IMPLEMENTACIÓN
SISTEMA DE INFORMACIÓN:

LA LIGA

GRUPO D2 JJ:
Borrajo Yusty, Valeria
Díaz-Meco Terres, Manuel
Gómez López, Javier
Mata Carrillo, Carlos
Porcel Esquivel, Buenaventura
Reyes De Toro, Jesús

UNIVERSIDAD
DE GRANADA

ÍNDICE

1. SUBSISTEMA DE CLUBES (Jesús Reyes De Toro)

1.1. Descripción

Subsistema de Clubes: Este subsistema gestionaría toda la información relacionada con los clubes que participan en la liga de fútbol. En cuanto al subsistema de clubes se pueden llevar a cabo las siguientes acciones: **La inserción de un club**, que se hará insertando su nombre, sede, año de fundación, jugadores asociados al club y el resto de información que se detalla con posterioridad. Para **dar de baja un club**, el usuario lo eliminará conociendo el id del club que será el nombre. Podemos también **modificar el club**, para ello se deberá conocer el nombre del club del que se quiere modificar algo. Para **asociar un club a un jugador**, deberemos de conocer el id del jugador y el nombre del club. Por último, tenemos la función de **listar los clubes.**

Contendría detalles de cada club, como nombre (20 caracteres), ciudad(10 caracteres), año de fundación (entero positivo), presidente (10 caracteres), patrocinadores (10 caracteres), y una lista de jugadores asociados a cada club (listado de cadena de caracteres). No puede haber dos clubes con el mismo nombre. Este subsistema estaría relacionado con el subsistema de Jugadores, ya que los jugadores están afiliados a clubes específicos.

Sobre los requisitos funcionales de nuestro subsistema podemos decir que podemos realizar las siguientes operaciones

1.2. Requisitos funcionales

- ***RF1.1*: *Dar de alta un club***
  - Entrada**:** Agente externo: administrador. Acción: Solicitar inserción. Requisitos de datos de entrada: *RDE1.1*.
  - Base de datos**:** Requisito de datos de escritura: *RDW1.1.*
  - Salida**:** Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
  - *RDE1.1*: Datos de entrada de alta del club:
    - Nombre del club: Cadena de caracteres (máximo 50 caracteres).
    - Ciudad o ubicación del club: Cadena de caracteres (máximo 50 caracteres).
    - Año de fundación: Entero de cuatro dígitos.
    - Presidente: Cadena de caracteres (máximo 50 caracteres).
    - DNI_Jugador: Lista de jugadores cada uno cadena de 9 caracteres (DNI del jugador) –ELIMINADO
    - Patrocinadores: Cadena de caracteres (10)
    - DNI_Entrenador: Cadena de caracteres (10) – ELIMINADO
  - *RDW1.1:* Datos almacenados del club
    - Lo mismo que *RDE1.1*

- ***RF1.2: Dar de baja un club***
    - Entrada: Agente externo: administrador. Acción: Solicitar borrado. Requisitos de datos de entrada: *RDE1.2*.
    - Base de datos: Requisito de datos de escritura: *RDW1.2*.
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE1.2*: Datos de entrada de baja de club:
        - Nombre: Cadena de caracteres (20) (nombre del club).
    - RDW1.2: Datos almacenados del club.
        - Los mismos que *RDW1.1*

- ***RF1.3*: *Mostrar listado de clubes***
    - Entrada: Agente externo: administrador. Acción: solicitar listado de clubes. Requisitos de datos de entrada: ninguno.
    - Base de datos: Requisito de datos de lectura: *RDR1.3.*
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: *RDS1.1*.
    - *RDR1.3*: Datos de clubes almacenados:
        - Los mismos datos que *RDW1.1*. Ademas se mostraran los entrenadores y jugadores asociados
    - *RDS1.3*: Listado de registros, cada uno de ellos con los mismos datos que *RDR1.1*.

- ***RF1.4*: *Modificar un club***
    - Entrada: Agente externo: administrador. Acción: Modificar información de un club. Requisitos de datos de entrada*: RDE1.4*.
    - Base de datos: Requisito de datos de escritura: *RDW1.4.*
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE1.4:* Datos de entrada de modificación del club:
        - Nombre del club: Cadena de caracteres (20) (nombre del club).
        - De los datos de *RDE1.1*, los que se desee modificar: Patrocinadores, Ciudad, Año de fundación, Presidente, Jugadores asociados – ELIMINADO
    - *RDW1.4:* Datos almacenados del club
        - Los utilizados en *RDE1.3,* excepto el Id del club que es inmutable

- ***RF1.5*: *Asociar un club a un jugador***

- Entrada: Agente externo: administrador. Acción: Asociar un jugador a un club. Requisitos de datos de entrada: *RDE1.5*.
- Base de datos: Requisito de datos de escritura: *RDW1.5*.
- Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
- *RDE1.5*: Datos de entrada de asociación de un jugador a un club:
  - DNI: Cadena de caracteres (9) (DNI del jugador).
  - Nombre: cadena de caracteres (20) (ID del club).
- *RDW1.5*: Listados de registros cada uno de ellos para el jugador especificado

- ***RF1.6*: *Asociar un club a un entrenador***
  - Entrada: Agente externo: administrador. Acción: Asociar un club a un entrenador. Requisitos de datos de entrada: *RDE1.6*.
  - Base de datos: Requisito de datos de escritura: *RDW1.6*.
  - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
  - *RDE1.6*: Datos de entrada de asociación de un jugador a un club:
    - DNI: Cadena de caracteres (9) (DNI del entrenador).
    - Nombre: cadena de caracteres (20) (ID del club).
  - *RDW1.6*: Listados de registros cada uno de ellos para el entrenador especificado

1.3. Restricciones semánticas

- ***RS1.1*:** Para dar de alta un club, el nombre debe ser único en la liga.
  - RF: *RF1.1* (Dar de alta un club).
  - RD(s): *RDE1.1* (Datos de entrada de alta del club).
  - Descripción: El sistema debe asegurarse de que no exista otro club en la liga con el mismo nombre antes de permitir el alta del club. Si el nombre ya está en uso, se debe devolver un error y no se debe crear el club.

- ***RS1.2:*** No se puede dar de baja un club si tiene jugadores asociados.
  - RF: *RF1.2* (Dar de baja un club).
  - RD(s): RDE1.2 (Datos de entrada de baja de club) y *RDS1.4* (Listados de registros de asociación de jugadores a clubes).
  - Descripción: El sistema debe verificar si el club que se intenta dar de baja tiene jugadores asociados. Si hay jugadores asociados al club, no se permitirá dar de baja el club y se devolverá un error.

- ***RS1.3*:** Al modificar un club, el nombre no puede ser cambiado a un nombre ya existente en la liga.
    - RF: *RF1.3* (Modificar un club).
    - RD(s): *RDE1.3* (Datos de entrada de modificación del club) y *RDS1.1* (Listado de registros de clubes almacenados).
    - Descripción: Cuando se modifica un club, el sistema debe asegurarse de que el nuevo nombre del club no coincida con el nombre de otro club en la liga. Si el nuevo nombre ya existe, se debe devolver un error y no se permitirá la modificación del nombre del club.

- ***RS1.4*:** No se pueden asociar jugadores a un club que no existe en la liga.
    - RF: *RF1.4* (Asociar un club a un jugador).
    - RD(s): RDE1.4 (Datos de entrada de asociación de un jugador a un club) y RDS1.1 (Listado de registros de clubes almacenados).
    - Descripción: Antes de asociar un jugador a un club, el sistema debe verificar si el club al que se intenta asociar realmente existe en la liga. Si el club no existe, se debe devolver un error y no se permitirá la asociación.

- ***RS1.5*:** Si no se ha jugado aún un partido, borrar el club y en cascada los partidos.
    - RF: *RF1.2* (Dar de baja a un club).
    - RD(s): *RDS1.1* (Listado de partidos jugados por clubes).
    - Descripción: El sistema debe verificar si existen partidos jugados por un club ya. En caso de que sí, deberemos borrar el club y en cascada los partidos. En caso de que no se haya jugado el partido, no podemos borrar el club.

1.4. DFD Clubes

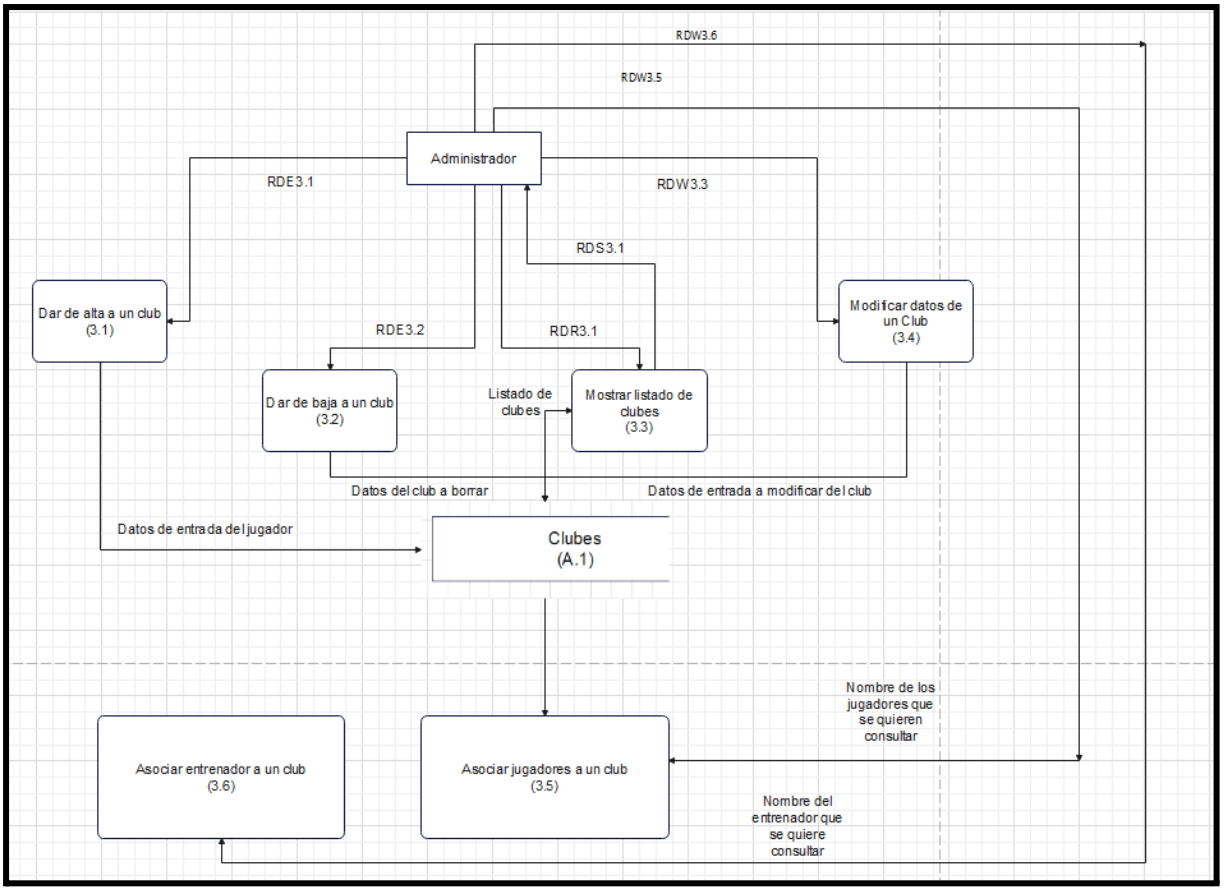


Figura 1.1: DFD Clubes

1.5. Esquema Externo DFD1 Clubes

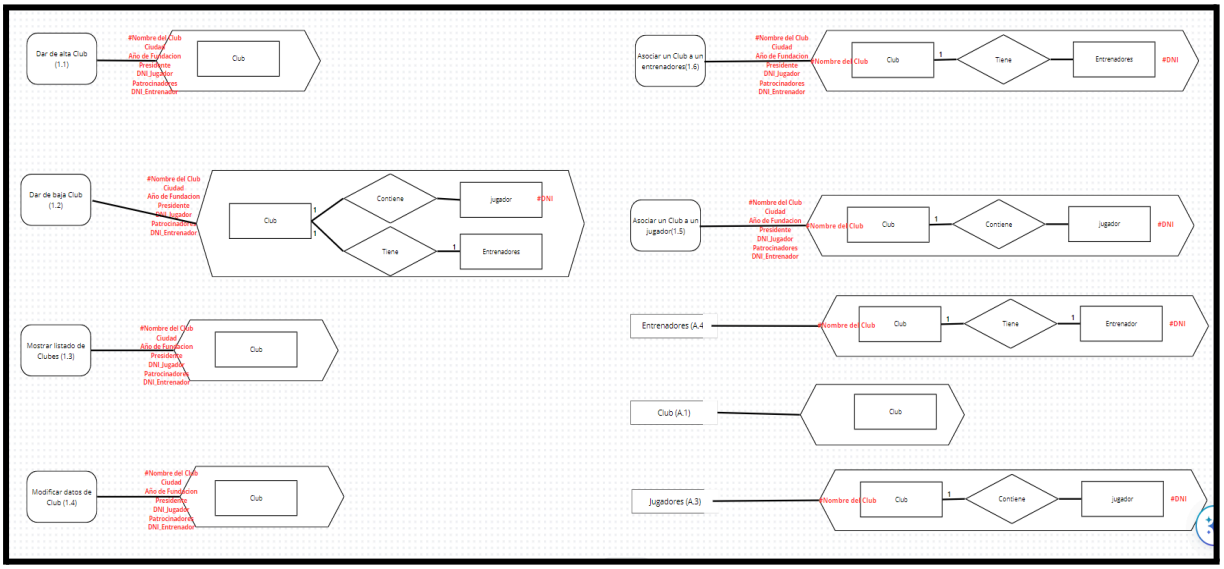


Figura 1.2: Esquemas externos DFD Clubes

1.6. Esquema conceptual Clubes

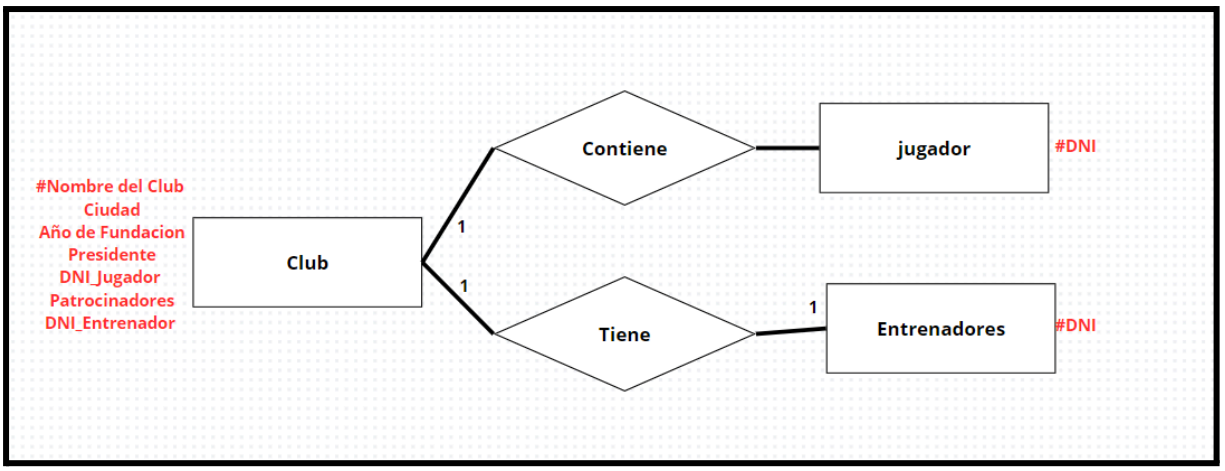


Figura 1.3: Esquema conceptual Subsistema Clubes

2. SUBSISTEMA DE PARTIDOS (Carlos Mata Carrillo)

2.1. Descripción

Este subsistema gestionaría la programación, seguimiento y registro de los partidos de la liga. Contendría detalles de la programación de partidos: fecha (fecha) y lugar (20 caracteres). Además también se almacenarán elementos estadísticos relevantes: goles (2 caracteres numéricos), tarjetas (2 caracteres numéricos), asistencias (2 caracteres numéricos), posesión de balón (3 caracteres numéricos), remates (3 caracteres numéricos), remates a portería (3 caracteres numéricos), corners (3 caracteres numéricos), fueras de juego (3 caracteres numéricos) y pases (4 caracteres numéricos). Los partidos estarían vinculados a clubes, árbitros, entrenadores y jugadores específicos, de forma que se tiene en cuenta la convocatoria de cada partido. Las estadísticas de partidos se basarán en los eventos registrados durante los partidos.
Para **insertar un partido** en el sistema, será necesario proveer el nombre de los dos equipos, fecha, lugar y árbitro, es decir, los datos que se saben antes de que se juegue. El resto de datos son opcionales para registrar un partido. El sistema permitirá **mostrar un listado de partidos** pero únicamente con los datos esenciales de cada uno (nombres de los equipos, ID de Partido, goles, lugar y fecha). El sistema también permitirá **mostrar un partido deseado** junto con sus datos estadísticos. Por último el sistema nos permitirá **modificar los datos de un partido** salvo su identificador principal.

2.2. Requisitos funcionales

- ***RF2.1**: **Insertar un partido**:*
  - Entrada: Agente externo: usuario (Administrador). Acción: Solicitar inserción. Requisitos de datos de entrada: *RDE2.1*.
  - Base de datos: Requisito de datos de escritura: *RDW2.1*.
  - Salida: Agente externo: usuario. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.

    - *RDE2.1*: Datos de entrada del partido:
        - ID Partido: Tipo entero que identifica de manera única a cada partido
        - Fecha: Tipo Date.
        - Lugar: cadena de caracteres (20 caracteres)
        - Goles: 2 caracteres numéricos cada una
        - Tarjetas: 2 caracteres numéricos cada una
        - Asistencias: 2 caracteres numéricos cada una
        - Posesión de balón: 3 caracteres numéricos cada una
        - Remates: 3 caracteres numéricos cada una
        - Remates a portería: 3 caracteres numéricos cada una
        - Corners: 3 caracteres numéricos cada una
        - Fueras de juego: 3 caracteres numéricos cada una
        - Pases: 4 caracteres numéricos cada una

    - *RDW2.1:* Datos almacenados del partido
        - Los mismos que *RDE2.1*

- ***RF2.2**: **Eliminar un partido***
    - Entrada: Agente externo: usuario (Administrador). Acción: Solicitar borrado. Requisitos de datos de entrada: *RDE2.2*.
    - Base de datos: Requisito de datos de escritura: *RDW2.2*.
    - Salida: Agente externo: usuario. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE2.2*: Datos de entrada de borrado de partido:
        - ID Partido: Tipo entero (ID Partido).
    - *RDW2.2:* Datos almacenados de partido
        - Los mismos datos que *RDE2.1* más todas las relaciones que se tengan almacenadas. Estas solo podrán ser con los clubes y/o jugadores (se puede saber el equipo pero no la convocatoria), pero no podrá ser la clasificación ya que un partido se asocia con este subsistema una vez jugado y como se explica en los requisitos semánticos, una partido jugado no puede ser eliminado del sistema.
- ***RF2.3**: **Mostrar listado de partidos solo con datos esenciales**.*
    - Entrada: Agente externo: Personal de la liga (Todas las acciones llevadas a cabo por el personal de la liga también las puede realizar el Administrador). Acción: Solicitar borrado. Requisitos de datos de lectura: ninguno.
    - Base de datos: Requisito de datos de lectura: *RDR2.3*
    - Salida: Agente externo: usuario. Acción: confirmación de resultado. Requisito de datos de salida: *RDS2.3*.
    - *RDR2.3*: Datos de partido almacenado:
        - ID Partido: Tipo entero que identifica de manera única a cada partido
        - Equipos: Array de enteros (IDs de los Equipos).
        - Fecha: Tipo Date.

        - Lugar: cadena de caracteres (20 caracteres)
        - Goles: Tipo pareja (2 caracteres numéricos cada una)
    - *RDS2.3:* Datos almacenados de partido
        - Los mismos datos que *RDR2.3*
- ***RF2.4**: **Modificar datos de un partido***
    - Entrada: Agente externo: usuario (Administrador). Acción: Solicitar borrado. Requisitos de datos de entrada: *RDE2.4.*
    - Base de datos: Requisitos de datos de escritura: *RDW2.4.*
    - Salida: Agente externo: usuario. Acción: confirmación de resultado. Requisito de datos de salida: ninguno
    - *RDE2.4:* Datos de entrada a modificar del partido.
        - ID partido
        - Datos que queramos modificar de *RDE2.1.*
    - *RDW2.4:* Datos almacenados de partido.
        - Los utilizados en *RDE2.1*, excepto el ID Partido que es inmutable
- ***RF2.5**: **Mostrar todos los datos de un partido***
    - Entrada: Agente externo: Personal de la Liga (y Administrador). Acción: Solicitar borrado. Requisitos de datos de lectura: ninguno.
    - Base de datos: Requisito de datos de lectura: *RDR2.5*
    - Salida: Agente externo: usuario. Acción: confirmación de resultado. Requisito de datos de salida: *RDS2.5*.
    - *RDR2.5*: Datos de partido almacenado:
        - Todos los datos de *RDE2.1.*
    - *RDS2.5:* Datos almacenados de partido
        - Los mismos datos que *RDR2.1.*

2.3. Restricciones semánticas

- ***RS2.1**:* La suma de la pareja de “Posesión de balón” tiene que ser estrictamente 100.
    - RF: *RF2.1.*
    - RD: *RDW2.1*
    - Descripción: Si la suma de las dos posesiones no es 100, se muestra un mensaje de advertencia y no permite guardar el dato.
- ***RS2.2**:* Un equipo no puede jugar más de un partido en un mismo día.
    - RF: *RF2.1.*
    - RD: *RDW2.1*
    - Descripción: Si al introducir un nuevo partido, el sistema comprueba que alguno de los equipos ya juega un partido en la misma fecha se devolverá un error y no se registrará el partido
- ***RS2.3:*** No se permitirá borrar un partido del sistema una vez jugado.
    - RF: *RF2.2.*
    - RD: *RDW2.2.*

- Descripción: Si se solicita borrar un partido cuya fecha es anterior a la solicitud de eliminación, se devolverá un error y no se ejecutará el borrado.

