Gestión de club US: Unión Sevilla

Realizado por:

-Guillermo de la Cruz Dorado

-José Manuel López Moreno

-Francisco Javier Boza Farrán

Índice

1.- Introducción

Un club de balonmano de la provincia de Sevilla, concretamente el Club Balonmano Unión Sevilla, necesita de una herramienta de gestión de ciertos aspectos para hacer más llevadera las tareas de administración del club. Dicho club nos pide facilitar a los oficiales y gestores un control sobre entrenadores, jugadores e información sobre estos (datos personales, categoría, lesiones, etc.), asistencias, partidos, equipos y competiciones, así como materiales y gastos derivados de los nombrados.

Los oficiales se encargarán de la introducción de los datos relacionados con cada partido, tales como el equipo local, equipo visitante, fecha, resultados, jugadoras, asistencias; la gestión de las pistas y material de entrenamiento, además del cobro de cuotas, pago de arbitraje y desplazamiento cuando sean necesarios.

Gracias a esta herramienta se automatizaran los procesos de control de los miembros y las tareas del club simplificando el acceso a estos y centralizándolos en una misma aplicación, consiguiéndose con una mera introducción de datos.

2.- Glosario de términos

-*Asistencia*: característica en el perfil de una jugadora que indica si se ha presentado a cierto partido o entrenamiento.

-*Categoría*: división según la edad de las jugadoras en benjamín(inferior a doce años) alevín (de doce a trece años) cadete (de catorce a quince años) juvenil (de dieciséis a diecisiete años) y sénior (mayor de dieciocho años).

-*Clasificación*: ordenación de los equipos según la puntuación obtenida de los partidos que se han disputado.

-*Club*: personal completo que forman la entidad, oficiales, jugadoras, encargados, directiva, etc.

-*Competición*: conjunto de partidos realizados en cierto orden entre varios equipos en los que se disputa una posición más alta en la clasificación.

-*Costo arbitraje*: precio que se ha de pagar al árbitro del encuentro, el cual varía según categoría, horario y distancia a la que se encuentra el lugar donde se lleva a cabo el partido respecto a la capital.

-*Cuota*: pago que deben llevar a cabo mensualmente las jugadoras para poder participar en los eventos y entrenamientos del club y jugar los partidos.

-*Equipo*: selección de las jugadoras según su categoría y características para participar en las competiciones.

-*Jugadora*: persona preparada para jugar en un equipo según sus características.

-*Liga*: subcategoría de competición en la que los participantes juegan los partidos a doble vuelta, es decir, juegan un partido de ida y uno de vuelta.

-*Oficiales*: personal del club encargado del equipo según categoría, de dirigir los partidos, organizar desplazamientos, pagar arbitraje, etc; engloba a entrenadores y delegados, los entrenadores se harán cargo de los equipos asignados a su cargo y los delegados serán los encargados de recoger actas e informes de partidos, entrenamientos, etc.

-*Posición*: lugar asignado al jugador en la pista, tales como portero, pivotes, laterales, centrales y extremos. Dicho rol depende de las cualidades físicas y técnico-tácticas del jugador.

3.- Objetivos

Teniendo en cuenta las historias de usuario y las necesidades que los distintos miembros del club de las que se nos han puesto en conocimiento, hemos elaborado los principales objetivos a cumplir por este proyecto:

- Seguimiento del cobro de las mensualidades y del pago de gastos derivados de la competición.

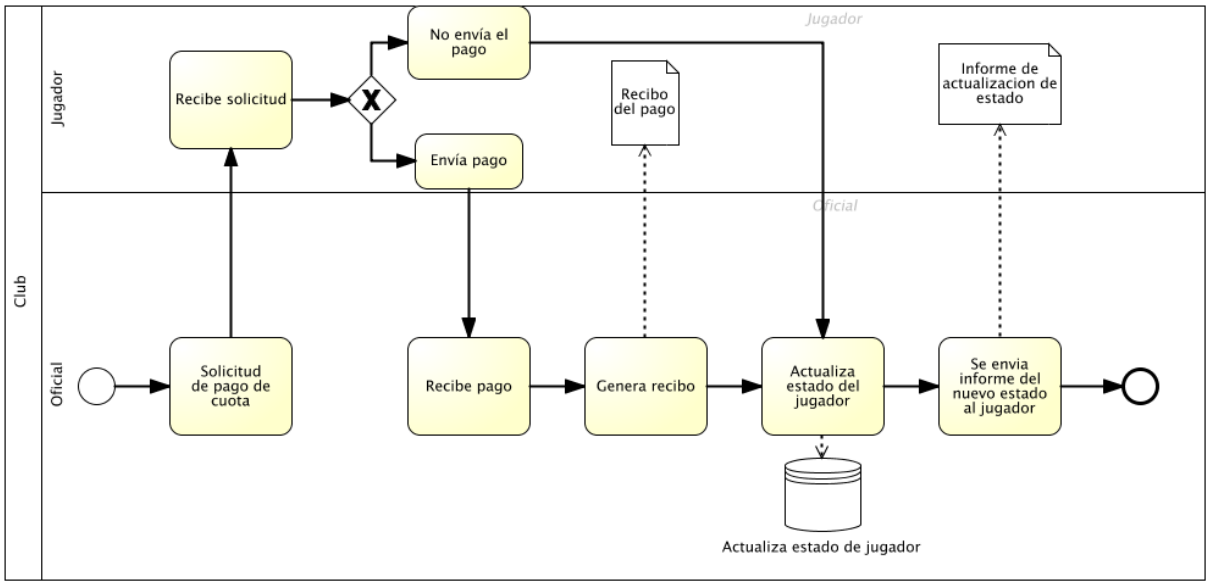
- Seguimiento de los eventos y actualización de resultados y clasificaciones.

- Control de las plantillas, asistencia y lesiones de las jugadoras en partidos y entrenamientos.

- Gestión y reserva de pistas y material necesario para los equipos del que dispone el club.

4.- Procesos de negocio

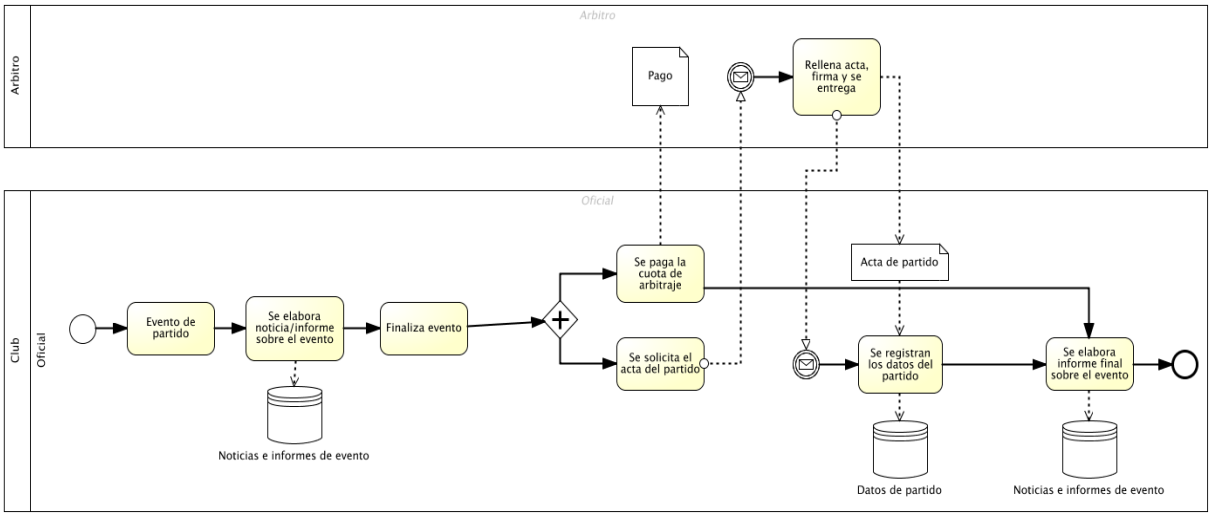
4.1 - *Cobro de cuotas*:



-Descripción:

Proceso de cobro de cuotas a las jugadoras, el pago del mismo les permitirá poder participar con el equipo en entrenamientos, partidos y competiciones.

4.2 - Evento de partido y/o competición:

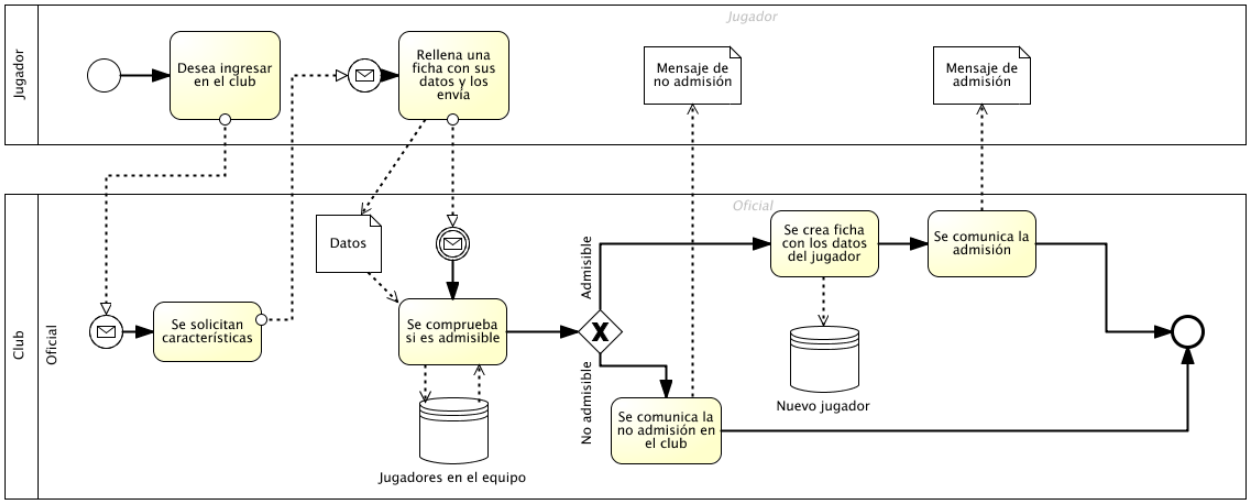


-Descripción:

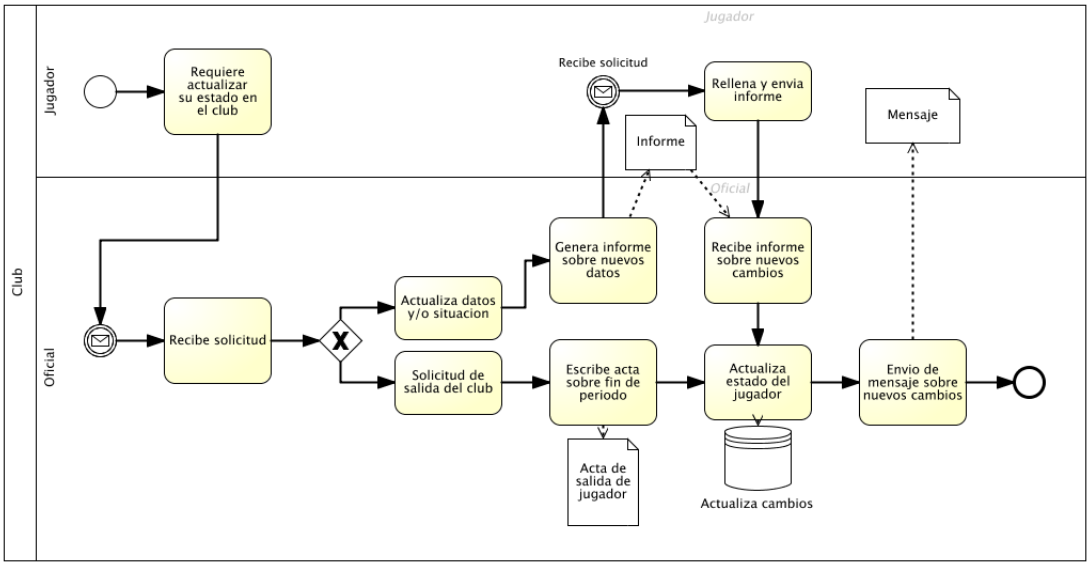
Proceso en el que se recogen los datos de un evento, para luego poder elaborar noticias, tablas de calificaciones, estadísticas, además de registrar el pago del arbitraje y el viaje (si lo hubiera).

4.3 - Seguimiento de la plantilla:

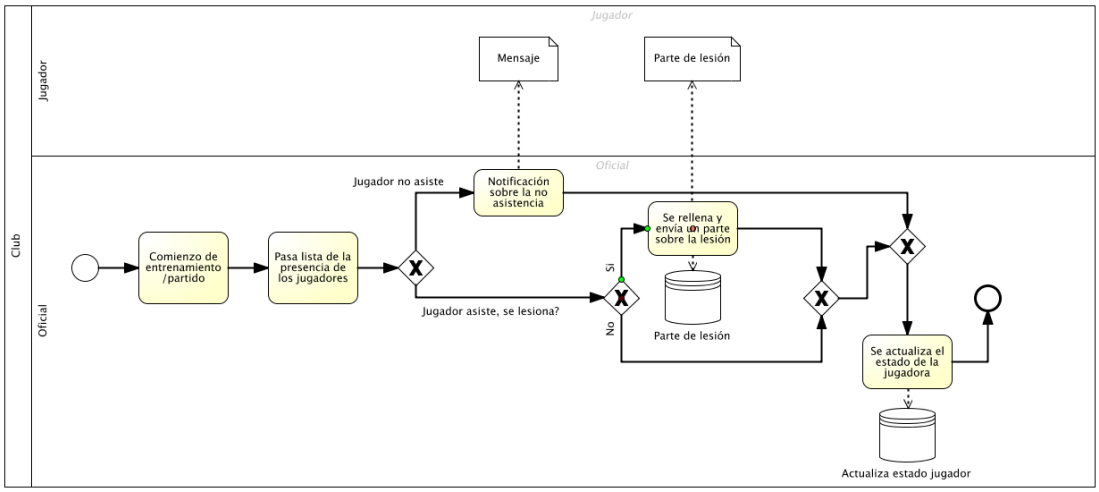
-Ingreso de jugador



-Nuevo estado del jugador (nuevos datos o eliminación)



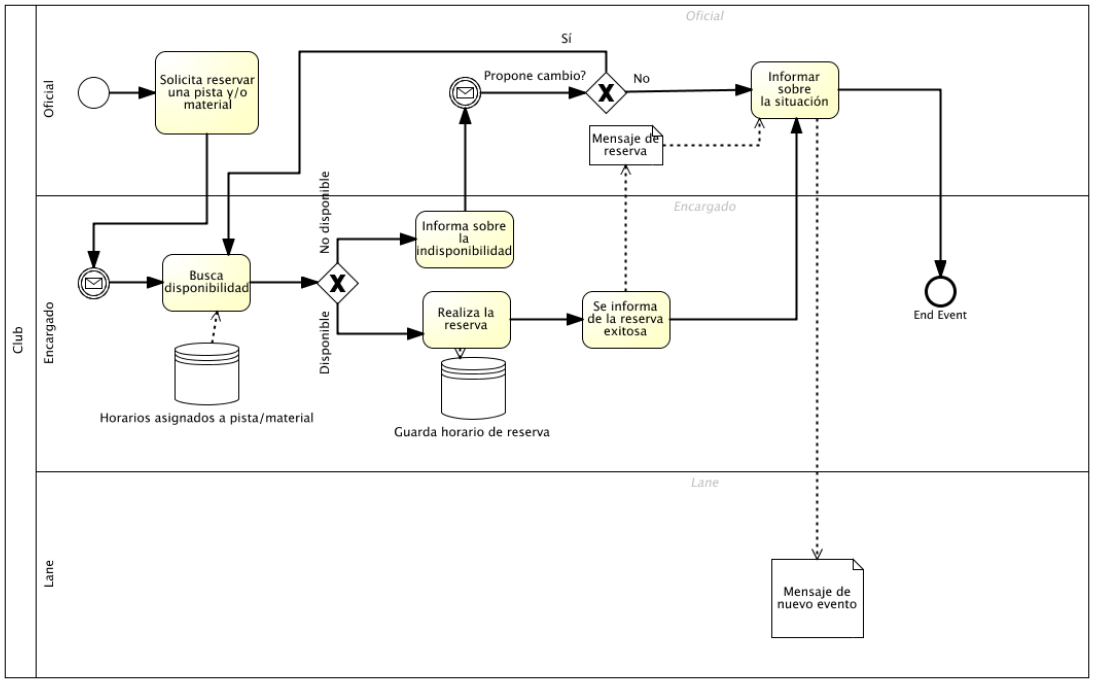
-Asistencia y lesiones



-Descripción:

Procesos que permitirá llevar un control de la plantilla, tanto si un nuevo jugador ingresa al club, modifica alguno de sus datos o su condición (lesionada) o desea salir del club; además de llevar un registro de su asistencia a eventos (entrenamientos, partidos) y de sus lesiones.

4.4 - *Reserva de material y/o pista*



-Descripción:

Proceso por el cual se solicita la reserva de algún material o pista para un posible entrenamiento o partido, pudiendo consultar los horarios de disponibilidad y/o las personas a las que han sido asignadas.

5.- Historias

-Cuotas:

Como Oficial

Quiero un control del pago de las cuotas por parte de las jugadoras

Para llevar al día y tener centralizado las jugadoras que estén al corriente del pago y cuáles no, para así llevar a cabo las acciones precisas como dar de alta o de baja en un equipo según corresponda.

-Cuotas:

Como Jugadora

Quiero tener acceso a mi historial de cuotas

Para poder saber en todo momento mi estado en el club, y si estoy al corriente, o no de mis cuotas.

-Cuotas:

Como Oficial

Quiero un control de la cuantía total de las cuotas del club

Para llevar la contabilidad de los gastos del equipo y saber si ya se ha cubierto el mínimo necesario para dar de alta el equipo en las competiciones y poder hacer frente a gastos de arbitrajes, material, etc.

