

Universidad Internacional del Ecuador



Escuela de Ingeniería

INGENIERIA EN SISTEMAS DE LA INFORMACION

Evaluación en Contacto con el Docente

Christian Eduardo Pintado Lojano

Docente: Monica Patricia Salazar Tapia

Quito, 23 de agosto 2025

I.	Proyecto Integrador.....	3
1.1	Nombre del proyecto Implementación de código en Python aplicado para el juego Piedra Papel o tijera	3
1.1.1	Introducción	3
1.2	Descripción general del problema o la situación que busca atender el proyecto.....	3
1.2.1	Objetivo.....	4
1.2.2	Propósito del proyecto.....	4
1.2.3	Instrucciones para el desarrollo del proyecto.	4
1.3	Desarrollo de entregables	5
1.3.1	Cronograma de actividades	5
1.3.2	Diagrama de Uso juego piedra papel o tijera	7
1.3.3	Diagrama de Arquitectura	8
1.3.4	Seudo Código.....	9
1.3.5	Diagrama de flujo.....	9
1.3.6	Código Python.....	11
1.3.6.1	Código Python corrección en función a lo comentado.	11
II.	Conclusiones:.....	11
III.	Bibliografía	12

I. Proyecto Integrador

1.1 Nombre del proyecto Implementación de código en Python aplicado para el juego

Piedra Papel o tijera

1.1.1 Introducción

Vivimos en una era donde las nuevas tecnologías están cambiando rápidamente todos los aspectos de nuestra vida. Desde cómo nos comunicamos, trabajamos y aprendemos, hasta cómo interactuamos con el mundo que nos rodea, la tecnología ha pasado de ser una simple herramienta a convertirse en un motor de transformación profunda en la sociedad.

Este proyecto tiene como objetivo reflexionar sobre cómo estas tecnologías están impactando nuestro presente y cómo podrían moldear nuestro futuro. Para lograrlo, fue necesario comprender los fundamentos de la programación, ya que gran parte del desarrollo tecnológico actual se basa en la lógica computacional.

Aprender a programar no solo significa escribir código. También implica desarrollar una forma de pensar ordenada, lógica y eficiente. A través del estudio de la **resolución de problemas**, aprendemos a identificar situaciones reales que pueden ser resueltas paso a paso, utilizando algoritmos. Este enfoque ayuda a mejorar nuestra capacidad para analizar, planificar y construir soluciones prácticas.

Además, el manejo de datos, la validación de entradas, el control del flujo de decisiones y la organización del software en partes más pequeñas (módulos) son habilidades estratégicas que nos permiten entender cómo funcionan las tecnologías que usamos todos los días.

Por último, este enfoque técnico se complementa con una mirada ética y social. No se trata solo de saber programar, sino de comprender cómo nuestras creaciones tecnológicas afectan a las personas y al entorno. Así, formamos no solo desarrolladores, sino también ciudadanos conscientes y responsables en un mundo cada vez más digital.

1.2 Descripción general del problema o la situación que busca atender el proyecto

En el contexto de la asignatura, se identificó la necesidad de desarrollar habilidades básicas de programación mediante la resolución de problemas concretos. Uno de los principales desafíos para quienes inician en este campo es comprender cómo estructurar soluciones lógicas y funcionales a partir de situaciones cotidianas.

Este proyecto propone como caso de estudio el desarrollo del juego Piedra, Papel o Tijera, una dinámica sencilla pero ideal para aplicar conceptos fundamentales de programación como la entrada y validación de datos, estructuras condicionales, ciclos de repetición y control de flujo. A través de este ejercicio, no solo se aprende a escribir código, sino que también desarrolla su capacidad para analizar problemas, diseñar algoritmos y construir soluciones funcionales.

1.2.1 Objetivo

Adquirir una comprensión práctica del proceso de construcción de un programa desde cero, utilizando como caso de estudio el juego Piedra, Papel o Tijera.

El proyecto busca fortalecer el pensamiento lógico, la capacidad de resolución de problemas y fomentar la autonomía en el aprendizaje de tecnologías digitales mediante la aplicación de fundamentos de programación.

1.2.2 Propósito del proyecto

La aplicación del juego Piedra, Papel o Tijera fue desarrollada utilizando el lenguaje de programación Python, aplicando principios fundamentales de lógica computacional. El código se estructura en torno a un ciclo de juego que permite la interacción entre dos jugadores, validando sus elecciones y determinando el resultado de cada ronda.

