Stripe set up

Start with command line

* npx create-next-app stripe\_payments

Under app folder create two more folders app/api/prices/route.js

Una vez finalizado el proceso de instalación puedes dar el siguiente comando:

* npm run dev

Posteriormente entrar en localhost:3000 y veras el framwork de next trabajando

Después en el navegador deberás crear una cuenta en stripe y obtener las llaves publica y secreta respectivamente [pk y sk] estas en modo developers y en test mode:

* posteriormente en la carpeta src/app/.env.local crear ese archivo donde se introducira la llave secreta [sk] que se obtuvo en el sitio web de stripe, dentro del archivo .env.local se creara la siguiente variable:

|  |
| --- |
| STRIPE\_SECRET\_KEY=sk\_test\_51PTcC8068sYNCeUIMqURVZjJz71UGz3BY2ZiQc7HVsE6EFFPfaUY0OzuXmYhIZ5IgEcYycqxBP7R60WaVdCGNlpW00IfHYhl4j |

* npm i stripe

EN EL ARCHIVO route.js SE INTRODUCIRA LO SIGUIENTE:

Esto creara una ruta de backend que se comunicara con stripe

|  |
| --- |
| import { NextResponse } from 'next/server'  import {Stripe} from 'stripe' //se importa una clase llamada Stripe la cual se debe instanciar e introducirle la llave secreta generada en stripe test mode  export async function GET() {  const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY) //Esto genera un objeto por lo cual se denomina como una constante o instancia  const prices = await stripe.prices.list() //aqui se genera la consulta de precios se usa await ya que sera asincona  console.log(prices) //aqui se pueden ver los precios en la consola no en la pagina    return NextResponse.json({  message: 'Hello from /api/prices',  })  } |

A computer screen shot of a computer screen

Description automatically generated

Con esto logramos ver el listado de productos que se han generado dentro de la pagina api de stripe

Ahora si se quiere ver en el navegador se puede hacer así:

|  |
| --- |
| import { NextResponse } from 'next/server'  import {Stripe} from 'stripe' //se importa una clase llamada Stripe la cual se debe instanciar e introducirle la llave secreta generada en stripe test mode  export async function GET() {  const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY) //Esto genera un objeto por lo cual se denomina como una constante o instancia  const prices = await stripe.prices.list() //aqui se genera la consulta de precios se usa await ya que sera asincona  console.log(prices) //aqui se pueden ver los precios en la consola no en la pagina    return NextResponse.json(prices.data); //aqui se desplegara la lista de informacion de productos creados en stripe  } |

You can see this information of the object created named list

A screen shot of a computer

Description automatically generated

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Posteriormente se deberá generar una subcarpeta dentro de la carpeta app llamada pricing app/pricing/page.jsx

A screenshot of a computer program

Description automatically generated

Dentro del achivo page.jsx se deberá cargar el siguiente código:

|  |
| --- |
| import {Stripe} from 'stripe'  /\*  Para este ejercicio es lo mismo que en la api/prices/route.js solamente que en este caso se hara dentro de un componente  \*/  async function loadPrices() {      const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY);      const prices = await stripe.prices.list();      return prices.data;  }  //React server component  async function PricingPage() {      const prices = await loadPrices(); //Aqui se ejecuta la orden para ver los precios      console.log(prices); //se debe desplegar los precios en consola de buscador      return (          <div>              <h1>Pricing</h1>              {                  prices.map(price => (                      <div key={price.id}>                          <h3>{price.nickname}</h3>                          <h2>{price.unit\_amount / 100}</h2>                          <h2>{price.unit\_amount\_decimal}</h2>                          <button>                              Buy                          </button>                      </div>                  ))              }          </div>          );  }    export default PricingPage; |

El resultado será el siguiente:

A screenshot of a phone

Description automatically generated

En la pagina de stripe encontraras la descripción del producto dishes en donde veras el precio del producto estará en centenas

A screenshot of a computer

Description automatically generated

Por lo que para sacar el precio unitario deberás dividir entre 100 para obtener el precio real