- ***RS2.4:*** No se permitirá modificar el identificador único de un partido una vez registrado.
  - RF: *RF2.4.*
  - RD: *RDW2.4*
  - Descripción: Si se intenta modificar el ID de Partido una vez que ya está registrado el partido, se devolverá un error y no se ejecutará la modificación.
- ***RS2.5*:** Un árbitro no puede estar asignado a más de un partido en un mismo día.
  - RF:2.1
  - RD(s):*RDW5.1*
  - Descripción: Si a un árbitro se le intenta asignar el ID de un partido que tiene lugar el mismo día que otro partido que ya se le había asignado, el sistema devuelve un error.

2.4. DFD Partidos

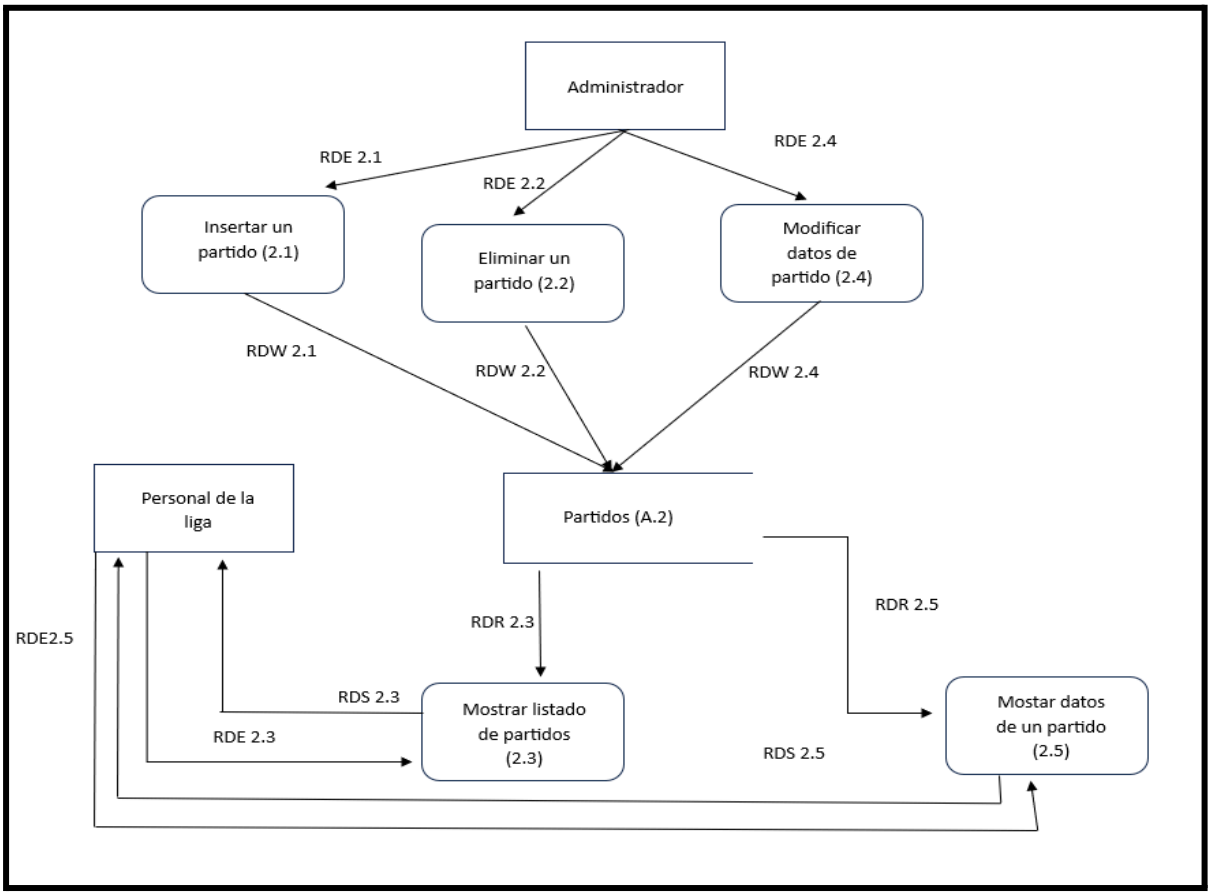
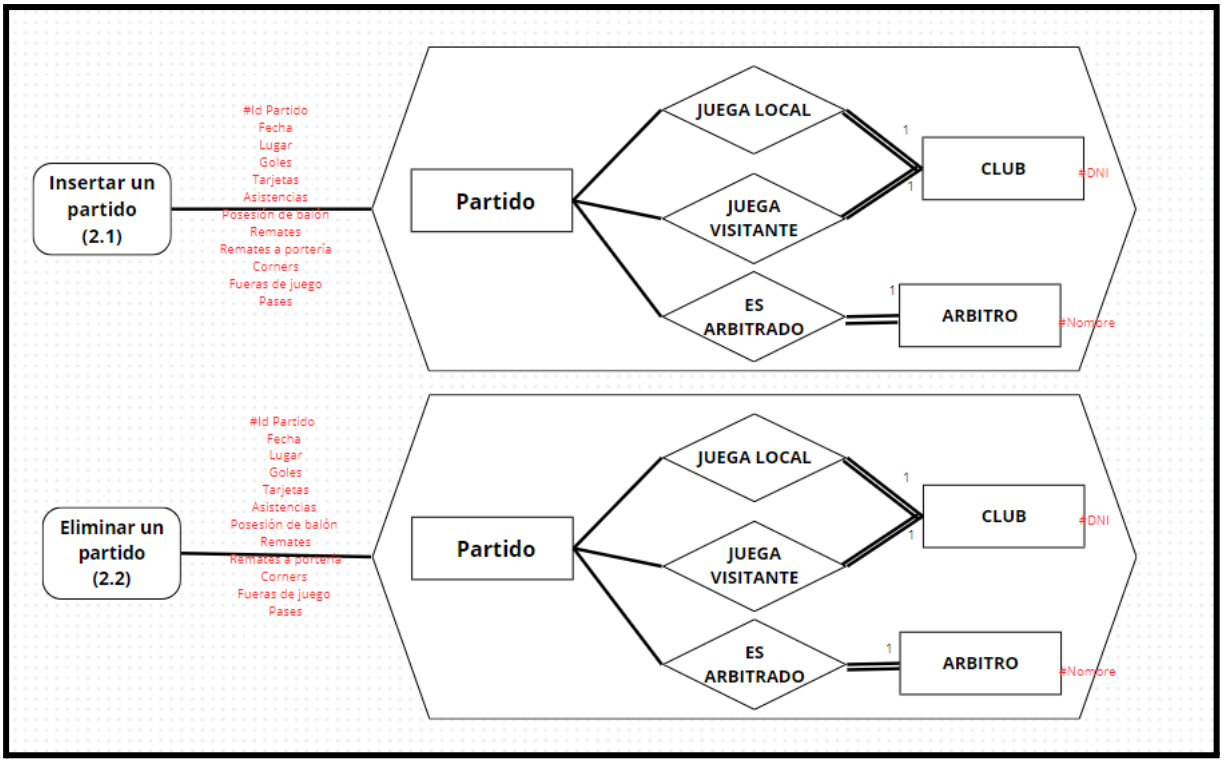
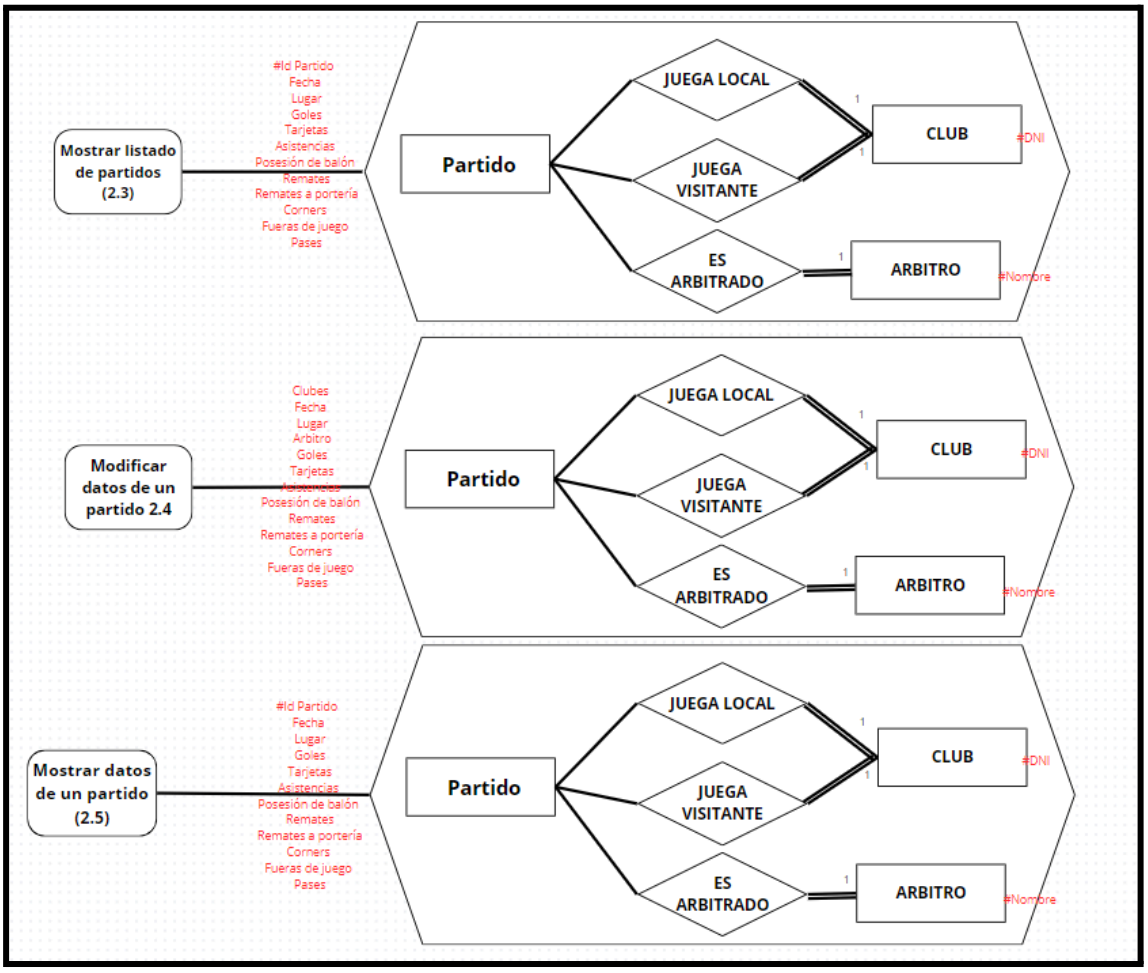


Figura 2.1: DFD Partidos

2.5. Esquema externo DFD1 Partidos


Insertar un partido (2.1)
#Id Partido
Fecha
Lugar
Goles
Tarjetas
Asistencias
Posesión de balón
Remates
Remates a portería
Corners
Fueras de juego
Pases
Partido
JUEGA LOCAL
JUEGA VISITANTE
ES ARBITRADO
CLUB
#DNI
ARBITRO
#Nombre
Eliminar un partido (2.2)
#Id Partido
Fecha
Lugar
Goles
Tarjetas
Asistencias
Posesión de balón
Remates
Remates a portería
Corners
Fueras de juego
Pases
Partido
JUEGA LOCAL
JUEGA VISITANTE
ES ARBITRADO
CLUB
#DNI
ARBITRO
#Nombre


Mostrar listado de partidos (2.3)
#Id Partido
Fecha
Lugar
Goles
Tarjetas
Asistencias
Posesión de balón
Remates
Remates a portería
Corners
Fueras de juego
Pases
Partido
JUEGA LOCAL
JUEGA VISITANTE
ES ARBITRADO
CLUB
#DNI
ARBITRO
#Nombre
Modificar datos de un partido 2.4
Clubes
Fecha
Lugar
Arbitro
Goles
Tarjetas
Asistencias
Posesión de balón
Remates
Remates a portería
Corners
Fueras de juego
Pases
Partido
JUEGA LOCAL
JUEGA VISITANTE
ES ARBITRADO
CLUB
#DNI
ARBITRO
#Nombre
Mostrar datos de un partido (2.5)
#Id Partido
Fecha
Lugar
Goles
Tarjetas
Asistencias
Posesión de balón
Remates
Remates a portería
Corners
Fueras de juego
Pases
Partido
JUEGA LOCAL
JUEGA VISITANTE
ES ARBITRADO
CLUB
#DNI
ARBITRO
#Nombre

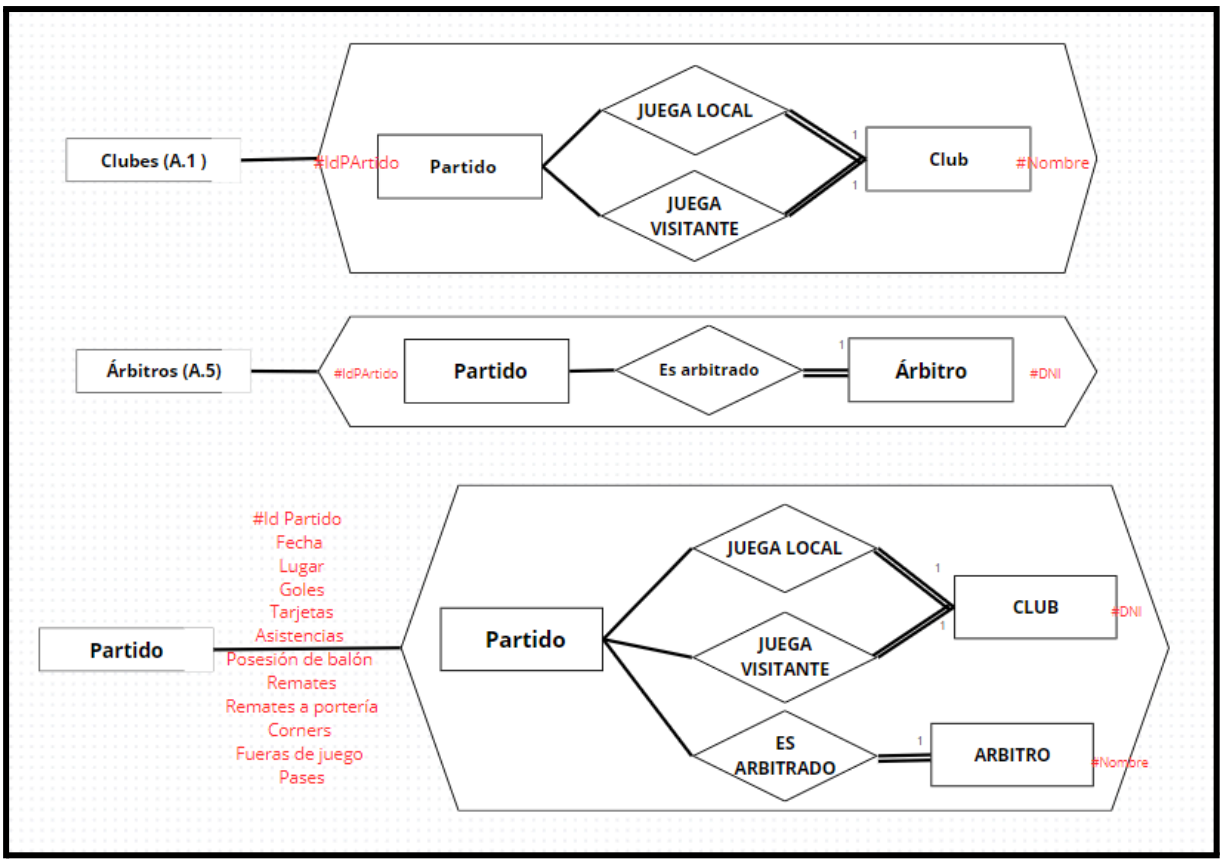


Figura 2.2: Esquemas externos de DFD Partidos

2.6. Esquema conceptual Partidos

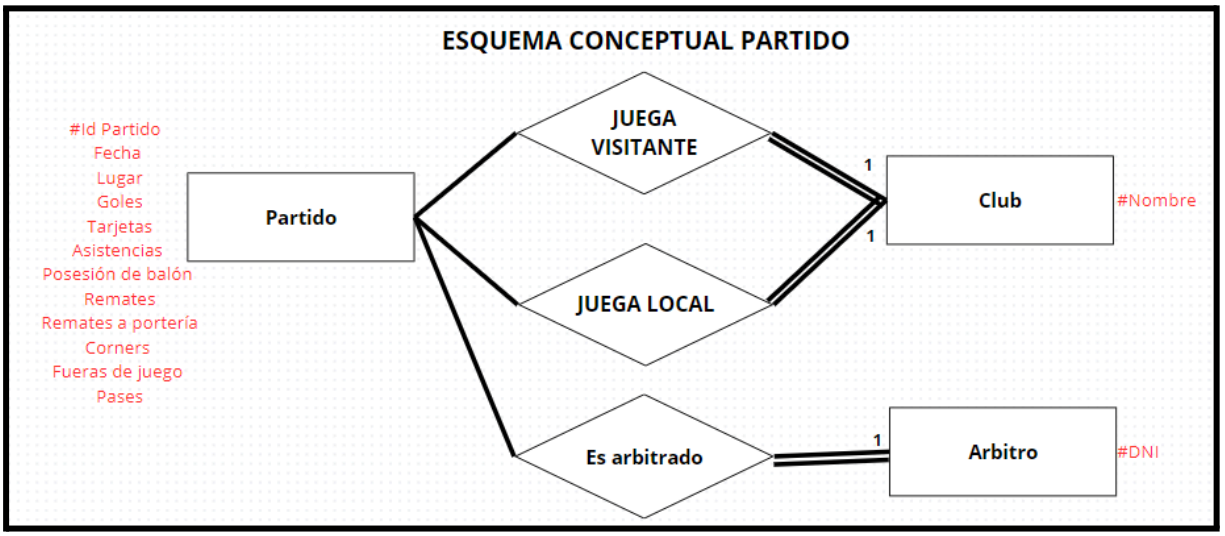


Figura 2.3: Esquema conceptual del subsistema Partidos

3. SUBSISTEMA DE JUGADORES (Javier Gomez Lopez)

3.1. Descripción

El subsistema de jugadores estará creado para varios usuarios (los administradores) que manejarán la información y situación de los jugadores dentro de nuestra liga. De cada jugador, almacenaremos su nombre (en una serie de hasta 20 caracteres), su apellido (en una serie de hasta 40 caracteres), su DNI (en una serie de hasta 9 caracteres, fijo una vez que esté en base de datos), su estatura (en números de 3 dígitos, enteros positivos), el club al que pertenece, fecha de nacimiento (en una serie de datos de tipo Date), su nacionalidad (en una serie de hasta 10 caracteres). Además, cada jugador tendrá asociado un número de goles, tarjetas y asistencias (en series de 2 caracteres numéricos cada dato, todos positivos).
Para **dar de alta a un nuevo jugador**, el usuario (administrador) deberá proporcionar su nombre, apellido, DNI (el cual debe de estar correctamente escrito), estatura, fecha de nacimiento (debe de ser mayor de 18 años para jugar) y nacionalidad, datos que el sistema almacenará, confirmando la inserción o dando un error. Para **dar de baja a un jugador**, el usuario deberá proporcionar el DNI del jugador, confirmando el borrado o dando un error. El sistema también permitirá **mostrar un listado de jugadores** con todos sus datos a petición del usuario. Para **modificar los datos de un jugador**, el usuario deberá proporcionar los datos que desea modificar (no tiene que ser necesario modificar todos), y el sistema almacenará, en caso de cambio, dichos datos, confirmando la modificación o dando un error. Por último, el sistema también permitirá al usuario, previa indicación del club deseado, un **listado de jugadores por club** con todos sus datos.

3.2. Requisitos funcionales

- ***RF3.1: Dar de alta un jugador***.
    - Entrada: Agente externo: administrador. Acción: Solicitar inserción. Requisitos de datos de entrada: *RDE3.1*.
    - Base de datos: Requisito de datos de escritura: *RDW3.1.*
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE3.1*: Datos de entrada de alta del jugador:
        - Nombre: Cadena de caracteres (20).
        - Apellido: Cadena de caracteres (40).
        - DNI: Cadena de caracteres (9),
        - Estatura: Cadena de números (3).
        - Fecha de nacimiento: Tipo Date.
        - Nacionalidad: Cadena de caracteres (10).
        - Lista de partidos jugados: Array de enteros (IDs) de los partidos. (Dinámico).
        - Histórico de clubes de la liga: Array de cadenas de caracteres (20) de los nombres de los clubes (Dinámico)
    - *RDW3.1*: Datos almacenados del jugador.

        - Los mismo que RDE3.1.
- ***RF3.2: Dar de baja un jugador.***
    - Entrada: Agente externo: administrador. Acción: Solicitar borrado. Requisitos de datos de entrada: *RDE3.2.*
    - Base de datos: Requisito de datos de escritura: *RDW3.2.*
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE3.2:* Datos de entrada de baja de jugador.
        - DNI: Cadena de caracteres (9).
    - *RDW3.2*: Datos almacenados del jugador.
        - Los mismos que *RDW3.1*. De existir una relación con otros elementos del sistema, se hará un borrado en cascada.

