**Nombres:**

Julian Diaz Ramirez

Andres Felipe Puentes

Juan David Muñoz

Sebastian Vega

## Introducción

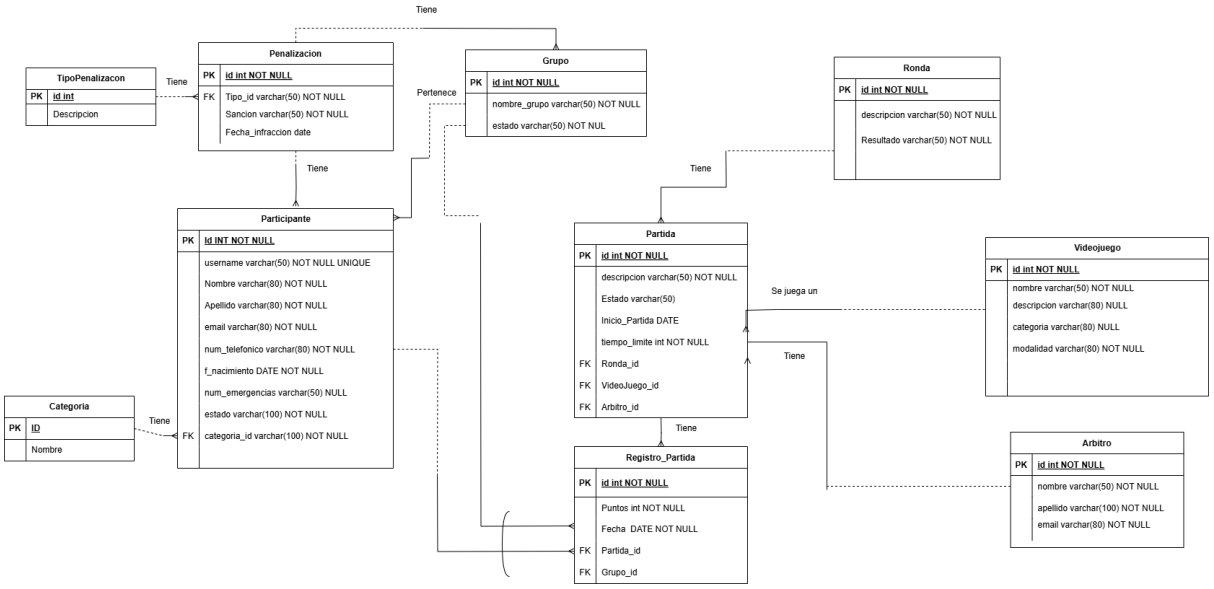
Nuestro enfoque se basa en un diseño cuidadosamente estructurado que cumple con las necesidades del torneo, garantizando la escalabilidad, seguridad y flexibilidad necesarias para adaptarse a diferentes modalidades y reglas de competencia. Estamos listos para mostrarles cómo nuestra propuesta asegura un sistema optimizado y preparado para los desafíos del desarrollo de videojuegos en la era moderna.

## Reglas de Negocio

1. Cada jugador debe inscribirse con un nombre, apellido, correo electrónico, identificación y un username único. No se permite usernames repetidos ni que el usuario pueda volver a registrarse si ya ha sido dado de baja o descalificado. El usuario que es menor de edad tendrá que tener un número de contacto de un acudiente.
2. Los jugadores no podrán inscribirse simultáneamente en la categoría grupal y en la categoría individual. Solo se permitirá la inscripción en una de las modalidades.
3. Para la modalidad de equipos, cada equipo deberá estar conformado por un mínimo de 3 personas. Deben elegir un nombre único que no se repita entre los ya registrados. Un jugador solo puede pertenecer a un equipo durante todo el torneo. Además, cada equipo deberá contar con un líder o capitán registrado para poder participar en el torneo. Este rol podrá cambiar entre miembros del equipo a lo largo de las rondas.
4. El torneo tendrá varias rondas: clasificatoria, semifinales y finales. Los jugadores o equipos avanzarán o serán eliminados según los resultados de cada ronda.
5. Cada ronda será organizada para un videojuego específico que se asignará de manera aleatoria. En cada ronda, se cambiará de videojuego dentro de la misma categoría. Por ejemplo, si están jugando un videojuego de fútbol, en la siguiente ronda jugarán otro videojuego deportivo, como uno de baloncesto.
6. Los árbitros o jueces deberán calificar el desempeño de los jugadores y equipos en cada ronda, lo que puede afectar su reputación.
7. El sistema debe mantener un registro de todos los partidos jugados en el torneo, incluyendo la información sobre los participantes, el videojuego utilizado y el resultado final de cada partida.
8. En caso de empate en una ronda, se jugará una partida adicional para definir al equipo o jugador que avanza. Las reglas del videojuego o del torneo definirán los criterios de desempate.

