Estas indicaciones se encuentran en la pagina de Wonpi : Usando tan solo unas pocas líneas de HTML, integra nuestro **Widget** de pagos, para que tus clientes puedan pagar sin tener que salir de tu sitio web.

Widget Web

Permite que tus clientes completen el pago sin salir de tu sitio web, dentro de nuestro widget de pagos, simplemente incluyendo una etiqueta <script> (debes poder incluir Javascript en tu sitio). Puedes ver un ejemplo a continuación:

Paga con Wompi

Usa HTTPS

Te recomendamos fuertemente usar **HTTPS** en tu sitio web, ya que no sólo tus clientes se sentirán más seguros a la hora de navegar y hacer pagos, sino que también [evitarás que tu sitio sea marcado como "No seguro"](https://ar.godaddy.com/blog/google-chrome-68-introduce-https-como-un-deber-ser-de-la-nueva-seguridad/), un cambio que Google Chrome alertó que haría desde el **24 de julio de 2018**. Algo importante a tener en cuenta ya que sólo este navegador representa más del 60% del tráfico en la web

**Paso 1: Alista tu llave pública de comercio**

Recuerda que hay llaves de sandbox y producción!

Las llaves de sandbox (entorno de pruebas) tienen el prefijo pub\_test\_, mientras que las de producción tienen el prefijo pub\_prod\_. Para transacciones con dinero real usa la llave de **Producción**; para hacer transacciones de prueba, mientras integras y desarrollas, usa la llave de **Sandbox**.

**Paso 2: Genera una referencia única de pago**[**​**](https://docs.wompi.co/docs/colombia/widget-checkout-web/#paso-2-genera-una-referencia-%C3%BAnica-de-pago)

Para cada compra que tus clientes hagan en tu website, deberás generar una **referencia única de pago** (similar a como funciona un número de factura en el mundo real). De esta manera podrás hacer seguimiento a cada transacción que se complete en Wompi y evitarás duplicar transacciones por accidente. Esto quiere decir que una vez se complete una transacción, no podrás reutilizar una referencia que ya exista en tu base de datos.

Recomendamos usar **referencias numéricas o alfanuméricas**, que **pueden o no** incluir guiones (-) o guiones bajos (\_). Los siguientes son ejemplos de referencias válidas:

3b4393bafed398ba2

54937

10384200283

58e281-177ab976cbf9-d162d2

38932\_3293298

0f760951ed\_a0086f

**Paso 3: Genera una firma de integridad**[**​**](https://docs.wompi.co/docs/colombia/widget-checkout-web/#paso-3-genera-una-firma-de-integridad)

Para validar la integridad de la información de la transacción y evitar alteraciones, Wompi utiliza un hash criptográfico asimétrico.

Para generar este hash criptográfico, busca el secreto de **integridad** accediendo al dashboard de comercios en "Desarrolladores >

Es importante que aclarar que **este Secreto para Integridad es diferente de tu Llave Privada y Llave Pública**.

Después de tener listo tu secreto de integración, deberas generar un hash SHA256 con la siguiente información (el orden importa):

Referencia de la transacción: sk8-438k4-xmxm392-sn2m

Monto de la transacción en centavos: 2490000

Moneda de la transacción: COP

Secreto de integridad: prod\_integrity\_Z5mMke9x0k8gpErbDqwrJXMqsI6SFli6

Estos valores se concatenan:

"<Referencia><Monto><Moneda><SecretoIntegridad>"

Y lo encriptamos con SHA256:

Te recomendamos fuertemente crear este hash criptografico en tu servidor y nunca en tu frontend, pues expondrías el secreto de integración a un potencial atacante

En Ruby:

var cadenaConcatenada =

"sk8-438k4-xmxm392-sn2m2490000COPprod\_integrity\_Z5mMke9x0k8gpErbDqwrJXMqsI6SFli6";

//Ejemplo

const encondedText = new TextEncoder().encode(cadenaConcatenada);

const hashBuffer = await crypto.subtle.digest("SHA-256", encondedText);

const hashArray = Array.from(new Uint8Array(hashBuffer));

const hashHex = hashArray.map((b) => b.toString(16).padStart(2, "0")).join(""); // "37c8407747e595535433ef8f6a811d853cd943046624a0ec04662b17bbf33bf5"

**Paso 4: URL de redirección**

Al finalizar una transacción, opcionalmente, Wompi puede redirigir al usuario a una URL (que debe pertenecer a tu sitio web), en la cual podrás consultar el resultado final (status) de la transacción. Esto lo puedes hacer usando el id de la transacción, el cual estará disponible como un parámetro de la URL.

Así por ejemplo, si tu URL es:

https://mitienda.com.co/pagos/respuesta

La URL a la que Wompi redirigirá es similar a la siguiente:

<https://mitienda.com.co/pagos/respuesta?id=01-1531231271-19365>

Así, puedes usar el parámetro id disponible en la URL para verificar la transacción usando nuestro API apuntando a la URL https://production.wompi.co/v1/transactions/<ID\_TRANSACCION>. Por ejemplo https://production.wompi.co/v1/transactions/01-1531231271-19365 (este ID puede NO ser real y sirve solo para dar un ejemplo)

**Paso 5: Parámetros de la transacción**

Para cada transacción puedes proveer parámetros como el monto a cobrar, la moneda en la que quieres cobrar, etc. Algunos de estos parámetros son obligatorios y otros son opcionales.

Parámetros obligatorios

Los siguientes son los parámetros obligatorios que debes tener en cuenta para crear una transacción:

public-key (Llave pública de comercio): Llave pública de comercio.

currency (Moneda): Moneda en la que vas a hacer el cobro. La única moneda disponible actualmente es COP (pesos colombianos).

amount-in-cents (Monto en centavos): Monto a cobrar, en centavos. Por ejemplo si deseas cobrar $95.000 COP, deberás ingresar: 9500000

reference (Referencia única de pago): Referencia única de pago.

signature:integrity (Firma de integridad): Es un hash criptográfico que utilizamos para validar la integridad de la información de la transacción y evitar alteraciones.

