Universidad Internacional del Ecuador

Escuela de Ingeniería

INGENIERIA EN SISTEMAS DE LA INFORMACION

Aprendizaje Autónomo 1.

Christian Eduardo Pintado Lojano

Docente: Monica Patricia Salazar Tapia

Quito, 10 de agosto de 2025

I. Inicio del Desarrollo de Software. / Configuración del entorno

1.1 Se genera el Diagrama de flujo para el juego piedra papel o tijera:

1.1.1 Seudo Código

```
INICIO

    Inicializar los puntos de JugadorNo1 en 0
    Inicializar los puntos de JugadorNo2 en 0
    Establecer Juego_Activo como Verdadero

    MIENTRAS Juego_Activo sea Verdadero:

        Solicitar la elección de JugadorNo1 (Piedra, Papel o
Tijera)
        Solicitar la elección de JugadorNo2 (Piedra, Papel o
Tijera)

        SI la elección de alguno de los jugadores no es
válida:
            Mostrar mensaje de error y continuar al
siguiente ciclo (volver a pedir elección)

        SI la elección de ambos jugadores es la misma:
            Mostrar mensaje de "Empate" y continuar al
siguiente ciclo

        SI las elecciones son diferentes:

            Comprobar las combinaciones y asignar puntos al
jugador correspondiente:
                SI JugadorNo1 elige "Piedra" y JugadorNo2
elige "Tijera":
                    JugadorNo1 gana un punto
                SI JugadorNo1 elige "Papel" y JugadorNo2
elige "Piedra":
                    JugadorNo1 gana un punto
                SI JugadorNo1 elige "Tijera" y JugadorNo2
elige "Papel":
                    JugadorNo1 gana un punto
                SI JugadorNo2 elige "Piedra" y JugadorNo1
elige "Tijera":
                    JugadorNo2 gana un punto
                SI JugadorNo2 elige "Papel" y JugadorNo1
elige "Piedra":
                    JugadorNo2 gana un punto
                SI JugadorNo2 elige "Tijera" y JugadorNo1
elige "Papel":
                    JugadorNo2 gana un punto
```

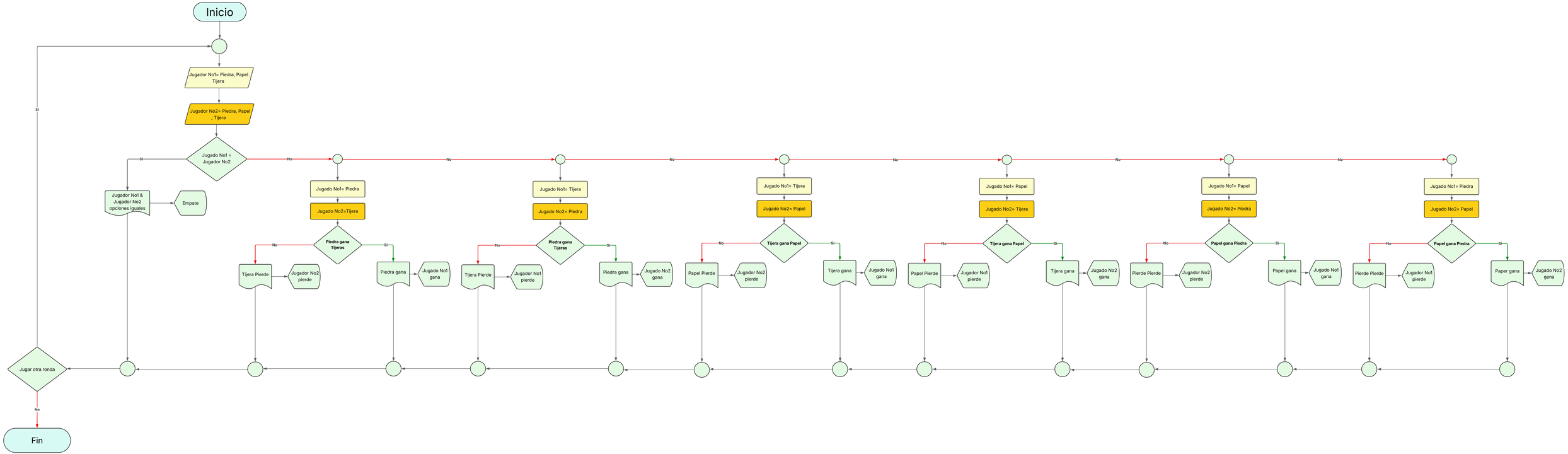
```
            Mostrar los puntajes actuales de ambos jugadores

            Preguntar si desean jugar otra vez (si/no):
                SI la respuesta es "no":
                    Establecer Juego_Activo como Falso

        Mostrar mensaje de "Juego Terminado"
        Mostrar el puntaje final de ambos jugadores

FIN
```