La lógica del programa se basa en los siguientes componentes:

- Inicialización de variables: Se definen los puntajes de ambos jugadores y se establece una condición para mantener el juego activo.
- Entrada de datos: El sistema solicita a cada jugador que elija entre "Piedra", "Papel" o "Tijera".
- Validación: Se verifica que las entradas sean válidas. Si no lo son, se muestra un mensaje de error y se repite la solicitud.
- Comparación de elecciones: Se evalúan las combinaciones posibles para determinar al ganador de la ronda.
- Asignación de puntos: Según las reglas del juego, se otorgan puntos al jugador que haya ganado la ronda.
- Control de flujo: Se muestra el puntaje actual y se pregunta si desean continuar jugando. Si la respuesta es negativa, el juego finaliza mostrando los resultados finales.

Este enfoque permite aplicar estructuras condicionales (if, elif, else), bucles (while), y manejo de entradas (input()), consolidando habilidades esenciales en programación. Además, el código fue documentado y gestionado mediante un repositorio en GitHub, lo que facilitó el control de versiones, la colaboración y la mejora continua del proyecto.

1.2.3 Instrucciones para el desarrollo del proyecto.

El proyecto se desarrolló a lo largo de 8 semanas, estructurado en 4 unidades de desarrollo teórico práctico en la lógica programación, cada una abordando aspectos fundamentales de la lógica de programación y el desarrollo de software.

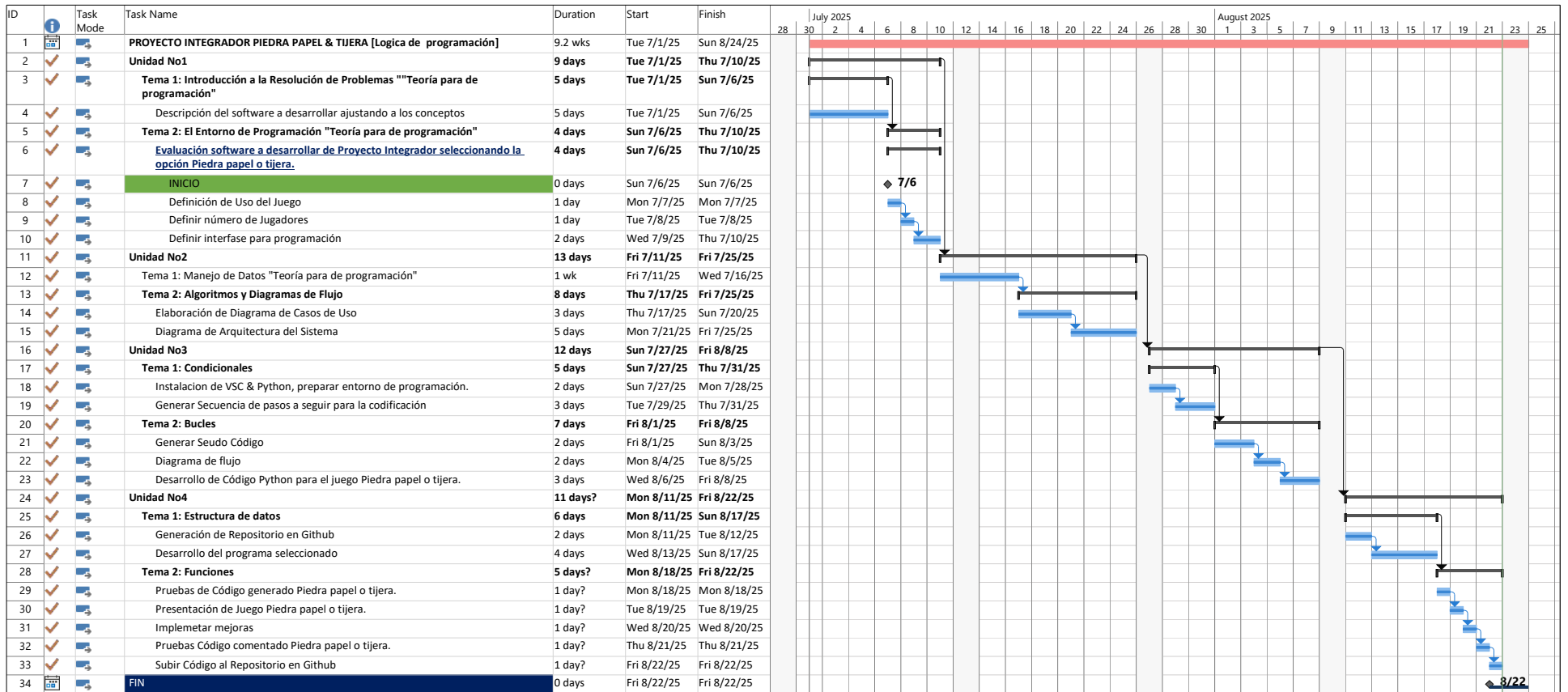
El objetivo fue construir una aplicación funcional del juego Piedra, Papel o Tijera, aplicando conocimientos teóricos y prácticos.