Parámetros opcionales

Los siguientes son parámetros opcionales que, aunque no sean necesarios, proveen una mejor experiencia de integración:

redirect-url (URL de redirección): Es la URL a la que el usuario será redirigido luego completar el proceso de pago, conteniendo el id de la transacción respectiva.

shipping-address (Información de envío): Es la información de dirección de envío del cliente, donde recibirá los productos y/o servicios, si aplica. Los datos que se pueden enviar son los siguientes:

address-line-1: (Obligatorio) para la dirección del lugar de la entrega

address-line-2: para referencias extras

country: (Obligatorio) para el código ISO 3166-1 Alpha-2 (2 letras mayúsculas) del país donde se encuentra la dirección (ej: CO)

city: (Obligatorio) para especificar la ciudad donde se encuentra la dirección

phone-number: (Obligatorio) para el número de teléfono de quien recibe

region: (Obligatorio) para la región donde se encuentra la dirección

name: para el nombre de quien recibe

postal-code: para el código postal

collect-shipping (Activar formulario de envío): Este parámetro activa la vista de información de envío, y si se diligenciaron los campos anteriores, aparecerán prellenados en la vista.

customer-data (Información del pagador): Es la información del pagador, la cual se prellenara en la vista de "Ingresa tus datos". Los datos permitidos son:

email: para el correo electrónico del pagador

full-name: para el nombre completo del pagador

phone-number: para el número de teléfono del pagador, debe ir acompañado del campo phone-number-prefix

phone-number-prefix: para el prefijo o código del país del teléfono del pagador (ej: +57), debe ir acompañado del campo phone-number

legal-id: para el número de documento de identidad del pagador, este parámetro activa el campo de documento de identidad del pagador en la vista de "Ingresa tus datos" y debe ir acompañado del campo legal-id-type

legal-id-type: para el tipo de documento del pagador, este parámetro activa el campo de documento de identidad del pagador en la vista de "Ingresa tus datos" y debe ir acompañado del campo legal-id. Los tipos de documento permitidos son:

CC: Cédula de Ciudadanía

CE: Cédula de Extranjería

NIT: Número de Identificación Tributaria

PP: Pasaporte

TI: Tarjeta de Identidad

DNI: Documento Nacional de Identidad

RG: Carteira de Identidade / Registro Geral

OTHER: Otro

collect-customer-legal-id: Activa el campo de documento de identidad del pagador, usando true como valor. Este parámetro activa el campo de documento de identidad del pagador en la vista de "Ingresa tus datos". Si se diligenciaron los campos de legal\_id y legal\_id\_type de customer\_data, se prellenara con dicha información

tax-in-cents (Detalle de impuestos en pago): Es la información de impuestos en la que puedes detallar el tipo de impuesto y el monto del impuesto dentro del precio total de la transacción en centavos. Más adelante se explica la manera de usarlo en las distintas formas de integración. Los tipos de impuestos permitidos son los siguientes:

VAT: para el IVA (Impuesto de Valor Agregado)

CONSUMPTION: para el Impuesto al Consumo

expiration-time: Fecha y hora en formato ISO8601 (UTC+0000), activa un contador regresivo indicando el tiempo restante para la expiración del inicio del pago

Los impuestos no se sumarán al monto de la transacción

Es importante resaltar que los impuestos enviados en el objeto taxes no se sumarán al total de la transacción.

Por ejemplo, en una transacción cuyo total (amount\_in\_cents) es de COP$119,000 y cuyo IVA es de COP$19,000, este último monto ya hace parte del total, implicando entonces que: la base sin impuestos ($100,000) + el IVA ($19,000) = el total ($119,000).

En otras palabras, Wompi no sumará $19,000 a los $119,000, sino que simplemente compartirá esta información tributaria con el respectivo procesador del pago.

**Paso 6: Escoge un método de integración**[**​**](https://docs.wompi.co/docs/colombia/widget-checkout-web/#paso-6-escoge-un-m%C3%A9todo-de-integraci%C3%B3n)

Escoge una de las dos opciones de integración:

[**Widget**](https://docs.wompi.co/docs/colombia/widget-checkout-web/#widget): Permite que tus clientes completen el pago dentro de tu sitio web en nuestro Widget.

Widget

Este es el método más simple para que tus clientes completen un pago sin salir de tu sitio web. Con tan solo unas líneas de HTML, un botón de pagos se mostrará automáticamente.

Al hacer clic en el botón, el cliente continúa el proceso de pago dentro de nuestro widget, sin salir de tu sitio web (si quieres un botón a la medida, lee la sección de Botón personalizado).

A continuación verás un ejemplo del botón generado con el mismo código que encuentras debajo:

Permite que tus clientes completen el pago sin salir de tu sitio web, dentro de nuestro widget de pagos, simplemente incluyendo una etiqueta <script> (debes poder incluir Javascript en tu sitio). Puedes ver un ejemplo a continuación:

Paga con Wompi

<form>

<script

src="https://checkout.wompi.co/widget.js"

data-render="button"

data-public-key="pub\_test\_X0zDA9xoKdePzhd8a0x9HAez7HgGO2fH"

data-currency="COP"

data-amount-in-cents="4950000"

data-reference="4XMPGKWWPKWQ"

data-signature:integrity="37c8407747e595535433ef8f6a811d853cd943046624a0ec04662b17bbf33bf5"

></script>

</form>

El código que ves arriba tiene los parámetros mínimos necesarios para generar un botón de pago que permite a tu cliente pagar en nuestro widget. Así, sólo necesitas incluir una etiqueta <script> con los parámetros de la transacción, dentro de una etiqueta <form> en tu código, para generar el botón.

Sólo necesitas tener en cuenta tres cosas para generar el botón:

Incluye el parámetro data-render="button", que indica que quieres generar un botón automáticamente.

Para los parámetros listados arriba debes usar el prefijo data- en cada uno, para especificarlo (i.e. data-reference, data-currency, data-amount-in-cents, etc.) y usar guiones (-) en vez de guiones bajos (\_) para los nombres de cada parámetro.

Para los parámetros de shipping-address, customer-data y tax-in-cents debes usar dos puntos (:) para especificar el tipo de información que se desea.

Un ejemplo con todos los parámetros se ve como el siguiente:

<form>

<script

src="https://checkout.wompi.co/widget.js"

data-render="button"

data-public-key="pub\_test\_X0zDA9xoKdePzhd8a0x9HAez7HgGO2fH"

data-currency="COP"

data-amount-in-cents="7890000"

data-reference="37DNKF84S92N1S"

data-signature:integrity="37c8407747e595535433ef8f6a811d853cd943046624a0ec04662b17bbf33bf5"

data-redirect-url="https://transaction-redirect.wompi.co/check"

data-expiration-time="2023-06-09T20:28:50.000Z"

data-tax-in-cents:consumption="590000"

data-tax-in-cents:vat="1290000"

data-customer-data:email="lola@perez.com"

data-customer-data:full-name="Lola Perez"

data-customer-data:phone-number="3019777777"

data-customer-data:phone-number-prefix="+57"

data-customer-data:legal-id="123456789"

data-customer-data:legal-id-type="CC"

data-shipping-address:address-line-1="Carrera 123 # 4-5"

data-shipping-address:address-line-2="apto 123"

data-shipping-address:country="CO"

data-shipping-address:city="Bogota"

data-shipping-address:phone-number="3019988888"

data-shipping-address:region="Cundinamarca"

data-shipping-address:name="Pedro Perez"

></script>

</form>

Paso 7: Escucha el evento de una transacción

Usa siempre los eventos para finalizar tu integración

Al haber integrado el Widget o Web Checkout en tu website, sólo resta que escuches un Evento en tu servidor, para enterarte cuando una transacción finalizó. No utilices la redirección como método de validación de tus transacciones, sino únicamente con propósitos informativos para tus usuarios.

