

# LÓGICA DE PROGRAMACIÓN

PROYECTO FINAL

SAMUEL GUASCO

# Razones y Objetivos del Proyecto

- Aplicación de conocimientos clave: Elegimos este proyecto porque nos permitió poner en práctica conceptos fundamentales de la programación en Python, como estructuras de control, funciones y validación de entradas. Además, nos dio la oportunidad de integrar librerías externas como random, threading, getpass y sys, ampliando nuestro conocimiento sobre su uso en proyectos reales.
- Mejora continua y optimización: A través de este juego, trabajamos en la optimización del código, la experiencia del usuario y la implementación de buenas prácticas como la modularización y la documentación. Recibir y aplicar retroalimentación de compañeros y profesor fue clave para depurar errores y mejorar la lógica del programa.
- Desarrollo de habilidades prácticas: Buscamos fortalecer nuestro aprendizaje en Python mediante la práctica, al tiempo que aprendimos a utilizar herramientas como GitHub para el control de versiones. Además, enfocamos nuestro trabajo en construir una solución funcional, estructurada y orientada a la experiencia del usuario.

PIEDRA, PAPEL Y  
TIJERAS



# DESARROLLO DEL PROYECTO

# SEMANA 1-2

## DIAGRAMA DE FLUJO

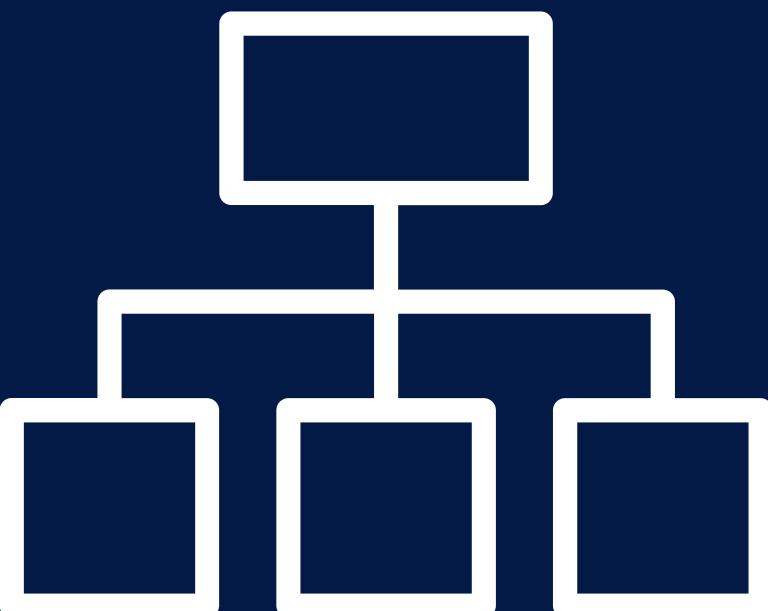
Este tipo de diagrama nos permitió desarrollar y entender la lógica de los programas que se crean. Se aprendió a usar correctamente la simbología y lograr tener un diagrama entendible antes de desarrollar.

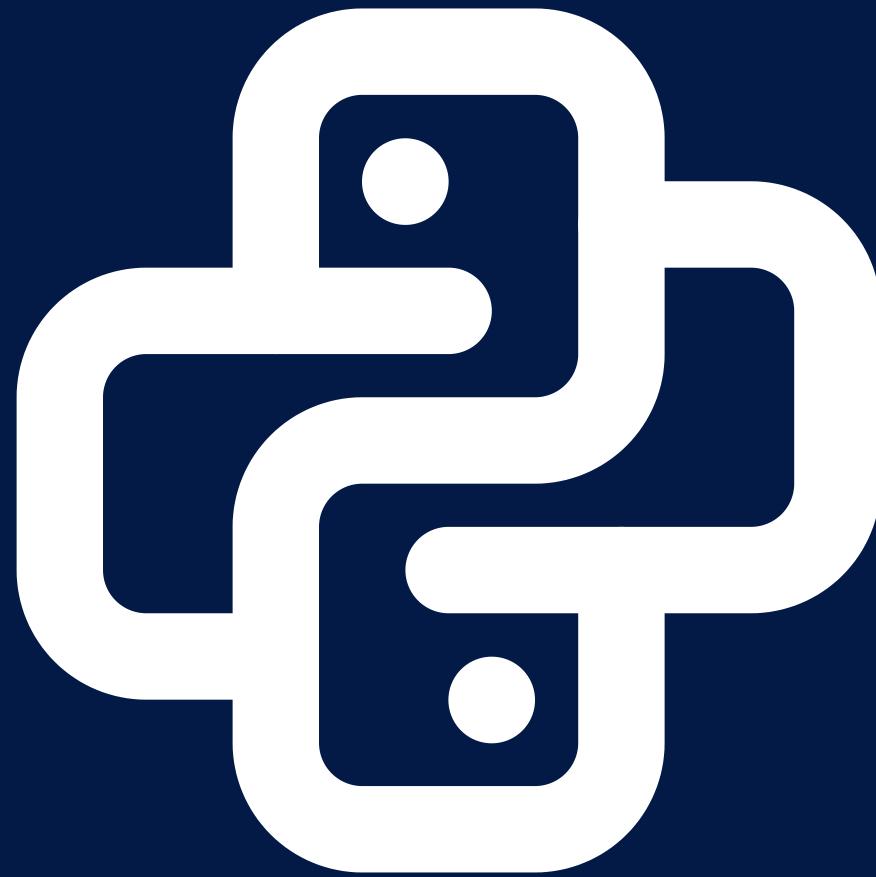
## DIAGRAMAS UML

Estos diagramas nos permitieron entender los componentes del juegos y la interacción de los usuarios.