1. Unidad 1: Fundamentos de programación
 - Semana 1: Introducción a la resolución de problemas y descripción del software.
 - Semana 2: Estudio del entorno de programación y evaluación de la propuesta del juego.
Actividades: Definición del uso del juego, número de jugadores e interfaz.

2. Unidad 2: Estructuración lógica
 - Semana 3: Manejo de datos y elaboración de diagramas de casos de uso.
 - Semana 4: Diseño de la arquitectura del sistema. Actividades: Diagramas de flujo, definición de algoritmos y estructura del juego.
3. Unidad 3: Codificación
 - Semana 5: Instalación del entorno (VSC & Python), uso de condicionales.
 - Semana 6: Implementación de bucles, generación de pseudo código y desarrollo del código en Python.
4. Unidad 4: Integración y presentación
 - Semana 7: Uso de estructuras de datos, creación del repositorio en GitHub y desarrollo final del programa.
 - Semana 8: Aplicación de funciones, pruebas del código, presentación del juego, mejoras e integración final en el repositorio.

1.3 Desarrollo de entregables

1.3.1 Cronograma de actividades



Project: Crograma Piedra Pepel
Date: Sat 8/23/25

Task

Split

Milestone

Summary

Project Summary

Inactive Task

Inactive Milestone

Inactive Summary

Manual Task

Duration-only

Manual Summary Rollup

Manual Summary

Start-only

Finish-only

External Tasks

External Milestone

Deadline

Critical

Critical Split

Baseline

Baseline Milestone

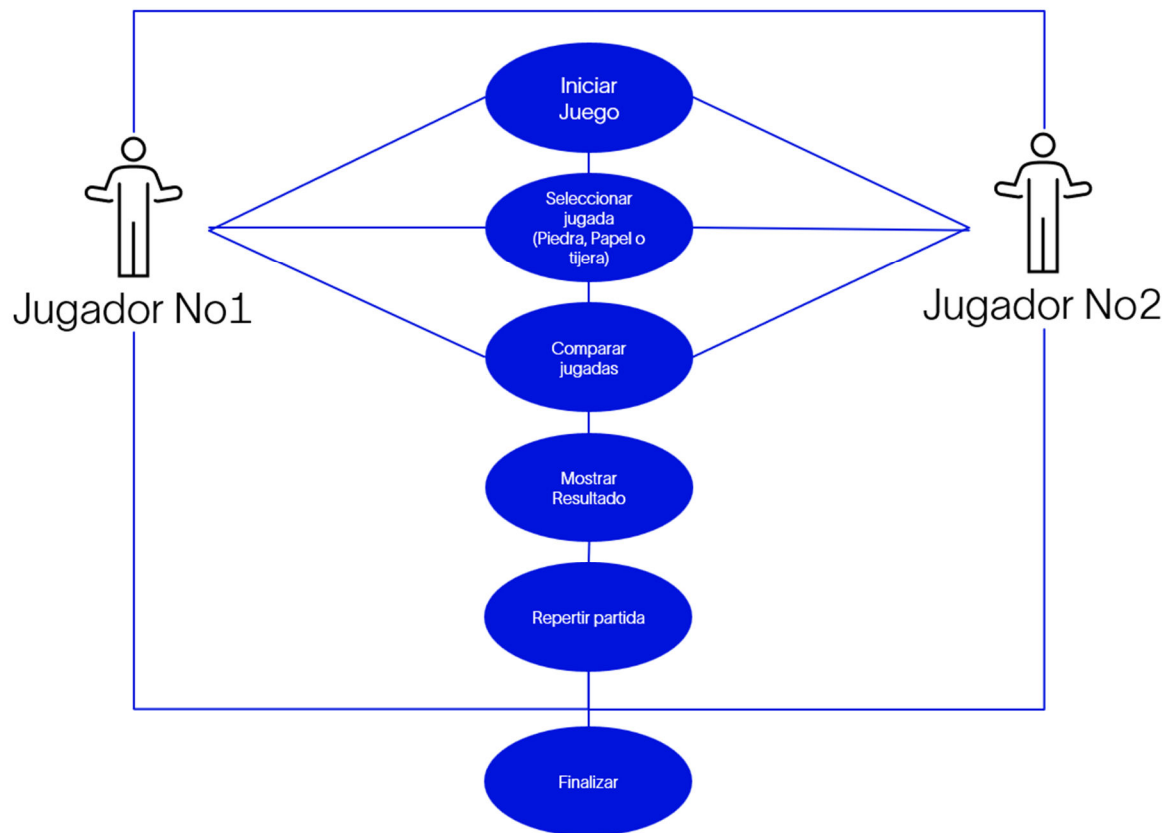
Baseline Summary

Progress

Manual Progress

Slack

1.3.2 Diagrama de Uso juego piedra papel o tijera



El sistema se activa a través de una interfaz para dos jugadores los cuales deben completar el registro de jugador 1 y 2.

Cada jugador elige una opción: piedra, papel o tijera.

Elegido las opciones el sistema compara las elecciones de cada jugador y aplica las reglas establecidas para el juego.

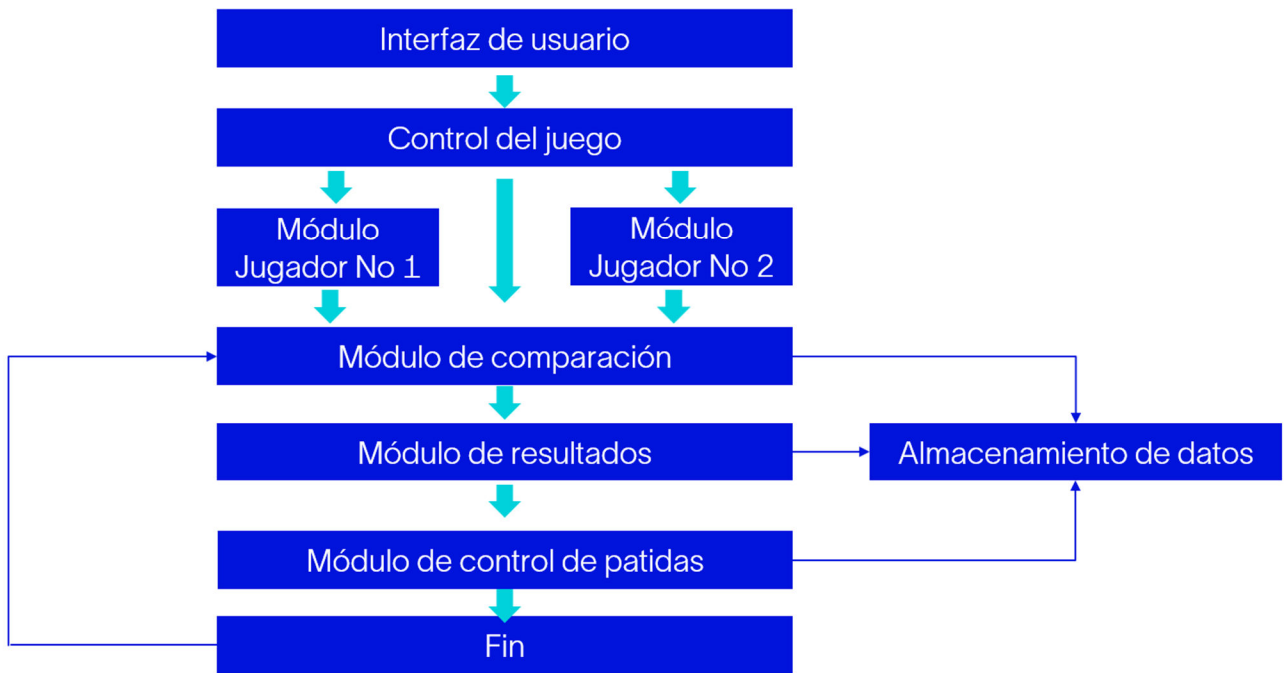
Lógica del juego:

- Piedra gana a tijera.
- Tijera gana a papel.
- Papel gana a piedra.
- Si ambos eligen lo mismo, es empate.

