|  |
| --- |
| programación orientada a objetos – 314 |
| Práctica 03 |
| Curso 2015-2016 |
|  |
| **Asier González López** |
| **10/01/2016** |

3º University of Wales (Centro Sanluis)

|  |
| --- |
|  |

http://www.centrosanluis.com/resources/img/logo_sanluis.gif

Índice

[Abstract 1](#_Toc440279929)

[Convenciones utilizadas en la base de datos 1](#_Toc440279930)

[Tablas 1](#_Toc440279931)

[Columnas 1](#_Toc440279932)

[Diagrama Entidad Relación 1](#_Toc440279933)

[Modelo esquema relacional 2](#_Toc440279934)

[Análisis de la base de datos 2](#_Toc440279935)

[Usuarios 2](#_Toc440279936)

[Equipos y jugadores 2](#_Toc440279937)

[Ligas, jornadas y partidos 2](#_Toc440279938)

[Estadísticas (Acciones, goles y sanciones) 3](#_Toc440279939)

[Patrón de diseño DAO y VO 3](#_Toc440279940)

# Abstract

The purpose of this document is to explain how I designed the database for the oriented programming cousework number 3. On this document I also explain which design pattern I used to manage de data from the database using the object oriented programming language.

# Convenciones utilizadas en la base de datos

En la base de datos diseñada se siguen algunas convenciones las cuales se han considerado para llevar una constancia en el nombramiento de las tablas y columnas de toda la base de datos. De esta manera, se facilita su manejo, ya que se sigue un patrón de reglas.

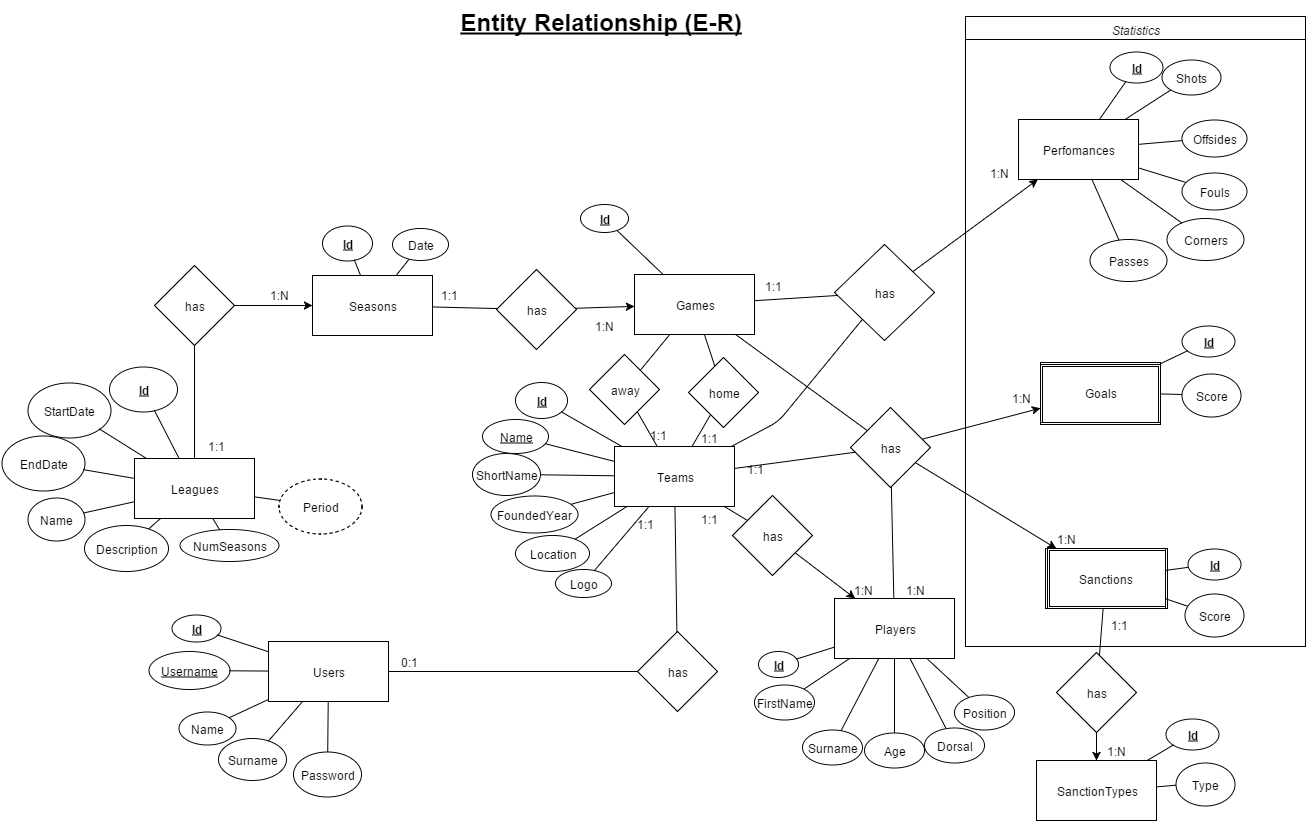
## Tablas

Todas las tablas o relaciones se nombran capitalizándolas y en plural.

## Columnas

Todas las columnas o atributos de las tablas se nombran en singular, siguiendo la convención de *camelcase*, pero con la primera letra de la palabra también se coloca en mayúsculas.