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Para poder darle formato a la interfaz de usuario se hará la siguiente modificación:

|  |
| --- |
| return (          <div className="flex justify-center items-center h-screen">              <div>                  <header>                      <h1 className="text-center my-5">Pricing</h1>                  </header>                  <div className="flex gap-x-2">                      {prices.map(price => (                      <div key={price.id} className="bg-slate-300 mb-2 p-7">                          <h3>{price.nickname}</h3>                          <h2 className="text-3xl font-bold">{price.unit\_amount / 100}$</h2>                          <h2 className="text-3xl font-bold">{price.unit\_amount\_decimal}.00</h2>                          <button className="bg-sky-500 text-white px-4 py-2 rounded">                              Buy                          </button>                      </div>                  ))}                  </div>              </div>          </div>      );  }    export default PricingPage; |

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Si necesitas poner el listado de menor a mayor puedes usar la siguiente línea de código:

|  |
| --- |
| const sortedPrices = prices.data.sort((a, b) => a.unit\_amount - b.unit\_amount) //Esta linea de codig acomoda los precios de menor a mayor  return sortedPrices; //Aqui retorna el valor de los items acomodados por precio |
|  |

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Para poder desplegar el checkout se debera generar una nueva carpeta dentro de api/checkout/route.js

A screen shot of a computer

Description automatically generated

|  |
| --- |
| import { NextResponse } from 'next/server'  export async function POST() {      return NextResponse.json({ message: "checkout" })  }  //para probar este mensaje se podra realizar con thunderclient, postman u otro plugin pero no desde el navegador  //o incluso hacer una peticion post en el boton co un onClick |

Para poder generar la respuesta del botón se modificara la lógica del botón en app/pricing/page.jsx y se creara un componente

Para poder hacer esto se necesitara generar una nueva carpeta en app/components/ButtonCheckout.jsx y se introducira el siguiente código:

|  |
| --- |
| "use client" //Es un componente cliente por tal motivo se debe colocar "use client"  function ButtonCheckout() {      return (      <button          className="bg-sky-500 text-white px-4 py-2 rounded"          onClick={() => {              console.log("click");//Aqui mostrara un click cada vez que se haga un click en el boton buy en la consola del navegador          }}      >          Buy      </button>      );  }  export default ButtonCheckout; |

Y entonces el botón se modificara como sigue:

|  |  |
| --- | --- |
| <button className="bg-sky-500 text-white px-4 py-2 rounded">                              Buy                          </button>  ANTES | import ButtonCheckout from '../components/ButtonCheckout'   <ButtonCheckout />  DESPUES |

Esto mostrara un clcik en l consola del navegador cada vez que se presione el botón buy en la pagina de precios

A screenshot of a computer

Description automatically generated

//////////////////////////////////////////////////////////////////////////////////////////////////////////

Como queremos que se muestre el id del precio en el componente ButtonCheckout en la pagina app/pricing/page.jsx se modificara así:

|  |  |
| --- | --- |
| <ButtonCheckout priceId={price.id}/>  Se le añade la propiedad deseada | Se deberá añadir esa propiedad del componente en el código del ButtonCheckout así:  function ButtonCheckout({ priceId }) {  /  /  /  console.log(priceId); |

Aquí el resultado:

A screenshot of a computer

Description automatically generated

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Esta practica nos sirve para poder realizar una petición, para lo cual es necesario el id precio del producto para poder realizar la compra:

Para esto haremos uso de fetch como sigue:

|  |
| --- |
| "use client" //Es un componente cliente por tal motivo se debe colocar "use client"  function ButtonCheckout({ priceId }) { //El priceId se obtendra del componente insertado como propiedad "priceId={price.id}" en el <ButtonCheckout />      return (      <button          className="bg-sky-500 text-white px-4 py-2 rounded"          onClick={ async () => { //se cambia a funcion asincrona              console.log("click");//Aqui mostrara un click cada vez que se haga un click en el boton buy              console.log(priceId);//Aqui se vera en la consola el valor identificador del precio al hacer click en el boton buy              const res = await fetch('/api/checkout', { //Aqui se solicita la informacion del codigo dentro de api/checkout/route.js                  method: 'POST', //Este metodo es para postear el resultado pero por defecto es el metodo get                  body: JSON.stringify({ priceId }), //Aqui se genera el objeto del valor del precio id                  headers: { 'Content-Type': 'application/json'}              })              const data = await res.json() //Aqui se convierte la respuesta a un objeto de JS              console.log(data) //Aqui se despliega la informacion en la consola          }}      >          Buy      </button>      );  }  export default ButtonCheckout; |