-Material/pistas

Como Oficial

Quiero un medio de control de pistas/material

Para saber las condiciones y el material que dispongo para entrenar y evitar conflictos con otros oficiales y poder preparar mejor el trabajo a realizar teniendo en cuenta los materiales de los que puedo disponer.

-Material/pistas

Como Jugadora

Quiero conocer el horario de pistas

Para poder conocer exactamente en que sitio me toca entrenar cada día y así no depender del boca a boca para conocer el lugar donde deba de ir que pueden llegar a confusión.

-Plantillas/Entrenadores

Como Oficial

Quiero tener un listado con los datos de las jugadoras de mi equipo

Para así poder reaccionar de forma precisa a la hora de una urgencia o tener que atender a incidencias que ocurran, y también poder tener centralizado los datos de mi equipo para poder hacer una gestión de las fichas de forma más correcta y fácil.

-Plantillas/Entrenadores

Como Director Deportivo

Quiero tener un listado con los datos, historia y titulaciones de mis Oficiales

Para poder realizar las tareas de dirección deportiva y asignación de equipos acorde del historial de los entrenadores y monitores (oficiales) y no tener problemas con las titulaciones que limitan el poder hacerse cargo de un equipo según cual es la categoría de la cual forma parte ese equipo.

-Desplazamientos/arbitrajes

Como Oficial

Quiero llevar un listado de los lugares donde se practican las competiciones y la distancia de este a la capital

Para poder llevar un cálculo del coste de los desplazamientos a ese lugar dependiendo del kilometraje y el costo de la gasolina al igual que el coste arbitral que depende de esos factores, y de la categoría de la competición.

-Clasificación/resultados

Como jugadora

Quiero disponer de los resultados de mi equipo y de los partidos de los contrarios

Para poder ver el progreso de mi equipo y las clasificaciones correspondientes a este y a los demas equipos del club en un mismo lugar

-Clasificación/resultados

Como Oficial

Quiero disponer de resultados, clasificaciones, y próximos partidos de mi equipo.

Para poder planificar mas consecuentemente la temporada y los entrenamientos teniendo en cuenta los equipos a los que me tengo que enfrentar en fechas más próximas.

-Asistencia

Como Oficial

Quiero disponer de un control de asistencia a los partidos/entrenamientos

Para poder llevar un control con más facilidad de las jugadoras, premiar a las que vienen a entrenar y no hacerlo a las que no asisten asi como preocuparse por los motivos por los cuales una jugadora deje de asistir.

-Lesiones

Como Oficial

Quiero llevar un historial de lesiones de las jugadoras

Para poder tener un control de los días de baja que debe estar así como detectar recaídas y posibles fallos en las cargas de entrenamiento, además de poder hacer un plan especifico de entrenamiento para jugadoras dependiendo del tipo de lesión que hayan superado.

6.- Requisitos

REQUISITOS DE INFORMACIÓN

IRQ-01: Información sobre las jugadoras

El sistema deberá almacenar la información correspondiente a las jugadoras que componen los equipos. En concreto: sus datos personales (nombre, apellidos, número e imagen del DNI, edad, foto, peso, altura, dirección, mano hábil, posición, dorsal, categoría) y el historial de fechas y pagos de cuotas.

IRQ-02: Información sobre los oficiales

El sistema deberá almacenar la información correspondiente a los oficiales pertenecientes a los equipos y al club. En concreto: sus datos personales (nombre, apellidos, numero e imagen del DNI, foto), su titulación y la categoría a la que pertenece el equipo del que es encargado así como su sueldo correspondiente.

IRQ-03: Información sobre los equipos

El sistema deberá almacenar la información básica sobre los equipos. En concreto un identificador para diferenciarlo de los demás, la categoría a la que pertenece, las jugadoras que lo componen y la liga en la que compiten.

IRQ-04: Control de lesiones

El sistema tendrá que permitir a los oficiales tener un historial de lesiones de las jugadoras, permitiendo tener un control de las jugadoras que están lesionadas, la fecha de lesión y de recuperación, y las recaídas.

IRQ-05: Control de asistencia

El sistema deberá permitir a los oficiales llevar el control de asistencia de las jugadoras, permitiendo anotar si una jugadora va a un partido y/o entrenamiento concretos, pudiendo notificar la ausencia continuada a la jugadora.

IRQ-06: Control de pista y material

El sistema tendrá que permitir llevar el control sobre los horarios de pistas disponibles y quien las posee, así como reservar en el caso de que sea necesario para un oficial u otro. En concreto, también deberá controlar que persona posee el material y en el stock, que hay material suficiente, su precio, y el proveedor que al que lo tenemos que adquirir en el caso de necesitarlo.

IRQ-07: Información de los partidos

El sistema deberá guardar información de los partidos que se disputen para su posterior análisis. En concreto los equipos local y visitante, la categoría a la que pertenecen los equipos, el lugar y su emplazamiento, la distancia al emplazamiento, fecha donde se realiza o se va a realizar el partido, el coste del arbitraje, el tipo de competición (amistoso, copa o liga) y el resultado del partido.

IRQ-08: Información de la competición

El sistema deberá llevar un control de la competición en la que participan los equipos. En concreto el tipo de los partidos (copa, liga o amistosos), la fecha de inicio y fin de la competición, la categoría y la clasificación generada a partir de esos resultados.

IRQ-09: Información de clasificaciones

El sistema deberá guardar los resultados de las competiciones. En concreto los puntos y la posición en la tabla de cada equipo.

REGLAS DE NEGOCIO

RN-01: Jugadoras y categoría

Una jugadora no puede pertenecer a una categoría diferente a la correspondiente a su edad. En concreto, las categorías correspondientes a las edades son Senior--> nacidas el 95 o anterior, juvenil nacidas entre 96 y 97, cadete nacidas entre 98 y 99, infantil nacidas entre 2000 y 2001, alevín nacidas entre 2002 y 2003, y benjamín 2004 o posterior (refiriéndonos a la temporada 2013/2014).

RN-02: Oficiales y categoría

Un oficial no podrá ser responsable de un equipo que milite en una categoría superior a la permitida según su titulación.

En concreto, para ser oficial de Senior se debe tener titulación nacional, para ser oficial de juvenil se debe tener titulación territorial, para ser oficial cadete o infantil se debe tener categoría provincial y para ser oficial alevín o benjamín se debe tener categoría de monitor.

RN-03 Jugadoras y cuota

Una jugadora no podrá disputar los partidos oficiales (liga y copa) si no está al corriente del pago de las cuotas.

RN-04 Jugadoras y equipo

Un equipo no podrá tener más de 18 jugadoras activas y menos de 10 para poder disputar competiciones.

REQUISITOS NO FUNCIONALES

RNF-01 Pago de arbitraje y desplazamiento

Si el partido se juega en casa, solo se deberá pagar el costo del arbitraje. Sin embargo, si el partido se juega fuera, además del pago del arbitraje, se debe pagar el costo del desplazamiento.

7.- Pruebas de aceptación

-Cuotas:

El sistema deberá mostrar el estado en el que se encuentran las jugadoras respecto al pago de las cuotas, avisando a la jugadora y a los oficiales correspondientes de los retrasos.

-Prueba de aceptación:

El sistema deberá mostrar un listado con la relación de cuotas y jugadoras y avisará mediante correo electrónico a la jugadora y al oficial encargado de su equipo.

-Cuotas:

El sistema deberá mostrar la cuantía total de las cuotas del club, llevando el control de de la relación ingresos-gastos.

-Prueba de aceptación:

El sistema deberá enviar un aviso cuando el oficial correspondiente a la inscripción de los equipos pueda disponer de los ingresos necesarios para hacer esa inscripción.
El sistema deberá avisar cuando a falta de 7 días de la fecha estipulada anteriormente por la federación como fecha fin de inscripción de equipos aun no se cubra la cuantía suficiente para hacerlo.

-Material/pistas

El sistema deberá mostrar las pistas y material disponibles y ocupados y cuál es el equipo que los posee.

-Prueba de aceptación:

El sistema permitirá registrar reservas de pistas y material disponible y enviará un aviso de confirmación de reserva al correo del oficial correspondiente.
El sistema deberá enviar un aviso por correo electrónico a las jugadoras correspondientes en el caso de que cambie el estado de reserva de pista y/o el lugar de la misma.

-Plantillas/Entrenadores

El sistema deberá presentar un listado de las personas del club, diferenciando entre jugadoras y oficiales.

-Prueba de aceptación:

Se pide un listado de jugadoras filtrado por categorías y el listado aparece.
En el sistema se modifica o registra una nueva persona, se pide ese listado y aparecen los datos nuevos/modificados
Se elimina una persona y se pide el listado, y la persona ya no aparece en el listado.
Si se introduce una jugadora, el sistema deberá clasificarla en una categoría según su edad.
Si un equipo de una categoría supera las 18 jugadoras, el oficial deberá recibir un aviso del sistema indicándole que debe hacer una separación de equipos en esa categoría.

-Desplazamientos/arbitrajes

El sistema deberá contener un listado de los lugares donde se practican las competiciones y la distancia de este a la capital, calculando gastos de kilometraje y arbitraje.

-Prueba de aceptación:

El sistema debe calcula el gasto de gasolina desde la capital al lugar de partido, se le pide el cálculo, y lo enseña por pantalla, si el partido es en la capital, el gasto es 0.
Se le añade un partido nuevo, y el sistema debe calcular y mostrar el gasto de arbitraje dependiendo del kilometraje, día, hora y categoría del partido.

-Partidos/Clasificación/resultados

El sistema debe mostrar resultados y clasificación de los equipos.

-Prueba de aceptación:

Al sistema se le pide la clasificación y resultados de un equipo y este devuelve un listado con los partidos jugados en esa competición, y los resultados de estos, así como un listado de los equipos ordenados según partidos ganados, empatados y perdidos.
Si el partido jugado es amistoso solo se enseñara el resultado y no computa para ninguna clasificación.
Si al sistema se le piden los partidos que aun no se han jugado debe devolver un listado con los partidos posteriores a la fecha actual sin resultado alguno.
Si al sistema se le introduce un partido de tipo amistoso, este debe empezar y acabar en la misma fecha.

-Asistencia

El sistema deberá llevar un listado de las jugadoras con el porcentaje de asistencia a los eventos.

-Prueba de aceptación:

si la jugadora falta repetidamente a los eventos se le enviará un aviso al correo electrónico indicándole la incidencia.
Si la jugadora falta ese día al evento, el porcentaje disminuye.

-Lesiones

El sistema deberá llevar un historial de lesiones de las jugadoras así como generar un parte de lesión cuando sea preciso.

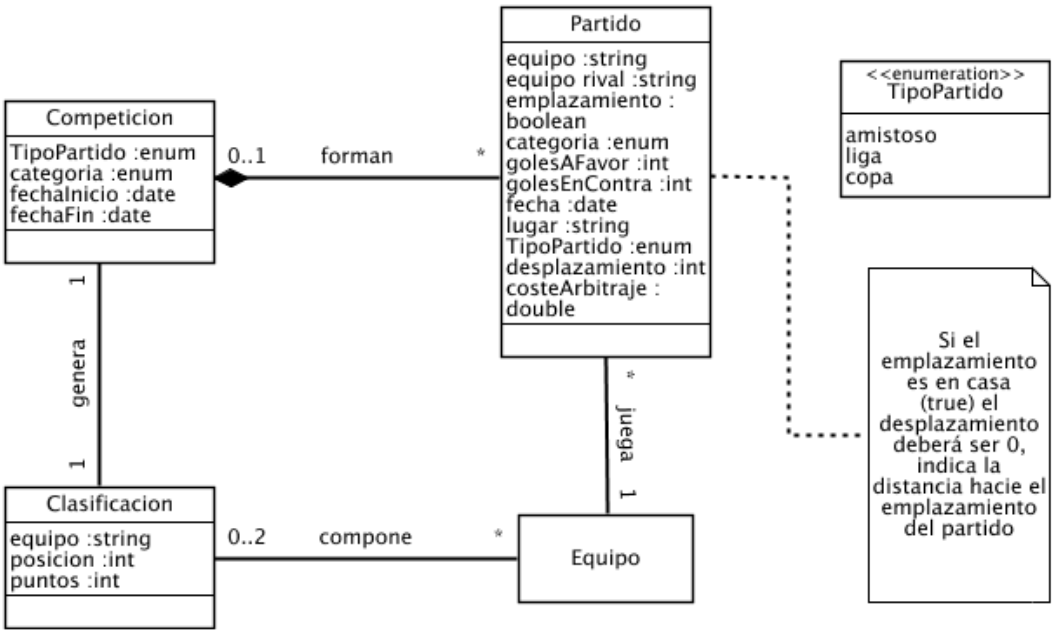
-Prueba de aceptación:

Si se produce una lesión el sistema deberá actualizarse con los datos de la nueva lesión, así como generar un parte de accidente para que la persona sea tratada en la mutua.
Si la jugadora recibe el alta, el sistema deberá actualizarse indicando que la jugadora está disponible para entrenar de nuevo, enviando un aviso al oficial correspondiente.

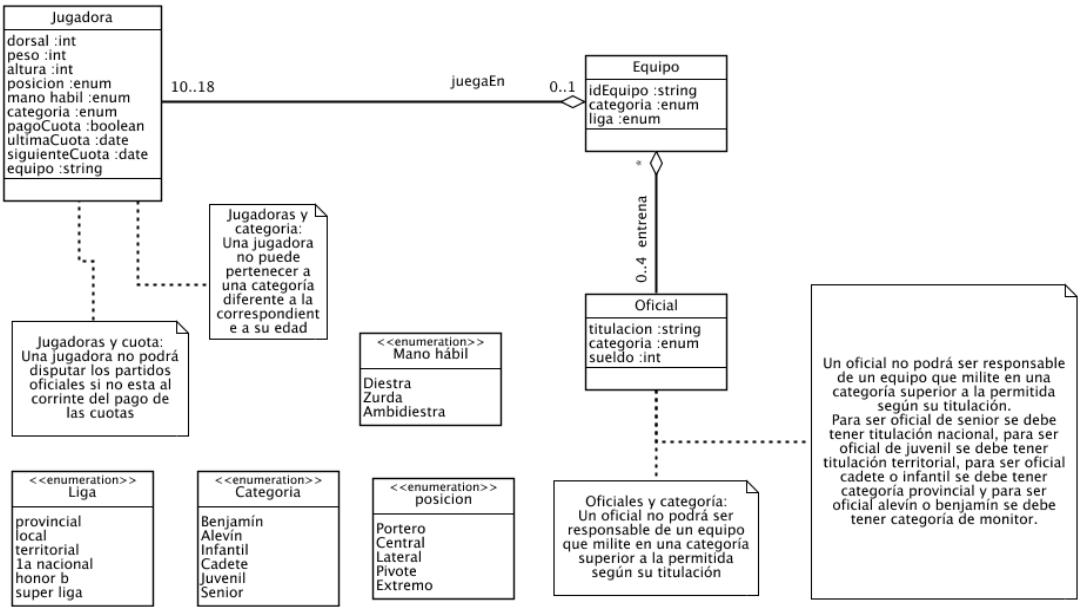
8.- Modelado conceptual

8.1 – Diagramas de clase UML

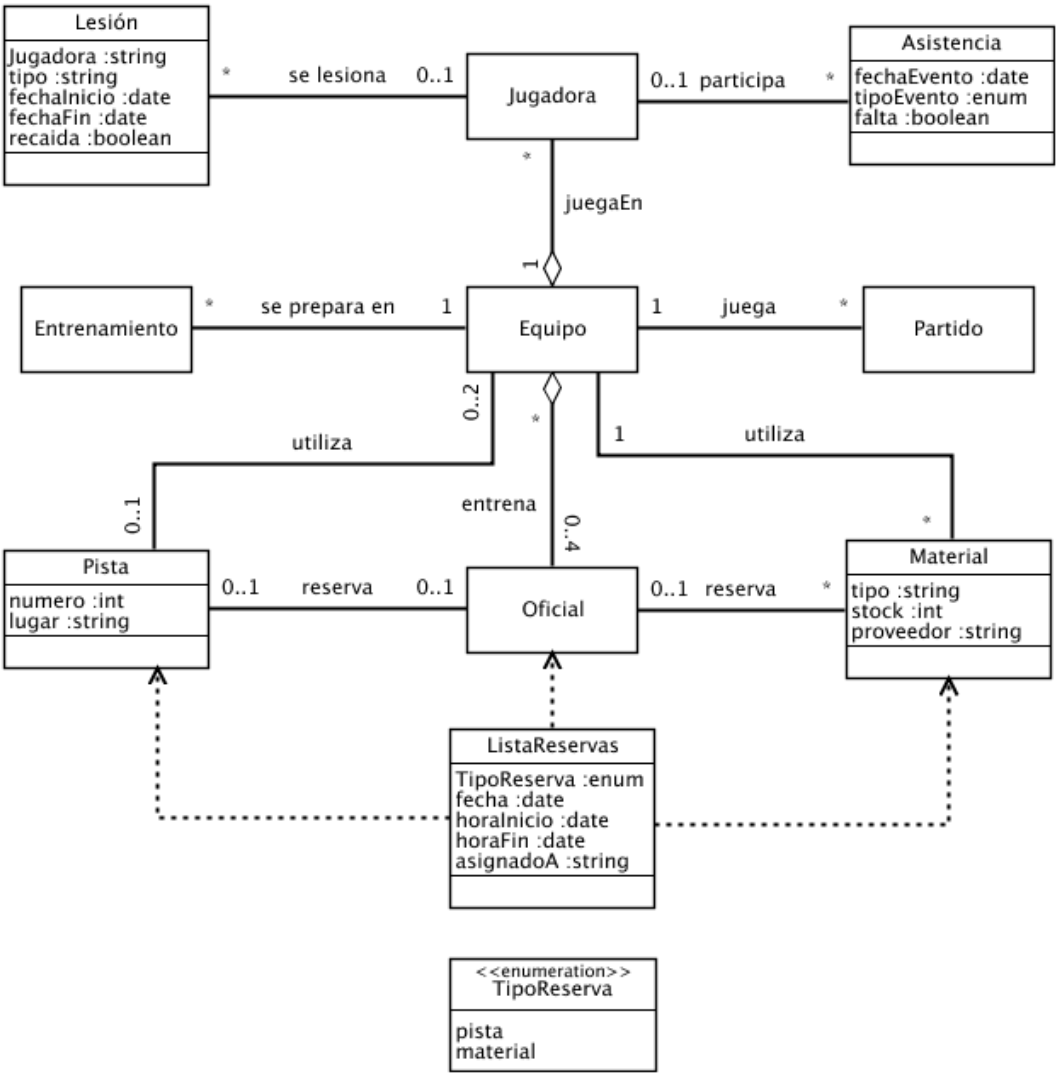
+UML-1: Diagrama de competición y partidos



