Perfecto, Eduardo.  
Estaré aquí para retomar exactamente desde donde lo dejaste.  
¡Éxitos con tu proyecto y que descanses! 🌙💻📊

Para convertir de jpg a Excel

https://online2pdf.com/es/convertir-jpg-a-excel#

prueba2.py

python prueba2.py > salida.txt 2>&1

python pruebahtml.py > salida.txt 2>&1

python firma.py > salida.txt 2>&1

python app.py > salida.txt 2>&1

CREA EL HTML. DESDE PYTHON

crear\_html.py

python crear\_html.py > salida.txt 2>&1

CREAR LISTAS PDF

crear\_lista.py

Python crear\_lista.py > salida.txt 2>&1

import pandas as pd

from datetime import datetime

# Texto fuente (hora en España, sin conversión)

texto = """

09:15 31 Mayo Ferrol 2.70 Empate 2.95 Sporting Gij 2.75

11:30 31 Mayo Malaga 2.15 Empate 3.20 Burgos CF 3.45

13:30 01 Jun Cartagena 13.00 Empate 6.25 Mirandes 1.20

11:30 01 Jun Almeria 1.25 Empate 6.00 CD Tenerife 10.00

11:30 01 Jun Huesca 2.15 Empate 3.00 Cordobense 3.60

11:30 01 Jun Deportivo 5.50 Empate 4.33 Elche 1.50

11:30 01 Jun Levante 1.53 Empate 4.20 Eibar 4.75

11:30 01 Jun Racing Santander 2.75 Empate 3.50 Granada 2.40

11:30 01 Jun Real Oviedo 1.70 Empate 3.10 Cadiz 6.25

14:00 01 Jun Cordoba 1.60 Empate 4.00 Albacete 5.25

"""

# Mapeo de meses en español a inglés

meses = {

"Ene": "Jan", "Feb": "Feb", "Mar": "Mar", "Abr": "Apr", "May": "May", "Mayo": "May",

"Jun": "Jun", "Jul": "Jul", "Ago": "Aug", "Sep": "Sep", "Oct": "Oct", "Nov": "Nov", "Dic": "Dec"

}

# Función para clasificar la jornada según la hora

def clasificar\_jornada(hora\_str):

h = int(hora\_str.split(':')[0])

if h < 12:

return "Mañana"

elif h < 18:

return "Tarde"

else:

return "Noche"

partidos = []

for linea in texto.strip().split('\n'):

partes = linea.split()

hora = partes[0]

dia = partes[1]

mes\_es = partes[2]

mes\_en = meses.get(mes\_es, mes\_es)

fecha\_str = f"{dia} {mes\_en} 2025"

fecha\_mostrar = f"{dia} {mes\_es} 2025"

fecha\_obj = datetime.strptime(fecha\_str, "%d %b %Y")

cuotas\_indices = [i for i, x in enumerate(partes) if any(c.isdigit() for c in x) and '.' in x]

cuota\_local\_idx = cuotas\_indices[0]

cuota\_empate\_idx = cuotas\_indices[1]

cuota\_visitante\_idx = cuotas\_indices[2]

local = " ".join(partes[3:cuota\_local\_idx])

cuota\_local = partes[cuota\_local\_idx]

visitante = " ".join(partes[cuota\_empate\_idx + 1:cuota\_visitante\_idx])

jornada = clasificar\_jornada(hora)

partidos.append({

"Fecha": fecha\_mostrar,

"Hora": hora,

"Jornada": jornada,

"Equipo Local": local,

"Equipo Visitante": visitante,

"Cuota Local": cuota\_local

})

# Crear el DataFrame y guardar como Excel

df = pd.DataFrame(partidos)

df.to\_excel("partidos\_liga2\_01\_06\_2025\_sin\_conversion.xlsx", index=False)

print("Archivo Excel generado correctamente: partidos\_liga2\_01\_06\_2025\_sin\_conversion.xlsx")

este es el ultimo salio bien hay que revisar las posiciones

import pandas as pd

from datetime import datetime

# Texto fuente con horarios en hora de Colombia (ya corregidos)

texto = """

09:15 31 Mayo Ferrol 2.70 Empate 2.95 Sporting Gij 2.75

11:30 31 Mayo Malaga 2.15 Empate 3.20 Burgos CF 3.45

13:30 01 Jun Cartagena 13.00 Empate 6.25 Mirandes 1.20

11:30 01 Jun Almeria 1.25 Empate 6.00 CD Tenerife 10.00

11:30 01 Jun Huesca 2.15 Empate 3.00 Cordobense 3.60

11:30 01 Jun Deportivo 5.50 Empate 4.33 Elche 1.50

11:30 01 Jun Levante 1.53 Empate 4.20 Eibar 4.75

11:30 01 Jun Racing Santander 2.75 Empate 3.50 Granada 2.40

11:30 01 Jun Real Oviedo 1.70 Empate 3.10 Cadiz 6.25

14:00 01 Jun Cordoba 1.60 Empate 4.00 Albacete 5.25

"""

# Mapeo de meses en español a inglés para datetime

meses = {

"Ene": "Jan", "Feb": "Feb", "Mar": "Mar", "Abr": "Apr", "May": "May", "Mayo": "May",

"Jun": "Jun", "Jul": "Jul", "Ago": "Aug", "Sep": "Sep", "Oct": "Oct", "Nov": "Nov", "Dic": "Dec"

}

# Diccionario de posiciones actualizado, incluyendo Málaga con posición 5

posiciones = {

"Levante": 1,

"Elche": 2,

"Almeria": 3,

"Mirandes": 4,

"Malaga": 5, # Málaga corregido sin tilde para buscar bien

"Sporting Gij": 6,

"Burgos CF": 7,

"Granada": 8,

"Eibar": 9,

"Cadiz": 10,

"Cordoba": 11,

"Racing Santander": 12,

"Huesca": 13,

"Deportivo": 14,

"Cartagena": 15,

"Ferrol": 16,

"Real Oviedo": 17,

"Albacete": 18,

"CD Tenerife": 19,

"Cordobense": 20

}

# Correcciones de nombres (tilde, abreviaturas, etc.) para buscar en posiciones

correcciones\_equipos = {

"Málaga": "Malaga", # tilde eliminada para búsqueda correcta

"Sporting Gij": "Sporting Gij",

"CD Tenerife": "CD Tenerife",

"Cordobense": "Cordobense",

"Burgos CF": "Burgos CF",

# Agrega más si fuera necesario

}

def corregir\_nombre(equipo):

return correcciones\_equipos.get(equipo, equipo)

def obtener\_jornada\_por\_hora(hora\_str):

hora\_dt = datetime.strptime(hora\_str, "%H:%M")

if hora\_dt.hour >= 6 and hora\_dt.hour < 12:

return "Mañana"

elif hora\_dt.hour >= 12 and hora\_dt.hour < 19:

return "Tarde"

else:

return "Noche"

# Procesar texto línea por línea

partidos = []

for linea in texto.strip().split('\n'):

partes = linea.split()

hora\_col = partes[0]

dia = partes[1]

mes\_es = partes[2]

mes\_en = meses.get(mes\_es, mes\_es)

fecha\_str = f"{dia} {mes\_en} 2025"

fecha\_obj = datetime.strptime(fecha\_str, "%d %b %Y")

# Buscar índices de cuotas para separar equipos

cuotas\_indices = [i for i, x in enumerate(partes) if any(c.isdigit() for c in x) and '.' in x]

cuota\_local\_idx = cuotas\_indices[0]

cuota\_empate\_idx = cuotas\_indices[1]

cuota\_visitante\_idx = cuotas\_indices[2]

# Equipos local y visitante

equipo\_local = " ".join(partes[3:cuota\_local\_idx])

equipo\_visitante = " ".join(partes[cuota\_empate\_idx + 1:cuota\_visitante\_idx])

# Cuota local

cuota\_local = partes[cuota\_local\_idx]

# Jornada según hora

jornada = obtener\_jornada\_por\_hora(hora\_col)

partidos.append({

"Fecha": fecha\_obj.strftime("%d %b %Y"),

"Hora": hora\_col,

"Jornada": jornada,

"Equipo Local": equipo\_local,

"Equipo Visitante": equipo\_visitante,

"Cuota Local": cuota\_local

})

# Crear DataFrame

df = pd.DataFrame(partidos)

# Función para obtener posición

def obtener\_posicion(equipo, tipo):

nombre\_corregido = corregir\_nombre(equipo)

pos = posiciones.get(nombre\_corregido)

if pos is None:

print(f"Advertencia: No se encontró posición para el equipo {tipo}: '{equipo}' (corregido a '{nombre\_corregido}')")

return "N/A"

return pos

# Agregar columnas de posición

df['PosLocal'] = df['Equipo Local'].apply(lambda x: obtener\_posicion(x, "local"))

df['PosVisit'] = df['Equipo Visitante'].apply(lambda x: obtener\_posicion(x, "visitante"))

# Reordenar columnas según pedido

df = df[["Fecha", "Hora", "Jornada", "Equipo Local", "PosLocal", "PosVisit", "Equipo Visitante", "Cuota Local"]]

# Guardar archivo Excel (modifica ruta según donde quieras guardarlo)

archivo\_excel = r"C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\INSTALADOR DE PYTHON\PROGRAMA DE PRUEBA PYTHON\partidos\_liga2\_01\_06\_2025\_con\_posiciones\_jornada\_hora.xlsx"

df.to\_excel(archivo\_excel, index=False)

print(f"Archivo Excel generado correctamente: {archivo\_excel}")

Ultima versión

import pandas as pd

from datetime import datetime

# Texto fuente con horarios en hora de Colombia (ya corregidos)

texto = """

