 py-4 px-4 shadow-[0\_5px\_15px\_0\_rgba(0,0,0,.15)] rounded-md transition-opacity duration-500 ${

        shouldDisplayCart ? "opacity-100 z-10" : "opacity-0"

      }`}

    >

    {cartCount && cartCount > 0 ? (

        <>

        <div

            onClick={() => clearCart()}

            >

             <ClearIcon/>

        </div>

          {Object.values(cartDetails).map((item) => (

            <div className="flex items-center gap-4 mb-3" key={item.id}>

            <p>

                          <img

                          src={item.image}

                          alt={item.title}

                          style={{ width: '60px', height: '60px' }}

                         />

                         </p>

            <div>

             {item.title} <span className="text-xs">({item.quantity})</span>

            </div>

            <div className="ml-auto">{item.price} TND</div>

            <button className="hover:bg-emerald-50 transition-colors rounded-full duration-500 p-1"

            onClick={() => removeItem(item.id)}

            >

            <DeleteIcon color="secondary"/>

            </button>

          </div>

          ))}

                <article className="mt-3 flex flex-col">

                    Total : {totalPrice} TND

                   <button className="bg-emerald-50 hover:bg-emerald-500 hover:text-white transition-colors duration-500 text-emerald-500 py-3 px-5 rounded-md w-100">

                     {status !== "loading" ? "Proceed to checkout" : "Loading..."}

                  </button>

               </article>

        </>

      ) : (

        <div className="p-5">You have no items in your cart</div>

      )}

    </div>

  );

}

Une condition pour que le panier ait une opacité de 1 lorsque shouldDisplayCart est vrai et une opacité de 0 lorsque shouldDisplayCart est égal à false.

Les classes transition-opacity et duration-500 pour que la transition entre l'opacité soit un peu plus douce.

# Ajouter le composant dans le menu



'use client';

import Container from 'react-bootstrap/Container';

import Nav from 'react-bootstrap/Nav';

import Navbar from 'react-bootstrap/Navbar';

import Link from 'next/link';

import HomeIcon from '@mui/icons-material/Home';

import AccountCircleIcon from '@mui/icons-material/AccountCircle';

import HelpIcon from '@mui/icons-material/Help';

import LogoutIcon from '@mui/icons-material/Logout';

import { useSession, signIn, signOut } from 'next-auth/react'

import { useRouter } from 'next/navigation'

import ShoppingBasketIcon from '@mui/icons-material/ShoppingBasket';

import ShoppingCartIcon from '@mui/icons-material/ShoppingCart';

import { useShoppingCart } from "use-shopping-cart" ;

import CartModal from "./shoppingCart/cartModal";

function Menu() {

const router = useRouter()

  const { handleCartClick, cartCount } = useShoppingCart();

  const { data: session } = useSession()

  return (

    <>

      <Navbar bg="dark" data-bs-theme="dark">

        <Container>

          <Navbar.Brand href="#home">Books Universe</Navbar.Brand>

          <Nav className="me-auto">

            <Nav.Link as={Link} href="/"><HomeIcon/>Home</Nav.Link>

                        {session? <Nav.Link onClick={() => signOut()}><LogoutIcon/>Se déconnecter</Nav.Link>

            : <Nav.Link onClick={() => signIn()}><AccountCircleIcon/>Se connecter</Nav.Link>

            }

           <Nav.Link onClick={() => router.push('/shoppingClient/cartProducts')}>

            <ShoppingBasketIcon style={{ color: 'aqua' }}/> Shopping Cart </Nav.Link>

         <Nav.Link as={Link} href="/"><HelpIcon/>Aide</Nav.Link>

          </Nav>

          <button className="relative" onClick={() => handleCartClick()}>

                <ShoppingCartIcon color="warning"/>

                <div className="rounded-full flex justify-center items-center bg-red-500 text-xs text-white absolute w-6 h-5 bottom-6 -right-1">

