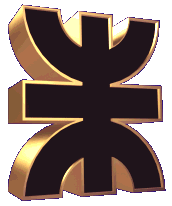
|  |
| --- |
| **Nombre y Apellido** |
| Cappellini Darío |
| Carella Germán |
| Centurion Damian |
| Gonzalez Santiago |

|  |  |
| --- | --- |
| **Fecha De Presentación:** | 22/10/2014 |
| **Fecha de Devolución:** |  |
| **Calificación** |  |
| **Firma Profesor** |  |



**GRUPO N° 7**

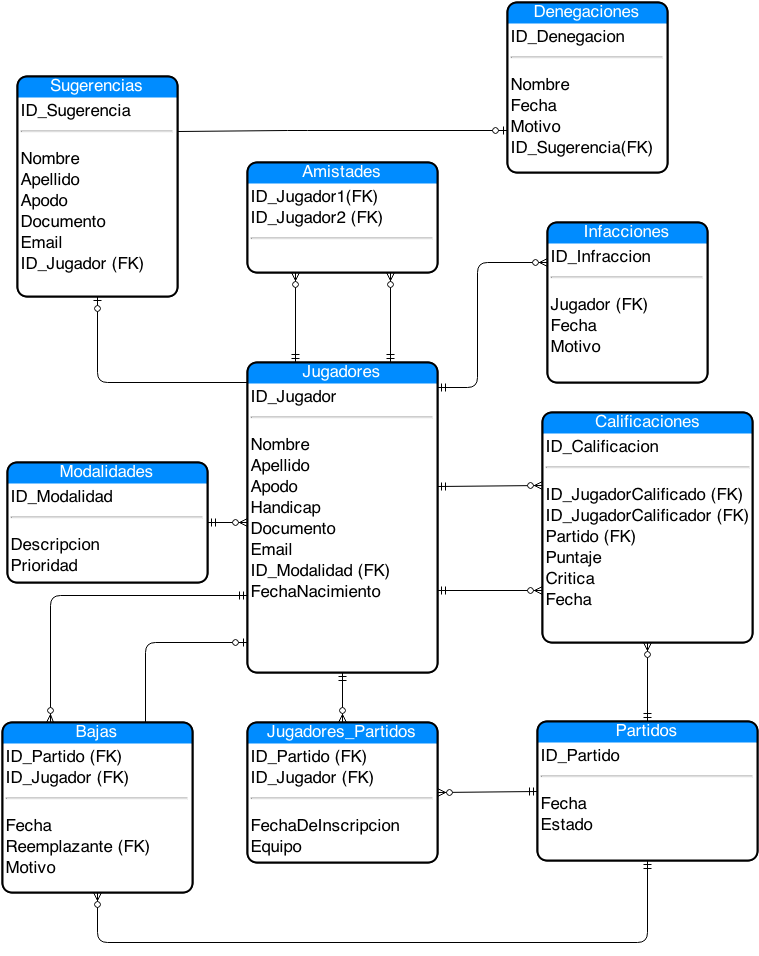
TP Anual: 8va Entrega

***Diseño de Sistemas 2014***

***Miércoles mañana***

**Modelo de datos**

El siguiente diagrama muestra las entidades que aparecen en nuestro modelo de datos:



Los atributos que aparecen por arriba de la barra separadora conforman la primary key. Marcamos con (FK) los atributos que son foreign key. Incluimos los tipos de dato de cada atributo y algunas aclaraciones necesarias para entender el modelo:

**Jugadores**

* ID\_Jugador: int
* Nombre: varchar(50)
* Apellido: varchar(50)
* Apodo: varchar(50)
* Documento: varchar(8)
* Email: varchar(50)
* FechaNacimiento: DateTime
* Handicap: tinyint
* ID\_Modalidad: int

El ID\_Modalidad se refiere al tipo de inscripción (estándar, solidaria o condicional) y hace referencia a una tabla que tiene los datos de estos tipos. Cada fila de esta tabla representa un objeto del dominio.