- ***RF3.3: Mostrar listado de jugadores.***
    - Entrada: Agente externo: administrador. Acción: solicitar listado de jugadores. Requisitos de datos de entrada: ninguno.
    - Base de datos: Requisito de datos de lectura: *RDR3.3*.
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: *RDS3.3.*
    - *RDR3.3*: Datos de jugadores almacenados:
        - Los mismos datos que *RDW3.1*.
        - *RDS3.3:* Listado de registros, cada uno de ellos con los mismos datos que *RDW3.1.*
- ***RF3.4: Modificar datos de un jugador.***
    - Entrada: Agente externo: administrador. Acción: modificar datos de un jugador. Requisitos de datos de entrada: *RDE3.4.*
    - Base de datos: Requisito de datos de escritura: *RDW3.4.*
    - Salida: Agente externo: administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - RDE3.4: Datos de entrada de modificación del jugador:
        - DNI: Cadena de caracteres (9), que no se modificará, solo servirá para búsqueda.
        - Los mismo que *RDE3.1*.
    - *RDW3.4*: Datos almacenados del jugador.
        - Los utilizados en *RDE3.4*.
- ***RF3.5: Mostrar listado de jugadores de un club***.
    - Entrada: Agente externo: personal del club (los administradores pueden realizar todas las acciones de este agente externo). Acción: solicitar listado de jugadores de un club en específico. Requisitos de datos de entrada: *RDE3.5*.
    - Base de datos: Requisito de datos de lectura: *RDR3.5.*
    - Salida: Agente externo: entrenador, personal del club. Acción: confirmación de resultado. Requisito de datos de salida: *RDS3.5*.
    - *RDE3.5*: Datos de entrada de listado de jugadores por club:

        - Club: Nombre del club.
    - *RDR3.5*: Datos de jugadores de dicho club almacenados:
        - Los mismos datos que *RDW3.*1.
    - *RDS3.5*: Listado de registros, cada uno de ellos con los mismos datos que *RDW3.1*, para el club especificado.

3.3. Restricciones semánticas

- ***RS3.1***: Para dar de alta a un jugador, el DNI debe de estar correctamente escrito.
    - RF: *RF3.1.*
    - RD(s): *RDW3.1.*
    - Descripción: Si el DNI que introduce el usuario no es un DNI correcto, es decir, no sigue el formato de 8 números y una letra, o esta letra no es la correcta respecto a esos 8 números, no se inserta el nuevo jugador y se devuelve un error.
- **RS3.2**: Cada jugador sólo podrá estar inscrito una vez en el sistema por temporada. ○ RF: RF3.1. DESAPARECE ESTE RF
    - RD(s): RDW3.1.
    - Descripción: Si el jugador (su DNI) ya está inscrito en el sistema con un en la actual temporada, el sistema no insertará al nuevo jugador y se devuelve un error.
- **RS3.3:** El DNI de cada jugador será fijo una vez esté en base de datos.
    - RF: RF3.4.
    - RD(s): RDW3.4.
    - Descripción: El sistema nunca modificará el DNI en base de datos, solo servirá de búsqueda del jugador que se desea modificar.

3.4. DFD Jugadores

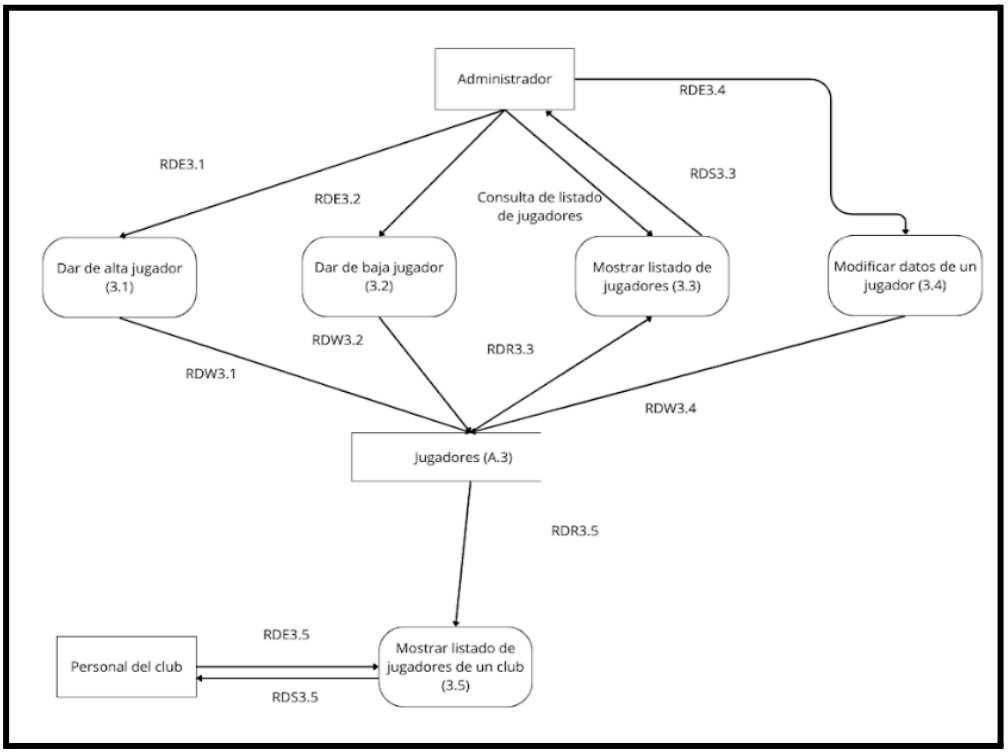
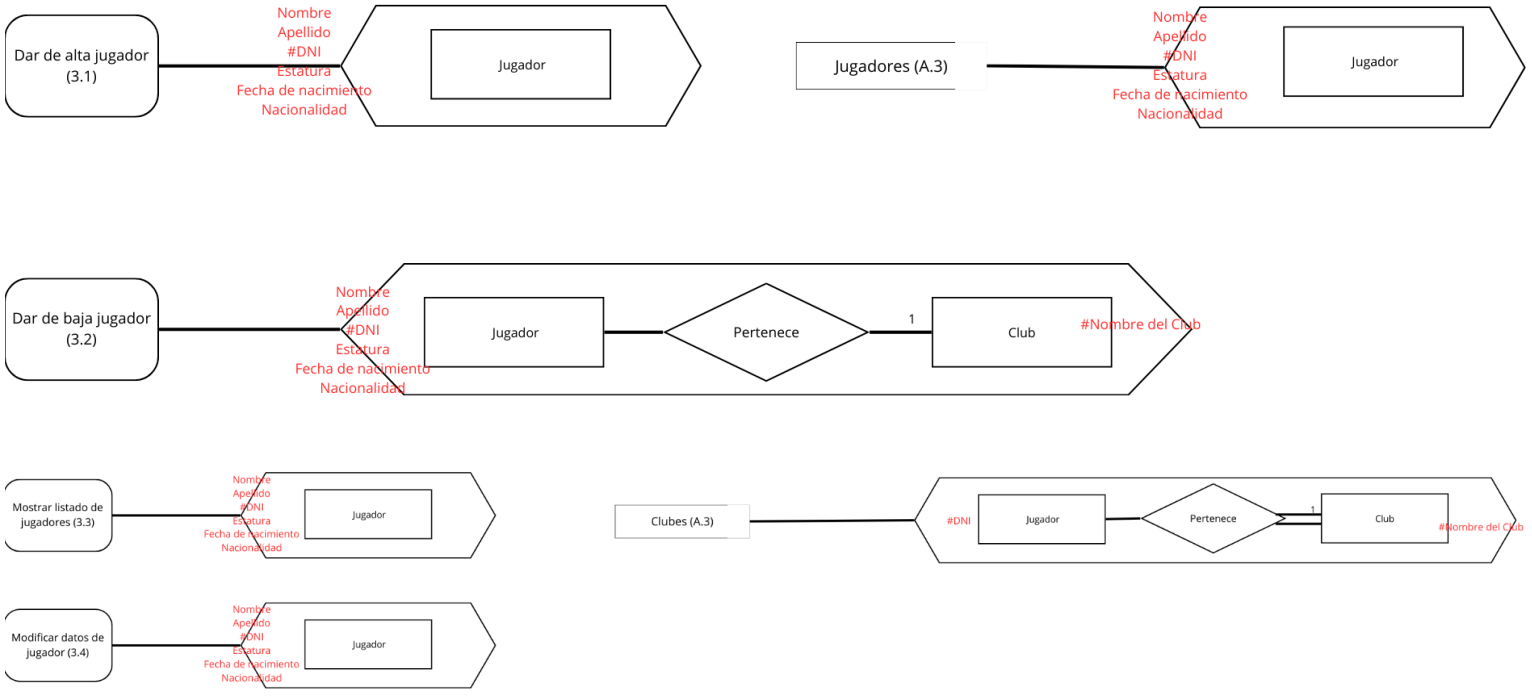


Figura 3.1: DFD Jugadores

3.5. Esquemas externos de DFD1 en Jugadores



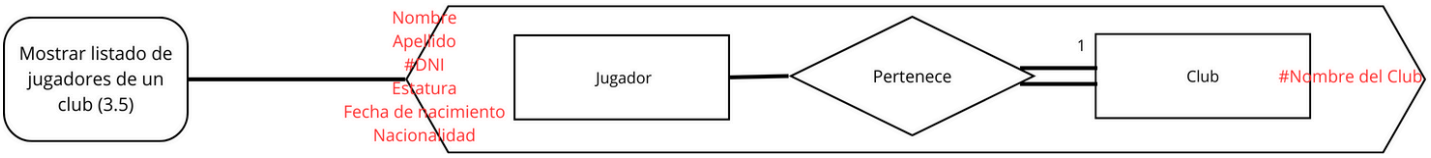


Figura 3.2: Esquemas externos Jugadores
https://t.ly/E6NHt

3.6. Esquema conceptual Jugadores

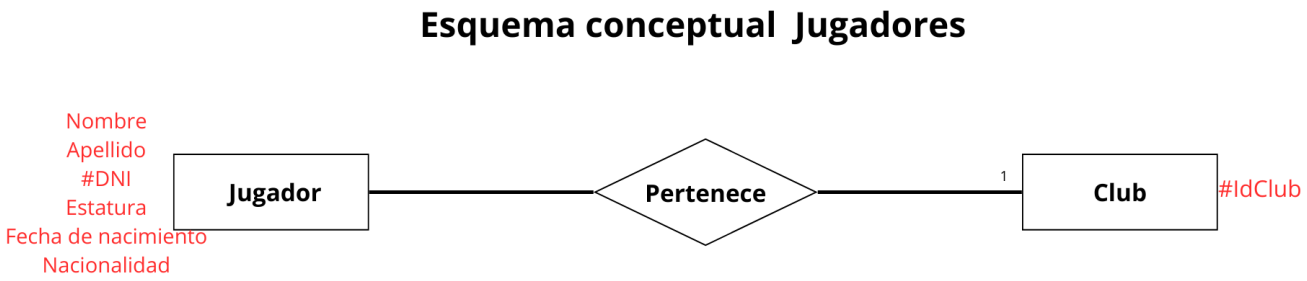


Figura 3.3: Esquema conceptual Jugadores

4. SUBSISTEMA DE ENTRENADORES (Valeria Borrajo Yusty)

4.1. Descripción

Este subsistema se encarga de gestionar la información proporcionada sobre los entrenadores de cada equipo que forma la liga.
Se presenta una ficha personal, la cual contiene el nombre (en una serie de hasta 20 caracteres), apellidos (en una serie de hasta 40 caracteres), fecha de nacimiento (en una serie de datos tipo Date) y DNI (en una serie de 9 caracteres numéricos menos el último que es una letra).
En primer lugar, por una parte se puede **dar de alta** un nuevo entrenador, donde el usuario ha de almacenar los datos personales, confirmando la inserción o dando un error. Y por otra parte, se puede **dar de baja** proporcionando el DNI del entrenador y así confirmando el borrado o dando nuevamente un error.
El sistema permitirá al personal de la liga **visualizar un listado** de entrenadores donde se muestra detalladamente su experiencia y palmarés.
Además, para **modificar los datos** de un entrenador, se introducen aquellos que se deseen modificar y el sistema los almacena, confirmando que se ha hecho el cambio correctamente o dando fallo.
Por último, el personal de la liga podrá pedir que se le **muestre el club asociado a un entrenador** , con previa indicación del entrenador deseado.

4.2. Requisitos funcionales

- ***RF4.1: Dar de alta un entrenador***.
    - Entrada: Agente externo: usuario (Administrador). Acción: Solicitar inserción. Requisitos de datos de entrada: *RDE4.1*.
    - Base de datos: Requisito de datos de escritura: *RDW4.1.*
    - Salida: Agente externo: usuario (Administrador). Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE4.1*: Datos de entrada de alta del entrenador:
        - Nombre: Cadena de caracteres (20).
        - Apellido: Cadena de caracteres (40).
        - DNI: Cadena de caracteres (9).
        - Fecha de nacimiento: Tipo Date.
    - *RDW4.1*: Datos almacenados del entrenador:
        - Los mismos que *RDE4.1.*

- ***RF4.2: Dar de baja un entrenador.***
    - Entrada: Agente externo: usuario (Administrador). Acción: Solicitar borrado. Requisitos de datos de entrada: *RDE4.2*.
    - Base de datos: Requisito de datos de escritura: *RDW4.*2.
    - Salida: Agente externo: usuario (Administrador). Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE4.2*: Datos de entrada de baja del entrenador:

        - DNI: Cadena de caracteres (9).
    - *RDW4.2*: Datos almacenados del entrenador:
        - Los mismos datos que *RDW4.1*, y aquellos que tengan relación con otros objetos de la BD, se realizará un borrado en cascada.

- ***RF4.3: Mostrar listado de entrenadores***.
    - Entrada: Agente externo: personal de la liga (los administradores pueden realizar todas las acciones de este agente externo). Acción: solicitar listado de entrenadores. Requisitos de datos de entrada: ninguno.
    - Base de datos: Requisito de datos de lectura: *RDR4.3*.
    - Salida: Agente externo: personal de la liga. Acción: confirmación de resultado. Requisito de datos de salida: RDS4.3.
    - *RDR4.3*: Datos de entrenadores almacenados:
        - Los mismos datos que *RDW4.1*, pero no se mostrará el DNI de cada entrenador, solo su nombre, para respetar la privacidad de los campos.
    - *RDS4.3*: Listado de registros, cada uno de ellos con los mismos datos que *RDR4.3*.

- ***RF4.4: Modificar datos de un entrenador.***
    - Entrada: Agente externo: usuario (Administrador). Acción: Modificar información de un entrenador. Requisitos de datos de entrada: *RDE4.4.*
    - Base de datos: Requisito de datos de escritura: *RDW4.4.*
    - Salida: Agente externo: personal de la liga(los administradores pueden realizar todas las acciones de este agente externo). . Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE4.4*: Datos de entrada de modificación del entrenador:
        - DNI: Cadena de caracteres (9).
        - De los datos de *RF4.1*, los que se desee modificar de nuestro entrenador.
    - *RDW4.4*: Datos almacenados del entrenador.
        - Los utilizados en *RDE4.4*, excepto el DNI que no se puede modificar, es inmutable.
- ***RF4.5: Entrenador de un club de una temporada.***
    - Entrada: Agente externo: personal de la liga (los administradores pueden realizar todas las acciones de este agente externo). Acción: solicitar dato donde se muestra el ID de un entrenador y el nombre del club que está entrenado. Requisitos de datos de entrada: *RDE4.5.*
    - Base de datos: Requisito de datos de lectura: *RDR4.5*.
    - Salida: Agente externo: personal de la liga. Acción: confirmación de resultado. Requisito de datos de salida: *RDS4.5.*
    - *RDE4.5*: Datos de entrada del entrenador con el club que entrena.
        - Los mismos datos que *RDW4.1*, pero no se mostrará el DNI de cada entrenador, solo su nombre y el nombre del club que tiene asociado.
    - *RDR4.5*: Datos del entrenador de dicho club almacenados:

        - Los mismos datos que *RDE4.1*.
    - *RDS4.5*: Listado de registros, cada uno de ellos con los mismos datos que *RDE4.1*, para el entrenador especificado.

4.3. Restricciones semánticas

- ***RS4.1***: Para dar de alta a un entrenador, el DNI debe de estar correctamente escrito.
    - RF: *RF4.1.*
    - RD(s): *RDW4.1*.
    - Descripción: Si el DNI que introduce el usuario no es un DNI correcto, es decir, no sigue el formato de 8 números y una letra, o esta letra no es la correcta respecto a esos 8 números, no se inserta el nuevo jugador y se devuelve un error.
- ***RS4.2***: Cada entrenador solo puede estar asociado a un club.
    - RF: *RF4.1.*
    - RD(s): *RDW4.1*.
    - Descripción: Si un entrenador (su DNI) ya está inscrito en el sistema asociado a un club, el sistema no insertará al nuevo entrenador y devolverá un error.
- ***RS4.3***: Dar de baja a un entrenador no existente.
    - RF:4.2.
    - RD(s): *RDW4.2.*
    - Descripción: Si se intenta dar de baja un entrenador utilizando un DNI que no corresponde a ningún entrenador registrado en el sistema, no se realizará ninguna acción de borrado y devolverá un error.
- ***RS4.4***: Edad mínima del entrenador.
    - RF: *RF4.1.*
    - RD(s): *RDW4.1*.
    - Descripción: El entrenador debe ser mayor de edad. Al registrarse en el sistema se calcula usando la fecha de nacimiento proporcionada, y si es menor de 18 años, no se permitirá la inserción y se devolverá un error.

4.4. DFD Entrenadores

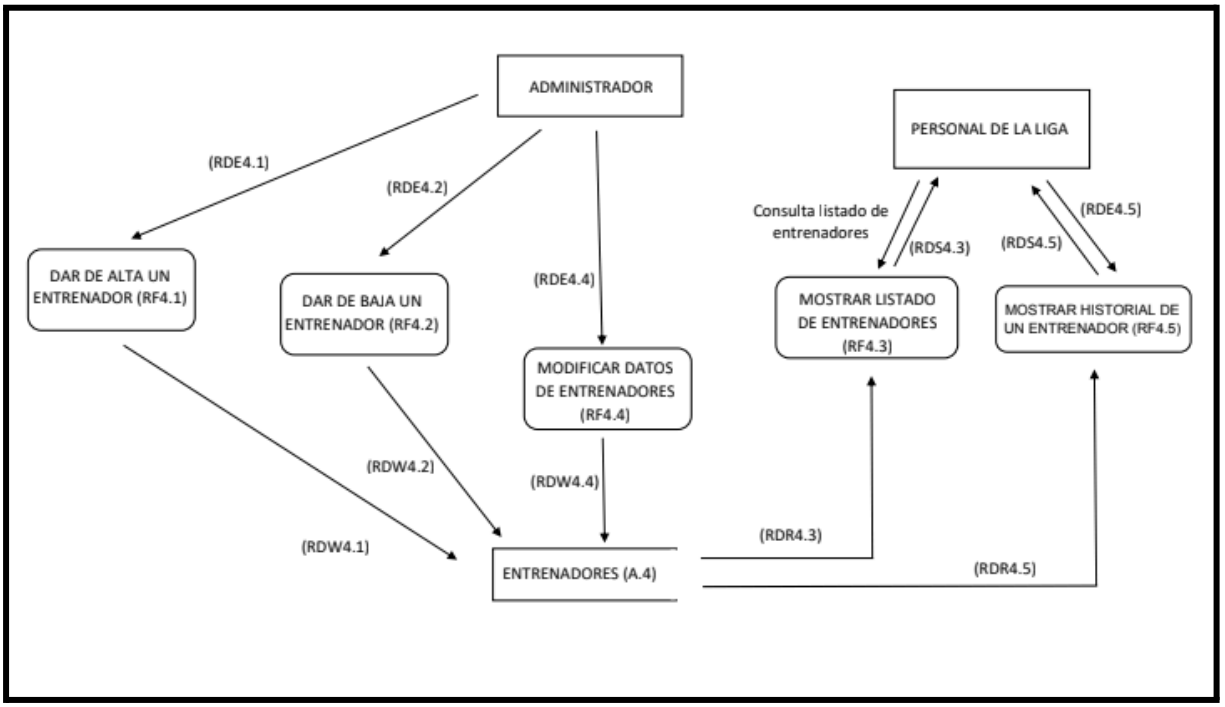


Figura 4.1: DFD Entrenadores

4.5. Esquema Externo DFD1 Entrenadores

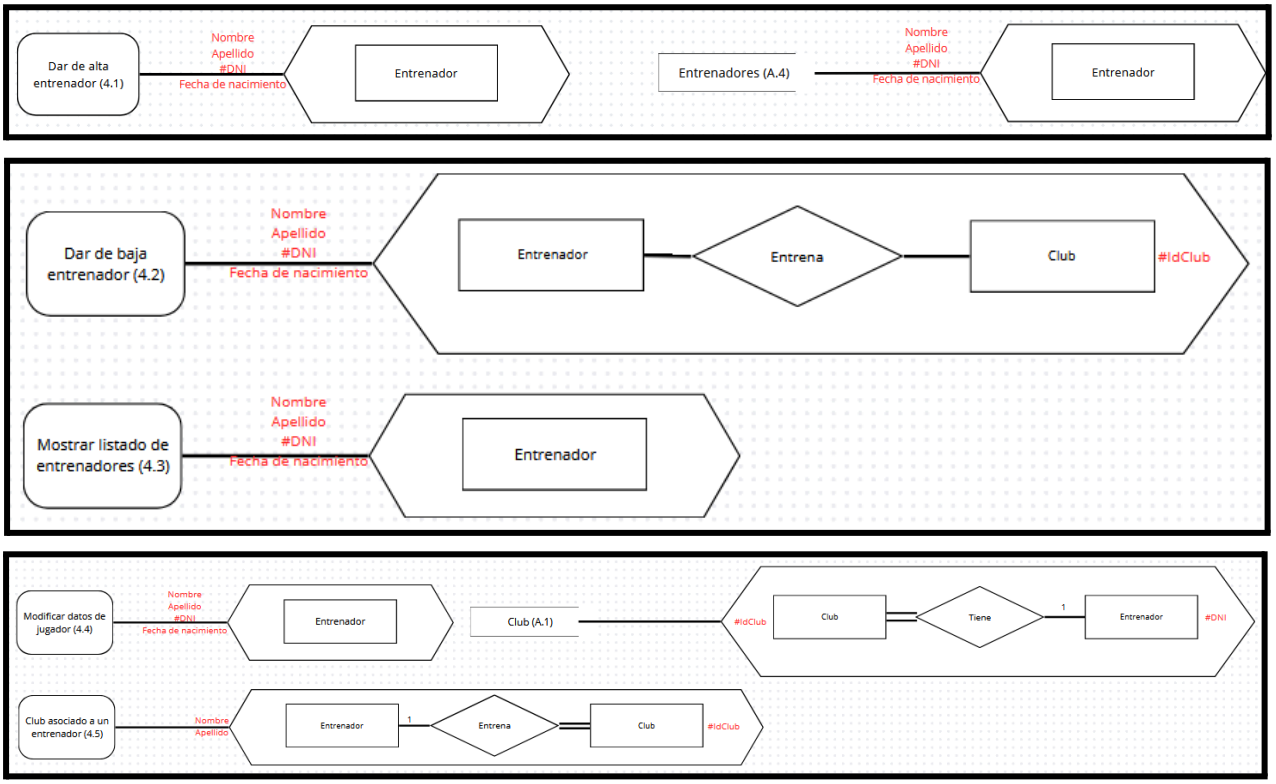


Figura 4.2: Esquemas externos de DFD Entrenadores

4.6. Esquema conceptual Entrenadores

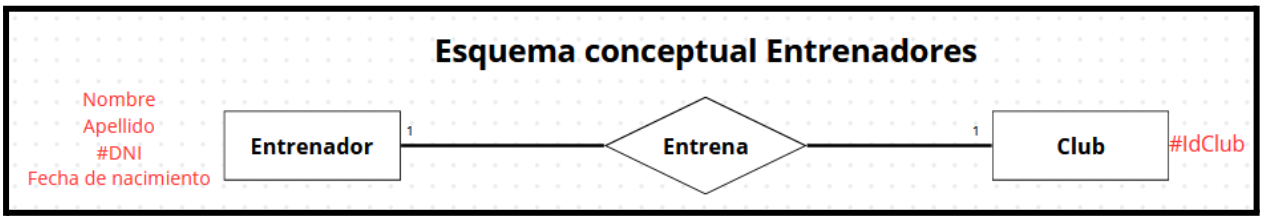


Figura 4.3: Esquema conceptual del Subsistema Entrenadores

5. SUBSISTEMA DE ÁRBITROS (Buenaventura Porcel Esquivel)

5.1. Descripción

Este subsistema se va a desarrollar con el objetivo de gestionar la información sobre los árbitros que supervisarán cada partido de liga. De cada árbitro se almacenará su DNI (9 caracteres de los cuales los 8 primeros serán números y el último una letra), nombre (20 caracteres), apellidos (40 caracteres), fecha de nacimiento (10 caracteres con la especificación de formato tipo fecha), colegio de procedencia (20 caracteres con la especificación "Colegio" + "Comunidad autónoma" + "de Árbitros"), un histórico de partidos arbitrados y una lista de futuras asignaciones.

