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, 19 julio de 2025

I. Investigar los tipos de Diagramas de funcionalidad y arquitectura de aplicaciones que existen y seleccionar uno de cada uno.

1.1.1 Diagrama de Casos de Uso [UML]

Este tipo de diagrama pertenece al lenguaje UML (Unified Modeling Language) y se utiliza para representar las interacciones entre los usuarios (actores) y el sistema.
Sirve para entender qué funcionalidades ofrece el sistema desde el punto de vista del usuario.
Elementos principales:

Actor
El actor en un diagrama de caso de uso de es cualquier entidad que desempeñe un papel en un sistema determinado. Puede ser una persona, una organización o un sistema externo y normalmente se dibuja como el esqueleto que se muestra a continuación.

Caso de uso
Un caso de uso representa una función o una acción dentro del sistema. Está dibujado como un óvalo y nombrado con la función.

Sistema
El sistema se utiliza para definir el alcance del caso de uso y se dibuja como un rectángulo. Este es un elemento opcional pero útil cuando se visualizan sistemas grandes. Por ejemplo, puede crear todos los casos de uso y luego utilizar el objeto del sistema para definir el alcance que abarca su proyecto. O incluso puedes usarlo para mostrar las diferentes áreas cubiertas en los diferentes lanzamientos.

Paquete
El paquete es otro elemento opcional que es extremadamente útil en diagramas complejos. De manera similar a los diagramas de clase, los paquetes se utilizan para agrupar los casos de uso. Se dibujan como la imagen que se muestra a continuación.

1.1.2 Diagrama de Arquitectura del Sistema

Este diagrama describe la estructura general del sistema, incluyendo sus componentes principales y cómo se comunican entre sí.
Puede representarse de varias formas, dependiendo del enfoque (por ejemplo, lógico, físico, de despliegue).

Tipos comunes:

Arquitectura lógica: Muestra módulos, capas (por ejemplo, presentación, lógica de negocio, datos).
Arquitectura física: Describe servidores, redes, dispositivos y cómo están conectados.
Arquitectura de software: Componentes de software, servicios, APIs, bases de datos, etc.

Ventajas:
La posibilidad de colaborar, reducir los riesgos, aumentar la eficacia y la escalabilidad son las ventajas de los diagramas de arquitectura.

Colaboración mediante Diagramas de Arquitectura
Los diagramas de arquitectura fomentan la colaboración entre desarrolladores, diseñadores y otros miembros del equipo al ofrecer una visión compartida del sistema. Esta representación común:

- Mejora la comunicación durante el diseño.
- Facilita el desarrollo de componentes eficaces.
- Ayuda a identificar problemas potenciales desde etapas tempranas.
- Contribuye al cumplimiento de los objetivos del proyecto

Reducción del Riesgo con Diagramas de Arquitectura
Los diagramas de arquitectura permiten identificar riesgos potenciales en etapas tempranas del desarrollo, como errores en la lógica, suposiciones incorrectas o pruebas insuficientes. Al detectar estos problemas a tiempo, los equipos pueden realizar ajustes anticipadamente, reduciendo así la probabilidad de fallos críticos en fases posteriores del proyecto.

Eficiencia
Los diagramas de arquitectura ofrecen una visión clara de los componentes y la estructura del sistema, lo que permite a las partes interesadas identificar y resolver problemas con mayor precisión y rapidez. Además, facilitan el mantenimiento y la escalabilidad del sistema, haciendo que los cambios continuos sean más eficientes y controlados.

Escalabilidad
Estos diagramas permiten a las partes interesadas visualizar formas eficientes de escalar un sistema. Por ejemplo, pueden mostrar si la arquitectura es centralizada o distribuida. Dado que los sistemas distribuidos escalan mejor, los componentes monolíticos pueden ser optimizados o reemplazados a tiempo. Asimismo, las representaciones gráficas ayudan a comprender cómo se almacenan y transfieren los datos, permitiendo detectar cuellos de botella y proponer soluciones antes de que afecten el rendimiento

II. Descripción del software a desarrollar

El software para desarrollar es un juego interactivo de Piedra, Papel o Tijera para dos jugadores. Cada jugador seleccionará una opción (piedra, papel o tijera) y el sistema comparará ambas elecciones para determinar el ganador.
El juego permitirá repetir rondas y finalizar la sesión cuando los jugadores lo deseen.
Este software se desarrollará inicialmente como línea de código.

2.1 Antecedentes, origen histórico

China (siglo II a.C.):
El juego se originó en China durante la dinastía Han, donde se conocía como *shoushiling*, que significa “orden de gestos con la mano”.
Japón (siglo XVII):
Se popularizó como *jan-ken*, usando los gestos que conocemos hoy: piedra (fist), papel (open

hand), tijera (two fingers). Jan-ken se convirtió en parte de la cultura japonesa y se usaba para tomar decisiones cotidianas.