                {cartCount}

                </div>

          </button>

**<CartModal />**

        </Container>

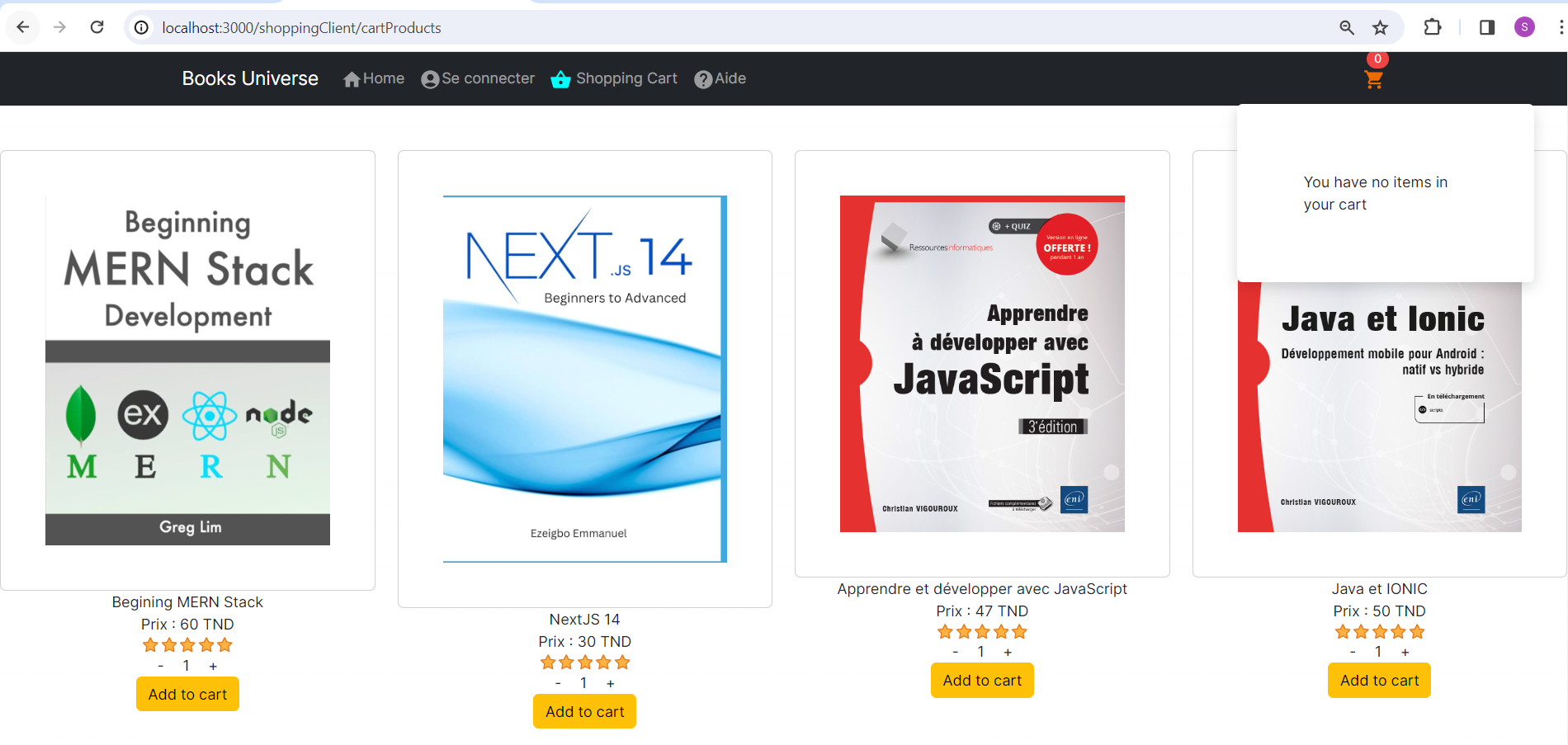
      </Navbar>

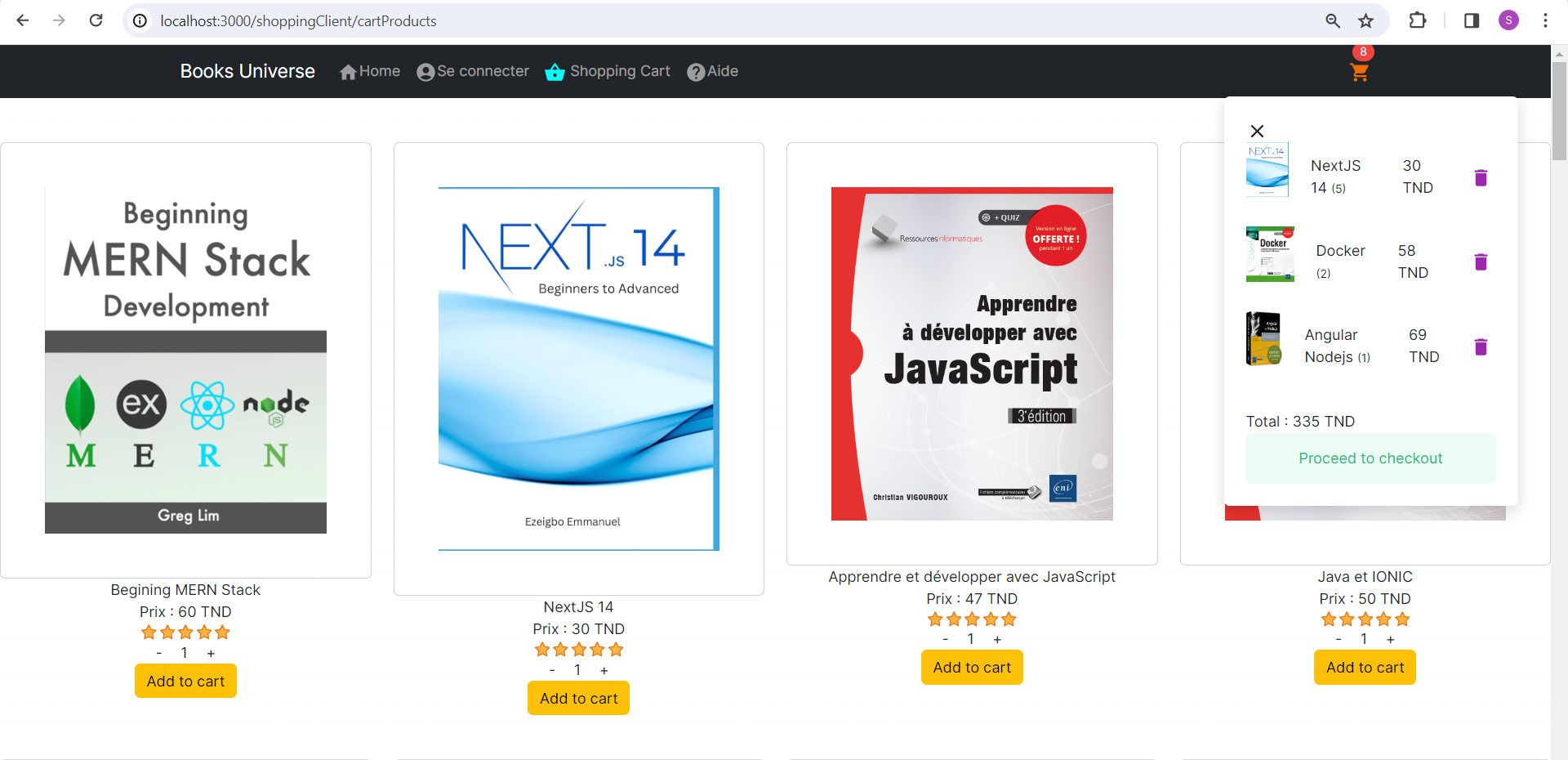
     </>

  );

}

export default Menu;





# Payement Stripe

### Site Stripe

Stripe est une plate-forme de traitement des paiements qui vous permet d'ajouter une page de paiement prédéfinie à un site Web et d'accepter et de gérer les paiements en ligne. Il est très populaire en raison de sa facilité d'utilisation, de sa documentation complète, de la prise en charge des paiements locaux, des options de personnalisation et de marque, des abonnements, de la facturation et de la prévention de la fraude.

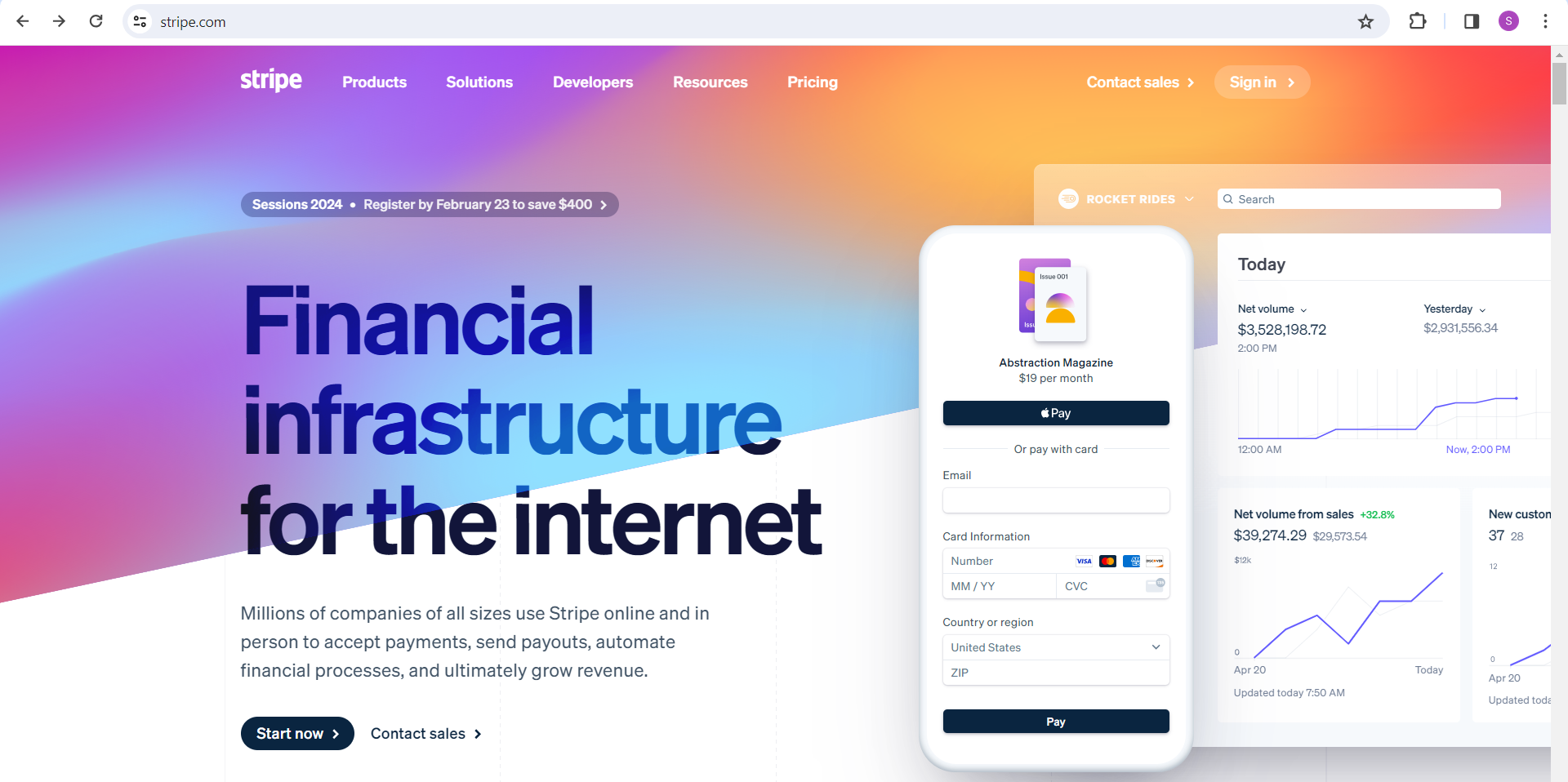
Un flux de paiement avec Stripe se divise en deux étapes :

1. Côté client, dans votre code frontend, vous collectez et tokenisez les informations de paiement du client à l'aide de Checkout ou Elements, puis envoyez le jeton résultant à votre serveur backend.

