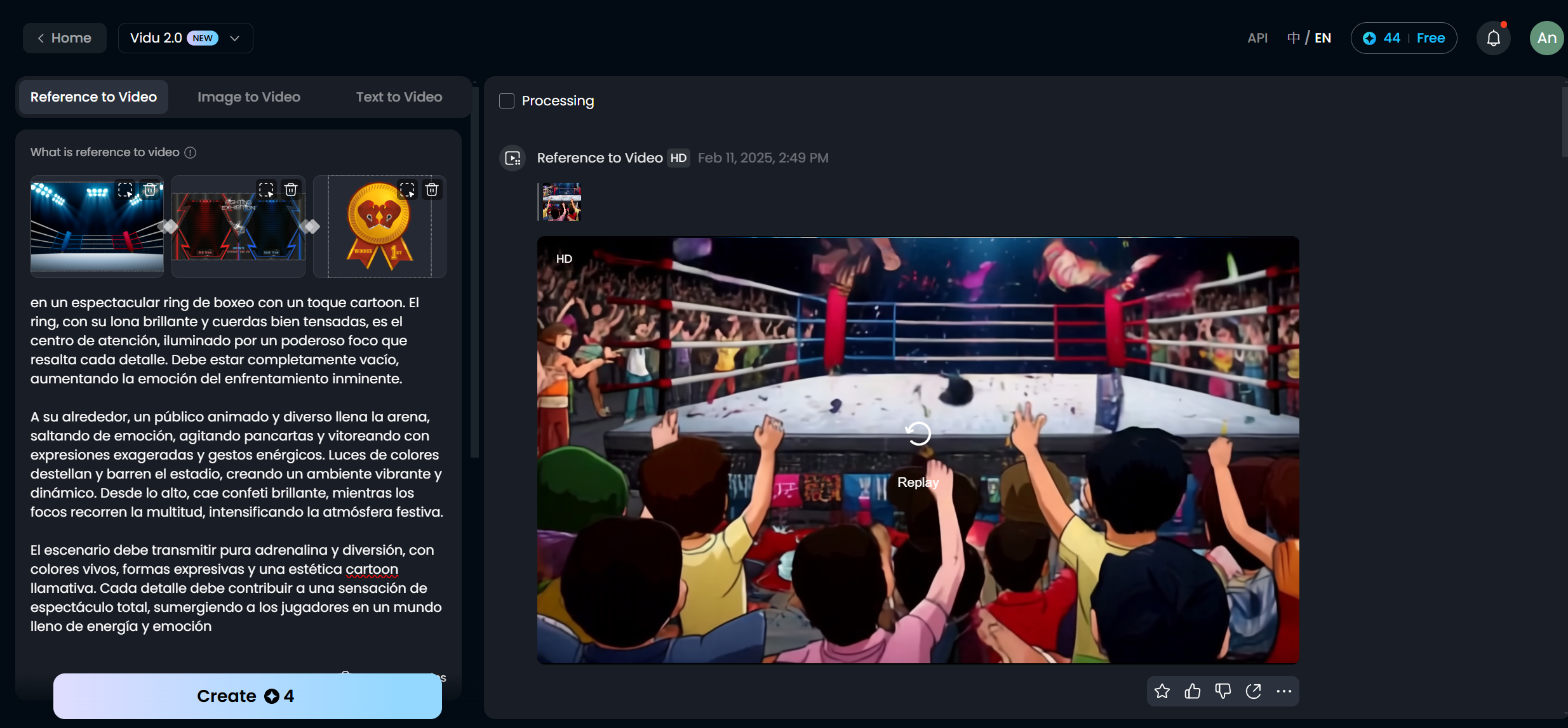
Version 1



**📌 ¿Qué servicio te presta Railway?**

**Railway** es una plataforma de **hosting y despliegue de aplicaciones** en la nube. En otras palabras, te permite alojar tu proyecto para que cualquier persona pueda acceder a él desde cualquier parte del mundo.

**✅ Lo que Railway hace por ti:**

1️⃣ **Toma los archivos que subiste a GitHub** y los usa para desplegar tu aplicación en un servidor en la nube.  
2️⃣ **Genera una URL pública** para que puedas acceder a tu aplicación sin necesidad de configurar servidores manualmente.

