ASIGNATURA: PROGRAMACIÓN WEB

Profesores: Nicolás Emilio García Pedraja - Javier Pérez Rodríguez

Aplicación Web - Trabajo de Prácticas.

Aplicación de Gestión de una Liga Deportiva

Emilio López Piña i12lopie@uco.es

UNIVERSIDAD DE CÓRDOBA
ESCUELA POLITÉCNICA SUPERIOR
DEPARTAMENTO DE INFORMÁTICA Y ANÁLISIS NUMÉRICO

10 de Febrero de 2.014

ÍNDICE

MANUAL DE USUARIO

1. INTRODUCCIÓN

La aplicación web para la gestión de una liga de fútbol permite consultar los datos sobre dicha competición, resultados, clasificaciones, información de equipos y de sus jugadores, colegiados que arbitran en la competición, sanciones y noticias sobre la misma. También permite añadir datos y modificar los mismos a los usuarios registrados en el sistema.

2. TIPOS DE USUARIO

El sistema contempla dos grupos de usuarios:

– **Usuarios no autorizado**s: son aquellos que navegan por el sistema y pueden ver toda la información disponible sobre la competición. Sin embargo, no tienen derechos para añadir y modificar dicha información.

– **Administradores globales**: tienen acceso a todos los apartados del sistema: administrar calendario, clasificaciones, equipos, árbitros y noticias.

3. PANTALLA DEL SISTEMA

En la página inicial del sistema aparece la cabecera de la aplicación, con los diferentes menús, la lista de equipos, últimas noticias, breve información sobre el usuario en el caso de que se haya iniciado sesión y un mensaje enlazando con la pantalla de inicio o registro en caso contrario.

4. REGISTRO DE UN NUEVO USUARIO

Para poder registrarse en la aplicación, añadir y modificar información sobre la competición, hay que acceder al formulario de registro mediante el enlace “Registrarse”.

– “Nombre propio”:
– “Apellidos”:
– El “**Nombre de usuario**” es único y obligatorio. Si el campo introducido ya se encuentra en la base de datos se mostrará un mensaje de error. En caso de dejar este campo en blanco, también se mostrará un mensaje de error.
– El campo de “**Dirección de correo electrónico**” también es obligatorio y único en la base de datos.
– La “**Contraseña**” es obligatoria como los campos de formulario citados anteriormente. Para asegurar que su valor es el correcto, se debe introducir dos veces, en caso contrario aparecerá un mensaje de error.

5. ACCESO AL SISTEMA

Para ingresar en el sistema, debemos pulsar el enlace de “Iniciar sesión” situado a la derecha de la página inicial. A continuación, debemos rellenar el siguiente formulario de acceso:

Una vez que hayamos logrado registrarnos en el sistema, éste nos direccionará a la pantalla inicial donde aparecerá nuestro nombre de usuario.

6. EDITAR/BORRAR USUARIOS

- Editar usuario: Para modificar algún dato de un usuario hay que estar registrado como ese usuario. Para editar la información hay que pulsar el botón “Editar” situado junto al nombre de usuario y obtendremos un formulario similar al de registro de usuario.

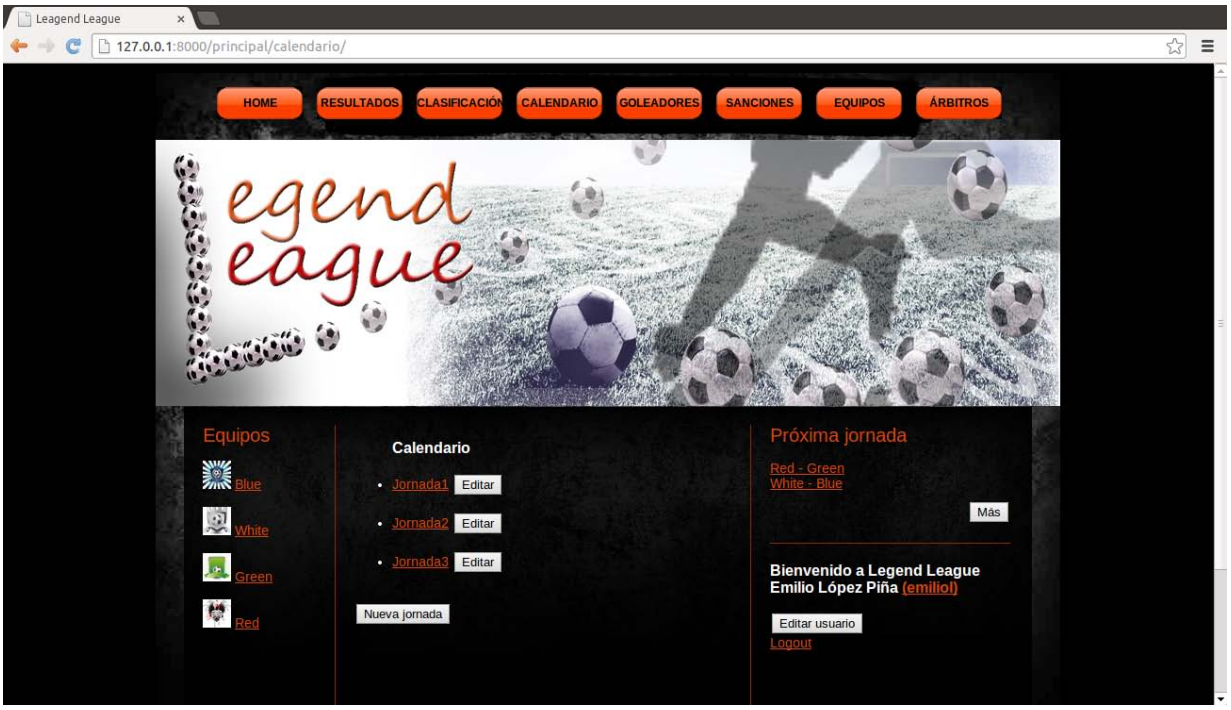
-Eliminar usuario: Para eliminar el usuario también hay que estar registrado como ese usuario. Para ello hay que acceder a los detalles del perfil, una vez dentro se pulsará el botón “Eliminar” y procedemos a la eliminación inmediata del sistema de dicho usuario.