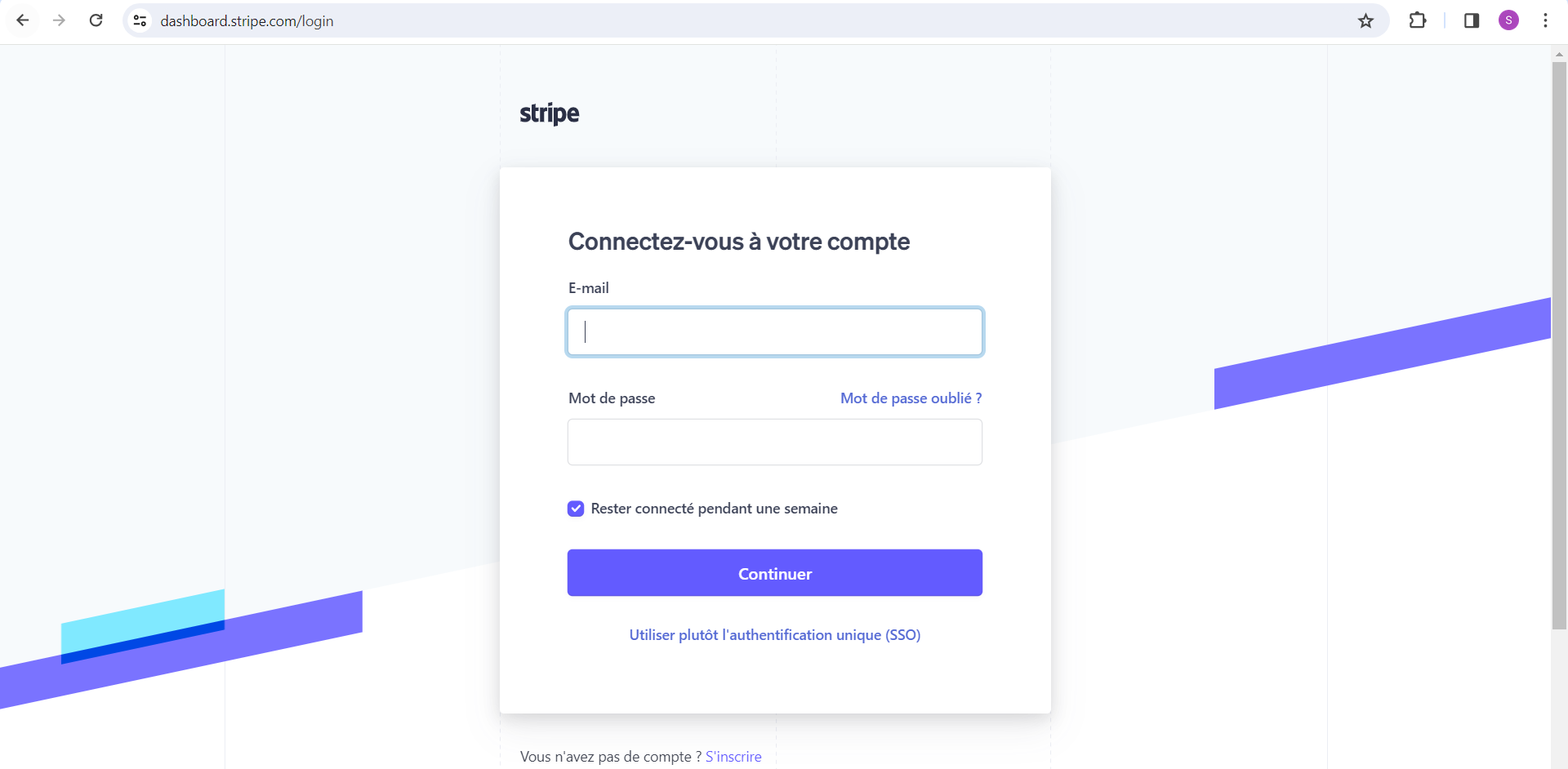
2. Côté serveur, dans votre code api, vous utilisez le jeton dans une requête API, par ex. pour créer une charge ou un client.

Créer un compte Stripe :

[https://stripe.com](https://stripe.com/fr-us)



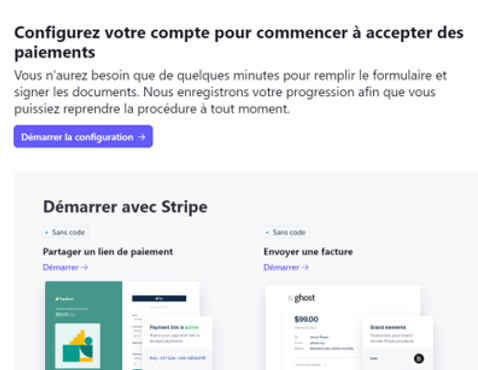




Remplir les étapes de l’inscription.

Puis :

<https://dashboard.stripe.com/test/dashboard>

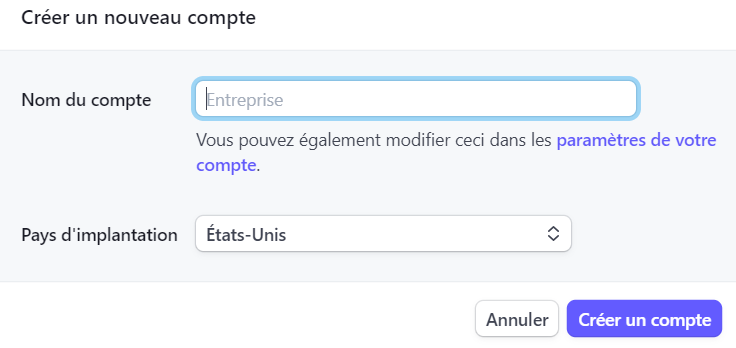


Cliquer sur :



Puis choisir :

