DOCUMENTACIÓN DEL SISTEMA DE GESTIÓN DE LEAGUE OF LEGENDS

1. INTRODUCCIÓN: DESCRIPCIÓN DEL PROBLEMA

El proyecto consiste en un sistema de gestión de información para League of Legends, un popular videojuego de estrategia en línea. El sistema permite almacenar y manipular datos relacionados con los distintos elementos del juego:

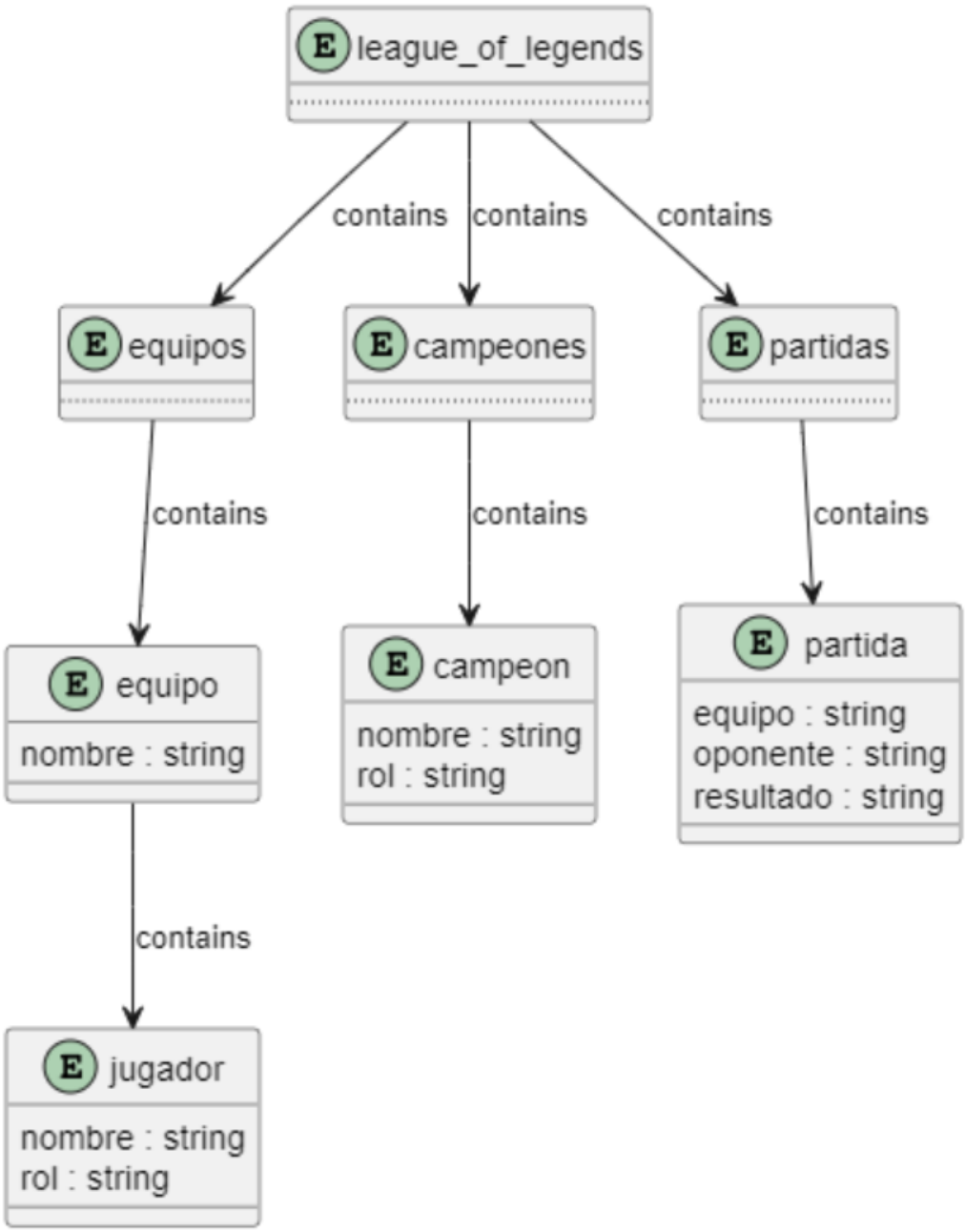
- **Campeones**: Los personajes jugables dentro del juego, cada uno con características únicas.
- **Jugadores**: Usuarios que participan en el juego, cada uno con sus estadísticas personales.
- **Equipos**: Agrupaciones de jugadores que compiten juntos en partidas.
- **Partidas**: Enfrentamientos entre equipos con resultados y estadísticas asociadas.