09:15 31 Mayo Ferrol 2.70 Empate 2.95 Sporting Gij 2.75

11:30 31 Mayo Malaga 2.15 Empate 3.20 Burgos CF 3.45

13:30 01 Jun Cartagena 13.00 Empate 6.25 Mirandes 1.20

11:30 01 Jun Almeria 1.25 Empate 6.00 CD Tenerife 10.00

11:30 01 Jun Huesca 2.15 Empate 3.00 Cordobense 3.60

11:30 01 Jun Deportivo 5.50 Empate 4.33 Elche 1.50

11:30 01 Jun Levante 1.53 Empate 4.20 Eibar 4.75

11:30 01 Jun Racing Santander 2.75 Empate 3.50 Granada 2.40

11:30 01 Jun Real Oviedo 1.70 Empate 3.10 Cadiz 6.25

14:00 01 Jun Cordoba 1.60 Empate 4.00 Albacete 5.25

"""

# Mapeo de meses en español a inglés para datetime

meses = {

"Ene": "Jan", "Feb": "Feb", "Mar": "Mar", "Abr": "Apr", "May": "May", "Mayo": "May",

"Jun": "Jun", "Jul": "Jul", "Ago": "Aug", "Sep": "Sep", "Oct": "Oct", "Nov": "Nov", "Dic": "Dec"

}

# Diccionario de posiciones actualizado, incluyendo Málaga con posición 5

posiciones = {

"Levante": 1,

"Elche": 2,

"Almeria": 3,

"Mirandes": 4,

"Malaga": 5, # Málaga corregido sin tilde para buscar bien

"Sporting Gij": 6,

"Burgos CF": 7,

"Granada": 8,

"Eibar": 9,

"Cadiz": 10,

"Cordoba": 11,

"Racing Santander": 12,

"Huesca": 13,

"Deportivo": 14,

"Cartagena": 15,

"Ferrol": 16,

"Real Oviedo": 17,

"Albacete": 18,

"CD Tenerife": 19,

"Cordobense": 20

}

# Correcciones de nombres (tilde, abreviaturas, etc.) para buscar en posiciones

correcciones\_equipos = {

"Málaga": "Malaga", # tilde eliminada para búsqueda correcta

"Sporting Gij": "Sporting Gij",

"CD Tenerife": "CD Tenerife",

"Cordobense": "Cordobense",

"Burgos CF": "Burgos CF",

# Agrega más si fuera necesario

}

def corregir\_nombre(equipo):

return correcciones\_equipos.get(equipo, equipo)

def obtener\_jornada\_por\_hora(hora\_str):

hora\_dt = datetime.strptime(hora\_str, "%H:%M")

if hora\_dt.hour >= 6 and hora\_dt.hour < 12:

return "Mañana"

elif hora\_dt.hour >= 12 and hora\_dt.hour < 19:

return "Tarde"

else:

return "Noche"

# Procesar texto línea por línea

partidos = []

for linea in texto.strip().split('\n'):

partes = linea.split()

hora\_col = partes[0]

dia = partes[1]

mes\_es = partes[2]

mes\_en = meses.get(mes\_es, mes\_es)

fecha\_str = f"{dia} {mes\_en} 2025"

fecha\_obj = datetime.strptime(fecha\_str, "%d %b %Y")

# Buscar índices de cuotas para separar equipos

cuotas\_indices = [i for i, x in enumerate(partes) if any(c.isdigit() for c in x) and '.' in x]

cuota\_local\_idx = cuotas\_indices[0]

cuota\_empate\_idx = cuotas\_indices[1]

cuota\_visitante\_idx = cuotas\_indices[2]

# Equipos local y visitante

equipo\_local = " ".join(partes[3:cuota\_local\_idx])

equipo\_visitante = " ".join(partes[cuota\_empate\_idx + 1:cuota\_visitante\_idx])

# Cuota local

cuota\_local = partes[cuota\_local\_idx]

# Jornada según hora

jornada = obtener\_jornada\_por\_hora(hora\_col)

partidos.append({

"Fecha": fecha\_obj.strftime("%d %b %Y"),

"Hora": hora\_col,

"Jornada": jornada,

"Equipo Local": equipo\_local,

"Equipo Visitante": equipo\_visitante,

"Cuota Local": cuota\_local

})

# Crear DataFrame

df = pd.DataFrame(partidos)

# Función para obtener posición

def obtener\_posicion(equipo, tipo):

nombre\_corregido = corregir\_nombre(equipo)

pos = posiciones.get(nombre\_corregido)

if pos is None:

print(f"Advertencia: No se encontró posición para el equipo {tipo}: '{equipo}' (corregido a '{nombre\_corregido}')")

return "N/A"

return pos

# Agregar columnas de posición

df['PosLocal'] = df['Equipo Local'].apply(lambda x: obtener\_posicion(x, "local"))

df['PosVisit'] = df['Equipo Visitante'].apply(lambda x: obtener\_posicion(x, "visitante"))

# Reordenar columnas según pedido

df = df[["Fecha", "Hora", "Jornada", "Equipo Local", "PosLocal", "PosVisit", "Equipo Visitante", "Cuota Local"]]

# Guardar archivo Excel (modifica ruta según donde quieras guardarlo)

archivo\_excel = r"C:\Users\WIN10\Desktop\DISEÑO DE PAGINAS WEB\INSTALADOR DE PYTHON\PROGRAMA DE PRUEBA PYTHON\partidos\_liga2\_01\_06\_2025\_con\_posiciones\_jornada\_hora.xlsx"

df.to\_excel(archivo\_excel, index=False)

print(f"Archivo Excel generado correctamente: {archivo\_excel}")

VERSION DEL 03/06/2025 SIN HTM

import pandas as pd

from datetime import datetime

# Pedir variables iniciales

PAIS = input("Nombre del país: ").strip()

LIGA = input("Nombre de la liga: ").strip()

FECHA1 = input("Fecha inicial (DD-MM-YYYY): ").strip()

FECHA2 = input("Fecha final (DD-MM-YYYY): ").strip()

TERMINAR = input("Escribe 'fin' para terminar el proceso ahora o presiona Enter para continuar: ").strip().lower()

# Salir si TERMINAR es 'fin'

if TERMINAR == "fin":

print("🚫 Proceso finalizado por el usuario antes de ingresar partidos.")

else:

fecha\_inicio = datetime.strptime(FECHA1, "%d-%m-%Y").date()

fecha\_fin = datetime.strptime(FECHA2, "%d-%m-%Y").date()

partidos = []

print("\n➡️ Ingresa los datos de cada partido.")

print("✏️ Escribe 'fin' como Equipo Local para terminar.\n")

while True:

equipo\_local = input("Equipo Local: ").strip()

if equipo\_local.lower() == "fin":

break

equipo\_visitante = input("Equipo Visitante: ").strip()

fecha = input("Fecha del partido (DD-MM-YYYY): ").strip()

hora = input("Hora del partido (HH:MM, formato 24h): ").strip()

cuota\_local = float(input("Cuota Local: ").strip())

cuota\_empate = float(input("Cuota Empate: ").strip())

try:

fecha\_dt = datetime.strptime(fecha, "%d-%m-%Y").date()

if not (fecha\_inicio <= fecha\_dt <= fecha\_fin):

print("⚠️ La fecha no está dentro del rango especificado. Intenta otra.")

continue

except ValueError:

print("⚠️ Formato de fecha inválido. Usa DD-MM-YYYY.")

continue

# Calcular jornada según la hora

hora\_dt = datetime.strptime(hora, "%H:%M")

if 6 <= hora\_dt.hour < 12:

jornada = "Mañana"

elif 12 <= hora\_dt.hour < 19:

jornada = "Tarde"

else:

jornada = "Noche"

partidos.append({

"Fecha": fecha\_dt.strftime("%d %b %Y"),

"Hora": hora,

"Jornada": jornada,

"PAIS": PAIS,

"LIGA": LIGA,

"Equipo Local": equipo\_local,

"Equipo Visitante": equipo\_visitante,

"Cuota Local": cuota\_local,

"Cuota Empate": cuota\_empate

})

# Crear DataFrame

df = pd.DataFrame(partidos)

if not df.empty:

archivo\_nombre = f"partidos\_{LIGA.replace(' ', '\_')}\_{FECHA1}\_a\_{FECHA2}.xlsx"

df.to\_excel(archivo\_nombre, index=False)

print("\n✅ Partidos registrados:")

print(df.to\_string(index=False))

print(f"\n📁 Archivo Excel generado: {archivo\_nombre}")

else:

print("⚠️ No se ingresaron partidos. No se creó el archivo.")

ULTIMA VERSION CON HTM PERO EXCEL DATOS ERRADOS

html\_code = """

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Generador de Partidos</title>

<style>

body { font-family: Arial, sans-serif; margin: 20px; }

label { display: block; margin-top: 10px; }

table { width: 100%; margin-top: 20px; border-collapse: collapse; }

th, td { padding: 8px; border: 1px solid #ccc; text-align: center; }

button { margin-top: 15px; padding: 10px 20px; font-size: 16px; }

</style>

<script src="https://cdnjs.cloudflare.com/ajax/libs/xlsx/0.18.5/xlsx.full.min.js"></script>

</head>

<body>

<h2>Generador de Partidos</h2>

<label>País: <input type="text" id="pais"></label>

<label>Liga: <input type="text" id="liga"></label>

<label>Fecha Inicio (DD-MM-YYYY): <input type="text" id="fechaInicio"></label>

<label>Fecha Fin (DD-MM-YYYY): <input type="text" id="fechaFin"></label>

<button onclick="generarPartidos()">Generar Partidos</button>

<button onclick="exportarExcel()">Exportar a Excel</button>

<table id="tablaPartidos" style="display:none;">

<thead>

<tr>

<th>Fecha</th><th>Hora</th><th>Jornada</th><th>País</th><th>Liga</th>

<th>Equipo Local</th><th>Equipo Visitante</th><th>Cuota Local</th><th>Cuota Empate</th>