Una vez un usuario haya finalizado una transacción, Wompi te informará a través de un Evento que la misma llegó a un estado final. Para ello deberás proveer una URL de Eventos (a webhook), donde Wompi te enviará un objeto JSON con la información completa de la transacción. Haz clic acá y visita la guía de Eventos para conocer en detalle todo sobre esta funcionalidad.

Botón personalizado (opcional)

Si quieres ofrecer una integración personalizada a tus clientes, como por ejemplo un botón con estilos propios, o abrir el widget dada cierta acción de un usuario, simplemente debes seguir los siguientes pasos:

Paso 1: Incluye el script del widget

Agrega esta etiqueta preferiblemente dentro del <head> de tu HTML:

<script

type="text/javascript"

src="https://checkout.wompi.co/widget.js"

></script>

Paso 2: Configura los datos de la transacción

Configura una instancia del checkout con el objeto de configuración, cuyos campos se muestran a continuación con valores de ejemplo. Todos son obligatorios, excepto redirectUrl.

var checkout = new WidgetCheckout({

currency: 'COP',

amountInCents: 2490000,

reference: 'AD002901221',

publicKey: 'pub\_fENJ3hdTJxdzs3hd35PxDBSMB4f85VrgiY3b6s1',

signature: {integrity : '3a4bd1f3e3edb5e88284c8e1e9a191fdf091ef0dfca9f057cb8f408667f054d0'}

redirectUrl: 'https://transaction-redirect.wompi.co/check', // Opcional

expirationTime: '2023-06-09T20:28:50.000Z', // Opcional

taxInCents: { // Opcional

vat: 1900,

consumption: 800

}

customerData: { // Opcional

email:'lola@gmail.com',

fullName: 'Lola Flores',

phoneNumber: '3040777777',

phoneNumberPrefix: '+57',

legalId: '123456789',

legalIdType: 'CC'

}

shippingAddress: { // Opcional

addressLine1: "Calle 123 # 4-5",

city: "Bogota",

phoneNumber: '3019444444',

region: "Cundinamarca",

country: "CO"

}

})

Paso 3: Abre el widget

Finalmente, en el momento en que quieras abrir el widget que configuraste anteriormente, simplemente debes llamar la función open pasándole como parámetro una función de respuesta (callback) que te entregará información sobre la transacción tan pronto esta finalice. Por ejemplo:

checkout.open(function (result) {

var transaction = result.transaction;

console.log("Transaction ID: ", transaction.id);

console.log("Transaction object: ", transaction);

});

AQUÍ EMPIEZA COMOHACER UNA PRUEBA

Datos de prueba en Sandbox

Para realizar una transacción de pruebas sólo debes asegurarte que estás usando la llave pública de comercio para el ambiente Sandbox. Recuerda que esta tiene el prefijo pub\_test\_.

A continuación verás los datos de prueba necesarios para cada uno de los métodos de pago:

Nequi

Para realizar transacciones aprobadas o rechazadas en el ambiente Sandbox sólo debes tener en cuenta los siguientes números:

3991111111 para generar una transacción aprobada (APPROVED)

3992222222 para generar una transacción declinada (DECLINED)

Ten en cuenta que cualquier otro número que utilices resultará en una transacción con status final en ERROR.

Por ejemplo:

{

// Otros campos de la transacción a crear...

