Descripción del problema DDSI

Antonio Checa Molina
Iñaki Madinabeitia
Bruno Santidrián
Darío Sierra

15 de octubre de 2017

Índice

1. Gestión de un registro de partidas en juegos generales

El problema a resolver es mantener un registro rápido y funcional de partidas para cualquier tipo de juegos, deportes o competiciones con el fin de facilitar análisis y entrenamiento de jugadores. Hay que desarrollar un sistema de información de organización de las partidas y de los juegos.

Al principio, un grupo de gestores necesita proporcionar información de un juego y de sus partidas: aquello que se guarda, como la puntuación, los equipos, los jugadores de los equipos o un vídeo del partido. Una vez guardados estos atributos, el usuario podrá acceder al registro de partidas de un juego concreto e incluir aquellas de las que tenga datos.

Por ejemplo, si se añade el juego baloncesto junto a un conjunto de atributos como {Puntuación, Equipos, Puntuación de cada parcial, vídeos}, cada partida contendrá esta información y los usuarios podrán acceder a esta. En esta práctica nos restringiremos a un pequeño número de juegos, realizando el diseño de base de datos para cada uno.

1.1. Consulta

La función principal es que el usuario pueda buscar con facilidad partidas en función de los atributos. Necesita permitir al usuario fijar uno de los atributos, buscar en un rango de parámetros o hacer búsquedas flexibles en cualquiera de los juegos soportados.

Imaginemos que estamos con Baloncesto con muchos partidos de varios equipos ya añadidos. La aplicación debe permitir al usuario buscar aspectos simples como 'Todos los partidos del Unicaja' o 'Partidos en los que ganó el Real Madrid' o 'Qué equipo jugó en tal partido como titular'. Todo esto son consultas a una base de datos, pero la aplicación debe permitir al usuario realizarlas sin tener conocimiento en absoluto sobre bases de datos, apoyadas con una intuitiva interfaz gráfica.

También debe permitir búsqueda donde el usuario otorgue uno o varios parámetros. Con la idea de usar la consulta en esta aplicación para estudiar al rival, también debe permitir buscar partidos donde, según el ejemplo anterior, la diferencia en el marcador fue menos de 10, o donde tal jugador jugó tantos minutos, o donde tal jugador metió tantos puntos. En estos ejemplos, el parámetro sería los números.

Como la aplicación debe permitir hacer consultas sin que el usuario conozca sobre cómo se consulta a una base de datos, debemos limitar qué aspectos se pueden consultar. En resumen, la aplicación permitirá mostrar cualquier atributo de cualquier partida, mostrar las partidas seleccionadas tras hacer una restricción del valor de los atributos (con paso de parámetros en algunos casos concretos); y todo esto siempre en el ámbito del mismo juego.

En el ejemplo de antes, un usuario podría buscar los partidos del Golden State Warriors en los que haya participado Curry y que haya metido más de 30 puntos, o en los que haya participado el Unicaja y haya perdido.

1.2. Consejos

En todos los juegos en los que se guarde el estilo de juego, la aplicación aconseja cómo jugar contra un oponente en concreto, recomendando al usuario aquellos estilos que más probabilidades tengan de ganar contra ese adversario.

Tanto los distintos estilos como los consejos asociados a estos dependen del juego a tratar. Por ejemplo, en juegos de estrategia como el ajedrez, se podrían analizar las estrategias más utilizadas por un jugador y recomendar al usuario que estudie determinadas jugadas que le puedan dar ventaja, en el caso de juegos de mazos de cartas (e.g. Magic, Hearthstone) también sería posible recomendar algunas configuraciones de mazos, incluso en algunos deportes se podrían recomendar alineaciones y jugadores.

Debido a la diferencias en los aspectos recomendables entre distintos juegos será necesario que cada vez que se dé soporte a un nuevo juego se especifiquen las posibles recomendaciones (requisitos funcionales), los datos necesarios para llevarlas a cabo (requisitos de datos) y la forma de analizar

estos datos para proporcionar los consejos (implementación).