Finalizado la comparación se muestra resultado

El sistema muestra el resultado de la ronda, determinando quién ganó o si fue empate, los datos son almacenados, generando la interrogante de repetir partida o finalizar.

1.3.3 Diagrama de Arquitectura



1.3.4 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

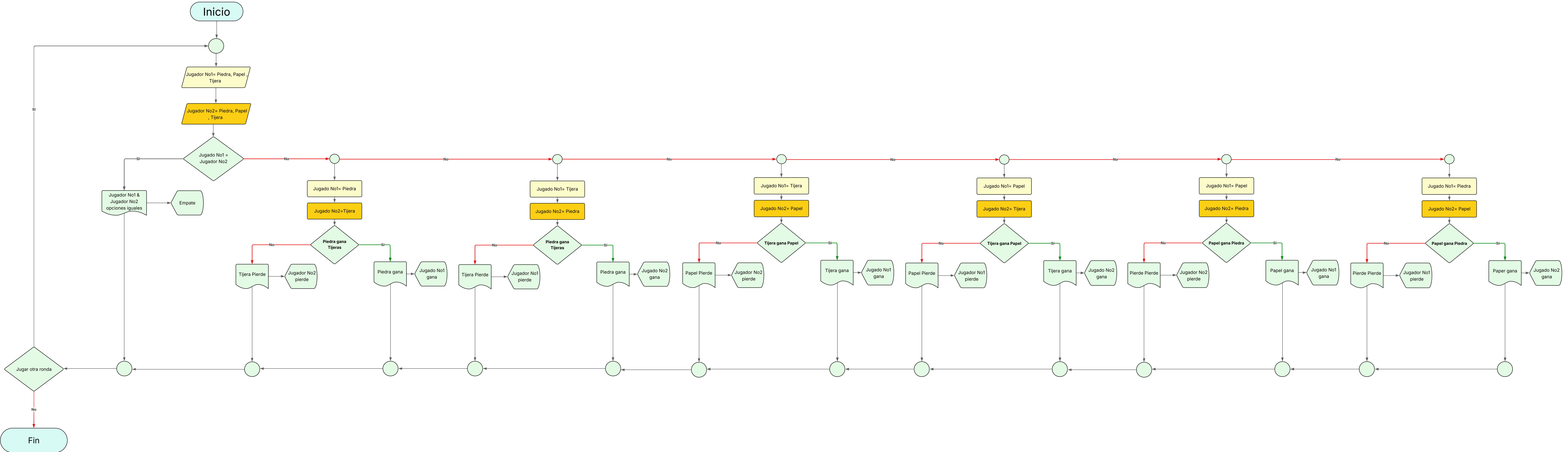
            Mostrar los puntajes actuales de ambos jugadores

        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
```

1.3.5 Diagrama de flujo

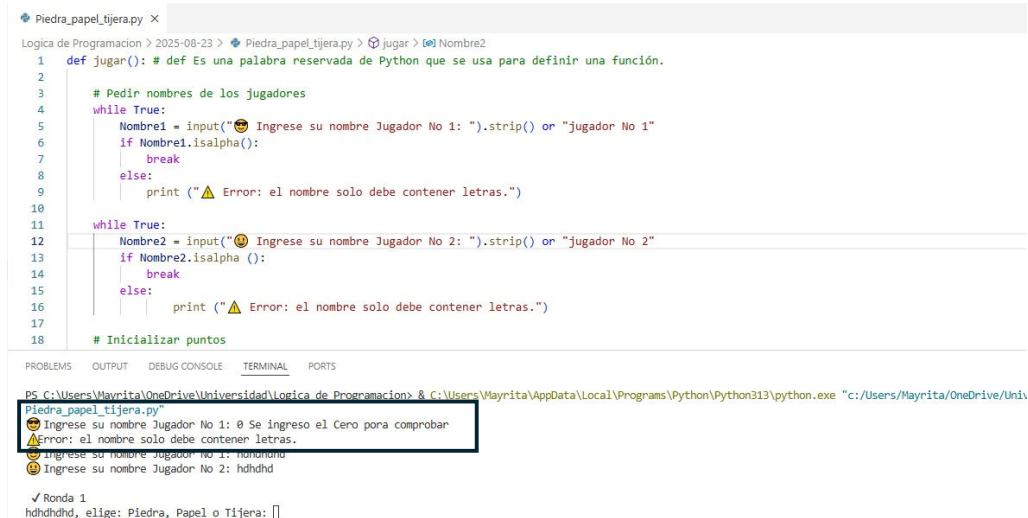


1.3.6 Código Python

https://github.com/PintadoLojano/LOGICA-DE-PROGRAMACION_CHRISTIAN-PINTADO/blob/main/Piedra_papel_tijera.py

1.3.6.1 Código Python corrección en función a lo comentado.

Se incluye la corrección de restringir al código en la fase de solicitud de nombres validar el ingreso de solo texto el cual es comprobado en lo resaltado en el cuadro.