Para **dar de alta** un Árbitro, será necesario que el usuario introduzca su DNI, nombre, apellidos, fecha de nacimiento. Todos estos datos serán almacenados por el sistema, confirmando la inserción o devolviendo un error. Para **dar de baja** un Árbitro será necesario especificar el DNI del mismo, confirmando el borrado o devolviendo un error. El sistema también **mostrará un listado** con todos los árbitros por solicitud del usuario, mostrando todos los datos de estos. En el caso de que fuera necesario hacer una **modificación** en alguno de los datos de un árbitro, se indicará el DNI del árbitro a modificar y el campo que es necesario ajustar, devolviendo el sistema una confirmación del cambio o un mensaje de error. Además de estas funcionalidades, el usuario podrá solicitar que el sistema muestre un listado de los **próximos partidos asignados** a un árbitro, mostrando el ID del partido.

5.2. Requisitos funcionales

- ***RF5.1: Dar de alta un árbitro***.
    - Entrada: Agente externo: Administrador. Acción: Solicitar inserción. Requisitos de datos de entrada: *RDE5.1*.
    - Base de datos: Requisito de datos de escritura: *RDW5.1.*
    - Salida: Agente externo: usuario. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE5.1*: Datos de entrada de alta del árbitro:
        - Nombre: Cadena de caracteres (20).
        - Apellido: Cadena de caracteres (40).
        - DNI: Cadena de caracteres (9),

        - **Colegio de árbitros: Cadena de caracteres (20),**
        - Fecha de nacimiento: Tipo Date.
        - **Lista de partidos arbitrados: Array de enteros (IDs) de los partidos. (Dinámico). (Eliminado)**
        - **Lista de futuras asignaciones: Array de enteros (IDs) de los partidos que vaya a arbitrar en las próximas jornadas. (Dinámico) (Eliminado)**
    - *RDW5.1*: Datos almacenados del árbitro:
        - Los mismos datos que RDE5.1.
- ***RF5.2: Dar de baja un árbitro.***
    - Entrada: Agente externo: Administrador. Acción: Solicitar borrado. Requisitos de datos de entrada: *RDE5.2*.
    - Base de datos: Requisito de datos de escritura: *RDW5.2.*
    - Salida: Agente externo: Administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE5.2*: Datos de entrada de baja del árbitro:
        - DNI: Cadena de caracteres (9).
    - *RDW5.2:* Datos almacenados del árbitro:
        - Los mismos datos que *RDW5.1* y aquellos que tengan relación con otros objetos de la BD, se realizará un borrado en cascada.
- ***RF5.3: Mostrar listado de árbitros de la liga.***
    - Entrada: Agente externo: Personal del club (el administrador puede realizar todas las acciones del personal del club). Acción: Solicitar listado de árbitros. Requisitos de datos de entrada: ninguno.
    - Base de datos: Requisito de datos de lectura: *RDR5.3*.
    - Salida: Agente externo: personal del club, árbitro. Acción: confirmación de resultado. Requisito de datos de salida: *RDS5.3.*
    - *RDR5.3*: Datos de árbitros almacenados:
        - Los mismos que *RDW5.1*, pero no se mostrará el DNI de cada árbitro, solo su nombre, para respetar la privacidad de los campos.
    - RDS5.3: Listado de registros cada uno con los mismos datos que *RDR5.3*.

- ***RF5.4: Modificar datos de un árbitro***.
    - Entrada: Agente externo: Administrador. Acción: modificar datos de un árbitro. Requisitos de datos de entrada: *RDE5.4*.
    - Base de datos: Requisito de datos de escritura: *RDW5.4.*
    - Salida: Agente externo: Administrador. Acción: confirmación de resultado. Requisito de datos de salida: ninguno.
    - *RDE5.4*: Datos de entrada de modificación del árbitro:
        - DNI: Cadena de caracteres (9).
        - De los datos de *RDE5.1*, los que se desee modificar de nuestro jugador.
    - *RDW5.4*: Datos almacenados del jugador.
        - Los utilizados en *RDE5.4,* excepto el DNI que es inmutable.

- ***RF5.5: Mostrar listado de futuros partidos asignados a un árbitro.***
    - Entrada: Agente externo: personal de la liga. Acción: Solicitar listado de futuros partidos asignados. Requisitos de datos de entrada*: RDE5.5*.
    - Base de datos: Requisito de datos de escritura: *RDR5.5.*
    - Salida: Agente externo: personal de la liga. Acción: confirmación de resultado. Requisito de datos de salida: *RDS5.5.*
    - *RDE5.5*: Datos de entrada del árbitro:
        - Los mismos que *RDE5.2.*
    - *RDR5.5*: Datos de entrada del árbitro del que queremos mostrar los partidos asignados:
        - Los mismos que RDW5.1, solo el campo de futuras asignaciones.
    - *RDS5.5*: Listado de registros con el ID del partido asignado (array de enteros).

5.3. Restricciones semánticas

- ***RS5.1***: Para dar de alta a un árbitro, el DNI debe de estar correctamente escrito.
    - RF: *RF5.1*.
    - **RD(s): *RDW5.1***.
    - Descripción: Si el DNI que introduce el usuario no es un DNI correcto, es decir, no sigue el formato de 8 números y una letra, o esta letra no es la correcta respecto a esos 8 números, no se inserta el nuevo jugador y se devuelve un error.
- ***RS5.2***: Cada árbitro sólo podrá estar inscrito una vez en el sistema por temporada.
    - RF: R5.1.
    - **RD(s): *RDW5.1*.**
    - Descripción: Si el árbitro (su DNI) ya está inscrito en el sistema con un en la actual temporada, el sistema no insertará al nuevo árbitro y se devuelve un error.
- ***RS5.3****:* La fecha de nacimiento de un árbitro debe ser anterior a la fecha actual.
    - RF:5.1
    - **RD(s): *RDW5.1***
    - Descripción: Si la fecha de nacimiento del árbitro es igual o posterior al día actual, el sistema no insertará al nuevo árbitro y devolverá un error.

5.4. DFD Árbitros

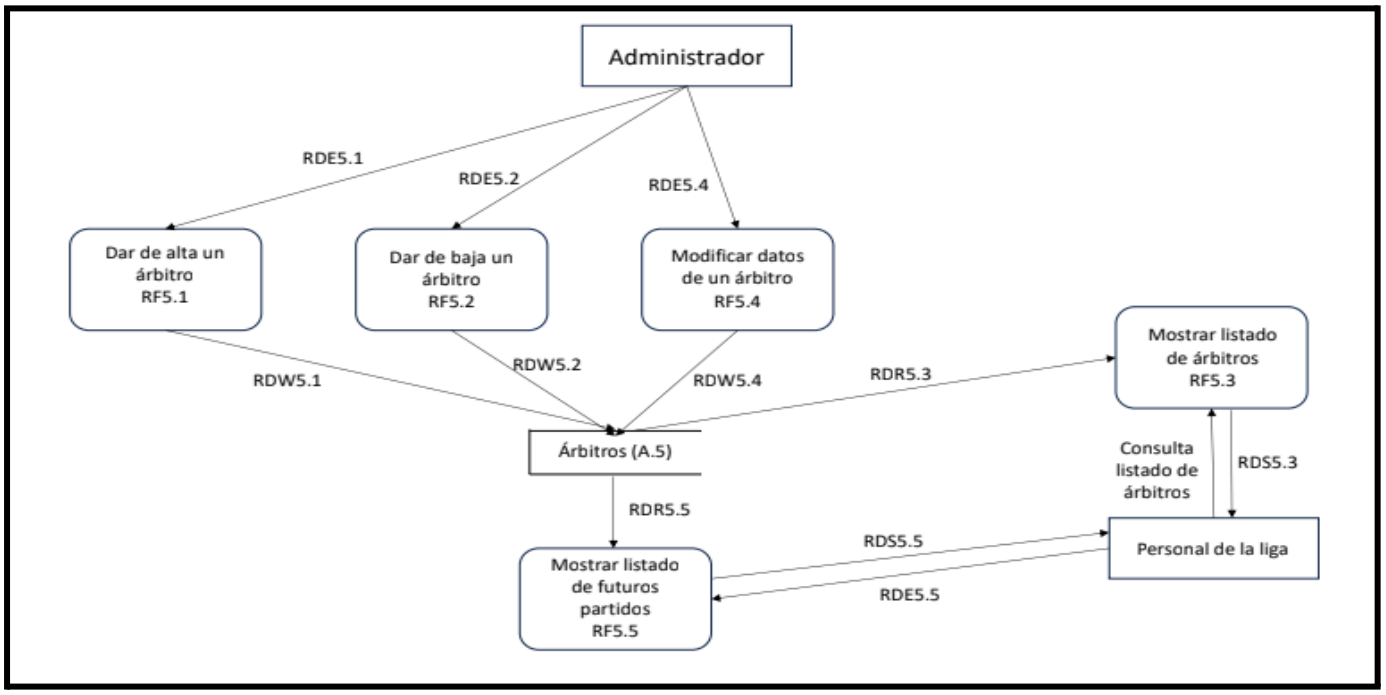


Figura 5.1: DFD Árbitros

5.5. Esquema Externo DFD1 Arbitros

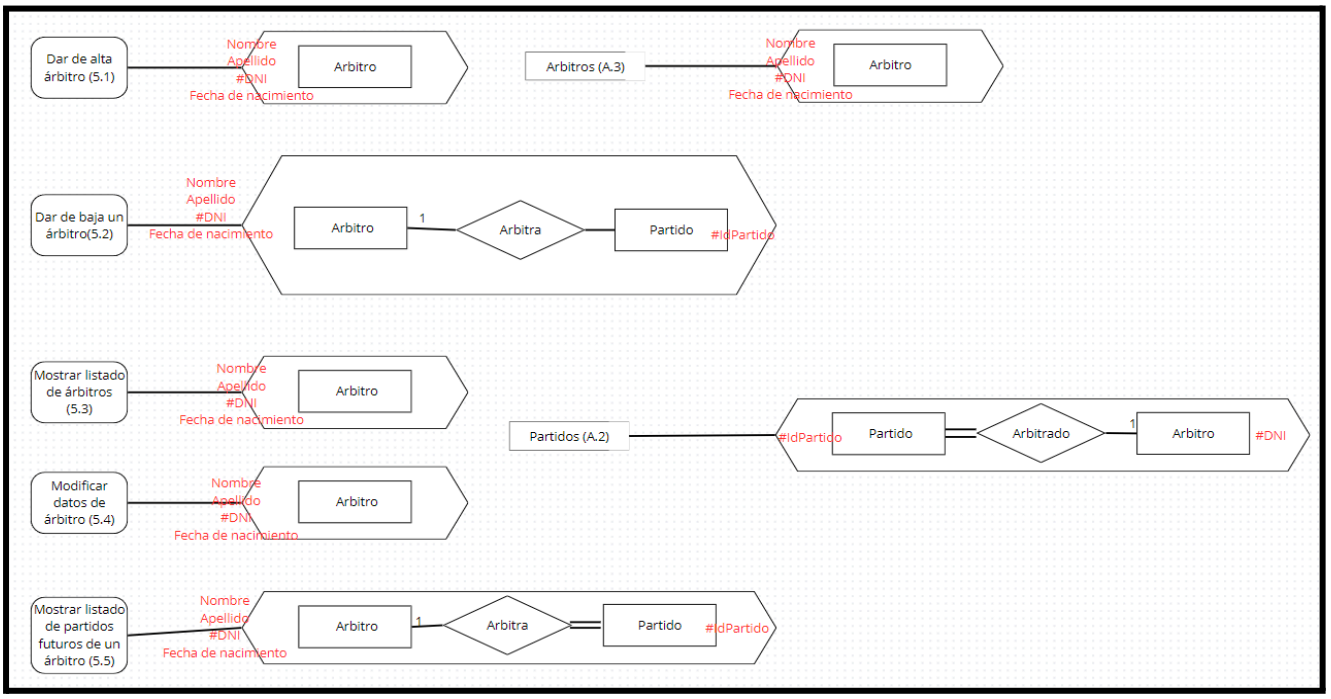


Figura 5.2: Esquemas externos de DFD Árbitros

5.6. Esquema conceptual Arbitros

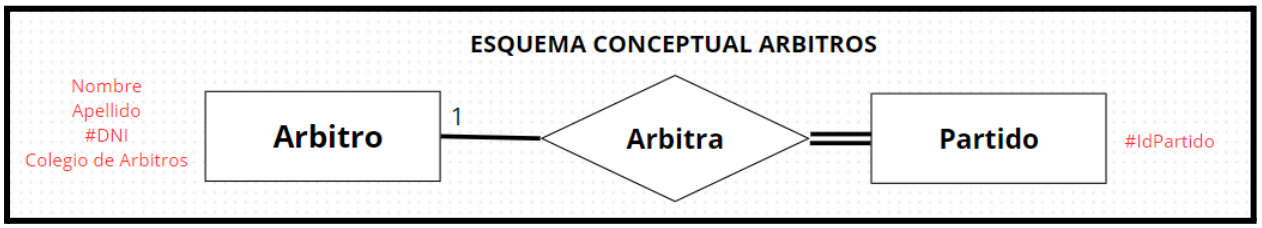


Figura 5.3: Esquemas conceptual del Subsistema Árbitros

6. SUBSISTEMA DE CLASIFICACIÓN (Manuel Díaz-Meco Terres)

6.1. Descripción

Este subsistema es el encargado de calcular el resultado y la posición de cada equipo dentro de nuestra liga de fútbol. Los datos almacenados en este subsistema serían: los puntos de cada equipo en la clasificación general, la posición, así como los partidos jugados: ganados, perdidos y empatados. También se almacenaría los goles marcados, los recibidos y la diferencia. Todos estos datos se almacenarán en una serie de hasta 5 caracteres numéricos. Por último, se indicaría si el equipo ha mejorado, empeorado o se ha mantenido igual respecto a la jornada anterior.

Para **asignar una jornada**, se necesitará el id del partido al que se le quiera asignar esa jornada, comprobando si el id de la jornada es positivo, se indicará si hay un error o si el resultado ha sido asignada correctamente, actualizando de esta forma los puntos del club y los partidos jugados. Para **eliminar una jornada** ya introducida será necesario el ID de la Jornada, **al igual que antes comprobamos que el id de jornada es positivo** y el resultado será la confirmación o un mensaje de error. Para **modificar una jornada** de la clasificación se necesitarán el ID de la Jornada a modificar y el ID del partido oportuno, el id de la jornada ha de ser positivo y tiene que haberse introducido previamente. Para **actualizar la clasificación** se hará un barrido por todas las jornadas para poder calcular las posiciones por jornada y por club mediante los datos de los partidos asignados en cada jornada. Por último, el sistema también permitirá **mostrar clasificación general**, en la que se mostrarán todos los datos mencionados en el párrafo anterior.

6.2. Requisitos funcionales

- **RF6.1:** ***Asignar Jornada***
  - Entrada**:** Agente externo: administrador. Acción: asignar una jornada a un partido. Requisitos de entrada: *RDE6.1.*
  - Base de datos**:** Requisito de datos de escritura: *RDW6.1*
  - Salida**:** Agente externo: administrador. Acción: confirmación de la inserción. Requisitos de datos de salida: ninguno.
  - *RDE6.1*: Datos de entrada para la asignación de una jornada**:**

        - ID de la Jornada.
        - ID del Partido asociado a la Jornada.
    - *RDW6.1:*
        - Los mismos que *RDE6.1*.

- **RF6.2:** ***Eliminar Jornada***
    - Entrada: Agente externo: administrador. Acción: eliminar resultado. Requisitos de entrada: *RDE6.2*.
    - Base de datos: Requisitos de datos de escritura: *RDW6.2*.
    - Salida: Agente externo: administrador. Acción: confirmación de eliminación. Requisitos de datos de salida: ninguno
    - *RDE6.2*: Datos de entrada para la eliminación de una jornada**:**
        - ID de la Jornada.
    - *RDW6.2:*
        - Los mismos que *RDE6.2*.

- **RF6.3:** ***Modificar Jornada***
    - Entrada**:** Agente externo: administrador. Acción: modificar una jornada ya insertada. Requisitos de entrada: *RDE6.3*
    - Base de dato**s:** Requisitos de datos de escritura: *RDW6.3*
    - Salida**:** Agente externo: administrador. Acción: confirmación de la modificación. Requisitos de datos de salida: ninguno.
    - *RDE6.3*: Datos de entrada para la modificación de la jornada**:**
        - ID de la Jornada.
        - ID del Partido a modificar.
    - *RDW6.3:*
        - El ID de Partido pertinente.

- **RF6.4*: Actualizar clasificación***
    - Entrada**:** Agente externo: administrador. Acción: actualizar la clasificación.
    - Base de datos: Requisitos de datos de escritura: *RDW6.4*
    - Salida: Agente externo: administrador. Acción: confirmación de la actualización. Requisitos de datos de salida: ninguno.
    - *RDR6.4*: Datos para el cálculo:
        - Clubes que juegan en cada jornada.
        - Goles marcados por club en cada jornada.
    - *RDW6.4:* Datos a actualizar
        - Puntos actualizados de un club en la clasificación.
        - Goles (marcados/recibidos/diferencia) actualizados de un club en la clasificación.
        - Posición actualizada en la clasificación.
        - Actualización de los partidos jugados (ganados/perdidos/empatados) de un club en la clasificación.

        - Actualización del indicador que muestra si un club ha mejorado, empeorado o se ha mantenido igual respecto a la jornada anterior.

- **RF6.5*: Mostrar Clasificación***
    - Entrada**:** Agente externo: administrador. Acción: solicitar muestreo de datos. Requisito de datos de entrada: ninguno.
    - Base de datos**:** Requisitos de datos de lectura: *RDR6.5*
    - Salida**:** Agente externo: usuario. Acción: confirmación resultado. Requisitos de datos de salida: *RDS6.5.*
    - *RDR6.5*: Datos a mostrar:
        - Posición de cada club.
        - Puntos de cada club
        - Goles de cada club (marcados/recibidos/diferencia).
        - Partidos jugados de cada club (ganados/perdidos/empatados).
        - Indicador de mejora respecto a la anterior jornada.
    - *RDS6.5: Requisitos de datos de salida:*
        - Los mismos que *RDR6.5*

6.3. Restricciones semánticas

- ***RS6.1*:** Un club no puede tener una posición negativa.
    - RF: RF6.4 (Actualizar clasificación).
    - RD: RDW6.4 (Datos a actualizar).
    - Descripción: A la hora de actualizar la clasificación, el sistema debe asegurarse de que ninguna de las posiciones puede estar fuera del intervalo [1, nºde clubes].
- ***RS6.2:*** Un club no puede tener un número de puntos negativos.
    - RF: RF6.4 (Actualizar clasificación).
    - RD: RDW6.4 (Datos a actualizar).
    - Descripción: A la hora de actualizar la clasificación, el sistema debe asegurarse de que los puntos totales de ningún club sean negativos.
- ***RS6.3*:** Los goles de un club (marcados y recibidos) no pueden ser negativos, la diferencia sí.
    - RF: RF6.4 (Actualizar clasificación).
    - RD: RDW6.4 (Datos a actualizar).
    - Descripción: A la hora de actualizar la clasificación, el sistema debe asegurarse de que los goles marcados y recibidos por un club sean mayores o iguales que 0.
- ***RS6.4*:** La suma entre partidos ganados, perdidos y empatados tiene que ser igual a los partidos totales jugados por un club.
    - RF: RF6.4 (Actualizar clasificación).
    - RD: RDW6.4 (Datos a actualizar).