El problema principal que resuelve este sistema es la necesidad de gestionar grandes volúmenes de información relacionada con el ecosistema de League of Legends, permitiendo almacenar los datos en formato XML, consultarlos y actualizarlos cuando sea necesario. El sistema sigue un enfoque similar a NoSQL (aunque usando XML como formato de persistencia), con estructuras de documentos flexibles que permiten almacenar información jerárquica y relacionada.

2. MODELO ENTIDAD-RELACIÓN

El modelo entidad-relación (ER) del sistema se compone de las siguientes entidades y relaciones:

- **Campeón**: Representa a los personajes jugables en el juego. Cada campeón tiene un nombre único, rol, dificultad, poder, fecha de lanzamiento, habilidades y un estado de disponibilidad.
- **Jugador**: Representa a los usuarios que juegan el juego. Cada jugador tiene un nombre único, rol, nivel, experiencia, fecha de registro y un estado de actividad.
- **Equipo**: Representa a los grupos de jugadores que compiten juntos. Cada equipo tiene un nombre único, una lista de jugadores, fecha de creación, número de victorias y derrotas.
- **Partida**: Representa un enfrentamiento entre dos equipos. Cada partida tiene un equipo principal, un oponente, resultado, fecha, duración, estadísticas y puntuaciones de ambos equipos.
- **Relaciones**:
    - Un **jugador** puede pertenecer a un **equipo** (1:N).
    - Un **equipo** puede jugar múltiples **partidas** (1:N).
    - Un **campeón** puede ser jugado por múltiples **jugadores** en diferentes **partidas** (N:M).



2.2 DESCRIPCIÓN DE LOS CAMPOS

CAMPEÓN

- **nombre**: String - Identificador único del campeón (PK)
- **rol**: String - Función principal del campeón (top, jungla, mid, adc, support)
- **dificultad**: int - Nivel de dificultad del campeón (1-10)
- **poder**: double - Nivel de poder del campeón (0-100)
- **fechaLanzamiento**: Date - Fecha en que el campeón fue lanzado
- **habilidades**: List - Colección de habilidades del campeón
- **disponible**: boolean - Indica si el campeón está disponible para jugar

JUGADOR

- **nombre**: String - Identificador único del jugador (PK)
- **rol**: String - Posición que juega principalmente
- **nivel**: int - Nivel actual del jugador (1-100)
- **experiencia**: double - Puntos de experiencia acumulados
- **fechaRegistro**: Date - Fecha en que el jugador se registró
- **activo**: boolean - Indica si el jugador está activo actualmente

EQUIPO

- **nombre**: String - Identificador único del equipo (PK)
- **jugadores**: List - Colección de jugadores que pertenecen al equipo
- **fechaCreacion**: Date - Fecha en que se creó el equipo
- **victorias**: int - Número de partidas ganadas
- **derrotas**: int - Número de partidas perdidas

PARTIDA

- **equipo**: String - Nombre del equipo principal
- **oponente**: String - Nombre del equipo contrario
- **resultado**: String - Resultado de la partida (Ganado/Perdido/Empate)
- **fecha**: Date - Fecha en que se jugó la partida
- **duracionMinutos**: int - Duración de la partida en minutos
- **estadisticas**: Map<String, Integer> - Mapa de estadísticas de la partida
- **puntuacionEquipo**: int - Puntuación obtenida por el equipo principal
- **puntuacionOponente**: int - Puntuación obtenida por el equipo contrario

