Campus:

Santa Fe

Materia:

Construcción de software y toma de decisiones

Nombre de la actividad:

Modelación de Base de Datos del reto

Nombre y Matrícula:

Rebeca Davila Araiza A01029805
Darío Cuauhtémoc Peña Mariano A01785420
Santiago Arista Viramontes A01028372

Profesor:

Esteban Castillo Juarez

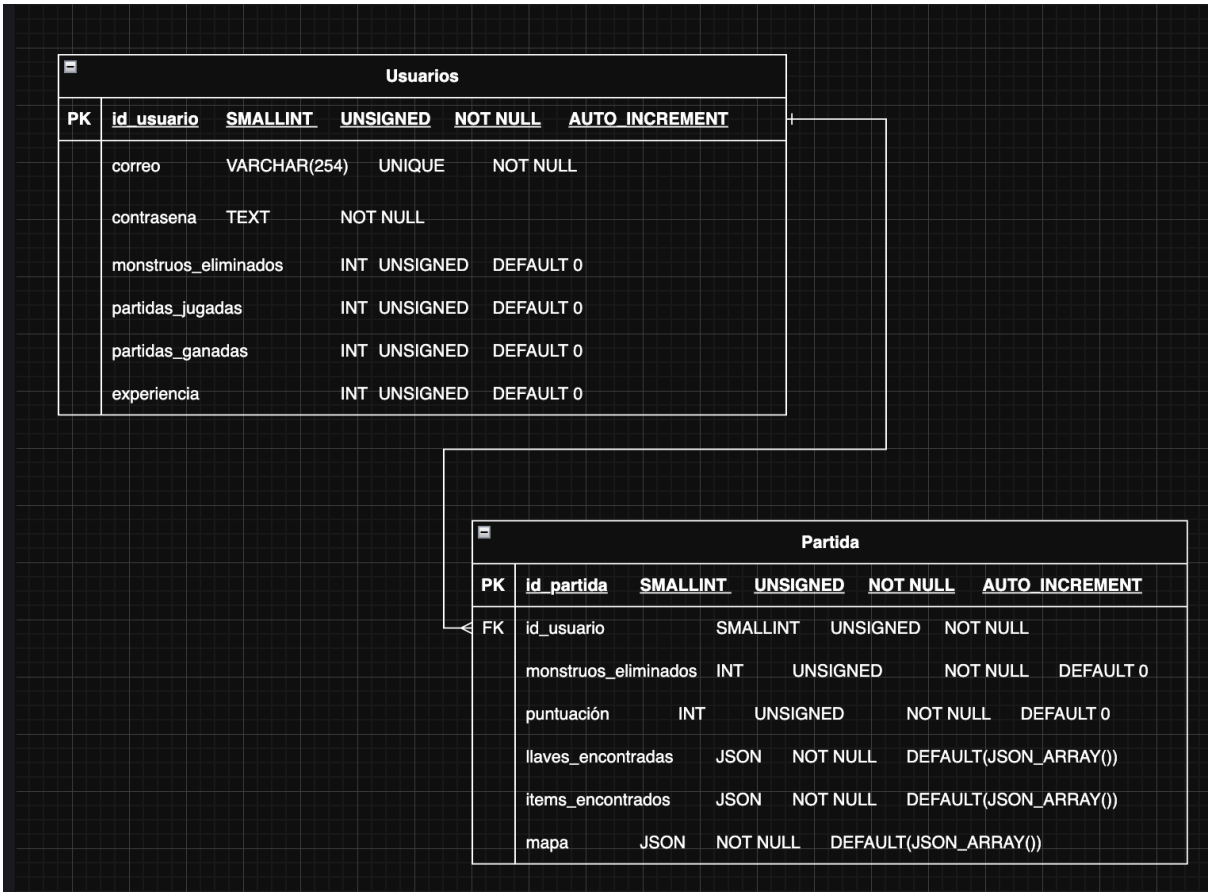
Grupo:

601

Fecha:

31/03/2025

Modelo Entidad-Relación Base de datos del reto



Justificación del modelo

NOTA: ES IMPORTANTE RECALCAR QUE LA BASE DE DATOS SE ACTUALIZARÁ TRAS HABER RESUELTO DUDAS CON EL PROFESOR PARA QUE LA MISMA CUMPLA CON LAS PRIMERAS 3 FORMAS NORMALES.