</tr>

</thead>

<tbody id="cuerpoTabla"></tbody>

</table>

<script>

function generarPartidos() {

const pais = document.getElementById("pais").value;

const liga = document.getElementById("liga").value;

const fechaInicio = document.getElementById("fechaInicio").value;

const fechaFin = document.getElementById("fechaFin").value;

if (!pais || !liga || !fechaInicio || !fechaFin) {

alert("⚠️ Por favor completa todos los campos.");

return;

}

// Datos actualizados de los partidos

const partidos = [

{ fecha: "07-06-2025", hora: "12:00", local: "Busan I'Park", visitante: "Ansan Greeners", cuotaLocal: 1.90, cuotaEmpate: 3.30 },

{ fecha: "07-06-2025", hora: "15:00", local: "Gyeongnam FC", visitante: "Hwaseong", cuotaLocal: 2.00, cuotaEmpate: 3.10 },

{ fecha: "08-06-2025", hora: "18:00", local: "Gimpo FC", visitante: "Cheonan City", cuotaLocal: 2.10, cuotaEmpate: 3.00 },

{ fecha: "08-06-2025", hora: "20:00", local: "Gyeongnam FC", visitante: "Cheonan City", cuotaLocal: 2.20, cuotaEmpate: 3.00 },

{ fecha: "09-06-2025", hora: "10:00", local: "Ansan Greeners", visitante: "Hwaseong", cuotaLocal: 1.80, cuotaEmpate: 3.40 }

];

const tbody = document.getElementById("cuerpoTabla");

tbody.innerHTML = ""; // Limpiar antes de cargar nuevos

partidos.forEach(p => {

const [h, m] = p.hora.split(":").map(Number);

let jornada = "";

if (h >= 6 && h < 12) jornada = "Mañana";

else if (h >= 12 && h < 19) jornada = "Tarde";

else jornada = "Noche";

const fila = tbody.insertRow();

fila.innerHTML = `

<td>${p.fecha}</td>

<td>${p.hora}</td>

<td>${jornada}</td>

<td>${pais}</td>

<td>${liga}</td>

<td>${p.local}</td>

<td>${p.visitante}</td>

<td>${p.cuotaLocal}</td>

<td>${p.cuotaEmpate}</td>

`;

});

document.getElementById("tablaPartidos").style.display = "table";

}

function exportarExcel() {

const tabla = document.getElementById("tablaPartidos");

if (tabla.style.display === "none") {

alert("⚠️ Primero debes generar los partidos.");

return;

}

const wb = XLSX.utils.table\_to\_book(tabla, {sheet:"Partidos"});

XLSX.writeFile(wb, "partidos\_generados.xlsx");

}

</script>

</body>

</html>

"""

with open("index.html", "w", encoding="utf-8") as f:

f.write(html\_code)

print("Archivo 'index.html' creado correctamente.")

SCRIPT QUE GENERA LAS 924 COMBINACIONES UNA POR UNA Y UN ZIP TAMBIEN y ARREGLADO

import pandas as pd

from itertools import combinations

from reportlab.lib.pagesizes import letter

from reportlab.lib import colors

from reportlab.lib.units import inch

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer

from reportlab.lib.styles import getSampleStyleSheet

import os

import zipfile

# Leer Excel

df = pd.read\_excel("partidos.xlsx")

# Crear carpeta para guardar PDFs

pdf\_folder = "pdf\_combinaciones"

os.makedirs(pdf\_folder, exist\_ok=True)

# Generar todas las combinaciones de 6 partidos

partidos\_indices = df.index.tolist()

combinaciones = list(combinations(partidos\_indices, 6))

print(f"Generando {len(combinaciones)} PDFs...")

# Columnas que se mostrarán en la tabla

columnas = ['FECHA', 'HORA', 'JOR', 'PAIS', 'PARTIDO', 'CUOTA']

for i, combo in enumerate(combinaciones, 1):

pdf\_path = os.path.join(pdf\_folder, f"combinacion\_{i}.pdf")

doc = SimpleDocTemplate(pdf\_path, pagesize=letter)

elementos = []

styles = getSampleStyleSheet()

# Título

titulo = Paragraph(f"<b>Combinación #{i}</b>", styles['Title'])

elementos.append(titulo)

elementos.append(Spacer(1, 12))

# Crear datos para la tabla

data = [columnas] # Encabezados

for idx in combo:

row = df.loc[idx]

data.append([str(row[col]) for col in columnas])

# Crear tabla

tabla = Table(data)

# Estilo de la tabla

estilo = TableStyle([

('BACKGROUND', (0, 0), (-1, 0), colors.yellow), # Fondo amarillo para encabezado

('BACKGROUND', (0, 1), (-1, -1), colors.whitesmoke), # Fondo blanco para filas

('TEXTCOLOR', (0, 0), (-1, 0), colors.black),

('GRID', (0, 0), (-1, -1), 0.5, colors.black), # Bordes finos

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('FONTNAME', (0, 1), (-1, -1), 'Helvetica'),

('FONTSIZE', (0, 0), (-1, -1), 9),

('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),

])

tabla.setStyle(estilo)

elementos.append(tabla)

# Construir PDF

doc.build(elementos)

# Crear archivo ZIP con todos los PDFs

zip\_path = "combinaciones\_partidos.zip"

with zipfile.ZipFile(zip\_path, 'w', compression=zipfile.ZIP\_DEFLATED) as zipf:

for filename in os.listdir(pdf\_folder):

if filename.endswith(".pdf"):

zipf.write(os.path.join(pdf\_folder, filename), arcname=filename)

print(f"Todos los PDFs generados y comprimidos en '{zip\_path}'")

3. **Script en Python para enviar el correo con el PDF adjunto NO SE PROBO**

import os

import boto3

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email.mime.application import MIMEApplication

# Cargar las credenciales de AWS

with open("credentials.txt", "r") as f:

credentials = dict(line.strip().split("=") for line in f)

aws\_access\_key\_id = credentials["aws\_access\_key\_id"]

aws\_secret\_access\_key = credentials["aws\_secret\_access\_key"]

region = credentials["region"]

# Configurar el cliente SES

ses\_client = boto3.client(

"ses",

aws\_access\_key\_id=aws\_access\_key\_id,

aws\_secret\_access\_key=aws\_secret\_access\_key,

region\_name=region

)

# Función para enviar el correo

def send\_email\_with\_attachment(sender, recipient, subject, body, attachment\_path):

# Crear el objeto del mensaje

msg = MIMEMultipart()

msg["From"] = sender

msg["To"] = recipient

msg["Subject"] = subject

# Agregar el cuerpo del mensaje

msg.attach(MIMEText(body, "html"))

# Adjuntar el archivo PDF

with open(attachment\_path, "rb") as f:

part = MIMEApplication(f.read(), Name=os.path.basename(attachment\_path))

part["Content-Disposition"] = f'attachment; filename="{os.path.basename(attachment\_path)}"'

msg.attach(part)

# Enviar el correo

try:

response = ses\_client.send\_raw\_email(

Source=sender,

Destinations=[recipient],

RawMessage={"Data": msg.as\_string()}

)

print(f"Correo enviado con éxito: {response}")

except Exception as e:

print(f"Error al enviar el correo: {e}")

# Parámetros del correo

sender = "tuemail@dominio.com" # Reemplaza con tu correo verificado en SES

recipient = "destinatario@dominio.com" # Reemplaza con el correo del destinatario

subject = "Combinación de apuestas"

body = "<h1>Adjunto encontrarás tu combinación de apuestas en formato PDF.</h1>"

attachment\_path = "combinacion\_1.pdf" # Ruta al archivo PDF generado

# Enviar el correo

send\_email\_with\_attachment(sender, recipient, subject, body, attachment\_path)

PARA ENVIAR POR TELEGRAN

¡Claro! Aquí tienes un script **unificado en Python** que:

* Lee el Excel partidos.xlsx
* Genera la combinación de partidos solicitada (por número de combinación)
* Crea el PDF con esa combinación
* Envía el PDF por Telegram a un chat (número o grupo) usando un bot

**Cómo usarlo**

1. Prepara tu archivo partidos.xlsx en la misma carpeta.
2. Crea un bot en Telegram (busca @BotFather en Telegram, crea bot y obtén el **bot token**).
3. Obtén el **chat\_id** del receptor (puedes usar bots o APIs para obtenerlo).
4. Cambia las variables BOT\_TOKEN y CHAT\_ID en el script.
5. Ejecuta el script, te pedirá el número de combinación.
6. Se enviará el PDF por Telegram.

python

CopiarEditar

import pandas as pd

from itertools import combinations

from reportlab.lib.pagesizes import letter

from reportlab.lib import colors

from reportlab.lib.units import inch

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer

from reportlab.lib.styles import getSampleStyleSheet

import os

import requests

# Configura aquí tu BOT token y chat id (usuario o grupo)

BOT\_TOKEN = 'TU\_BOT\_TOKEN\_AQUI'

CHAT\_ID = 'TU\_CHAT\_ID\_AQUI'

# Leer Excel

df = pd.read\_excel("partidos.xlsx")

# Crear carpeta para guardar PDFs

pdf\_folder = "pdf\_combinaciones"

os.makedirs(pdf\_folder, exist\_ok=True)

# Generar todas las combinaciones de 6 partidos

partidos\_indices = df.index.tolist()

combinaciones = list(combinations(partidos\_indices, 6))

# Función para crear el PDF de una combinación específica

def crear\_pdf\_combinacion(numero\_combinacion):

if numero\_combinacion < 1 or numero\_combinacion > len(combinaciones):

raise ValueError(f"El número de combinación debe estar entre 1 y {len(combinaciones)}")

combo = combinaciones[numero\_combinacion - 1]

pdf\_path = os.path.join(pdf\_folder, f"combinacion\_{numero\_combinacion}.pdf")

doc = SimpleDocTemplate(pdf\_path, pagesize=letter)

elementos = []

styles = getSampleStyleSheet()

