Video explicativo:
https://youtu.be/XnzLjIwWqvU

Repositorio Github:
jujocachu/pipati2.0

Lógica de Programación

Docente: Paula Vizcaíno

Actividad: Evaluación en Contacto con el Docente

Nombre del estudiante: Juan José Carrasco Chuchuca

Desarrollo de "Piedra, Papel o Tijera"

Para este proyecto, hemos elegido desarrollar una versión digital del clásico juego "Piedra, Papel o Tijera". Este juego simple pero estratégico sirve como una excelente plataforma para demostrar principios de comunicación efectiva y desarrollo de software.

Diagrama de Casos de Uso (UML)

Representaremos las funcionalidades del sistema, los roles de los usuarios y los posibles resultados del juego, asegurando una comprensión clara de las interacciones.

Diagrama de Arquitectura de Capas

Utilizaremos una arquitectura de capas para separar responsabilidades. Esto facilita el desarrollo, mantenimiento y escalabilidad, permitiendo futuras mejoras como el almacenamiento de resultados.

Análisis del Problema: Reglas de Juego

El juego "Piedra, Papel o Tijera" es un enfrentamiento entre dos jugadores que eligen simultáneamente una de las tres opciones. Las reglas son fundamentales para la lógica del juego y el desarrollo del software.

Piedra

Representa la fuerza bruta y la solidez. La piedra es invencible frente a las tijeras, pero vulnerable al papel.

Papel

Simboliza la envoltura y la anulación. El papel supera a la piedra, pero es susceptible a las tijeras.

Tijera

Encarna el corte y la precisión. Las tijeras ganan al papel, pero pierden ante la piedra.

Diseño Funcional: Diagrama de Casos de Uso

El Diagrama de Casos de Uso (UML) ilustra cómo los usuarios interactúan con el sistema y las funcionalidades principales del juego. Cada caso de uso representa una acción específica que un jugador puede realizar.

Casos de Uso Principales

- Iniciar Juego
- Seleccionar Opción (Piedra, Papel o Tijera)
- Comparar Elecciones
- Mostrar Resultado
- Continuar o Salir del Juego

Interacción del Jugador

El diagrama resalta el flujo de interacción, desde el inicio del juego hasta la toma de decisiones del jugador y la visualización del resultado.

La claridad de este diagrama es crucial para que el equipo de desarrollo comprenda la lógica y construya un sistema robusto.

Diseño de Arquitectura del Sistema

La arquitectura del sistema se divide en capas para una gestión eficiente y modular. Esta estructura permite a los desarrolladores trabajar en componentes específicos sin afectar otras partes del juego.

Capa de Interfaz de Usuario (UI)

- Muestra las opciones de juego y los resultados.
- Permite a los jugadores interactuar con el juego mediante selecciones.
- Maneja la opción de salir del juego.