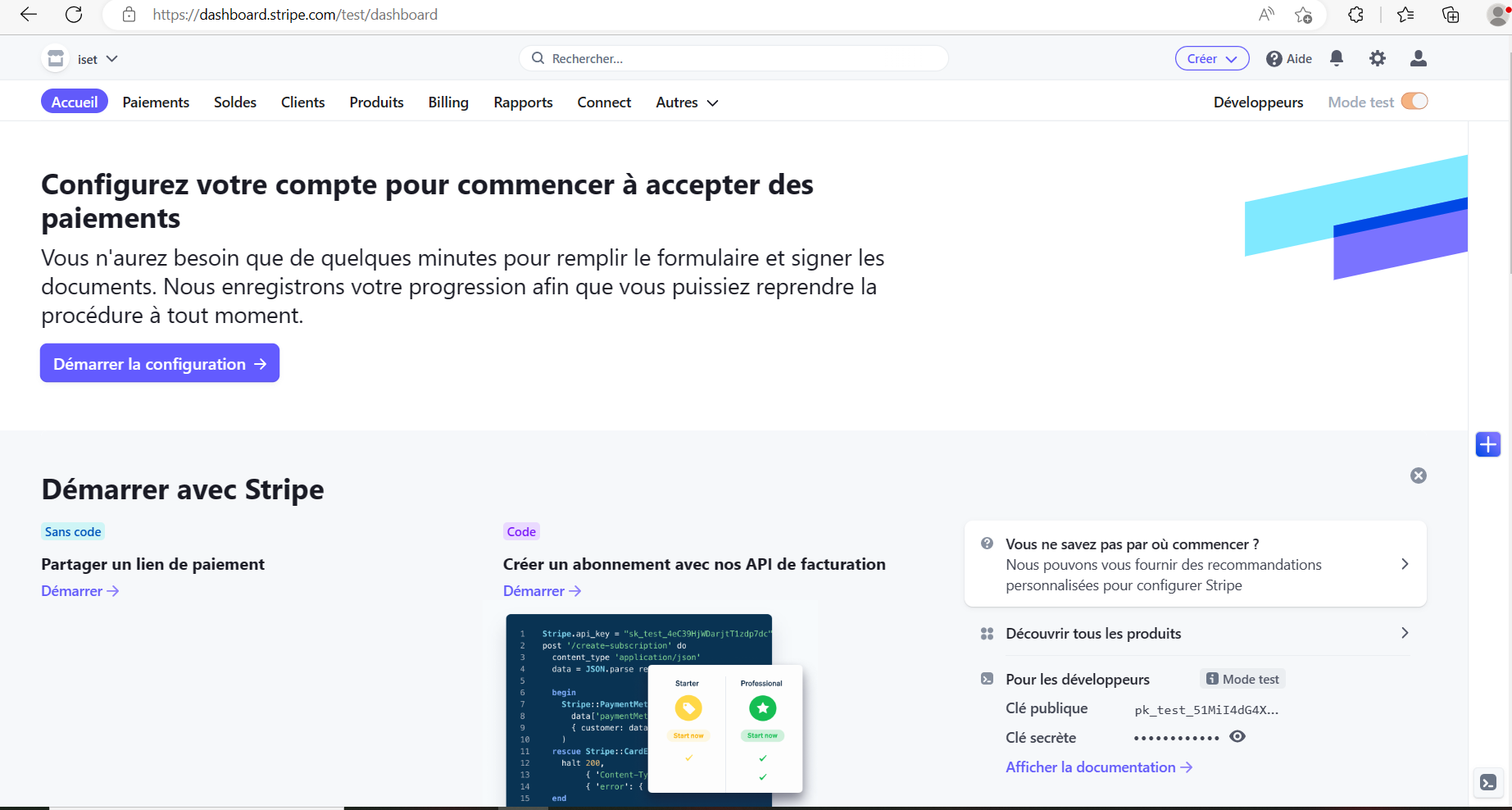




Donner un nom et enregistrer.



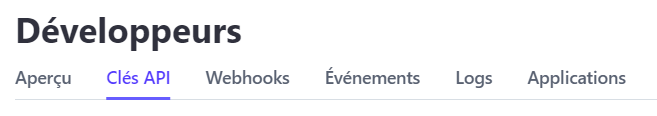
On obtient :



Ensuite, cliquer sur Développeurs (coin en haut à droite)



Puis cliquer sur Clés API





Puis copier les valeurs des clés.

### Projet NextJS

Ensuite dans le projet Next JS installer la dépendance :

npm install stripe @stripe/stripe-js

Dans le fichier .env créez la variable NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY et y coller la valeur de clé publique copiée.

Puis faire la même chose pour STRIPE\_SECRET\_KEY

Après, ajouter la variable NEXT\_PUBLIC\_URL Utilisée dans successUrl et cancelUrl par la suite.



NEXT**\_**PUBLIC**\_**STRIPE**\_**PUBLISHABLE**\_**KEY = pk\_test\_51KtYRUD3HS4vNAwatvmqAEXLKKX11UOcpkHfLnw9UPI9kZ7AJCOeLkqik61wHFXLmRGHUd4aNBvp45v82DpskKl300bMfznwlE

STRIPE**\_**SECRET**\_**KEY = sk\_test\_51KtYRUD3HS4vNAwa7ANL32HQqRTywhV3JHWIp3BxAIHv04bWoz22aKlRs9Q1L6znSX2i5fu5i3Xkl9i2Goz7jAkC00LL0T3lTL

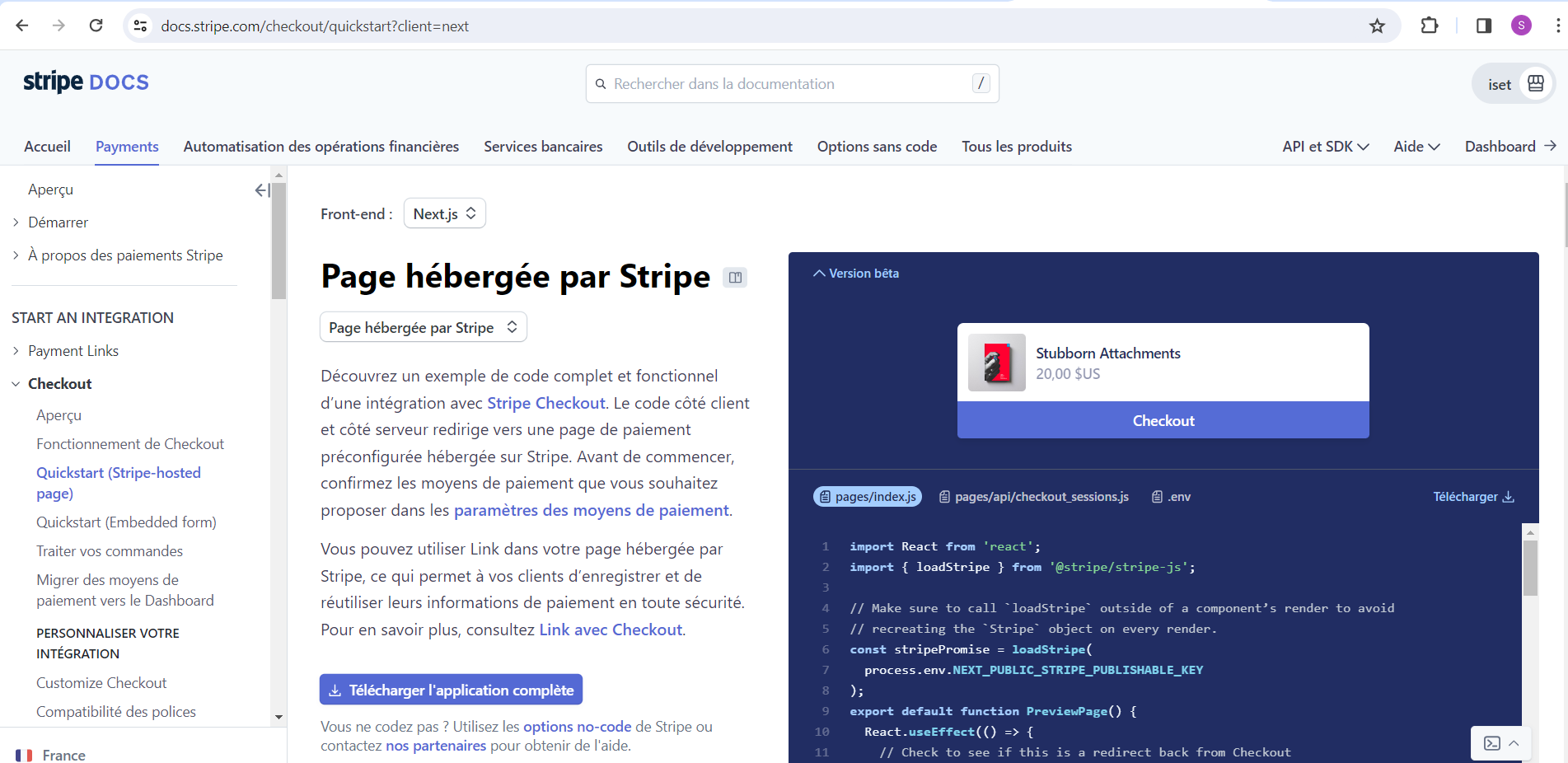
NEXT**\_**PUBLIC**\_**URL = http://localhost:3000

**Remarque** : le préfixe NEXT\_PUBLIC expose la variable au navigateur. Dans ce cas, ce n'est pas grave car votre clé publique n'est utilisée que pour identifier votre compte avec Stripe - ce n'est pas secret. Mais, assurez-vous de ne pas utiliser ce préfixe avec d'autres variables d'environnement censées rester secrètes.