7. ADMINISTRACIÓN: AGREGAR, EDITAR Y ELIMINAR ENTIDADES

Estas acciones solo pueden ser realizadas por usuarios registrados en el sistema.

8. GESTIÓN DEL CALENDARIO

Para poder consultar todos los partidos previstos de la competición accedemos al menú “CALENDARIO”.

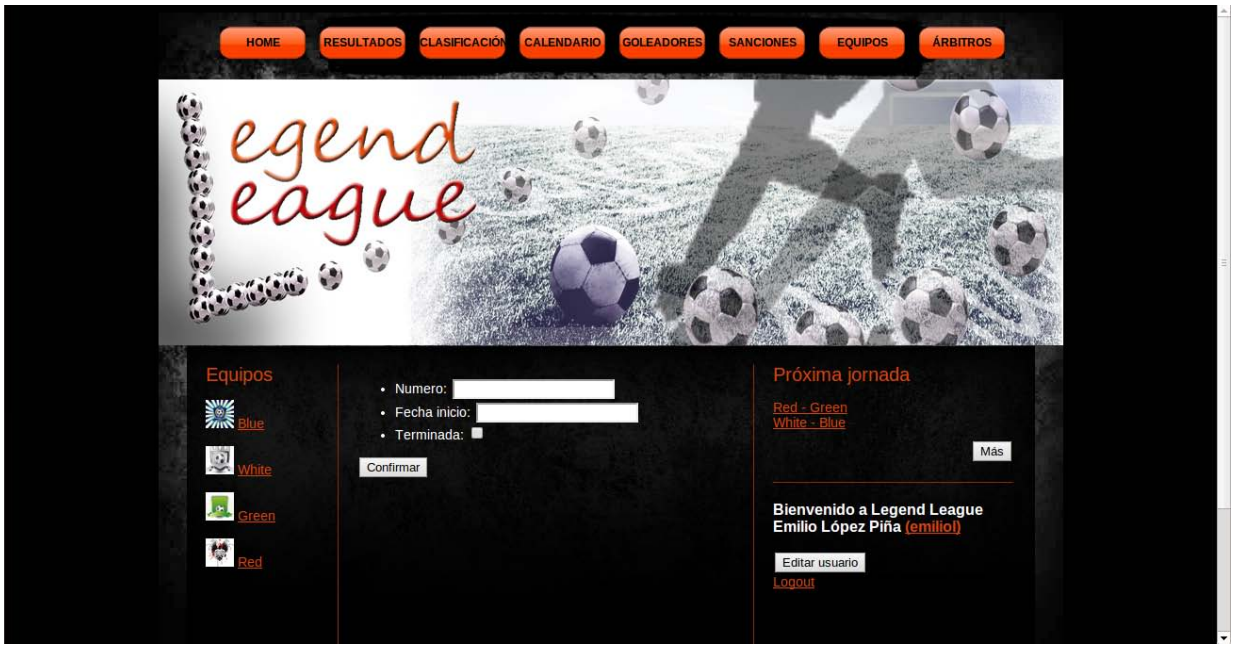


A continuación, aparece un listado con las jornadas de la competición. A partir de este menú podremos modificar, añadir y borrar jornadas.

8.1 Gestión jornada

8.1.1 Añadir jornada

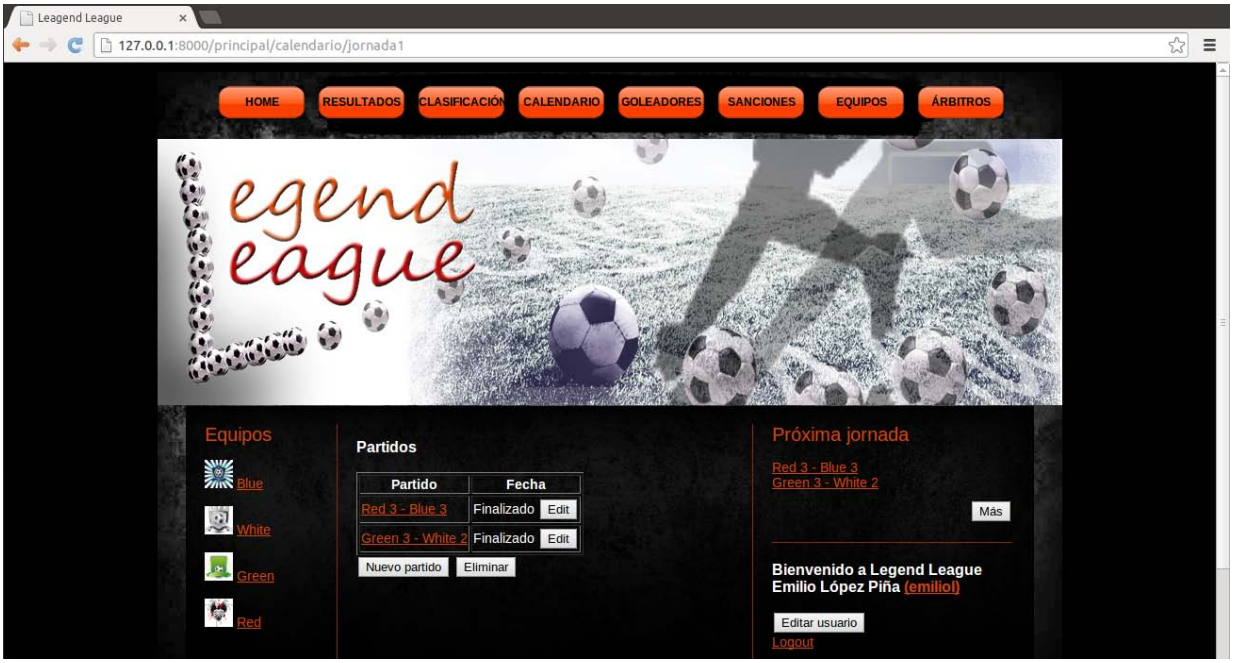
Para añadir una nueva jornada pulsamos el botón “Nueva jornada”.