    - Descripción: A la hora de actualizar la clasificación, el sistema debe asegurarse de que la suma entre los partidos ganados, empatados y perdidos sea igual a los partidos totales jugados por cada club.
- ***RS6.5*:** Dos clubes no pueden tener la misma posición.
    - RF: RF6.4 (Actualizar clasificación).
    - RD: RDW6.4 (Datos a actualizar).
    - Descripción: A la hora de actualizar la clasificación, el sistema debe asegurarse de que dos clubes no pueden tener la misma posición. Para desempatar se tendrán en cuenta otras estadísticas (como los goles marcados).
- ***RS6.6*:** Se ha de garantizar la consistencia respecto a los resultados.
    - RF: RF6.4 (Actualizar clasificación).
    - RD: RDW6.4 (Datos a actualizar).
    - Descripción: Los datos ofrecidos por el subsistema 'Partidos' han de ser consecuentes con los datos reflejados en la clasificación. Poniendo un ejemplo, si el subsistema 'Partidos' indica que un determinado club ha ganado 3 partidos, consecuentemente se debe mostrar en la clasificación que el club ha ganado 3 partidos.

6.4. DFD Clasificación

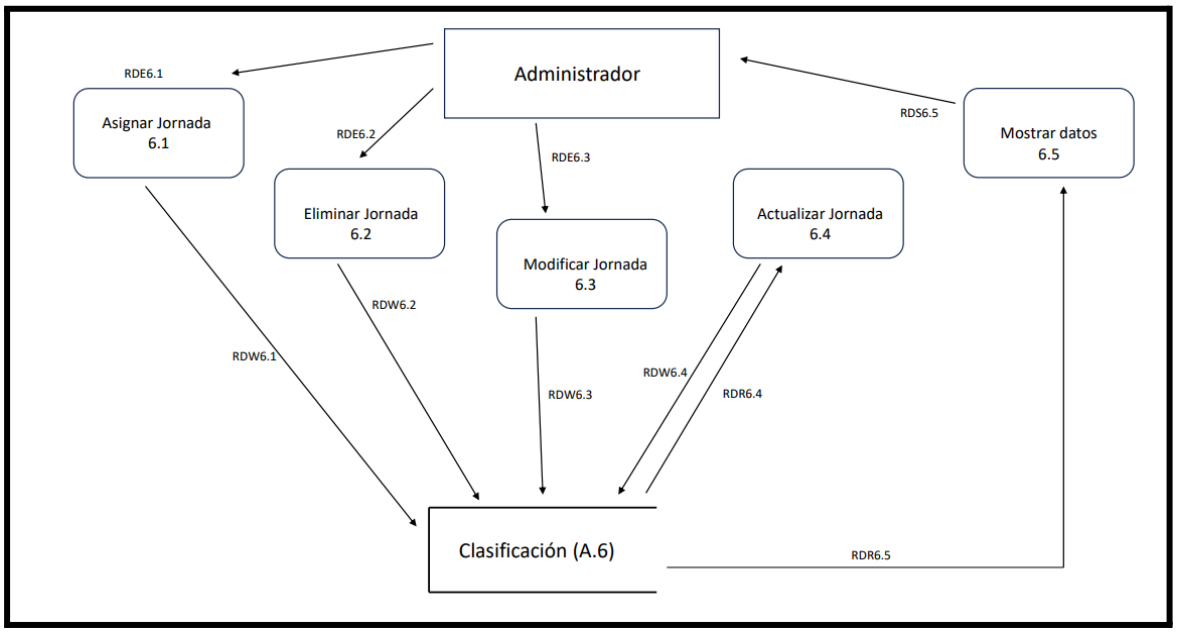


Figura 6.1: DFD Clasificación

6.5. Esquema Externo DFD1 Clasificación

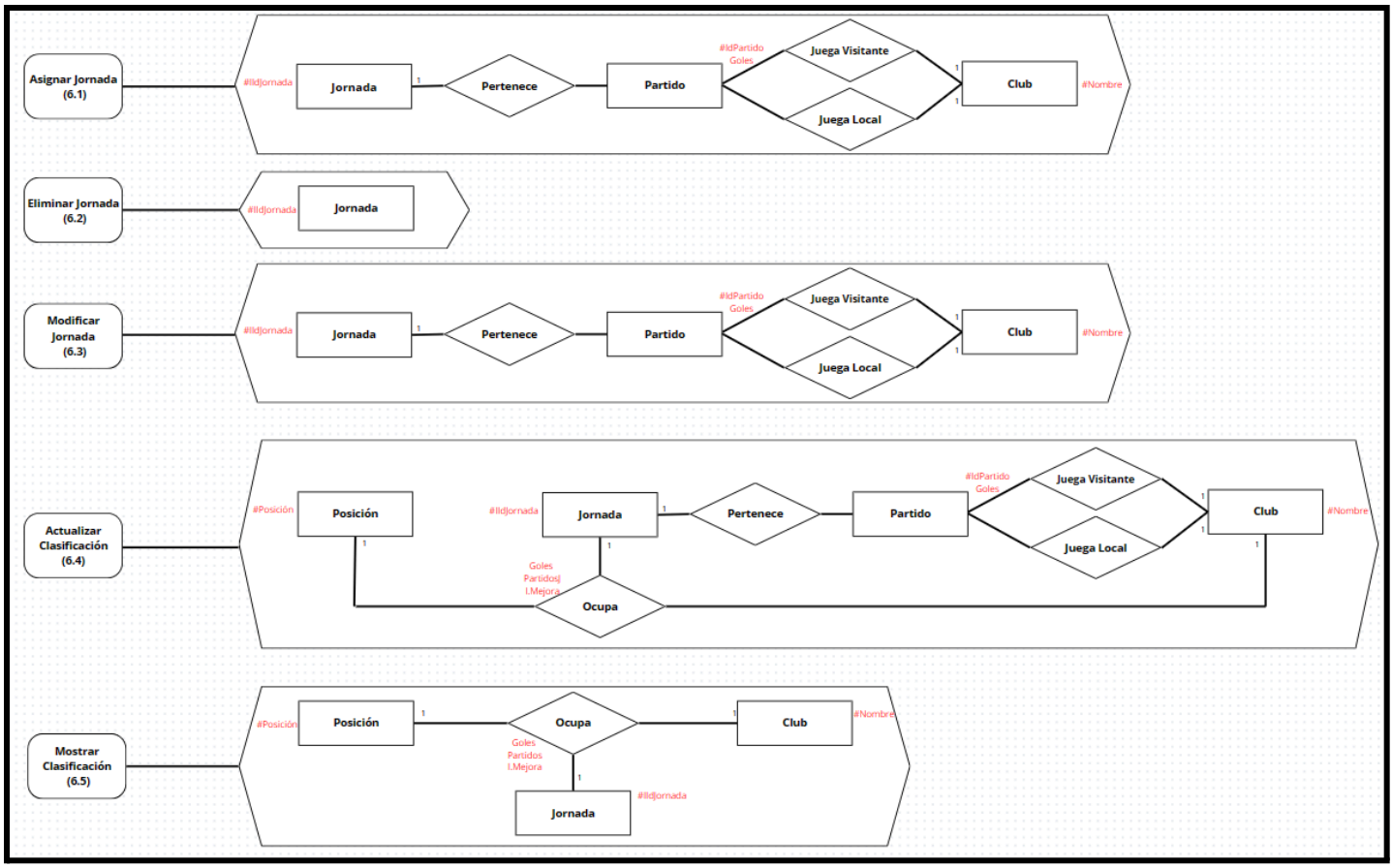


Figura 6.2: Esquemas externos de Procesos DFD Clasificación

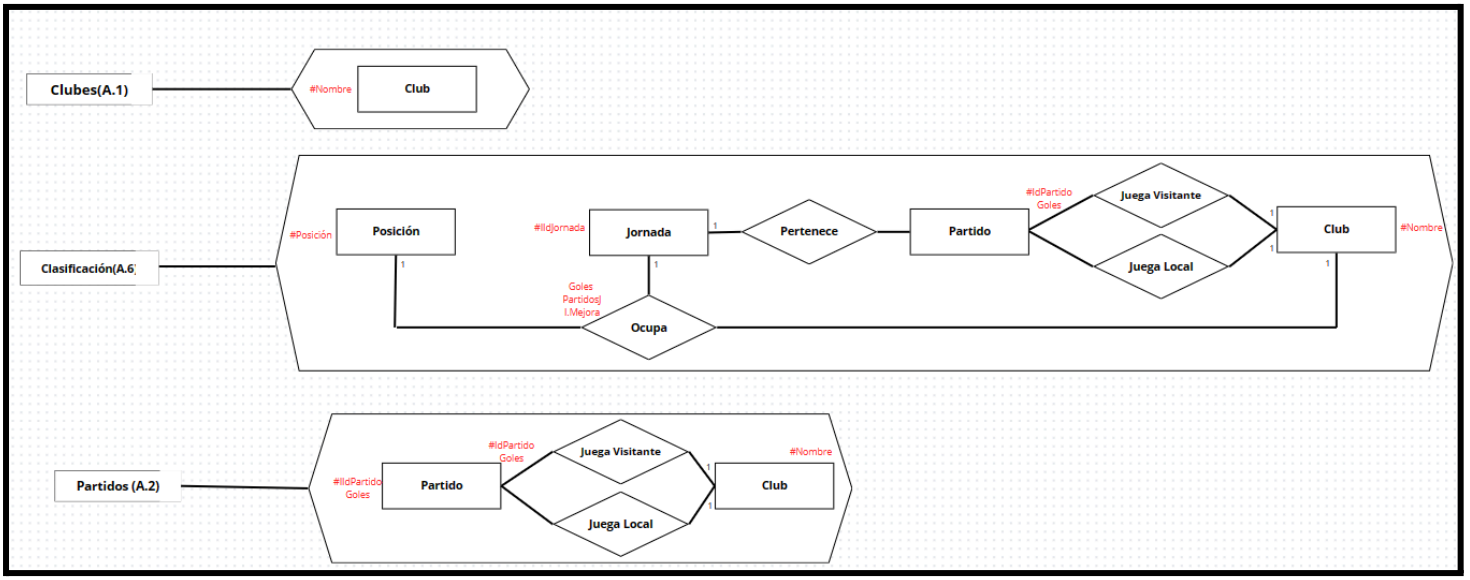


Figura 6.3: Esquemas externos de Almacenes DFD Clasificación

6.6. Esquema Conceptual Clasificación

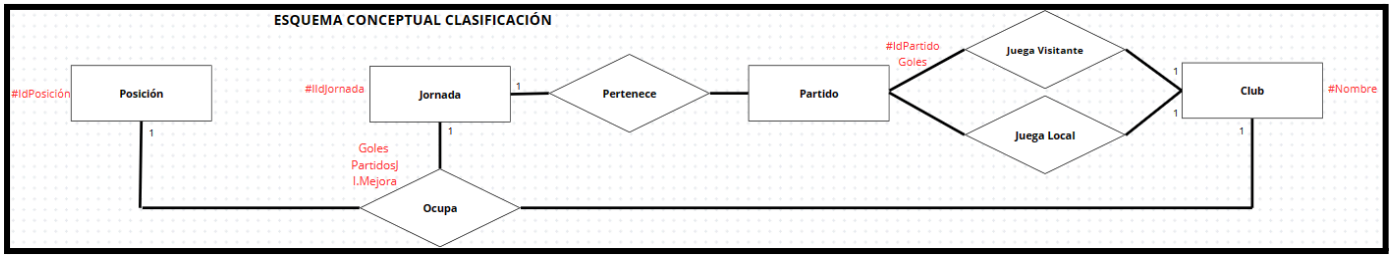


Figura 6.4: Esquemas Conceptual del Subsistema Clasificación

7. DFD Armazón

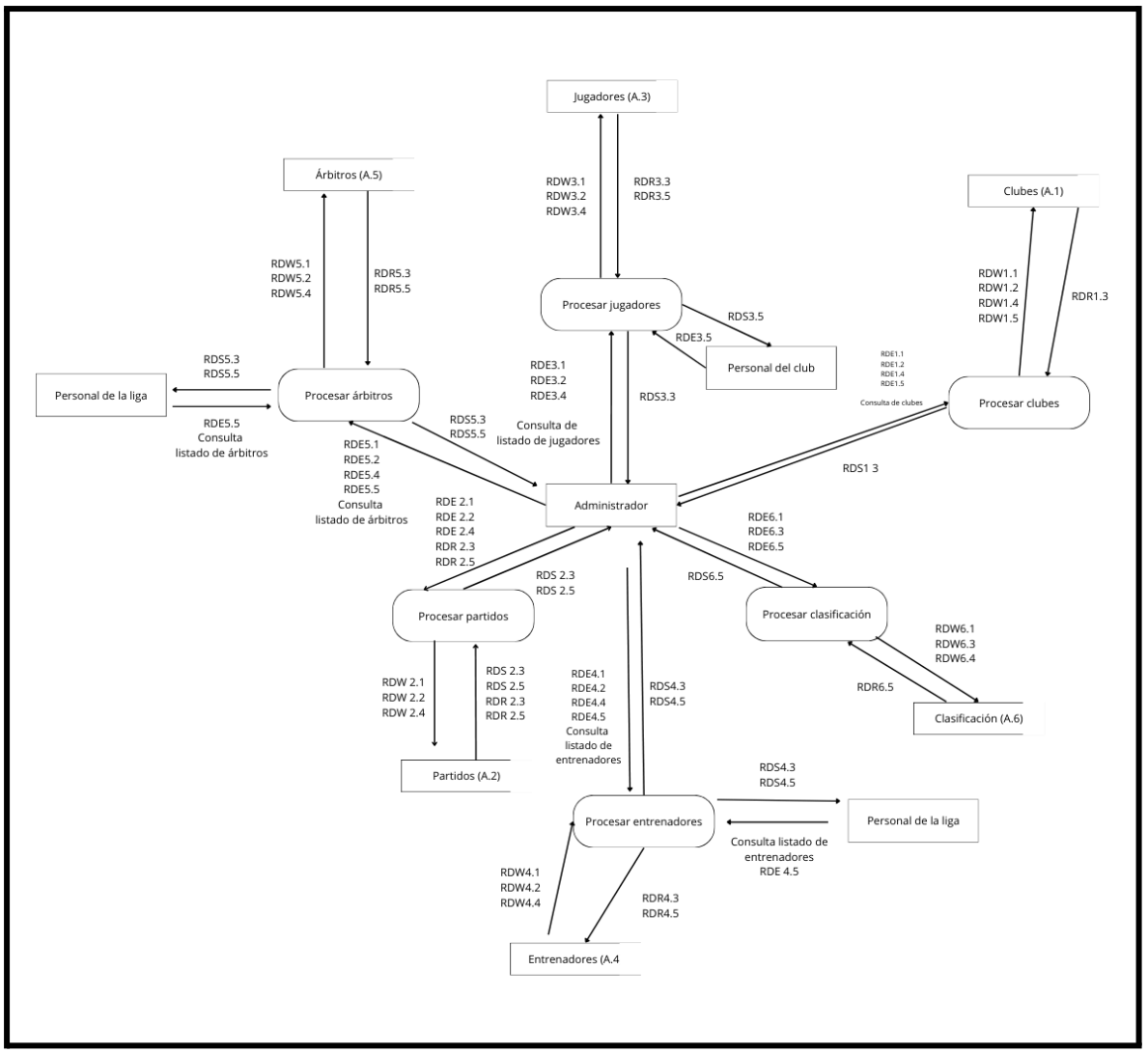


Figura 7: DFD Armazón

Link para verlo ampliado:
https://www.canva.com/design/DAFwwb2qu9Q/pPswI7fe3K6Z_IFQPewxMQ/edit