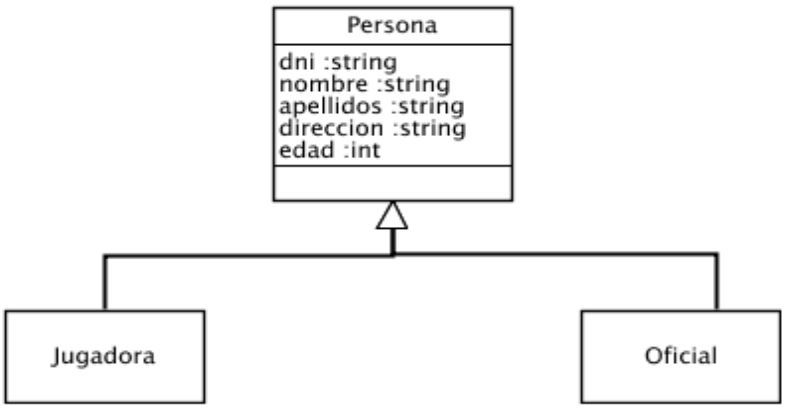
+UML-2: Diagrama de Jugadoras, Oficiales y equipo



+UML-3: Diagrama de lesiones y asistencias, y pista y materiales

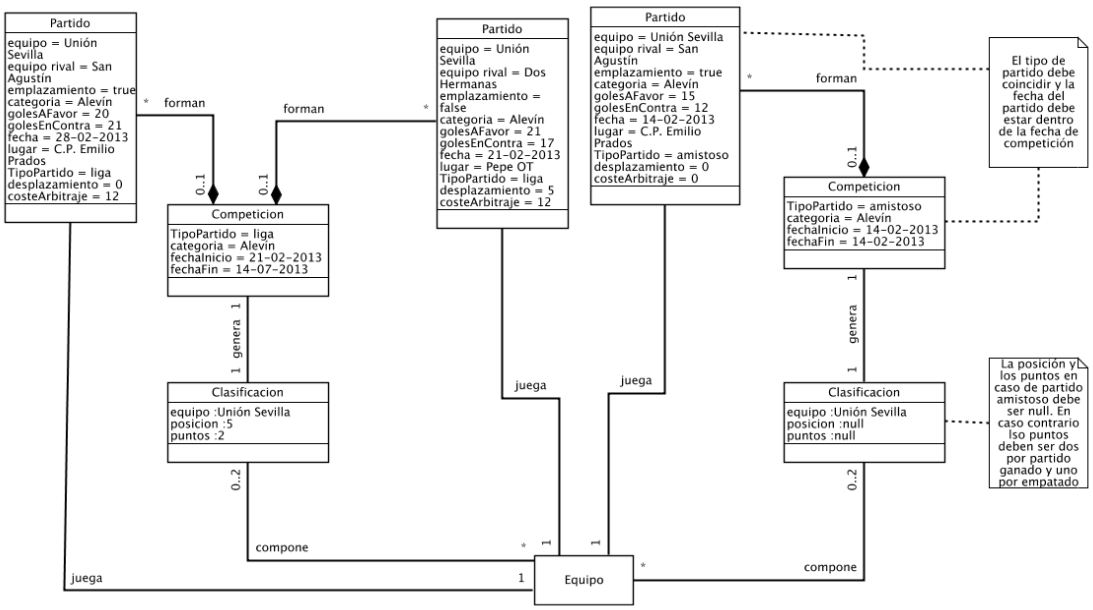


+UML-4: Jerarquía de personas

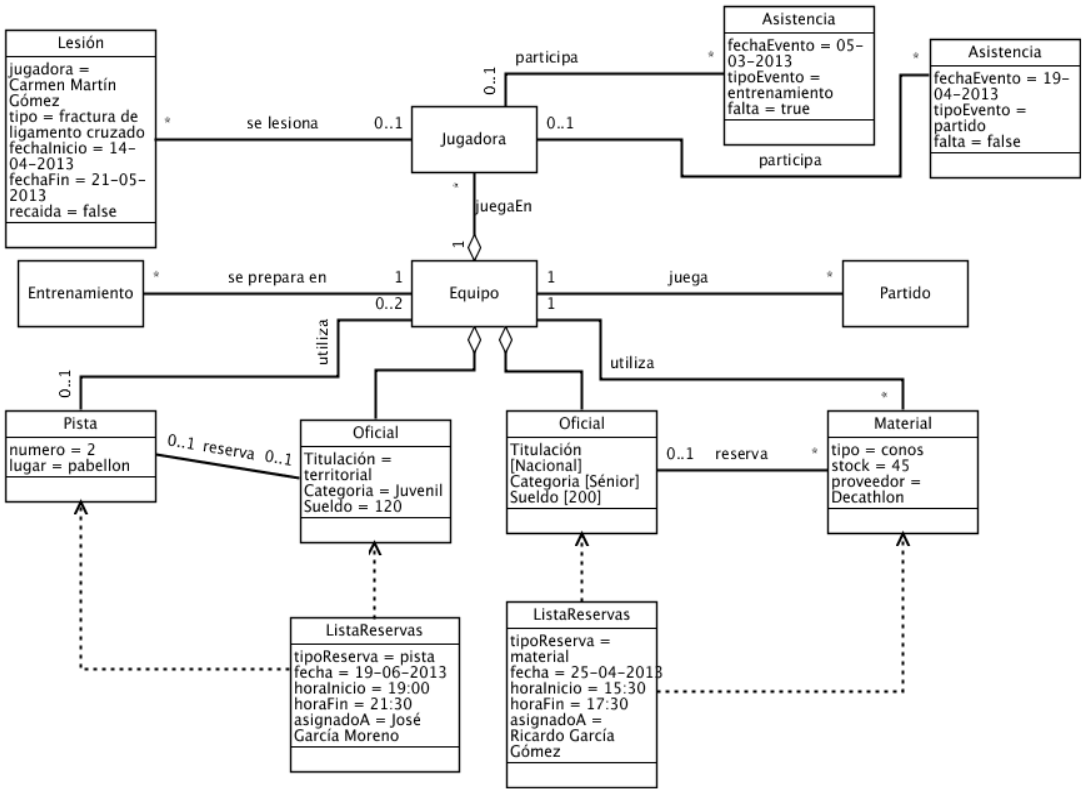


8.2 – Escenarios de prueba

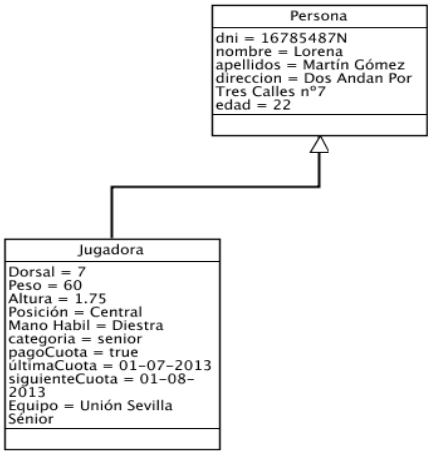
+Escenario 1: Competición, partidos y clasificaciones



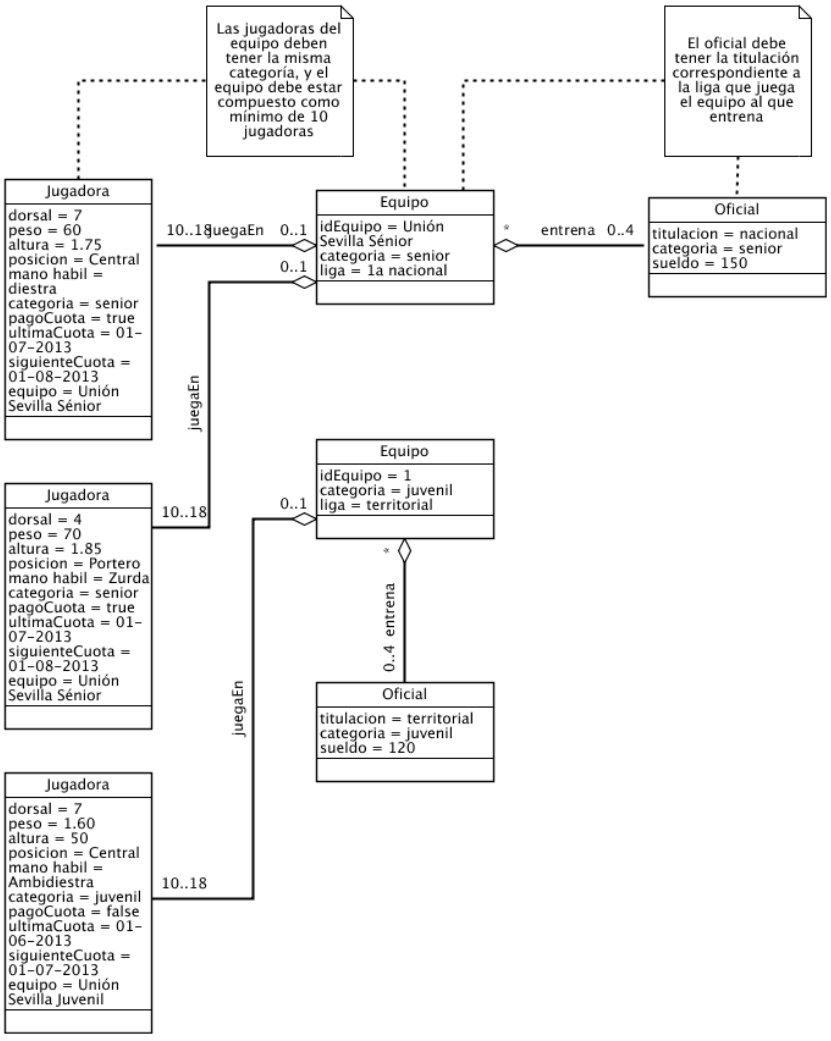
+Escenario 2: Lesión, asistencias y reservas



+Escenario 3: Persona



+Escenario 4: Jugadoras y oficiales, equipos pertenecientes al club



9.- Matrices de trazabilidad

-*Requisitos/Diagramas uml*

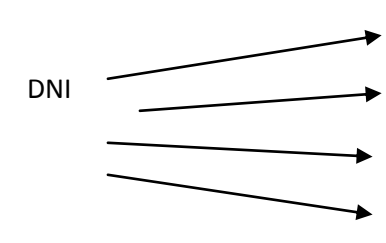
Req/Uml	Uml-1	Uml-2	Uml-3	Uml-4
IRQ-01		X	X	X
IRQ-02		X	X	X
IRQ-03	X	X	X	
IRQ-04		X		
IRQ-05		X		
IRQ-06		X		
IRQ-07	X			
IRQ-08	X			
RN-01			X	
RN-02			X	
RN-03			X	
RN-04			X	
RNF-01	X			

-Historias de usuario/Diagramas uml

Hist/Uml	Uml-1	Uml-2	Uml-3	Uml-4
Cuotas			X	
Material/pistas		X		
Plantillas/Entrenadores			X	X
Desplazam./Arbitrajes	X			
Clasificación/Resultados	X			
Asistencia		X		
Lesiones		X		

10.- Modelo relacional en 3FN

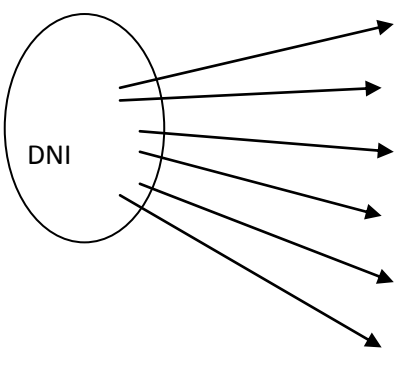
Relación persona



PK(DNI)

FK()

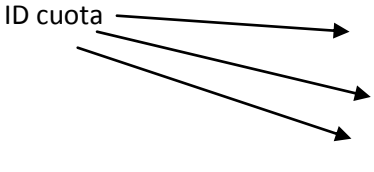
Relación Jugadora



PK(DNI, Dorsal)

FK(Id Equipo/Equipo, DNI/Persona)

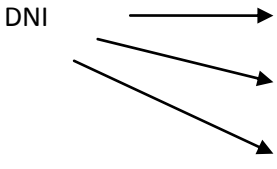
Relacion Cuotas



pk(id cuota)

fk(DNI/Persona)

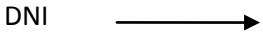
Relación Oficial



PK(DNI)

FK(DNI/Persona)

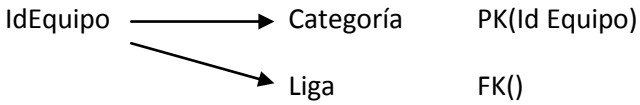
Relación oficial/equipo



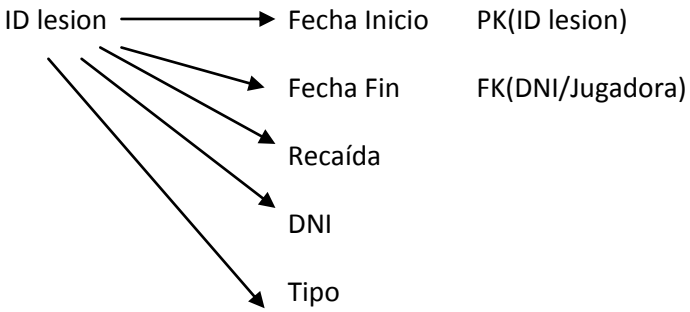
PK(DNI)

FK(Dni/Oficial, ID Equipo/Equipo)

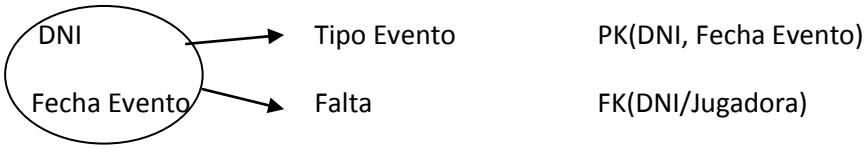
Relación Equipo



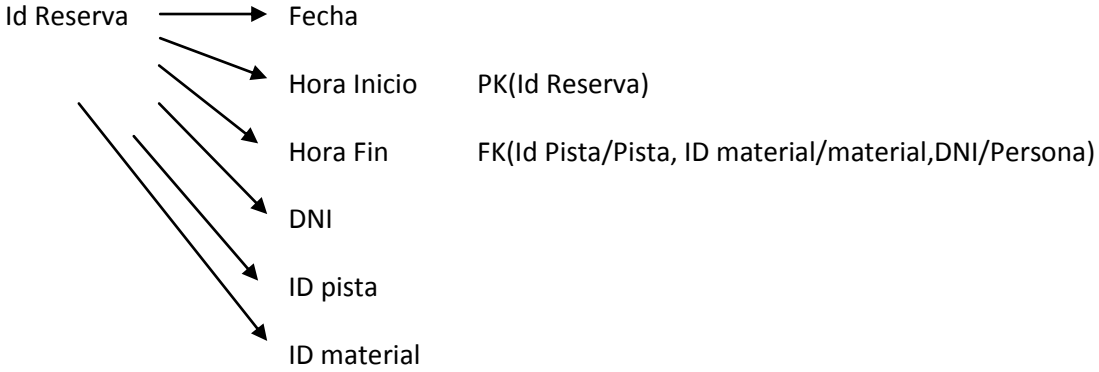
Relación Lesión



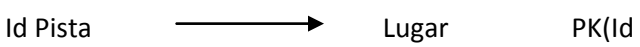
Relación Asistencia



Relación Reservas

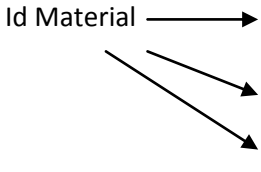


Relación Pista



FK()

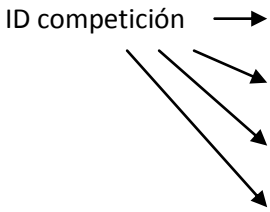
Relación Material



PK(Id Material)

FK()

Relación Competición



PK(ID competición)

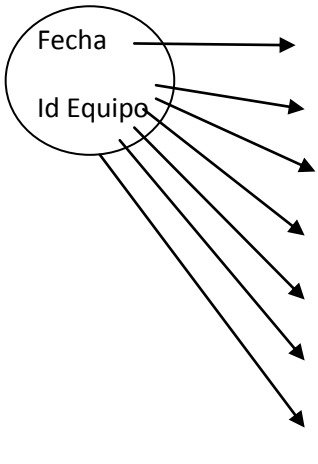
FK()

Relación Clasificación

PK(Id Equipo, Id Competicion)

FK(Id Equipo/Equipo, Id Competición/Competición)

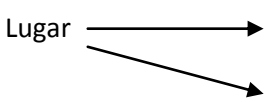
Relación Partido



PK(Fecha, Id Equipo)

FK(Id Equipo/Equipo,
Id Competición/Competición
Lugar/Lugar Partido)

Relación Lugar partido



PK(Lugar)

FK()

-Justificación de jerarquía sobre la clase Persona:

En nuestro proyecto incluimos una tabla Persona, de la que heredan Jugadora y Oficial, para tener en cuenta todas las personas relacionas con el club; de esta forma, tenemos una conexión directa en la consulta, evitando recorrer más tablas.