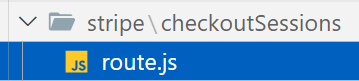
### checkout\_sessions.js

On peut retrouver le code original dans la documentation de Stripe, lequel il faut faire quelques modifications :

<https://stripe.com/docs/checkout/quickstart?client=next>



Sous **/src/app/api** créer le dossier **stripe** puis le dossier **checkoutSessions** ayantle fichier **route.js**



import { NextResponse} from "next/server";

import {headers} from "next/headers";

import Stripe from 'stripe';

if (!process.env.STRIPE\_SECRET\_KEY) {

    throw new Error('STRIPE\_TEST\_SECRET\_KEY is missing. Please set the environment variable.');

}

const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY);

export async function POST(req, res) {

    const headersList = headers();

    const {cartDetails} = await req.json();

    const cartDetailsArray= Object.values(cartDetails);

      const transformedItems = cartDetailsArray.map((item) => ({

        price\_data: {

            currency: "usd",

            product\_data: {

                name: item.title,

                images: [item.image],

            },

            unit\_amount: item.price \* 100,

        },

        quantity: item.quantity,

    }))

    try {

        const session = await stripe.checkout.sessions.create({

            payment\_method\_types: ["card"],

            line\_items: transformedItems,

            mode: "payment",

            success\_url: `${headersList.get("origin")}/shoppingClient/success`,

            cancel\_url: `${headersList.get("origin")}/`,

        });

      return NextResponse.json({url: session.url,sessionId: session.id });

    } catch (err) {

        console.log(err)

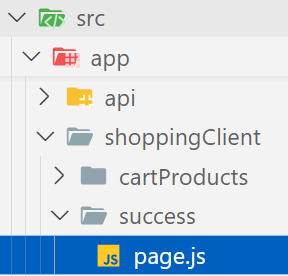
        return NextResponse.json({error: "Error creating checkout session"});

    }

}

### La page Success

Sous **/src/app/shoppingClient/** (pour respecter la route mise dans success\_url: `${req.headers.origin}/shoppingClient/success`), créer le dossier **success** dans lequel on créera le fichier **page.js** ayant ce contenu.



"use client";

import React,{useEffect} from 'react'

import Link from 'next/link'

import { useShoppingCart } from "use-shopping-cart";

const Success = () => {

  const { clearCart } = useShoppingCart();

  useEffect(() => {

    clearCart();

  }, []);

  return (

    <div className='h-screen grid place-items-center'>

      <div className='text-center'>

        <h1 className='text-8xl font-bold'>Thank You</h1>

        <p className='text-center text-2xl'>Order Placed Successfully</p>

        <Link href="/shoppingClient/cartProducts">

          <p className='bg-red-600 text-white py-4 px-12 mt-4 hover:bg-red-800 cursor-pointer'>Continue Shopping</p>

        </Link>

      </div>

    </div>

  )

}

export default Success

Le nombre total d'articles de notre panier doit revenir à 0 après avoir effectué le paiement avec clearCart.

Enfin, modifier cartModal.



"use client";

import { useState } from "react";

import { useShoppingCart } from "use-shopping-cart";

import DeleteIcon from '@mui/icons-material/Delete';

import ClearIcon from '@mui/icons-material/Clear';

import {loadStripe} from '@stripe/stripe-js';

export default function cartModal() {

    const { shouldDisplayCart,cartDetails,cartCount , removeItem , clearCart, totalPrice,} = useShoppingCart();

    const [status, setStatus] = useState("idle");

    async function handleClickStripe(event) {

      event.preventDefault();

      if (cartCount > 0) {

        setStatus("loading");

        try {

          const stripe = await loadStripe(process.env.NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY );

          if (!stripe) throw new Error('Stripe failed to initialize.');

          const checkoutResponse = await fetch('/api/stripe/checkoutSessions', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify({cartDetails}),

        })

          const {sessionId} = await checkoutResponse.json();

          const stripeError = await stripe.redirectToCheckout({sessionId});

          if (stripeError) {

              console.error(stripeError);

          }

         } catch (error) {

          console.error(error);

          setStatus("redirect-error");

        }

      } else {

        setStatus("no-items");

      }

    }

  return (

    <div

      className={`bg-white flex flex-col absolute right-3 md:right-9 top-14 w-80 py-4 px-4 shadow-[0\_5px\_15px\_0\_rgba(0,0,0,.15)] rounded-md transition-opacity duration-500 ${

        shouldDisplayCart ? "opacity-100 z-10" : "opacity-0"

      }`}

    >

    {cartCount && cartCount > 0 ? (

        <>

        <div

            onClick={() => clearCart()}

            >

             <ClearIcon/>

        </div>

          {Object.values(cartDetails).map((item) => (

            <div className="flex items-center gap-4 mb-3" key={item.id}>

            <p>

                          <img

                          src={item.image}

                          alt={item.title}

                          style={{ width: '60px', height: '60px' }}

                         />

                         </p>

            <div>

             {item.title} <span className="text-xs">({item.quantity})</span>

            </div>

            <div className="ml-auto">{item.price} TND</div>

            <button className="hover:bg-emerald-50 transition-colors rounded-full duration-500 p-1"

            onClick={() => removeItem(item.id)}

            >

            <DeleteIcon color="secondary"/>

            </button>

          </div>

          ))}

                <article className="mt-3 flex flex-col">

                    Total : {totalPrice} TND

                  <button

                   className="bg-amber-50 hover:bg-amber-500 hover:text-white transition-colors duration-500 text-amber-500 py-3 px-5 rounded-md w-100"