9. El sistema debe permitir la asignación de penalizaciones o descalificaciones a jugadores o equipos por incumplimiento de las reglas del torneo. Las penalizaciones podrán incluir la pérdida de puntos, descalificación de una ronda o la eliminación completa del torneo. Además, las penalizaciones deberán reflejarse en el estado del jugador o equipo.
10. El sistema debe permitir la asignación de árbitros para supervisar las partidas importantes, como semifinales y finales. Los árbitros tendrán la autoridad para detener la partida en caso de irregularidades, y su decisión será definitiva. Habrá un juez por partida.
11. La reputación de un jugador se acumulará en función de su comportamiento durante el torneo. Cada vez que un jugador abandone una partida, insulte a otros jugadores o cometa infracciones según el reglamento, se le restarán puntos de reputación. Las infracciones reiteradas podrán llevar a sanciones más graves, como la descalificación o la imposibilidad de inscribirse en futuros torneos. El sistema deberá registrar la fecha, tipo de infracción y la sanción aplicada.
12. Si un jugador o equipo abandona un partido o se rinde antes de que finalice, el sistema debe registrar esto y aplicar una penalización en puntos o avanzar automáticamente al oponente al siguiente nivel
13. Cada partida tendrá un tiempo máximo asignado. Si al final del tiempo no se ha definido un ganador, el resultado se determinará según las reglas específicas del videojuego o de la ronda en cuestión (por ejemplo, el equipo o jugador con más puntos acumulados al final del tiempo).
14. En cada modalidad, ya sea individual o grupal, el sistema proporcionará una ventaja inicial en puntos de manera aleatoria. Estos puntos de ventaja no estarán relacionados con el desempeño anterior del jugador o equipo, y su objetivo es generar mayor imprevisibilidad en las partidas.
15. Cada equipo podrá inscribir uno o más jugadores sustitutos antes de la primera ronda. Estos jugadores podrán reemplazar a los titulares en caso de necesidad, pero solo antes del inicio de una partida. Cada cambio de jugador titular por sustituto deberá registrarse y notificar a los organizadores del torneo. Los sustitutos deben cumplir con los requisitos de inscripción y no podrán pertenecer a más de un equipo.
16. Cada árbitro debe registrar su nombre, apellido, correo electrónico y tarjeta profesional que lo avale como experto en torneos de videojuegos.

**MODELO ENTIDAD RELACIÓN**




//Describir los cambios y modificaciones o correcciones que se realizaron del MER

## 1. Las categorías como entidades, no atributos

Crear una nueva entidad llamada `Categoria` que permita almacenar los diferentes tipos de categorías (por ejemplo, "individual", "grupal") en lugar de tenerlo como un atributo en otras tablas.

## 2. Tipo de penalización como entidad débil, no atributo

Crear una entidad `TipoPenalizacion` que contenga las diferentes infracciones posibles. Esta tabla puede ser una entidad débil de `Penalizacion`.

## 3. Revisión de la relación entre rondas y partidas

La relación entre `Ronda` y `Partida` debe ser más clara. La relación actual no especifica si una ronda contiene varias partidas o si una partida es única para una ronda.

- **Implementación:**
  - Si una `Ronda` puede tener múltiples `Partida`, se debe agregar una clave foránea `ronda_id` en `Partida` que apunte a `Ronda`, indicando que una ronda puede contener múltiples partidas.
  - Alternativamente, si una partida solo pertenece a una ronda específica y no se reutiliza, mantener la relación actual pero aclarar su uso.

## 4. Revisión de la relación entre participantes y grupos para evitar redundancia

La relación de pertenencia a grupo debe evitar redundancias y solo debe existir cuando un participante realmente está en un grupo.

- **Implementación:**
  - Verificar que en `Participante` no se incluya información redundante sobre pertenencia a grupo (por ejemplo, utilizando un campo booleano para indicar si pertenece a un grupo).
  - En lugar de duplicar información, establecer una relación directa desde `Grupo` hacia `Participante` para indicar miembros. La tabla intermedia puede ser opcional si no se requiere una relación extra detallada.

**Triggers**

- Validación de tiempo máximo

  Termina automáticamente una partida si se alcanza el tiempo límite.

- Asignación de puntos de ventaja

  Asigna puntos iniciales de manera aleatoria al inicio de cada partida, asegurándose de que no se repitan entre participantes.

Regla de Negocio asociada:

En cada modalidad, ya sea individual o grupal, el sistema proporcionará una ventaja inicial en puntos de manera aleatoria. Estos puntos de ventaja no estarán relacionados con el desempeño anterior del jugador o equipo, y su objetivo es generar mayor imprevisibilidad en las partidas.