11.- Modelo tecnológico

```
-- 1. Creacion Tablas.sql


drop table Persona cascade constraints;

drop table Jugadora cascade constraints;

drop table Equipo cascade constraints;

drop table Competicion cascade constraints;

drop table LugarPartido cascade constraints;

drop table OficialEquipo cascade constraints;

drop table Reservas;

drop table Oficial cascade constraints;

drop table Lesion;

drop table Asistencia;

drop table Pista cascade constraints;

drop table Material;

drop table Clasificacion;

drop table PARTIDO;

drop table cuotas cascade constraints;




create table Competicion(

  IDCOMPETICION integer,

  TipoCompeticion varchar2(8) not null check(Tipocompeticion = 'amistoso' or Tipocompeticion = 'liga' or
Tipocompeticion ='copa' ),

  Categoria varchar2(8) not null check(Categoria = 'benjamin' or Categoria = 'alevin' or Categoria = 'infantil'

                    or Categoria = 'cadete' or Categoria = 'juvenil' or Categoria = 'senior'),

  FechaInicio date,

  FechaFin date,
```

```
  primary key (IDCompeticion)
);

create table LugarPartido(
  Lugar varchar2(50) not null,
  Desplazamiento integer,
  CosteArbitraje float,
  primary key (Lugar)
);

--Equipo: Hecha
create table Equipo(
  IDEquipo integer,
  Categoria varchar2(8) not null
                check(Categoria = 'benjamin' or Categoria = 'alevin' or Categoria = 'infantil'
                    or Categoria = 'cadete' or Categoria = 'juvenil' or Categoria = 'senior'),
  Liga varchar2(11) not null check (Liga in ('provincial','local','territorial','1ªnacional','honor b','superliga')),
  primary key (IDEquipo)
);

create table Persona(
    DNI char(9),
    Nombre varchar2(50) not null,
    Apellidos varchar2(50) not null,
    DIRECCION varchar2(50),
    fechanacimiento date not null,
    primary key(DNI)
    );

CREATE TABLE Jugadora (DNI CHAR(9),
            Dorsal NUMBER,--triger aqui, debe ser positivo
```

```
        Peso NUMBER,--idem dorsal
        POSICION varchar2(7) check (POSICION in ('central','lateral','pivote','portero','extremo')),
        Manohabil varchar2(1) check (Manohabil in ('D','Z','A')),
        Categoria varchar2(8) not null check(Categoria = 'benjamin' or Categoria = 'alevin' or Categoria = 'infantil'
                or Categoria = 'cadete' or Categoria = 'juvenil' or Categoria = 'senior'),
        IDEquipo integer,
        PRIMARY KEY (DNI),
        foreign key(IDEquipo) references Equipo,
        foreign key(DNI) references Persona
);



CREATE TABLE Oficial (DNI CHAR(9),
        Sueldo NUMBER,--triger mayor o igual que 0
        Titulacion varchar2(11) not null check( Titulacion in ('Nacional','Territorial','Provincial','Monitor')),
        Categoria varchar2(8) not null check(Categoria = 'benjamin' or Categoria = 'alevin' or Categoria = 'infantil'
                or Categoria = 'cadete' or Categoria = 'juvenil' or Categoria = 'senior'),
        PRIMARY KEY (DNI),
        foreign key(DNI) references Persona
       );
create table LESION(
        idlesion integer,
        DNI CHAR(9),
        Tipo varchar2(50) not null,
        FechaInicio date,
        FechaFin date,--triger aqui
        RECAIDA char(2) check ( RECAIDA in ('si','no')),
        PRIMARY KEY (idlesion),
        foreign key(DNI) references Jugadora
       );
CREATE TABLE Asistencia
```

```
            (DNI CHAR(9),
            TipoEvento varchar2(50) not null,
            FechaEvento date,
            Falta char(2) not null check (Falta in ('si','no')),
            PRIMARY KEY (DNI,FechaEvento),
            foreign key(DNI) references Jugadora
);
create table PISTA
            (IDpista integer,
            LUGAR varchar2(50) not null,
            PRIMARY KEY (IDpista)
           );
create table MATERIAL
            (IDmaterial integer,
            Stock number,--triguer debe ser mayor o igual a 0
            Proveedor varchar2(50),
            TIPO varchar2(50) not null,
            PRIMARY KEY (IDmaterial)
           );
CREATE TABLE Clasificacion
            (IDEquipo integer,
            IDCOMPETICION integer,
            Puntos number default 0,
            PRIMARY KEY (IDEquipo,IDCompeticion),
            foreign key(IDEquipo) references Equipo,
            foreign key(IDCompeticion) references Competicion
           );
CREATE TABLE Partido
            (IDEquipo integer,
            IDCompeticion integer,
            Fecha date,
```

```
            EquipoRival varchar2(50) not null,
            Emplazamiento varchar2(50),
            Categoria varchar2(8) not null check(Categoria = 'benjamin' or Categoria = 'alevin' or Categoria = 'infantil'
                    or Categoria = 'cadete' or Categoria = 'juvenil' or Categoria = 'senior'),
            GolesFavor number,--triger mayor a 0
            GolesContra number,--idem arriba
            Lugar varchar2(50) not null,
            TipoPartido varchar2(8) not null check(TipoPartido = 'amistoso' or TipoPartido = 'liga' or TipoPartido
='copa' ),
            PRIMARY KEY (Fecha,IDEquipo),
            foreign key(IDEquipo) references Equipo,
            foreign key(IDCompeticion) references Competicion,
            foreign key(Lugar) references LugarPartido
           );

create table OficialEquipo(
 DNI char(9),
 IDEquipo integer,
 primary key(DNI),
 foreign key(DNI) references Oficial,
 foreign key(IDEquipo) references Equipo
 );

 --Pista

create table Reservas(
 IDReserva integer,
 DNI char(9),
 HoraInicio date,
 HORAFIN date,--triger aqui
 IDPISTA integer,
```

```
 idmaterial integer,
 primary key(IDRESERVA),
 foreign key(IDPISTA) references PISTA,
 foreign key(IDMATERIAL) references MATERIAL,
 foreign key(DNI) references Persona
 );

 create table CUOTAS(
        idcuota integer,
        dni char(9),
        PAGOCUOTA char(2) not null check (PAGOCUOTA in ('si','no')),
        FECHAPAGO date,
        primary key(IDCUOTA),
        foreign key (DNI) references PERSONA
);

-- 1. Secuencias.sql

//SECUENCIAS//
drop sequence sec_competicion;
drop sequence sec_equipo;
drop sequence sec_reserva;
drop sequence sec_lesion;
drop sequence sec_pista;
drop sequence sec_material;
drop sequence sec_cuota;

create sequence sec_competicion increment by 1 start with 1;
create sequence sec_equipo increment by 1 start with 1;
create sequence sec_reserva increment by 1 start with 1;
create sequence sec_lesion increment by 1 start with 1;
```

```
create sequence sec_pista increment by 1 start with 1;

create sequence sec_material increment by 1 start with 1;

create sequence sec_cuota increment by 1 start with 1;


//TRIGGERS//

-- 2.Trigger_Costes.sql


CREATE OR REPLACE TRIGGER CostesPartido

 BEFORE

 INSERT or update ON LugarPartido

 FOR EACH ROW


BEGIN

 IF (:new.Desplazamiento < 0 or :new.CosteArbitraje < 0)

 THEN raise_application_error

 (-20800,'el desplazamiento y el coste de arbitraje debe ser mayor o igual a 0');

 END IF;

END;


-- 3.Trigger_Goles.sql


CREATE OR REPLACE TRIGGER  Goles

 BEFORE

 INSERT or update ON Partido

 FOR EACH ROW


BEGIN

 IF (:new.GolesFavor < 0 or :new.GolesContra < 0)

 THEN raise_application_error

 (-20100,'los goles no pueden ser negativos');

 END IF;
```

```
END;

-- 4.Trigger_Jugadora.sql

CREATE OR REPLACE TRIGGER Atributos
 BEFORE
 INSERT or update ON Jugadora
 FOR EACH ROW

BEGIN
 IF (:new.Dorsal < 1)
 THEN raise_application_error
 (-20800,'el dorsal debe ser positivo');
 END IF;
 IF (:new.Peso < 1)
 THEN raise_application_error
 (-20800,'el peso debe ser positivo');
 END IF;
END;

-- 5.Trigger_LimiteJugadoras.sql

create or replace
trigger MAXJUGADORAS
 BEFORE
 INSERT or update ON Jugadora
 for each row
declare jugadoras integer;
begin
 select COUNT(*) into JUGADORAS from JUGADORA where IDEQUIPO = :new.IDEQUIPO group by IDEQUIPO;
 IF (jugadoras > 17)
```

```
 then RAISE_APPLICATION_ERROR
 (-20800,'un equipo no puede estar compuesto por mas de 18 jugadoras');
 END IF;

END;

--6. Trigger_LimiteOficiales.sql
create or replace
trigger MAXoficiales
 before
 INSERT or update ON oficialequipo
 for each row
declare OFICIALES integer;
begin
 select COUNT(*) into OFICIALES from OFICIALequipo where IDEQUIPO = :new.idequipo group by  IDEQUIPO;

 IF (inserting and OFICIALES > 3)
 then RAISE_APPLICATION_ERROR
 (-20800,'un equipo no puede estar compuesto por mas de 4 oficiales');
 END IF;

END;

--7. Trigger_Material.sql
CREATE OR REPLACE TRIGGER materiales
 BEFORE
 INSERT or update ON Material
 FOR EACH ROW

BEGIN
 IF (:new.Stock < 0)
```

```
 THEN raise_application_error
 (-20900,'el stock no puede ser negativo');
 END IF;
END;

--8. Trigger_Oficial.sql
CREATE OR REPLACE TRIGGER sueldos
 BEFORE
 INSERT or update ON Oficial
 FOR EACH ROW

BEGIN
 IF (:new.Sueldo < 0)
 THEN raise_application_error
 (-20900,'el sueldo no puede ser negativo');
 END IF;
END;
--9. Trigger_Fechas.sql
CREATE OR REPLACE TRIGGER FechasCompeticion
BEFORE INSERT OR UPDATE ON Competicion for each row
begin
 if( :new.FechaInicio > :OLD.FechaFin) then raise_application_error(-20600,'Fecha de inicio posterior a la fecha
final');
 end if;

 if( :new.FechaFin < :OLD.FechaInicio) then raise_application_error(-20600,'Fecha final anterior a la fecha de
inicio');
 end if;

 if( :new.FechaFin < :NEW.FechaInicio) then raise_application_error(-20600,'Error con las fechas');
 end if;
```

```
  if( :NEW.FechaInicio > :new.FechaFin) then raise_application_error(-20600,'Error con las fechas');

  end if;

END;

--10. Trigger_Fechas2.sql

create or replace trigger FechasLesion

BEFORE INSERT OR UPDATE ON Lesion for each row


begin

  if( :new.FechaInicio > :OLD.FechaFin) then raise_application_error(-20600,'Fecha de inicio posterior a la fecha
final');

  end if;


  if( :new.FechaFin < :OLD.FechaInicio) then raise_application_error(-20600,'Fecha final anterior a la fecha de
inicio');

  end if;


  if( :new.FechaFin < :NEW.FechaInicio) then raise_application_error(-20600,'Error con las fechas');

  end if;


  if( :NEW.FechaInicio > :new.FechaFin) then raise_application_error(-20600,'Error con las fechas');

  end if;

end;

--11. Trigger_Fechas3.sql

CREATE OR REPLACE TRIGGER FechasReservas

BEFORE INSERT OR UPDATE ON Reservas for each row

begin

  if( :new.HoraInicio > :OLD.HoraFin) then raise_application_error(-20600,'Hora de inicio posterior a la hora final');

  end if;


  if( :new.HoraFin < :OLD.HoraInicio) then raise_application_error(-20600,'Hora final anterior a la hora de inicio');

  end if;
```

```
 if( :new.HoraFin < :NEW.HoraInicio) then raise_application_error(-20600,'Error con las horas');

 end if;


 if( :NEW.HoraInicio > :new.HoraFin) then raise_application_error(-20600,'Error con las horas');

 end if;

end;


//FUNCIONES//

--1. Funcion_AssertEquals.sql

CREATE OR REPLACE FUNCTION ASSERT_EQUALS (salida BOOLEAN, salida_esperada BOOLEAN) RETURN VARCHAR2
AS

BEGIN

 IF (salida = salida_esperada) THEN

   RETURN 'EXITO';

 ELSE

   RETURN 'FALLO';

 end if;

END ASSERT_EQUALS;

--2. Funcion_Coste_Arbitro.sql

create or replace function ARBITRO ( L_LUGAR in LUGARPARTIDO.LUGAR%type, l_desplazamiento in
LUGARPARTIDO.DESPLAZAMIENTO%type )

RETURN float IS W_cost LUGARPARTIDO.COSTEARBITRAJE%type;

begin

 SELECT ((l_desplazamiento*0.2)+15) INTO W_cost FROM lugarpartido WHERE lugar = l_lugar;

 return W_COST;

end ARBITRO;


//PROCEDURES//

--3. ProcedureActualizalugarpartido.sql

CREATE OR REPLACE PROCEDURE actualiza_lugarpartido(


    P_LUGAR            in LUGARPARTIDO.LUGAR%type,
```

```
    P_DESPLAZAMIENTO     in LUGARPARTIDO.DESPLAZAMIENTO%type)

IS BEGIN

 update lugarpartido set costearbitraje = arbitro(p_lugar,p_desplazamiento) where lugar = P_LUGAR;

 commit work;
end;
--4. ProcedureAsistencia.sql
CREATE OR REPLACE PROCEDURE nuevo_asistencia(

    a_dni            IN asistencia.dni%TYPE,
    a_tipo           IN asistencia.tipoevento%TYPE,
    a_fecha          IN asistencia.fechaevento%TYPE,
    a_falta          IN asistencia.falta%type)
IS BEGIN

 INSERT INTO asistencia VALUES(a_dni,a_tipo,a_fecha,a_falta);

 commit work;
end;
--5. ProcedureClasificacion.sql
create or replace procedure nueva_clasificacion
      (c_IDEquipo in clasificacion.IDEquipo%type,
      c_IDCompeticion in clasificacion.IDCompeticion%type,
      c_Puntos in clasificacion.puntos%type)
is
```

```
begin
insert into clasificacion values(c_idequipo,c_IDCompeticion,c_puntos);
commit work;
end;
--6. ProcedureCompeticion.sql
create or replace procedure nueva_competicion
      (
      c_Tipocompeticion in Competicion.Tipocompeticion%type,
      c_Categoria in Competicion.Categoria%type,
      c_FechaInicio in Competicion.FechaInicio%type,
      c_FechaFin in Competicion.FechaFin%type)
is
begin
insert into COMPETICION(IDCOMPETICION, TIPOCOMPETICION, CATEGORIA, FECHAINICIO, FECHAFIN)
      values(sec_competicion.nextval, c_Tipocompeticion, c_Categoria, c_FechaInicio, c_FechaFin);
commit work;
end;
--7. ProcedureCuotas.sql
create or replace procedure nueva_cuota(
       c_dni in Cuotas.DNI%type,
       c_pagocuota in Cuotas.PagoCuota%type,
       c_fechapago in Cuotas.FechaPago%type)
is begin
   insert into Cuotas values(sec_cuota.nextval, c_dni, c_pagocuota, c_fechapago);
   commit work;
end;
--8. ProcedureEquipo.sql
create or replace procedure nuevo_equipo(
     n_categoria in equipo.categoria%type,
     n_liga      in equipo.liga%type)
is
```

```
begin


  insert into equipo values (sec_equipo.nextval,n_categoria,n_liga);


commit work;

end;

--9. ProcedureJugadora.sql

create or replace procedure nueva_jugadora(

     n_dni            in persona.dni%type,

     n_nombre         in persona.nombre%type,

     n_apellidos      in persona.apellidos%type,

     n_direccion      in persona.direccion%type,

     n_fechanacimiento in persona.fechanacimiento%type,

     n_dorsal         in jugadora.dorsal%type,

     n_peso           in jugadora.peso%type,

     n_posicion       in jugadora.posicion%type,

     n_manohabil      in jugadora.manohabil%type,

     n_idequipo       in jugadora.idequipo%type)

IS


BEGIN


  insert into persona values(n_dni,n_nombre,n_apellidos,n_direccion,n_fechanacimiento);


  if(to_number(to_char(n_fechanacimiento,'yy'))-to_number(to_char(sysdate,'yy')) > 17) then insert into jugadora
values(n_dni,n_dorsal,n_peso,n_posicion,n_manohabil,'senior',n_idequipo);

  else

     if(to_number(to_char(n_fechanacimiento,'yy'))-to_number(to_char(sysdate,'yy')) > 15) then insert into jugadora
values(n_dni,n_dorsal,n_peso,n_posicion,n_manohabil,'juvenil',n_idequipo);

     else if(to_number(to_char(n_fechanacimiento,'yy'))-to_number(to_char(sysdate,'yy')) > 13) then insert into
jugadora values(n_dni,n_dorsal,n_peso,n_posicion,n_manohabil,'cadete',n_idequipo);

          else if(to_number(to_char(n_fechanacimiento,'yy'))-to_number(to_char(sysdate,'yy')) > 11) then insert into
jugadora values(n_dni,n_dorsal,n_peso,n_posicion,n_manohabil,'infantil',n_idequipo);
```

```
            else if(to_number(to_char(n_fechanacimiento,'yy'))-to_number(to_char(sysdate,'yy')) > 9) then insert into
jugadora values(n_dni,n_dorsal,n_peso,n_posicion,n_manohabil,'alevin',n_idequipo);

                else insert into jugadora values(n_dni,n_dorsal,n_peso,n_posicion,n_manohabil,'benjamin',n_idequipo);

                end if;

            end if;

         end if;

     end if;

  end if;


  commit work;


END;

--10. ProcedureLesion.sql

create or replace procedure nueva_lesion(

        l_DNI in Lesion.DNI%type,

        l_Tipo in Lesion.Tipo%type,

        l_FechaInicio in Lesion.FechaInicio%type,

        l_FechaFin in Lesion.FechaFin%type,

        l_Recaida in Lesion.Recaida%type)


is

begin

  insert into LESION(IDLESION,DNI, TIPO, FECHAINICIO, FECHAFIN, RECAIDA)

        values(sec_lesion.nextval,l_DNI, l_Tipo, l_FechaInicio, l_FechaFin, l_Recaida);

commit work;

end;

--11. Procedurelugarpartido.sql

CREATE OR REPLACE PROCEDURE nuevo_lugarpartido(


    P_LUGAR             in LUGARPARTIDO.LUGAR%type,

    P_DESPLAZAMIENTO    in LUGARPARTIDO.DESPLAZAMIENTO%type)
```

```
IS BEGIN


  insert into LUGARPARTIDO values(P_LUGAR,P_DESPLAZAMIENTO,(P_desplazamiento*0.2+15));


  commit work;
end;
--12. Proceduremmaterial.sql
CREATE OR REPLACE PROCEDURE nuevo_material(

     m_stock             IN material.stock%TYPE,
     m_proveedor         IN material.proveedor%TYPE,
     m_tipo              in material.tipo%type
     )
IS BEGIN


  INSERT INTO material VALUES(sec_material.nextval,m_stock,m_proveedor,m_tipo);
  commit work;
end;
--12. Proceduremmaterial.sql
create or replace procedure Nuevo_Oficial(
    p_DNI in Persona.DNI%type,
    p_Nombre in Persona.Nombre%type,
    p_Apellidos in Persona.Apellidos%type,
    P_DIRECCION in PERSONA.DIRECCION%type,
    p_fechanacimiento in PERSONA.FECHANACIMIENTO %type,
    o_Sueldo in Oficial.Sueldo%type,
    o_Titulacion in Oficial.Titulacion%type,
```

```
   o_Categoria in Oficial.Categoria%type)
is
begin
   insert into PERSONA(DNI, NOMBRE, APELLIDOS, DIRECCION, FECHANACIMIENTO)
       values(p_DNI, p_Nombre, p_Apellidos, p_Direccion, p_fechanacimiento);
   insert into Oficial(DNI, Sueldo, Titulacion, Categoria)
       values(p_DNI, o_Sueldo, o_Titulacion, o_Categoria);
commit work;
end;

   --14. Procedureoficialequipo.sql

CREATE OR REPLACE PROCEDURE nuevo_oficialequipo(

    oe_dni              IN oficialequipo.dni%TYPE,
    oe_equipo           IN oficialequipo.idequipo%TYPE
    )

IS BEGIN


 INSERT INTO oficialequipo VALUES(oe_dni,oe_equipo);



 commit work;
end;
--15. ProcedurePartido.sql
CREATE OR REPLACE PROCEDURE nuevo_partido(
    P_EQUIPO            in PARTIDO.IDEQUIPO%type,
    p_competicion       in partido.idcompeticion%type,
    p_fecha             IN partido.fecha%TYPE,
```

```
    p_rival             IN partido.equiporival%TYPE,
    p_emplazamiento     IN partido.emplazamiento%TYPE,
    p_gf                IN partido.golesfavor%TYPE,
    p_categoria         IN partido.categoria%TYPE,
    p_gc                IN partido.golescontra%TYPE,
    p_lugar             IN partido.lugar%TYPE,
    p_tipopartido       IN partido.tipopartido%type)
is begin

 insert into PARTIDO values(P_EQUIPO,
 p_competicion, p_fecha,p_rival,p_emplazamiento,p_categoria,p_gf,p_gc,p_lugar,p_tipopartido );

 COMMIT WORK;

end;
--16. Procedurepista.sql
create or replace procedure NUEVO_PISTA(
    p_Lugar             IN pista.lugar%TYPE
    )
IS BEGIN

 INSERT INTO pista VALUES(sec_pista.nextval,p_lugar);

 commit work;
end;
--17. ProcedureReserva.sql
create or replace procedure nueva_reserva(
    n_DNI       in reservas.dni%type,
    n_HoraInicio  in reservas.horainicio%type,
    N_HORAFIN    in RESERVAS.HORAFIN%type,
    N_IDPISTA    in RESERVAS.IDPISTA%type,
```

```
     n_idmaterial  in reservas.idmaterial%type)

is

begin


   insert into Reservas values(sec_reserva.nextval,n_dni,n_horainicio,n_horafin,N_IDPISTA,n_idmaterial);


commit work;

end;


//PAQUETES DE PRUEBAS//

--1.1 PaqueteAsistencia.sql

CREATE OR REPLACE PACKAGE Pruebas_asistencia AS


  PROCEDURE INICIALIZAR;

  PROCEDURE insertar (nombre_prueba VARCHAR2, n_dni char, n_tipoevento VARCHAR2, n_fechaevento date,
n_falta char, salidaEsperada BOOLEAN);

  PROCEDURE ACTUALIZAR (NOMBRE_PRUEBA VARCHAR2, N_DNI CHAR, N_TIPOEVENTO VARCHAR2,
N_FECHAEVENTO DATE, N_FALTA CHAR, SALIDAESPERADA BOOLEAN);

  PROCEDURE eliminar (nombre_prueba VARCHAR2,n_dni char, salidaEsperada BOOLEAN);


END pruebas_asistencia;

/


CREATE OR REPLACE PACKAGE BODY Pruebas_asistencia AS

 /* INICIALIZACIÓN */

 PROCEDURE inicializar AS

 BEGIN

   /* Borrar contenido de la tabla */

    DELETE FROM asistencia;

   NULL;

 END inicializar;

/* PRUEBA PARA LA INSERCIÓN DE DEPARTAMENTOS */
```

```
  PROCEDURE insertar (nombre_prueba VARCHAR2,n_dni char, n_tipoevento VARCHAR2, n_fechaevento date,
n_falta char, salidaEsperada BOOLEAN) AS

    salida BOOLEAN := true;

    N_ASISTENCIA ASISTENCIA%ROWTYPE;

    n_dnival char;

  BEGIN


    /* Insertar departamento */

    nuevo_asistencia(n_dni, n_tipoevento, n_fechaevento, n_falta);


    /* Seleccionar departamento y comprobar que los datos se insertaron correctamente */

    N_DNIVAL := N_DNI;

    SELECT * INTO N_ASISTENCIA FROM ASISTENCIA WHERE DNI=N_DNIVAL;

    IF (n_asistencia.dni <> n_dni or n_asistencia.tipoevento <> n_tipoevento or n_asistencia.fechaevento <>
n_fechaevento or n_asistencia.falta <> n_falta) THEN

      salida := false;

    END IF;

    COMMIT WORK;


    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END insertar;


  PROCEDURE actualizar (nombre_prueba VARCHAR2, N_DNI CHAR, N_TIPOEVENTO VARCHAR2, N_FECHAEVENTO
DATE, N_FALTA CHAR, salidaEsperada BOOLEAN) AS

    SALIDA BOOLEAN := TRUE;

    n_asistencia asistencia%ROWTYPE;

  BEGIN
```

```
  UPDATE asistencia SET tipoevento=n_tipoevento, fechaevento=n_fechaevento, FALTA=n_falta WHERE dni=n_dni;



  SELECT * INTO N_ASISTENCIA FROM ASISTENCIA WHERE DNI=N_DNI;

  IF (n_asistencia.tipoevento<>n_tipoevento or n_asistencia.fechaevento<>n_fechaevento or n_asistencia.falta <>
n_falta) THEN

    salida := false;

  END IF;

  COMMIT WORK;


  /* Mostrar resultado de la prueba */

  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


  EXCEPTION

  WHEN OTHERS THEN

      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

      ROLLBACK;

END actualizar;



PROCEDURE eliminar (nombre_prueba VARCHAR2, n_dni char, salidaEsperada BOOLEAN) AS

 SALIDA BOOLEAN := TRUE;

 n_dn char;

BEGIN


  DELETE FROM asistencia WHERE dni=n_dni;


  SELECT COUNT(*) INTO n_dn FROM asistencia WHERE dni=n_dni;

  IF (n_dn <> 0) THEN

    salida := false;
```

```
   END IF;
   COMMIT WORK;

   /* Mostrar resultado de la prueba */
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

   EXCEPTION
   WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK;
 END eliminar;

END PRUEBAS_asistencia;
--2.1 PaqueteClasificacion.sql
CREATE OR REPLACE PACKAGE Pruebas_clasificacion AS

  PROCEDURE INICIALIZAR;
  PROCEDURE INSERTAR (NOMBRE_PRUEBA VARCHAR2,n_idequipo varchar2,n_idcompeticion varchar2,
N_PUNTOS NUMBER, SALIDAESPERADA BOOLEAN);
  PROCEDURE ACTUALIZAR (NOMBRE_PRUEBA VARCHAR2,n_idequipo varchar2,n_idcompeticion
varchar2,n_puntos number, SALIDAESPERADA BOOLEAN);
  PROCEDURE eliminar (nombre_prueba VARCHAR2,n_idequipo varchar2,n_idcompeticion varchar2, salidaEsperada
BOOLEAN);

END pruebas_clasificacion;
/

CREATE OR REPLACE PACKAGE BODY Pruebas_clasificacion AS
 /* INICIALIZACIÓN */
 PROCEDURE inicializar AS
 BEGIN
   /* Borrar contenido de la tabla */
```

```
    DELETE FROM clasificacion;

   NULL;

  END inicializar;


   PROCEDURE insertar (nombre_prueba VARCHAR2,n_idequipo varchar2,n_idcompeticion varchar2,n_puntos
number, salidaEsperada BOOLEAN) AS

   SALIDA BOOLEAN := TRUE;

   N_CLASIFICACION CLASIFICACION%ROWTYPE;

   N_IDEQ VARCHAR2;

   n_idclas varchar2;

  BEGIN




   nueva_clasificacion(n_puntos);




   N_IDEQ := N_IDEQUIPO;

   N_IDCLAS := N_IDCOMPETICION;

   SELECT * INTO N_CLASIFICACION FROM CLASIFICACION WHERE IDEQUIPO=N_IDEQ AND
IDCOMPETICION=N_IDCLAS;

   IF (n_clasificacion.idequipo <> n_idequipo or n_clasificacion.idcompeticion <> n_idcompeticion or
n_clasificacion.puntos <> n_puntos) THEN

     salida := false;

   END IF;

   COMMIT WORK;


   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;
```

```
 END insertar;



 PROCEDURE actualizar (nombre_prueba VARCHAR2,n_idequipo varchar2,n_idcompeticion varchar2,n_puntos
number, salidaEsperada BOOLEAN) AS

  SALIDA BOOLEAN := TRUE;

  n_clasificacion clasificacion%ROWTYPE;

 BEGIN


  /

  UPDATE clasificacion SET  puntos=n_puntos WHERE idequipo=n_idequipo and idcompeticion=n_idcompeticion;



  SELECT * INTO N_CLASIFICACION FROM CLASIFICACION WHERE IDEQUIPO=N_IDEQUIPO AND
IDCOMPETICION=N_IDCOMPETICION;

  IF (n_clasificacion.puntos<>n_puntos) THEN

    salida := false;

  END IF;

  COMMIT WORK;


  /* Mostrar resultado de la prueba */

  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


  EXCEPTION

  WHEN OTHERS THEN

      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

      ROLLBACK;

 END ACTUALIZAR;
```

```
  PROCEDURE eliminar (nombre_prueba VARCHAR2, n_idequipo varchar2,n_idcompeticion varchar2, salidaEsperada
BOOLEAN) AS

   SALIDA BOOLEAN := TRUE;

   n_dn number;

 BEGIN



   DELETE FROM clasificacion WHERE (idequipo=n_idequipo) and (idcompeticion_idcompeticion);



   SELECT COUNT(*) INTO n_dn FROM clasificacion WHERE idequipo=n_idequipo and
idcompeticion=n_idcompeticion;

   IF (n_dn <> 0) THEN

     salida := false;

   END IF;

   COMMIT WORK;


   /* Mostrar resultado de la prueba */

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;

 END eliminar;


END PRUEBAS_clasificacion;

--3.1 PaqueteCompeticion.sql

CREATE OR REPLACE PACKAGE Pruebas_Competicion AS


  PROCEDURE inicializar;
```

```
  PROCEDURE insertar (nombre_prueba VARCHAR2, n_tipocompeticion VARCHAR2, n_categoria VARCHAR2,
n_fechainicio date, n_fechafin date, salidaEsperada BOOLEAN);

  PROCEDURE actualizar (nombre_prueba VARCHAR2, n_idcompeticion number, n_tipocompeticion VARCHAR2,
n_categoria VARCHAR2, n_fechainicio date, n_fechafin date, salidaEsperada BOOLEAN);

  PROCEDURE eliminar (nombre_prueba VARCHAR2, n_idcompeticion number, salidaEsperada BOOLEAN);


END pruebas_competicion;

/


CREATE OR REPLACE PACKAGE BODY Pruebas_Competicion AS

 /* INICIALIZACIÓN */

 PROCEDURE inicializar AS

 BEGIN

  /* Borrar contenido de la tabla */

   DELETE FROM competicion;

  NULL;

 END inicializar;


 PROCEDURE insertar (nombre_prueba VARCHAR2, n_tipocompeticion VARCHAR2, n_categoria VARCHAR2,
n_fechainicio date, n_fechafin date, salidaEsperada BOOLEAN) AS

  salida BOOLEAN := true;

  n_competicion competicion%ROWTYPE;

  n_idcom number(38);

 BEGIN



  nueva_competicion(n_tipocompeticion, n_categoria, n_fechainicio, n_fechafin);



  n_idcom := sec_competicion.currval;

  SELECT * INTO n_competicion FROM competicion WHERE idcompeticion=n_idcom;

  IF (n_competicion.tipocompeticion<>n_tipocompeticion or n_competicion.categoria<>n_categoria or
```

```
    n_competicion.fechainicio<>n_fechainicio or n_competicion.fechafin<>n_fechafin) THEN
   salida := false;
  END IF;
  COMMIT WORK;

  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END insertar;


  PROCEDURE actualizar (nombre_prueba VARCHAR2, n_idcompeticion number, n_tipocompeticion VARCHAR2,
n_categoria VARCHAR2, n_fechainicio date, n_fechafin date, salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  n_competicion competicion%ROWTYPE;
 BEGIN


  UPDATE competicion SET tipocompeticion=n_tipocompeticion WHERE idcompeticion=n_idcompeticion;


  SELECT * INTO n_competicion FROM competicion WHERE idcompeticion=n_idcompeticion;
  IF (n_competicion.tipocompeticion<>n_tipocompeticion) THEN
   salida := false;
  END IF;
  COMMIT WORK;

  /* Mostrar resultado de la prueba */
```

```
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END actualizar;




PROCEDURE eliminar (nombre_prueba VARCHAR2, n_idcompeticion number, salidaEsperada BOOLEAN) AS
 salida BOOLEAN := true;
 n_competicion INTEGER;
BEGIN



 DELETE FROM competicion WHERE idcompeticion=n_idcompeticion;



 SELECT COUNT(*) INTO n_competicion FROM competicion WHERE idcompeticion=n_idcompeticion;
 IF (n_competicion <> 0) THEN
   salida := false;
 END IF;
 COMMIT WORK;

 /* Mostrar resultado de la prueba */
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
```

```
        ROLLBACK;

  END eliminar;


END PRUEBAS_competicion;

--4.1 PaqueteCuotas.sql

CREATE OR REPLACE PACKAGE Prueba_Cuotas AS


   PROCEDURE inicializar;

   PROCEDURE insertar (nombre_prueba VARCHAR2, pr_dni varchar2, pr_pagocuota char, pr_fechapago date,
salidaEsperada BOOLEAN);

   PROCEDURE actualizar (nombre_prueba VARCHAR2,pr_idcuota integer, pr_dni varchar2, pr_pagocuota char,
pr_fechapago date, salidaEsperada BOOLEAN);

   PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_idcuota integer,salidaEsperada BOOLEAN);


END Prueba_Cuotas;

/


 CREATE OR REPLACE PACKAGE BODY Prueba_Cuotas AS


  --inicializacion

  PROCEDURE inicializar AS

  BEGIN



     DELETE FROM Cuotas;


    NULL;

  END inicializar;


  --insercion

  PROCEDURE insertar (nombre_prueba VARCHAR2, pr_dni varchar2, pr_pagocuota char, pr_fechapago date,
salidaEsperada BOOLEAN) AS
```

```
  salida BOOLEAN := true;
  cuo Cuotas%ROWTYPE;
  pr_idcuota integer;
BEGIN


  nueva_cuota(pr_dni, pr_pagocuota, pr_fechapago);


  pr_idcuota:=sec_cuota.currval;
  SELECT * INTO cuo FROM Cuotas WHERE idcuota=pr_idcuota;
  IF (cuo.dni<>pr_dni or cuo.pagocuota<>pr_pagocuota or cuo.fechapago<>pr_fechapago) THEN
    salida := false;
  END IF;
  COMMIT WORK;


  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
END insertar;

--actualizacion
 PROCEDURE actualizar (nombre_prueba VARCHAR2,pr_idcuota integer, pr_dni varchar2, pr_pagocuota char,
pr_fechapago date, salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  cuo Cuotas%ROWTYPE;
```

```
BEGIN



 UPDATE Cuotas SET dni=pr_dni, pagocuota=pr_pagocuota, fechapago=pr_fechapago WHERE idcuota=pr_idcuota;



 SELECT * INTO cuo FROM Cuotas WHERE idcuota=pr_idcuota;
 IF (cuo.dni<>pr_dni or cuo.pagocuota<>pr_pagocuota or cuo.fechapago<>pr_fechapago) THEN
  salida := false;
 END IF;
 COMMIT WORK;

 -- Mostrar resultado de la prueba
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END actualizar;



--ELIMINACIÓN
PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_idcuota integer,salidaEsperada BOOLEAN) AS
 salida BOOLEAN := true;
  n_cuotas INTEGER;
BEGIN



 DELETE FROM Cuotas WHERE idcuota=pr_idcuota;
```

```
    SELECT COUNT(*) INTO n_cuotas FROM Cuotas WHERE idcuota=pr_idcuota;

    IF (n_cuotas <> 0) THEN

      salida := false;

    END IF;

    COMMIT WORK;


    -- Mostrar resultado de la prueba

