# Lógica de programación

Grace Carrasco

# Contexto del Trabajo Autónomo 1

- Selección de un software a desarrollar.
- Representación mediante diagramas funcionales y arquitectura de software.
- Aplicación práctica del análisis y diseño en un programa real.
- Ejemplo de software elegido: Juego del Ahorcado.

# Diagramas Funcionales

Los diagramas funcionales son herramientas gráficas que permiten visualizar qué hace un sistema y cómo se comporta sin necesidad de ver el código. Su función es definir y entender las funcionalidades antes de la implementación.

| Diagrama | Qué muestra | Para qué sirve |
|---|---|---|
| **Casos de Uso (UML)** | **Actores y funciones principales del sistema.** | **Definir requisitos funcionales e identificar quién hace qué.** |
| **Actividades (UML)** | **Flujo de pasos, decisiones y bucles de un proceso.** | **Documentar flujos de trabajo y analizar procesos alternativos.** |
| **Flujo de Datos (DFD)** | **Circulación de la información entre procesos, datos y usuarios.** | **Comprender cómo se mueven los datos y separar lógica funcional de lo técnico.** |
| BPMN | Procesos de negocio con tareas, eventos y decisiones. | Comunicar procesos de manera estándar y precisa. |
| SIPOC | Proveedores, entradas, proceso, salidas y clientes. | Dar una visión de alto nivel de un sistema/proceso. |
| Diagrama de Flujo (Flowchart) | Secuencia lógica de pasos de un algoritmo. | Planificar algoritmos y detallar funciones paso a paso. |
| Pseudocódigo | Descripción textual de un algoritmo. | Expresar la lógica antes de programar y evitar errores de diseño. |

# Diagrama de Caso de Uso (Use Case Diagram)

Software elegido: Juego del Ahorcado.

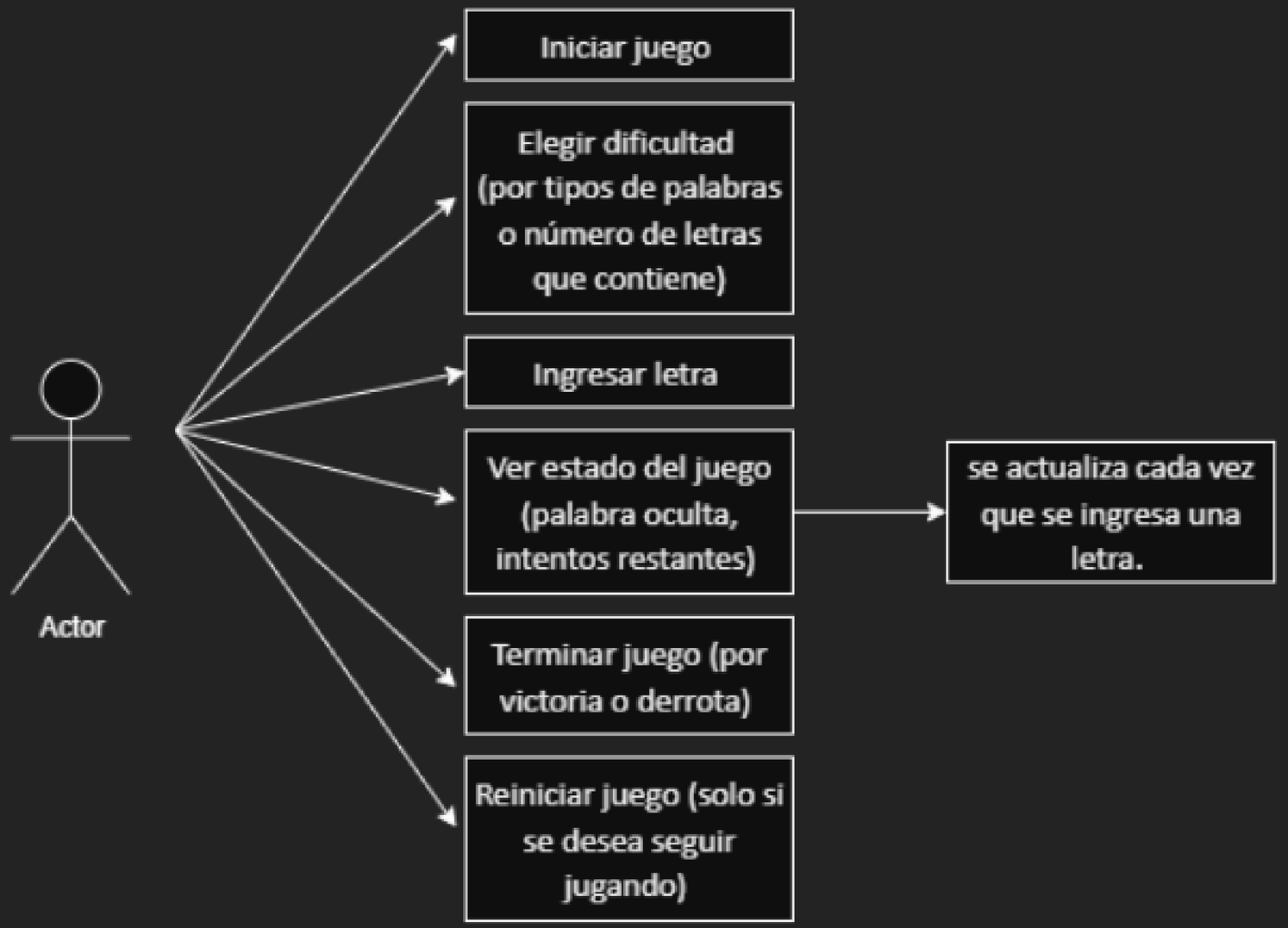
Con este diagrama se mostrará que puede hacer el usuario y que funcionalidades ofrecerá el sistema. La intención es entender los requisitos funcionales.