8. DFD Caja Negra

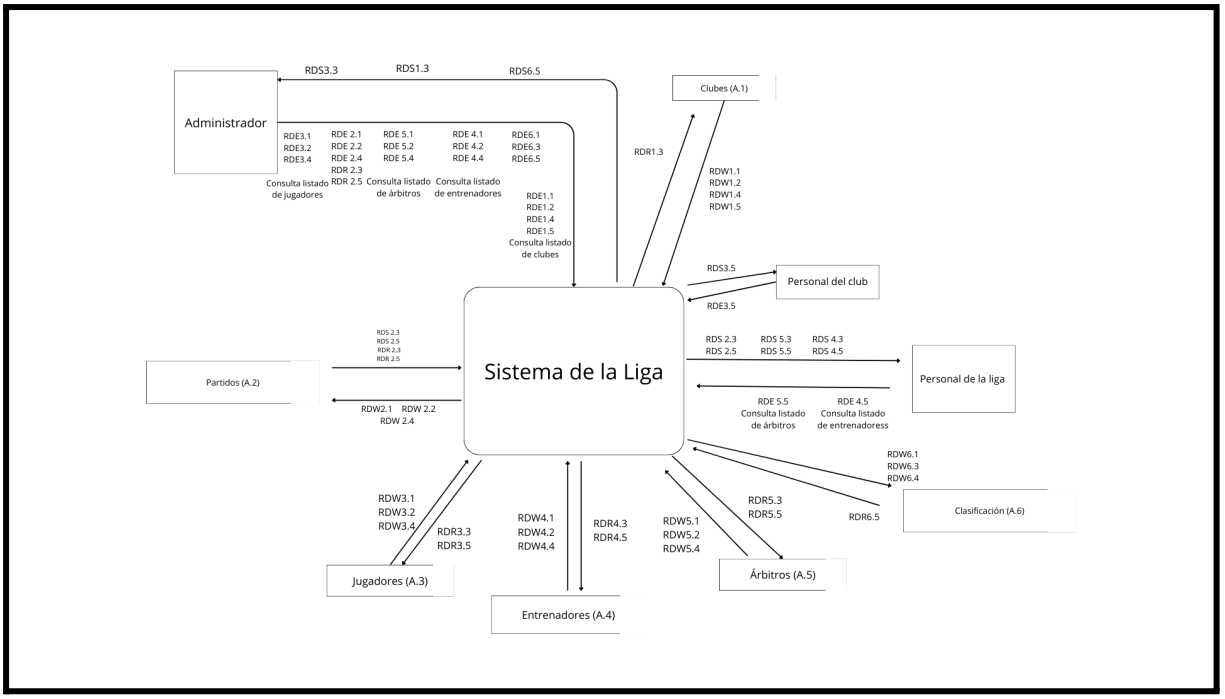


Figura 8: DFD Caja Negra

https://www.canva.com/design/DAFwwrLwbDU/S06a5Hz4RvMg6GT0kIqwmA/edit

9. Esquemas externos DFD Armazón

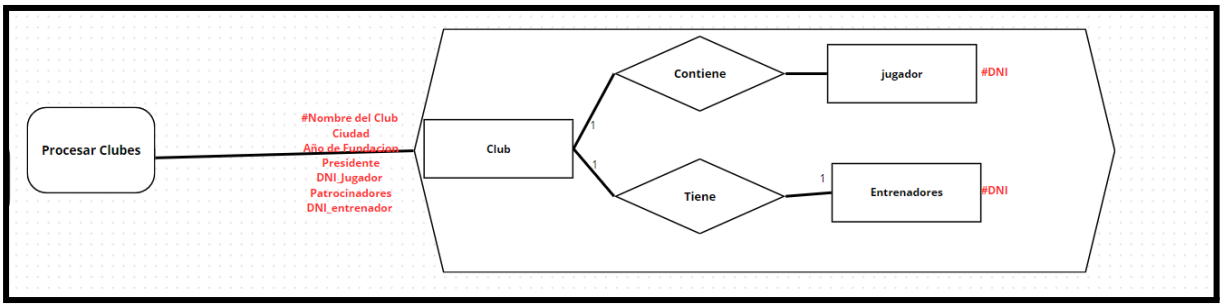


Figura 9.1: Esquema Externo Procesar Club DFD Armazón

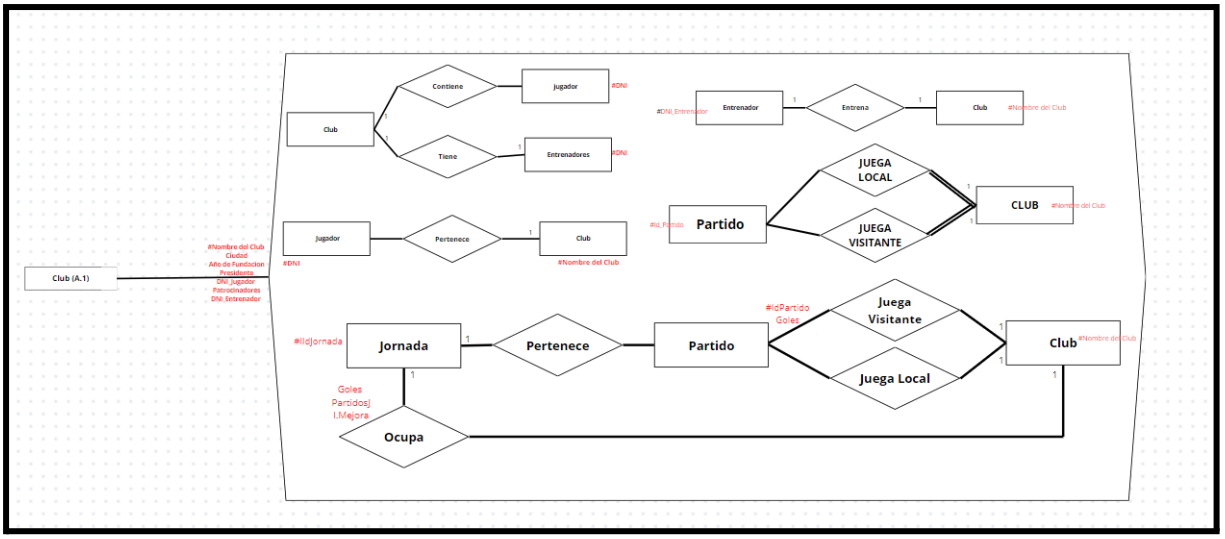


Figura 9.2: Esquema Externo Almacén Club DFD Armazón

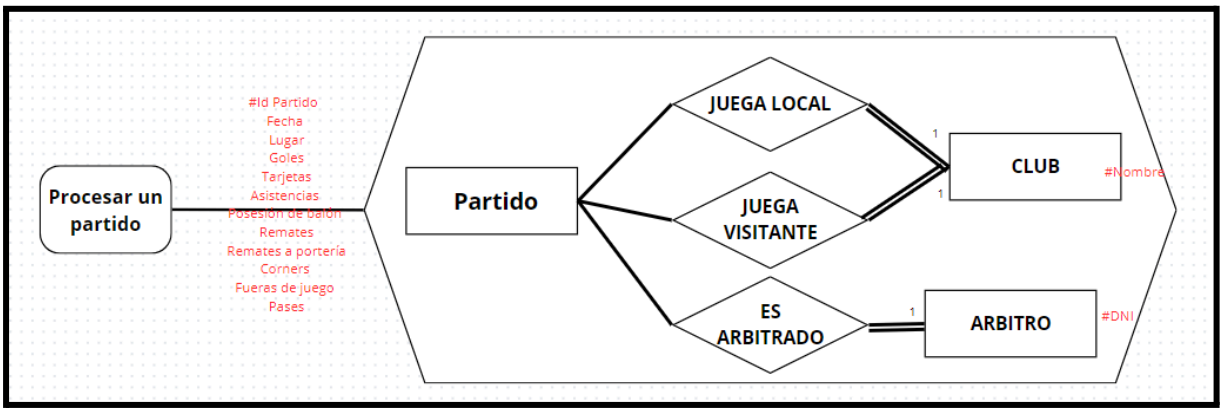


Figura 9.3: Esquema Externo Procesar Partido DFD Armazón

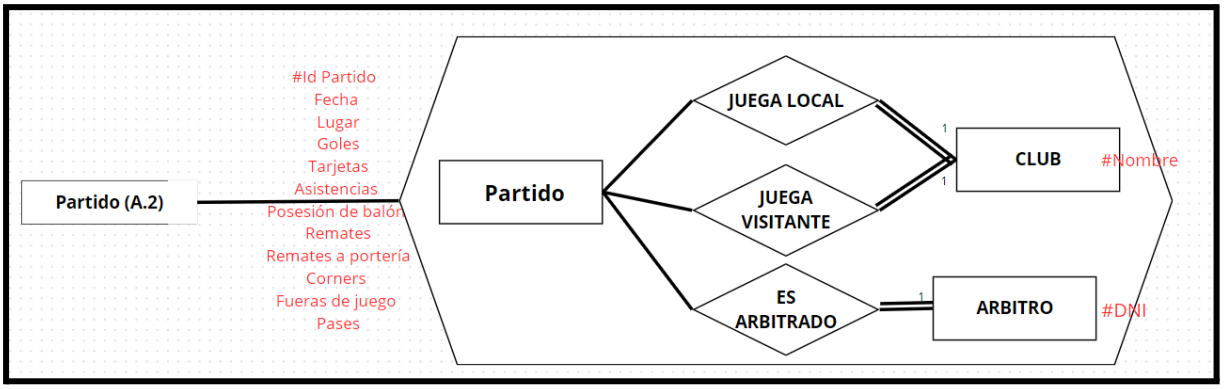


Figura 9.4: Esquema Externo Almacén Partido DFD Armazón

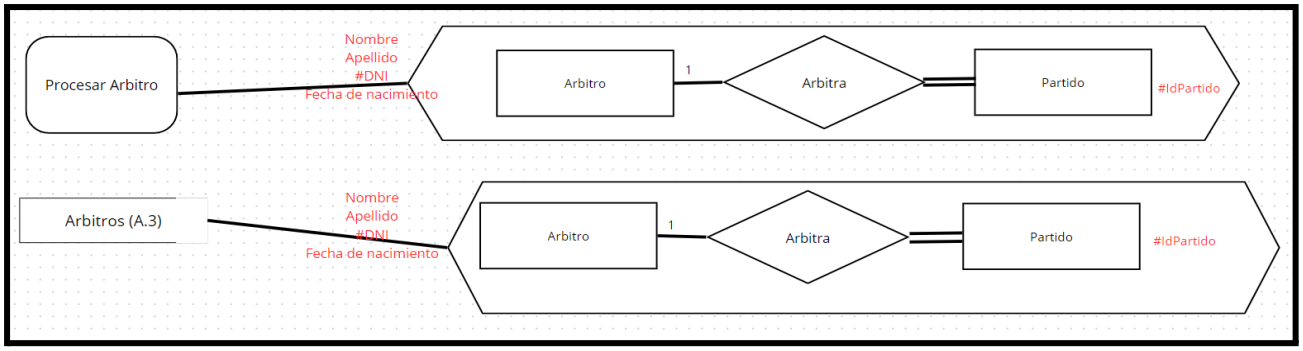


Figura 9.5: Esquema Externo Procesar Árbitro y Almacén Árbitro DFD Armazón

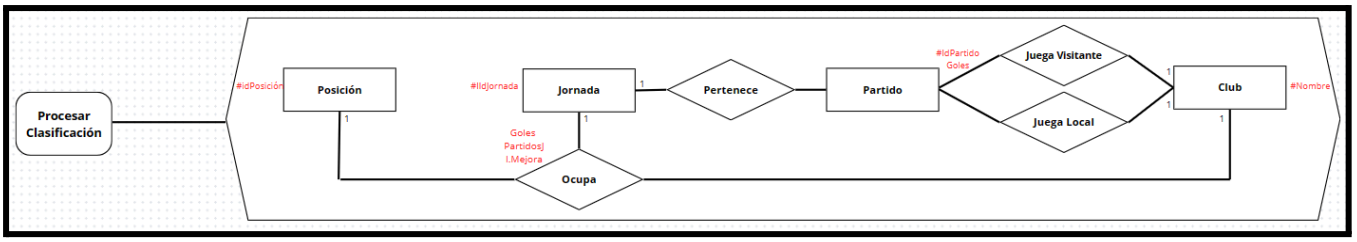


Figura 9.6: Esquema Externo Procesar Clasificación DFD Armazón

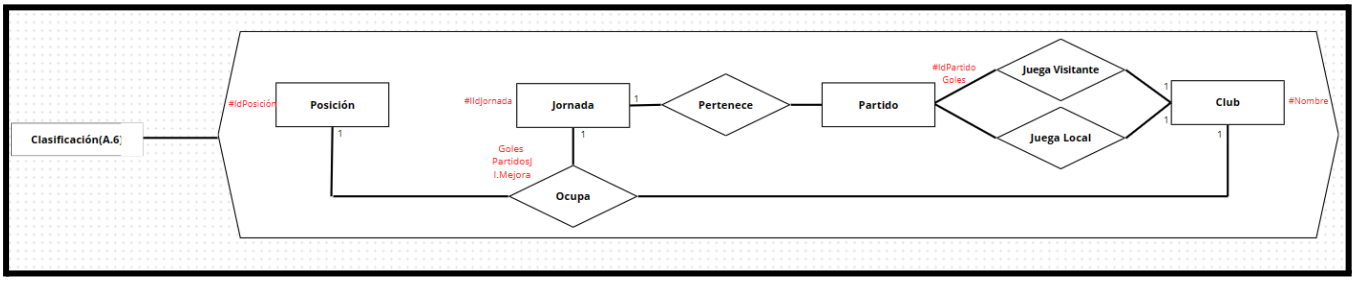


Figura 9.7: Esquema Externo Almacén Clasificación DFD Armazón

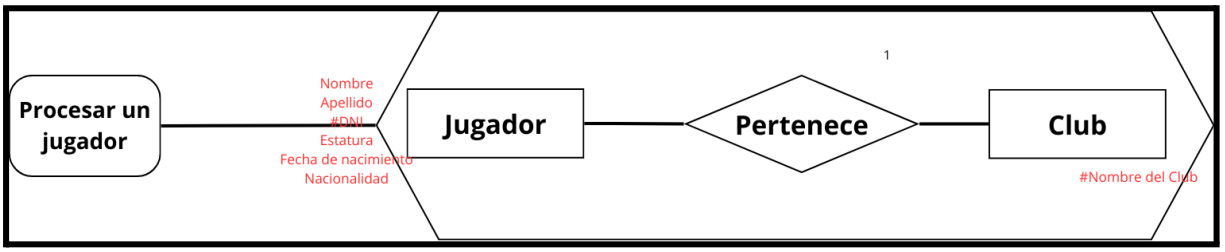


Figura 9.8: Esquema Externo Procesar un Jugador DFD Armazón

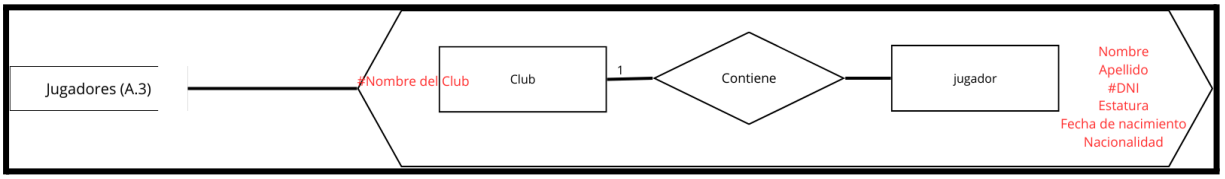


Figura 9.9: Esquema Externo almacén Jugador DFD Armazón

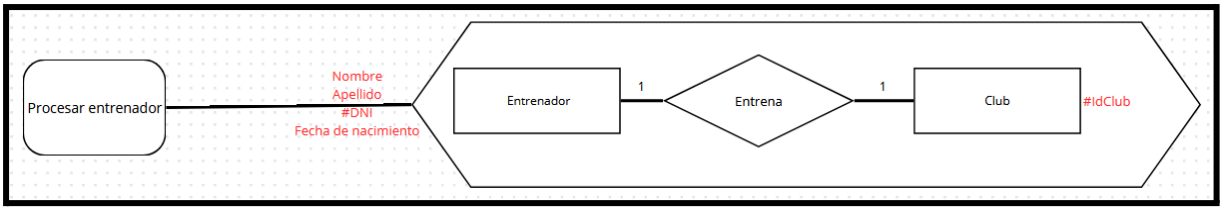


Figura 9.10: Esquema Externo Procesar un Entrenador DFD Armazón

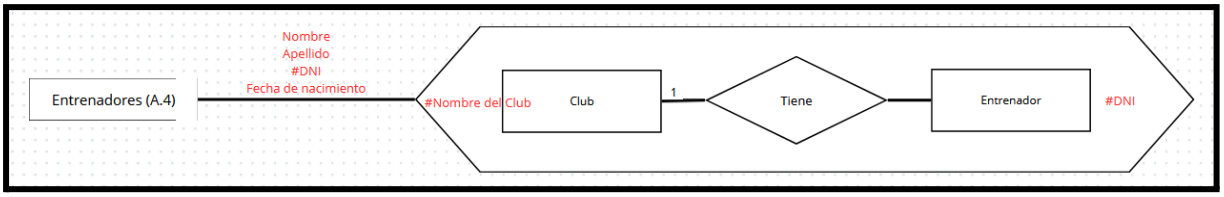


Figura 9.11: Esquema Externo almacén Entrenador DFD Armazón

10. Esquema conceptual DFD Armazón

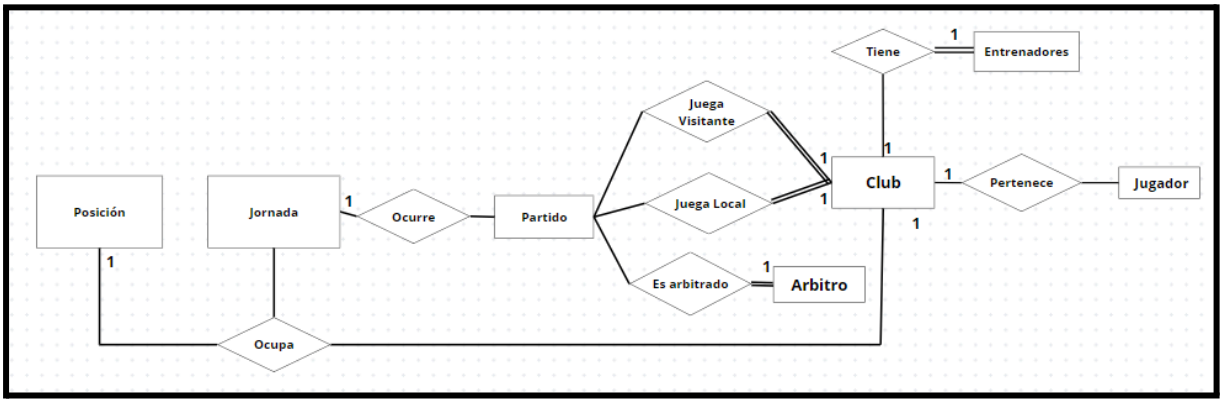


Figura 10: Esquema conceptual DFD Armazón

11. Paso a tablas del modelo conceptual

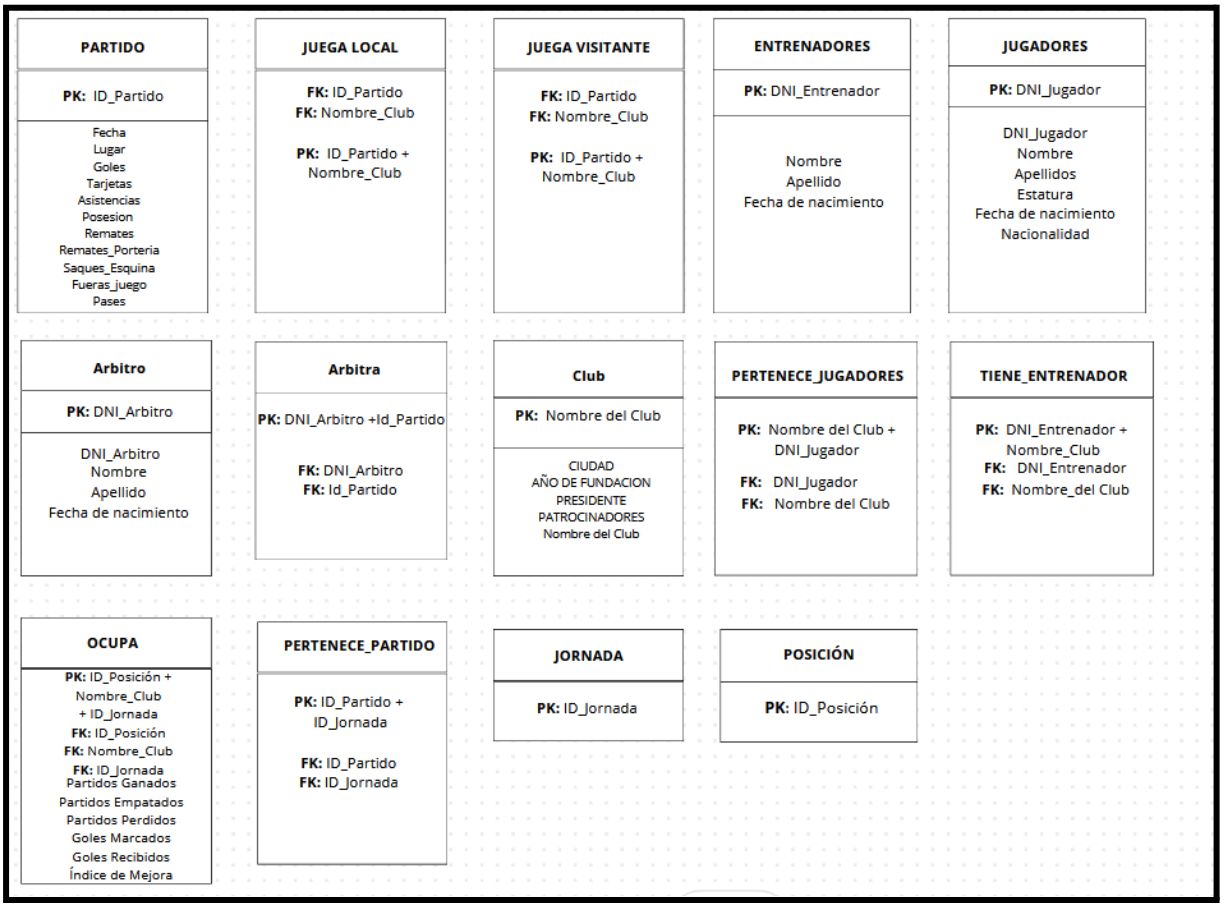


Figura 11: Tablas del modelo conceptual

12. Dependencias funcionales y normalización

12.1. Subsistema Clubes

En el siguiente subsistema lo que vamos a hacer es realizar el análisis sobre nuestras 3 tablas (Club, Pertenece_jugadores, Tiene_Entrenador). En las 2 últimas tablas de nuestro subsistema de clubes cumplimos con la primera forma normal de Boyce-Codd ya que tenemos nuestra llave primaria (PK) y no hay redundancia ni celdas con valores múltiples. Por tanto ya se encuentran normalizadas de por sí.

Dentro de la tabla de "Club" se puede comprobar que esta también está en la primera forma normal por que son atómicos todos los atributos. Además cumple con la segunda forma normal ya que todos los atributos de nuestro club tienen dependencia funcional de la llave primaria (Nombre del Club, PK). Esto quiere decir que, los patrocinadores, el año de fundación, presidente, etc dependen todos de cual sea el nombre del club, ya que para cada club estos atributos pueden tomar valores distintos. Por último, también vemos que se cumple la tercera forma normal ya que ninguno de los atributos dependen funcionalmente de otros atributos distintos de la llave primaria. Ni los entrenadores dependen de los patrocinadores, ni del año de fundación…etc.

Cumple por tanto con la normalización de Boyce-Codd (nuestros determinantes son atributos claves PK).

12.2. Subsistema Partidos

En este subsistema tenemos que analizar tres tablas (partidos, juega local y juega visitante). Nos damos cuenta que en la tabla de partidos, hay atributos que van a contener más de un dato, por tanto no estamos cumpliendo una de las condiciones de la primera forma normal. Para ello, separamos los atributos en dos. Como sabemos que siempre va a existir esta dupla (es decir, un equipo va a tener un número de pases y el otro equipo otro número), estaríamos cumpliendo de esta forma la primera forma normal.

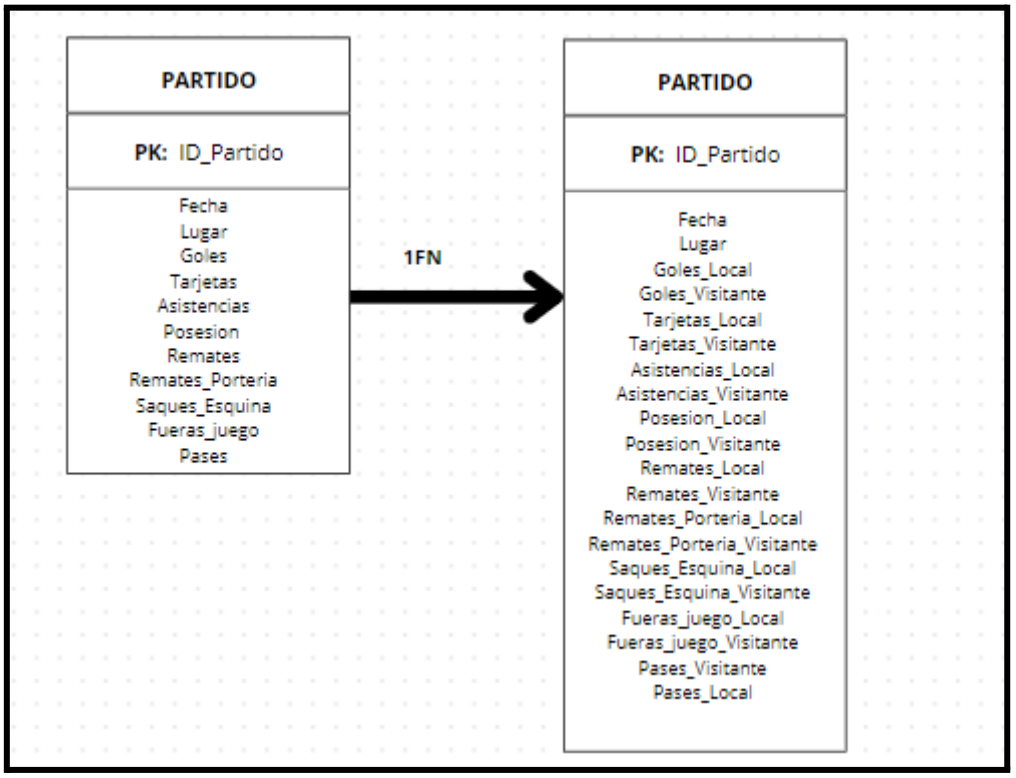


Figura 12.2.1: Primera forma normal

Las otras dos tablas del subsistema cumplen con la primera forma normal (tenemos nuestra clave primaria, no hay redundancia, no hay celdas con múltiples datos). Quedaría así:

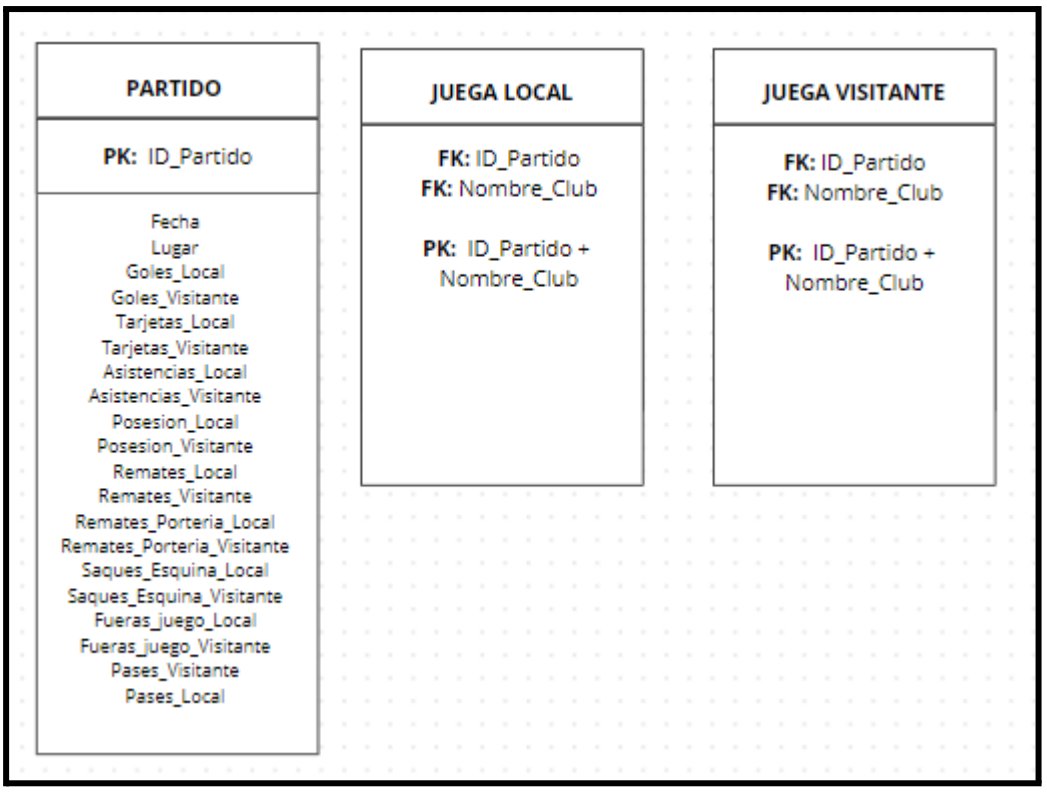


Figura 12.2.2: Tablas subsistema Partido normalizadas

Ahora pasamos a analizar las dependencias funcionales. En el caso de la tabla partido, todos los atributos son dependientes de ID_Partido. Las dependencias funcionales serían ID_Partido → Fecha, ID_Partido → Lugar … etc. De esta forma hemos comprobado que nuestra tabla ya se encuentra en la segunda forma normal. En el caso de las otras dos tablas, los atributos (que son las claves externas) también depende funcionalmente de la clave primaria. No encontramos ningún atributo que depende parcialmente de esta, por tanto también cumple la segunda forma normal. En cuanto a la tercera forma normal, para cumplirla, debemos eliminar todas las dependencias transitivas que podamos encontrar, es decir, no puede haber atributos que dependan funcionalmente de otros que no sean la clave. En nuestro caso no hay dependencias de este tipo, por tanto ya estamos en la tercera forma normal. Finalmente vamos a comprobar que se cumpla la forma normal de Boyce-Codd. Para ello, todos los determinantes deben ser atributos claves o triviales con la clave. En la tabla partido no vamos a tener ningún problema con esto pues la tabla es muy simple y la clave está compuesta por un solo atributo. Todas las dependencias funcionales tienen como determinante a ID_Partido y ID_Partido con sí mismo es una dependencia trivial. Si nos fijamos en las otras dos tablas tenemos estas dependencias:

Determinante	**Dependiente**
ID_Partido, Nombre_Club	ID_Partido
ID_Partido, Nombre_Club	Nombre_Club

Vemos que las dependencias funcionales son una dependencia funcional trivial de la clave. Por tanto, también se cumple la forma normal de Boyce-Codd.