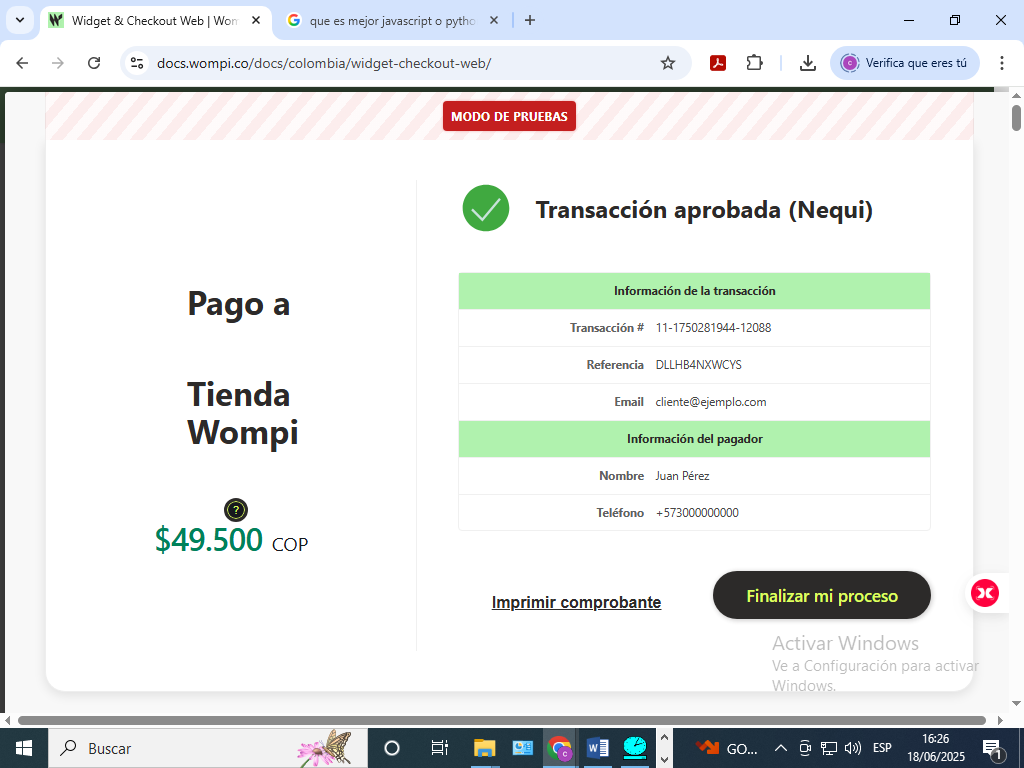
"payment\_method": {

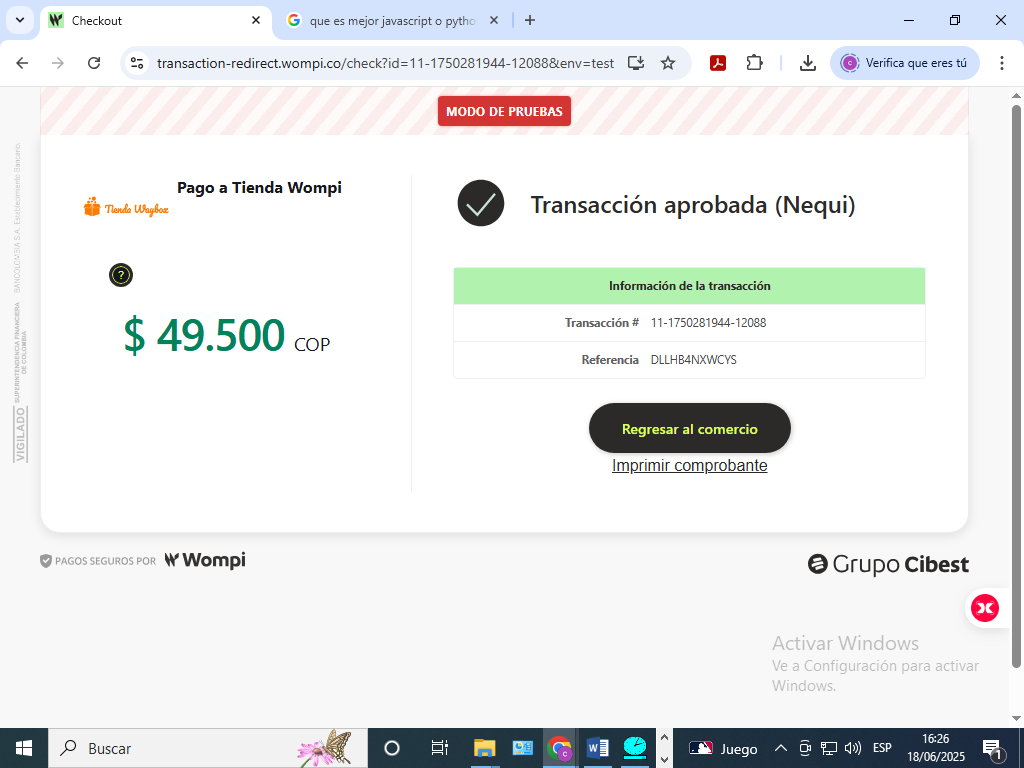
"type": "NEQUI",

"phone\_number": "3991111111" // Esto resultará current una transacción APROBADA

}

}





SE LE DIO A CHATGTP LA INFORMACION PARA ANALISIS AQUÍ ESTA RESPUESTA Y TAMBIEN SE HACE PREGUNTA DE SUGERENCIAS:

📥 **Información de integración con el widget de pagos de Wompi – Resumen estructurado y almacenado**

✅ **1. Requisitos Previos**

**HTTPS obligatorio:** Para mayor seguridad y evitar advertencias del navegador.

**JavaScript habilitado en el sitio.**

**Llave pública del comercio:**

Sandbox (pruebas): pub\_test\_...

Producción: pub\_prod\_...

✅ **2. Pasos para Integración con Widget Wompi**

**Paso 1: Llave pública**

Necesaria para autenticar la transacción.

Usar la de entorno adecuado: sandbox o producción.

**Paso 2: Generar referencia única**

Ejemplos válidos:

3b4393bafed398ba2

58e281-177ab976cbf9-d162d2

38932\_3293298

No reutilizar referencias.

**Paso 3: Generar firma de integridad**

**Formato:** "<Referencia><Monto><Moneda><SecretoIntegridad>"

**Encriptar con SHA256.**

**Ejemplo de hash:**  
Entrada: sk8-438k4-xmxm392-sn2m2490000COPprod\_integrity\_Z5mMke9x0k8gpErbDqwrJXMqsI6SFli6  
Hash SHA256: 37c8407747e595535433ef8f6a811d853cd943046624a0ec04662b17bbf33bf5

⚠️ Generar SIEMPRE en el backend (servidor).

**Paso 4: URL de redirección (opcional)**

Redirige al usuario después del pago.

Formato: https://tuweb.com/pagos/respuesta?id=<ID\_TRANSACCION>

Verifica el estado con la API:  
https://production.wompi.co/v1/transactions/<ID\_TRANSACCION>

**Paso 5: Parámetros de la transacción**

🔒 Obligatorios

| **Parámetro** | **Descripción** |
| --- | --- |
| data-public-key | Llave pública del comercio |
| data-currency | Moneda (solo COP por ahora) |
| data-amount-in-cents | Monto en centavos |
| data-reference | Referencia única del pago |
| data-signature:integrity | Firma SHA256 para validar integridad |

🎁 Opcionales

**data-redirect-url**

**data-expiration-time**: ISO 8601

**data-tax-in-cents:vat, data-tax-in-cents:consumption**

**data-customer-data:...**

**data-shipping-address:...**

**collect-shipping, collect-customer-legal-id**

✅ **6. Métodos de integración**

🎯 Widget automático (botón)

html

CopiarEditar

<form>

<script

src="https://checkout.wompi.co/widget.js"

data-render="button"

data-public-key="pub\_test\_X0zDA9xoKdePzhd8a0x9HAez7HgGO2fH"

data-currency="COP"

data-amount-in-cents="4950000"

data-reference="4XMPGKWWPKWQ"

data-signature:integrity="HASH\_AQUI"

></script>

</form>

🎨 Widget personalizado

html

CopiarEditar

<script src="https://checkout.wompi.co/widget.js"></script>

<script>

var checkout = new WidgetCheckout({

currency: 'COP',

amountInCents: 2490000,

reference: 'AD002901221',

publicKey: 'pub\_fENJ3hdTJxdzs3hd35PxDBSMB4f85VrgiY3b6s1',

signature: { integrity: 'HASH\_AQUI' },

redirectUrl: 'https://miweb.com/respuesta',

expirationTime: '2023-06-09T20:28:50.000Z',

taxInCents: { vat: 1900, consumption: 800 },

customerData: {

email: 'lola@gmail.com',

fullName: 'Lola Flores',

phoneNumber: '3040777777',

phoneNumberPrefix: '+57',

legalId: '123456789',

legalIdType: 'CC'

},

shippingAddress: {

addressLine1: 'Calle 123 # 4-5',

city: 'Bogota',

phoneNumber: '3019444444',

region: 'Cundinamarca',

country: 'CO'

}

});

checkout.open(function(result) {

var transaction = result.transaction;

console.log("Transaction ID:", transaction.id);

});

</script>

✅ **7. Eventos de confirmación (Webhook)**

**⚠️ Redirección no valida pagos.**

Crear un endpoint para recibir **webhooks** de eventos.

Wompi enviará un JSON con info de la transacción.