Actor principal:
- Jugador (Usuario)

Casos de uso:
1. Iniciar juego
2. Elegir dificultad (por tipos de palabras o número de letras que contiene)
3. Ingresar letra
4. Ver estado del juego (palabra oculta, intentos restantes)
5. Terminar juego (por victoria o derrota)
6. Reiniciar juego (solo si se desea seguir jugando)

Relaciones:
- El jugador interactúa con todos los casos.
- "Ver estado del juego" se actualiza cada vez que se ingresa una letra.

# Diagrama de Caso de Uso (Use Case Diagram)



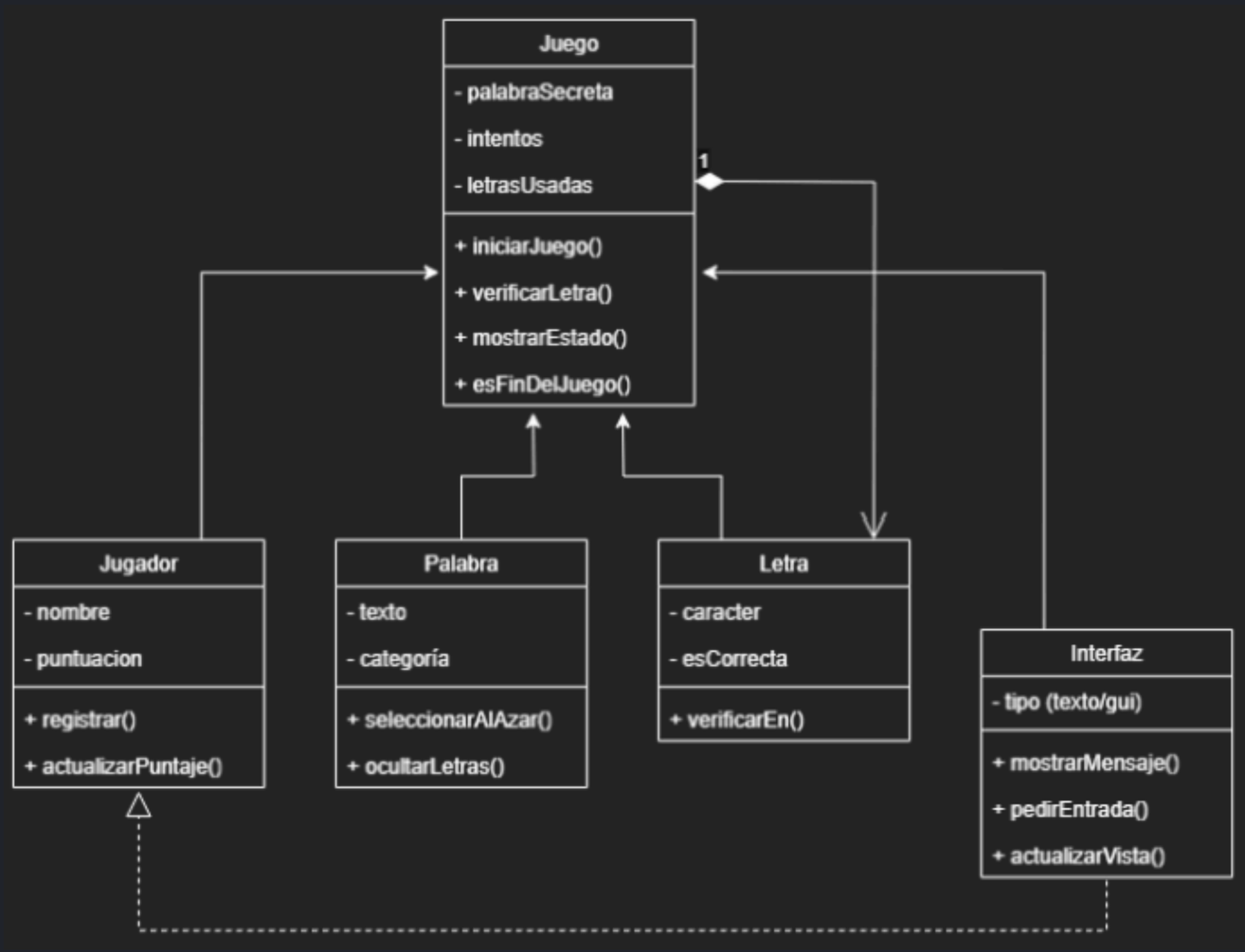
# Arquitectura de Software

Software elegido: Juego del Ahorcado.