Europa (siglo XIX–XX):

El juego llegó a Europa a través del contacto con Asia, y se adaptó como "rock-paper-scissors" en inglés. En español, se tradujo como "piedra, papel o tijera".

2.2 Teoría de Operación del juego:

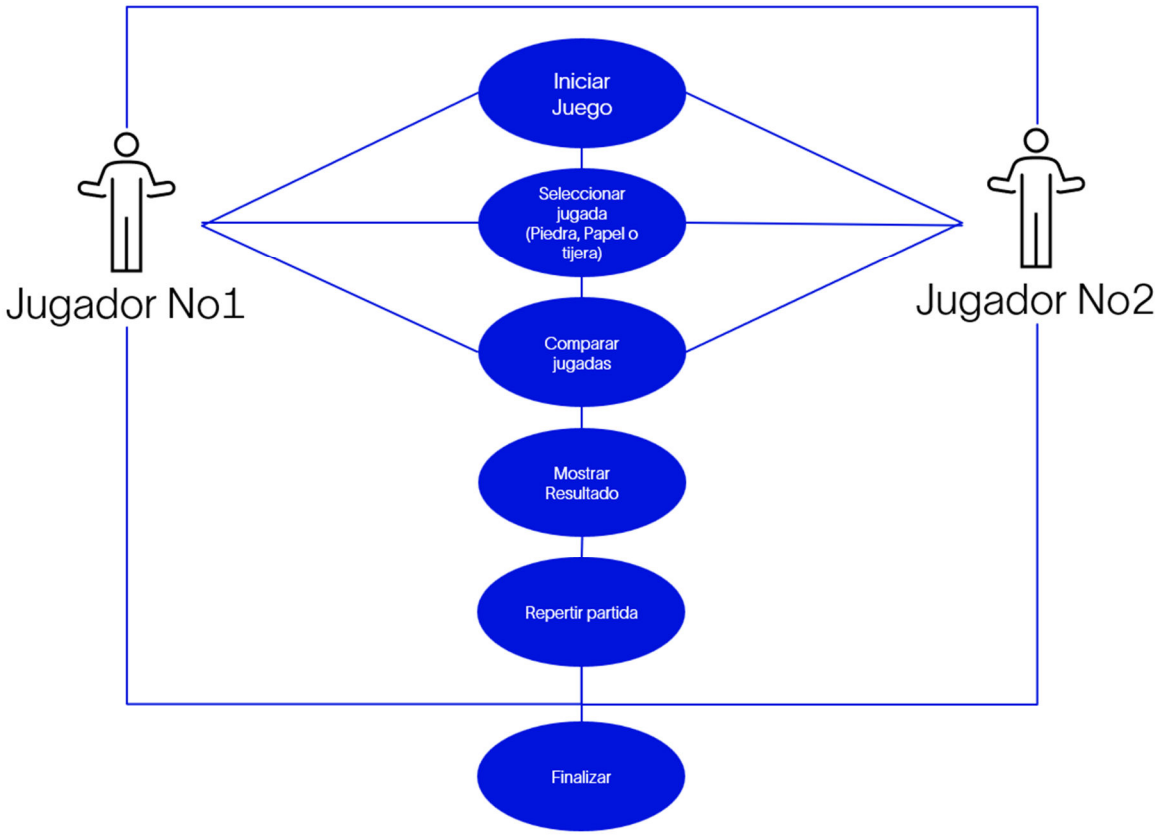
- **Piedra** vence a **tijera** (la rompe).
- **Tijera** vence a **papel** (lo corta).
- **Papel** vence a **piedra** (la envuelve).

Es un juego de **estrategia simple y azar**, usado para tomar decisiones rápidas, como quién empieza un juego o quién gana un turno

El juego **Piedra, papel o tijera** está diseñado principalmente para **dos jugadores**, pero también se puede adaptar para más personas con algunas variaciones.

III. Diagrama de Uso juego piedra papel o tijera

Las funcionalidades del sistema se han modelado mediante un diagrama de casos de uso. Los actores principales son los dos jugadores, quienes interactúan con el sistema para iniciar el juego, seleccionar sus jugadas, ver los resultados y decidir si desean continuar o finalizar la partida. El sistema se encarga de validar las jugadas, compararlas y mostrar el resultado.



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.

3.1 Diagrama de Arquitectura

El diseño de arquitectura del sistema para el juego Piedra, Papel o Tijera se ha planteado a nivel macro, identificando los módulos principales que conforman la aplicación y cómo interactúan entre sí para garantizar un flujo lógico y funcional del juego.