*//Código*

- Validación de datos obligatorios:

  Si el jugador es menor de edad, asegurar que se registre el número de contacto de un acudiente

- Control de partidas empatadas

  Detecta automáticamente partidas que terminan en empate y genera una nueva partida con los mismos participantes para desempatar.

**CONSULTAS COMPLEJAS Y SUBCONSULTAS**

## Consultas complejas

**Lista de participantes sancionados con detalle de la infracción:**
sql
Copiar código

```
SELECT
    p.username,
    p.Nombre,
    p.Apellido,
    pen.Sancion,
    pen.Fecha_infraccion,
    tp.Descripcion AS Tipo_Infraccion
FROM
    Participante p
INNER JOIN
    Penalizacion pen ON p.id = pen.id
INNER JOIN
    TipoPenalizacion tp ON pen.Tipo_id = tp.id;
```

1.

**Resultados de partidas por grupo y videojuego:**
sql
Copiar código

```
SELECT

    g.nombre_grupo AS Grupo,

    v.nombre AS Videojuego,

    rp.Puntos,

    rp.Fecha

FROM

    Registro_Partida rp

INNER JOIN

    Grupo g ON rp.Grupo_id = g.id

INNER JOIN

    Partida pa ON rp.Partida_id = pa.id

INNER JOIN

    Videojuego v ON pa.VideoJuego_id = v.id;
```

2.

**Cantidad de partidas jugadas en cada ronda:**
sql
Copiar código

```
SELECT

    r.descripcion AS Ronda,

    COUNT(pa.id) AS Total_Partidas

FROM

    Ronda r

INNER JOIN

    Partida pa ON r.id = pa.Ronda_id
```

```
GROUP BY

    r.descripcion;
```

3.

**Ranking de grupos con mayor puntaje acumulado:**
sql
Copiar código

```
SELECT

    g.nombre_grupo AS Grupo,

    SUM(rp.Puntos) AS Puntaje_Total

FROM

    Registro_Partida rp

INNER JOIN

    Grupo g ON rp.Grupo_id = g.id

GROUP BY

    g.nombre_grupo

ORDER BY

    Puntaje_Total DESC;
```

4.

**Participantes y su grupo con el estado de la partida más reciente:**
sql
Copiar código

```
SELECT

    p.username,

    p.Nombre,

    g.nombre_grupo,

    pa.Estado AS Estado_Partida

FROM
```

```
    Participante p
INNER JOIN
    Grupo g ON p.id = g.id
INNER JOIN
    Registro_Partida rp ON g.id = rp.Grupo_id
INNER JOIN
    Partida pa ON rp.Partida_id = pa.id
WHERE
    pa.Inicio_Partida = (
        SELECT MAX(Inicio_Partida)
        FROM Partida
    );
```

5.

## Subconsultas

**Grupos con más de 3 penalizaciones:**
sql
Copiar código

```
SELECT
    g.nombre_grupo
FROM
    Grupo g
WHERE
    (SELECT COUNT(*)
     FROM Penalizacion pen
     WHERE pen.id = g.id) > 3;
```

1.

**Detalles de partidas con duración mayor al promedio:**
sql
Copiar código

```
SELECT

    pa.descripcion,

    pa.Inicio_Partida,

    pa.tiempo_limite

FROM

    Partida pa

WHERE

    pa.tiempo_limite > (

        SELECT AVG(tiempo_limite)

        FROM Partida

    );
```

2.

**Participantes en una categoría específica:**
sql
Copiar código

```
SELECT

    p.username,

    p.Nombre,

    c.Nombre AS Categoria

FROM

    Participante p

INNER JOIN

    Categoria c ON p.categoria_id = c.ID

WHERE
```

```
    c.Nombre = 'Nombre_de_la_Categoria';
```

3.

**Partidas con más de 2 grupos registrados:**
sql
Copiar código

```
SELECT

    pa.descripcion,

    COUNT(rp.Grupo_id) AS Total_Grupos

FROM

    Partida pa

INNER JOIN

    Registro_Partida rp ON pa.id = rp.Partida_id

GROUP BY

    pa.descripcion

HAVING

    COUNT(rp.Grupo_id) > 2;
```

4.

**Penalizaciones más recientes por tipo:**
sql
Copiar código

```
SELECT

    tp.Descripcion,

    MAX(pen.Fecha_infraccion) AS Fecha_Reciente

FROM

    Penalizacion pen

INNER JOIN

    TipoPenalizacion tp ON pen.Tipo_id = tp.id
```

```
GROUP BY
    tp.Descripcion;
```

5.

Estas consultas y subconsultas permiten explorar relaciones, analizar datos y obtener información significativa de este modelo relacional. Si necesitas alguna consulta específica o personalización, ¡puedes indicármelo!

4o