# Título

titulo = Paragraph(f"<b>Combinación #{numero\_combinacion}</b>", styles['Title'])

elementos.append(titulo)

elementos.append(Spacer(1, 12))

# Columnas que se mostrarán en la tabla

columnas = ['FECHA', 'HORA', 'JOR', 'PAIS', 'PARTIDO', 'CUOTA']

# Crear datos para la tabla

data = [columnas] # encabezados

for idx in combo:

row = df.loc[idx]

data.append([str(row[col]) for col in columnas])

# Crear tabla

tabla = Table(data)

# Estilo de la tabla: toda la tabla fondo amarillo

estilo = TableStyle([

('BACKGROUND', (0, 0), (-1, -1), colors.yellow), # Fondo amarillo en toda la tabla

('TEXTCOLOR', (0, 0), (-1, -1), colors.black),

('GRID', (0, 0), (-1, -1), 0.5, colors.black),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'), # encabezados en negrita

('FONTNAME', (0, 1), (-1, -1), 'Helvetica'), # resto en normal

('FONTSIZE', (0, 0), (-1, -1), 9),

('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),

])

tabla.setStyle(estilo)

elementos.append(tabla)

doc.build(elementos)

return pdf\_path

# Función para enviar PDF por Telegram

def enviar\_pdf\_telegram(pdf\_path):

url = f'https://api.telegram.org/bot{BOT\_TOKEN}/sendDocument'

with open(pdf\_path, 'rb') as f:

files = {'document': f}

data = {'chat\_id': CHAT\_ID}

response = requests.post(url, data=data, files=files)

if response.status\_code == 200:

print(f"PDF enviado correctamente a Telegram.")

else:

print(f"Error enviando PDF: {response.text}")

def main():

print(f"Hay {len(combinaciones)} combinaciones posibles.")

try:

numero = int(input("Ingresa el número de combinación que quieres generar y enviar: "))

pdf\_path = crear\_pdf\_combinacion(numero)

print(f"PDF generado en: {pdf\_path}")

enviar\_pdf\_telegram(pdf\_path)

except Exception as e:

print(f"Error: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

Si quieres, puedo ayudarte a:

* Crear el bot paso a paso
* Obtener el chat\_id
* Adaptar el script para que reciba el chat\_id como parámetro o desde un archivo

¿Quieres?

**Resumen para mañana:**  
— Tienes el script unificado para:  
 • Leer el Excel de partidos  
 • Generar la combinación de partidos que elijas  
 • Crear el PDF con fondo amarillo en toda la tabla  
 • Enviar el PDF vía Telegram usando un bot (requiere BOT\_TOKEN y CHAT\_ID)

**Lo siguiente sería:**  
✔️ Crear el bot en Telegram y obtener el token  
✔️ Obtener el chat\_id del destinatario  
✔️ Probar el envío automático del PDF por Telegram desde el script

Cuando quieras, aquí estoy para ayudarte con esos pasos o cualquier duda que tengas. ¡Nos vemos mañana! 👋😊

Principio del formulario

Final del formulario

ULTIMA VERSION DONDE PIDE UNA COMBINACION Y GENERA EL PDF

import os

import webbrowser

import pandas as pd

from itertools import combinations

from flask import Flask, request, send\_file

from reportlab.lib.pagesizes import letter

from reportlab.lib import colors

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer

from reportlab.lib.styles import getSampleStyleSheet

from threading import Timer

# ========== FLASK APP ==========

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

# ========== CONFIGURACIÓN INICIAL ==========

# Leer archivo Excel

df = pd.read\_excel("partidos.xlsx")

# Generar todas las combinaciones posibles de 6 partidos

partidos\_indices = df.index.tolist()

combinaciones = list(combinations(partidos\_indices, 6))

# Crear carpeta para PDFs si no existe

pdf\_folder = "pdf\_combinaciones"

os.makedirs(pdf\_folder, exist\_ok=True)

# ========== HTML EMBEBIDO ==========

html\_form = """

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Generar Combinación</title>

</head>

<body>

<h2>Generar PDF de Combinación</h2>

<form action="/generar" method="post">

<label for="combo\_num">Número de Combinación (1-924):</label>

<input type="number" id="combo\_num" name="combo\_num" min="1" max="924" required>

<br><br>

<button type="submit">Generar Combinación</button>

</form>

</body>

</html>

"""

# ========== RUTAS FLASK ==========

@app.route('/')

def index():

return html\_form

@app.route('/generar', methods=['POST'])

def generar\_pdf():

try:

combo\_num = int(request.form['combo\_num'])

except ValueError:

return "Número inválido. Debe ser un número entre 1 y 924."

if not (1 <= combo\_num <= len(combinaciones)):

return f"El número debe estar entre 1 y {len(combinaciones)}."

combo = combinaciones[combo\_num - 1]

pdf\_path = os.path.join(pdf\_folder, f"combinacion\_{combo\_num}.pdf")

if not os.path.exists(pdf\_path):

columnas = ['FECHA', 'HORA', 'JOR', 'PAIS', 'PARTIDO', 'CUOTA']

doc = SimpleDocTemplate(pdf\_path, pagesize=letter)

elementos = []

styles = getSampleStyleSheet()

# Título

elementos.append(Paragraph(f"<b>Combinación #{combo\_num}</b>", styles['Title']))

elementos.append(Spacer(1, 12))

# Tabla de partidos

data = [columnas]

for idx in combo:

row = df.loc[idx]

data.append([str(row[col]) for col in columnas])

tabla = Table(data)

tabla.setStyle(TableStyle([

('BACKGROUND', (0, 0), (-1, 0), colors.yellow),

('BACKGROUND', (0, 1), (-1, -1), colors.whitesmoke),

('TEXTCOLOR', (0, 0), (-1, 0), colors.black),

('GRID', (0, 0), (-1, -1), 0.5, colors.black),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('FONTNAME', (0, 1), (-1, -1), 'Helvetica'),

('FONTSIZE', (0, 0), (-1, -1), 9),

('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),

]))

elementos.append(tabla)

doc.build(elementos)

return send\_file(pdf\_path, as\_attachment=True)

# ========== ABRIR NAVEGADOR AUTOMÁTICAMENTE ==========

def abrir\_navegador():

webbrowser.open\_new("http://localhost:5000")

# ========== EJECUTAR APP ==========

if \_\_name\_\_ == "\_\_main\_\_":

print("Servidor iniciándose... Espere un momento.")

Timer(1, abrir\_navegador).start() # Abrir navegador tras 1 segundo

app.run(debug=False, port=5000)

LLAVES WOMPI

URL de Eventos : https://wompi-backend-zodk.onrender.com/webhook

**Llave pública**

pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG

**Llave pública**

pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG

**Llave privada**

prv\_test\_NTN6kv4XuC5i7Y3bWFRHVlQkBNRhIqvc

**Llave privada**

prv\_test\_NTN6kv4XuC5i7Y3bWFRHVlQkBNRhIqvc

**SECRETO DE INTEGRIDAD EN WOMPI SE ENCUENTRA EN DESARROLLO PROGRAMADORES**

**Integridad**

prod\_integrity\_vGzFQ729AV62mNMXFFtUQDRVUHOJ5kvy

**Integridad**

test\_integrity\_CxvWC5XqHUC8eownzkyCMKjfOujwVmqk

https://eorlzwmbuwv5oh9.m.pipedream.net

Ahora seguimos con el Paso 2: Verificar que la firma se genere correctamente

Vamos a modificar tu HTML para que:

Genere la firma SHA-256.

La muestre en la consola y en la pantalla.

Así validamos que la firma esté saliendo correctamente antes de iniciar el widget.

c677a9d4e4327d27b284e14b66212e5bbc22c297ebab5e5f74931d7291b151fe

d0c9f4c9cf9be7fcbac287b2891759856e18bd258964a153a5182deb44a87976

ngron

2y6VjbFzZgk9lTNQ598TUeB8WI2\_7LqfRuNK2QWdQRjzvooU1

ngrok config add-authtoken 2y6VjbFzZgk9lTNQ598TUeB8WI2\_7LqfRuNK2QWdQRjzvooU1

ngrok http 5000

url correcta:

Forwarding https://78b4-179-32-115-122.ngrok-free.app -> <http://localhost:5000>

redirectUrl: "https://78b4-179-32-115-122.ngrok-free.app/gracias"

<https://6518-179-32-115-122.ngrok-free.app/boton_wompi.html>

boton\_wompi.html

VERSION CON WOMPI PERO NO SALE TRANSACCION

import os

import webbrowser

import pandas as pd

from itertools import combinations

from flask import Flask, request, send\_file

from reportlab.lib.pagesizes import letter

from reportlab.lib import colors

from reportlab.platypus import SimpleDocTemplate, Table, TableStyle, Paragraph, Spacer

from reportlab.lib.styles import getSampleStyleSheet

from threading import Timer

import requests

# ==================== TU CONFIGURACIÓN ====================

# 💡 Coloca aquí tus claves reales de Wompi Sandbox

WOMPI\_PUBLIC\_KEY = "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG"

WOMPI\_PRIVATE\_KEY = "prv\_test\_NTN6kv4XuC5i7Y3bWFRHVlQkBNRhIqvc"

# 💡 Coloca aquí la URL que te da ngrok cada vez que lo inicias

NGROK\_URL = "https://711f-179-32-115-122.ngrok-free.app -> http://localhost:5000"

# ==========================================================

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

# ========== CARGAR DATOS DEL EXCEL ==========

df = pd.read\_excel("partidos.xlsx")

partidos\_indices = df.index.tolist()

combinaciones = list(combinations(partidos\_indices, 6))

pdf\_folder = "pdf\_combinaciones"

os.makedirs(pdf\_folder, exist\_ok=True)

# ========== FORMULARIO HTML ==========

html\_form = f"""