* **Ejemplo:** <https://minigame-production.up.railway.app>  
  3️⃣ **Automatiza el despliegue:** Cada vez que subas cambios a GitHub (git push origin main), Railway los detecta y actualiza la aplicación automáticamente.  
  4️⃣ **Proporciona una infraestructura lista para usar:** No necesitas configurar un servidor, ni preocuparte por bases de datos, ni administrar puertos o dominios

**Configuración del Proyecto en tu PC**

✅ **Ubicamos tu proyecto en la ruta:**

* C:\Users\ANGV\Documents\Proyecto\Mini

✅ **Inicializamos Git en el proyecto:**

sh

CopiarEditar

git init

✅ **Configuramos tu usuario y correo en Git:**

sh

CopiarEditar

git config --global user.name "ANGV125936"

git config --global user.email "angv@hotmail.es"

✅ **Creamos el primer commit (guardado de cambios en Git):**

sh

CopiarEditar

git add .

git commit -m "Versión inicial del proyecto"

**2️⃣ Subimos el Proyecto a GitHub**

✅ **Creamos un repositorio en GitHub:**

* <https://github.com/ANGV125936/MiniGame>

✅ **Vinculamos el proyecto local con GitHub:**

sh

CopiarEditar

git remote add origin https://github.com/ANGV125936/MiniGame.git

✅ **Subimos los archivos a GitHub:**

sh

CopiarEditar

git push -u origin main

✅ **Confirmamos que los archivos aparecen en GitHub correctamente.**

**3️⃣ Configuración en Railway**

✅ **Instalamos Railway CLI:**

sh

CopiarEditar

npm install -g @railway/cli

✅ **Nos logueamos en Railway:**

sh

CopiarEditar

railway login

✅ **Creamos un nuevo proyecto en Railway:**

sh

CopiarEditar

railway init

* Elegimos el nombre del proyecto: **MiniGame**
* Se generó el link: [Railway Project](https://railway.com/project/99657f75-f4f3-4387-bb56-4c89dd4ae04e)

✅ **Vinculamos el proyecto de Railway con la carpeta local:**

sh

CopiarEditar

railway link

✅ **Hicimos un primer deploy automático desde Railway:**

* **Railway detectó automáticamente tu código en GitHub**
* **Construyó y desplegó la aplicación en:**  
  👉 [**https://minigame-production.up.railway.app**](https://minigame-production.up.railway.app)

**4️⃣ Validamos el Despliegue**

✅ **Abrimos la URL de Railway:**

* [**https://minigame-production.up.railway.app**](https://minigame-production.up.railway.app)
* ¡Tu aplicación ya está en la web y accesible para cualquier persona! 🎉

**📌 Próximos Pasos**

Ahora que tu proyecto ya está en la web, puedes:

✅ **Hacer cambios en tu código y subirlos a GitHub:**

sh

CopiarEditar

git add .

git commit -m "Descripción del cambio"

git push origin main

**Railway detectará automáticamente los cambios y actualizará la web.**

✅ **Configurar un dominio personalizado (si quieres una URL más profesional).**

✅ **Optimizar la aplicación según lo que necesites.**

## **Creación del Proyecto en Railway**

Railway es la plataforma en la que alojamos nuestro backend para gestionar las partidas del juego.

### ****🔹 Pasos realizados:****

1. **Registro en Railway** y creación de un nuevo proyecto llamado MiniGame.
2. **Instalación de Railway CLI** en Windows con:

sh

CopiarEditar

npm install -g @railway/cli

1. **Inicialización del proyecto en Railway:**

sh

CopiarEditar

railway init

1. **Enlace del proyecto local con Railway:**

sh

CopiarEditar

railway link

1. **Configuración de Git y GitHub** para gestionar el código:

sh

CopiarEditar

git init

git remote add origin https://github.com/ANGV125936/MiniGame.git

1. **Primer despliegue del código en GitHub y Railway**:

sh

CopiarEditar

git add .

git commit -m "Primer commit"

git push origin main

## **2️⃣ Configuración de PostgreSQL en Railway**

Para almacenar el historial de partidas, usamos **PostgreSQL** en Railway.

### ****🔹 Pasos realizados:****

1. **Activamos una base de datos PostgreSQL** en Railway.
2. **Obtuvimos la URL de conexión** en la configuración de Railway.
3. **Configuramos la variable de entorno en Windows:**

sh

CopiarEditar

set DATABASE\_PUBLIC\_URL=postgresql://postgres:contraseña@junction.proxy.rlwy.net:50761/railway

1. **Verificamos que la variable esté correctamente asignada:**

sh

CopiarEditar

echo %DATABASE\_PUBLIC\_URL%

## **3️⃣ Desarrollo del Backend con Flask**

Se creó un **servidor Flask** para manejar las peticiones del juego.

### ****🔹 Instalación de dependencias:****

sh

CopiarEditar

pip install flask psycopg2-binary python-dotenv

### ****🔹 Creación del archivo**** app.py ****con Flask:****

Este código maneja las peticiones de guardar y obtener partidas.

python

CopiarEditar

from flask import Flask, request, jsonify

import psycopg2

from dotenv import load\_dotenv

import os

# Cargar variables de entorno

load\_dotenv()

# Configurar Flask

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

# Conectar a la base de datos PostgreSQL

DATABASE\_URL = os.getenv("DATABASE\_PUBLIC\_URL")

try:

conn = psycopg2.connect(DATABASE\_URL)

cursor = conn.cursor()

print("✅ Conexión a la base de datos exitosa")

except Exception as e:

print(f"❌ Error al conectar a la base de datos: {e}")

exit()

# Ruta para verificar que el servidor está corriendo

@app.route('/')

def home():

return "¡Servidor Flask funcionando correctamente!"