                   onClick={(event)=>handleClickStripe(event)}

                   >

                     {status !== "loading" ? "Proceed to checkout" : "Loading..."}

                  </button>

               </article>

        </>

      ) : (

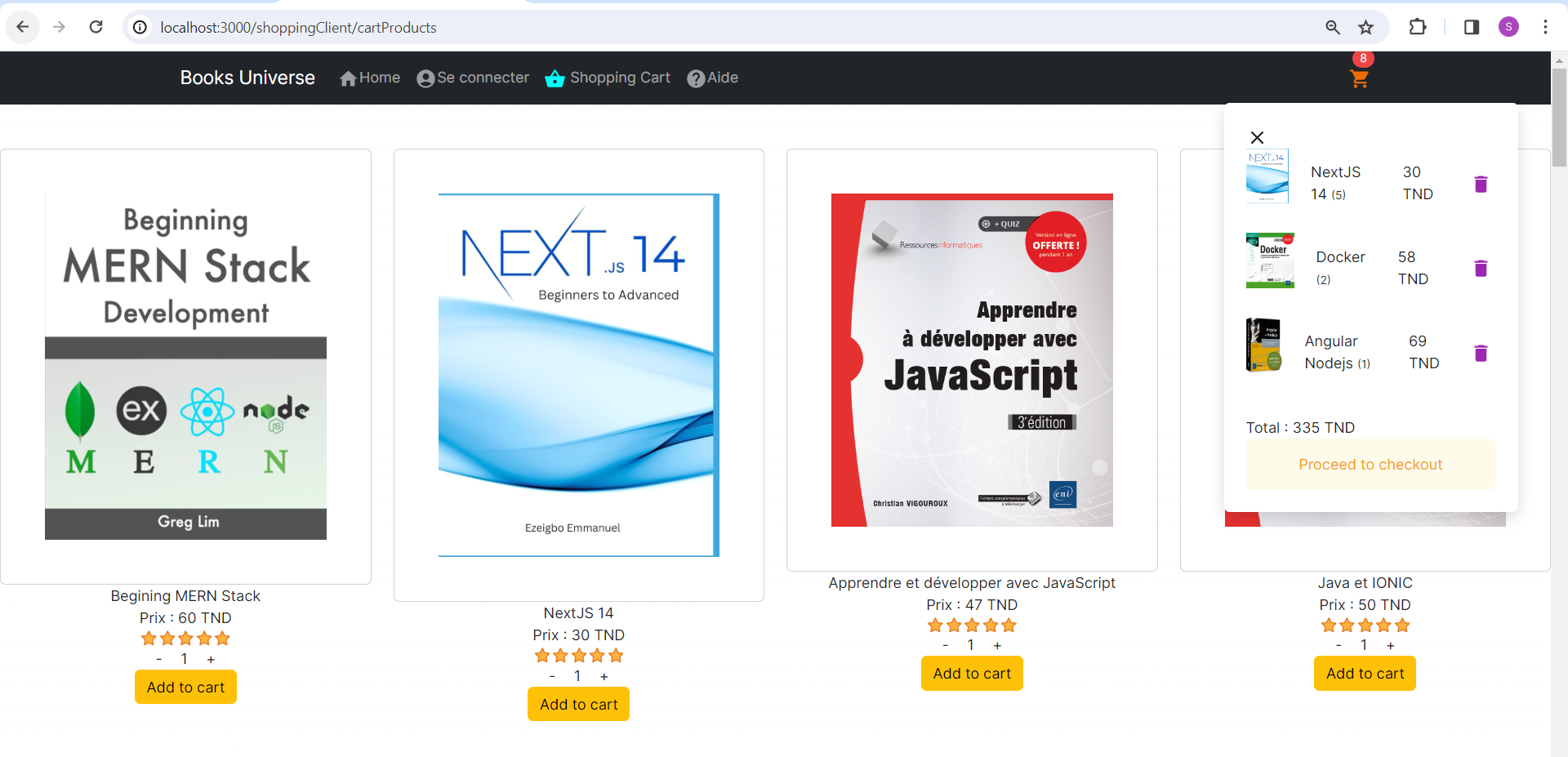
        <div className="p-5">You have no items in your cart</div>