“Número”: aquí incluimos el número de la jornada que es único.
“Fecha comienzo”: incluye la fecha de comienzo de la jornada.

8.1.2 Visualizar jornada

Para poder ver los detalles de cada jornada pulsamos en el enlace de cada uno de ellas y aparecerá la siguiente página.



Como podemos apreciar, aparecen los equipos que participan en un determinado encuentro junto con el resultado en el caso de que se haya celebrado el mismo, y la fecha de su celebración.
NOTA: las opciones referentes a cada partido se verán en siguientes secciones.

8.1.3 Editar jornada

Para editar una jornada existente volvemos al menú “Calendario” y pulsamos en el botón “Editar” correspondiente a cada jornada aparecerá un menú similar al de “Nueva jornada” para modificar los datos de la misma. Tras modificar los datos y pulsar “Aceptar” el sistema nos redirigirá al menú de “Calendario”.

8.1.4 Eliminar jornada

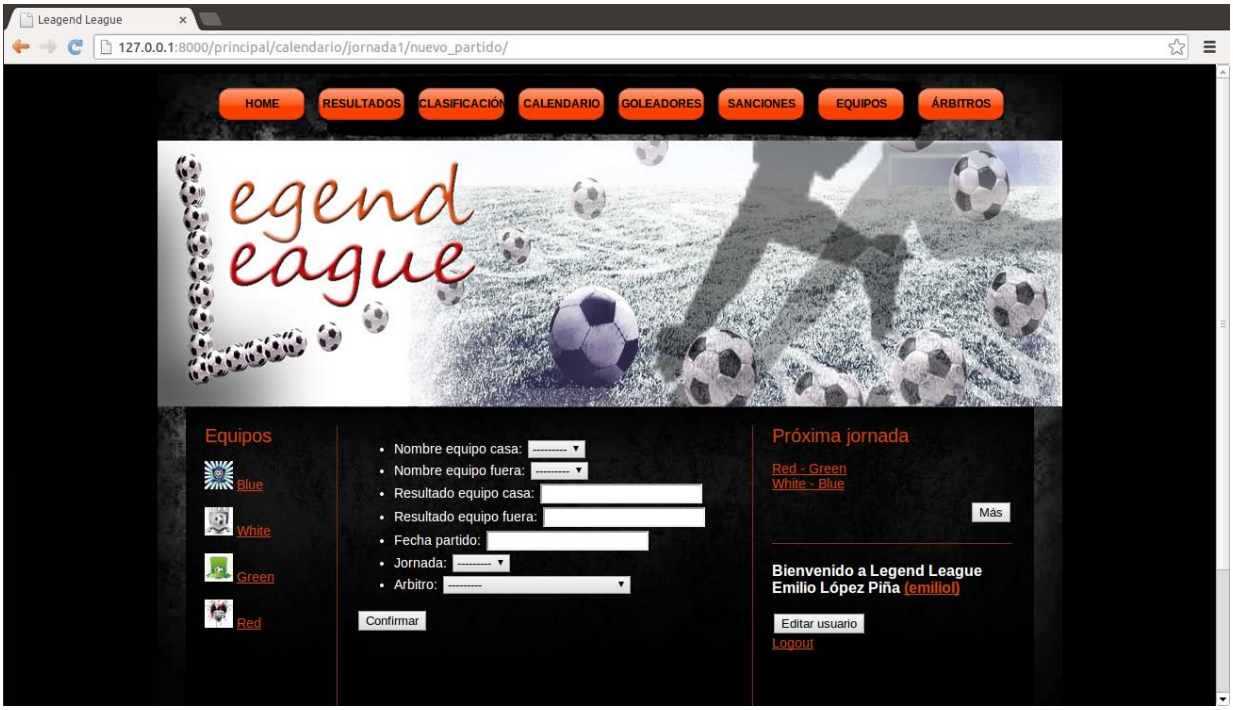
Si queremos eliminar una jornada, nos situaremos en el menú “Calendario” y accedemos a los detalles de cada jornada pulsando el nombre de la misma. Una vez dentro de dicha información pulsamos la opción “Eliminar” para eliminar la jornada del sistema. La aplicación nos redirigirá al menú “Calendario” donde nos mostrará el listado de las jornadas existentes.

8.2 Gestión partidos

Para llevar a cabo las operaciones relativas a esta sección se debe situar en la información de la jornada deseada, donde se nos mostrará el conjunto de partidos de dicha jornada.

8.2.1 Añadir partido

Una vez en el listado de partidos de la jornada, pulsamos el botón “Nuevo partido” para añadir uno nuevo a la jornada ya existente.



Una vez dentro, deberemos rellenar los siguientes campos para añadir un nuevo partido.
- “Nombre equipo casa”: se deberá elegir un equipo de la lista de equipos del sistema.
- “Nombre equipo fuera”: se deberá elegir un equipo de la lista de equipos disponibles.
- “Resultado equipo casa”: este campo se rellenará una vez que haya finalizado el encuentro, no siendo obligatorio cuando este no se haya celebrado.
- “Resultado equifo fuera”: similar a “Resultado equipo fuera”.
- “Fecha de partido”: el formato de fecha será el siguiente DD/MM/AÑO
- “Jornada”: se eligirá una jornada de las anteriormente creadas.

8.2.2 Visualizar partido

Para acceder a la información de cada partido pulsamos el enlace del mismo y el sistema nos proporcionará los siguientes datos:

- Título encuentro: Nombre de los equipos implicados junto con el resultado.
- “Goles”: jugadores que han anotado uno o mas goles en el encuentro disputado.
- “Árbitro”: colegiado que ha pitado el encuentro.
- “Ámarillas”: jugadores amonestados con tarjeta amarilla.
- “Rojas”: jugadores expulsados del encuentro.
- “Estadio”: Nombre del estadio donde se ha celebrado el partido.