**Partido**

* ID\_Partido: int
* Fecha: DateTime
* Estado: CHAR(1).

El estado se refiere a uno de los siguientes: abierto, cerrado (que tiene los 10 jugadores confirmados) y con equipos generados. Cada fila de esta tabla representa un objeto del dominio.

**Jugadores\_Partidos**

* ID\_Partido: int
* ID\_Jugador: int
* FechaDeInscripcion: DateTime
* Equipo: tinyint

Esta tabla tiene una fila por cada jugador que está anotado a un partido. Otra forma de verlo es que, si hay n jugadores inscriptos al partido 1, va a haber n filas que tengan partido igual a 1. El campo equipo sólo se utiliza para partidos con equipos generados, y toma el valor 1 o 2 según el equipo asignado al jugador. Si no hubiera equipos generados, este campo es NULL.

**Modalidades**

* ID\_Modalidad: int
* Descripcion: varchar(20)
* Prioridad: tinyint

Esta tabla tiene una entrada por cada tipo de inscripción posible. Mapea el tipo de inscripción con su prioridad (recordar que la prioridad tenía un rol al determinar la inscripción de los jugadores). NO se guarda una fila por cada objeto “tipo de inscripción” ya que estos objetos no tienen estado que sea necesario persistir.

**Calificaciones**

* ID\_Calificacion: int
* ID\_JugadorCalificado: int
* ID\_JugadorCalificador: int
* ID\_Partido: int
* Puntaje: tinyint (1-10)
* Critica: varchar(255)
* Fecha: DateTime

Se almacena tanto el jugador que emitió la calificación como el jugador calificado. Cada fila de esta tabla representa un objeto del dominio.

**Infracciones**

* ID\_Infraccion: int
* ID\_Jugador: int
* Fecha: DateTime
* Motivo: varchar(255)

Cada fila de esta tabla representa un objeto del dominio.

**Amistades**

* ID\_Jugador1: int
* ID\_Jugador2: int

Esta tabla guarda un registro por cada amistad. Se guardan los pares de jugadores amigos.

**Sugerencias**

* ID\_Sugerencia: int
* Nombre: varchar(50)
* Apellido: varchar(50)
* Apodo: varchar(50)
* FechaNacimiento: DateTime
* Documento: varchar(8)
* Email: varchar(50)
* ID\_Jugador: int

Cada fila de esta tabla representa un objeto del dominio. Agregamos un ID\_Jugador para incluir la posibilidad de que, una vez aprobada una sugerencia, se sepa en qué jugador se convirtió.

**Denegaciones**

* ID\_Denegacion: int
* Fecha: DateTime
* Motivo: varchar(255)
* ID\_Sugerencia: int

Cada fila de esta tabla representa un objeto del dominio. Agregamos un ID\_Sugerencia para incluir la posibilidad de saber cuál fue la sugerencia que se rechazó.

**Bajas**

* ID\_Partido: int
* ID\_Jugador: int
* Fecha: DateTime
* Reemplazante: int
* Motivo: varchar(255)

Esta tabla se agrega para cumplir el requisito 3.d “reflejar cuando un jugador se da de baja a un partido”.

**Requerimientos adicionales**

* Saber los jugadores malos: creamos una función que, dado un ID de jugador, nos dice si es malo. Para saber los jugadores malos basta con hacer select ID\_Jugador from jugadores where Fn\_EsMalo(jugador)
* Saber los jugadores traicioneros: creamos un stored procedure que devuelve una tabla con los jugadores traicioneros.
* Saber los jugadores que pueden mejorar: creamos una función GetEdad() que permite saber la edad de un jugador. Utilizando esta función junto con la función EsMalo() definida anteriormente se puede cumplir este requerimiento sin repetir código
* Reflejar que un jugador se da de baja
* Agregar una infracción cuando alguien se da de baja sin reemplazante: creamos un trigger sobre la tabla bajas que, al detectar que se está insertando un registro cuyo reemplazante es NULL, le crea una infracción al jugador.