## ARQUITECTURA DE SOFTWARE

La arquitectura de Software, permitió entender los dispositivos de entrada, los procesos y los dispositivos de salida.





# SEMANA 3-4

**Primeras**  
líneas de código

**Github**  
Configuración de  
Entorno

**Estructura**  
Inicial

# SEMANA 5 - 6

## Funciones

Se empezó a crear la funciones para que el código vaya funcionando de acuerdo a lo que nos enseñaban en clase y según las indicaciones establecidas.

## Librerías

Se usó mayormente librería que eran de Python, esto facilitó mucho su importe. No se tuvo que instalar ninguna librería aparte que sea más compleja.

## Investigación

Este proyecto conlleva de mucha investigación porque el curso tiene una parte de autoaprendizaje. EN esta se pudo investigar para lograr algunas líneas de código que lograban solucionar algunos problemas que se tenía y alinearse a lo que se planteaba en un inicio.

# SEMANA 7-8

En estas semanas se recibió retroalimentación por parte de compañeros y de la profesora en la ultima entrega. En estas semanas el objetivo era pulir el código para la presentación final del proyecto de lógica de programación



# CARACTERÍSTICAS DEL JUEGO

# MENÚ PRINCIPAL Y OPCIONES DE JUEGO

🎮 Bienvenido al Juego: Piedra, Papel o Tijeras 🎮

1. Jugar solo (contra la computadora)
2. Modo multijugador (2 jugadores)
3. Reglas
4. Ver estadísticas
5. Salir

Seleccione una opción (1-5): █

# ESTADÍSTICAS Y REGLAS

## 📊 Estadísticas del juego:

Total de partidas jugadas: 6

Jugador 1 ganó: 0 veces

Jugador 2 ganó: 0 veces

Computador ganó: 6 veces

Empates: 0 veces

## 📜 Historial de partidas:

Partida 1: sam perdió - Computadora ganó

Partida 2: sam perdió - Computadora ganó

Partida 3: sam perdió - Computadora ganó

Partida 4: sam perdió - Computadora ganó

Partida 5: sam perdió - Computadora ganó

Partida 6: sam perdió - Computadora ganó

## 📜 REGLAS DEL JUEGO 📜

1 Piedra vence a Tijeras, Tijeras vence a Papel, y Papel vence a Piedra.

2 Si ambos jugadores eligen la misma opción, es un empate.

3 En modo multijugador, cada jugador elige en secreto su opción.

4 El juego continúa hasta que decidas salir.

# VALIDACIONES Y CONTROL DE ERRORES

```
samuel eligió:  
Computadora eligió: Papel  
Traceback (most recent call last):  
  File "c:\Users\samue\L-gicaDeProgramacion_PiedraPapelTijeras\juego.py", line 250, in <module>  
    mostrar_menu()  
  File "c:\Users\samue\L-gicaDeProgramacion_PiedraPapelTijeras\juego.py", line 31, in mostrar_menu  
    jugar_contra_pc()  
  File "c:\Users\samue\L-gicaDeProgramacion_PiedraPapelTijeras\juego.py", line 121, in jugar_contra_pc  
    jugar_contra_pc() # Se reinicia la función para jugar otra vez.  
    ~~~~~~  
  
  File "c:\Users\samue\L-gicaDeProgramacion_PiedraPapelTijeras\juego.py", line 101, in jugar_contra_pc  
    resultado = determinar_ganador(eleccion_usuario, eleccion_pc)  
    ~~~~~~  
  
  File "c:\Users\samue\L-gicaDeProgramacion_PiedraPapelTijeras\juego.py", line 212, in determinar_ganador  
    return "jugador1" if condiciones_ganadoras[jugador1] == jugador2 else "jugador2"  
    ~~~~~~  
  
KeyError: ''  
PS C:\Users\samue\L-gicaDeProgramacion_PiedraPapelTijeras> |
```



```
• Modo: Jugador vs Computadora  
Ingrrese su nombre: sam  
sam, ¿cuántas rondas deseas jugar? (Ingrese un número): 3  
  
⌚ Ronda 1 de 3  
  
Opciones: 1) Piedra 2) Papel 3) Tijeras  
sam, elija una opción (1-3): 6  
✖ Opción inválida. Por favor, elige 1, 2 o 3.
```

```
⌚ Ronda 1 de 2  
  
Opciones: 1) Piedra 2) Papel 3) Tijeras  
xd, elija una opción (1-3): |  
⏰ Se acabó el tiempo, perdiste esta ronda automáticamente.
```



```
⌚ Ronda 1 de 3  
  
Opciones: 1) Piedra 2) Papel 3) Tijeras  
sam, elija una opción (1-3):  
⏰ Se acabó el tiempo, perdiste esta ronda automáticamente. Presiona ENTER 2
```

# TECNOLOGÍAS UTILIZADAS

```
# Juego Piedra, Papel o Tijeras con lógica de comparación
# Importación de librerías necesarias
import sys # Proporciona acceso a funciones y variables del sistema, como la entrada estándar
import random # Para la elección aleatoria de la computadora
import getpass # Para ocultar la entrada en modo multijugador
import threading # Para implementar el temporizador
import time # Para manejar las pausas del temporizador
```

# CONCLUSIONES

- Aplicación de conceptos clave
- Importancia del manejo de errores
- Aprendizaje sobre la experiencia de usuario
- Primera experiencia en entornos de programación

# **THANK YOU**

**BY SAMUEL GUASCO  
INGENIERIA EN SOFTWARE**