```
1 def jugar(): # def Es una palabra reservada de Python que se usa para definir una función.
2
3     # Pedir nombres de los jugadores
4     while True:
5         Nombre1 = input("😄 Ingrese su nombre Jugador No 1: ").strip() or "jugador No 1"
6         if Nombre1.isalpha():
7             break
8         else:
9             print ("⚠️ Error: el nombre solo debe contener letras.")
10
11     while True:
12         Nombre2 = input("😄 Ingrese su nombre Jugador No 2: ").strip() or "jugador No 2"
13         if Nombre2.isalpha():
14             break
15         else:
16             print ("⚠️ Error: el nombre solo debe contener letras.")
17
18     # Inicializar puntos
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\Mayrita\OneDrive\Universidad\Logica de Programacion> & C:\Users\Mayrita\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/Mayrita/OneDrive/Uni... Piedra_papel_tijera.py"

😄 Ingrese su nombre Jugador No 1: 0 Se ingreso el Cero pona comprobar

⚠️ Error: el nombre solo debe contener letras.

😄 Ingrese su nombre Jugador No 1: nononono

😄 Ingrese su nombre Jugador No 2: hdhdhd

✓ Ronda 1

hdhdhdhd, elige: Piedra, Papel o Tijera: []

Antes de guardar Nombre1 y Nombre2, ahora se valida con **isalpha()**.
Si contiene solo letras → válido.
Si contiene números, espacios o símbolos → **muestra error y vuelve a pedir.**
Se mantiene la misma lógica del juego.

II. Conclusiones:

1. Aplicación de arquitectura modular como base tecnológica
El desarrollo del juego Piedra, Papel o Tijera permitió implementar una arquitectura modular, dividiendo el sistema en componentes funcionales como la interfaz de usuario, la validación de jugadas, la comparación de resultados y el control del flujo del juego. Esta estructura refleja cómo las nuevas tecnologías permiten construir soluciones escalables, comprensibles y adaptables, facilitando su evolución hacia sistemas más complejos como aplicaciones con inteligencia artificial o interfaces gráficas interactivas.
2. Visualización estructurada del sistema mediante diagramas
El uso de diagramas de casos de uso y arquitectura fue de gran importancia para representar de forma clara las funcionalidades del sistema. Estas herramientas, propias del desarrollo tecnológico moderno, permiten entender cómo interactúan los usuarios con las aplicaciones y cómo se organizan internamente. Esta capacidad de modelar sistemas es fundamental en una sociedad donde la tecnología está cada vez más integrada en procesos educativos, laborales y sociales.

3. Importancia del análisis previo en el desarrollo tecnológico

El análisis inicial del problema permitió definir con precisión los requerimientos funcionales del software, anticipar errores y planificar el desarrollo de forma eficiente. Este enfoque demuestra cómo las nuevas tecnologías no solo requieren habilidades técnicas, sino también pensamiento crítico y estratégico. Comprender el problema antes de programar es esencial para crear soluciones tecnológicas que respondan a necesidades reales y tengan un impacto positivo en la sociedad

III. Bibliografía

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
4. https://github.com/PintadoLojano/LOGICA-DE-PROGRAMACION_CHRISTIAN-PINTADO