Campus Santa Fe

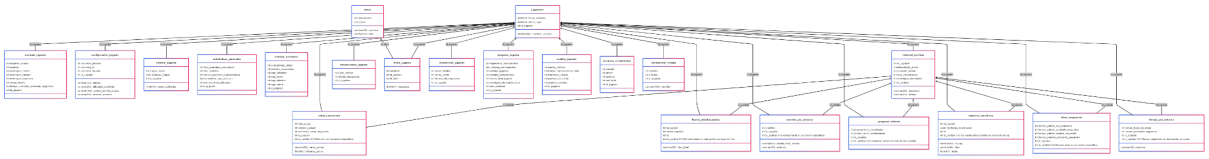
2. Ejercicio de Modelación de Base de Datos del reto

Construcción de software y toma de decisiones (Gpo 501)

Katia Abigail Álvarez Contreras
Emiliano Delgadillo Osorio
Ángel Montemayor Dávila

5 - mayo / 2025

Modelo Entidad-Relación:

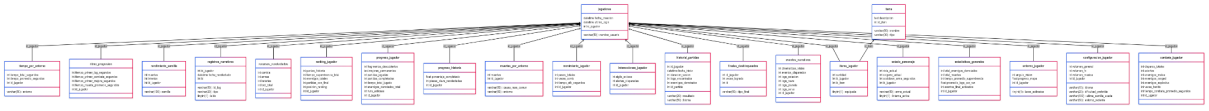


Justificación:

El modelo Entidad Relación que vemos en la parte de arriba es el que proponemos para gestionar de manera integral toda la información de nuestro videojuego, en nuestro modelo tenemos como entidad principal “jugadores” que tienen como atributos la fecha de creación, el último login y su identificador para reconocerlo, desde esta entidad se relacionan múltiples tablas que almacenan estadísticas, progreso, interacciones, inventario, entre otras cosas.

Nuestro modelo usa llaves foráneas como id_jugador e id_partida, para relacionarse entre tablas, pero también permite registrar datos tanto a nivel histórico como por sesión individual, lo que facilita las consultas de la misma manera.

Originalmente, teníamos el modelo de Entidad Relación que se ve a continuación:



Pero después de analizarlo añadimos más conexiones que se ven en nuestro modelo final a traves de id_partida y historial_partidas, para que rastrear datos a nivel de sesiones individuales fuera accesible, además de facilitar consultas temporales, ya sea estadística por sesión o comparaciones entre partidas y finalmente el garantizar que elementos como progreso, estado del personaje y eventos estén contextualizados.

Hay un total de 22 entidades:

- jugadores
- combate_jugador

- configuracion_jugador
- entorno_jugador
- estadisticas_generales
- estado_personaje
- eventos_narrativos
- finales_desbloqueados
- historial_partidas
- interacciones_jugador
- items
- items_jugador
- movimiento_jugador
- muertes_por_entorno
- progreso_historia
- progreso_jugador
- ranking_jugador
- recursos_recolectados
- registros_narrativos
- rendimiento_semilla
- ritmo_progresion
- tiempo_por_entorno

Pensamos nuestro modelo para que no hubiera redundancia ni duplicaciones, por otro lado, todas las entidades se vinculan mediante id_jugador como mencionamos anteriormente y así formar relaciones y cardinalidades.

Relaciones 1:1

Relación	Justificación
jugadores → configuracion_jugador	Cada jugador tiene solo una configuración actual.
jugadores → entorno_jugador	Solo se guarda el entorno activo del jugador.
jugadores → estadisticas_generales	Estadísticas acumuladas del jugador, no por partida.
jugadores → progreso_jugador	Progreso total del jugador en todos sus juegos.
jugadores → ranking_jugador	Un único lugar en el ranking general.

Relaciones 1:N

Relación	Justificación
jugadores → historial_partidas	Un jugador puede tener muchas partidas.
jugadores → items_jugador	Un jugador puede poseer muchos ítems.
jugadores → combate_jugador	Puede haber múltiples registros de combate (por partida).
jugadores → estado_personaje	Se puede registrar el estado del personaje por sesión.
jugadores → eventos_narrativos	Se pueden registrar múltiples eventos activados.
jugadores → finales_desbloqueados	Un jugador puede desbloquear distintos finales.
jugadores → interacciones_jugador	Se pueden registrar múltiples acciones de sigilo/alertas.
jugadores → movimiento_jugador	Un jugador puede tener varias estadísticas de movimiento.
jugadores → muertes_por_entorno	Puede morir en varios entornos diferentes.
jugadores → progreso_historia	Se puede registrar el progreso por sesión.
jugadores → recursos_recolectados	Puede recolectar recursos en distintas sesiones.
jugadores → registros_narrativos	Puede haber leído varios logs.
jugadores → rendimiento_semilla	Puede haber probado varias semillas.
jugadores → ritmo_progresion	Se mide el ritmo por sesión/run.
jugadores → tiempo_por_entorno	Juega en distintos entornos durante diferentes runs.

Relaciones N:M

Relación	Tabla intermedia
jugadores ↔ items	items_jugador

Un jugador puede tener múltiples ítems, y un ítem puede pertenecer a múltiples jugadores.

De igual manera podemos clasificar por sesiones varias entidades, para lo cual usamos historial_partidas, por ejemplo:

Relaciones 1:N

Relación	Justificación
historial_partidas → estado_personaje	Registrar cómo termina cada sesión.
historial_partidas → progreso_historia	Seguir progreso narrativo por run.
historial_partidas → finales_desbloqueados	Asociar el final con la partida que lo activó.
historial_partidas → registros_narrativos	Log narrativo recolectado en una partida.
historial_partidas → muertes_por_entorno	En qué entornos murió en esa sesión.
historial_partidas → tiempo_por_entorno	Tiempo que pasó en cada entorno por sesión.
historial_partidas → ritmo_progresion	Medir tiempos clave durante una sesión.

Finalmente, consideramos que muestreo modelo es escalable, o sea que en cualquier momento podríamos implementar nuevas entidades y la lógica no se perdería, además nos ayuda a hacer nuestros análisis de una manera más sencilla, cumplimos con los principios de que no haya redundancia y que estén bien definidos, pero sobre todo lo más importante es que nos permite trazar los datos por sesión sin perder la vista global en caso de que lo necesitemos. También cabe aclarar que este es nuestro primer planteamiento del modelo, con la retroalimentación que recibamos y conforme al desarrollo del juego algunas cosas podrían cambiar o mejorar, aunque es una idea bastante solida para minimizar los cambios a futuro.