    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END eliminar;


END Prueba_Cuotas;

--5.1 PaqueteEquipo.sql

CREATE OR REPLACE PACKAGE PRUEBAS_EQUIPO AS


   PROCEDURE INICIALIZAR;

   PROCEDURE INSERTAR (NOMBRE_PRUEBA VARCHAR2, W_CATEGORIA VARCHAR2,W_LIGA
VARCHAR2,SALIDAESPERADA BOOLEAN);

   PROCEDURE ACTUALIZAR (NOMBRE_PRUEBA VARCHAR2,W_IDEQUIPO integer, W_CATEGORIA
VARCHAR2,W_LIGA varchar2, SALIDAESPERADA BOOLEAN);

   PROCEDURE eliminar (nombre_prueba VARCHAR2,w_idequipo INTEGER, salidaEsperada BOOLEAN);


END PRUEBAS_EQUIPO;

/


  CREATE OR REPLACE PACKAGE BODY PRUEBAS_equipo AS
```

```
/* INICIALIZACIÓN */

PROCEDURE inicializar AS

BEGIN


  /* Borrar contenido de la tabla */

   DELETE FROM equipo;


  NULL;

END inicializar;



  PROCEDURE insertar (NOMBRE_PRUEBA VARCHAR2, W_CATEGORIA VARCHAR2,W_LIGA
varchar2,SALIDAESPERADA BOOLEAN) AS

   SALIDA BOOLEAN := TRUE;

   n_equipo equipo%ROWTYPE;

   w_idequipo INTEGER;

  BEGIN



   nuevo_equipo(W_CATEGORIA,W_liga);


   W_IDEQUIPO := SEC_EQUIPO.CURRVAL;

   SELECT * INTO N_EQUIPO FROM EQUIPO WHERE IDEQUIPO=W_IDEQUIPO;

   IF (n_equipo.categoria <> w_categoria or n_equipo.liga <> w_liga) THEN

     salida := false;

   END IF;

   COMMIT WORK;


   /* Mostrar resultado de la prueba */

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
```

```
  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END insertar;


  PROCEDURE actualizar (nombre_prueba VARCHAR2,w_idequipo integer,W_CATEGORIA VARCHAR2,W_LIGA
varchar2, salidaEsperada BOOLEAN) AS
   SALIDA BOOLEAN := TRUE;
   equipos equipo%ROWTYPE;
 BEGIN


   UPDATE equipo SET categoria=W_CATEGORIA, liga=w_liga WHERE idequipo=w_idequipo;


   SELECT * INTO EQUIPOS FROM EQUIPO WHERE IDEQUIPO=W_IDEQUIPO;
   IF (equipos.categoria <> w_categoria or equipos.liga <> w_liga) THEN
    salida := false;
   END IF;
   COMMIT WORK;

   /* Mostrar resultado de la prueba */
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

   EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END actualizar;
```

```
 PROCEDURE eliminar (nombre_prueba VARCHAR2,w_idequipo INTEGER, salidaEsperada BOOLEAN) AS
  SALIDA BOOLEAN := TRUE;
  n_equipo INTEGER;
 BEGIN



  DELETE FROM equipo WHERE idequipo=w_idequipo;



  SELECT COUNT(*) INTO N_EQUIPO FROM EQUIPO WHERE IDEQUIPO=W_IDEQUIPO;
  IF (n_equipo <> 0) THEN
    salida := false;
  END IF;
  COMMIT WORK;

  /* Mostrar resultado de la prueba */
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END eliminar;

END PRUEBAS_equipo;
/
--6.1 PaqueteJugadora.sql
CREATE OR REPLACE PACKAGE Pruebas_jugadora AS
```

```
  PROCEDURE INICIALIZAR;

  PROCEDURE INSERTAR (NOMBRE_PRUEBA VARCHAR2, N_DNI CHAR,N_DORSAL NUMBER,N_PESO
NUMBER,N_POSICION VARCHAR2,N_MANOHABIL VARCHAR2,N_CATEGORIA VARCHAR2,IDEQUIPO INTEGER,
SALIDAESPERADA BOOLEAN);

  PROCEDURE ACTUALIZAR (NOMBRE_PRUEBA VARCHAR2, n_dni char,n_dorsaL number,n_peso
number,n_posicion varchar2,n_manohabil varchar2,n_categoria varchar2,idequipo integer, SALIDAESPERADA
BOOLEAN);

  PROCEDURE eliminar (nombre_prueba VARCHAR2, n_dni char, salidaEsperada BOOLEAN);


END pruebas_jugadora;

/


CREATE OR REPLACE PACKAGE BODY Pruebas_jugadora AS

 /* INICIALIZACIÓN */

 PROCEDURE inicializar AS

 BEGIN

  /* Borrar contenido de la tabla */

   DELETE FROM jugadora;

  NULL;

 END inicializar;


  PROCEDURE insertar (nombre_prueba VARCHAR2,n_dni char,n_dorsal number,n_peso number,n_posicion
varchar2,n_manohabil varchar2,n_categoria varchar2,idequipo integer, salidaEsperada BOOLEAN) AS

   SALIDA BOOLEAN := TRUE;

   N_JUGADORA JUGADORA%ROWTYPE;

   n_dn char;

 BEGIN



  INSERT INTO JUGADORA VALUES(N_DNI CHAR,N_DORSAL NUMBER,N_PESO NUMBER,N_POSICION
VARCHAR2,N_MANOHABIL VARCHAR2,N_CATEGORIA VARCHAR2,IDEQUIPO INTEGER);
```

```
   N_DN := JUGADORA.DNI;

   SELECT * INTO N_JUGADORA FROM JUGADORA WHERE DNI=N_DN;

   IF (n_jugadora.dni <> n_dni or N_JUGADORA.DORSAL <> N_DORSAL OR n_jugadora.peso <> n_peso OR
n_jugadora.POSICION <> n_POSICION OR n_jugadora.MANOHABIL <> n_MANOHABIL OR n_jugadora.CATEGORIA <>
n_CATEGORIA OR n_jugadora.IDEQUIPO <> n_IDEQUIPO) THEN

     salida := false;

   END IF;

   COMMIT WORK;



   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));



   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;

 END insertar;




  PROCEDURE actualizar (nombre_prueba VARCHAR2, n_dni char,n_dorsaL number,n_peso number,n_posicion
varchar2,n_manohabil varchar2,n_categoria varchar2,idequipo integer, salidaEsperada BOOLEAN) AS

   SALIDA BOOLEAN := TRUE;

   n_JUGADORA JUGADORA%ROWTYPE;

  BEGIN




   UPDATE JUGADORA SET
DORSAL=n_DORSAL,PESO=n_PESO,POSICION=N_POSICION,MANOHABIL=n_MANOHABIL,CATEGORIA=N_CATEGORI
A,IDEQUIPO=n_IDEQUIPO WHERE idpista=n_idpista;




   SELECT * INTO N_JUGADORA FROM JUGADORA WHERE DNI=N_DNI;

   IF (N_JUGADORA.DORSAL <> n_DORSAL OR N_JUGADORA.PESO <> n_PESO OR N_JUGADORA.POSICION <>
N_POSICION OR N_JUGADORA.MANOHABIL <> n_MANOHABIL OR N_JUGADORA.CATEGORIA <> N_CATEGORIA OR
N_JUGADORA.IDEQUIPO <> n_IDEQUIPO) THEN
```

```
  salida := false;
 END IF;
 COMMIT WORK;

 /* Mostrar resultado de la prueba */
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END actualizar;



PROCEDURE eliminar (nombre_prueba VARCHAR2, n_DNI number, salidaEsperada BOOLEAN) AS
 SALIDA BOOLEAN := TRUE;
 N_DN CHAR;
BEGIN


 DELETE FROM JUGADORA WHERE DNI=n_DNI;



 SELECT COUNT(*) INTO N_DN FROM JUGADORA WHERE DNI=N_DNI;
 IF (n_DN <> 0) THEN
  salida := false;
 END IF;
 COMMIT WORK;

 /* Mostrar resultado de la prueba */
```

```
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END eliminar;


END PRUEBAS_JUGADORA;

--7.1 PaqueteLesion.sql

create or replace PACKAGE Pruebas_lesion AS


   PROCEDURE inicializar;

   PROCEDURE insertar (nombre_prueba VARCHAR2, n_dni char, n_tipo VARCHAR2, n_fechainicio date, n_fechafin
date ,salidaEsperada BOOLEAN);

   PROCEDURE actualizar (nombre_prueba VARCHAR2,  n_fechafin date, salidaEsperada BOOLEAN);

   PROCEDURE eliminar (nombre_prueba VARCHAR2, salidaEsperada BOOLEAN);


END pruebas_lesion;

/


create or replace PACKAGE BODY Pruebas_lesion AS

  /* INICIALIZACIÓN */

  PROCEDURE inicializar AS

  BEGIN

    /* Borrar contenido de la tabla */

     DELETE FROM lesion;

    NULL;

  END inicializar;


  PROCEDURE insertar (nombre_prueba VARCHAR2, n_dni char, n_tipo VARCHAR2, n_fechainicio date, n_fechafin
date ,salidaEsperada BOOLEAN) AS
```

```
  salida BOOLEAN := true;
  n_lesion lesion%ROWTYPE;
  ind number;
BEGIN



  nueva_lesion (n_dni , n_tipo , n_fechainicio, n_fechafin, null);
  ind:=sec_lesion.currval;


  SELECT * INTO n_lesion FROM lesion WHERE idlesion=ind;
  IF (n_lesion.dni<>n_dni or n_lesion.tipo<>n_tipo or
     n_lesion.fechainicio<>n_fechainicio or n_lesion.fechainicio<>n_fechainicio) THEN
   salida := false;
  END IF;
  COMMIT WORK;


  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


  EXCEPTION
  WHEN OTHERS THEN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
       ROLLBACK;
END insertar;



PROCEDURE actualizar (nombre_prueba VARCHAR2,  n_fechafin date, salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  n_lesion lesion%ROWTYPE;
  ind number;
BEGIN
  ind:=sec_lesion.currval;
```

```
  UPDATE lesion SET fechafin=n_fechafin WHERE idlesion=ind;



  SELECT * INTO n_lesion FROM lesion WHERE idlesion=ind;
  IF (n_lesion.fechafin<>n_fechafin) THEN
    salida := false;
  END IF;
  COMMIT WORK;

  /* Mostrar resultado de la prueba */
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
END actualizar;



PROCEDURE eliminar (nombre_prueba VARCHAR2, salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  n_lesion INTEGER;
  ind number;
BEGIN
  ind:=sec_lesion.currval;

  DELETE FROM lesion WHERE idlesion=ind;
```

```
   SELECT COUNT(*) INTO n_lesion FROM lesion WHERE idlesion=ind;

   IF (n_lesion <> 0) THEN

     salida := false;

   END IF;

   COMMIT WORK;


   /* Mostrar resultado de la prueba */

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

 END eliminar;


END pruebas_lesion;

--8.1 PaqueteLugarPartido.sql

CREATE OR REPLACE PACKAGE Prueba_LugarPartido AS


   PROCEDURE inicializar;

   PROCEDURE insertar (nombre_prueba VARCHAR2, pr_lugar VARCHAR2, pr_desplazamiento
integer,salidaEsperada BOOLEAN);

   PROCEDURE actualizar (nombre_prueba VARCHAR2, pr_lugar VARCHAR2, pr_desplazamiento
integer,salidaEsperada BOOLEAN);

   PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_lugar VARCHAR2,salidaEsperada BOOLEAN);


END Prueba_LugarPartido;

/


 CREATE OR REPLACE PACKAGE BODY Prueba_LugarPartido AS


  --inicializacion
```

```
PROCEDURE inicializar AS

BEGIN



  DELETE FROM LugarPartido;



 NULL;

END inicializar;



--insercion

 PROCEDURE insertar (nombre_prueba VARCHAR2, pr_lugar VARCHAR2, pr_desplazamiento integer,salidaEsperada
BOOLEAN) AS

  salida BOOLEAN := true;

  lugpar LugarPartido%ROWTYPE;

BEGIN



  nuevo_lugarpartido(pr_lugar, pr_desplazamiento);



  SELECT * INTO lugpar FROM LugarPartido WHERE Lugar=pr_lugar;

  IF (lugpar.desplazamiento<>pr_desplazamiento or lugpar.costearbitraje<>(pr_desplazamiento*0.2+15)) THEN

    salida := false;

  END IF;

  COMMIT WORK;



  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));



  EXCEPTION
```

```
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END insertar;

 --actualizacion
 PROCEDURE actualizar (nombre_prueba VARCHAR2, pr_lugar VARCHAR2, pr_desplazamiento
integer,salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  lugpar LugarPartido%ROWTYPE;

 BEGIN

  UPDATE LugarPartido SET desplazamiento=pr_desplazamiento WHERE lugar=pr_lugar;

  SELECT * INTO lugpar FROM LugarPartido WHERE lugar=pr_lugar;
  IF (lugpar.desplazamiento<>pr_desplazamiento or lugpar.costearbitraje<>(pr_desplazamiento*0.2+15)) THEN
   salida := false;
  END IF;
  COMMIT WORK;
  COMMIT WORK;

  -- Mostrar resultado de la prueba
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END actualizar;
```

```
 --ELIMINACIÓN
 PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_lugar VARCHAR2,salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
   n_lugar INTEGER;
 BEGIN



  DELETE FROM LugarPartido WHERE lugar=pr_lugar;



  SELECT COUNT(*) INTO n_lugar FROM LugarPartido WHERE lugar=pr_lugar;
  IF (n_lugar <> 0) THEN
   salida := false;
  END IF;
  COMMIT WORK;


  -- Mostrar resultado de la prueba
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END eliminar;


END Prueba_LugarPartido;
--9.1 PaqueteMaterial.sql
create or replace PACKAGE Pruebas_material AS
```

```
  PROCEDURE inicializar;

  PROCEDURE insertar (nombre_prueba VARCHAR2,n_stock number, n_proveedor varchar2, n_tipo varchar2
,salidaEsperada BOOLEAN);

  PROCEDURE actualizar (nombre_prueba VARCHAR2, n_stock VARCHAR2, salidaEsperada BOOLEAN);

  PROCEDURE eliminar (nombre_prueba VARCHAR2, salidaEsperada BOOLEAN);


END pruebas_material;

/

create or replace PACKAGE BODY Pruebas_material AS

 /* INICIALIZACIÓN */

 PROCEDURE inicializar AS

 BEGIN

  /* Borrar contenido de la tabla */

   DELETE FROM material;

  NULL;

 END inicializar;


 PROCEDURE insertar (nombre_prueba VARCHAR2,n_stock number, n_proveedor varchar2, n_tipo varchar2
,salidaEsperada BOOLEAN) AS

  salida BOOLEAN := true;

  n_material material%ROWTYPE;

  ind number;

 BEGIN



  nuevo_material(n_stock,n_proveedor,n_tipo);

  ind :=sec_material.currval;


  SELECT * INTO n_material FROM material WHERE idmaterial=ind;

  IF (n_material.stock<>n_stock or n_material.proveedor<>n_proveedor or

     n_material.tipo<>n_tipo) THEN

   salida := false;
```

```
 END IF;
 COMMIT WORK;

 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END insertar;


PROCEDURE actualizar (nombre_prueba VARCHAR2, n_stock VARCHAR2, salidaEsperada BOOLEAN) AS
 salida BOOLEAN := true;
 n_material material%ROWTYPE;
 ind number;
BEGIN

 ind:= sec_material.currval;

 UPDATE material SET stock=n_stock WHERE idmaterial=ind;


 SELECT * INTO n_material FROM material WHERE idmaterial=ind;
 IF (n_material.stock<>n_stock) THEN
  salida := false;
 END IF;
 COMMIT WORK;

 /* Mostrar resultado de la prueba */
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
```

```
 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END actualizar;



PROCEDURE eliminar (nombre_prueba VARCHAR2, salidaEsperada BOOLEAN) AS
 salida BOOLEAN := true;
 n_material INTEGER;
 ind number;
BEGIN

 ind:= sec_material.currval;

 DELETE FROM material WHERE idmaterial=ind;



 SELECT COUNT(*) INTO n_material FROM material WHERE idmaterial=ind;
 IF (n_material <> 0) THEN
   salida := false;
 END IF;
 COMMIT WORK;

 /* Mostrar resultado de la prueba */
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

 EXCEPTION
 WHEN OTHERS THEN
```

```
        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END eliminar;


END pruebas_material;

--10.1 PaqueteOficial.sql

CREATE OR REPLACE PACKAGE Prueba_Oficial AS


   PROCEDURE inicializar;

   PROCEDURE insertar (nombre_prueba VARCHAR2, pr_DNI char,pr_Nombre varchar2,pr_Apellidos
varchar2,pr_Direccion varchar2,pr_fechanacimiento date,pr_Sueldo number,pr_Titulacion varchar2, pr_Categoria
varchar2,salidaEsperada BOOLEAN);

   PROCEDURE actualizar (nombre_prueba VARCHAR2, pr_DNI char,pr_Nombre varchar2,pr_Apellidos
varchar2,pr_Direccion varchar2,pr_fechanacimiento date,pr_Sueldo number,pr_Titulacion varchar2, pr_Categoria
varchar2,salidaEsperada BOOLEAN);

   PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_DNI CHAR,salidaEsperada BOOLEAN);


END Prueba_Oficial;

/


 CREATE OR REPLACE PACKAGE BODY Prueba_Oficial AS


  --inicializacion

  PROCEDURE inicializar AS

  BEGIN



    DELETE FROM Oficial;


   NULL;

  END inicializar;


  --insercion
```

```
  PROCEDURE insertar (nombre_prueba VARCHAR2, pr_DNI char,pr_Nombre varchar2,pr_Apellidos
varchar2,pr_Direccion varchar2,pr_fechanacimiento date,pr_Sueldo number,pr_Titulacion varchar2, pr_Categoria
varchar2,salidaEsperada BOOLEAN) AS

    salida BOOLEAN := true;

    ofi Oficial%ROWTYPE;

    pers Persona%ROWTYPE;

  BEGIN


    nuevo_oficial(pr_DNI,pr_Nombre,pr_Apellidos, pr_Direccion,pr_fechanacimiento, pr_Sueldo, pr_Titulacion,
pr_Categoria);


    select * into pers from Persona where dni=pr_dni;

    SELECT * INTO ofi FROM oficial WHERE dni=pr_dni;


    IF (pers.nombre<>pr_nombre or pers.apellidos<>pr_apellidos or pers.direccion<>pr_direccion

                or pers.fechanacimiento<>pr_fechanacimiento or ofi.sueldo<>pr_sueldo or
ofi.titulacion<>pr_titulacion

                or ofi.categoria<>pr_categoria) THEN

      salida := false;

    END IF;

    COMMIT WORK;




    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END insertar;


  --actualizacion
```

```
  PROCEDURE actualizar (nombre_prueba VARCHAR2, pr_DNI char,pr_Nombre varchar2,pr_Apellidos
varchar2,pr_Direccion varchar2,pr_fechanacimiento date,pr_Sueldo number,pr_Titulacion varchar2, pr_Categoria
