Evaluación 3

Nombres: Lian Valenzuela

Juan Narria

Instrucciones:

1-Habilite una aplicación web utilizando librería bottle que permita responder las siguientes consultas:

Mensaje POST: Recibir jugada con los siguientes datos

-ID del jugador

-ID del Juego

-Valor de la jugada

Mensaje GET: Publicar el resultado de la jugada en una vista html simple indicando los siguientes datos:

-Nombre de los jugadores

-Valores de las jugadas

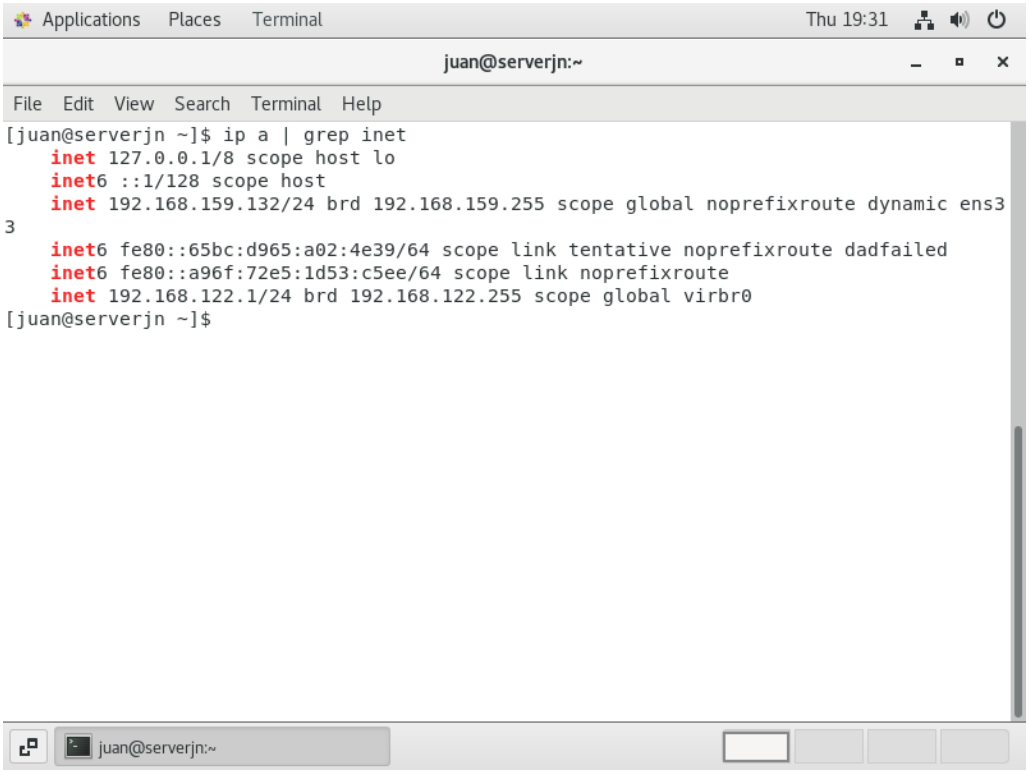
-Jugador Ganador

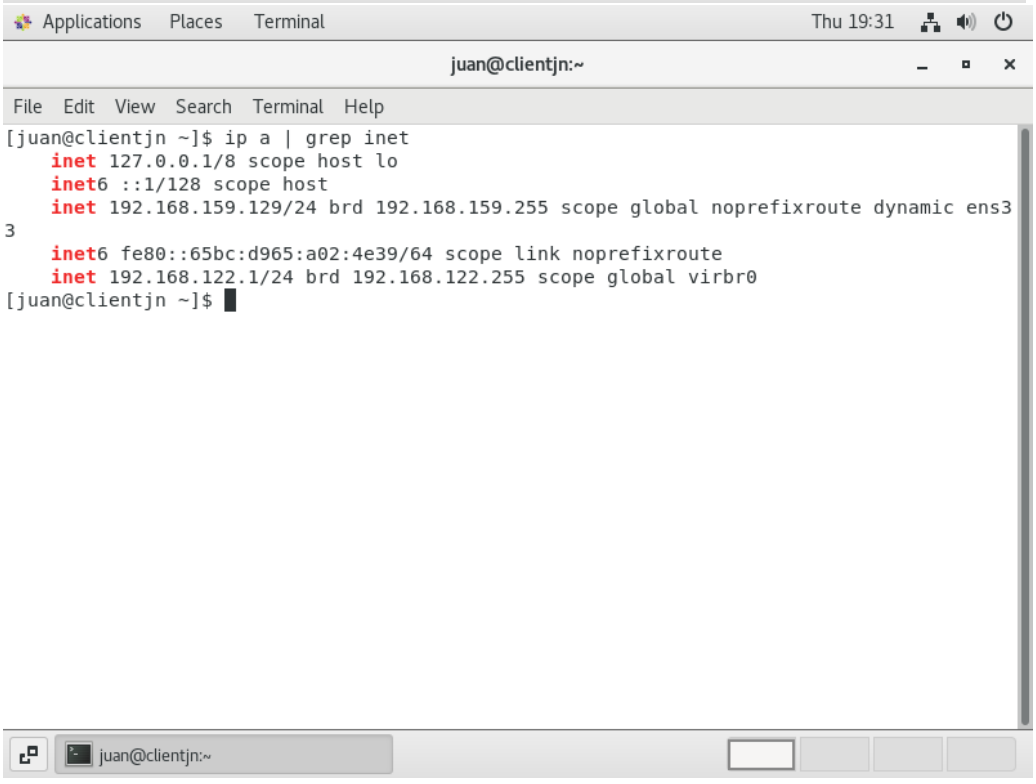
-Puntaje Acumulado

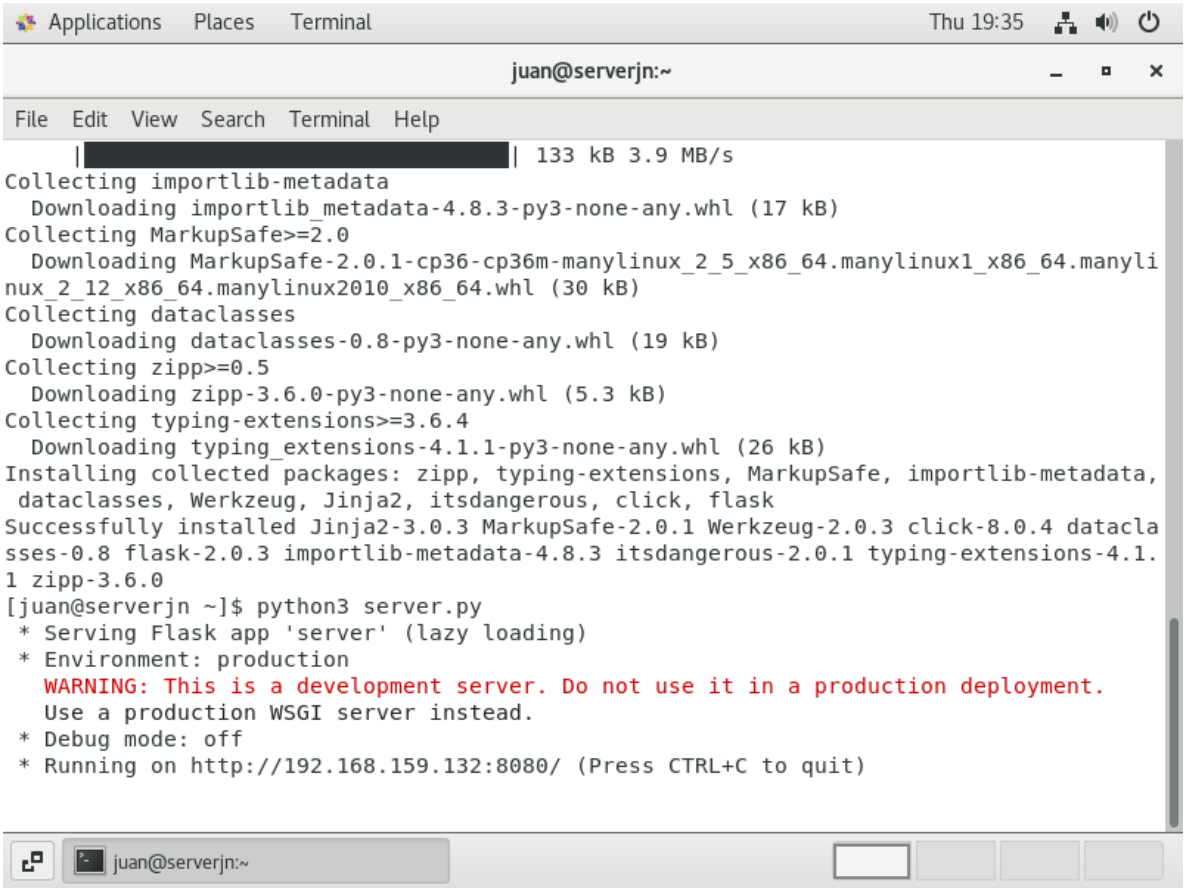
Mensaje GET: Publicar estado del servidor indicando los siguientes valores:

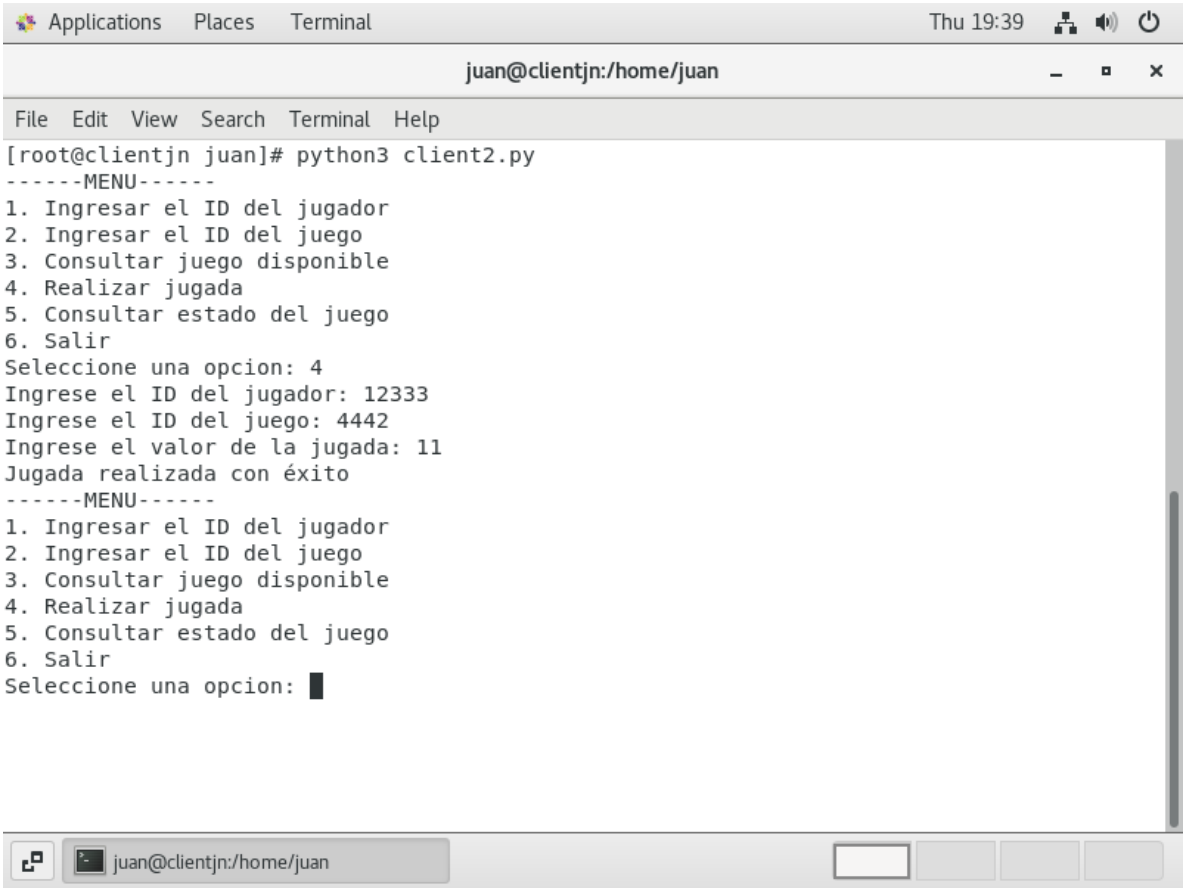
-Estado del servidor

-ID del juego en curso o disponible


Applications Places Terminal Thu 19:31
juan@serverjn:~
File Edit View Search Terminal Help
[juan@serverjn ~]$ ip a | grep inet
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
inet 192.168.159.132/24 brd 192.168.159.255 scope global noprefixroute dynamic ens3
3
inet6 fe80::65bc:d965:a02:4e39/64 scope link tentative noprefixroute dadfailed
inet6 fe80::a96f:72e5:1d53:c5ee/64 scope link noprefixroute
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
[juan@serverjn ~]$
juan@serverjn:~