También es posible que el subsistema de consejos use funcionalidades del subsistema de estadísticas, siempre que dicha funcionalidad esté implementada, para que las recomendaciones sean más certeras.

1.3. Inclusión de partidas

La inclusión de las partidas necesita que el usuario incluya aquellos atributos vitales en una partida, pudiendo o no incluir los opcionales. Necesitaría insertarlos por la interfaz gráfica.

El sistema de información que planteamos crear deberá incrementar el número de juegos disponibles con el avance del tiempo, para asegurar su uso continuado. Por tanto, distinguimos entre inclusión de juegos (trabajo de los administradores de la base de datos) y la inclusión de partidas (trabajo de los usuarios).

Una partida será un registro más de la tabla de datos asociada a cada juego. Como atributo vital se necesitará añadir el nombre del juego, así como la puntuación final de dicha partida; estos serán atributos comunes a cada a todas las partidas, pues todas pertenecerán a un juego y todas tendrán una puntuación asociada.

Los atributos opcionales serán aquellos inherentes a cada juego; no tiene sentido la inclusión de un atributo 'Jugadores del equipo' en un juego de cartas, ni un 'baraja del oponente' en uno que no lo sea. Aquí opcional no se entiende como 'poco relevante' pues estos atributos son la razón de ser del propio sistema de información. Los datos inherentes a cada juego, los que lo hacen distinguible del resto, serán objeto del análisis en el sistema.

Las partidas se incluirán desde la interfaz gráfica, donde una vez seleccionado el tipo de juego, se abrirán una serie de atributos (a definir por los administradores del sistema) que permitirán al jugador guardar su partida. Entre estos podremos encontrar: 'Jugadores del equipo', 'Mi Baraja', 'Baraja del oponente', 'Número de faltas cometido', 'Pichichi del encuentro', etc.

Aunque hemos clarificado que la inclusión de partidas debe ser trabajo de los usuarios, en juegos específicos (fútbol, baloncesto, etc), en los que, a diferencia de una partida entre dos jugadores de ajedrez, no queda claro quién debe almacenar el resultado. Es por esto, que se estudiará la posibilidad de responsabilizar a los administradores de mantener la base de datos actualizada para los juegos en los que se de esta situación.

1.4. Estadísticas

Por último, el sistema debería agregar las estadísticas que pida el usuario. Hay diversos modos de generar una gráfica. El usuario tendrá que indicar al sistema qué modo de salida de los datos quiere (por ejemplo, una gráfica 2D con líneas). Una vez indicado, tendrá que añadir sobre qué atributos del juego quiere la gráfica de datos (por ejemplo, sobre la puntuación de los partidos de los Lakers en cada mes). Después, el sistema debe generar la representación elegida de los datos en caso de que pueda y presentarla ante el usuario. En caso de que no pueda deberá mostrar un mensaje de error mencionando la causa.

Los posibles modos de una gráfica son: gráfico 2D (al menos el eje Y debe ser numérico), gráfico 3D (al menos el eje Y debe ser numérico), columna agrupada (al menos dos elementos numéricos por cada uno no numérico), circular (atributos numéricos positivos).

Al tratar los atributos el usuario debe ser capaz de realizar funciones como la media, la suma o contar datos que cumplan una restricción concreta. Para esto, el sistema debe ser capaz de proporcionar ayuda con todas las funciones que se pueden insertar, y gestionarlas para que funcionen como es debido.

Las estadísticas, datos y gráficas generadas se tendrán que guardar en el sistema, para que el usuario pueda acceder a ellas sin tener que generarlas de nuevo. Habrá acceso a un registro con orden inverso a la fecha de creación para facilitar la búsqueda en el mismo.

Cada modo de salida tiene unas restricciones sobre los atributos, en el caso en el que el usuario incluya unos datos erróneos para una gráfica concreta (por ejemplo, seleccionar el modo de disco para representar datos que no son números). En este caso, el error tendrá que hacer referencia al error concreto, y mostrarlo por pantalla para que el usuario entienda qué ha pasado.