Haciendo esto tu podras obtener el mensaje de checkout de la función POST() del archivo api/checkout/route.js

Cada vez que presionas el botón buy

A screenshot of a computer

Description automatically generated

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Para poder extraer la información que se necesita en el archivo api/checkout/route.js se deberá modificar el código como sigue:

Para poder generar una acción de compra

|  |
| --- |
| import { NextResponse } from 'next/server'  import { Stripe } from 'stripe'//Se hara uso de la creacion de sesiones en stripe es decir del formulario de compra  export async function POST(request) { //Para poder extraer la informacion se añade un request      const data = await request.json() //Aqui se genera un objeto de la informacion, data puede ser reemplazado por                                        //const {priceId} = await request.json() para anotar en line\_items: [{ price: priceId}]      const stripe = new Stripe(process.env.STRIPE\_SECRET\_KEY)//Aqui se hace uso nuevamente de la llave screta de stripe     const session = await stripe.checkout.sessions.create({ //Aqui se crea el formulario de compra, SE DENOMINA SESSION ES DECIR UN OBJETO PARA PODER VER EL RESULTADO EN CONSOLA          mode: 'payment', //modo: pago ESTE VALOR SERA AFECTADO POR LOS VALORES INTRODUCIDOS EN STRIPE P.EJ PARA PAGO RECURRENTE ES [mode: 'suscription]          payment\_method\_types: ['card'], //tipo de pago          line\_items: [  //aqui se deben introducir todos los productos en venta              {                  price: data.priceId, //Aqui se puede utilizar solo price: priceId pero se debera reemplazar const data por {priceId}                  quantity: 1  //Aqui se define la cantidad de productos de compra              }          ],          success\_url: 'http://localhost:3000/success' //Aqui al finalizar la compra se dirigira a una pagina de compra exitosa          cancel\_url: 'http://localhost:3000/pricing' //Aqui si se cancela la compra regresara a la pagina de pricing      })      console.log(session)      return NextResponse.json({ message: "checkout" })  }  //para probar este mensaje se podra realizar con thunderclient, postman u otro plugin pero no desde el navegador  //o incluso hacer una peticion post en el boton co un onClick  //la sesion es el formulario de compra |

Después de guardar todos los cambios en la pagina de pricing seleccionar algún producto y darle clcik en buy

Y se generara un mensaje en la consla del navegador

A screenshot of a computer

Description automatically generated

De la misma manera se desplegara información en la terminal del backend con información importante

A screenshot of a computer

Description automatically generated

En esta información se generara una url con información detallada del producto de compra, dar ctrl + click en la url para ver el link se vera algo similar:

A screenshot of a computer

Description automatically generated

Con esto generamos informacion de compra pero para poder usarla se hará lo siguiente:

Se enviara la url al frontend de la siguiente manera:

En el archivo en api/checkout/route.js se modificara el return:

|  |  |
| --- | --- |
| return NextResponse.json({ message: "checkout" })  ANTES | return NextResponse.json({ url: session.url, });  DESPUES |

Con esto en api/pricing/page.jsx cada vez que se de un clcik en el botón se esperara una respuesta un data de ButtonCheckout.jsx

Y entonces obtendremos una url esta vez en la consola:

A screenshot of a computer

Description automatically generated

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

Para poder leer esa url y redirigir a la pagina de pago se deberá introducir la siguiente línea de código en ButtonCheckout.jsx

|  |
| --- |
| window.location.href = data.url //Aqui el navegador leera la url y te dirigira hacia ella          }} |

Y así al presionar el botón de buy automáticamente generara la url y te redirigirá hacia ella

A white background with black text and a button

Description automatically generated A screenshot of a computer

Description automatically generated

En la siguiente formulario se debera llenar con informacion de prueba ya que desde el inicio de la creacion de cuenta de stripe

Estamos en tes mode o modo de prueba

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

Al procesar la infomacion de pago te redirigira a la pagina <http://localhost:3000/success>

Si le das clcik al boton back

A screenshot of a test

Description automatically generated

Te dirigira nuevamente a la pagina de <http://localhost:3000/pricing>

Finalemte en la API de Stripe podras ver la informacion generada de pago:

