Universidad Tecnológica de La Habana

“José Antonio Echeverría”



Facultad de Ingeniería Informática

**CUJAPSE**

*Tarea Final de Estructura de Datos*

*Entrega 1*

**Equipo #1:**

**Luis Alberto Pérez Alvarez**

**Alison Hidalgo Guerra**

**Ryan Negrete Menchaca**

**Rolando del Barrio Benítez**

**Patricia Tomé Romero**

**Maikel Alejandro García Bolivar**

**José Alberto Cabrera Morejón**

La Habana, Octubre 2025

# 1- Descripción del juego

Cujapse es un juego breve y humorístico que exagera situaciones comunes de la vida universitaria. El jugador interpreta a un estudiante de primer año de la facultad de informática y debe sobrevivir hasta el fin de año académico tomando decisiones que afectan cuatro estadísticas: **dinero**, **popularidad**, **estudios** y **consumo de cafeína**. Las decisiones no vienen con explicaciones directas, por lo que el jugador aprende por prueba y error. Los eventos se presentan en orden aleatorio y no se repiten dentro de la misma partida; el sistema guarda qué eventos ya ocurrieron para evitarlos en esa sesión.

**Premisa y tono**

* **Premisa:** Llevar al extremo y con humor las experiencias típicas del primer año universitario, transformando pequeñas decisiones en consecuencias inmediatas sobre la vida del estudiante.
* **Tono:** Irreverente, satírico y amable; objetivo: provocar identificación y risa más que crítica dura.
* **Objetivo emocional:** Que jugadores que vivieron situaciones similares recuerden anécdotas y se diviertan explorando resultados distintos mediante decisiones.

**Mecánicas centrales**

* **Personaje jugable:** Estudiante de primer año de informática.
* **Duración de la partida:** Un año académico dividido en turnos (por ejemplo, semanas o eventos).
* **Decisiones:** En cada turno el jugador elige entre 2 opciones; cada elección aplica cambios a las cuatro estadísticas.
* **Estadísticas:**
  + **Dinero** — recursos económicos del jugador.
  + **Popularidad** — aceptación social y redes dentro de la facultad.
  + **Estudios** — rendimiento académico y preparación.
  + **Consumo de cafeína** — energía, salud y dependencia de café.
* **Regla de equilibrio:** Ninguna estadística puede llegar a 0 ni a 100. Si una estadística se acerca a esos límites, se muestran consecuencias progresivas que advierten al jugador. El desafío es mantenerlas en equilibrio; nada es bueno en exceso ni en falta.
* **Aprendizaje por jugador:** No se muestran los efectos exactos de cada opción; el jugador descubre qué modifica cada estadística probando.

# 2- Estructuras de datos a utilizar

Para este juego se tiene pensado utilizar un árbol de decisión (Especialización del árbol general)como estructura base, ya que cada evento está compuesto por una cadena de situaciones en las que el jugador puede tomar diferentes caminos y obtener resultados variados. Esta lógica permite modelar cada evento como un conjunto de decisiones encadenadas, donde cada elección conduce a nuevas situaciones o consecuencias, manteniendo la narrativa dinámica y el sistema de estadísticas en constante evolución.

Para la gestión de los eventos y escenarios del juego, se utilizan estructuras tipo ArrayList, tanto para los objetos de tipo **Escenario** como para los de tipo **Evento**. Esta elección se debe a que el acceso a estos elementos se realiza principalmente mediante consultas, sin necesidad de realizar inserciones dinámicas durante la ejecución. El uso de listas lineales permite una recuperación rápida y ordenada de los elementos, optimizando el rendimiento en tiempo de juego.