8.2.3 Editar partido

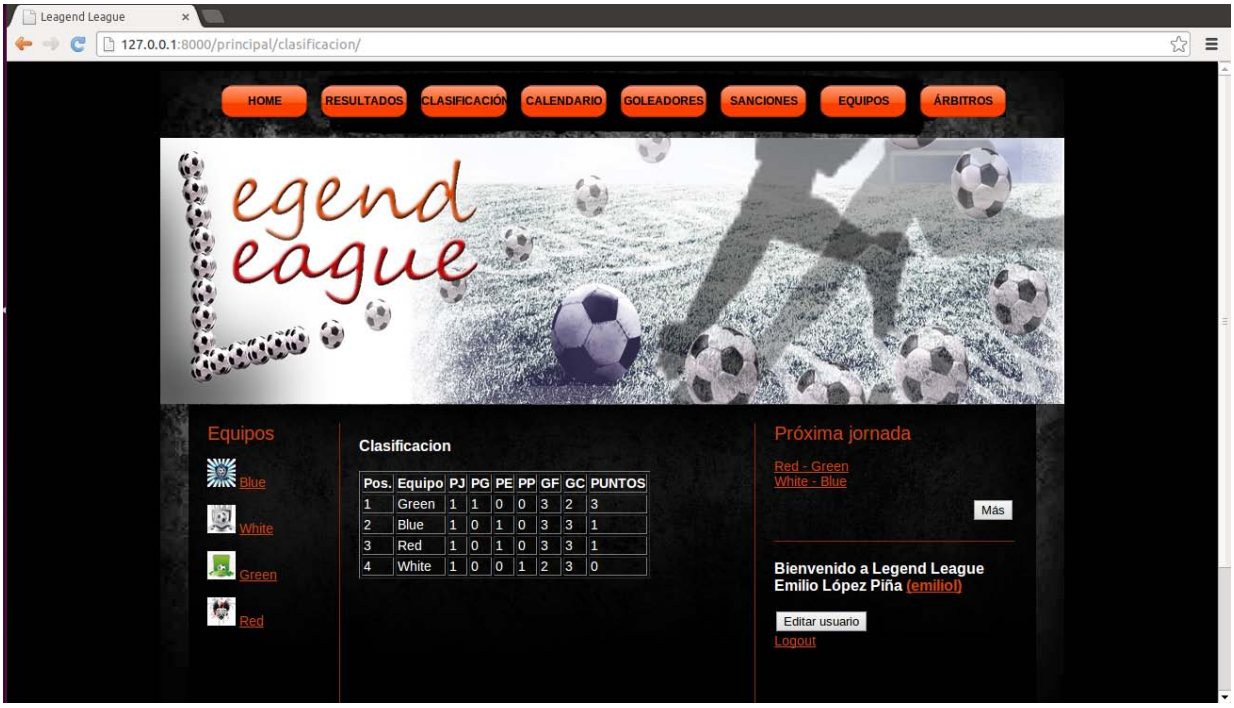
Si queremos modificar la información de un partido se haya celebrado o no, debemos situarnos en el listado de partido visto anteriormente, y pulsamos el botón “Editar” asociado al partido que deseemos modificar. Una vez dentro nos aparecerán los campos anteriormente rellenos con la posibilidad de modificarlos. Tras la modificación el sistema nos redireccionará al listado de partidos.

8.2.4 Eliminar partido

Para eliminar un partido de la lista debemos en primer lugar acceder a la lista de partidos de una determinada jornada, pulsar su enlace y en la descripción del mismo pulsar el botón “Eliminar”. Tras la eliminación del partido, el sistema nos redireccionará al listado de partidos.

9. CLASIFICACIÓN

Para acceder a la clasificación de la competición, debemos pinchar en el botón “Clasificación” del menú de la aplicación. El sistema mostrará la siguiente imagen:



En dicha página vemos junta al nombre de cada equipo, la posición que ocupa en la clasificación, el número de partidos jugados (PJ), partidos ganados (PG), partidos empatados (PE),

partidos perdidos (PP), goles a favor (GF), goles encajados (GC) y puntos.