1.1.2 Secuencia de pasos a seguir

II.Secuencia de Pasos del Juego:

1. **Inicio del Juego**
    - Inicializar los puntos de ambos jugadores a 0.
    - Establecer la variable Juego_Activo como True para iniciar el bucle.
2. **Solicitar Elección de los Jugadores**
    - Pedir al Jugador No 1 que elija entre "Piedra", "Papel" o "Tijera".
    - Pedir al Jugador No 2 que elija entre "Piedra", "Papel" o "Tijera".
3. **Validar Elecciones**
    - Comprobar si las elecciones de los jugadores son válidas (deben ser "piedra", "papel" o "tijera").
    - Si alguna elección es inválida, mostrar un mensaje de error y volver a pedir las elecciones.
4. **Comparar las Elecciones**
    - Si las elecciones son iguales, declarar un empate y volver al paso 2.
    - Si las elecciones son diferentes, comparar los valores de las elecciones para determinar al ganador.
5. **Asignar Puntos**
    - Según las combinaciones de las elecciones, asignar puntos:
        - **Piedra** vence a **Tijera** (Punto para el Jugador 1).
        - **Papel** vence a **Piedra** (Punto para el Jugador 1).
        - **Tijera** vence a **Papel** (Punto para el Jugador 1).
        - **Piedra** vence a **Tijera** (Punto para el Jugador 2).
        - **Papel** vence a **Piedra** (Punto para el Jugador 2).
        - **Tijera** vence a **Papel** (Punto para el Jugador 2).
6. **Mostrar Puntajes**
    - Mostrar el puntaje actual de ambos jugadores.
7. **Preguntar si Quieren Jugar Otra Vez**
    - Preguntar a los jugadores si quieren jugar otra vez.
    - Si la respuesta es "sí", continuar desde el paso 2.
    - Si la respuesta es "no", terminar el juego.
8. **Finalizar Juego**
    - Cuando el jugador decida no continuar, finalizar el juego.
    - Mostrar el puntaje final de ambos jugadores.
9. **Fin del Juego**
    - El juego termina, y el puntaje final es mostrado.
    -

2.1 Diagrama de flujo


Inicio
Jugador No1= Piedra, Papel , Tijera
Jugador No2= Piedra, Papel , Tijera
Jugado No1 = Jugador No2
Si
No
Jugador No1 & Jugador No2 opciones iguales
Empate
Jugado No1= Piedra
Jugado No2=Tijera
Piedra gana Tijeras
Tijera Pierde
Jugador No2 pierde
Piedra gana
Jugado No1 gana
Jugado No1= Tijera
Jugado No2= Piedra
Piedra gana Tijeras
Tijera Pierde
Jugador No1 pierde
Piedra gana
Jugado No2 gana
Jugado No1= Tijera
Jugado No2= Papel
Tijera gana Papel
Papel Pierde
Jugador No2 pierde
Tijera gana
Jugado No1 gana
Jugado No1= Papel
Jugado No2= Tijera
Tijera gana Papel
Papel Pierde
Jugador No1 pierde
Tijera gana
Jugado No2 gana
Jugado No1= Papel
Jugado No2= Piedra
Papel gana Piedra
Pierde Pierde
Jugador No2 pierde
Papel gana
Jugado No1 gana
Jugado No1= Piedra
Jugado No2= Papel
Papel gana Piedra
Pierde Pierde
Jugador No1 pierde
Paper gana
Jugado No2 gana
Jugar otra ronda
Si
No
Fin

2.2 Repositorio en Github.

Un repositorio en GitHub es esencial para el control de versiones, la colaboración, la accesibilidad, la visibilidad y la integración con otras herramientas, lo que facilita el desarrollo y aprendizaje continuo destaco las siguientes características.