12.3. Subsistemas Jugadores

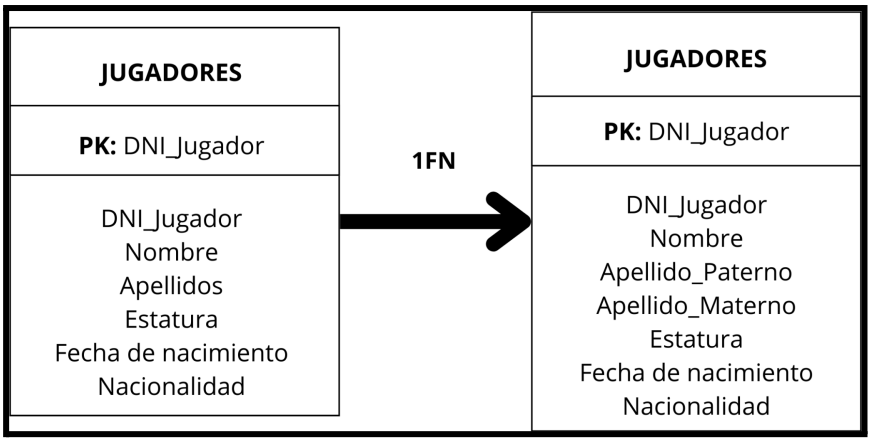


Figura 12.3: Esquema conceptual DFD Armazón

En este caso, las dependencias funcionales que observamos en la tabla son las siguientes:

- DNI_Jugador → Nombre
- DNI_Jugador → Apellido_Paterno
- DNI_Jugador → Apellido_Materno
- DNI_Jugador → Estatura
- DNI_Jugador → Fecha de nacimiento
- DNI_Jugador → Nacionalidad

El proceso de normalización ha sido el siguiente: para pasar a primera forma normal (1FN), hemos desdoblado el campo Apellidos en dos campos, Apellido_Paterno y Apellido_Materno para así cumplir con la atomicidad. En este punto ya hemos obtenido una tabla en Forma Normal de Boyce-Codd. Notamos que la tabla se encuentra en 2FN pues todos sus atributos no primos dependen de forma completa de las claves candidatas, ya que solo tenemos una sola clave candidata, el DNI. También, notamos que la tabla está en 3FN pues no presenta ninguna dependencia transitiva problemática. Por último, vemos que se encuentra en FNBC pues todo determinante es una clave candidata, en nuestro caso se cumple pues nuestra tabla está en 3FN y tiene una única clave candidata.

12.4. Subsistema Entrenadores

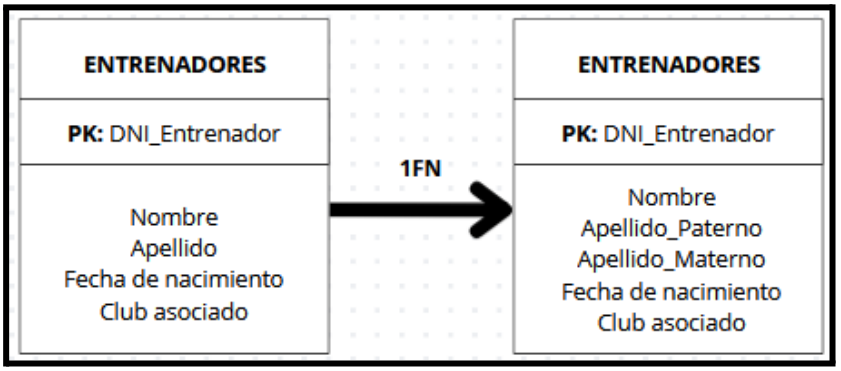


Figura 12.4: Esquema conceptual DFD Armazón

En este subsistema hay que analizar una tabla: Entrenadores.
Para realizar el proceso de Normalización, primero debemos de ver si cumple la primera forma normal (1FN). En este caso contiene una dependencia funcional del atributo 'Apellido', la cual hemos solucionado creando Apellido_Paterno y Apellido_Materno, y así cumple con la atomicidad.
Para la segunda forma normal (2FN), se requiere que la tabla esté en 1NF y que no tenga dependencias parciales, es decir, que todos los atributos no primos dependan de toda la clave candidata. En este caso, como la clave candidata (el DNI del Entrenador) no es compuesta, la tabla ya cumple con este requisito y, por lo tanto, está en 2NF.

Por último, se cumple la tercera forma normal (3NF) ya que está en 2NF y no tiene dependencias transitivas; esto significa que los atributos no primos deben depender solo de la clave candidata, y no debe haber dependencias entre los atributos no primos.

Así, se encuentra en la Forma normal de Boyce-Codd (FNBC) ya que se cumple que la tabla está en 3FN y todo determinante es una clave candidata.

12.5. Subsistema Árbitros

En el caso del subsistema de árbitros, partimos de las siguientes tablas:

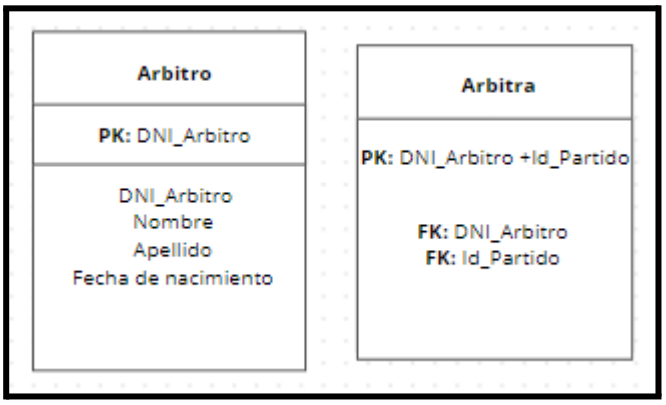


Figura 12.5: Esquema conceptual DFD Armazón

Como podemos comprobar a simple vista, existen dependencias funcionales en la tabla árbitro:

- DNI_Arbitro → Nombre
- DNI_Arbitro → Apellido_Paterno
- DNI_Arbitro → Apellido_Materno
- DNI_Arbitro → Fecha de nacimiento

Una vez eliminadas estas dependencias la tabla quedaría de la siguiente forma:

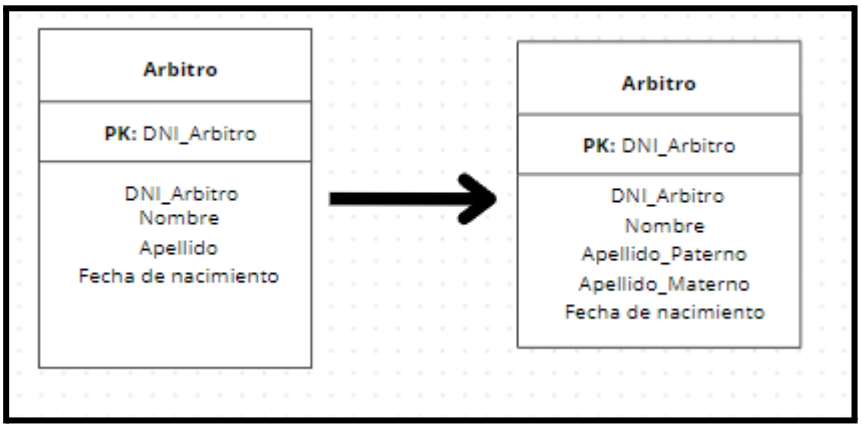


Figura 12.5.1: Esquema conceptual DFD Armazón

El proceso de normalización ha sido el siguiente: para pasar a primera forma normal (1FN), hemos tenido que desdoblar el campo Apellidos en dos campos, Apellido_Paterno y Apellido_Materno para así cumplir con la atomicidad. En este punto ya hemos obtenido una tabla en Forma Normal de Boyce-Codd. Notamos que la tabla se encuentra en 2FN pues todos sus atributos no primos dependen de forma completa de las claves candidatas, ya que solo tenemos una sola clave candidata, el DNI. También, notamos que la tabla está en 3FN pues no presenta ninguna dependencia transitiva problemática. Por último, vemos que se encuentra en FNBC pues todo determinante es una clave candidata, en nuestro caso se cumple pues nuestra tabla está en 3FN y tiene una única clave candidata.

12.6. Subsistema Clasificación

En este último subsistema tendremos que analizar cuatro tablas: ocupa, pertenece_partido, jornada y posición. Vemos que las tablas jornada y posición cumplen la forma normal de Boyce-Codd de forma trivial ya que únicamente tienen la clave primaria como atributo. Para la tabla pertenece_partido vemos que se cumple lo mismo que en la anterior tabla “juega_local” o “juega_visitante”, por lo que están en la forma normal de Boyce-Codd de igual forma.

Finalmente toca comprobar la tabla ocupa, la más importante de las 4. Vemos que está en primer formal normal ya que todos sus atributos son atómicos. También podemos comprobar que está en segunda forma normal ya que todos los atributos dependen funcionalmente de la clave primaria, los goles marcados, por ejemplo, dependen de la jornada que sea, el club y la posición, al igual que el resto de atributos. Cumple la tercera forma normal ya que no hay atributos que dependan funcionalmente de otros que no sean clave. Por último, dado que el determinante es un atributo clave, cumple la normalización Boyce-Codd.

13. Creación de tablas e inserción de tuplas

13.1. Subsistema Clubes

```
CREATE TABLE CLUB (
  Nombre_Club VARCHAR (50) PRIMARY KEY,
     Ciudad VARCHAR (50),
     Anio_Fundacion INT,
     Presidente VARCHAR (50),
     Patrocinadores VARCHAR(50)

);

CREATE TABLE PERTENECE_JUGADORES (
     Nombre_Club VARCHAR(50) NOT NULL,
     DNI_Jugador VARCHAR(9) NOT NULL,

     PRIMARY KEY (DNI_Jugador , Nombre_Club),
     FOREIGN KEY (DNI_Jugador) REFERENCES JUGADORES
(DNI_Jugador) ON DELETE CASCADE,
     FOREIGN KEY (Nombre_Club) REFERENCES CLUB
(Nombre_Club)
);

CREATE TABLE TIENE_ENTRENADOR (
     Nombre_Club VARCHAR(50) NOT NULL,
     DNI_Entrenador VARCHAR(9) NOT NULL,

     PRIMARY KEY (DNI_Entrenador, Nombre_Club ),
     FOREIGN KEY (DNI_Entrenador) REFERENCES ENTRENADORES
(DNI_Entrenador),
```

```
    FOREIGN KEY (Nombre_Club) REFERENCES CLUB
(Nombre_Club)
);
```

A continuación añadimos unas cuantas tuplas para la tabla de club:

- `INSERT INTO CLUB(Nombre_Club, Anio_Fundacion) VALUES ('FCGoodson', 2020);`
- `INSERT INTO CLUB(Nombre_Club) VALUES ('FCTiki');`
- `INSERT INTO CLUB(Nombre_Club, Anio_Fundacion, Presidente, Patrocinadores) VALUES ('UDd2jj', '2002', 'Don Pepito', 'Nike');`
- `INSERT INTO CLUB(Nombre_Club) VALUES ('FCEtsiit');`

Tuplas para la tabla de jugadores y club

- `INSERT INTO PERTENECE_JUGADORES (DNI_Jugador, Nombre_Club) VALUES ('12345678A', 'FCGoodson');`
- `INSERT INTO PERTENECE_JUGADORES (DNI_Jugador, Nombre_Club) VALUES ('98745632B', 'FCTiki');`
- `INSERT INTO PERTENECE_JUGADORES (DNI_Jugador, Nombre_Club) VALUES ('01234567C', 'UDd2jj');`

Tuplas para la tabla de club y entrenadores

- `INSERT INTO TIENE_ENTRENADOR (DNI_Entrenador, Nombre_Club) VALUES ('11111111A', 'FCGoodson');`
- `INSERT INTO TIENE_ENTRENADOR (DNI_Entrenador, Nombre_Club) VALUES ('22222222B', 'FCTiki');`
- `INSERT INTO TIENE_ENTRENADOR (DNI_Entrenador, Nombre_Club) VALUES ('33333333C', 'UDd2jj');`

13.2. Subsistema Partidos

Para este subsistema vamos a tener que crear tres tablas.

```
CREATE TABLE PARTIDO(
    ID_Partido INT PRIMARY KEY,
    Fecha Date,
    Lugar VARCHAR(20),
```

```
    Goles_Local NUMBER(3),
    Goles_Visitante NUMBER(3),
    Tarjetas_Local NUMBER(3),
    Tarjetas_Visitante NUMBER(3),
    Asistencias_Local NUMBER(3),
    Asistencias_Visitante NUMBER(3),
    Posesion_Local NUMBER(3),
    Posesion_Visitante NUMBER(3),
    Remates_Local NUMBER(3),
    Remates_Visitante NUMBER(3),
    Remates_Porteria_Local NUMBER(3),
    Remates_Porteria_Visitante NUMBER(3),
    Saques_Esquina_Local NUMBER(3),
    Saques_Esquina_Visitante NUMBER(3),
    Fueras_Juego_Local NUMBER(3),
    Fueras_Juego_Visitante NUMBER(3),
    Pases_Local NUMBER(4),
    Pases_Visitante NUMBER(4)
);

CREATE TABLE JUEGA_LOCAL (
  ID_Partido INT NOT NULL,
  Nombre_Club VARCHAR(50) NOT NULL,

  PRIMARY KEY (ID_Partido , Nombre_Club ),FOREIGN KEY
(ID_Partido ) REFERENCES PARTIDO (ID_Partido),FOREIGN KEY
(Nombre_Club ) REFERENCES CLUB  (Nombre_Club )
);

CREATE TABLE JUEGA_VISITANTE (
  ID_Partido INT NOT NULL,
  Nombre_Club VARCHAR(50) NOT NULL,

  PRIMARY KEY (ID_Partido , Nombre_Club ),
  FOREIGN KEY (ID_Partido ) REFERENCES PARTIDO
(ID_Partido),
  FOREIGN KEY (Nombre_Club ) REFERENCES CLUB
(Nombre_Club )
```

);

*CAMBIOS: Con respecto a la práctica 2. Las nuevas sentencias de creación de tablas de este subsistema son las siguientes.

```
CREATE TABLE Juega_Local(ID_Partido VARCHAR(20) NOT NULL,
Nombre_Club VARCHAR(50) NOT NULL, golesLocal NUMBER(3) ,
tarjetasLocal NUMBER(3), asistenciasLocal NUMBER(3),
posesionLocal NUMBER(3), rematesLocal NUMBER(3),
rematesPorteriaLocal NUMBER(3) , saquesEsquinaLocal
NUMBER(3), fuerasJuegoLocal NUMBER(3), pasesLocal
NUMBER(4))"
                    + " ,PRIMARY KEY (ID_Partido ,
Nombre_Club ),FOREIGN KEY (ID_Partido ) REFERENCES
PARTIDO (ID_Partido) ON DELETE CASCADE, FOREIGN KEY
(Nombre_Club ) REFERENCES CLUB  (Nombre_Club ) ON DELETE
CASCADE)

CREATE TABLE Juega_Visitante(ID_Partido VARCHAR(20) NOT
NULL, Nombre_Club VARCHAR(50) NOT NULL, golesVisitante
NUMBER(3) , tarjetasVisitante NUMBER(3),
asistenciasVisitante NUMBER(3), posesionVisitante
NUMBER(3), rematesVisitante NUMBER(3),
rematesPorteriaVisitante NUMBER(3) ,
saquesEsquinaVisitante NUMBER(3), fuerasJuegoVisitante
NUMBER(3), pasesVisitante NUMBER(4))"
                    + " PRIMARY KEY (ID_Partido ,
Nombre_Club ),FOREIGN KEY (ID_Partido ) REFERENCES
PARTIDO (ID_Partido) ON DELETE CASCADE, FOREIGN KEY
(Nombre_Club ) REFERENCES CLUB  (Nombre_Club ) ON DELETE
CASCADE)

CREATE TABLE Partido(ID_Partido VARCHAR(20) PRIMARY KEY
NOT NULL, Lugar VARCHAR(20), Fecha DATE)
```

13.3. Subsistemas Jugadores

En este caso, sólo tenemos que crear una tabla, que lo hacemos con la siguiente sentencia SQL:

```
CREATE TABLE JUGADORES(
     DNI_Jugador VARCHAR2(9) PRIMARY KEY,
     Nombre VARCHAR2(20) NOT NULL,
     Apellido_Paterno VARCHAR(20) NOT NULL,
     Apellido_Materno VARCHAR(20),
     Estatura NUMBER(3),
     Fecha_nacimiento Date NOT NULL,
     Nacionalidad VARCHAR2(10)
);
```

E insertamos 3 tuplas:

- `INSERT INTO JUGADORES (DNI_Jugador, Nombre, Apellido_Paterno) VALUES ('12345678A', 'Manuel', 'Diaz-Meco');`
- `INSERT INTO JUGADORES (DNI_Jugador, Nombre, Apellido_Paterno, Apellido_materno) VALUES ('98745632B', 'Javier', 'Gómez', 'López');`
- `INSERT INTO JUGADORES (DNI_Jugador, Nombre, Apellido_Paterno, Apellido_materno, Estatura, Nacionalidad) VALUES ('01234567C', 'Carlos', 'Mata', 'Carrillo', 177, 'Español');`

13.4. Subsistema de Entrenadores

Para este subsistema como ya hemos creado la tabla que relaciona los entrenadores con el club, solo vamos a tener que crear la tabla de entrenadores.

```
CREATE TABLE ENTRENADORES(
     DNI_Entrenador VARCHAR2(9) PRIMARY KEY,
     Nombre VARCHAR2(20) NOT NULL,
     Apellido_Paterno VARCHAR(20) NOT NULL,
     Apellido_Materno VARCHAR(20),
     Fecha_nacimiento Date
);
```

Ahora insertamos unas cuantas tuplas para la tabla:

- INSERT INTO Entrenadores(DNI_Entrenador, Nombre, Apellido_Paterno, Apellido_Materno, Fecha_nacimiento) VALUES ('11111111A', 'Pep', 'Guardiola', 'Sala', TO_DATE('1971-18-01', 'YYYY-DD-MM'));
- INSERT INTO Entrenadores(DNI_Entrenador, Nombre, Apellido_Paterno, Apellido_Materno, Fecha_nacimiento) VALUES ('22222222B', 'Jesús', 'Reyes', 'DeToro', TO_DATE('2002-12-02','YYYY-DD-MM'));
- INSERT INTO Entrenadores(DNI_Entrenador, Nombre, Apellido_Paterno, Apellido_Materno, Fecha_nacimiento) VALUES ('33333333C', 'Valeria', 'Borrajo', 'Yusty', TO_DATE('2002-17-03','YYYY-DD-MM'));

13.5. Subsistema Árbitros

En este caso, tenemos que crear la tabla Arbitro y la tabla Arbitra y la de Colegio de Arbitros, que lo hacemos con la siguientes sentencias SQL:

```
CREATE TABLE Arbitro(
     DNI_Arbitro VARCHAR2(9) PRIMARY KEY,
     Nombre VARCHAR2(20) NOT NULL,
     Apellido_Paterno VARCHAR2(20) NOT NULL,
     Apellido_Materno VARCHAR2(20),
     Fecha_nacimiento Date NOT NULL
);

CREATE TABLE Arbitra(
     ID_Partido INT NOT NULL,
     DNI_Arbitro VARCHAR2(9) NOT NULL,

     PRIMARY KEY (DNI_Arbitro , ID_Partido ),
     FOREIGN KEY (ID_Partido ) REFERENCES PARTIDO
     (ID_Partido),
     FOREIGN KEY (DNI_Arbitro ) REFERENCES Arbitro
     (DNI_Arbitro )
);
```