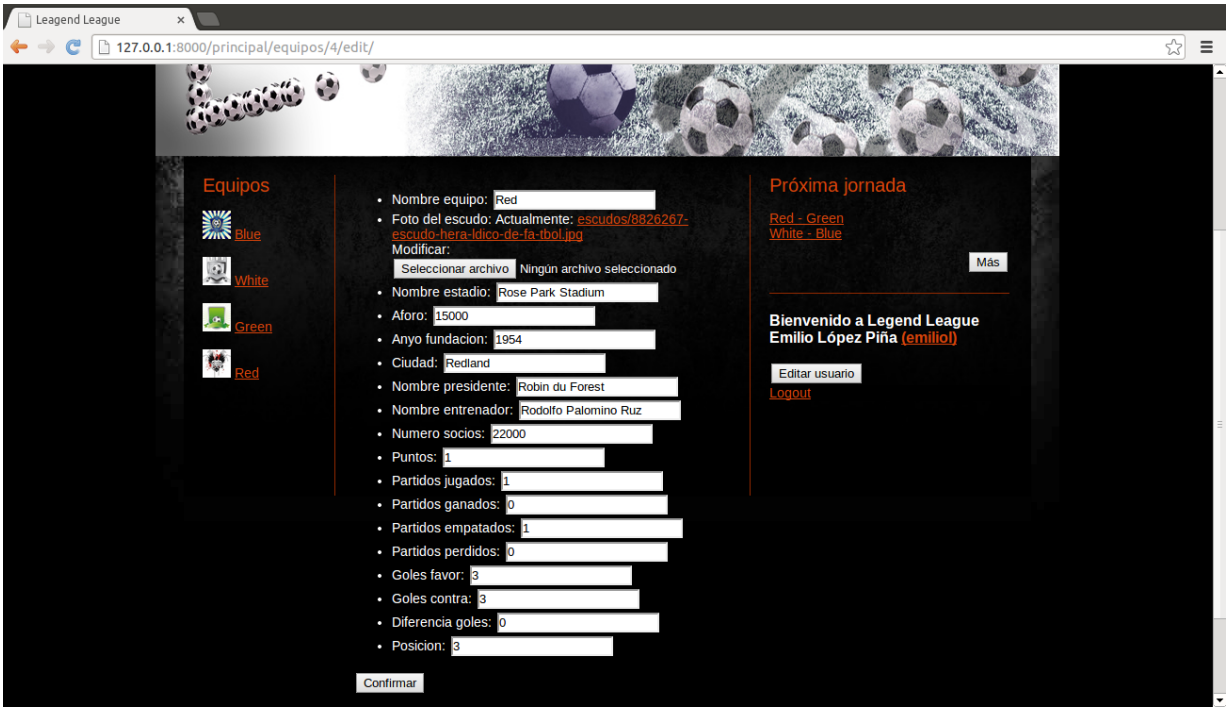
10. GESTIÓN DE EQUIPOS

Para ver los equipos que participan en la competición, lo podemos hacer a través del menú lateral. Sin embargo, para añadir, modificar o eliminar un determinado equipo es necesario acceder al menú “Equipos” desde el botón con el mismo nombre en la barra superior

10.1 Gestión datos equipo

10.1.1 Añadir equipo

Para añadir un nuevo equipo a la base de datos presionamos el botón “Nuevo equipo” del menú “Equipos”. Al presionar el botón anterior se nos mostrará la siguiente imagen:



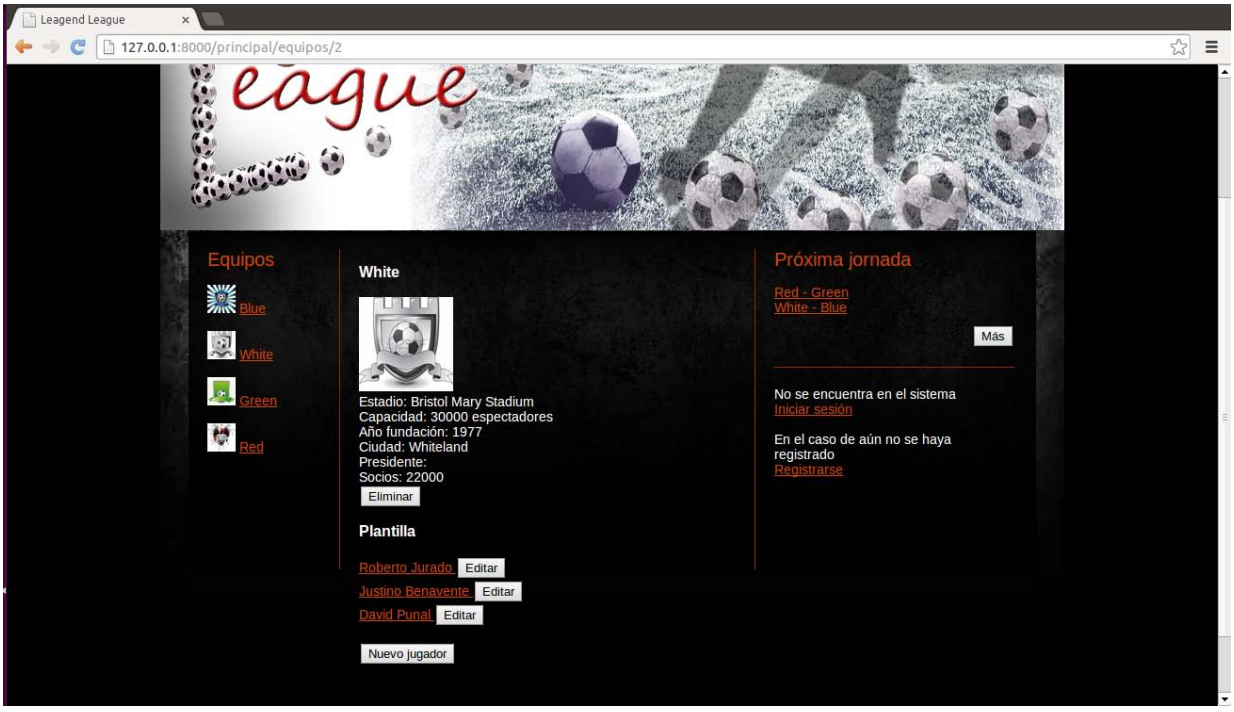
Deberemos rellenar los siguientes datos:
- "**Foto del escudo**": seleccionamos una foto de nuestro equipo con formato .jpg o .png que será tratada como escudo del equipo.
- "**Nombre equipo**"
- "**Nombre estadio"**
- **"Aforo":** capacidad máxima en número de espectadores del estadio del club.
- "**Anyo fundacion":** año en el que se creó el club.
- "**Ciudad":** ciudad sede del equipo.
- "**Nombre presidente":**
- "**Número socios":**
- **"Posición":** posición que ocupa en la clasificación.

Los siguientes está inicializados con el valor 0:
- **"Puntos":**
- **"Partidos jugados":**
- **"Partidos ganados":**
- **"Partidos empatados":**
- **"Partidos perdidos":**
- **"Goles favor":**
- **"Goles contra":**
- **"Diferencia goles":** diferencia de goles a favor y en contra.

Para guardar los valores anteriores se presiona el botón "Aceptar", tras esto, el sistema nos redireccionará al listado de equipos de la competición, donde aparecerá el equipo creado.

10.1.2 Visualizar equipo

Si queremos ver la información de un determinado equipo presionamos su nombre en el listado de equipos situado en la pantalla inicial o del menú "Equipos". Se mostrará la siguiente página:



Los campos visibles son a los anteriormente mencionados en la sección 10.1.1, además de los jugadores que forman el equipo.

10.1.3 Editar equipo

Para modificar la información de un determinado equipo pulsamos el botón “Editar” situado junto al nombre del mismo”. El sistema nos mostrará una página similar a la de creación de club, donde podremos modificar los campos.

Para guardar los cambios realizados presionamos el botón “Guardar” que nos llevará al listado de equipos del sistema.