Applications Places Terminal Thu 19:31
juan@clientjn:~
File Edit View Search Terminal Help
[juan@clientjn ~]$ ip a | grep inet
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
inet 192.168.159.129/24 brd 192.168.159.255 scope global noprefixroute dynamic ens3
3
inet6 fe80::65bc:d965:a02:4e39/64 scope link noprefixroute
inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
[juan@clientjn ~]$
juan@clientjn:~


Applications
Places
Terminal
Thu 19:35
juan@serverjn:~
File
Edit
View
Search
Terminal
Help
| 133 kB 3.9 MB/s
Collecting importlib-metadata
Downloading importlib_metadata-4.8.3-py3-none-any.whl (17 kB)
Collecting MarkupSafe>=2.0
Downloading MarkupSafe-2.0.1-cp36-cp36m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (30 kB)
Collecting dataclasses
Downloading dataclasses-0.8-py3-none-any.whl (19 kB)
Collecting zipp>=0.5
Downloading zipp-3.6.0-py3-none-any.whl (5.3 kB)
Collecting typing-extensions>=3.6.4
Downloading typing_extensions-4.1.1-py3-none-any.whl (26 kB)
Installing collected packages: zipp, typing-extensions, MarkupSafe, importlib-metadata, dataclasses, Werkzeug, Jinja2, itsdangerous, click, flask
Successfully installed Jinja2-3.0.3 MarkupSafe-2.0.1 Werkzeug-2.0.3 click-8.0.4 dataclasses-0.8 flask-2.0.3 importlib-metadata-4.8.3 itsdangerous-2.0.1 typing-extensions-4.1.1 zipp-3.6.0
[juan@serverjn ~]$ python3 server.py
* Serving Flask app 'server' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.159.132:8080/ (Press CTRL+C to quit)
juan@serverjn:~


Applications
Places
Terminal
Thu 19:39
juan@clientjn:/home/juan
File
Edit
View
Search
Terminal
Help
[root@clientjn juan]# python3 client2.py
------MENU------
1. Ingresar el ID del jugador
2. Ingresar el ID del juego
3. Consultar juego disponible
4. Realizar jugada
5. Consultar estado del juego
6. Salir
Seleccione una opcion: 4
Ingrese el ID del jugador: 12333
Ingrese el ID del juego: 4442
Ingrese el valor de la jugada: 11
Jugada realizada con éxito
------MENU------
1. Ingresar el ID del jugador
2. Ingresar el ID del juego
3. Consultar juego disponible
4. Realizar jugada
5. Consultar estado del juego
6. Salir
Seleccione una opcion:
juan@clientjn:/home/juan

2- Utilizando los comandos y librerías de lenguaje de programación Python, desarrolle un script que se comunique con el servidor del juego. Este script que se comunique con el servidor del juego. Este script debe resolver las siguientes tareas:

Habilitar un menú con las siguientes opciones:

-Ingresar ID del jugador

-Ingresar ID del juego

-Consultar juego Disponible

-Realizar jugada

-Consultar resultado del juego

REGLAS GENERALES

Cada jugada se debe escribir en un archivo csv en el directorio linux donde se ejecuta el juego. El valor de cada jugada de ser mediante la selección al azar de un numero entre 1 y 10. Gana el juego, el ID del jugador que genere el mayor valor, el jugador ganador será el mejor de 5 juegos.

Las columnas del archivo csv deben ser:

-ID del jugador

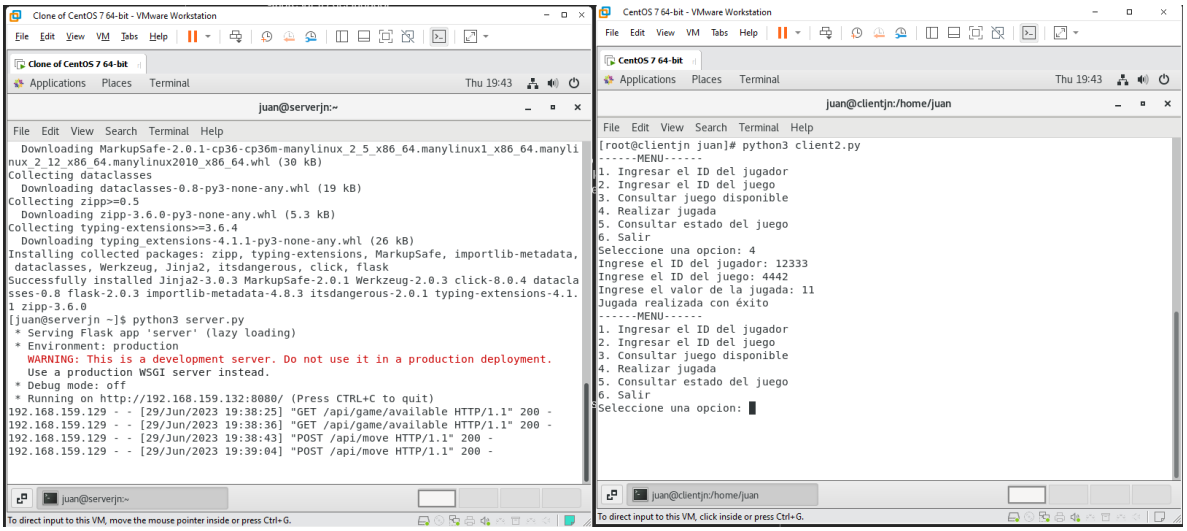
-ID del juego

-Valor de la jugada

-Numero de la jugada

-Jugador Ganador

Funcionamiento entre ambas Maquinas (Comunicación con el servidor) :


Clone of CentOS 7 64-bit - VMware Workstation
File Edit View VM Tabs Help
Clone of CentOS 7 64-bit
Applications Places Terminal
Thu 19:43
juan@serverjn:~
File Edit View Search Terminal Help
Downloading MarkupSafe-2.0.1-cp36-cp36m-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (30 kB)
Collecting dataclasses
Downloading dataclasses-0.8-py3-none-any.whl (19 kB)
Collecting zipp>=0.5
Downloading zipp-3.6.0-py3-none-any.whl (5.3 kB)
Collecting typing-extensions>=3.6.4
Downloading typing_extensions-4.1.1-py3-none-any.whl (26 kB)
Installing collected packages: zipp, typing-extensions, MarkupSafe, importlib-metadata, dataclasses, Werkzeug, Jinja2, itsdangerous, click, flask
Successfully installed Jinja2-3.0.3 MarkupSafe-2.0.1 Werkzeug-2.0.3 click-8.0.4 dataclasses-0.8 flask-2.0.3 importlib-metadata-4.8.3 itsdangerous-2.0.1 typing-extensions-4.1.1 zipp-3.6.0
[juan@serverjn ~]$ python3 server.py
* Serving Flask app 'server' (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://192.168.159.132:8080/ (Press CTRL+C to quit)
192.168.159.129 - - [29/Jun/2023 19:38:25] "GET /api/game/available HTTP/1.1" 200 -
192.168.159.129 - - [29/Jun/2023 19:38:36] "GET /api/game/available HTTP/1.1" 200 -
192.168.159.129 - - [29/Jun/2023 19:38:43] "POST /api/move HTTP/1.1" 200 -
192.168.159.129 - - [29/Jun/2023 19:39:04] "POST /api/move HTTP/1.1" 200 -
juan@serverjn:~
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
CentOS 7 64-bit - VMware Workstation
File Edit View VM Tabs Help
CentOS 7 64-bit
Applications Places Terminal
Thu 19:43
juan@clientjn:/home/juan
File Edit View Search Terminal Help
[root@clientjn juan]# python3 client2.py
------MENU------
1. Ingresar el ID del jugador
2. Ingresar el ID del juego
3. Consultar juego disponible
4. Realizar jugada
5. Consultar estado del juego
6. Salir
Seleccione una opcion: 4
Ingrese el ID del jugador: 12333
Ingrese el ID del juego: 4442
Ingrese el valor de la jugada: 11
Jugada realizada con éxito
------MENU------
1. Ingresar el ID del jugador
2. Ingresar el ID del juego
3. Consultar juego disponible
4. Realizar jugada
5. Consultar estado del juego
6. Salir
Seleccione una opcion:
juan@clientjn:/home/juan
To direct input to this VM, click inside or press Ctrl+G.