🧾 **Notas adicionales**

Impuestos (tax-in-cents) no se suman al amount-in-cents.

La información adicional (cliente, envío) es para experiencia del usuario, no afecta la lógica de pago.

Todos los datos sensibles y firmas se deben manejar en el backend.

¿Te gustaría que generemos un **ejemplo funcional en HTML** o un **backend en Node.js, PHP, Python u otro lenguaje** para esta integración?

Dijiste:

QUE ES LO MAS CONVENIENTE PARA LUEGO INTEGRARLOS CON OTROS PROGRAMAS Y MANEJADOR DE BASE DE DATOS GRATUITO PARA CONSTRUIR UNA PAGINA DE WEB GRATUITA QUE OFREZCA DATOS ESTADISTICOS Y OFRECER LISTAS DE PRONOSTICOS PARA REALIZAR PARLEYS DE JUEGOS DE FUTBOL

ChatGPT dijo:

Para construir una **página web gratuita** que ofrezca **datos estadísticos y pronósticos de fútbol** (como **listas de parlays**) y que además permita una **futura integración con pasarelas de pago como Wompi** y otros sistemas, te recomiendo una **arquitectura de desarrollo escalable, gratuita y modular**.

✅ **1. Frontend (Página Web Gratuita)**

**Opción recomendada:**

**[Next.js (React)]** + **Vercel (hosting gratuito)**

Permite SSR (Server-Side Rendering) y estáticas (Static Site Generation).

Vercel tiene **plan gratuito**, perfecto para hosting de páginas informativas o interactivas.

Altamente compatible con **API REST o GraphQL**, útil para conectar con bases de datos y Wompi.

Alternativas:

**Astro** o **Nuxt (si prefieres Vue)**.

**Netlify** para hosting si no usas Vercel.

HTML + JS puro si es algo simple.

✅ **2. Backend (API para estadísticas, usuarios, pagos, etc.)**

**Opción recomendada:**

**Node.js + Express** desplegado en **Render, Railway o Cyclic (gratuitos)**.

Puedes crear rutas como:

/api/pronosticos

/api/usuarios

/api/wompi/webhook

Ventajas:

Control total sobre la lógica del negocio.

Puedes usar librerías de scraping o consumir APIs deportivas.

Ideal para generar y verificar firmas SHA256 con Wompi.

✅ **3. Base de Datos Gratuita y Escalable**

**Opción recomendada:**

**PostgreSQL gratuito** en:

[Supabase](https://supabase.com)

[Neon](https://neon.tech)

[ElephantSQL](https://www.elephantsql.com)

Alternativa si prefieres NoSQL: **MongoDB Atlas** (plan gratuito también).

¿Por qué PostgreSQL?

Gratuito, potente y bien documentado.

Ideal para manejar usuarios, resultados, estadísticas, y registros de pagos.

Fácil de consultar desde backend o desde clientes directos con SQL.

✅ **4. Integración con Wompi**

Backend genera referencia + firma SHA256.

Frontend muestra widget de pago usando public-key.

Backend recibe Webhooks en una ruta protegida (/webhook/wompi) y actualiza la DB.

✅ **5. Fuente de Datos para Estadísticas y Juegos**

Para alimentar tu sistema con **partidos reales y datos útiles**, puedes usar:

APIs deportivas gratuitas:

[API-Football](https://www.api-football.com/) (gratuito limitado).

[Football-Data.org](https://www.football-data.org/)

ScoreBat API (videos + resultados)

[TheSportsDB](https://www.thesportsdb.com/)

✅ **6. Funciones adicionales que puedes ofrecer**

Pronósticos basados en estadísticas + IA básica (ej. usando Python en backend).

Envío de boletines por correo (Mailchimp o Resend).

Comparador de cuotas entre casas de apuestas.

🧩 Stack Tecnológico sugerido (todo gratuito):

| **Capa** | **Tecnología** |
| --- | --- |
| Frontend | Next.js + Vercel |
| Backend API | Node.js + Express |
| Base de Datos | PostgreSQL (Supabase) |
| Hosting | Vercel + Render |
| Pagos | Wompi (integración JS) |
| Datos fútbol | API-Football / TheSportsDB |
| Usuarios | Supabase Auth o JWT |

¿Quieres que te prepare un **starter kit** con:

un botón de pago de prueba con Wompi,

un sistema de base de datos para guardar apuestas o pronósticos,

y una plantilla para mostrar estadísticas y pronósticos?

Solo dime qué tecnologías quieres usar (React, HTML puro, etc.) y te lo armo.

Dijiste:

POR LOS MOMENTOS QUIERO QUIERO un botón de pago de prueba con Wompi, PARA EJECUTARLO Y VER SI FUNCIONA LUEGO HACEMOS EL RESTO

ChatGPT dijo:

Perfecto. A continuación te dejo un **ejemplo funcional y completo de un botón de pago de prueba con Wompi**, usando el **widget embebido** con una llave pública de pruebas (pub\_test\_...).

✅ **Botón de Pago de Prueba (Wompi)**

✔️ **Instrucciones**

Guarda este código en un archivo llamado index.html.

Ábrelo en cualquier navegador con conexión a Internet.

Verás un botón **"Paga con Wompi"** que abre el widget.

⚠️ Este ejemplo **no cobra dinero real**, ya que usa el entorno de pruebas.

📄 **Código HTML del botón de pago**

html

CopiarEditar

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<title>Botón de pago Wompi - Prueba</title>

</head>

<body>

<h2>Paga con Wompi (Modo de prueba)</h2>

<form>

<script

src="https://checkout.wompi.co/widget.js"

data-render="button"

data-public-key="pub\_test\_X0zDA9xoKdePzhd8a0x9HAez7HgGO2fH"

data-currency="COP"

data-amount-in-cents="4950000" <!-- $49500 COP -->

data-reference="TEST\_REF\_001"

data-signature:integrity="37c8407747e595535433ef8f6a811d853cd943046624a0ec04662b17bbf33bf5"

></script>

</form>

</body>

</html>

📌 Explicación de los valores

| **Campo** | **Descripción** |
| --- | --- |
| data-public-key | Llave pública del entorno sandbox |
| data-currency | Moneda (COP) |
| data-amount-in-cents | Monto total en centavos (4950000 = $49.500) |
| data-reference | Referencia única para el pago |
| data-signature:integrity | Hash SHA256 generado previamente (de ejemplo) |

🧪 ¿Qué sigue después de probarlo?

Si el botón **funciona y abre el widget**, entonces ya tienes lista la parte visual del pago.

Luego puedes conectarlo a un backend para:

Generar referencias y firmas dinámicas.

Escuchar webhooks para confirmar pagos.

Guardar en base de datos la info de pagos y usuarios.