10.1.4 Eliminar equipo

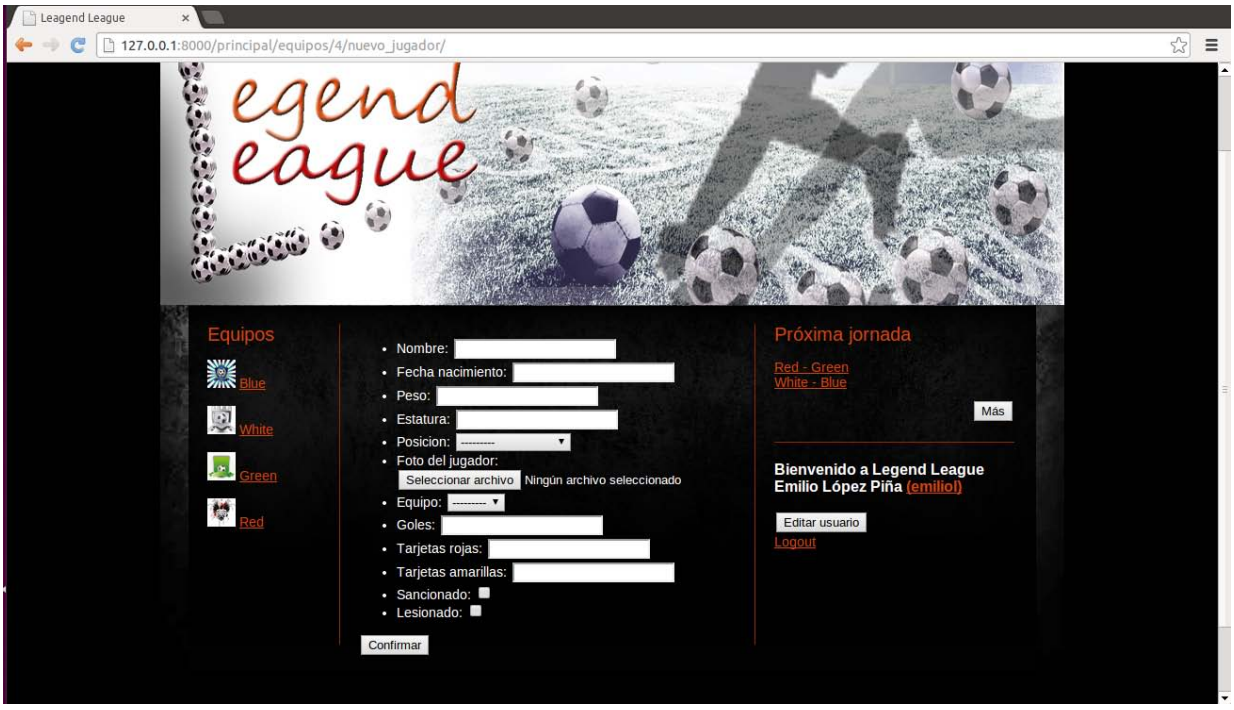
Si queremos eliminar del sistema a un equipo entramos en la información del equipo en cuestión y presionamos el botón “Eliminar”. El sistema nos llevará al listado de equipos del sistema.

10.2 Gestión jugadores

Para acceder al menú de jugadores de un equipo nos situaremos en la descripción de dicho equipo, vista anteriormente.

10.2.1 Añadir jugador

Si deseamos añadir un nuevo jugador al club donde nos encontramos pulsaremos la opción “Nuevo jugador”, a continuación se nos mostrará el siguiente formulario:

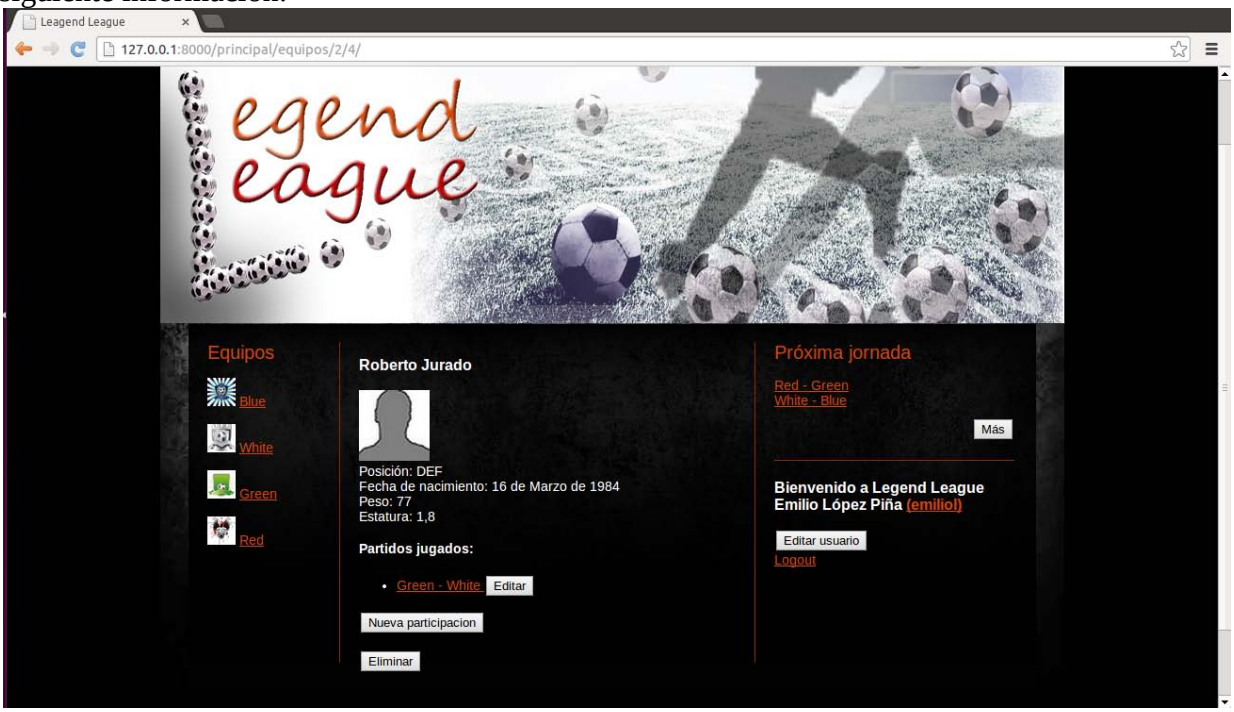


Los campos a rellenar son los siguientes:
- “Nombre”: Nombre y dos apellidos del jugador.
- “Fecha de nacimiento” con formato DD/MM/AÑO
- “Peso” en kilogramos del jugador.
- “Estatura” en metros del jugador.
- “Posición” que ocupa dentro del terreno de juego se podrá elegir entre portero, defensa, mediocentro y delantero.
-”Foto del jugador”: se selecciona una imagen del equipo del usuario en formato .jpg o .png.
-”Equipo”: se deberá seleccionar el equipo donde se encuentra jugando el jugador de entre los registrados en el sistema.