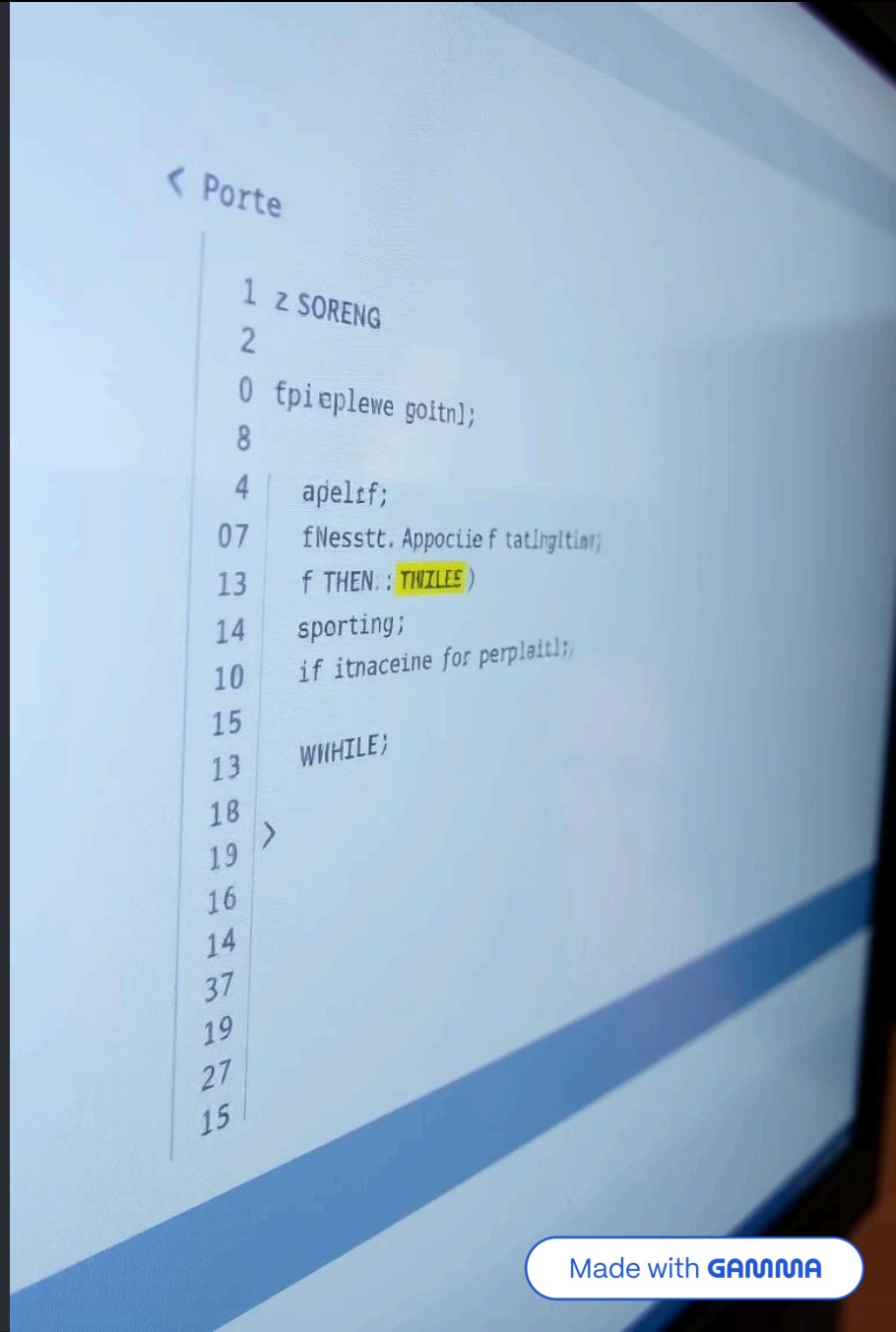
Capa de Lógica del Juego

- Contiene las reglas del juego para determinar el ganador.
- Gestiona las jugadas de los usuarios y la lógica de la "computadora".
- Coordina el estado del juego, incluyendo turnos, rondas y puntuaciones.

Pseudocódigo del Juego

El pseudocódigo describe la lógica del juego de manera estructurada, facilitando la implementación en cualquier lenguaje de programación. Detalla cada paso, desde el inicio hasta la determinación del ganador.

```
Inicio:
  Bucle principal (mientras el juego no termine):
    Turno del Jugador 1:
      Solicitar elección (piedra, papel, tijera o "salir").
      Si "salir", terminar el juego.
    Turno del Jugador 2:
      Solicitar elección (piedra, papel, tijera).
    Validar jugadas:
      Si alguna jugada es inválida, mostrar error y reiniciar ronda.
    Mostrar elecciones de ambos jugadores.
    Determinar ganador:
      Si empate, declarar empate.
      Si Jugador 1 gana según reglas (Piedra vs. Tijera, etc.), declarar Jugador 1 ganador.
      Si no, declarar Jugador 2 ganador.
    Mostrar resultado de la ronda.
Fin de bucle.
```