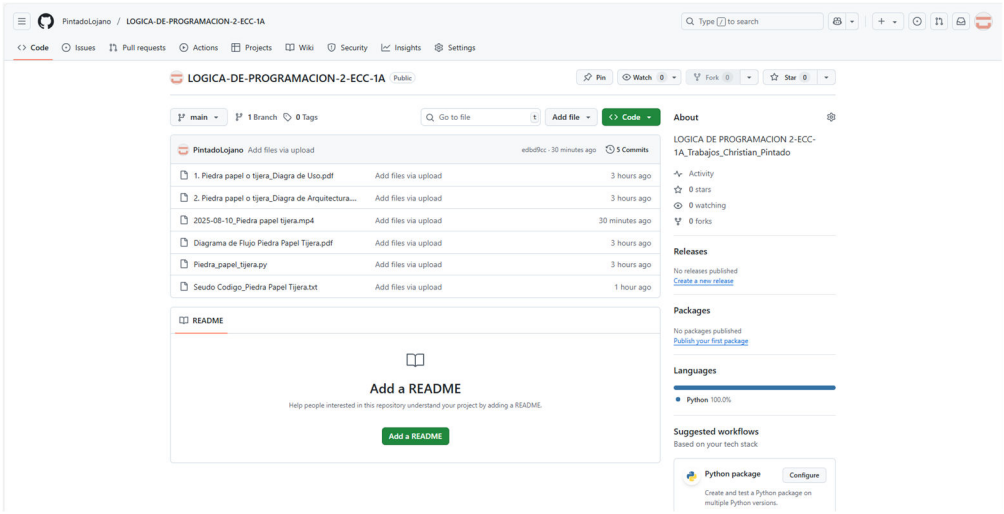
Control de versiones: GitHub permite llevar un registro de todas las modificaciones realizadas en el código, facilitando la restauración de versiones anteriores y el manejo de errores.

Colaboración: Facilita el trabajo en equipo mediante herramientas como las "pull requests" y "issues", lo que permite a los desarrolladores colaborar sin sobrescribir el trabajo de otros.

Accesibilidad y respaldo: Un repositorio en GitHub asegura que el proyecto esté respaldado en la nube y sea accesible desde cualquier lugar, además de permitir compartir el código fácilmente.

Visibilidad y aprendizaje: Subir el código a GitHub permite recibir retroalimentación de la comunidad, mejorar habilidades y aprender de otros desarrolladores.

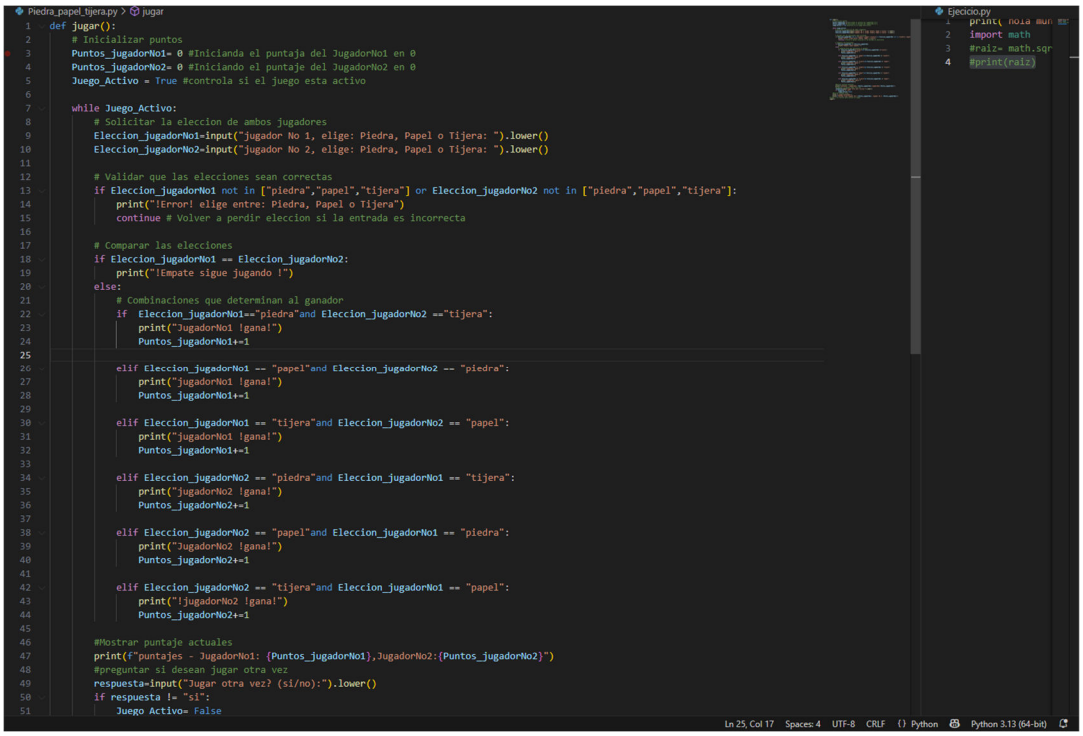
Integración con herramientas: GitHub se integra bien con herramientas de automatización y despliegue, mejorando la eficiencia del desarrollo.