¿Quieres que también te muestre cómo generar esa firma SHA256 en Node.js o en otro lenguaje (para que no sea fija y puedas cambiar el monto)?

TENER EN CUENTA PARA EL PASO 4 , SE CREAN 2 HTML

✅ ¿Por qué hay **2 HTML**?

**1️⃣ Primer HTML**  
Es tu página principal donde está el botón de pago de Wompi (es decir, **donde el usuario inicia el pago**).

**2️⃣ Segundo HTML**  
Es la **página de redirección**, a la que **Wompi lleva al usuario después del pago** con el ID de la transacción.

🧭 ¿Por qué deben estar separados?

Wompi **redirige automáticamente** a otra URL (que tú defines con data-redirect-url). No puedes hacer que “todo pase en el mismo HTML”, porque:

El botón de pago **no puede cambiar de contexto** una vez iniciado.

Wompi necesita redirigir a una página **nueva**, con el ID de la transacción como parámetro.

🧩 ¿Qué contiene cada uno?

| **Archivo** | **Función** | **Acción del usuario** |
| --- | --- | --- |
| index.html (o pago.html) | Contiene el botón de Wompi y la integración completa de pasos 1 al 4 | El usuario hace clic en “Pagar” |
| respuesta.html (o /pagos/respuesta) | Muestra el mensaje de agradecimiento + el ID de la transacción | Wompi redirige al finalizar el pago |

✅ ¿Qué puedes hacer tú?

Tener ambos archivos en tu estructura de sitio web. Por ejemplo:

bash

CopiarEditar

/pago.html

/pagos/respuesta.html

Y en tu HTML de integración, apuntas la redirección así:

html

CopiarEditar

data-redirect-url="https://tudominio.com/pagos/respuesta.html"

Si prefieres trabajar **todo dentro de un framework o backend (PHP, Node, etc.)**, también se puede, pero en una integración estática y de pruebas, **lo correcto es usar dos páginas**.

¿Listo para que avancemos al **Paso 5** o quieres que empaquete ambos archivos como una sola carpeta para probar?

Dijiste:

ok , esto quiere decir que cuando terminemos todos los pasos hay que hacer ub HTML de integración,donde se apunta la redirección así: data-redirect-url="https://tudominio.com/pagos/respuesta.html"

ChatGPT dijo:

Exactamente. Lo entendiste perfecto. ✅

🧱 ¿Qué vas a tener al final?

Tendrás al menos **dos archivos HTML bien definidos**:

✅ 1. **HTML principal de integración**

Este es donde se configura e integra el botón de pago de Wompi (con los pasos 1 al 7).  
En este archivo, en el <script> del widget, incluyes:

html

CopiarEditar

data-redirect-url="https://tudominio.com/pagos/respuesta.html"

✅ 2. **HTML de respuesta (redirección)**

Este es el archivo **respuesta.html**, donde llega el usuario después del pago. Aquí puedes mostrar:

Mensajes de agradecimiento

El ID de la transacción

O incluso consultar el estado final usando el API de Wompi si lo deseas más adelante

📌 ¿Por qué es así?

Wompi no puede “inyectar” información dentro del mismo HTML que contiene el botón de pago.  
Por eso **la redirección siempre es a una URL externa** que tú defines, y que debe **existir en tu dominio**.

respuesta.html (página estática para redirección)

html

CopiarEditar

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Respuesta de pago</title>

<script>

// Lee el parámetro ID desde la URL

const urlParams = new URLSearchParams(window.location.search);

const transactionId = urlParams.get('id');

</script>

</head>

<body>

<h2>Gracias por tu pago</h2>

<p>Estamos validando la transacción con ID: <strong id="transaction-id"></strong></p>

<script>

document.getElementById("transaction-id").textContent = transactionId || "No se recibió ID";

</script>

</body>

</html>

HTML COMPLETO DE INTEGRACION

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Integración con Wompi - EduPago</title>

</head>

<body>

<!--

Paso 1: Llave pública en modo sandbox

-------------------------------------

Llave pública usada para pruebas.

Prefijo pub\_test\_ indica que estamos en entorno de pruebas.

-->

<!--

Paso 2: Referencia única de pago

----------------------------------

Esta referencia nos permite identificar de forma única la transacción.

Se usará "edupago" como ejemplo en esta integración.

-->

<!--

Paso 3: Firma de integridad

----------------------------

Firma generada con SHA256 concatenando:

referencia + monto (en centavos) + moneda + secreto\_integridad

Resultado para esta transacción: e14f42c03241321cef316601e1da9a2cef4086178557c7638e940f526e06f454

Es recomendable generar esta firma siempre desde el servidor, no desde el frontend.

-->

<!--

Paso 4: URL de redirección

--------------------------

Esta es una URL estática que debe apuntar a una página de tu sitio.

En este caso usamos: respuesta.html (se construirá al final).

Luego debe reemplazarse por la URL real del sitio.

-->

<!--

Paso 5: Parámetros de la transacción

-------------------------------------

Incluimos solo los campos obligatorios:

- public-key

- currency

- amount-in-cents

- reference

- signature:integrity

-->

<!--

Paso 6: Widget de integración

------------------------------

Incluimos el botón de pago usando el widget oficial de Wompi.

El cliente podrá completar el pago directamente desde tu sitio.

-->

<!--

Paso 7: Webhook

----------------

El webhook se ha registrado en el dashboard:

https://wompi-backend-zodk.onrender.com/webhook

Wompi enviará eventos de actualización de transacción a esa URL.

Este paso no requiere código en el frontend, pero es crítico para la validación automática.

-->

<h1>EduPago - Pagos en línea</h1>

<p>Haz clic en el botón para realizar tu pago seguro con Wompi:</p>

<form>

<script

src="https://checkout.wompi.co/widget.js"

data-render="button"

data-public-key="pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG"

data-currency="COP"

data-amount-in-cents="150000"

data-reference="edupago"

data-signature:integrity="e14f42c03241321cef316601e1da9a2cef4086178557c7638e940f526e06f454"

data-redirect-url="https://tudominio.com/pagos/respuesta.html"

></script>

</form>

</body>

</html>

ESTE INDEX.HTML FUNCIONO BIEN PERO COMO TENIA UNA SOLA REFERENCIA , NO SE PUDO HACER SOLO UNA PRUEBA QUE GENERO COMPROBANTE

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Pago con Wompi - Integración Sandbox</title>

</head>

<body>

<h1>💳 Botón de pago con Wompi</h1>

<!-- 🔹 Paso 1: Llave pública de comercio (modo sandbox) -->

<!-- Llave utilizada: pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG -->