varchar2,salidaEsperada BOOLEAN) AS

   salida BOOLEAN := true;

   ofi Oficial%ROWTYPE;

   pers Persona%ROWTYPE;

 BEGIN


   --actualizar persona y oficial

   UPDATE Persona SET
nombre=pr_nombre,apellidos=pr_apellidos,direccion=pr_direccion,fechanacimiento=pr_fechanacimiento WHERE
DNI=pr_DNI;

   UPDATE Oficial SET sueldo=pr_sueldo, titulacion=pr_titulacion, categoria=pr_categoria WHERE DNI=pr_DNI;

   -- Seleccionar oficial y persona y comprobar que los campos se actualizaron correctamente

   SELECT * INTO pers FROM Persona WHERE DNI=pr_DNI;

   SELECT * INTO ofi FROM Oficial WHERE DNI=pr_DNI;

   IF (pers.nombre<>pr_nombre or pers.apellidos<>pr_apellidos or pers.direccion<>pr_direccion

                  or pers.fechanacimiento<>pr_fechanacimiento or ofi.sueldo<>pr_sueldo or
ofi.titulacion<>pr_titulacion

                  or ofi.categoria<>pr_categoria) THEN

     salida := false;

   END IF;

   COMMIT WORK;

   COMMIT WORK;


   -- Mostrar resultado de la prueba

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;

 END actualizar;
```

```
 --ELIMINACIÓN
 PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_DNI CHAR,salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
   n_DNIO INTEGER;
   n_DNIP INTEGER;
 BEGIN


  DELETE FROM Persona WHERE DNI=pr_DNI;
  DELETE FROM Oficial WHERE DNI=pr_DNI;


  SELECT COUNT(*) INTO n_DNIO FROM Oficial WHERE DNI=pr_DNI;
  SELECT COUNT(*) INTO n_DNIP FROM Persona WHERE DNI=pr_DNI;
  IF (n_DNIO <> 0 and n_DNIP<>0) THEN
    salida := false;
  END IF;
  COMMIT WORK;


  -- Mostrar resultado de la prueba
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END eliminar;


END Prueba_Oficial;
--11.1 PaqueteOficialEquipo.sql
```

```
CREATE OR REPLACE PACKAGE Prueba_OficialEquipo AS

  PROCEDURE inicializar;
  PROCEDURE insertar (nombre_prueba VARCHAR2, pr_DNI char,pr_equipo integer,salidaEsperada BOOLEAN);
  PROCEDURE actualizar (nombre_prueba VARCHAR2, pr_DNI char,pr_equipo integer,salidaEsperada BOOLEAN);
  PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_DNI char,salidaEsperada BOOLEAN);

END Prueba_OficialEquipo;
/

 CREATE OR REPLACE PACKAGE BODY Prueba_OficialEquipo AS

 --inicializacion
 PROCEDURE inicializar AS
 BEGIN


   DELETE FROM OficialEquipo;

  NULL;
 END inicializar;

 --insercion
 PROCEDURE insertar (nombre_prueba VARCHAR2, pr_DNI char,pr_equipo integer,salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  ofiequ OficialEquipo%ROWTYPE;
 BEGIN


  nuevo_oficialequipo(pr_DNI,pr_equipo);
```

```
  SELECT * INTO ofiequ FROM OficialEquipo WHERE DNI=pr_DNI;
  IF (ofiequ.idequipo<>pr_equipo) THEN
    salida := false;
  END IF;
  COMMIT WORK;


  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
END insertar;

--actualizacion
PROCEDURE actualizar (nombre_prueba VARCHAR2, pr_DNI char,pr_equipo integer,salidaEsperada BOOLEAN) AS
  salida BOOLEAN := true;
  ofiequ OficialEquipo%ROWTYPE;

BEGIN


  UPDATE OficialEquipo SET idequipo=pr_equipo WHERE DNI=pr_DNI;


  SELECT * INTO ofiequ FROM OficialEquipo WHERE DNI=pr_DNI;
  IF (ofiequ.idequipo<>pr_equipo) THEN
    salida := false;
```

```
 END IF;
 COMMIT WORK;


 -- Mostrar resultado de la prueba
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END actualizar;


--ELIMINACIÓN
PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_DNI char,salidaEsperada BOOLEAN) AS
 salida BOOLEAN := true;
  n_oficialequipo INTEGER;
BEGIN


 DELETE FROM OficialEquipo WHERE DNI=pr_DNI;


 SELECT COUNT(*) INTO n_oficialequipo FROM oficialequipo WHERE DNI=pr_DNI;
 IF (n_oficialequipo <> 0) THEN
   salida := false;
 END IF;
 COMMIT WORK;

 -- Mostrar resultado de la prueba
```

```
    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END eliminar;


END Prueba_OficialEquipo;

--12.1 PaquetePartido.sql

create or replace PACKAGE Pruebas_partido AS


    PROCEDURE inicializar;

    PROCEDURE insertar (nombre_prueba VARCHAR2, P_EQUIPO number,

    p_competicion number, p_fecha date ,p_rival varchar2,p_emplazamiento varchar2,

    p_categoria varchar2,p_gf number,p_gc number,p_lugar varchar2,p_tipopartido varchar2,salidaEsperada
BOOLEAN);

    PROCEDURE actualizar (nombre_prueba VARCHAR2,  p_equipo number, p_fecha date, p_gf number,p_gc number
, salidaEsperada BOOLEAN);

    PROCEDURE eliminar (nombre_prueba VARCHAR2, p_equipo number, p_fecha date,salidaEsperada BOOLEAN);


END pruebas_partido;

/


create or replace PACKAGE BODY Pruebas_partido AS

  /* INICIALIZACIÓN */

  PROCEDURE inicializar AS

  BEGIN

    /* Borrar contenido de la tabla */

     DELETE FROM partido;

    NULL;

  END inicializar;
```

```
  PROCEDURE insertar (nombre_prueba VARCHAR2, P_EQUIPO number, p_competicion number, p_fecha date
,p_rival varchar2,p_emplazamiento varchar2,

   p_categoria varchar2,p_gf number,p_gc number,p_lugar varchar2,p_tipopartido varchar2,salidaEsperada
BOOLEAN) AS

   salida BOOLEAN := true;

   n_partido partido%ROWTYPE;

 BEGIN




   nuevo_partido (P_EQUIPO, p_competicion,
p_fecha,p_rival,p_emplazamiento,p_categoria,p_gf,p_gc,p_lugar,p_tipopartido);



   SELECT * INTO n_partido FROM partido WHERE idequipo=p_equipo and fecha=p_fecha;

   IF (n_partido.idequipo<>p_equipo or n_partido.idcompeticion<>p_competicion or

      n_partido.fecha<>p_fecha or n_partido.equiporival<>p_rival or n_partido.emplazamiento<>p_emplazamiento
or

      n_partido.categoria<>p_categoria or n_partido.golesfavor<>p_gf or n_partido.golescontra<>p_gc or
n_partido.tipopartido<>p_tipopartido) THEN

     salida := false;

   END IF;

   COMMIT WORK;



   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));



   EXCEPTION

   WHEN OTHERS THEN

      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

      ROLLBACK;

 END insertar;




  PROCEDURE actualizar (nombre_prueba VARCHAR2,  p_equipo number, p_fecha date, p_gf number,p_gc number ,
salidaEsperada BOOLEAN) AS
```

```
 salida BOOLEAN := true;
 n_partido partido%ROWTYPE;
BEGIN

 UPDATE partido SET golesfavor=p_gf ,golescontra=p_gc WHERE idequipo=p_equipo and fecha=p_fecha;



 SELECT * INTO n_partido FROM partido WHERE idequipo=p_equipo and fecha=p_fecha;
 IF ( n_partido.golesfavor<>p_gf or n_partido.golescontra<>p_gc) THEN
  salida := false;
 END IF;
 COMMIT WORK;

 /* Mostrar resultado de la prueba */
 DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

 EXCEPTION
 WHEN OTHERS THEN
     DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
     ROLLBACK;
END actualizar;



PROCEDURE eliminar (nombre_prueba VARCHAR2, p_equipo number, p_fecha date,salidaEsperada BOOLEAN) AS
 salida BOOLEAN := true;
 n_partido INTEGER;
BEGIN

 DELETE FROM partido WHERE idequipo=p_equipo and fecha=p_fecha;
```

```
   SELECT COUNT(*) INTO n_partido FROM partido WHERE idequipo=p_equipo and fecha=p_fecha;

   IF (n_partido <> 0) THEN

     salida := false;

   END IF;

   COMMIT WORK;


   /* Mostrar resultado de la prueba */

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;

 END eliminar;


END pruebas_partido;

--13.1 PaquetePersona.sql

create or replace PACKAGE Pruebas_Persona AS


  PROCEDURE inicializar;

  PROCEDURE insertar (nombre_prueba VARCHAR2, n_dni VARCHAR2, n_nombre VARCHAR2, n_apellidos date,
n_direccion date, n_fechanacimiento date ,salidaEsperada BOOLEAN);

  PROCEDURE actualizar (nombre_prueba VARCHAR2, n_dni number, n_nombre VARCHAR2, salidaEsperada
BOOLEAN);

  PROCEDURE eliminar (nombre_prueba VARCHAR2, n_dni number, salidaEsperada BOOLEAN);


END pruebas_persona;

/


create or replace PACKAGE BODY Pruebas_persona AS

 /* INICIALIZACIÓN */
```

```
  PROCEDURE inicializar AS

  BEGIN

    /* Borrar contenido de la tabla */

     DELETE FROM persona;

    NULL;

  END inicializar;


  PROCEDURE insertar (nombre_prueba VARCHAR2, n_dni VARCHAR2, n_nombre VARCHAR2, n_apellidos date,
n_direccion date, n_fechanacimiento date ,salidaEsperada BOOLEAN) AS

    salida BOOLEAN := true;

    n_persona persona%ROWTYPE;

  BEGIN



    INSERT INTO persona VALUES(n_dni , n_nombre , n_apellidos , n_direccion , n_fechanacimiento);



    SELECT * INTO n_persona FROM persona WHERE dni=n_dni;

    IF (n_persona.dni<>n_dni or n_persona.nombre<>n_nombre or

        n_persona.apellidos<>n_apellidos or n_persona.direccion<>n_direccion or
n_persona.fechanacimiento<>n_fechanacimiento) THEN

      salida := false;

    END IF;

    COMMIT WORK;


    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END insertar;
```

```
 PROCEDURE actualizar (nombre_prueba VARCHAR2, n_dni number, n_nombre VARCHAR2, salidaEsperada
BOOLEAN) AS

   salida BOOLEAN := true;

   n_persona persona%ROWTYPE;

 BEGIN



   UPDATE persona SET nombre=n_nombre WHERE dni=n_dni;



   SELECT * INTO n_persona FROM persona WHERE dni=n_dni;

   IF (n_persona.nombre<>n_nombre) THEN

     salida := false;

   END IF;

   COMMIT WORK;


   /* Mostrar resultado de la prueba */

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;

 END actualizar;



 PROCEDURE eliminar (nombre_prueba VARCHAR2, n_dni number, salidaEsperada BOOLEAN) AS

   salida BOOLEAN := true;
```

```
  n_persona INTEGER;
 BEGIN



  DELETE FROM persona WHERE dni=n_dni;



  SELECT COUNT(*) INTO n_persona FROM persona WHERE dni=n_dni;
  IF (n_persona <> 0) THEN
    salida := false;
  END IF;
  COMMIT WORK;

  /* Mostrar resultado de la prueba */
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END eliminar;

END pruebas_persona;
--14.1 PaquetePista.sql
CREATE OR REPLACE PACKAGE Pruebas_pista AS

  PROCEDURE INICIALIZAR;
  PROCEDURE INSERTAR (NOMBRE_PRUEBA VARCHAR2, N_LUGAR VARCHAR2, SALIDAESPERADA BOOLEAN);
  PROCEDURE ACTUALIZAR (NOMBRE_PRUEBA VARCHAR2, N_IDPISTA NUMBER, N_LUGAR VARCHAR2,
SALIDAESPERADA BOOLEAN);
  PROCEDURE eliminar (nombre_prueba VARCHAR2, n_idpista number, salidaEsperada BOOLEAN);
```

```
END pruebas_pista;
/

CREATE OR REPLACE PACKAGE BODY Pruebas_pista AS
 /* INICIALIZACIÓN */
 PROCEDURE inicializar AS
 BEGIN
  /* Borrar contenido de la tabla */
   DELETE FROM pista;
  NULL;
 END inicializar;

 PROCEDURE insertar (nombre_prueba VARCHAR2, N_LUGAR VARCHAR2, salidaEsperada BOOLEAN) AS
  SALIDA BOOLEAN := TRUE;
  N_PISTA PISTA%ROWTYPE;
  n_idpist number;
 BEGIN


  INSERT INTO pista VALUES(sec_pista.nextval,n_lugar);


  N_IDPIST := SEC_COMPETICION.CURRVAL;
  SELECT * INTO N_PISTA FROM PISTA WHERE IDPISTA=N_IDPIST;
  IF (n_pista.lugar<>n_lugar) THEN
   salida := false;
  END IF;
  COMMIT WORK;

  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));
```

```
  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END insertar;


  PROCEDURE actualizar (nombre_prueba VARCHAR2,N_IDPISTA number, N_LUGAR VARCHAR2, salidaEsperada
BOOLEAN) AS
  SALIDA BOOLEAN := TRUE;
  n_pista pista%ROWTYPE;
 BEGIN


  UPDATE pista SET lugar=n_lugar WHERE idpista=n_idpista;


  SELECT * INTO N_PISTA FROM PISTA WHERE IDPISTA=N_IDPISTA;
  IF (n_pista.lugar<>n_lugar) THEN
   salida := false;
  END IF;
  COMMIT WORK;

  /* Mostrar resultado de la prueba */
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
```

```
 END actualizar;




 PROCEDURE eliminar (nombre_prueba VARCHAR2, n_idpista number, salidaEsperada BOOLEAN) AS
  SALIDA BOOLEAN := TRUE;
  n_pista number;
 BEGIN



  DELETE FROM pista WHERE idpista=n_idpista;


  /* Verificar que el departamento no se encuentra en la BD */
  SELECT COUNT(*) INTO N_PISTA FROM PISTA WHERE IDPISTA=N_IDPISTA;
  IF (n_idpista <> 0) THEN
    salida := false;
  END IF;
  COMMIT WORK;


  /* Mostrar resultado de la prueba */
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END eliminar;


END PRUEBAS_pista;
--15.1 PaqueteReservas.sql
CREATE OR REPLACE PACKAGE Prueba_Reservas AS
```

```
  PROCEDURE inicializar;

  PROCEDURE insertar (nombre_prueba VARCHAR2, pr_dni CHAR, pr_horaInicio date, pr_horaFin date, pr_idpista
integer, pr_idmaterial integer,salidaEsperada BOOLEAN);

  PROCEDURE actualizar (nombre_prueba VARCHAR2,pr_idreserva integer, pr_dni CHAR, pr_horaInicio date,
pr_horaFin date, pr_idpista integer, pr_idmaterial integer,salidaEsperada BOOLEAN);

  PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_idreserva integer,salidaEsperada BOOLEAN);


END Prueba_Reservas;

/


 CREATE OR REPLACE PACKAGE BODY Prueba_Reservas AS


  --inicializacion

  PROCEDURE inicializar AS

  BEGIN



    DELETE FROM Reservas;


   NULL;

  END inicializar;


  --insercion

  PROCEDURE insertar (nombre_prueba VARCHAR2, pr_dni CHAR, pr_horaInicio date, pr_horaFin date, pr_idpista
integer, pr_idmaterial integer,salidaEsperada BOOLEAN) AS

   salida BOOLEAN := true;

   res Reservas%ROWTYPE;

   pr_idreserva integer;

  BEGIN
```

```
    nueva_reserva(pr_dni,pr_horainicio,pr_horafin,pr_idpista,pr_idmaterial);



    pr_idreserva:=sec_reserva.currval;

    SELECT * INTO res FROM Reservas WHERE idreserva=pr_idreserva;

    IF (res.dni<>pr_dni or res.horainicio<>pr_horainicio or res.horafin<>pr_horafin or res.idpista<>pr_idpista or
res.idmaterial<>pr_idmaterial) THEN

      salida := false;

    END IF;

    COMMIT WORK;



    DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


    EXCEPTION

    WHEN OTHERS THEN

        DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

        ROLLBACK;

  END insertar;


  --actualizacion

  PROCEDURE actualizar (nombre_prueba VARCHAR2,pr_idreserva integer, pr_dni CHAR, pr_horaInicio date,
pr_horaFin date, pr_idpista integer, pr_idmaterial integer,salidaEsperada BOOLEAN) AS

    salida BOOLEAN := true;

    res Reservas%ROWTYPE;


  BEGIN



    UPDATE Reservas SET dni=pr_dni, horainicio=pr_horainicio, horafin=pr_horafin, idpista=pr_idpista,
idmaterial=pr_idmaterial WHERE idreserva=pr_idreserva;
```

```
   SELECT * INTO res FROM Reservas WHERE idpista=pr_idpista;

   IF (res.dni<>pr_dni or res.horainicio<>pr_horainicio or res.horafin<>pr_horafin or res.idpista<>pr_idpista or
res.idmaterial<>pr_idmaterial) THEN

     salida := false;

   END IF;

   COMMIT WORK;



   -- Mostrar resultado de la prueba

   DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));


   EXCEPTION

   WHEN OTHERS THEN

       DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));

       ROLLBACK;

 END actualizar;



 --ELIMINACIÓN

 PROCEDURE eliminar (nombre_prueba VARCHAR2, pr_idreserva integer,salidaEsperada BOOLEAN) AS

   salida BOOLEAN := true;

    n_reservas INTEGER;

 BEGIN



   DELETE FROM Reservas WHERE idreserva=pr_idreserva;



   SELECT COUNT(*) INTO n_reservas FROM Reservas WHERE idreserva=pr_idreserva;

   IF (n_reservas <> 0) THEN
```

```
    salida := false;
  END IF;
  COMMIT WORK;

  -- Mostrar resultado de la prueba
  DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(salida,salidaEsperada));

  EXCEPTION
  WHEN OTHERS THEN
      DBMS_OUTPUT.put_line(nombre_prueba || ':' || ASSERT_EQUALS(false,salidaEsperada));
      ROLLBACK;
 END eliminar;

END Prueba_Reservas;
```