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Generar Combinación</title>

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

</head>

<body>

<h2>1. Realizar pago con Nequi vía Wompi</h2>

<form>

<script

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

data-public-key="{WOMPI\_PUBLIC\_KEY}"

data-currency="COP"

data-amount-in-cents="200000"

data-reference="pago-combinacion-001"

data-redirect-url="{NGROK\_URL}/verificar-pago">

</script>

</form>

<hr>

<h2>2. Generar PDF de Combinación</h2>

<form action="/generar" method="post">

<label for="combo\_num">Número de Combinación (1-924):</label>

<input type="number" id="combo\_num" name="combo\_num" min="1" max="924" required>

<br><br>

<button type="submit">Generar Combinación</button>

</form>

</body>

</html>

"""

# ========== RUTAS FLASK ==========

@app.route('/')

def index():

return html\_form

@app.route('/generar', methods=['POST'])

def generar\_pdf():

try:

combo\_num = int(request.form['combo\_num'])

except ValueError:

return "Número inválido. Debe ser un número entre 1 y 924."

if not (1 <= combo\_num <= len(combinaciones)):

return f"El número debe estar entre 1 y {len(combinaciones)}."

combo = combinaciones[combo\_num - 1]

pdf\_path = os.path.join(pdf\_folder, f"combinacion\_{combo\_num}.pdf")

if not os.path.exists(pdf\_path):

columnas = ['FECHA', 'HORA', 'JOR', 'PAIS', 'PARTIDO', 'CUOTA']

doc = SimpleDocTemplate(pdf\_path, pagesize=letter)

elementos = []

styles = getSampleStyleSheet()

elementos.append(Paragraph(f"<b>Combinación #{combo\_num}</b>", styles['Title']))

elementos.append(Spacer(1, 12))

data = [columnas]

for idx in combo:

row = df.loc[idx]

data.append([str(row[col]) for col in columnas])

tabla = Table(data)

tabla.setStyle(TableStyle([

('BACKGROUND', (0, 0), (-1, 0), colors.yellow),

('BACKGROUND', (0, 1), (-1, -1), colors.whitesmoke),

('TEXTCOLOR', (0, 0), (-1, 0), colors.black),

('GRID', (0, 0), (-1, -1), 0.5, colors.black),

('ALIGN', (0, 0), (-1, -1), 'CENTER'),

('FONTNAME', (0, 0), (-1, 0), 'Helvetica-Bold'),

('FONTNAME', (0, 1), (-1, -1), 'Helvetica'),

('FONTSIZE', (0, 0), (-1, -1), 9),

('VALIGN', (0, 0), (-1, -1), 'MIDDLE'),

]))

elementos.append(tabla)

doc.build(elementos)

return send\_file(pdf\_path, as\_attachment=True)

@app.route('/verificar-pago')

def verificar\_pago():

transaction\_id = request.args.get("id")

if not transaction\_id:

return "<h3>Error: No se recibió el ID de transacción.</h3>"

headers = {

"Authorization": f"Bearer {WOMPI\_PRIVATE\_KEY}"

}

try:

response = requests.get(f"https://sandbox.wompi.co/v1/transactions/{transaction\_id}", headers=headers)

data = response.json()

estado = data["data"]["status"]

if estado == "APPROVED":

return """

<h3>✅ ¡Pago aprobado exitosamente!</h3>

<a href="/">Volver y generar combinación</a>

"""

else:

return f"<h3>❌ El pago no fue aprobado. Estado: {estado}</h3>"

except Exception as e:

return f"<h3>Error verificando pago: {e}</h3>"

# ========== EJECUTAR APP ==========

def abrir\_navegador():

webbrowser.open\_new("http://localhost:5000")

if \_\_name\_\_ == "\_\_main\_\_":

print("Servidor Flask iniciando en localhost:5000...")

Timer(1, abrir\_navegador).start()

app.run(debug=False, port=5000)

ULTIMA VERSION

**✅ Solución final y validada (junio 2025):**

* **Usa solo redirectUrl**
* **No uses onSuccess, onResponse, ni responseUrl**
* Debes tener una URL accesible (ngrok o dominio)

**✅ HTML actualizado y funcional (probado):**

html

CopiarEditar

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<title>Pago con Wompi</title>

</head>

<body>

<h2>Botón de pago Wompi</h2>

<button id="botonPagoWompi">Pagar con Wompi</button>

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

<script>

console.log("Script cargado y ejecutándose");

const checkout = new WidgetCheckout({

publicKey: "REEMPLAZA\_CON\_TU\_CLAVE\_PUBLICA", // clave Wompi pública (sandbox o prod)

currency: "COP",

amountInCents: 100000, // $1.000 pesos

reference: "referencia-prueba-001",

redirectUrl: "https://78b4-179-32-115-122.ngrok-free.app/gracias", // ✅ obligatorio, debe estar activa

customerData: {

email: "cliente@ejemplo.com"

}

});

document.getElementById("botonPagoWompi").addEventListener("click", () => {

console.log("Botón clickeado");

checkout.open();

});

</script>

</body>

</html>

**✅ Verifica lo siguiente:**

1. **ngrok esté corriendo**:

bash

CopiarEditar

ngrok http 5000

Y que la URL que ves (por ejemplo https://78b4-...) sea la que colocaste en redirectUrl.

1. El archivo boton\_wompi.html debe estar servido con:

bash

CopiarEditar

python -m http.server 5000

1. Accede desde navegador:

arduino

CopiarEditar

https://TU\_URL\_DE\_NGROK/boton\_wompi.html

1. **Debe abrir el modal de Wompi** al hacer clic en el botón.

**¿Quieres que te genere también una página de destino gracias.html?**

Puedo crearla para que se vea algo como:

✅ Gracias por tu pago. Revisa tu correo para más detalles.

¿Probamos eso como siguiente paso?

**Dijiste:**

por lo momentos dejamos hasta aqui continuamos mañana voy a dejar computador prendido para no perder el hilo...

**ChatGPT dijo:**

Perfecto, descansá tranquilo — lo dejamos todo listo para retomar mañana sin perder el hilo.  
Cuando quieras seguimos desde acá y terminamos la integración Wompi sin problemas.

¡Que tengas un buen descanso o almuerzo! 🌙💻😊  
Aquí estaré cuando vuelvas.

ULTIMA VERSION FUNCIONANDO CON WOMPI CON MONTO MINIMO DE $ 1500

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<title>Pago con Wompi</title>

</head>

<body>

<h2>Botón de pago Wompi</h2>

<button id="botonPagoWompi">Pagar con Wompi</button>

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

<script>

console.log("Script cargado y ejecutándose");

const checkout = new WidgetCheckout({

publicKey: "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG", // ← Tu clave sandbox

currency: "COP",

amountInCents: 150000, // $1.500 COP

reference: "prueba-integracion-001",

redirectUrl: "https://78b4-179-32-115-122.ngrok-free.app/gracias", // ← Redirige tras el pago

customerData: {

email: "cliente@ejemplo.com"

}

});

document.getElementById("botonPagoWompi").addEventListener("click", () => {

console.log("Botón clickeado");

checkout.open(

function (result) {

console.log("✅ Pago exitoso:", result);

},

function (error) {

console.error("❌ Error en el pago:", error);

}

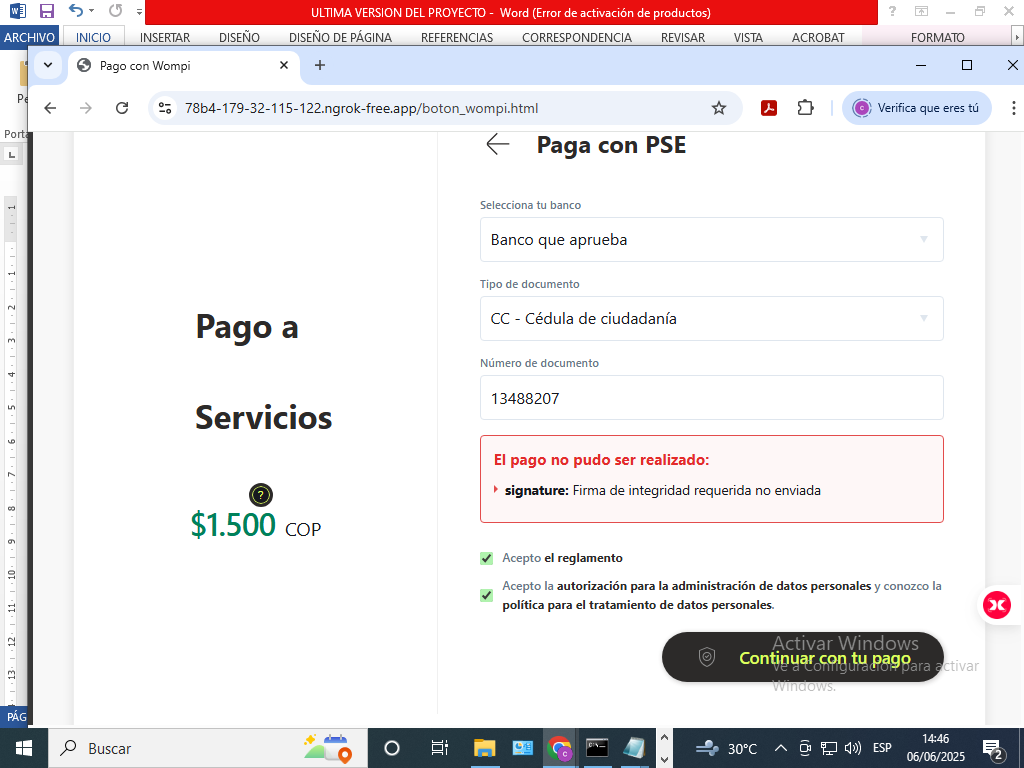
);