Para confirmar los datos incluidos presionamos el botón “Guardar”, el sistema nos direccionará a la página de información del equipo del que forma parte el jugador.

10.2.2 Visualizar jugador

Para ver la información de un jugador pulsamos sobre el enlace del mismo y aparecerá la siguiente información:



En la parte superior observamos el nombre del jugador.
- Foto del jugador
- “Posición”: aparecerá de la siguiente forma, POR (portero), DEF (defensa), MED (mediocentro), DEL (delantero).
- “Fecha de nacimiento”: veremos la fecha de nacimiento con el siguiente formato “dd de mes de año”.
- “Peso”.
- “Estatura”.

10.2.3 Editar jugador

Para modificar los datos de un jugador, nos situamos en la información del club al que pertenece y presionamos el botón “Editar” asociado al jugador a editar.

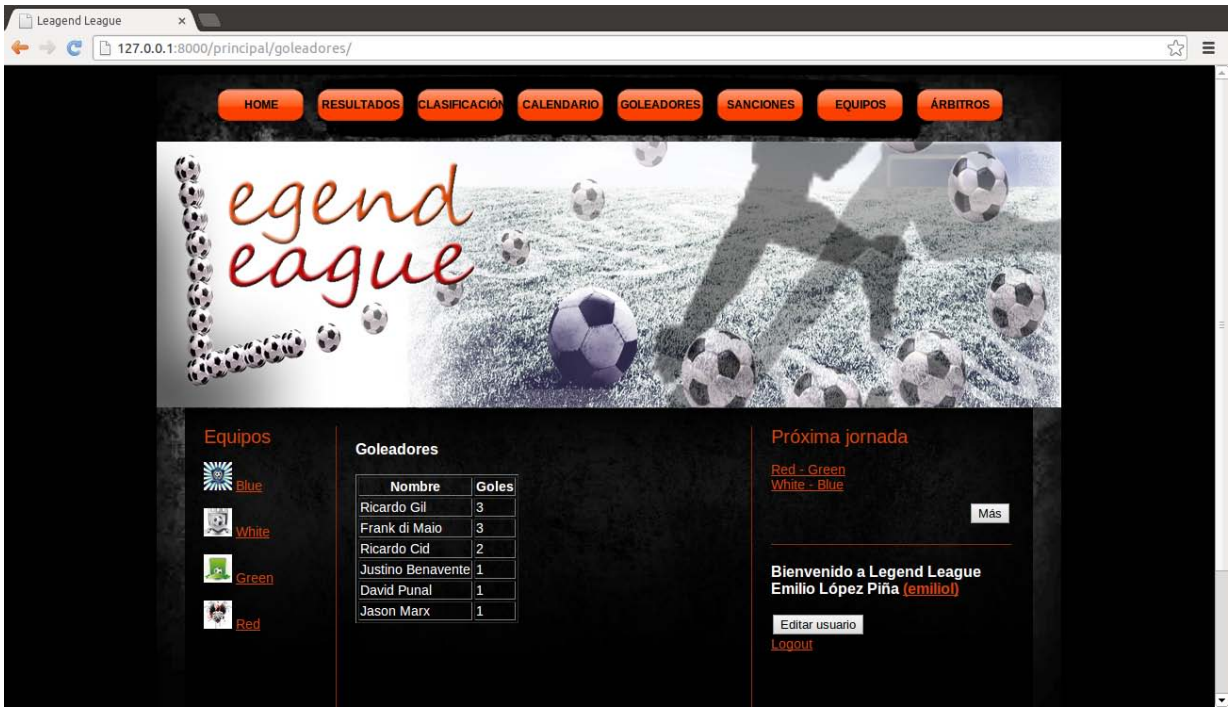
Los campos que aparecen son los mismos que los vistos en la sección anterior.
Para grabar los datos presionamos el botón “Guardar”, el sistema nos redireccionará a la página de información del club.

10.2.4 Eliminar jugador

Para eliminar a un jugador de la base de datos, previamente accedemos a la información del jugador y presionamos el botón “Eliminar”, tras esto el jugador ya no aparecerá y el sistema nos llevará a la página con la información del club.

11. GOLEADORES

Para ver la lista de goleadores de la competición presionamos el botón “Goleadores” de la barra superior.