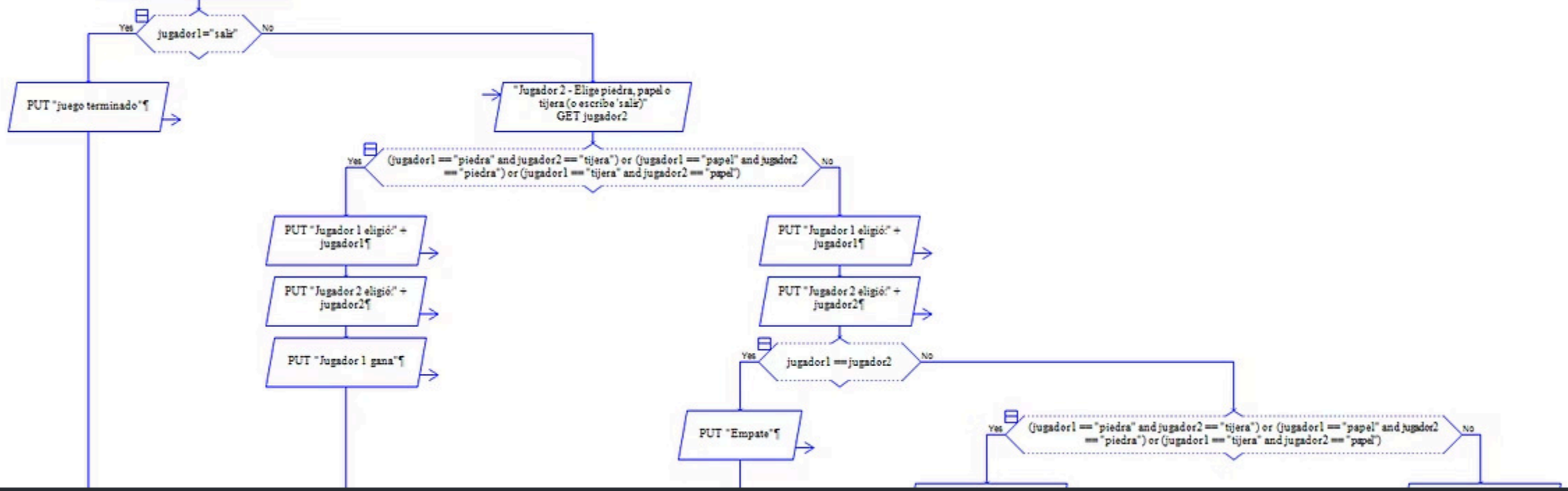


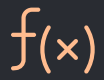
Diagrama de Flujo

El diagrama de flujo visualiza la secuencia de operaciones y decisiones en el juego. Es una herramienta clave para comprender el control del programa y detectar posibles fallos en la lógica antes de la codificación.

Este diagrama es esencial para el equipo de desarrollo, ya que proporciona una representación gráfica clara del algoritmo, permitiendo una implementación más eficiente y con menos errores.

Nueva Versión: Piedra, Papel o Tijera 2.0

Hemos implementado mejoras significativas para optimizar la funcionalidad y la experiencia de usuario, utilizando estructuras de datos más eficientes y lógicas de validación robustas.



Función jugar()

Engloba el código del juego en un bloque reutilizable para modularidad y limpieza.

Diccionario de Reglas

Una estructura que define quién gana a quién, facilitando la lógica de comparación.

Bucle Infinito

Asegura que el juego continúe hasta que el jugador decida salir, mejorando la jugabilidad.

Validación de Entradas

Verifica que las jugadas sean válidas, manejando errores y reiniciando la ronda si es necesario.

Determinación con `get`

Utiliza el método `get` para una búsqueda eficiente en el diccionario de reglas.