# Diagrama Entidad Relación



# Modelo esquema relacional

**Users** (Id, IdTeam, **Username**, Name, Surname, Password)

**Teams** (Id, **Name**, ShortName, FoundedYear, Location, Logo)

**Players** (Id, IdTeam, FirstName, Surname, Age, **Dorsal**, Position)

**Leagues** (Id, Name, StartDate, EndDate, Description, NumSeasons, Period)

**Seasons** (Id, IdLeague, Date)

**Games** (Id, IdSeason, IdHomeTeam, IdAwayTeam)

**Performances** (Id, IdGame, IdTeam, Shots, Passes, Fouls, Offsides, Corners)

**Goals** (Id, IdGame, IdTeam, IdPlayer, Score)

**SanctionTypes** (Id, Type)

**Sanctions** (Id, IdGame, IdPlayer, IdSanctionType)

# Análisis de la base de datos

## Usuarios

En esta aplicación se crea un usuario por defecto con el nombre de usuario “mikel” con la contraseña “123”. Al usuario se le asigna el id del equipo que le pertenece. Esto se hace para evitar tener que buscar por cada equipo, cual es el que contenga jugadores, ya que el equipo del usuario es el único que contiene jugadores.

## Equipos y jugadores

La relación **equipo** es una relación que contiene los datos de un equipo, como es el nombre, el cual es único, el año en el que se ha fundado, el nombre de la imagen del logo (la cual puede ser modificada mediante la aplicación por cada equipo), la localidad y un identificador del equipo auto incrementable.

Este identificador para el equipo se ha creado para dar facilidad a la identificación de cada equipo por un valor numérico (los cuales son más manejables que los valores de texto, como sería el nombre) y facilitar así las consultas entre diferentes relaciones. Además, en SQLite no tenemos la posibilidad de realizar una UPDATE, por lo que este aspecto nos va a servir para realizar dichas operaciones más eficientemente.

Un equipo puede tener 0 a muchos jugadores, por lo que tenemos también una relación entre equipos y jugadores.

## Ligas, jornadas y partidos

Las **ligas** contienen un periodo de duración, las cuales son manejadas con una fecha de inicio y una fecha final. Por ese motivo, el periodo de la ligua se calcularía a partir de estos campos (lo que se le denomina un atributo derivado). Dentro de la aplicación, también se aprovecha dicho periodo para calcular los días restantes para finalizar la liga desde la fecha actual.

Cada liga contiene una o más jornadas, donde el número de jornadas que contiene la liga está delimitado por la propia liga. Al crear una liga, se especifica el número de jornadas que esta queramos que contenga y después, se generará dicho número de jornadas.

Una **jornada** contiene la fecha de la misma (la fecha se deja para que la modifique el usuario), y en el caso de nuestra aplicación un partido. En esta aplicación vamos a tener un partido por jornada, el cual es el partido del usuario contra otro equipo de un oponente, en donde uno de ellos será el equipo local y mientras que el otro equipo será el visitante (generado de forma aleatoria). La idea de que una jornada tenga una relación de uno a muchos es porque de esta manera en un futuro, si se quieren jugar más partidos en una jornada, pueda ser escalable (ya que será posible).

Un **partido**, tiene el identificador del equipo local y del equipo visitante, así como su propio identificador, para poder identificar cada partido por esta clave primaria auto incrementable. También contendrá el identificador de la jornada (la cual es la clave foránea), la cual indica a que jornada pertenece el partido.

## Estadísticas (Acciones, goles y sanciones)

Dentro de las estadísticas entran las acciones realizadas por cada equipo y los goles conseguidos (junto al jugador que marcó el gol o los goles). También tenemos las sanciones, las cuales se asignaron a un jugador de un equipo dentro de un partido.

Las **acciones** se almacenan dentro de una relación donde se guardan el número de acciones, como los tiros a puerta, los córneres, las faltas, los pases, etc. Estas acciones están identificadas por un atributo id el cual es auto incrementado. Cada número de acciones pertenece dentro de un partido a un equipo, por lo que esta tabla contendrá la referencia al partido y equipo al que pertenece, dejándonos una relación ternaría entre acciones, equipo y partido.

Los **goles**, almacenarán a diferencia de las acciones, almacenarán también a que jugador pertenece el gol, pero no en todos los casos, ya que los equipos que no sean del usuario no tendrán jugador, por lo que también se almacenarán goles sin referencia a ningún jugador. Para conseguir esto tenemos la relación goles, en la que se almacenará el número de goles marcados por un jugador en un partido que pertenece a un equipo. Esta relación nos dará paso a una relación cuaternaria, ya que forman parte de la relación las tablas equipos, jugadores (no siempre), partidos y la propia tabla goles.

Para manejar las **sanciones**, tendremos una tabla en la que se almacenan los diferentes tipos de sanciones (como las tarjetas rojas y amarillas) llamada tipos de sanciones. Después desde las sanciones referenciaremos a un tipo de sanción y al jugador al que se le ha asignado dicha sanción dentro de un partido. Por lo que tendremos una relación ternaria entre partidos, jugadores y sanciones.

Tanto la tabla goles, como la de sanciones son entidades débiles, ya que ni una sanción ni un gol pueden existir sin participar en la relación dentro de un juego en un equipo (por un jugador).

# Patrón de diseño DAO y VO

En la parte de programación se utiliza el patrón DAO junto al VO para tratar con la base de datos utilizando la programación orientada a objetos.

Los objetos **DAO** (Data Access Object) son los encargados de acceder a la base de datos mediante la conexión para realizar la consulta deseada.

Los objetos **VO** (Value Object) son la representación de las tablas de la base de datos, donde se realiza un mapeo para conseguir un ORM, donde un objeto representa a un registro de una tabla, y un atributo del objeto hace referencia a una columna de dicha tabla. También si realizamos una colección de estos objetos, se representaría la tabla resultante de nuestra consulta.