##Código Cliente

```
# Importa el módulo requests
## Lian Valenzuela || Juan Narria
import requests

# Define la URL del servidor
SERVER_URL = "http://192.168.159.132:8080"

# Define una función para ingresar un ID
def ingresar_id(prompt):
    id = input(prompt)
    return id

# Define una función para consultar si un juego está disponible
def consultar_juego_disponible():
    response = requests.get(f"{SERVER_URL}/api/game/available")
    data = response.json()
    if data['state'] == "available":
        print("Juego disponible")
    else:
        print("No hay juego disponible en este momento")

# Define una función para realizar una jugada
def realizar_jugada():
    id_jugador = ingresar_id("Ingrese el ID del jugador: ")
    id_juego = ingresar_id("Ingrese el ID del juego: ")
    valor_jugada = input("Ingrese el valor de la jugada: ")

    payload = {
```

```
        "player_id": id_jugador,
        "game_id": id_juego,
        "move_value": valor_jugada
    }

    response = requests.post(f"{SERVER_URL}/api/move", json=payload)
    if response.status_code == 200:
        print("Jugada realizada con éxito")
    else:
        print(f"Error al realizar la jugada, código de estado: {response.status_code}")



## Lian Valenzuela || Juan Narria

# Define una función para consultar el resultado del juego
def consultar_resultado_juego():
    response = requests.get(f"{SERVER_URL}/api/game/result")
    data = response.json()
    print("Nombre de los jugadores:", data["jugadores"])
    print("Valores de las jugadas:", data["valores_jugadas"])
    print("Jugador ganador:", data["jugador_ganador"])
    print("Puntaje acumulado de los jugadores:", data["puntaje_acumulado"])

## Lian Valenzuela || Juan Narria

# Bucle principal para interactuar con el usuario
while True:
    print("------MENU------")
    print("1. Ingresar el ID del jugador")
```

```
print("2. Ingresar el ID del juego")
print("3. Consultar juego disponible")
print("4. Realizar jugada")
print("5. Consultar estado del juego")
print("6. Salir")
opcion = input("Seleccione una opcion: ")

if opcion == '1':
    id_jugador = ingresar_id("Ingrese el ID del jugador: ")
    print("ID del jugador ingresado: ", id_jugador)
elif opcion == '2':
    id_juego = ingresar_id("Ingrese el ID del juego: ")
    print("ID del juego ingresado: ", id_juego)
elif opcion == '3':
    consultar_juego_disponible()
elif opcion == '4':
    realizar_jugada()
elif opcion == '5':
    consultar_resultado_juego()
elif opcion == '6':
    print("Saliendo del programa")
    break
else:
    print("Opción no válida, selecciona una opción válida")
```

##Código Server

Lian Valenzuela || Juan Narria

```
# Importa los módulos necesarios
from flask import Flask, request, jsonify

# Crea una nueva aplicación Flask
app = Flask(__name__)

# Inicializa una lista para almacenar las jugadas
jugadas = []

# Define una ruta para comprobar si un juego está disponible
@app.route('/api/game/available', methods=['GET'])
def game_available():
    # Puedes cambiar esto para reflejar el estado actual del juego
    estado_juego = 'disponible'
    return jsonify({'state': estado_juego, 'game_id': 123})

# Define una ruta para recibir jugadas
@app.route('/api/move', methods=['POST'])
def receive_move():
    # Obtiene los datos enviados con la solicitud
    data = request.json
    # Añade los datos a la lista de jugadas
    jugadas.append(data)
    # Devuelve una respuesta para indicar que la jugada fue recibida
    return jsonify({'message': 'Move received'})
```

Lian Valenzuela || Juan Narria

```
# Define una ruta para obtener los resultados del juego
@app.route('/api/game/result', methods=['GET'])
def get_game_result():
    # Estos son sólo ejemplos, deberías reemplazarlos con los valores reales
    jugadores = ['Jugador 1', 'Jugador 2', 'Jugador 3']
    # Calcula los valores de las jugadas basándose en los datos almacenados
    valores_jugadas = [j['move_value'] for j in jugadas]
    # Encuentra el jugador ganador basándose en los valores de las jugadas
    jugador_ganador = jugadores[valores_jugadas.index(max(valores_jugadas))]
    # Calcula el puntaje acumulado de las jugadas
    puntaje_acumulado = sum(valores_jugadas)

    # Devuelve los resultados del juego como una respuesta JSON
    return jsonify({
        'jugadores': jugadores,
        'valores_jugadas': valores_jugadas,
        'jugador_ganador': jugador_ganador,
        'puntaje_acumulado': puntaje_acumulado
    })
```

Lian Valenzuela || Juan Narria

```
# Ejecuta la aplicación si este script se ejecuta directamente
if __name__ == '__main__':
    app.run(host='192.168.159.132', port=8080)
```

##Codigo github por si se necesita

https://github.com/Heztak/linux3