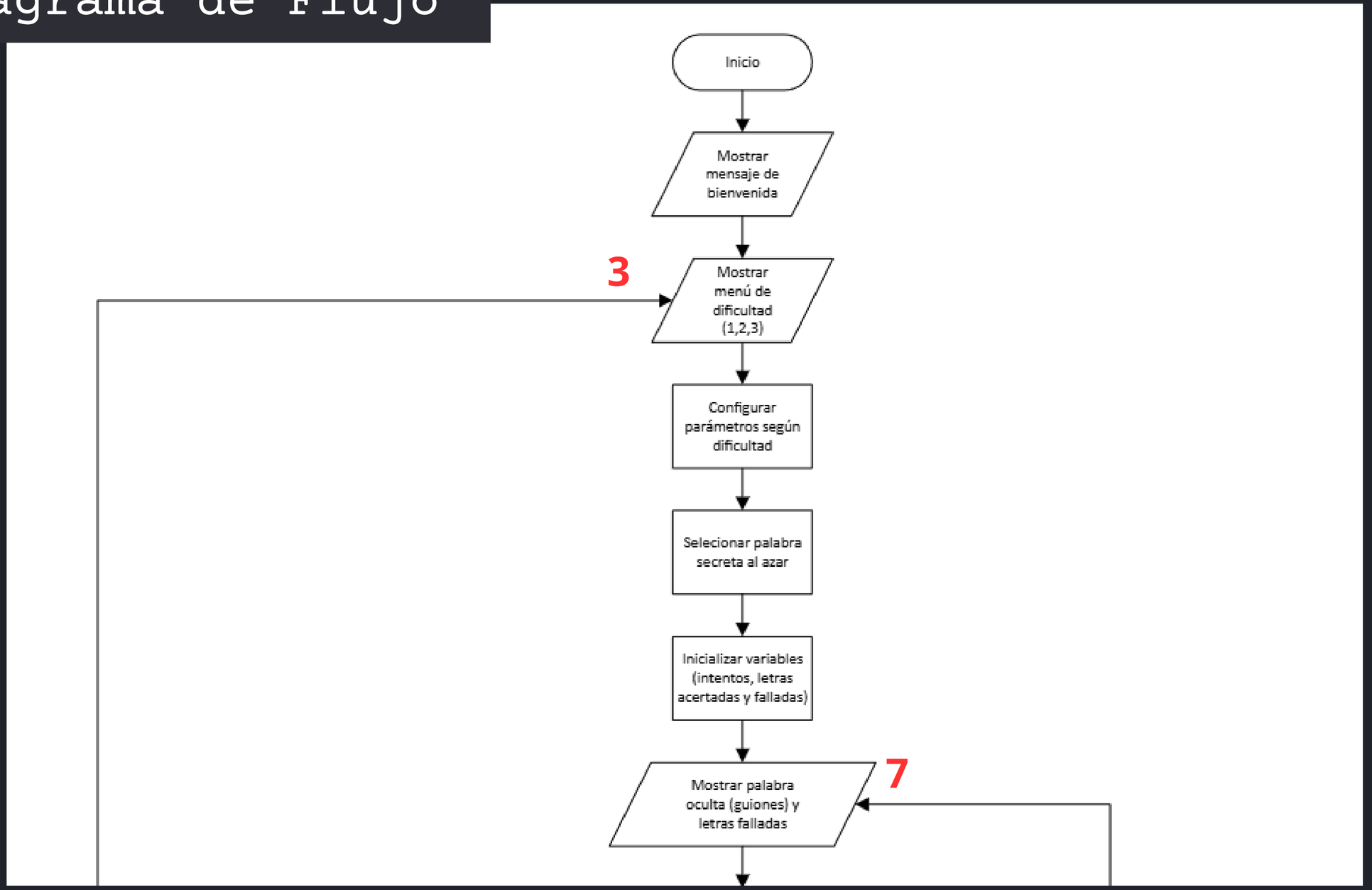
- Arquitectura en Capas
  - Presentación → Consola (interacción).
  - Lógica → Validación de intentos, comparación de letras, condiciones de victoria.
  - Datos → Listas de palabras.

- **Diagrama de Clases (UML)**
  - Juego: coordina la lógica.
  - Jugador: entradas del usuario.
  - Palabra: administra la palabra secreta.
  - Interfaz: muestra progreso y mensajes.