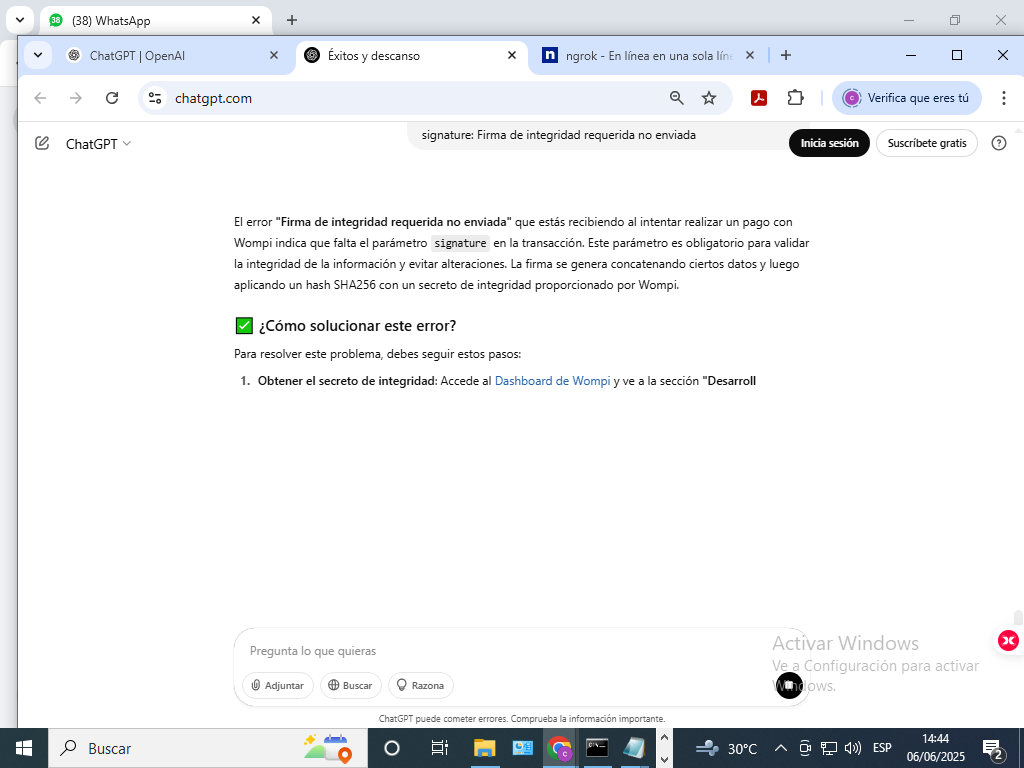
});

</script>

</body>

</html>





LO QUE QUICE DECIR ES CUANDO ENTRO A LA OPCION DE DESARROLLADORES EN LA PAGINA WOMPI NO APARECE NINGUNA OPCION DE secreto de integridad , CUANDO DOY CLICK A LA OPCION DE DESARROLLADORES APARECEN ESTA 4 OPCIONES Y NO SE CUAL ESCOGER Documentación técnica Comunidad Recursos gráficos Status Page

###### ChatGPT dijo:

Gracias por la información adicional. El error **"Firma de integridad requerida no enviada"** que estás recibiendo al intentar realizar un pago con Wompi indica que falta el parámetro signature en la transacción. Este parámetro es obligatorio para validar la integridad de la información y evitar alteraciones. La firma se genera concatenando ciertos datos y luego aplicando un hash SHA256 con un secreto de integridad proporcionado por Wompi.

### ✅ ¿Cómo solucionar este error?

Para resolver este problema, debes seguir estos pasos:

1. **Obtener el secreto de integridad**: Accede al [Dashboard de Wompi](https://www.wompi.co) y ve a la sección **"Desarrolladores > Programadores"**. Allí encontrarás el secreto de integridad, que tiene un formato similar a:

nginx

CopiarEditar

prod\_integrity\_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

1. **Generar la firma de integridad**: Utiliza el siguiente formato para concatenar los datos:

php-template

CopiarEditar

<reference><amount\_in\_cents><currency><expiration-time><secreto\_integridad>

* + <reference>: Referencia única de la transacción (por ejemplo, "prueba-integracion-001").
  + <amount\_in\_cents>: Monto de la transacción en centavos (por ejemplo, 150000 para $1.500 COP).
  + <currency>: Moneda de la transacción (por ejemplo, "COP").
  + <expiration-time>: Fecha y hora de expiración en formato ISO 8601 (por ejemplo, "2025-06-06T15:00:00.000Z").
  + <secreto\_integridad>: El secreto de integridad obtenido en el paso 1.

Concatenando estos valores, obtendrás una cadena como:

makefile

CopiarEditar

prueba-integracion-001150000COP2025-06-06T15:00:00.000Zprod\_integrity\_XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Luego, aplica el algoritmo SHA256 a esta cadena para obtener la firma.

**Nota**: Es altamente recomendable que esta generación de la firma se realice en el **backend** de tu servidor para mantener la seguridad del secreto de integridad.

1. **Incluir la firma en el script**: Una vez generada la firma, inclúyela en el parámetro signature de la siguiente manera:

javascript

CopiarEditar

signature: "valor\_de\_la\_firma\_generada"

Asegúrate de reemplazar "valor\_de\_la\_firma\_generada" con la firma real que obtuviste en el paso anterior.

### ✅ Ejemplo de cómo se vería el script con la firma incluida:

html

CopiarEditar

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<title>Pago con Wompi</title>

</head>

<body>

<h2>Botón de pago Wompi</h2>

<button id="botonPagoWompi">Pagar con Wompi</button>

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

<script>

console.log("Script cargado y ejecutándose");

const checkout = new WidgetCheckout({

publicKey: "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG", // ← Tu clave sandbox

currency: "COP",

amountInCents: 150000, // $1.500 COP

reference: "prueba-integracion-001",

redirectUrl: "https://78b4-179-32-115-122.ngrok-free.app/gracias", // ← Redirige tras el pago

customerData: {

email: "cliente@ejemplo.com"

},

signature: "valor\_de\_la\_firma\_generada" // ← Firma de integridad generada

});

document.getElementById("botonPagoWompi").addEventListener("click", () => {

console.log("Botón clickeado");

checkout.open(

function (result) {

console.log("✅ Pago exitoso:", result);

},

function (error) {

console.error("❌ Error en el pago:", error);

}

);

});

</script>

</body>

</html>

**Nota**: La firma debe generarse dinámicamente en el backend y no debe ser estática en el frontend para mantener la seguridad.

¿Te gustaría que te proporcione un ejemplo de cómo generar la firma de integridad en el backend utilizando un lenguaje específico como Node.js, Python o PHP?

Fuentes

Para proyectos de hobby Gratis $0 / mes 512 MB (RAM) 0,1 CPU Actualice para habilitar más funciones Las instancias gratuitas se desactivan tras periodos de inactividad. No admiten acceso SSH, escalado, trabajos puntuales ni discos persistentes. Seleccione cualquier tipo de instancia de pago para habilitar estas funciones. no sabes cuales son esos periodos de inactiva para evitar que me desactiven?

###### ChatGPT dijo:

¡Muy buena pregunta! Render **no cobra** por el plan gratuito, pero sí tiene una política de **inactividad** para ahorrar recursos. Aquí te explico cómo funciona exactamente:

### 🕒 ¿Qué significa "inactividad" en Render Free (Hobby)?

**Inactividad** = tu aplicación no recibe ninguna visita por un tiempo prolongado.

### ⏳ ¿Cada cuánto se "duerme" una app gratuita?

Render **pone en pausa (sleep mode)** tu app si no recibe tráfico durante aprox. **15 minutos seguidos** de inactividad.

Cuando eso pasa:

* La app entra en "modo suspendido".
* La próxima vez que alguien entre, Render tarda unos **20–30 segundos** en volver a arrancarla.

Esto es **normal** en apps gratuitas, y no afecta la funcionalidad (solo la velocidad del primer acceso tras dormir).

### ✅ ¿Qué puedes hacer para mantenerla "despierta"?

1. **Visitar tu app regularmente** desde el navegador (cada 10 minutos si estás en pruebas).
2. Usar herramientas gratuitas como [UptimeRobot](https://uptimerobot.com/) que hacen un "ping" a tu URL cada cierto tiempo.
3. En producción, mientras usuarios la visiten, **no se dormirá nunca.**

### 📌 Lo importante:

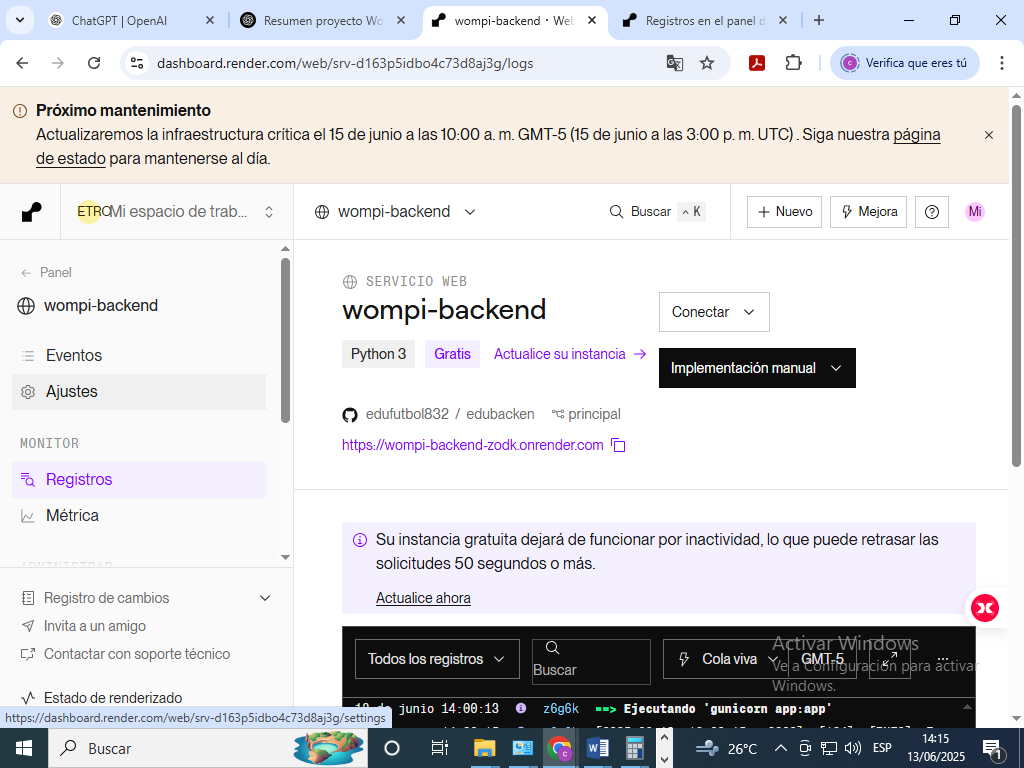
* Tu app **no se elimina**, solo se "duerme".
* Render **no borra tu proyecto** ni te cobra nada por eso.
* La URL sigue activa. Solo tarda un poco en cargar al reactivarse.

