UNIVERSIDAD INTERNACIONAL DEL ECUADOR

Escuela de Ingeniería en Ciberseguridad

Lógica de Programación

Aprendizaje Autónomo 1

Tema: Selección del Programa a desarrollar / Generación de Diagramas funcionales y Arquitectura de Software

Jorge Cano

Quito, 21 de Julio, 2025

Índice

Diseño del Juego del Ahorcado
Fase: Análisis y diseño
Producto Final: Juego del Ahorcado en lenguaje Python

1. Investigación de Diagramas de Funcionalidad y Arquitectura

Diagramas de Funcionalidad (UML)

Los diagramas de funcionalidad permiten visualizar los comportamientos y capacidades del sistema desde la perspectiva del usuario. Algunos de los principales tipos son:

- Diagrama de Casos de Uso (seleccionado): muestra los actores y las funcionalidades principales del sistema.
- Diagrama de Actividad: muestra flujos de trabajo entre actividades.
- Diagrama de Secuencia: muestra la interacción temporal entre objetos.
- Diagrama de Estados: muestra los estados por los que pasa un objeto.

Se seleccionó el diagrama de Casos de Uso, por su capacidad de representar claramente las funcionalidades del sistema desde el punto de vista del usuario.

Un diagrama de secuencia también podría resultar viable, pero la secuencia se representará posteriormente en el diagrama de flujo.

Diagramas de Arquitectura

Los diagramas de arquitectura permiten visualizar cómo se organiza internamente la aplicación a nivel de componentes, módulos o capas.

- Diagrama de Arquitectura en Capas (Layered Architecture) (seleccionada): separa la lógica en diferentes capas como presentación, lógica de negocio y datos.
- Diagrama de Componentes: muestra cómo se interconectan los componentes del sistema.
- Diagrama de Despliegue: muestra la distribución física del sistema (no es tan relevante aquí).
- Diagrama de Paquetes: agrupa clases/lógicas relacionadas.

Selección: es una forma clara y didáctica de mostrar la separación entre lógica de usuario, procesamiento y almacenamiento de datos, ideal para proyectos pequeños y educativos como este.

Tabla Comparativa entre diagrama funcional y diagrama de arquitectura

Característica	Diagrama Funcional	Diagrama de Arquitectura
Enfoque	Qué hace el sistema	Cómo está construido e implementado
Nivel	Lógico o de negocio	Técnico
Público objetivo	Stakeholders, analistas, diseñadores	Arquitectos, desarrolladores, DevOps
Etapa principal del proyecto	Análisis, diseño preliminar	Diseño detallado, implementación y operación
Ejemplos	Casos de uso, DFD, actividad	C4, componentes, despliegue, infraestructura

2. Resolución de Problemas (Análisis)

Objetivo: Adivinar una palabra secreta letra por letra con un número limitado de intentos. Si se supera el límite de errores, el jugador pierde.

Análisis del problema

Paso	Descripción
1. Comprender el problema	Simular el juego clásico de "El Ahorcado" mediante la interacción con el usuario.
2. Entradas	Palabra secreta (aleatoria), letras ingresadas por el jugador.
3. Procesos	Validación de letras, actualización del estado del juego, conteo de errores.
4. Salidas	Estado actual de la palabra, mensajes de victoria o derrota, número de intentos restantes.
5. Reglas	Hasta 6 errores permitidos, no se pueden repetir letras, letras deben ser válidas.

3. Diseño de Funcionalidades (Casos de Uso)

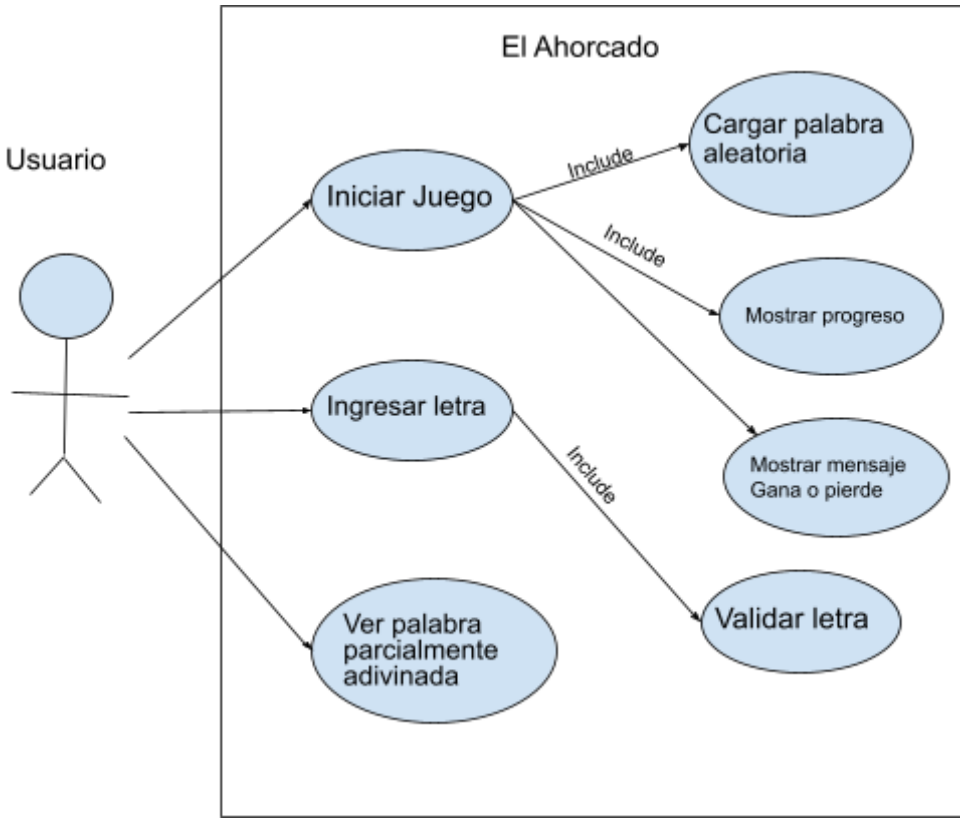
Actor: Jugador - usuario

Casos de Uso:

- Iniciar juego
- Seleccionar palabra aleatoria
- Ingresar letra
- Validar letra
- Mostrar progreso
- Actualizar intentos
- Mostrar mensaje final (ganar/perder)
- Reiniciar juego

Relaciones: Cada acción que realiza el jugador influye en la lógica del juego. El juego realiza decisiones en base a la entrada proporcionada por el usuario, como lo es: validar letras o mostrar progreso.

Diagrama de Casos de Uso



4. Diagrama de arquitectura (en capas)

Este tipo de diagrama mostrará cómo se organiza el software internamente. Las **tres capas** básicas que se usarán:

Capas:

1. **Capa de Presentación (Interfaz)**
    - Muestra menús y mensajes al jugador
    - Solicita letras
    - Muestra progresivamente el dibujo del ahorcado
2. **Capa de Lógica de Negocio**
    - Verifica si la letra está en la palabra
    - Lleva el conteo de errores
    - Verifica condiciones de victoria o derrota
3. **Capa de Datos**
    - Lista de palabras posibles
    - Historial de letras usadas
    - Símbolos utilizados para gráficos del ahorcado