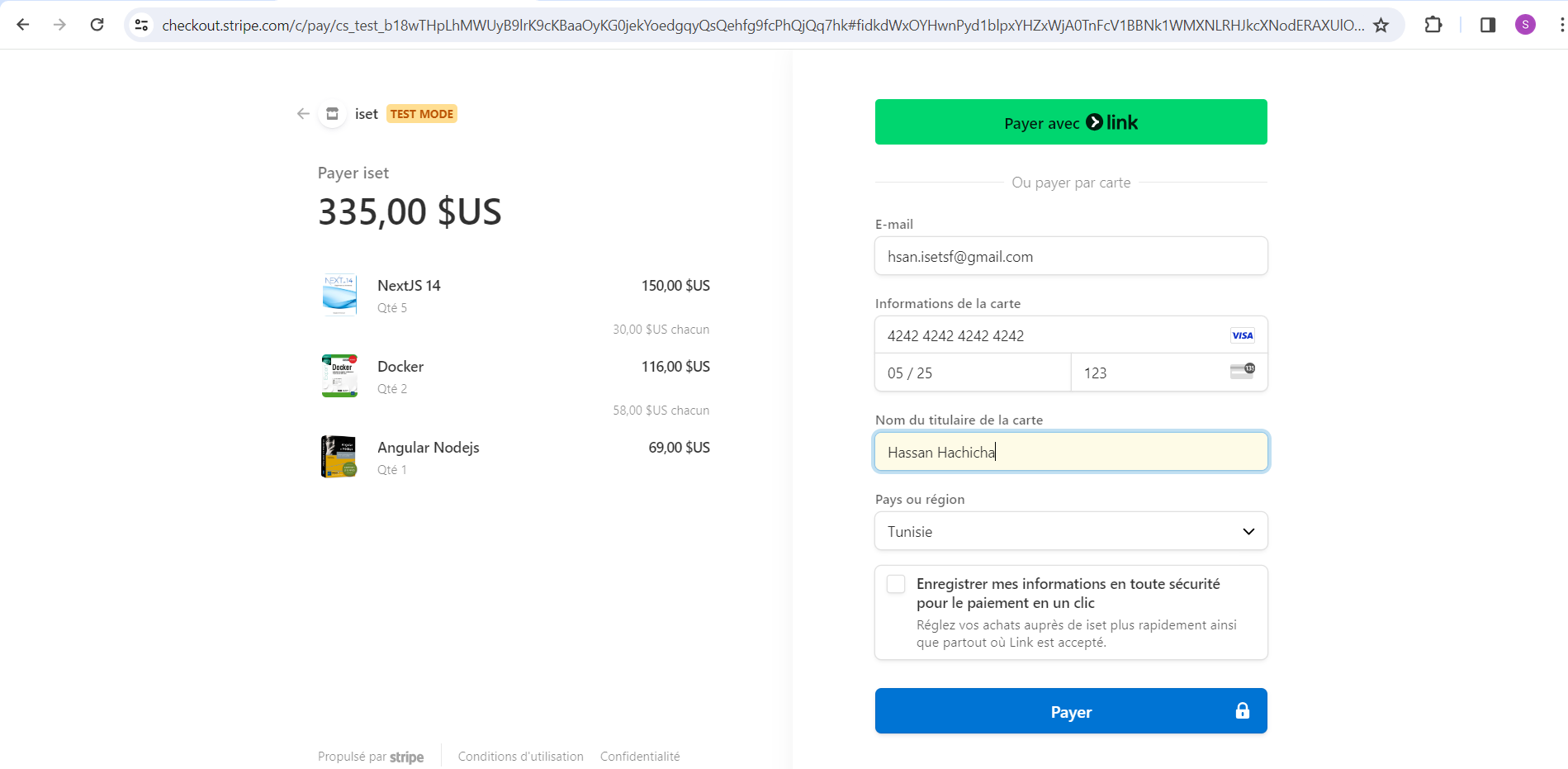
      )}

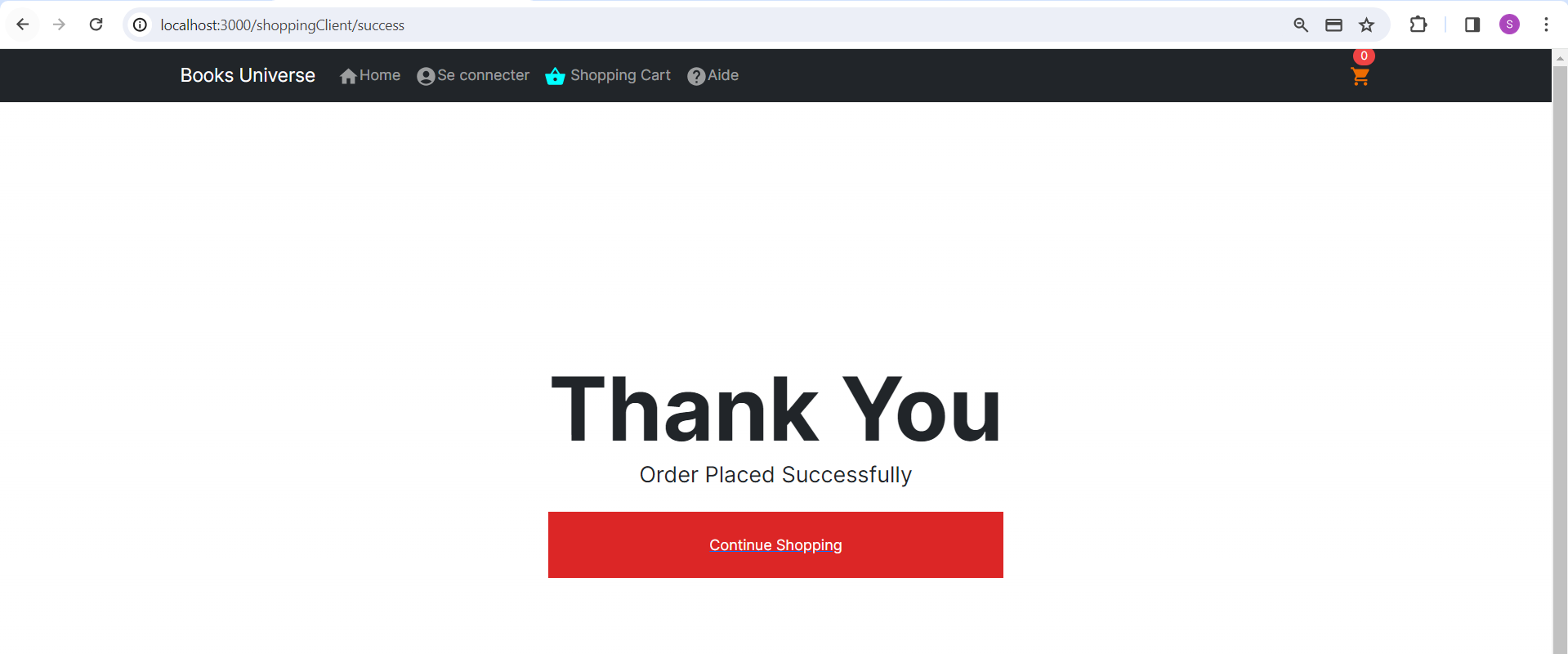
    </div>

  );

}







<https://dashboard.stripe.com/test/payments>



### Récupérer le nom du client

D’abord on sauvegarde la valeur de sessionId récupérée de Stripe dans localStorage



"use client";

import { useState } from "react";

import { useShoppingCart } from "use-shopping-cart";

import DeleteIcon from '@mui/icons-material/Delete';

import ClearIcon from '@mui/icons-material/Clear';

import {loadStripe} from "@stripe/stripe-js";

export default function cartModal() {

    const { shouldDisplayCart,cartDetails,cartCount , removeItem , clearCart, totalPrice,} = useShoppingCart();

    const [status, setStatus] = useState("idle");

    async function handleClickStripe(event) {

      event.preventDefault();

      if (cartCount > 0) {

        setStatus("loading");

        try {

          const stripe = await loadStripe(process.env.NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY );

          if (!stripe) throw new Error('Stripe failed to initialize.');

          const checkoutResponse = await fetch('/api/stripe/checkoutSessions', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify({cartDetails}),

        })

          const {sessionId} = await checkoutResponse.json();

**localStorage.setItem("sessionId",sessionId)**

          const stripeError = await stripe.redirectToCheckout({sessionId});

          if (stripeError) {

              console.error(stripeError);

          }

         } catch (error) {

          console.error(error);

          setStatus("redirect-error");

        }

      } else {

        setStatus("no-items");

      }

    }

  return (

    <div

      className={`bg-white flex flex-col absolute right-3 md:right-9 top-14 w-80 py-4 px-4 shadow-[0\_5px\_15px\_0\_rgba(0,0,0,.15)] rounded-md transition-opacity duration-500 ${

        shouldDisplayCart ? "opacity-100 z-10" : "opacity-0"

      }`}

    >

    {cartCount && cartCount > 0 ? (

        <>

        <div

            onClick={() => clearCart()}

            >

             <ClearIcon/>

        </div>

          {Object.values(cartDetails).map((item) => (

            <div className="flex items-center gap-4 mb-3" key={item.id}>

            <p>

                          <img

                          src={item.image}

                          alt={item.title}

                          style={{ width: '60px', height: '60px' }}

                         />

                         </p>

            <div>

             {item.title} <span className="text-xs">({item.quantity})</span>

            </div>

            <div className="ml-auto">{item.price} TND</div>

            <button className="hover:bg-emerald-50 transition-colors rounded-full duration-500 p-1"

            onClick={() => removeItem(item.id)}

            >

            <DeleteIcon color="secondary"/>

            </button>

          </div>

          ))}

                <article className="mt-3 flex flex-col">

                    Total : {totalPrice} TND

                  <button

                   className="bg-amber-50 hover:bg-amber-500 hover:text-white transition-colors duration-500 text-amber-500 py-3 px-5 rounded-md w-100"

                   onClick={(event)=>handleClickStripe(event)}

                   >

                     {status !== "loading" ? "Proceed to checkout" : "Loading..."}

                  </button>

               </article>

        </>

      ) : (

        <div className="p-5">You have no items in your cart</div>

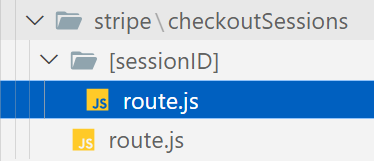
      )}

    </div>

  );

}

Puis on crée sous le dossier /src/app/api/stripe/checkoutSessions le dossier [sessionID] ayant le fichier route.js



import { NextResponse} from "next/server";

import Stripe from 'stripe';

if (!process.env.STRIPE\_SECRET\_KEY) {

    throw new Error('STRIPE\_TEST\_SECRET\_KEY is missing. Please set the environment variable.');

}

const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY);

//Récupérer le nom et l’email saisis dans la transaction stripe

export async function GET(\_, { params }) {

    try {

        const sessionID=params.sessionID

        const session = await stripe.checkout.sessions.retrieve(

          sessionID

        );

        const nomClient = session.customer\_details.name;

        const emailClient = session.customer\_details.email;

        return NextResponse.json({nomClient, emailClient });

    } catch (error) {

        console.log(err)

        return NextResponse.json({error: "Error with session"});

    }

}

Enfin on fait appel à cette route dans le composant success/page.js



"use client";

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

import Link from 'next/link'

import { useShoppingCart } from "use-shopping-cart";

const Success = () => {

  const { clearCart } = useShoppingCart();

  const dataFetchedRef = useRef(false);

  const [nomClient, setNomClient] = useState('');

  const [emailClient, setEmailClient] = useState('');

  const transaction = () => {

    fetch('http://localhost:3000/api/stripe/checkoutSessions/'+localStorage.getItem("sessionId"))

    .then(response => response.json())

    .then(data => { console.log(data)

        setNomClient(data.nomClient);

        setEmailClient(data.emailClient);

        localStorage.removeItem("sessionId")

    })

    .catch(error => console.error('Erreur lors de la récupération des détails de la transaction:', error));

  }

  useEffect(() => {

    clearCart();

    //Pour ne pas exécuter en double

    if (dataFetchedRef.current) return;

    dataFetchedRef.current = true;

//appeler la méthode

    transaction();

  }, []);

  return (

    <div className='h-screen grid place-items-center'>

      <div className='text-center'>

        <h1 className='text-8xl font-bold'>Thank You</h1>

        <p className='text-center text-2xl'>Order Placed Successfully</p>

        <h3> Payment made by </h3>

           <h3>Nom : {nomClient}</h3>

           <h3>Email : {emailClient}</h3>

        <Link href="/shoppingClient/cartProducts">

          <p className='bg-red-600 text-white py-4 px-12 mt-4 hover:bg-red-800 cursor-pointer'>Continue Shopping</p>