# Ruta para guardar el historial de partidas

@app.route('/guardar\_partida', methods=['POST'])

def guardar\_partida():

data = request.json # Recibir datos en formato JSON

player1 = data.get("player1")

player2 = data.get("player2")

winner = data.get("winner")

date = data.get("date")

if not player1 or not player2 or not winner or not date:

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

try:

cursor.execute("INSERT INTO game\_history (player1, player2, winner, date) VALUES (%s, %s, %s, %s)",

(player1, player2, winner, date))

conn.commit()

return jsonify({"message": "Partida guardada correctamente"}), 201

except Exception as e:

return jsonify({"error": str(e)}), 500

# Ruta para obtener el historial de partidas

@app.route('/historial', methods=['GET'])

def obtener\_historial():

try:

cursor.execute("SELECT \* FROM game\_history ORDER BY id DESC LIMIT 10")

partidas = cursor.fetchall()

return jsonify(partidas), 200

except Exception as e:

return jsonify({"error": str(e)}), 500

# Iniciar la aplicación Flask

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

app.run(debug=True)

### ****🔹 Ejecución del servidor Flask:****

sh

CopiarEditar

python app.py

Salida esperada en consola:

csharp

CopiarEditar

✅ Conexión a la base de datos exitosa

\* Running on http://127.0.0.1:5000

## **4️⃣ Pruebas con Postman y cURL**

Una vez el servidor estaba corriendo, realizamos pruebas de la API.

### ****🔹 Prueba para guardar una partida****

**Usando Postman o cURL:**

sh

CopiarEditar

curl -X POST http://127.0.0.1:5000/guardar\_partida \

-H "Content-Type: application/json" \

-d "{\"player1\": \"Juan\", \"player2\": \"Pedro\", \"winner\": \"Juan\", \"date\": \"2025-02-11 20:00:00\"}"

Salida esperada:

json

CopiarEditar

{

"message": "Partida guardada correctamente"

}

### ****🔹 Prueba para obtener historial de partidas****

sh

CopiarEditar

curl -X GET http://127.0.0.1:5000/historial

Salida esperada:

json

CopiarEditar

[

{

"id": 1,

"player1": "Juan",

"player2": "Pedro",

"winner": "Juan",

"date": "2025-02-11 20:00:00"

}

]

## **5️⃣ Integración con Railway**

Para desplegar la API en Railway:

### ****🔹 Subir los cambios a GitHub****

sh

CopiarEditar

git add .

git commit -m "Integración de Flask con PostgreSQL en Railway"

git push origin main

### ****🔹 Desplegar la API en Railway****

1. En Railway, seleccionamos el proyecto MiniGame.
2. Agregamos la variable DATABASE\_PUBLIC\_URL en **Variables de entorno**.
3. Railway detecta automáticamente el app.py y ejecuta el servidor.

### ****🔹 Prueba en producción****

La API ahora está disponible en:

arduino

CopiarEditar

https://minigame-production.up.railway.app

Para probarla:

sh

CopiarEditar

curl -X GET https://minigame-production.up.railway.app/historial

## **6️⃣ Integración con el juego "Piedra, Papel y Tijera"**

Para conectar el juego con la API, agregamos el siguiente código en **JavaScript** dentro del juego:

javascript

CopiarEditar

function guardarPartida(player1, player2, winner) {

fetch('https://minigame-production.up.railway.app/guardar\_partida', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({

player1: player1,

player2: player2,

winner: winner,

date: new Date().toISOString()

})

})

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

.then(data => console.log('✅ Partida guardada:', data))

.catch(error => console.error('❌ Error:', error));

}

// Llamamos a la función cuando termine una partida

guardarPartida("Juan", "Pedro", "Juan");

## **🎯 Conclusión**

✅ **Se logró:**  
✔ Crear y configurar un servidor Flask.  
✔ Conectar PostgreSQL en Railway.  
✔ Desarrollar una API REST para gestionar partidas.  
✔ Probar la API con Postman y cURL.  
✔ Integrar la API con el juego "Piedra, Papel y Tijera".  
✔ Desplegar en **GitHub** y **Railway** para acceso global.

### ****🚀 Próximos pasos****

📌 **Mejorar la seguridad:** Usar autenticación JWT para evitar accesos no autorizados.  
📌 **Crear una interfaz para ver historial:** En lugar de solo JSON, mostrarlo en HTML.  
📌 **Más juegos:** Conectar otros mini juegos al mismo sistema de almacenamiento.

¡Tu proyecto ya está en la nube! 🎉🚀 ¿Qué más te gustaría agregar o mejorar?