Interfaz de Usuario: Es el punto de entrada del sistema, donde ambos jugadores ingresan sus elecciones (piedra, papel o tijera) y visualizan los resultados de cada ronda. Esta interfaz puede ser textual (consola) o gráfica (GUI), dependiendo del entorno de desarrollo.

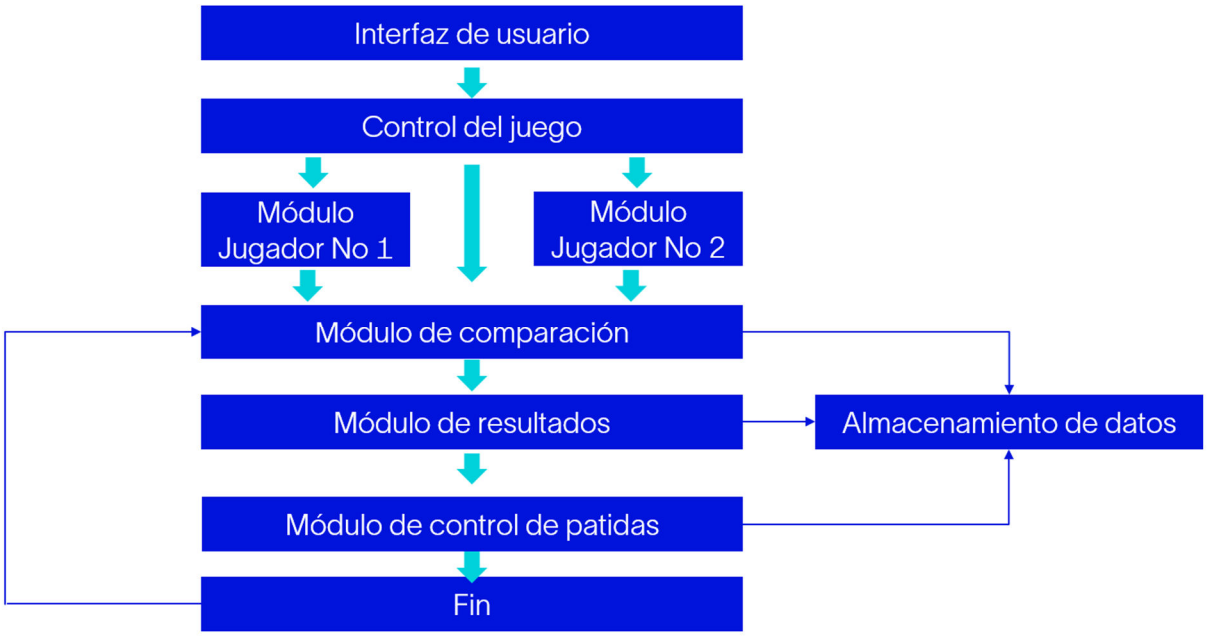
Módulo de Control de Jugadores: Este módulo gestiona las entradas de cada jugador, asegurando que las jugadas sean válidas y estén dentro de las opciones permitidas. También puede encargarse de manejar los turnos si se implementa una lógica secuencial.

Módulo de Comparación: Recibe las jugadas validadas de ambos jugadores y aplica las reglas del juego para determinar el resultado de la ronda. Este módulo encapsula la lógica central del juego.

Módulo de Resultados: Una vez determinada la jugada ganadora, este módulo genera el mensaje correspondiente (ganador, perdedor o empate) y lo envía a la interfaz de usuario para su visualización.

Módulo de Control de Juego: Coordina el flujo general del sistema. Se encarga de iniciar y finalizar el juego, controlar la repetición de rondas, y orquestar la interacción entre los demás módulos. Es el núcleo que mantiene la coherencia del ciclo de juego.

Este enfoque modular permite una implementación clara, mantenible y escalable, facilitando futuras mejoras como la incorporación de estadísticas, niveles de dificultad o una interfaz gráfica más avanzada.



IV. Conclusiones:

1. El desarrollo del juego permitió aplicar una arquitectura modular clara, dividiendo el sistema en componentes como la interfaz de usuario, validación de jugadas, comparación de resultados y control del flujo del juego. Esta separación facilita la comprensión del sistema, su mantenimiento y futuras ampliaciones, como la incorporación de estadísticas o una interfaz gráfica.

2. El uso de diagramas de casos de uso y arquitectura permitió visualizar de forma estructurada las funcionalidades del sistema y su comportamiento general. Estos diagramas fueron fundamentales

para identificar los actores, las interacciones clave y los módulos principales, sirviendo como base sólida para la fase de diseño y posterior implementación.

3. El análisis previo al desarrollo del software fue esencial para entender completamente el problema a resolver. Esta etapa permitió definir con precisión los requerimientos funcionales, anticipar posibles errores lógicos y establecer una hoja de ruta clara para el diseño e implementación del juego, asegurando así un desarrollo más eficiente y enfocado.

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