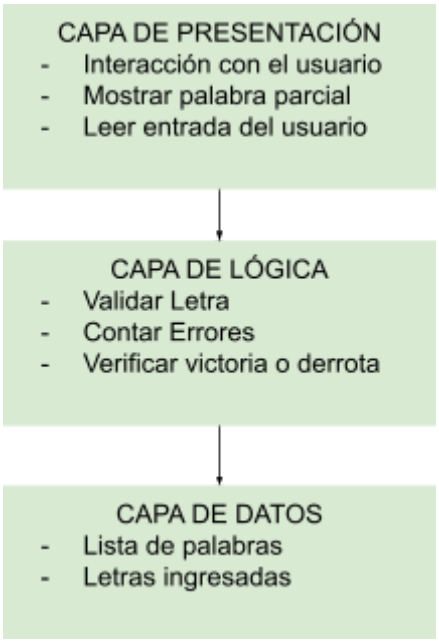
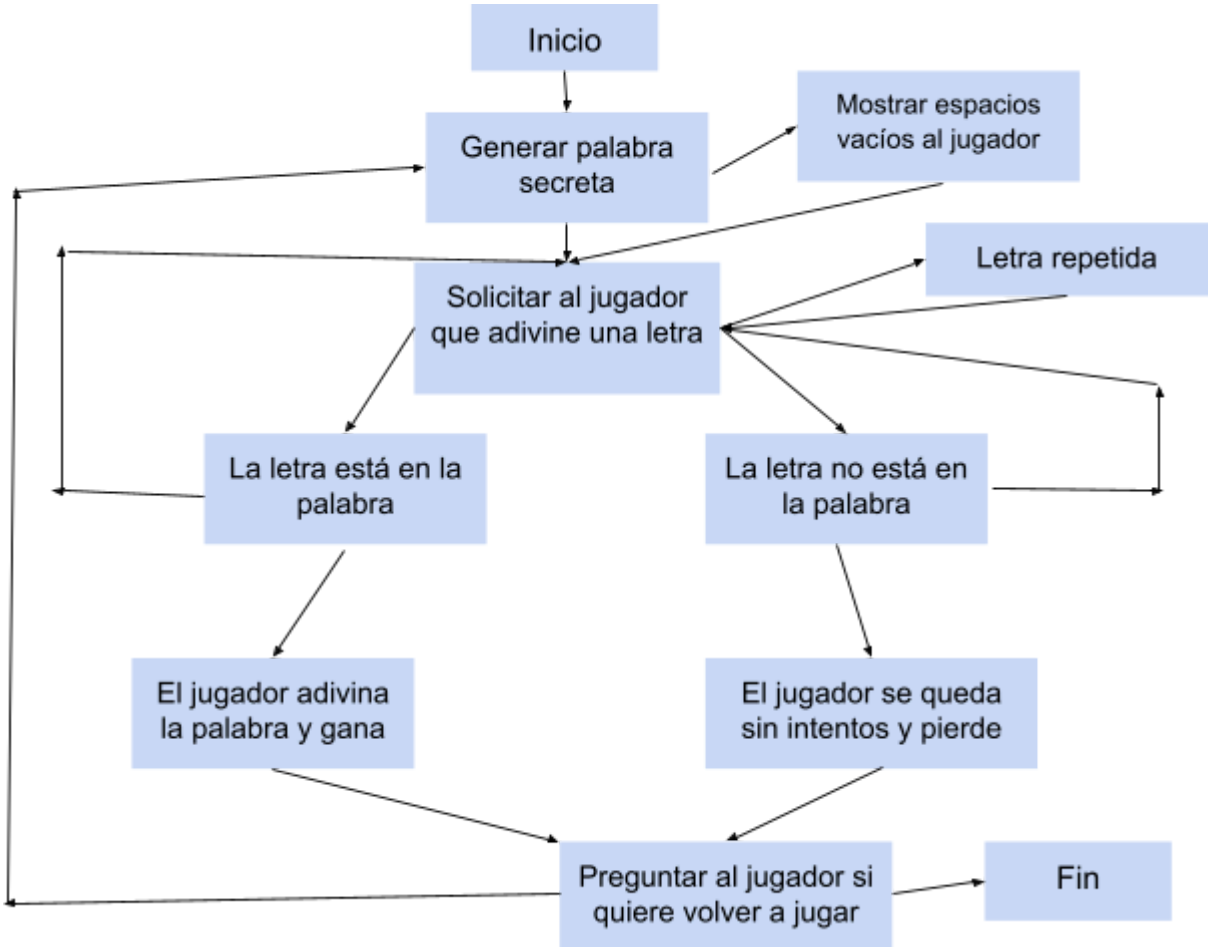


Diagrama de Flujo (Pseudo diagrama)



Conclusiones

1. La identificación temprana de funcionalidades mediante diagramas de casos de uso facilita una planificación estructurada del desarrollo
El uso de un diagrama de casos de uso permitió identificar y modelar de forma clara las interacciones clave entre el jugador y el sistema. Esta representación funcional sirvió como base para reducir ambigüedades y asegurar la trazabilidad de las funciones durante el desarrollo.

2. La arquitectura en capas garantiza modularidad, escalabilidad y mantenibilidad del sistema**.** Al separar la interfaz de usuario, la lógica del juego y el almacenamiento de datos (palabras), se estableció un diseño desacoplado. Esta estructura permite que cada capa pueda evolucionar de forma independiente, mejora la legibilidad y reutilización del código.

3. La resolución sistemática del problema permite traducir un desafío lógico en una solución computacional robusta

Al hacer un análisis de entradas, procesos y salidas se puede abstraer el comportamiento del juego en términos computacionales. Esto facilita la implementación del algoritmo en código, asegurando que el flujo lógico esté completamente definido antes de escribir una sola línea de código.

Enlace a video por medio de un browser:

file:///Users/jorgecano/Downloads/Grabando-20250722_234728.webm

(Por favor copiar y pegar el enlace en un browser, por problemas de conectividad no se pudo guardar el video dentro del entorno office)

Bibliografía

UML Use Case Diagram Tutorial, February 7th, 2018, Lucid Software:

https://www.youtube.com/watch?v=zid-MVo7M-E

Tipos de Diagramas UML:

https://www.lucidchart.com/blog/es/tipos-de-diagramas-uml

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Python Software Foundation. (2023). *The Python Tutorial*. https://docs.python.org/3/tutorial/

Lucidchart. (n.d.). *What is a use case diagram?*. https://www.lucidchart.com/pages/uml-use-case-diagram

Edraw Software. (2024). *Cómo diseñar un diagrama de arquitectura de software*. https://www.edrawsoft.com/es/how-to-design-software-architecture-diagram.html