<!-- 🔹 Paso 2: Referencia única de pago -->

<!-- Referencia usada: edupago -->

<!-- 🔹 Paso 3: Firma de integridad -->

<!-- Firma generada con referencia "edupago", valor $1.500 COP y secreto de integridad en modo sandbox -->

<!-- Firma generada: e14f42c03241321cef316601e1da9a2cef4086178557c7638e940f526e06f454 -->

<!-- 🔹 Paso 4: URL de redirección (página de confirmación estática) -->

<!-- Esta es una página que debe existir en tu dominio, por ejemplo: respuesta.html -->

<!-- IMPORTANTE: Es una página HTML estática que muestra el ID de transacción recibido por GET -->

<!-- Ejemplo: https://tudominio.com/pagos/respuesta.html?id=01-23432423-2342 -->

<!-- 🔹 Paso 5: Parámetros obligatorios de la transacción -->

<!-- Incluidos dentro del widget (públicamente visibles, no deben contener secretos privados) -->

<!-- 🔹 Paso 6: Método de integración usando el widget oficial de Wompi -->

<!-- Widget embebido dentro de tu sitio web -->

<!-- 🔹 Paso 7: Webhook ya configurado en el dashboard (ya lo configuraste en Wompi) -->

<!-- Para pruebas, fue reemplazado por: https://eorlzwmbuwv5oh9.m.pipedream.net -->

<form>

<script

src="https://checkout.wompi.co/widget.js"

data-render="button"

data-public-key="pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG"

data-currency="COP"

data-amount-in-cents="150000"

data-reference="edupago"

data-signature:integrity="e14f42c03241321cef316601e1da9a2cef4086178557c7638e940f526e06f454"

data-redirect-url="https://tudominio.com/respuesta.html"

></script>

</form>

<hr/>

<p>💡 <strong>Modo Sandbox:</strong> Esta página simula un botón de pago para pruebas con Wompi. No se realiza ningún cobro real.</p>

</body>

</html>

ESTE PROCESO SE EMPIEZA HACER PARA REALIZAR VARIAS PRUEBAS

Paso 1

Perfecto, aquí tienes el archivo app.py completamente generado con todos los cambios necesarios para recibir peticiones desde el formulario de pagos, generar la firma correctamente, y listo para probar en Render con Flask:

### ✅ Código completo: app.py

python

CopiarEditar

from flask import Flask, request, jsonify

import hashlib

app = Flask(\_\_name\_\_)

# 🔐 Clave de integridad del entorno sandbox de Wompi

SECRET = "test\_integrity\_CxvWC5XqHUC8eownzkyCMKjfOujwVmqk"

@app.route('/')

def index():

return 'Servidor Flask activo'

# 📬 Endpoint para generar la firma de integridad

@app.route('/generate-signature', methods=['POST'])

def generate\_signature():

data = request.get\_json()

amount = data.get('amount\_in\_cents')

currency = data.get('currency')

reference = data.get('reference')

if not amount or not currency or not reference:

return jsonify({"error": "Faltan parámetros"}), 400

# 🔐 Concatenar en el orden: referencia + monto + moneda + llave privada

cadena = f"{reference}{amount}{currency}{SECRET}"

firma = hashlib.sha256(cadena.encode('utf-8')).hexdigest()

return jsonify({"signature": firma})

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

### ✅ ¿Qué hace este archivo?

| **Sección** | **Función** |
| --- | --- |
| / | Verifica que el servidor esté activo |
| /generate-signature | Genera la firma hash SHA256 que Wompi necesita para validar el botón de pago |

## 🪜 PASOS SIGUIENTES

### 🔧 Paso 1: Guarda este archivo como app.py en tu carpeta del proyecto

**Ruta esperada:**  
C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend\app.py

### 💻 Paso 2: Sube a GitHub (si usas GitHub)

Abre una terminal (CMD o Git Bash) y ejecuta estos comandos:

bash

CopiarEditar

cd "C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend"

git add app.py

git commit -m "Actualización: endpoint /generate-signature en Flask"

git push origin main

RESULTADOS DESPUES DE SER EJECUTADOS COMANDOS DESDE EL CURSOR:

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>git add app.py

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>git commit -m "Actualización: endpoint /generate-signature en Flask"

[main 4f50ef2] Actualización: endpoint /generate-signature en Flask

1 file changed, 20 insertions(+), 29 deletions(-)

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>git push origin main

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 4 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 829 bytes | 276.00 KiB/s, done.

Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (1/1), completed with 1 local object.

To https://github.com/edufutbol832/edubacken.git

c811068..4f50ef2 main -> main

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>

Paso 2

## OPCIÓN RECOMENDADA: Entrar con GitHub

Como **ya usaste GitHub antes** para conectar tu backend a Render, **lo mejor es volver a iniciar sesión con GitHub**. Así Render cargará automáticamente todos tus servicios vinculados.

### 🔗 Link directo que te permite elegir GitHub:

Haz clic en este enlace:

👉 https://dashboard.render.com/login

Ese enlace **te llevará directamente a la pantalla de inicio de sesión** con estas opciones:

* **Sign in with GitHub** ← ✅ haz clic aquí.
* Sign in with Google
* Sign in with Email

## ✅ ¿Qué sigue después de entrar?

Una vez entres con GitHub:

1. Verás tu **Dashboard de Render**.
2. Ahí debe aparecer el servicio que creaste, por ejemplo:

CopiarEditar

wompi-backend

Web Service - Flask

1. Haz clic en ese nombre (la tarjeta).
2. Entrarás al panel del servicio.
3. Pulsa el botón **Manual Deploy** (Desplegar manualmente) si has hecho cambios en GitHub.
4. Espera que diga: ✅ “Live” o “Build complete”.

## 🟨 ¿Y si no ves tu proyecto?

Si por alguna razón no aparece el proyecto:

* Es posible que hayas iniciado sesión con otro método (como Google o email).
* En ese caso, cierra sesión y entra con el método correcto (GitHub).

¿Listo para seguir este paso o prefieres hacerlo manual sin GitHub? Te ayudo en cualquiera de los dos casos.

##### Dijiste:

Ungrouped Services active (1) suspended (0) all (1) Search services Search services Select all rows Service Name 1 Select row wompi-backend Options Status Deployed Runtime Python 3 Region oregon Deployed 45min

###### ChatGPT dijo:

¡Perfecto! 🙌 Ya estás dentro de tu **servicio wompi-backend** en Render. Y confirmamos que:

* ✅ El servicio se llama: wompi-backend
* ✅ Está **activo (Deployed)** y con **Runtime: Python 3**
* ✅ Región: Oregon
* ✅ Último despliegue: hace 45 minutos

**Paso 3 Subir los cambios a GitHub**

Claro, aquí tienes nuevamente el archivo app.py completamente actualizado, listo para generar firmas para Wompi usando Flask con el secreto de integridad del entorno **sandbox**.