Para el diseño de la base de datos de Nineshions, decidimos adoptar un modelo que optimizara la gestión de la información relevante para el progreso de los jugadores sin sobrecargar la base de datos con detalles que pueden ser manejados dentro del juego. Este modelo de base de datos equilibra eficiencia y flexibilidad, permitiendo almacenar datos clave sin sobrecargar la estructura con información que puede ser procesada directamente en el entorno del juego.

Estructura y Justificación del Modelo

- **Almacenamiento del Mapa y los Cuartos dentro de la Partida**

  En lugar de crear una estructura relacional detallada para los cuartos y su distribución dentro del juego, optamos por almacenar el mapa en un campo de tipo JSON dentro de la tabla Partida. Esto nos permite mayor flexibilidad en la

representación del entorno del juego y facilita la carga dinámica de los escenarios sin requerir consultas excesivas a la base de datos.

- **Gestión de Progreso del Usuario**

  La tabla Usuario almacena datos relevantes sobre el rendimiento del jugador, como el número de monstruos eliminados, partidas jugadas, partidas ganadas y experiencia acumulada. Estos valores permiten generar estadísticas de progreso sin necesidad de registrar cada acción en detalle, lo que optimiza el rendimiento de la base de datos.

- **Registro de Partidas**

  La tabla Partida almacena el estado de cada sesión de juego, incluyendo las llaves e ítems encontrados (en formato JSON para una mayor flexibilidad), el número de monstruos eliminados y la puntuación obtenida. Además, se mantiene una relación con el usuario mediante una clave foránea, lo que permite vincular las partidas con los jugadores de manera eficiente.

- **Delegación de Lógica al Juego**

  La base de datos está diseñada para almacenar únicamente la información esencial del usuario y sus partidas, mientras que la lógica detallada de generación y evolución de cuartos, mecánicas de combate y eventos especiales se manejará directamente dentro del motor del juego. Esto garantiza un mejor rendimiento y una experiencia más fluida para el usuario.

Justificación y Explicación de las Tablas

Usuarios:

Esta tabla almacena la información principal de cada jugador, permitiendo registrar su progreso general en el juego sin necesidad de guardar detalles innecesarios. Cada usuario es identificado mediante un id_usuario, y se mantienen estadísticas clave como partidas jugadas y experiencia acumulada.

Estructura y Explicación:

id_usuario: SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY

Identificador único de cada usuario.

correo: VARCHAR(254) NOT NULL UNIQUE

Correo electrónico del usuario, usado para autenticación.

contrasena: TEXT NOT NULL

Almacena la contraseña del usuario de forma encriptada.

monstruos_eliminados: INT UNSIGNED DEFAULT 0

Contador de los monstruos eliminados en todas las partidas jugadas.

partidas_jugadas: INT UNSIGNED DEFAULT 0

Número total de partidas jugadas por el usuario.

partidas_ganadas: INT UNSIGNED DEFAULT 0

Número total de partidas ganadas por el usuario.

experiencia: INT UNSIGNED DEFAULT 0

Cantidad de experiencia acumulada por el usuario.

Partida:

Cada vez que un usuario juega una partida, se genera un registro en esta tabla. Aquí se almacenan estadísticas de la sesión, como los monstruos eliminados, la puntuación obtenida y la información del mapa en formato JSON, lo que permite mayor flexibilidad en la estructura de los niveles.

Estructura y Explicación:

id_partida: SMALLINT UNSIGNED AUTO_INCREMENT PRIMARY KEY

Identificador único de la partida.

id_usuario: SMALLINT UNSIGNED NOT NULL

Referencia al usuario que jugó la partida.

monstruos_eliminados: INT UNSIGNED DEFAULT 0

Cantidad de monstruos eliminados en esta partida.

puntuacion: INT UNSIGNED DEFAULT 0

Puntuación obtenida en la partida.

llaves_encontradas: JSON NOT NULL DEFAULT (JSON_ARRAY(false, false, false, false, false, false, false, false, false))

Registro de las llaves encontradas en la partida.

items_encontrados: JSON NOT NULL DEFAULT (JSON_ARRAY(false, false, false))

Registro de los ítems encontrados en la partida.

mapa: JSON NOT NULL DEFAULT (JSON_ARRAY())

Información del mapa y su disposición. Almacenado dentro de la partida para no perder el mapa si el jugador sale del juego durante la partida.

Relaciones:

- Partida(id_usuario) está vinculado a Usuario(id_usuario), asegurando que cada partida pertenece a un usuario registrado.
- Si un usuario es eliminado, sus partidas se eliminan (ON DELETE CASCADE), evitando la pérdida de datos.