//PRUEBAS//

```
--1.2 PruebaAsistencia.sql
SET SERVEROUTPUT ON;

BEGIN

PRUEBAS_ASISTENCIA.INICIALIZAR;
PRUEBAS_ASISTENCIA.INSERTAR('Prueba 1 - Inserción','11111111a','entrenamiento','22/01/2014','no',TRUE);
PRUEBAS_ASISTENCIA.INSERTAR('Prueba 2 - Prueba de asistencia
null','11111111b','entrenamiento','21/01/2014',NULL,FALSE);
PRUEBAS_ASISTENCIA.ACTUALIZAR('Prueba 3 - Actualización del
asistencia','11111111a','entrenamiento','21/01/2014','si',TRUE);
pruebas_asistencia.ELIMINAR('Prueba 4 - Eliminar pista','11111111a',true);

END;
--2.2 PruebaClasificacion.sql
SET SERVEROUTPUT ON;
```

```
BEGIN


PRUEBAS_CLASIFICACION.INICIALIZAR;

PRUEBAS_CLASIFICACION.INSERTAR('Prueba 1 - Inserción',1,1,23,TRUE);

PRUEBAS_CLASIFICACION.INSERTAR('Prueba 2 - Prueba de puntos null',1,1,NULL,FALSE);

PRUEBAS_CLASIFICACION.ACTUALIZAR('Prueba 3 - Actualización de puntos',1,1,26,TRUE);

pruebas_clasificacion.ELIMINAR('Prueba 4 - Eliminar clasificacion',1,1,true);


END;

--3.2 PruebaCompeticion.sql

SET SERVEROUTPUT ON;


BEGIN


pruebas_competicion.INICIALIZAR;

pruebas_competicion.INSERTAR('Prueba 1 - Inserción','liga','senior','10/01/2014','10/04/2014',true);

pruebas_competicion.INSERTAR('Prueba 2 - Prueba de disparador, fechafin antes de fecha
inicio','liga','senior','10/03/2014','10/02/2014',false);

pruebas_competicion.INSERTAR('Prueba 3 - Inserción con categoria null','liga',null,'10/01/2014','10/04/2014',false);

pruebas_competicion.INSERTAR('Prueba 4 - Inserción con tipocompeticion
null',null,'senior','10/01/2014','10/04/2014',false);

pruebas_competicion.ACTUALIZAR('Prueba 5 - Actualización del
tipocompeticion',sec_competicion.currval,'amistoso',false);

pruebas_competicion.ACTUALIZAR('Prueba 6 - Actualización del tipocompeticion a
null',sec_competicion.currval,null,false);

pruebas_competicion.ELIMINAR('Prueba 7 - Eliminar competicion',sec_competicion.currval,true);


END;

--4.2 PruebaCuotas.sql

SET SERVEROUTPUT ON;


BEGIN
```

```
 /**********************************************************************

     PRUEBAS DE LAS OPERACIONES SOBRE LA TABLA Cuotas

 **********************************************************************/

 Prueba_Cuotas.INICIALIZAR;

 Prueba_Cuotas.INSERTAR('Prueba 1 - Inserción cuota correcta', '49091675s', 'si', '12/5/2012', true);

 Prueba_Cuotas.INSERTAR('Prueba 2 - Inserción cuota con persona inexistente', '49091075s', 'si', '12/5/2012',
false);

 Prueba_Cuotas.ACTUALIZAR('Prueba 3 - Actualizar cuota correcta','10', '49091675s', 'no', '12/6/2012', true);

 Prueba_Cuotas.ACTUALIZAR('Prueba 4 - Actualizar cuota inexistente', '150', '49091675s', 'no', '12/6/2012', false);

 Prueba_Cuotas.ELIMINAR('Prueba 5 - Eliminar cuota', '10', true);

 Prueba_Cuotas.ELIMINAR('Prueba 5 - Eliminar cuota inexistente', '1500', false);


END;

--5.2 PruebaEquipo.sql

SET SERVEROUTPUT ON;

BEGIN

PRUEBAS_EQUIPO.INICIALIZAR;

PRUEBAS_EQUIPO.INSERTAR('Prueba 1 - Inserción Equipo','senior','local',TRUE);

PRUEBAS_EQUIPO.INSERTAR('Prueba 2 - Inserción equipo con liga null','cadete',NULL,FALSE);

PRUEBAS_EQUIPO.INSERTAR('Prueba 3 - Inserción equipo con categoria null',NULL,'local',FALSE);

PRUEBAS_EQUIPO.ACTUALIZAR('Prueba 4 - Actualización del categoria de
equipo',SEC_EQUIPO.CURRVAL,'juvenil','local',TRUE);

PRUEBAS_EQUIPO.ACTUALIZAR('Prueba 5 - Actualización del liga de
equipo',SEC_EQUIPO.CURRVAL,'juvenil','provincial',TRUE);

PRUEBAS_EQUIPO.ACTUALIZAR('Prueba 6 - Actualización a null',SEC_EQUIPO.CURRVAL,NULL,NULL,FALSE);

pruebas_equipo.ELIMINAR('Prueba 7 - Eliminar equipo',sec_equipo.currval,true);

END;

--6.2 PruebaJugadora.sql

SET SERVEROUTPUT ON;


BEGIN
```

```
PRUEBAS_JUGADORA.INICIALIZAR;

PRUEBAS_Jugadora.INSERTAR('Prueba 1 - Inserción','11111111a',12,60,'central','D','senior',1,TRUE);

PRUEBAS_JUGADORA.INSERTAR('Prueba 2 - Prueba de nombre null',NULL,FALSE);

PRUEBAS_JUGADORA.ACTUALIZAR('Prueba 3 - Actualización del lugar',SEC_PISTA.CURRVAL,'malaga',TRUE);

pruebas_Jugadora.ELIMINAR('Prueba 4 - Eliminar Jugadora','11111111a',true);


END;

--7.2 PruebasLesion.sql

SET SERVEROUTPUT ON;


BEGIN

 pruebas_lesion.INICIALIZAR;

 pruebas_lesion.INSERTAR('Prueba 1 - Inserción lesion','123456789','desgarro
anal','23/01/2014','31/01/2014',true);

 pruebas_lesion.INSERTAR('Prueba 2 - Inserción lesion con tipo null',null,null,'23/01/2014','31/01/2014',false);

 pruebas_lesion.ACTUALIZAR('Prueba 3 - Actualización del fecha fin','01/02/2014',true);

 pruebas_lesion.ACTUALIZAR('Prueba 4 - Disparador, fecha fin anterior a fechainicio','31/01/2013',false);

 pruebas_lesion.ELIMINAR('Prueba 5 - Eliminar lesion',true);

END;

--8.2 PruebaLugarPartido.sql

SET SERVEROUTPUT ON;


BEGIN


 /**********************************************************************

     PRUEBAS DE LAS OPERACIONES SOBRE LA TABLA LUGARPARTIDO

 **********************************************************************/

 Prueba_LugarPartido.INICIALIZAR;

 Prueba_LugarPartido.INSERTAR('Prueba 1 - Inserción lugar correcta', 'Dos Hermanas', 10, true);

 Prueba_LugarPartido.INSERTAR('Prueba 2 - Inserción distancia negativa incorrecta', 'Lora del Río',-9 , false);

 Prueba_LugarPartido.ACTUALIZAR('Prueba 3 - Actualizar distancia', 'Dos Hermanas',  9, false); --fallo del codigo?
```

```
  Prueba_LugarPartido.ACTUALIZAR('Prueba 6 - Actualizar distancia con dato negativo', 'Dos Hermanas', -7, false);

  Prueba_LugarPartido.ACTUALIZAR('Prueba 7 - Actualizar lugar no existente', 'Zombieland', 9, false);

  Prueba_LugarPartido.ELIMINAR('Prueba 9 - Eliminar empleado', 'Dos Hermanas', true);


END;

--9.2 PruebaMaterial.sql

SET SERVEROUTPUT ON;


BEGIN

  pruebas_material.INICIALIZAR;

  pruebas_material.INSERTAR('Prueba 1 - Inserción material',10,'decathlon','pista',true);

  pruebas_material.INSERTAR('Prueba 2 - Inserción material con tipo null',10,null,null,false);

  pruebas_material.ACTUALIZAR('Prueba 3 - Actualización del stock',15,true);

  pruebas_material.ACTUALIZAR('Prueba 4 - Disparador, stock menor que 0',-2,false);

  pruebas_material.ELIMINAR('Prueba 5 - Eliminar material',true);

END;

--10.2 PruebaOficial.sql

SET SERVEROUTPUT ON;



BEGIN


  /*****************************************************************************

      PRUEBAS DE LAS OPERACIONES SOBRE LA TABLA OFICIAL, Y SOBRE LA TABLA PERSONA

  *****************************************************************************/

  Prueba_Oficial.INICIALIZAR;   --si el primero da exito, el segundo da fallo y viceversa, estando los dos bien

  Prueba_Oficial.INSERTAR('Prueba 1 - Inserción Oficial correcto', '49091675s','Don Nadie','Espacio en blanco', 'Calle
Sierpes', '12/12/1992', 150, 'Nacional', 'alevin', true);

  Prueba_Oficial.INSERTAR('Prueba 2 - Inserción Oficial con sueldo negativo', '49091675K','Antonio','Martin', 'Calle
Sierpes', '12/12/1992', -2, 'Nacional', 'alevin', false);

  Prueba_Oficial.INSERTAR('Prueba 3 - Inserción Oficial con titulacion incorrecta', '49091675R','Antonia','Martin',
'Calle Sierpes', '12/12/1992', 250, 'Nacionalista', 'alevin', false);
```

```
  Prueba_Oficial.INSERTAR('Prueba 4 - Inserción Oficial con categoria incorrecta', '49091675R','Antonia','Martin',
'Calle Sierpes', '12/12/1992', 250, 'Nacional', 'fjnañh', false);

  Prueba_Oficial.ACTUALIZAR('Prueba 5 - Actualizar Oficial correctamente', '49091675R','Antonia','Martin', 'Calle
Sierpes', '12/12/1992', 250, 'Nacional', 'benjamin', true);

  Prueba_Oficial.ACTUALIZAR('Prueba 6 - Actualizar Oficial erroneamente', '49091675R','Antonia','Martin', 'Calle
Sierpes', '12/12/1992', -75, 'Nacional', 'benjamin', false);

  Prueba_Oficial.ACTUALIZAR('Prueba 7 - Actualizar Oficial inexistente', '49091667d','Antonia','Martin', 'Calle
Sierpes', '12/12/1992', 250, 'Nacional', 'benjamin', false);

  Prueba_Oficial.ELIMINAR('Prueba 9 - Eliminar oficial', '49091675m', true);


END;

--11.2 PruebaOficialEquipo.sql

SET SERVEROUTPUT ON;


BEGIN


  /**********************************************************************

      PRUEBAS DE LAS OPERACIONES SOBRE LA TABLA OFICIALEQUIPO

  **********************************************************************/

  Prueba_OficialEquipo.INICIALIZAR;

  Prueba_OficialEquipo.INSERTAR('Prueba 1 - Inserción oficialequipo correcta', '49091675s', 3, true);

  Prueba_OficialEquipo.INSERTAR('Prueba 2 - Inserción oficialequipo incorrecta', '49091675s',-9 , false);

  Prueba_OficialEquipo.ACTUALIZAR('Prueba 3 - Actualizar oficialequipo correctamente', '49091675s',  9, true); --
fallo del codigo?

  Prueba_OficialEquipo.ACTUALIZAR('Prueba 6 - Actualizar equipo con ID negativo', '49091675s', -7, false);

  Prueba_OficialEquipo.ACTUALIZAR('Prueba 7 - Actualizar oficial no existente', 'sfasf', 9, false);

  Prueba_OficialEquipo.ELIMINAR('Prueba 9 - Eliminar oficialequipo', '49091675s', true);


END;

--12.2 PruebaPartido.sql

SET SERVEROUTPUT ON;


BEGIN
```

```
 pruebas_partido.INICIALIZAR;

 pruebas_partido.INSERTAR('Prueba 1 - Inserción partido',1,2,'23/03/2014','Lora del Rio','Polideportivo Local','senior',3,3,'Lora del Rio','local',true);

 pruebas_partido.INSERTAR('Prueba 2 - Inserción partido con categoria null',1,2,'23/03/2014','Lora del Rio','Polideportivo Local',null,3,3,'Lora del Rio','local',false);

 pruebas_partido.ACTUALIZAR('Prueba 3 - Actualización de goles',1,'23/03/2014',5,5,true);

 pruebas_partido.ACTUALIZAR('Prueba 4 - Disparador, goles a favor negativo',1,'23/03/2014',-2,2,false);

 pruebas_partido.ACTUALIZAR('Prueba 5 - Disparador, goles en contra negativo',1,'23/03/2014',2,-2,false);

 pruebas_partido.ELIMINAR('Prueba 6 - Eliminar partido',1,'23/03/2014',true);

END;

--13.2 PruebaPersona.sql

SET SERVEROUTPUT ON;

BEGIN

pruebas_persona.INICIALIZAR;

pruebas_persona.INSERTAR('Prueba 1 - Inserción persona','123456789','Pepito','Grillo','cartones','21/02/1984',true);

pruebas_persona.INSERTAR('Prueba 2 - Inserción persona con nombre null','12345678A',null,'Grillo','cartones','21/02/1984',false);

pruebas_persona.ACTUALIZAR('Prueba 3 - Actualización del nombre de persona','123456789','Jiminy',true);

pruebas_persona.ACTUALIZAR('Prueba 4 - Actualización del nombre de persona a null','123456789',null,false);

pruebas_persona.ELIMINAR('Prueba 5 - Eliminar persona','123456789',true);

END;

--14.2 PruebPista.sql

SET SERVEROUTPUT ON;


BEGIN


PRUEBAS_PISTA.INICIALIZAR;

PRUEBAS_pista.INSERTAR('Prueba 1 - Inserción','paquito',TRUE);

PRUEBAS_PISTA.INSERTAR('Prueba 2 - Prueba de nombre null',NULL,FALSE);

PRUEBAS_PISTA.ACTUALIZAR('Prueba 3 - Actualización del lugar',SEC_PISTA.CURRVAL,'malaga',TRUE);

pruebas_pista.ELIMINAR('Prueba 4 - Eliminar pista',sec_pista.currval,true);
```

```
END;

--15.2 PruebaReservas.sql

SET SERVEROUTPUT ON;



BEGIN


 /**********************************************************************

      PRUEBAS DE LAS OPERACIONES SOBRE LA TABLA LUGARPARTIDO

 **********************************************************************/

 Prueba_Reservas.INICIALIZAR;

 Prueba_Reservas.INSERTAR('Prueba 1 - Inserción reserva correcta', '49091675K','12/5/2012','12/5/2012',3,2, true);
--falla porque no hay reservas creadas

 Prueba_Reservas.INSERTAR('Prueba 2 - Inserción datos negativos incorrecta',
'49091675K','12/5/2012','12/5/2012',3,-2, false);

 Prueba_Reservas.ACTUALIZAR('Prueba 3 - Actualizar reserva','1', '49091675K','12/5/2012','12/5/2012',3,3, true); --
no hay reservas creadas

 Prueba_Reservas.ACTUALIZAR('Prueba 4 - Actualizar reserva con dato negativo', '1',
'49091675K','12/5/2012','12/5/2012',-2,3, false);

 Prueba_Reservas.ACTUALIZAR('Prueba 5 - Actualizar reserva no existente', '450',
'49091675K','12/5/2012','12/5/2012',3,4, false);

 Prueba_Reservas.ELIMINAR('Prueba 9 - Eliminar reserva', '1', true);

 Prueba_Reservas.ELIMINAR('Prueba 9 - Eliminar reserva inexistente', '140', false); --no funciona


END;
```

ANEXO

-Fotografías-

Reunión

ACTA de Reunión

Siendo las 21 horas, del día miércoles, 16 de octubre de 2013, se reúne en las instalaciones del Club Balonmano Unión Sevilla, D. Juan Antonio Bascón y D. José Manuel López, con el fin de confirmar el acuerdo de la elaboración de un proyecto informático con el fin de ayudar al club en lo que necesite por parte de José Manuel López, Guillermo De La Cruz y Fco. Javier Boza, siguiendo los requisitos hablados tanto en persona como por correo electrónico entre ambas partes, en las fechas anterior a esta y verificando la validez de los documentos relacionados con lo comentado anteriormente y los documentos fotográficos aportados, dando vía libre a la ejecución del proyecto con los requisitos aportados.