### ✅ app.py — versión final para firmar transacciones:

python

CopiarEditar

from flask import Flask, request, jsonify

import hashlib

app = Flask(\_\_name\_\_)

# 🔐 Secreto de integridad proporcionado por Wompi (modo sandbox)

SECRET = "test\_integrity\_CxvWC5XqHUC8eownzkyCMKjfOujwVmqk"

@app.route('/')

def index():

return 'Servidor Flask activo'

# 🚀 Endpoint para generar la firma SHA256

@app.route('/generate-signature', methods=['POST'])

def generate\_signature():

data = request.get\_json()

# ⚠️ Validamos que todos los datos estén presentes

amount = data.get('amount\_in\_cents')

currency = data.get('currency')

reference = data.get('reference')

if not amount or not currency or not reference:

return jsonify({"error": "Faltan parámetros"}), 400

# 🧮 Concatenación en orden requerido por Wompi

cadena = f"{reference}{amount}{currency}{SECRET}"

# 🔐 Firma con SHA256

firma = hashlib.sha256(cadena.encode('utf-8')).hexdigest()

# 🟢 Devolvemos la firma como JSON

return jsonify({"signature": firma})

# 🔧 Solo para pruebas locales

if \_\_name\_\_ == '\_\_main\_\_':

app.run()

### 🧩 ¿Qué hace este backend?

* Escucha en la raíz / para verificar que está activo.
* Responde en /generate-signature y devuelve la firma SHA256 para una transacción.
* Usa el **secreto de integridad de Wompi sandbox**.

### ✅ Próximo paso:

Ahora que ya tienes el archivo app.py listo:

1. Guarda este archivo en tu carpeta del backend local.
2. **Sube los cambios a GitHub** con los comandos:

bash

CopiarEditar

git add app.py

git commit -m "Agregado endpoint /generate-signature para Wompi"

git push origin main

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>git add app.py

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>git commit -m "Agregado endpoint /generate-signature para Wompi"

[main 0c42392] Agregado endpoint /generate-signature para Wompi

1 file changed, 8 insertions(+), 3 deletions(-)

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>git push origin main

Enumerating objects: 5, done.

Counting objects: 100% (5/5), done.

Delta compression using up to 4 threads

Compressing objects: 100% (3/3), done.

Writing objects: 100% (3/3), 573 bytes | 95.00 KiB/s, done.

Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)

remote: Resolving deltas: 100% (2/2), completed with 2 local objects.

To https://github.com/edufutbol832/edubacken.git

4f50ef2..0c42392 main -> main

C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\wompi-backend>

DESPUES DE HACER LAS PRUEBAS ESTE HMTL YA SIRVIO PARA HACER VARIAS PAGOS..

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

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

<title>Pago con Wompi - Referencia dinámica</title>

</head>

<body>

<h1>💳 Pago con Wompi - Referencia Dinámica</h1>

<div id="button-container">

<!-- Aquí se inyectará el botón Wompi cuando tengamos la firma -->

<p>Cargando botón de pago...</p>

</div>

<script>

async function generarReferencia() {

// Puedes personalizar la referencia como prefieras

return 'edupago-' + Date.now();

}

async function obtenerFirma(reference) {

const payload = {

amount\_in\_cents: 150000,

currency: "COP",

reference: reference

};

try {

const response = await fetch('https://wompi-backend-zodk.onrender.com/generate-signature', {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(payload)

});

if (!response.ok) throw new Error('Error en la solicitud al backend');

const data = await response.json();

return data.signature;

} catch (error) {

console.error('Error obteniendo firma:', error);

alert('Error al obtener la firma. Revisa consola.');

}

}

async function cargarBotonWompi() {

const reference = await generarReferencia();

const signature = await obtenerFirma(reference);

if (!signature) {

document.getElementById('button-container').innerHTML = '<p>Error al cargar el botón de pago.</p>';

return;

}

// Limpio contenedor

const contenedor = document.getElementById('button-container');

contenedor.innerHTML = '';

// Creo el script para Wompi

const script = document.createElement('script');

script.src = "https://checkout.wompi.co/widget.js";

script.setAttribute('data-render', 'button');

script.setAttribute('data-public-key', 'pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG');

script.setAttribute('data-currency', 'COP');

script.setAttribute('data-amount-in-cents', '150000');

script.setAttribute('data-reference', reference);

script.setAttribute('data-signature:integrity', signature);

script.setAttribute('data-redirect-url', 'https://tudominio.com/respuesta.html'); // Cambia esto a tu URL real

contenedor.appendChild(script);

console.log('Referencia generada:', reference);

console.log('Firma obtenida:', signature);

}

// Ejecutamos la función al cargar la página

cargarBotonWompi();

</script>

</body>

</html>

Otro parece completo tengo pero no se si sea igual que al anterior tengo duda

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Pago con Wompi - Referencia dinámica</title>

</head>

<body>

<h1>💳 Botón de pago con Wompi</h1>

<div id="wompi-container">

<p>Generando referencia y firma...</p>

</div>

<script>

// 🔄 Paso 1: Generar referencia única

const referencia = "edupago-" + Date.now();

console.log("Referencia generada:", referencia);

// 🔄 Paso 2: Llamar a tu backend para obtener la firma

const datos = {

amount\_in\_cents: 150000,

currency: "COP",

reference: referencia

};

fetch("https://wompi-backend-zodk.onrender.com/generate-signature", {

method: "POST",

headers: {

"Content-Type": "application/json"

},

body: JSON.stringify(datos)

})

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

.then(data => {

if (!data.signature) throw new Error("Firma no recibida");

// 🔄 Paso 3: Mostrar el botón de pago con datos completos

const script = document.createElement("script");

script.src = "https://checkout.wompi.co/widget.js";

script.setAttribute("data-render", "button");

script.setAttribute("data-public-key", "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG");

script.setAttribute("data-currency", "COP");

script.setAttribute("data-amount-in-cents", "150000");

script.setAttribute("data-reference", referencia);

script.setAttribute("data-signature:integrity", data.signature);

script.setAttribute("data-redirect-url", "https://wompi-backend-zodk.onrender.com/respuesta.html");

const container = document.getElementById("wompi-container");

container.innerHTML = ""; // Limpiar mensaje inicial

container.appendChild(script);

})

.catch(error => {

document.getElementById("wompi-container").innerHTML = `<p style="color:red;">❌ Error al generar firma o referencia.</p><p>${error.message}</p>`;

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

});

</script>

<hr/>

<p>🧪 <strong>Modo Sandbox:</strong> Puedes hacer múltiples pruebas sin errores de "referencia duplicada".</p>

</body>

</html>