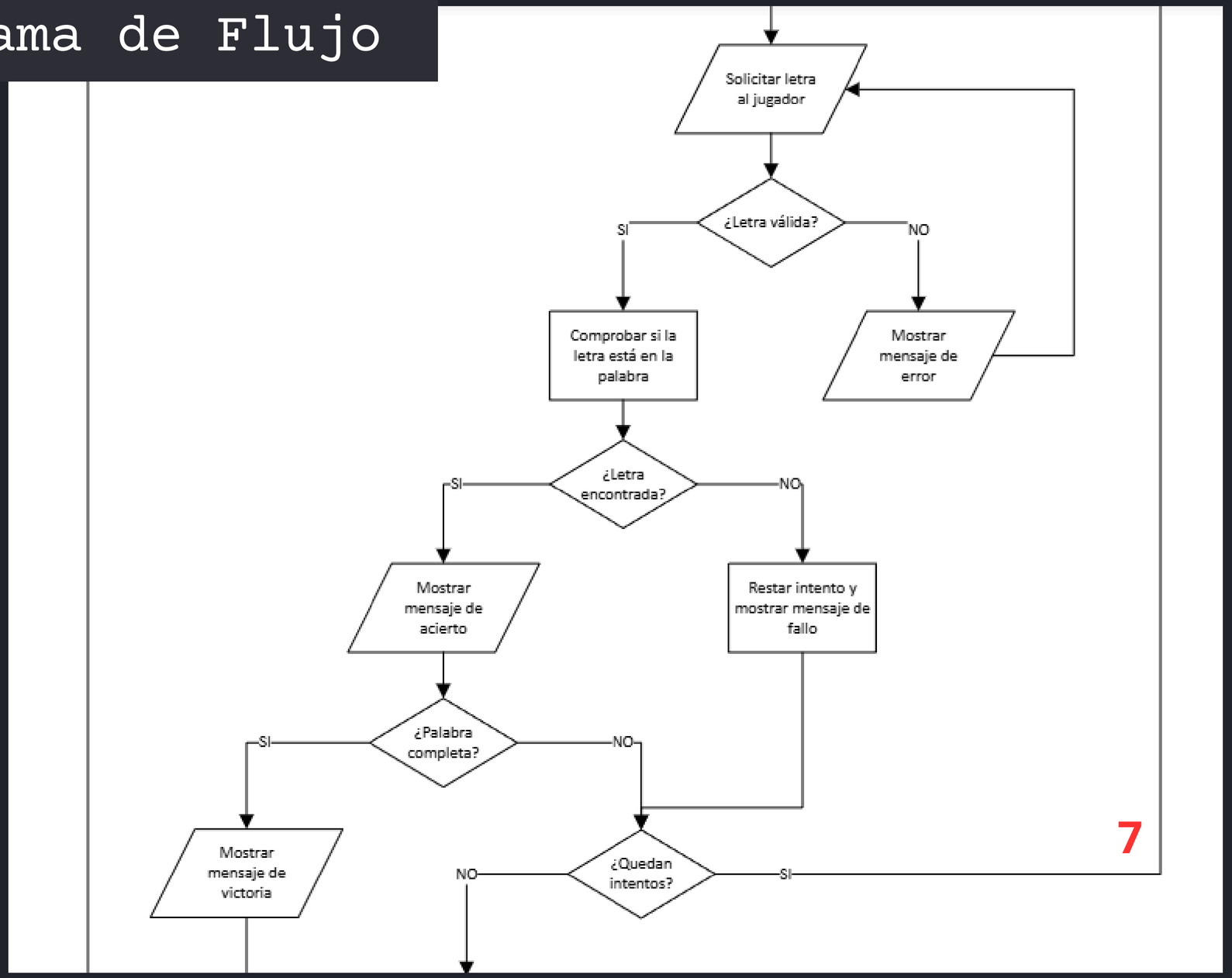
Ejemplo: Relación → Juego usa Palabra y Jugador.