#### ✅ Paso 5: Crear servicio

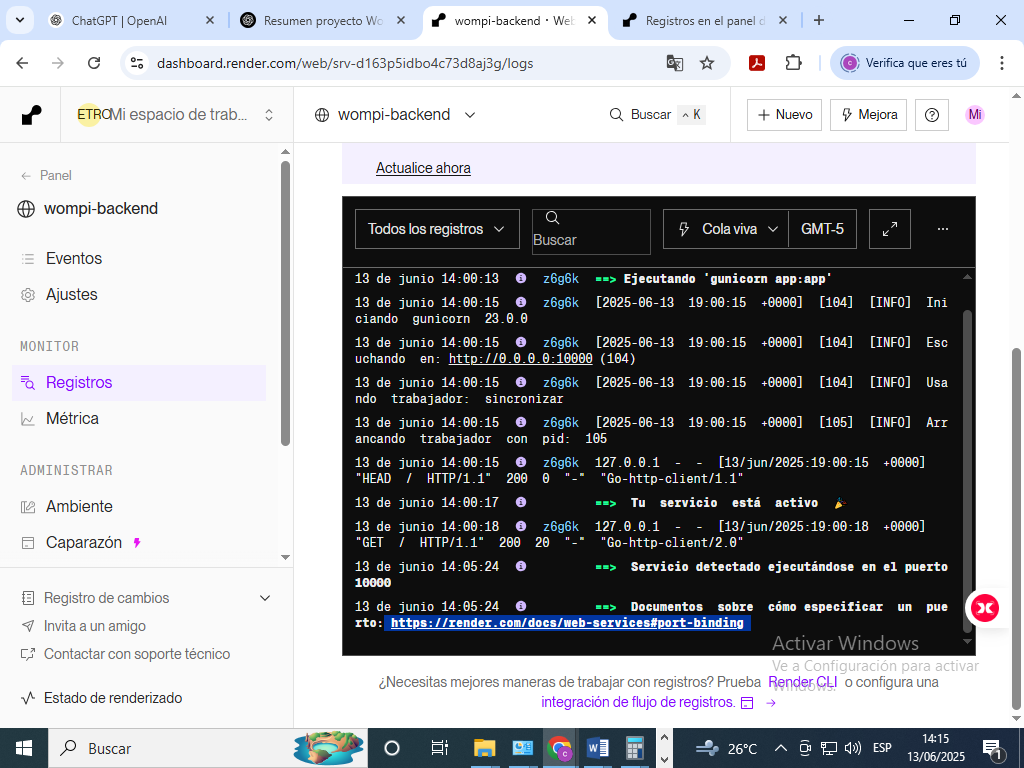
* Revisa todo y haz clic en el botón **"Create Web Service"**

tu nueva URL pública en Render

[https://wompi-backend-zodk.onrender.com](https://wompi-backend-zodk.onrender.com/)



<https://render.com/docs/web-services#port-binding>



### ✅ Conclusión hasta aquí

Tu backend Flask fue **desplegado con éxito** y tu URL pública está funcionando:

🔗 [**https://wompi-backend-zodk.onrender.com**](https://wompi-backend-zodk.onrender.com)

**PARA RECORDAR WOMPI**

**Correo :** [totalfutbol1967@gmail.com](mailto:totalfutbol1967@gmail.com)

**Usuario: pagolista14**

**Clave: Edupagos9**

**Para entrar a wimpo entra con usuario y clave**

Ultimo app.py

from flask import Flask, request, jsonify

from flask\_cors import CORS

import hmac

import hashlib

import base64

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

CORS(app)

# 🛠️ CAMBIA ESTA LÍNEA CON TU CLAVE PRIVADA DE PRUEBAS DE WOMPI (debe comenzar con 'prv\_test\_...')

PRIVATE\_KEY = "prv\_test\_XXXXXXXXXXXXXXXXXXXXXXXXXXXX" # ← 🔁 REEMPLAZA AQUÍ CON TU CLAVE

@app.route("/")

def home():

return "Servidor Flask activo"

@app.route("/webhook", methods=["POST"])

def webhook():

received\_signature = request.headers.get("X-Integrity-Signature")

raw\_body = request.get\_data()

if not received\_signature:

return jsonify({"error": "Firma no enviada"}), 400

computed\_signature = base64.b64encode(

hmac.new(PRIVATE\_KEY.encode(), raw\_body, hashlib.sha256).digest()

).decode()

if hmac.compare\_digest(computed\_signature, received\_signature):

print("✅ Firma válida. Webhook recibido correctamente.")

return jsonify({"message": "Firma válida"}), 200

else:

print("❌ Firma inválida.")

return jsonify({"error": "Firma inválida"}), 400

@app.route("/gracias")

def gracias():

return "<h2>¡Gracias por tu pago!</h2>", 200

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host="0.0.0.0", port=10000)

### 📞 Soporte adicional

Si después de seguir estos pasos el problema persiste, te recomiendo contactar directamente al soporte técnico de Wompi para obtener asistencia personalizada. Puedes hacerlo a través de los siguientes canales:

* **WhatsApp**: +57 322 280 4391 (lunes a viernes, de 8 a.m. a 5 p.m.)
* **Formulario de soporte**: <https://soporte.wompi.co/hc/es-419/requests/new>
* **Chat en vivo**: Disponible en el sitio web oficial de Wompi durante el horario laboral.

Ultimo index html para revisarlo y volver hacer pruebas

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8" />

<title>Pago con Wompi</title>

</head>

<body>

<h2>Botón de pago Wompi</h2>

<button id="botonPagoWompi">Pagar con Wompi</button>

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

<script>

console.log("Script cargado y ejecutándose");

// 👉 1. Llave pública de PRUEBAS (desde el panel de Programadores)

const publicKey = "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG";

// 👉 2. Datos del pago

const amountInCents = 150000; // $1.500 COP

const currency = "COP";

const reference = "prueba-integracion-001";

// 👉 3. Llave de integridad (copiada de la sección "Integridad")

const integritySecret = "150000COPprueba-integracion-001test\_integrity\_CxvWC5XqHUC8eownzkyCMKjfOujwVmqk"; // ← reemplázala aquí

// 👉 4. Concatenación para la firma

const stringToSign = `${amountInCents}${currency}${reference}${integritySecret}`;

// 👉 5. Generar firma SHA-256

async function generateSignature(message) {

const msgBuffer = new TextEncoder().encode(message);

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

return Array.from(new Uint8Array(hashBuffer))

.map(b => b.toString(16).padStart(2, "0"))

.join("");

}

// 👉 6. Botón de pago

document.getElementById("botonPagoWompi").addEventListener("click", async () => {

const signature = await generateSignature(stringToSign);

console.log("✅ Firma generada:", signature);

const checkout = new WidgetCheckout({

publicKey: publicKey,

currency: currency,

amountInCents: amountInCents,

reference: reference,

redirectUrl: "https://wompi-backend-zodk.onrender.com/gracias", // ← tu backend

signature: {

integrity: signature // ← esto es obligatorio para evitar el error de integridad

},

customerData: {

email: "cliente@ejemplo.com",

fullName: "Juan Pérez",

phoneNumber: "3000000000",

phoneNumberPrefix: "+57"

}

});

checkout.open(

function (result) {

console.log("✅ Pago exitoso:", result);

},

function (error) {

console.error("❌ Error en el pago:", error);

}

);

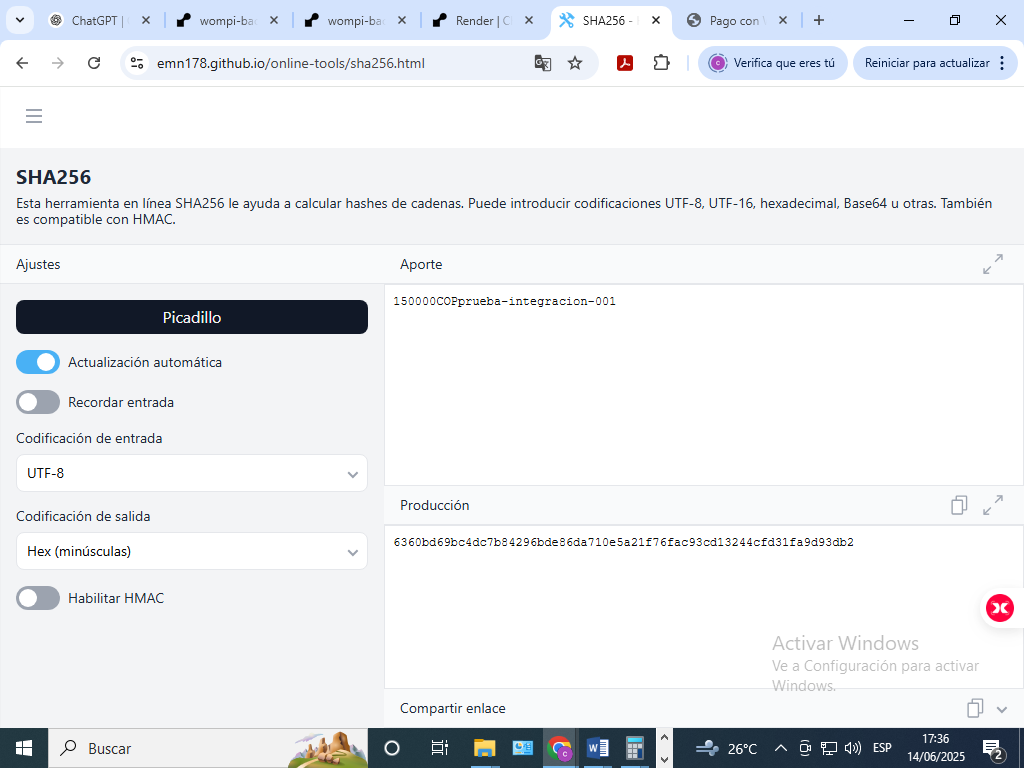
});