Ahora procedemos a insertar tuplas en la tabla Arbitro:

- ```
  INSERT INTO Arbitro (DNI_Arbitro, Nombre,
  Apellido_Paterno, Apellido_Materno,
  Fecha_nacimiento) VALUES ('23456789H', 'Mateu',
  'Lahoz', 'Diaz', TO_DATE('10-09-1968',
  'DD-MM-YYYY'));
  ```

- ```
  INSERT INTO Arbitro (DNI_Arbitro, Nombre,
  Apellido_Paterno, Apellido_Materno,
  Fecha_nacimiento) VALUES ('27136617L', 'Alberto',
  'Munuera', 'Montero', TO_DATE('06-05-1979',
  'DD-MM-YYYY'));
  ```
- ```
  INSERT INTO Arbitro (DNI_Arbitro, Nombre,
  Apellido_Paterno, Apellido_Materno,
  Fecha_nacimiento) VALUES ('55636689B', 'Juan',
  'Iturralde', 'Gonzalez',
  TO_DATE('11-01-1988','DD-MM-YYYY'));
  ```

Inserción de tuplas de arbitra:

- ```
  INSERT INTO Arbitra (DNI_Arbitro,ID_Partido) VALUES
  ('55636689B', '0000');
  ```
- ```
  INSERT INTO Arbitra (DNI_Arbitro,ID_Partido) VALUES
  ('27136617L', '0001');
  ```
- ```
  INSERT INTO Arbitra (DNI_Arbitro,ID_Partido) VALUES
  ('23456789H', '0002');
  ```

13.6. Subsistema Clasificación

Creamos las tablas necesarias:

```
CREATE TABLE JORNADA (
     ID_Jornada INT PRIMARY KEY);
CREATE TABLE POSICIÓN (
     ID_Posición INT PRIMARY KEY);
CREATE TABLE PERTENECE_PARTIDO (
     ID_Jornada INT NOT NULL,
     ID_Posición INT NOT NULL,
```

```
     PRIMARY KEY (ID_Jornada, ID_Posición),
     FOREIGN KEY (ID_Jornada) REFERENCES JORNADA
    (ID_Jornada),
     FOREIGN KEY (ID_Posición) REFERENCES POSICIÓN
    (ID_Posición)
    );
```

CREATE TABLE OCUPA (

```
            ID_Posicion INT NOT NULL,
            Nombre_Club VARCHAR(50) NOT NULL,
            ID_Jornada INT NOT NULL,
            Partidos_Ganados INT,
            Partidos_Empatados INT,
            Partidos_Perdidos INT,
        Goles_Marcados INT,
        Goles_Recibidos INT,
        Indice_Mejora VARCHAR(1),

        PRIMARY KEY (ID_Posicion, Nombre_Club,
    ID_Jornada),
        FOREIGN KEY (ID_Posicion) REFERENCES POSICIÓN
    (ID_Posición),
        FOREIGN KEY (Nombre_Club ) REFERENCES CLUB
    (Nombre_Club),
        FOREIGN KEY (ID_Jornada) REFERENCES JORNADA
    (ID_Jornada)
    );
```

Rellenamos las tablas, la tabla ocupa no podemos hacerlo manualmente porque se hace a partir de los cálculos hechos con los datos de los partidos de la jornada correspondiente.

- `INSERT INTO JORNADA (ID_Jornada) VALUES('1');`
- `INSERT INTO JORNADA (ID_Jornada) VALUES('2');`
- `INSERT INTO JORNADA (ID_Jornada) VALUES('3');`
- `INSERT INTO POSICIÓN (ID_Posición) VALUES ('1');`
- `INSERT INTO POSICIÓN (ID_Posición) VALUES ('2');`
- `INSERT INTO POSICIÓN (ID_Posición) VALUES ('3');`
- `INSERT INTO PERTENECE_PARTIDO (ID_Jornada, ID_Partido) VALUES ('1', '0000');`

- `INSERT INTO PERTENECE_PARTIDO (ID_Jornada, ID_Partido) VALUES ('1', '0001');`
- `INSERT INTO PERTENECE_PARTIDO (ID_Jornada, ID_Partido) VALUES ('1', '0002');`

14. Transacciones lógicas

14.1. Subsistema de clubes

Las transacciones del subsistema de clubes implican diversas operaciones en la base de datos, y su lógica se puede describir de la siguiente manera. La primera transacción se lleva a cabo durante la inicialización del sistema. En el método crearTablas(), se ejecutan varias sentencias SQL para crear las tablas CLUB, PERTENECE_JUGADORES, y TIENE_ENTRENADOR. Además, se establece un trigger (before_delete_club) asociado a la eliminación de un club. Una vez completadas estas sentencias, la transacción se da por finalizada.
La transacción de alta de un nuevo club se realiza en el método altaNuevoClub(). El usuario proporciona datos básicos del club como nombre, ciudad, año de fundación, presidente y patrocinadores. Se verifica si el club ya existe antes de realizar la inserción en la tabla CLUB. En caso de existir, se deshace la transacción y se muestra un mensaje de error.

La transacción de baja de un club se encuentra en el método bajaClub(). El usuario ingresa el nombre del club a eliminar, se verifica su existencia, y se procede a borrarlo de la tabla CLUB. La información del club eliminado se muestra al usuario.

La transacción para modificar datos de un club está presente en el método modificarClub(). El usuario puede elegir entre modificar patrocinadores, ciudad, año de fundación o presidente del club. Se utilizan savepoints para gestionar posibles fallos en la modificación, y se ofrece la opción de deshacer cambios.

La transacción de asociación de un club a un jugador se realiza en el método asociarClubAJugador(). El usuario ingresa el nombre del club y el DNI del jugador, y se verifica la existencia del club antes de realizar la inserción en la tabla PERTENECE_JUGADORES.

La transacción de asociación de un club a un entrenador se lleva a cabo en asociarClubAEntrenador(). Similar a la asociación con jugadores, se verifica la existencia del club antes de realizar la inserción en la tabla TIENE_ENTRENADOR.

El método mostrarListadoClubes() realiza la transacción de mostrar un listado detallado de todos los clubes en el sistema. Muestra información como nombre, ciudad, año de fundación, presidente, patrocinadores, jugadores y entrenadores asociados.

La transacción principal del subsistema se encuentra en menuPrincipalClubes(), donde se presenta un menú que permite al usuario realizar diversas operaciones, gestionando internamente las transacciones asociadas a cada opción.

Por último, cabe mencionar también que antes de realizar operaciones críticas, se establece un savepoint al inicio de la transacción. Por ejemplo, al dar de alta un nuevo club o al modificar los datos de un club. También durante la transacción, se pueden establecer savepoints adicionales para puntos específicos donde se quiera realizar una operación crítica, como hemos hecho en modificar un club, donde era algo más atareado. Por último, si ocurre algún error durante la transacción, se ha ejecutado un rollback en las transacciones para deshacer todas las modificaciones realizadas desde el último savepoint.

14.2. Subsistema de partidos

Las transacciones del subsistema partidos son un poco más complejas debido al orden que hay que tener en cuenta a la hora de crear las tablas y las relaciones que hay entre ellas. Las primeras transacciones son las de creación de las tablas. Esto hay que hacerlo en dos métodos diferentes. Primero para crear la tabla de *partido*. El segundo para crear las tablas que dependen de esta y de las de otros subsistemas. Este es el caso de *juega local*, *visitante* y *arbitra*.

La siguiente transacción consiste en dar de alta un nuevo partido. En este caso no tenemos en cuenta los datos estadísticos sino que solamente se añaden los datos básicos de un partido como son el id (generado automáticamente), el lugar, la fecha, los equipos y el árbitro. Todos estos datos menos el id del partido deben ser proporcionados por el usuario que quiera insertar el partido en la base de datos. Al dar de alta un nuevo partido estamos añadiendo una nueva entrada en 4 tablas diferentes: *partido, arbitra, juega_local y juega_visitante.* Durante todo este proceso, las modificaciones que se realizan en la base de datos forman parte de la transacción y por tanto no se reflejan en el sistema. También se ofrece la posibilidad de deshacer todas esas modificaciones, haciendo un rollback al punto de inicio y finalizando la transacción

Otra transacción que cabe la pena destacar es la de modificar los datos de un partido. Esta funcionalidad se añade para poder insertar los datos estadísticos del partido en cuestión. El usuario deberá proporcionar el identificador del partido para posteriormente aportar los datos.

El resto de transacciones son las de mostrar datos, ya sean de un partido solamente o de todos los partidos que hay en el sistema.

14.3. Subsistema de jugadores

La primera transacción la identificamos en la inicialización del sistema. Dicha transacción conlleva dos sentencias SQL, la primera para crear la tabla de Jugadores y la segunda para crear el disparador asociado a la mayoría de edad de un jugador. Una vez realizadas estas dos sentencias, podemos dar la transacción por finalizada.

La siguiente transacción a destacar es la modificación de los datos del jugador. Esta transacción se inicia en el momento en el que el usuario entra al menú de modificación de datos, y no finaliza hasta que el mismo usuario indica que desea guardar los cambios realizados en el jugador. Durante todo este proceso, las modificaciones que se realizan en la base de datos forman parte de la transacción y por tanto no se reflejan en el sistema. También se ofrece la posibilidad de deshacer todas esas modificaciones, haciendo un rollback al punto de inicio y finalizando la transacción.

El resto de operaciones que se realizan en el subsistema sólo implican una sentencia SQL, y por tanto son transacciones atómicas y elementales que no merecen una explicación detallada más allá de la propia funcionalidad que realizan, como dar de alta a un jugador, borrarlo del sistema o mostrar listados de jugadores.

14.4. Subsistema de entrenadores

En el subsistema 'Entrenadores' podemos encontrar en primer lugar la transacción para Crear y Eliminar Tablas:
‘crearTablaEntrenadores’ incluye la creación de la tabla Entrenadores en la base de datos. Si ésta falla, la transacción se revierte para mantener la consistencia de la base de datos. ‘borrarTablasEntrenadores’ y ‘borrarTablaEntrenador’ eliminan todo el contenido de la tabla Entrenadores y luego la tabla misma.

En segundo lugar identificamos la transacción de Alta y Baja de Entrenadores:
‘EntrenadorAlta’ incluye la inserción de los detalles del entrenador en la base de datos, y ‘EntrenadorBaja’ elimina un entrenador específico.
Ambos utilizan ‘commit’ para asegurar que los cambios se reflejen en la base de datos solo si todos los pasos se completan con éxito.

Por último, podemos observar la transacción Modificación de Datos de un Entrenador, ‘Entrenadoresmodificar’, que comienza cuando el usuario elige modificar los datos de un entrenador y termina cuando decide guardar o descartar los cambios. Durante la transacción, los cambios no se reflejan en la base de datos hasta que se ejecuta un ‘commit’. Si el usuario elige no guardar los cambios, se utiliza rollback para descartar todas las modificaciones desde el inicio de la transacción.

Las demás operaciones, como listar entrenadores o mostrar el club asociado a un entrenador, son atómicas y no requieren un manejo de transacciones especial.

14.5. Subsistema de árbitros

La primera transacción que he identificado dentro de mi subsistema ha sido la de creación de la tabla Árbitro, que contiene 2 sentencias SQl, la primera de ellas para la creación de la tabla y la segunda es el disparador para controlar que la edad del árbitro introducida por el usuario en ningún momento es inferior a 18 años. Cuando se realizan estas dos sentencias, se da por finalizada la transacción.

La otra transacción destacable la encontramos en la función de modificar datos de un árbitro, la cual se inicia en el momento en el que el usuario selecciona desde el menú de subsistema la opción de modificar los datos de un árbitro registrado y finaliza en el momento en el que el usuario decide salir de la opción de modificar los datos, guardándose en dicho momento los cambios realizados hasta entonces. Los cambios no se guardan hasta este momento ya que forman parte de la misma transacción, por lo que no se reflejan en la base de datos hasta el final de esta. Además, se ofrece la opción al usuario de salir sin guardar los cambios, haciendo uso de un rollback, que nos llevará al inicio de la transacción y la finalizará.

Los demás métodos no contienen transacciones reseñables, ya que la mayoría de ellos solo están compuestos por sentencias SQL atómicas, tales como dar de alta un nuevo árbitro, dar de baja un árbitro existente o mostrar los partidos asignados a un árbitro.

14.6. Subsistema de clasificación

Las transacciones del sistema clasificación tienen un orden de complejidad similar a las del subsistema partidos, ya que hay que tener en cuenta el orden de creación de las tablas para el correcto funcionamiento del sistema. En primer lugar creamos la tabla jornada con su respectiva clave primaria ID_Jornada, esta tabla nos permite poder crear las siguientes tres tablas, claves en el desarrollo del subsistema. La creación de ‘Posición’, ‘Ocurre’ y ‘Ocupa’ involucran elementos del subsistema partidos y el de clubes, es muy importante por este motivo que estas tablas sean las últimas en crearse de toda la base de datos ya que todos los subsistemas están involucrados en estas tablas.

La transacción del la función para obtener la máxima jornada es una única sentencia SQL que no tiene mayor relevancia y que es necesaria para el método para actualizar la clasificación.

La siguiente transacción relevante es la creación del disparador necesario para el método de asignar un partido a una jornada, este trigger verifica que el id de la jornada es positivo y aplica esta restricción a la tabla ocurre y a la tabla jornada, las cuales son accedidas en el método para asignar la jornada. En este método se tienen como argumentos el id del partido que se va a asignar al id de la jornada que se quiera, las transacciones que hay son las que actualizan las tablas jornada primero y después ocurre con estos datos.

A continuación tenemos el método para eliminar una jornada de la tabla jornada, en el cual se pasa como argumento el id de la jornada que se quiere eliminar que será usado en la sentencia SQL, borrando en cascada el resto de tuplas enlazadas con la jornada correspondiente

(gracias a la definición que se ha hecho de las tablas ocurre, ocupa y posición) completando así la transacción.

Tenemos otro trigger, este es muy parecido al anterior, salvo que también comprueba que el id de la jornada que se quiere modificar ya está en la base de datos. Como es lógico, este disparador está enlazado con el método de modificar jornada, el cual tiene como parámetro el id de la jornada a modificar y posteriormente se pide al usuario que ingrese el partido que quiere introducir en la jornada, eliminando así el anterior valor que hubiera. Esta transacción consta únicamente de una sentencia update de SQL.

Para actualizar la clasificación se requerirían ciertas transacciones para el correcto cálculo de las posiciones de cada club en cada jornada en función de los goles anotados y recibidos por partido, recogiendo los datos de este subsistema. Para mostrar la clasificación hemos de ejecutar una sentencia SQL mostrando así las tuplas de la tabla posición que cumplen la condición de que la jornada de esas tuplas es la última registrada en la base de datos.

Destacar que no se han usado rollback's o savepoint's como en otros subsistemas porque en el único método en el que se requeriría un uso de estos sería en el de modificar la jornada (siguiendo la tónica de los otros subsistemas en los que se hace uso de estos elementos para verificar las transacciones), sin embargo, dado que se modifica un único dato no merece la pena el uso de estos sofisticados procedimientos.

15. Disparadores

15.1. Disparadores del subsistema de clubes

El disparador "before_delete_club" se establece en el subsistema de clubes y está asociado a la eliminación de un club. Este trigger se ejecuta antes de que se elimine un registro de la tabla CLUB. Su función principal es verificar si hay jugadores asociados al club que se está intentando eliminar. En caso de que los haya, no se borrara el Club. Se encuentra en el método de crearTablas().

```
DELIMITER //
    CREATE TRIGGER before_delete_club
    BEFORE DELETE ON CLUB
    FOR EACH ROW
    BEGIN
        DECLARE numJugadores INT
        SELECT COUNT(*) INTO numJugadores
        FROM PERTENECE_JUGADORES
        WHERE Nombre_Club = OLD.Nombre_Club;
        IF numJugadores > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el club. Tiene
jugadores asociados.';
```

```
            END IF;
        END //
    DELIMITER ;
```

15.2. Disparadores del subsistema de partidos

El disparador para el sistema de partidos comprueba que cuando insertamos la posesión de balón en las tablas Juega_Local y Juega_Visitante, la suma de ambas debe ser estrictamente igual a 100. Como la inserción desde el menú principal del subsistema se hace al mismo tiempo para local y visitante y no se hace el commit hasta que se tienen los datos de ambos equipos, siempre que hayamos insertado la posesión de un equipo se habrá insertado la del otro.

```
create or replace TRIGGER verificar_posesion
            AFTER INSERT ON Juega_Local
            FOR EACH ROW
                 DECLARE
                 total_posesiones INT;
                 pos_local INT;
                 pos_vis INT;
                 BEGIN

                 SELECT PosesionLocal INTO pos_local
                 FROM Juega_Local
                 WHERE ID_Partido = :NEW.ID_Partido;

                 -- Obtener la posesión visitante
                 SELECT PosesionVisitante INTO pos_vis
                 FROM Juega_Visitante
                 WHERE ID_Partido = :NEW.ID_Partido;

                 --Calculamos el total de la posesion
                 total_posesiones := pos_local + pos_vis;

                 -- Verificar que la suma es igual a 100
                 IF total_posesiones <> 100 THEN
                      RAISE_APPLICATION_ERROR(-20001, 'La
suma de las posesiones de balón no es igual a 100');
                 END IF;
            END;
```

15.3. Disparadores del subsistema de jugadores

El disparador de este subsistema evita que se guarde en base de datos a una instancia de jugador, cuya fecha de nacimiento indique que el jugador es menor de edad. El código es el siguiente:

```
CREATE OR REPLACE TRIGGER validar_edad_jugador
BEFORE INSERT ON Jugadores
FOR EACH ROW
DECLARE fecha_nacimiento_valida BOOLEAN;
BEGIN
     fecha_nacimiento_valida := (SYSDATE -
     NEW.fecha_nacimiento) >= 6570;
IF NOT fecha_nacimiento_valida THEN
RAISE_APPLICATION_ERROR(-20001, 'El jugador debe de
ser mayor de edad!');
END IF;
END
```

15.4. Disparador del subsistema de entrenadores

Este disparador asegura una de las restricciones semánticas del subsistema. Esta es que un entrenador no pueda ser menor de edad. Por tanto, cuando se introduce su fecha de nacimiento se calcula la diferencia de meses que hay con la fecha actual y se divide entre 12 para calcular los años.

```
CREATE OR REPLACE TRIGGER verificar_edad_entrenador
BEFORE INSERT OR UPDATE ON Entrenadores
FOR EACH ROW
DECLARE
     v_edad NUMBER;
BEGIN

     v_edad := TRUNC(MONTHS_BETWEEN(SYSDATE,
:NEW.Fecha_nacimiento) / 12);

     IF v_edad < 18 THEN
     RAISE_APPLICATION_ERROR(-20001, 'El entrenador debe tener
al menos 18 años.');
     END IF;
```

```
END;
```

15.5. Disparadores del subsistema de árbitros

El disparador del subsistema se encarga de que ningún árbitro que sea menor de edad pueda ser registrado en el sistema:

```
CREATE OR REPLACE TRIGGER validar_edad_arbitro
BEFORE INSERT ON Arbitro
FOR EACH ROW
DECLARE fecha_nacimiento_valida BOOLEAN;
BEGIN
fecha_nacimiento_valida := (SYSDATE -
NEW.fecha_nacimiento) >= 6570;
IF NOT fecha_nacimiento_valida THEN
RAISE_APPLICATION_ERROR(-20001, 'El arbitro debe de ser
mayor de edad!');
END IF;
END
```

15.6. Disparadores del subsistema de clasificación

El primero de los disparadores se encarga de verificar que el id de jornada introducido no es negativo, tanto para la tabla jornada como para la tabla ocurre:

```
CREATE TRIGGER VerificarJornadaPositivaEnJornada
BEFORE INSERT ON Jornada
FOR EACH ROW
BEGIN
IF NEW.ID_Jornada <= 0 THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El ID de la
jornada debe ser positivo';
END IF;
END;

CREATE TRIGGER VerificarJornadaPositivaEnOcurre
BEFORE INSERT ON Ocurre
FOR EACH ROW
BEGIN
IF NEW.ID_Jornada <= 0 THEN
```

```
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El ID de la
jornada debe ser positivo';
END IF;
END;
```

El segundo disparador realiza prácticamente la misma funcionalidad solo que también verifica que el id de jornada introducido ya ha sido introducido en la base de datos:

```
"CREATE TRIGGER VerificarJornadaAntesDeModificar
BEFORE UPDATE ON Ocurre
FOR EACH ROW
BEGIN
IF NEW.ID_Jornada <= 0 THEN
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El ID de la
jornada debe ser positivo';
END IF;
IF NOT EXISTS (SELECT 1 FROM Jornada WHERE ID_Jornada =
NEW.ID_Jornada) THEN "
SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La jornada no
existe en la base de datos';
END IF;
END;"
```

16. Motivación para elección del software

Los motivos para elegir usar Java como lenguaje de programación son dos principalmente: el uso de Oracle BD, que es el sistema gestor de bases de datos que tenemos implementado en la escuela, y el trabajo previo ya realizado en el seminario 1.

En primer lugar, el uso de Oracle BD está motivado por la facilidad de uso que nos aporta usar un sistema ya implementado y en funcionamiento en la escuela, evitando así tener que realizar nosotros algún tipo de configuración y puesta a punto de un sistema gestor de bases de datos por nuestra cuenta. Además, el tener dicho SGDB en un servidor facilita la interacción en la misma base de datos por todos los integrantes del grupo.

En segundo lugar, todo el trabajo realizado durante el seminario 1 nos sirvió para coger confianza y habilidad en el desarrollo de un front-end sencillo, con un back-end elaborado en Java. Además, Java al ser de Oracle, tiene software propietario (JDBC) para la interacción con la base de datos, haciendo todas las operaciones de interacción con la base de datos mucho más sencillas y amigables.