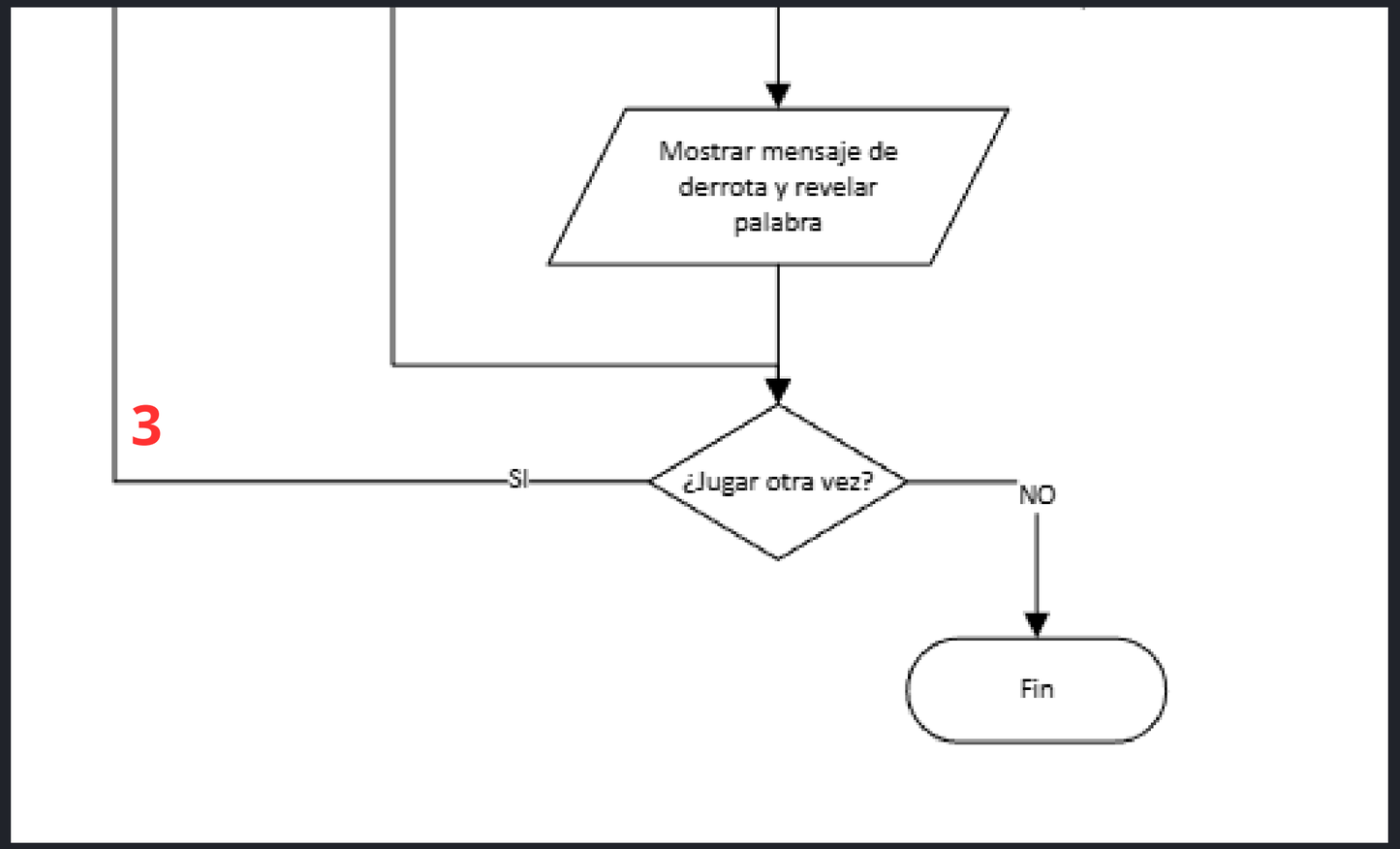
# Diagrama de Flujo



# Diagrama de Flujo



# Diagrama de Flujo



# Implementación en Código (Python)

Conceptos aplicados:

- Listas: palabras por nivel de dificultad.
- Set: letras adivinadas sin repetirse.
- Bucles while y for: repetir intentos y recorrer letras.
- Condicionales if-elif-else: validar entradas y definir victoria/derrota.
- Funciones: jugar_ahorcado() centraliza la lógica.

Ejemplo:

```
if letra_jugador in palabra_secreta:
    print("¡Acierto!")
else:
    intentos_restantes -= 1
```

# Funcionalidades del Juego

- Selección de dificultad (fácil, medio, difícil).
- Adivinanza de letras con validación de entradas.
- Visualización del progreso con guiones bajos.
- Límite de 6 intentos.
- Opción de reiniciar partida.

Ejemplo en ejecución:

```
Palabra: g _ t o
Intentos restantes: 3
```

# Conclusiones

Software desarrollado: Juego del Ahorcado

- Demuestra cómo los conceptos básicos de programación se integran en un sistema completo.
- Combina teoría (diagramas y arquitectura) con práctica (código ejecutable).
- Ejemplo ideal para aprender programación de manera divertida y aplicable.