</script>

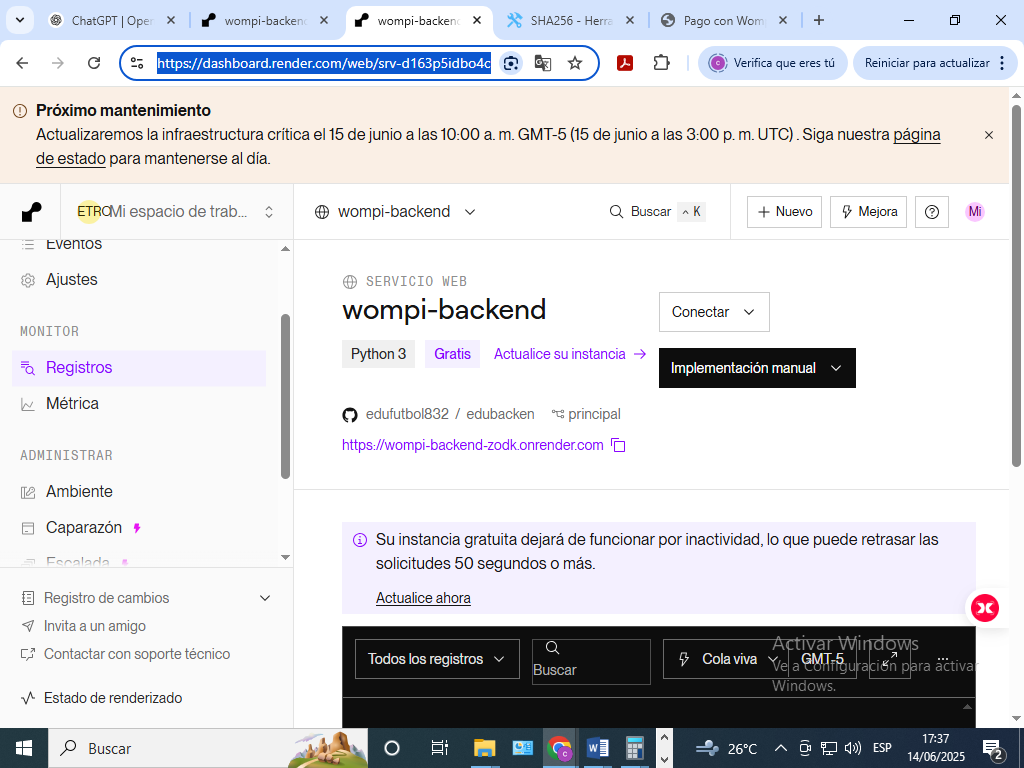
</body>

</html>

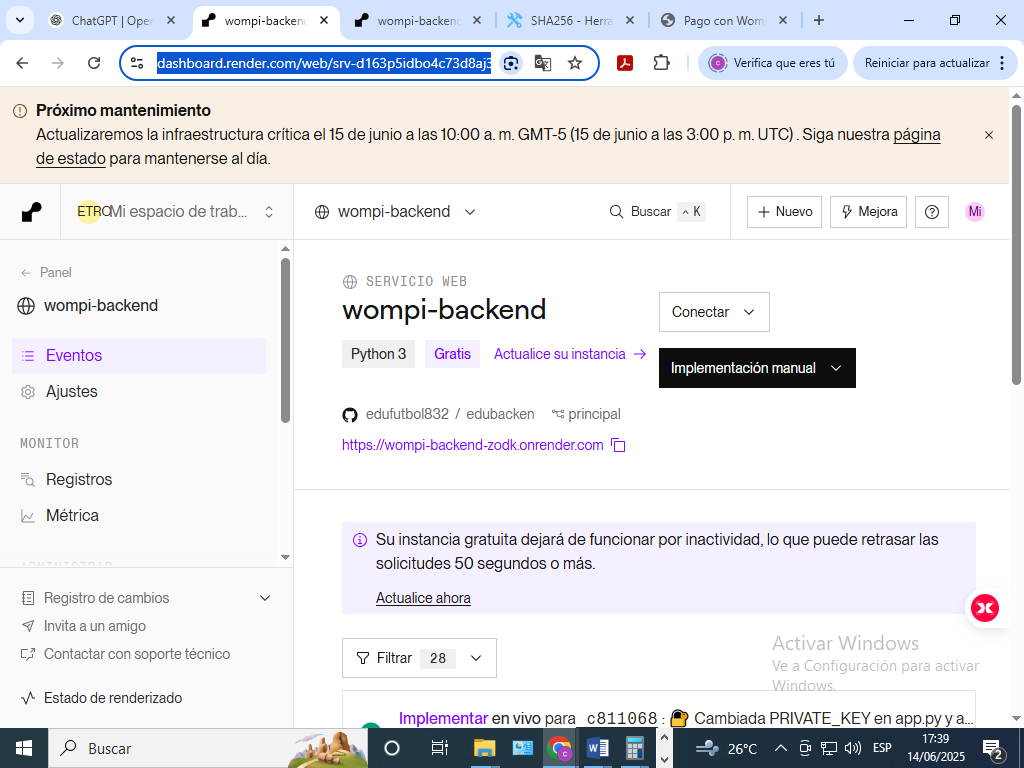
https://emn178.github.io/online-tools/sha256.html



<https://dashboard.render.com/web/srv-d163p5idbo4c73d8aj3g/logs>



<https://dashboard.render.com/web/srv-d163p5idbo4c73d8aj3g/events>



ULTIMO HTML

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Pago con Wompi (Prueba Final)</title>

</head>

<body>

<h2>Pago con Wompi</h2>

<button id="pagar">Pagar con Wompi</button>

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

<script>

const publicKey = "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG";

const amountInCents = 150000;

const currency = "COP";

const reference = "prueba-integracion-001";

// ⚠️ Pega aquí tu clave test\_integrity exacta

const integritySecret = "test\_integrity\_CxvWC5XqHUC8eownzkyCMKjfOujwVmqk";

const stringToSign = `${amountInCents}${currency}${reference}${integritySecret}`;

async function generateSignature(message) {

const msgBuffer = new TextEncoder().encode(message);

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

return Array.from(new Uint8Array(hashBuffer))

.map(b => b.toString(16).padStart(2, "0"))

.join("");

}

document.getElementById("pagar").addEventListener("click", async () => {

const signature = await generateSignature(stringToSign);

console.log("✅ Firma generada:", signature);

const checkout = new WidgetCheckout({

publicKey: publicKey,

currency: currency,

amountInCents: amountInCents,

reference: reference,

redirectUrl: "https://wompi-backend-zodk.onrender.com/gracias",

signature: {

integrity: signature

},

customerData: {

email: "cliente@ejemplo.com",

fullName: "Juan Pérez",

phoneNumber: "3000000000",

phoneNumberPrefix: "+57"

}

});

checkout.open(

(result) => console.log("✅ Pago exitoso:", result),

(error) => console.error("❌ Error en el pago:", error)

);

});

</script>

</body>

</html>

GENERAR FIRMA SHARP

<!DOCTYPE html>

<html lang="es">

<head>

<meta charset="UTF-8">

<title>Pago con Wompi (Prueba Final)</title>

</head>

<body>

<h2>Pago con Wompi</h2>

<button id="pagar">Pagar con Wompi</button>

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

<script>

const publicKey = "pub\_test\_K9UBxtRMrdrjaNBNdpuqoLYxmh8feExG";

const amountInCents = 150000;

const currency = "COP";

const reference = "prueba-integracion-001";

// ⚠️ Pega aquí tu clave test\_integrity exacta

const integritySecret = "test\_integrity\_CxvWC5XqHUC8eownzkyCMKjfOujwVmqk";

const stringToSign = `${amountInCents}${currency}${reference}${integritySecret}`;

async function generateSignature(message) {

const msgBuffer = new TextEncoder().encode(message);

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

return Array.from(new Uint8Array(hashBuffer))

.map(b => b.toString(16).padStart(2, "0"))

.join("");

}

document.getElementById("pagar").addEventListener("click", async () => {

const signature = await generateSignature(stringToSign);

console.log("✅ Firma generada:", signature);

const checkout = new WidgetCheckout({

publicKey: publicKey,

currency: currency,

amountInCents: amountInCents,

reference: reference,

redirectUrl: "https://wompi-backend-zodk.onrender.com/gracias",

signature: {

integrity: signature

},

customerData: {

email: "cliente@ejemplo.com",

fullName: "Juan Pérez",

phoneNumber: "3000000000",

phoneNumberPrefix: "+57"

}

});

checkout.open(

(result) => console.log("✅ Pago exitoso:", result),

(error) => console.error("❌ Error en el pago:", error)

);

});

</script>

</body>

</html>

https://wompi.com/es/co/que-es-wompi