3. DISEÑO DE LA TRANSFORMACIÓN (ESQUEMA DE REPARTO)

La transformación de este modelo orientado a objetos a XML sigue un esquema jerárquico que refleja las relaciones entre entidades:

```
<liga>
  <!-- Lista de campeones -->
  <campeones>
    <campeon>
      <nombre>...</nombre>
      <rol>...</rol>
      <dificultad>...</dificultad>
      <poder>...</poder>
      <fechaLanzamiento>...</fechaLanzamiento>
      <habilidades>
        <habilidad>...</habilidad>
        <habilidad>...</habilidad>
        <!-- ... -->
      </habilidades>
      <disponible>...</disponible>
    </campeon>
    <!-- Más campeones -->
  </campeones>
```

```
  <!-- Lista de equipos -->
  <equipos>
    <equipo>
      <nombre>...</nombre>
      <fechaCreacion>...</fechaCreacion>
      <victorias>...</victorias>
      <derrotas>...</derrotas>
      <jugadores>
        <jugador>
          <nombre>...</nombre>
          <rol>...</rol>
          <nivel>...</nivel>
          <experiencia>...</experiencia>
          <fechaRegistro>...</fechaRegistro>
          <activo>...</activo>
        </jugador>
        <!-- Más jugadores -->
      </jugadores>
    </equipo>
    <!-- Más equipos -->
  </equipos>

  <!-- Lista de partidas -->
  <partidas>
    <partida>
      <equipo>...</equipo>
      <oponente>...</oponente>
      <resultado>...</resultado>
      <fecha>...</fecha>
      <duracionMinutos>...</duracionMinutos>
      <estadisticas>
        <estadistica>
          <clave>kills</clave>
          <valor>...</valor>
        </estadistica>
        <estadistica>
          <clave>deaths</clave>
          <valor>...</valor>
        </estadistica>
        <!-- Más estadísticas -->
      </estadisticas>
      <puntuacionEquipo>...</puntuacionEquipo>
      <puntuacionOponente>...</puntuacionOponente>
    </partida>
    <!-- Más partidas -->
  </partidas>
</liga>
```

Este esquema de reparto permite:

1. **Integridad referencial**: Las referencias entre entidades se mantienen mediante nombres únicos.

2. **Flexibilidad**: La estructura jerárquica permite reflejar relaciones complejas.
3. **Extensibilidad**: Facilita la adición de nuevos elementos sin modificar el esquema base.
4. **Consulta eficiente**: Mediante XPath o XQuery se pueden realizar consultas complejas sobre el documento XML.

El sistema emplea este esquema para persistir todos los datos del juego en un único documento XML estructurado, que puede ser leído y manipulado por la aplicación según las necesidades del usuario.

4. JUSTIFICACIÓN DEL USO DE XML FRENTE A SQL

La elección de XML como formato de almacenamiento para este sistema, en lugar de una base de datos SQL tradicional, está motivada por las siguientes razones:

1. **Flexibilidad en la estructura de datos**: XML permite representar datos jerárquicos y anidados de forma natural, como las habilidades de un campeón o los jugadores de un equipo, sin necesidad de normalizar las tablas o crear relaciones complejas como sería necesario en SQL.

2. **Evolución del esquema**: En un juego como League of Legends, donde constantemente se añaden nuevos campeones, ítems y mecánicas, XML permite modificar la estructura de los documentos sin requerir alteraciones en el esquema de la base de datos, algo que en SQL implicaría migraciones y cambios en las tablas.

3. **Integración y portabilidad**: XML es un formato estándar y universalmente soportado, lo que facilita la integración con otras aplicaciones y servicios. Los archivos XML pueden ser leídos y procesados en cualquier plataforma o lenguaje de programación sin necesidad de un DBMS.

4. **Representación de datos semi-estructurados**: Las estadísticas y atributos de partidas y campeones pueden variar significativamente, y XML permite representar estos datos semi-estructurados de manera más natural que las tablas rígidas de SQL.

5. **Simplicidad en el despliegue**: Al utilizar archivos XML, el sistema no requiere la instalación y configuración de un sistema gestor de bases de datos, simplificando el despliegue y la distribución de la aplicación, especialmente en entornos con recursos limitados.

6. **Legibilidad y accesibilidad**: Los archivos XML son legibles por humanos, lo que facilita la depuración, el desarrollo y la comprensión de la estructura de datos, a diferencia de los formatos binarios utilizados por algunos DBMS.

7. **Consultas con XPath/XQuery**: XML soporta consultas complejas utilizando lenguajes especializados como XPath y XQuery, permitiendo extraer información específica sin necesidad de utilizar SQL, con la ventaja adicional de poder navegar por la estructura jerárquica de forma natural.

Esta decisión de diseño permite aprovechar las ventajas de los modelos NoSQL orientados a documentos (representados aquí mediante XML), manteniendo la flexibilidad y expresividad necesarias para modelar el complejo ecosistema de datos de League of Legends.