        </Link>

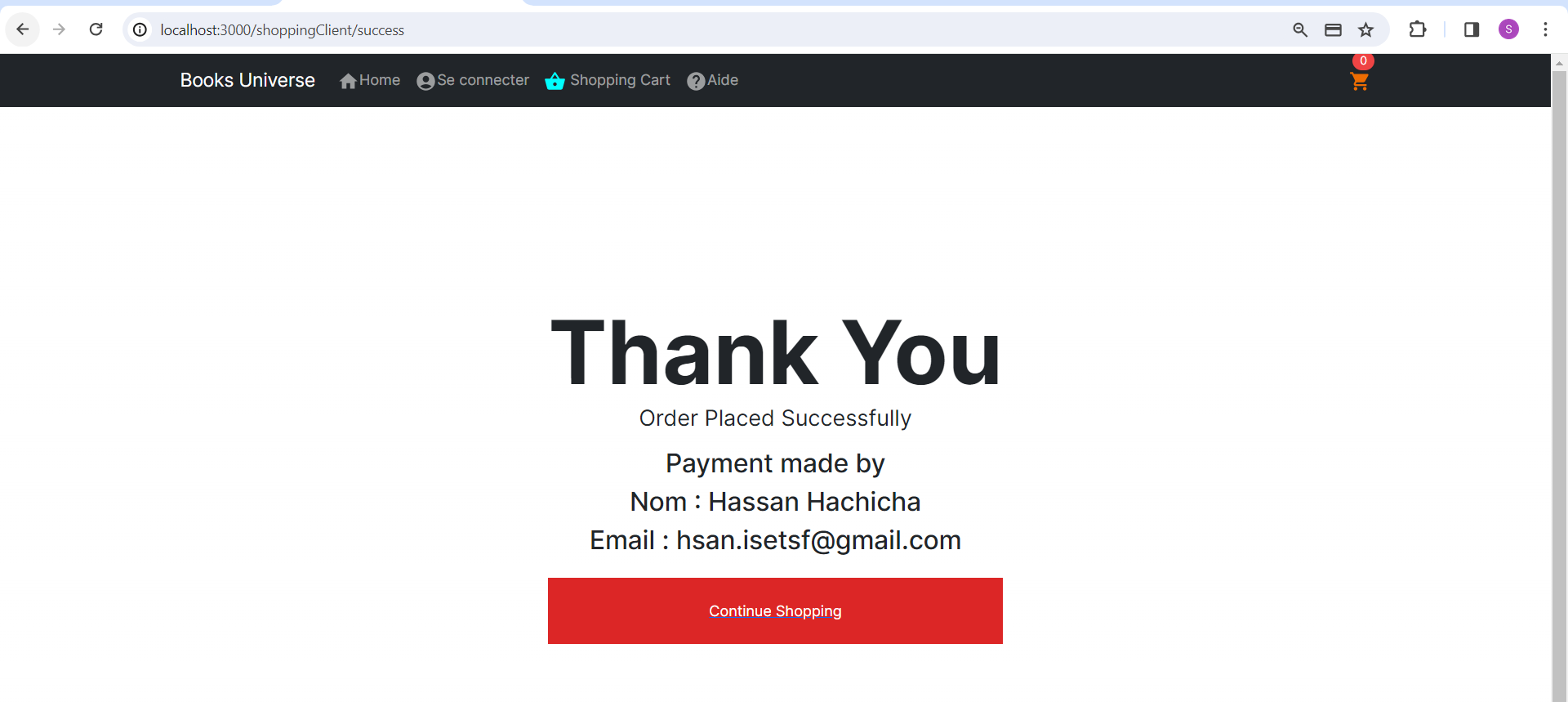
      </div>

    </div>

  )

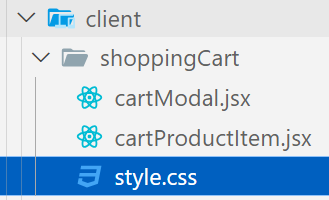
}

export default Success



# Impression du cart

Créer d’abord le style CSS, en créant le fichier style.css sous /src/components/client/shoppingCart



@media print {

    @page { margin: 0; }

    body { margin: 1.6cm; }

    body \* {

      visibility: hidden;

    }

    #partieImprimable, #partieImprimable \* {

      visibility: visible;

    }

    #partieImprimable {

      position: absolute;

      left: 0;

      top: 0;

    }

    #a-exclure {

        display: none;

      }

  }

Puis modifier cartModal.jsx en définissant les zones imprimables et les zones non imprimables.



"use client";

import { useState } from "react";

import { useShoppingCart } from "use-shopping-cart";

import DeleteIcon from '@mui/icons-material/Delete';

import ClearIcon from '@mui/icons-material/Clear';

import {loadStripe} from "@stripe/stripe-js";

**import './style.css';**

export default function cartModal() {

    const { shouldDisplayCart,cartDetails,cartCount , removeItem , clearCart, totalPrice,} = useShoppingCart();

    const [status, setStatus] = useState("idle");

    async function handleClickStripe(event) {

      event.preventDefault();

      if (cartCount > 0) {

        setStatus("loading");

        try {

          const stripe = await loadStripe(process.env.NEXT\_PUBLIC\_STRIPE\_PUBLISHABLE\_KEY );

          if (!stripe) throw new Error('Stripe failed to initialize.');

          const checkoutResponse = await fetch('/api/stripe/checkoutSessions', {

            method: 'POST',

            headers: {

                'Content-Type': 'application/json',

            },

            body: JSON.stringify({cartDetails}),

        })

          const {sessionId} = await checkoutResponse.json();

          localStorage.setItem("sessionId",sessionId)

          const stripeError = await stripe.redirectToCheckout({sessionId});

          if (stripeError) {

              console.error(stripeError);

          }

         } catch (error) {

          console.error(error);

          setStatus("redirect-error");

        }

      } else {

        setStatus("no-items");

      }

    }

  return (

    <div

      className={`bg-white flex flex-col absolute right-3 md:right-9 top-14 w-80 py-4 px-4 shadow-[0\_5px\_15px\_0\_rgba(0,0,0,.15)] rounded-md transition-opacity duration-500 ${

        shouldDisplayCart ? "opacity-100 z-10" : "opacity-0"

      }`}

    >

    {cartCount && cartCount > 0 ? (

        <>

        <div

            onClick={() => clearCart()}

            >

             <ClearIcon/>

        </div>

**<div id="partieImprimable">**

          {Object.values(cartDetails).map((item) => (

            <div className="flex items-center gap-4 mb-3" key={item.id}>

            <p>

                          <img

                          src={item.image}

                          alt={item.title}

                          style={{ width: '60px', height: '60px' }}

                         />

                         </p>

            <div>

             {item.title} <span className="text-xs">({item.quantity})</span>

            </div>

            <div className="ml-auto">{item.price} TND</div>

**<div id="a-exclure">**

            <button className="hover:bg-emerald-50 transition-colors rounded-full duration-500 p-1"

            onClick={() => removeItem(item.id)}

            >

            <DeleteIcon color="secondary"/>

            </button>

         </div>

          </div>

          ))}

                <article className="mt-3 flex flex-col">

                    Total : {totalPrice} TND

        <div id="a-exclure">

         {/\* zone d'impression \*/}

         <button

                    className="bg-emerald-50 hover:bg-emerald-500 hover:text-white transition-colors duration-500 text-emerald-500 py-3 px-5 rounded-md w-100"

                    onClick={()=> window.print()}

                   >

                    Imprimer

                  </button>

        {/\* zone validationn de payement \*/}

                  <button

                   className="bg-amber-50 hover:bg-amber-500 hover:text-white transition-colors duration-500 text-amber-500 py-3 px-5 rounded-md w-100"

                   onClick={(event)=>handleClickStripe(event)}

                   >

                     {status !== "loading" ? "Proceed to checkout" : "Loading..."}

                  </button>

**</div>**

               </article>

**</div>**

        </>

      ) : (

        <div className="p-5">You have no items in your cart</div>

      )}

    </div>

  );

}