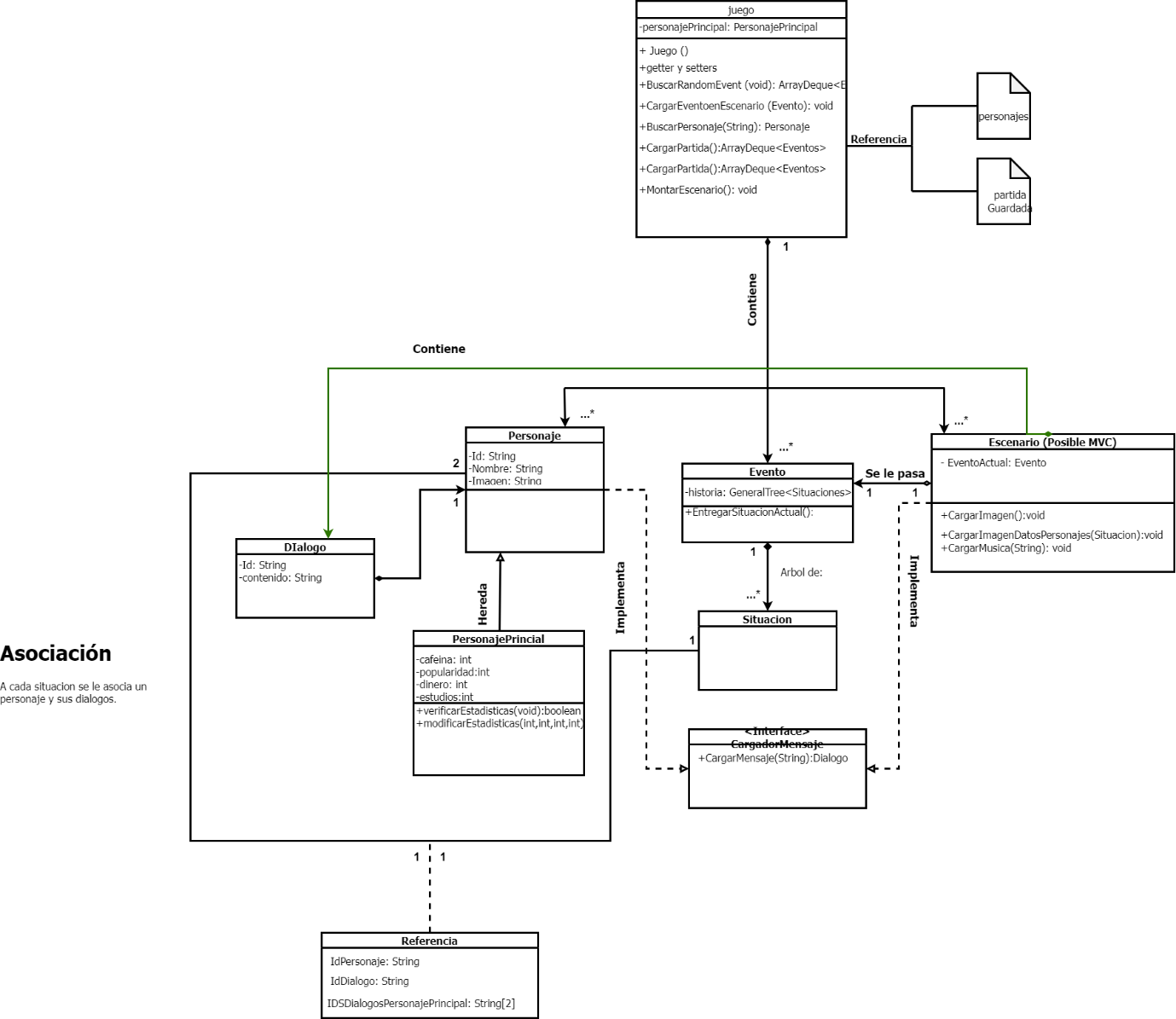
Por otro lado, los **personajes** del juego se almacenan en **archivos binarios**. Esta decisión responde a dos factores clave:

1. Los personajes contienen **diálogos** y **imágenes** asociadas, lo que implica un mayor volumen de datos en comparación con los eventos o escenarios.
2. Para **reducir el uso de recursos** y evitar mantener en memoria estructuras pesadas, se opta por guardar cada personaje en un fichero binario individual, al que se accede únicamente cuando se necesita cargar dicho personaje en el juego.

Este enfoque permite mantener la eficiencia del sistema, minimizar el consumo de memoria y facilitar la escalabilidad del contenido narrativo y visual del juego.

Además de modelar los eventos como árboles de decisión, se contempla el uso de una **cola de eventos** para gestionar su ejecución durante la partida. Una vez que los eventos son **aleatorizados y seleccionados** desde el ArrayList, se almacenan en una cola que permite **cargarlos uno por uno en el orden en que fueron elegidos**. Esta estrategia simplifica la lógica de carga, ya que cada evento se procesa secuencialmente sin necesidad de búsquedas adicionales ni reordenamientos dinámicos.

# 3- Diagrama de clases

Esta es una primera vista de lo que se tiene pensado hacer para este proyecto:

# 4- Ideas para la interfaz

Esto también es un boceto de lo que se tiene pensado hacer para la interfaz