2. Análisis de requisitos

Ejemplo

- **RFX.Y Nombre**: Descripción

  Requisitos de entrada: Descripción opcional de entrada

  - RDX.Y
  - RDX.Y

  Manejamiento de datos: Descripción opcional de entrada

  - RDX.Y

  Salida: (Puede no haber nada) Descripción opcional de salida

  - RDX.Y

2.1. Primer subsistema, consulta

2.1.1. Requisito de datos

2.1.2. Requisitos funcionales

2.1.3. Restricciones semánticas

2.2. Segundo subsistema, consejos

2.2.1. Requisitos de datos

- **RD1: Jugador de pokémon** contra el que diseñar un equipo. Se requiere la identificación del jugador.
- **RD2: Debilidades y fortalezas de tipos pokémon,** para cada tipo se require del multiplicador de ataque contra los demás tipos.
- **RD3: Información de ataques pokémon,** para cada ataque se requiere:
  - Tipo del ataque
  - Potencia del ataque
  - Precisión del ataque
  - Si es físico, especial o de estado
  - Puntos de poder
- **RD4: Estadísticas base de los pokemon,** para cada pokémon se requiere:
  - Puntos de salud
  - Puntos de ataque
  - Puntos de defensa
  - Puntos de ataque especial
  - Puntos de defensa especial
  - Puntos de velocidad
- **RD5: Equipos pokémon usados por un jugador** en las partidas registradas. Se compone de:
  - Una lista de como máximo seis pokémon por partida.

- Los atributos relacionados con cada pokemon (opcionales), estos son:
  - Id del pokémon
  - Nombre de los ataques
  - Objeto asignado
  - Naturaleza
  - Distibución de IVs
  - Distribución de EVs

- **RD6: Equipo pokémon** diseñado para vencer a un oponente. Consta de:
  - Una lista de como máximo seis pokémon.
  - Los atributos relacionados con cada pokemon (opcionales), estos son:
    - Id del pokémon
    - Nombre de los ataques
    - Objeto asignado
    - Naturaleza
    - Distibución de IVs
    - Distribución de EVs
- **RD7: Jugador de Hearthstone** contra el que diseñar el mazo. Se requiere su identificación.
- **RD8: Atributos de las cartas de Hearthsone.** Para cada carta se requiere:
  -
- **RD9: Mazos de Hearthstone usados por un jugador** en las partidas registradas. Se compone de:
  - Un heroe
  - Una lista de identificadores de treinta cartas
- **RD10: Estilo de juego de un jugador de Hearthsone,** puede ser ofensivo, defensivo, mixto ...
- **RD11: Mazo de Hearthstone** diseñado para vencer a un oponente. Consta de:
  - Un heroe
  - Una lista de indentificadores de treinta cartas

2.2.2. Requisitos funcionales

- **RF1: Recomendar equipos pokémon:**
  Requisitos de entrada:
  - RD1
  - RD2
  - RD3
  - RD4

  Manejo de datos:
  - RD5

  Salida:
  - RD6

- **RF2: Recomendar mazos de Hearthstone:**
  Requisitos de entrada:
  - RD7
  - RD8

  Manejo de datos:
  - RD9
  - RD10

  Salida:
  - RD11

2.2.3. Restricciones semánticas

2.3. Tercer subsistema, inclusión

2.3.1. Requisito de datos

- **RD3.1: Atributos Principales**, proporcionados por el usuario, se componen de:
  - Puntuación de la partida.
  - Nombre del juego
- **RD3.2: Atributos Secundarios**, se componen de:
  - Nombre
  - Valor del atributo

  La lista de atributos secundarios la definirań los administradores del sistema.

2.3.2. Requisitos funcionales

- **RF3.1 Selección del juego**: El usuario selecciona uno de entre los juegos que recoge el sistema.
  - RD3.1

  Salida: En pantalla se muestran los juegos posibles.
- **RF3.2 Inclusión de una partida**: El usuario inserta el atributo principal y crea un nuevo registro de partida.
  - RD3.1

  Requisitos de entrada: Una vez escogido el juego se inserta la puntuación de la partida.