PintadoLojano/LOGICA-DE-PROGRAMACION-2-ECC-1A: LOGICA DE PROGRAMACION 2-ECC-1A_Trabajos_Christian_Pintado

III. Desarrollo del programa seleccionado

En base a lo anterior de genere en Python a codificación:



```python
def jugar():
    # Inicializar puntos
    Puntos_jugadorNo1= 0 #Iniciando el puntaja del JugadorNo1 en 0
    Puntos_jugadorNo2= 0 #Iniciando el puntaje del JugadorNo2 en 0
    Juego_Activo = True #controla si el juego esta activo

    while Juego_Activo:
        # Solicitar la eleccion de ambos jugadores
        Eleccion_jugadorNo1=input("jugador No 1, elige: Piedra, Papel o Tijera: ").lower()
        Eleccion_jugadorNo2=input("jugador No 2, elige: Piedra, Papel o Tijera: ").lower()

        # Validar que las elecciones sean correctas
        if Eleccion_jugadorNo1 not in ["piedra","papel","tijera"] or Eleccion_jugadorNo2 not in ["piedra","papel","tijera"]:
            print("!Error! elige entre: Piedra, Papel o Tijera")
            continue # Volver a perdir eleccion si la entrada es incorrecta

        # Comparar las elecciones
        if Eleccion_jugadorNo1 == Eleccion_jugadorNo2:
            print("!Empate sigue jugando !")
        else:
            # Combinaciones que determinan al ganador
            if  Eleccion_jugadorNo1=="piedra"and Eleccion_jugadorNo2 =="tijera":
                print("JugadorNo1 !gana!")
                Puntos_jugadorNo1+=1

            elif Eleccion_jugadorNo1 == "papel"and Eleccion_jugadorNo2 == "piedra":
                print("jugadorNo1 !gana!")
                Puntos_jugadorNo1+=1

            elif Eleccion_jugadorNo1 == "tijera"and Eleccion_jugadorNo2 == "papel":
                print("jugadorNo1 !gana!")
                Puntos_jugadorNo1+=1

            elif Eleccion_jugadorNo2 == "piedra"and Eleccion_jugadorNo1 == "tijera":
                print("jugadorNo2 !gana!")
                Puntos_jugadorNo2+=1

            elif Eleccion_jugadorNo2 == "papel"and Eleccion_jugadorNo1 == "piedra":
                print("JugadorNo2 !gana!")
                Puntos_jugadorNo2+=1

            elif Eleccion_jugadorNo2 == "tijera"and Eleccion_jugadorNo1 == "papel":
                print("!jugadorNo2 !gana!")
                Puntos_jugadorNo2+=1

        #Mostrar puntaje actuales
        print(f"puntajes - JugadorNo1: {Puntos_jugadorNo1},JugadorNo2:{Puntos_jugadorNo2}")
        #preguntar si desean jugar otra vez
        respuesta=input("Jugar otra vez? (si/no):").lower()
        if respuesta != "si":
            Juego_Activo= False
```

IV. Conclusiones:

- Refuerzo de habilidades de programación: La práctica de crear el juego "Piedra, Papel o Tijeras" permite aplicar conceptos fundamentales de programación, como el uso de variables, estructuras condicionales y manejo de entradas, ayudando a mejorar la lógica y la fluidez en la codificación.

- Importancia de la validación y control de flujo: Validar las elecciones de los jugadores y manejar correctamente las condiciones de empate y ganador son esenciales para crear un juego funcional y robusto, lo que resalta la importancia de validar entradas y usar bucles para controlar el flujo del juego.

- Oportunidad de expansión y mejora: El proyecto inicial sirve como base para agregar nuevas características, como un sistema de rondas, estadísticas o una interfaz gráfica, lo que abre oportunidades para mejorar el juego y aplicar conocimientos a proyectos más complejos en el futuro.

V. Bibliografia

1. https://aws.amazon.com/es/what-is/architecture-diagramming/

2. Baez López, P., Gonzales Barrón, J. M., Hernández Vázquez, E. R., Saavedra Ventura,

A. L., & Tovar Hernández, A. L. (2023). Piedra, Papel o Tijera - Apuntes de Ingeniería de Software. https://www.studocu.com/es-mx/document/tecnologico-de-estudios-superiores-de-ixtapaluca/ingenieria-de-software/piedra-papel-o-tijera-apuntes/81907523

3. https://es.wikipedia.org/wiki/Piedra,_papel_o_tijera