  Manejo de datos: Se leen el atributo proporcionado y se el registro en la base de datos

  Salida: Mensaje de conformidad.
- **RF3.3 Modificación de un registro**: Una vez creado, el usuario podrá añadir atributos opcionales al registro.

  Requisitos de entrada: Lista de los atributos a insertar
  - RF3.2

  Manejo de datos: Se leen los atributos seleccionados y se inserta la gráfica en la base de datos.
  - RD3.2

  Salida: A pantalla se muestra una interfaz en la que se representa el registro completo.

2.3.3. Restricciones semánticas

2.4. Cuarto subsistema, estadísticas

2.4.1. Requisito de datos

- **RD4.1: Atributos de entrada a una gráfica 2D**, proporcionados por el usuario, se componen de:
  - Atributo de la partida sobre la X
  - Atributo de la partida sobre la Y
- **RD4.2: Atributos de una partida**, se componen de:
  - Nombre
  - Valor del atributo

2.4.2. Requisitos funcionales

- **RF4.1 Realizar gráfica 2D de unos atributos**: El usuario selecciona dos atributos, de los cuales se realiza una gráfica 2D

  Requisitos de entrada: Escoger gráfica 2D y elegir qué dos atributos se escogen
  - RD4.1

  Manejo de datos: Se leen los dos atributos seleccionados y se inserta la gráfica en la base de datos
  - RD4.2

  Salida: Gráfica 2D
  - RD4.3
- **RF4.2 Realizar gráfica 3D de unos atributos**: El usuario selecciona tres atributos, de los cuales se realiza una gráfica 3D

  Requisitos de entrada: Escoger gráfica 3D y elegir qué tres atributos se escogen
  - RD4.4

  Manejo de datos: Se leen los tres atributos seleccionados y se inserta la gráfica en la base de datos
  - RD4.5

  Salida: Gráfica 3D
  - RD4.6
- **RF4.3 Realizar gráfica de columnas agrupadas de unos atributos**: El usuario selecciona dos atributos para las columnas y un tercero para el eje de abscisas, de los cuales se realiza la gráfica

  Requisitos de entrada: Escoger gráfica columnas agrupadas y elegir qué atributos se escogen
  - RD4.7

  Manejo de datos: Se leen los atributos seleccionados y se inserta la gráfica en la base de datos.
  - RD4.8

  Salida: Gráfica

- RD4.9

- **RF4.4 Realizar gráfica circular de unos atributos**: El usuario selecciona un atributo, del cual se hace una gráfica porcentual en forma de círculo

  Requisitos de entrada: Escoger gráfica circular y elegir qué atributos se escoge.

  - RD4.10

  Manejo de datos: Se lee el atributo seleccionado y se inserta la gráfica en la base de datos.

  - RD4.11

  Salida: Gráfica.

  - RD4.12

- **RF4.5 Realizar una función elemental sobre un atributo concreto**: El usuario selecciona una función elemental (detalladas en la entrada de este requisito) y un atributo sobre el que operar, y se devuelve el resultado de la operación.

  Requisitos de entrada: El atributo y la función, a elegir entre media, varianza, suma, contar, máximo o mínimo.

  - RD4.13

  Manejo de datos: Se lee el atributo seleccionado y se opera.

  - RD4.14

  Salida: Resultado de la función.

  - RD4.15

- **RF4.6 Conseguir las últimas gráficas pedidas**: El usuario le da a un botón para ver las últimas gráficas que ha pedido.

  Requisitos de entrada: La activación de un botón.

  - RD4.16

  Manejo de datos: Se cogen las gráficas realizadas en función de su fecha.

  - RD4.17

  Salida: Gráficas, ordenadas de más a menos recientes.

  - RD4.18

2.4.3. Restricciones semánticas