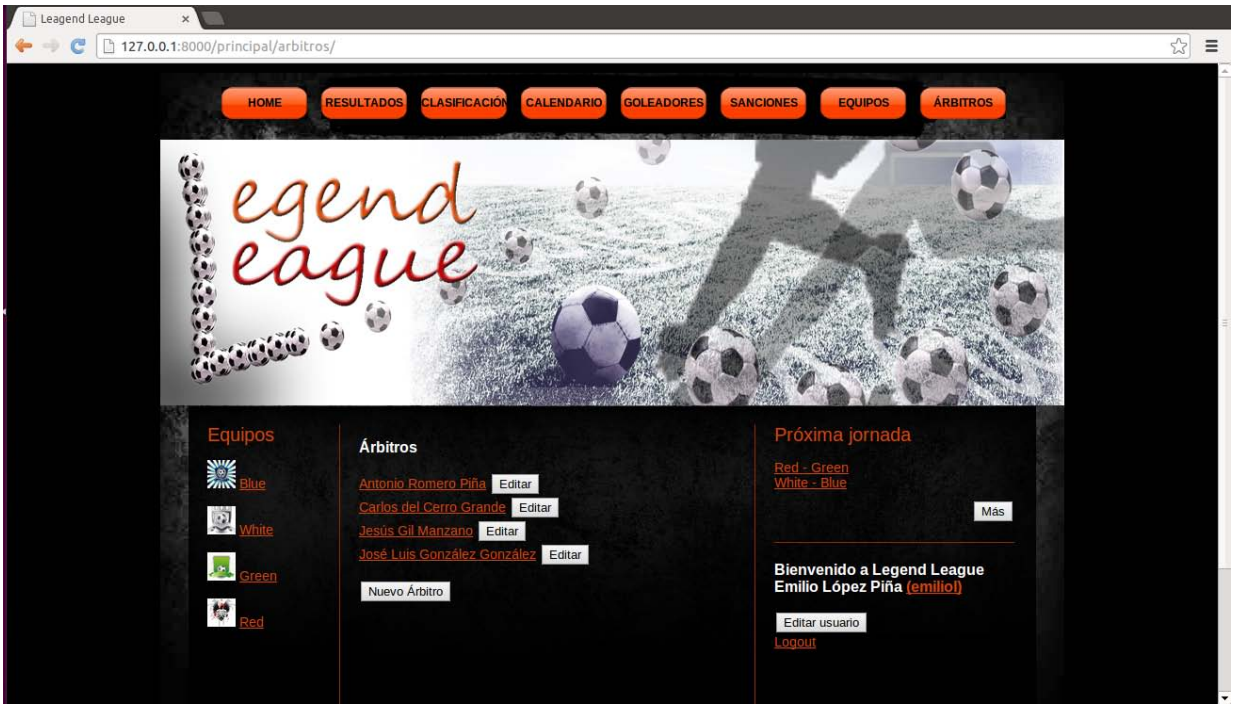
En la página se muestran los nombres de los jugadores que han marcado al menos un gol en la competición seguidos del número de goles anotados.

12. SANCIONES

Al presionar la opción “Sanciones” de la barra superior, el sistema nos mostrará la lista de los jugadores que han sido amonestados durante la competición.

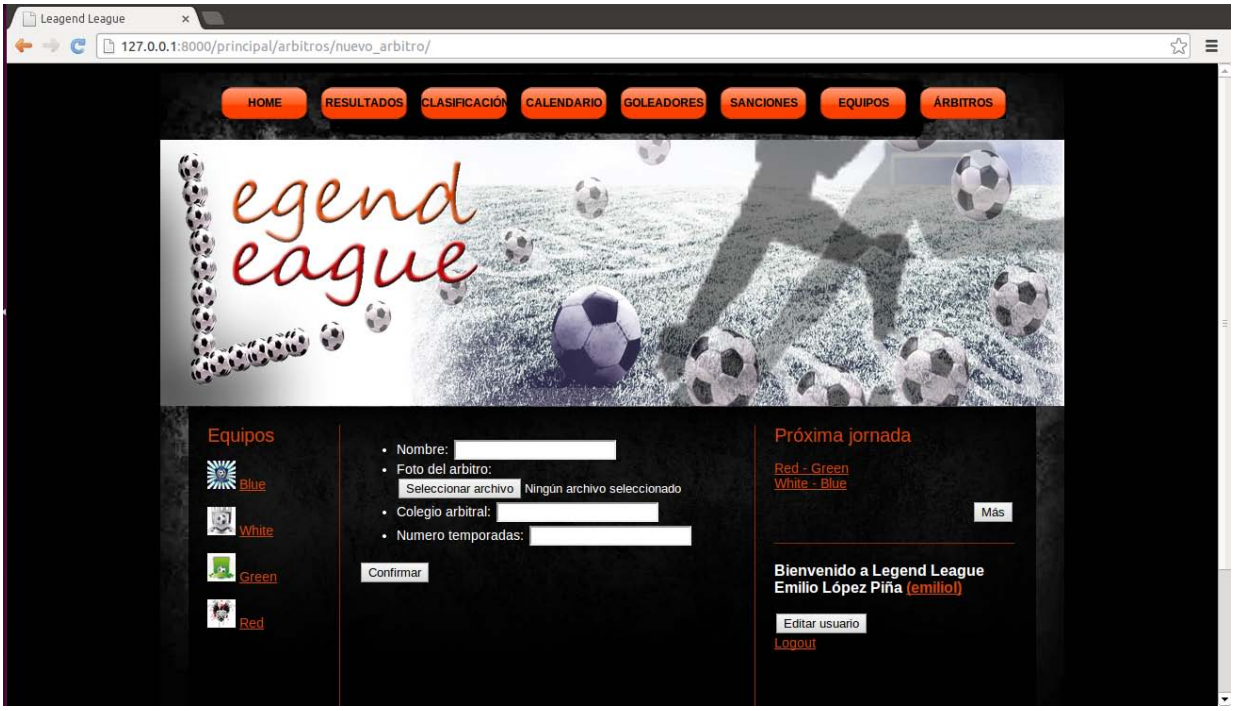
13. ÁRBITROS

Para acceder al menú de árbitros pulsamos el botón “Arbitros” de la barra superior de la pantalla. Aparecerá la siguiente imagen:



13.1 Añadir árbitro

Para crear un árbitro pinchamos el botón “Nuevo Árbitro”, veremos la siguiente imagen:



Deberemos rellenar los siguientes campos:
- "Nombre" y apellidos del árbitro:
- "Foto del árbitro": seleccionar una imagen con formato .jpg o .png.
- "Colegio arbitral": se debe escribir el colegio donde se encuentra inscrito el árbitro.
- "Número temporadas": temporadas en las que ha pitado en la que competición.

Presionamos "Guardar" para grabar los datos del árbitro. El sistema nos llevará al listado de árbitros de la competición.

13.2 Visualizar árbitro

Para ver toda la información de un determinado árbitro presionamos el enlace a dicha información, mostrará la siguiente información:

Los campos incluídos se explican en la sección anterior.

13.3 Editar árbitro

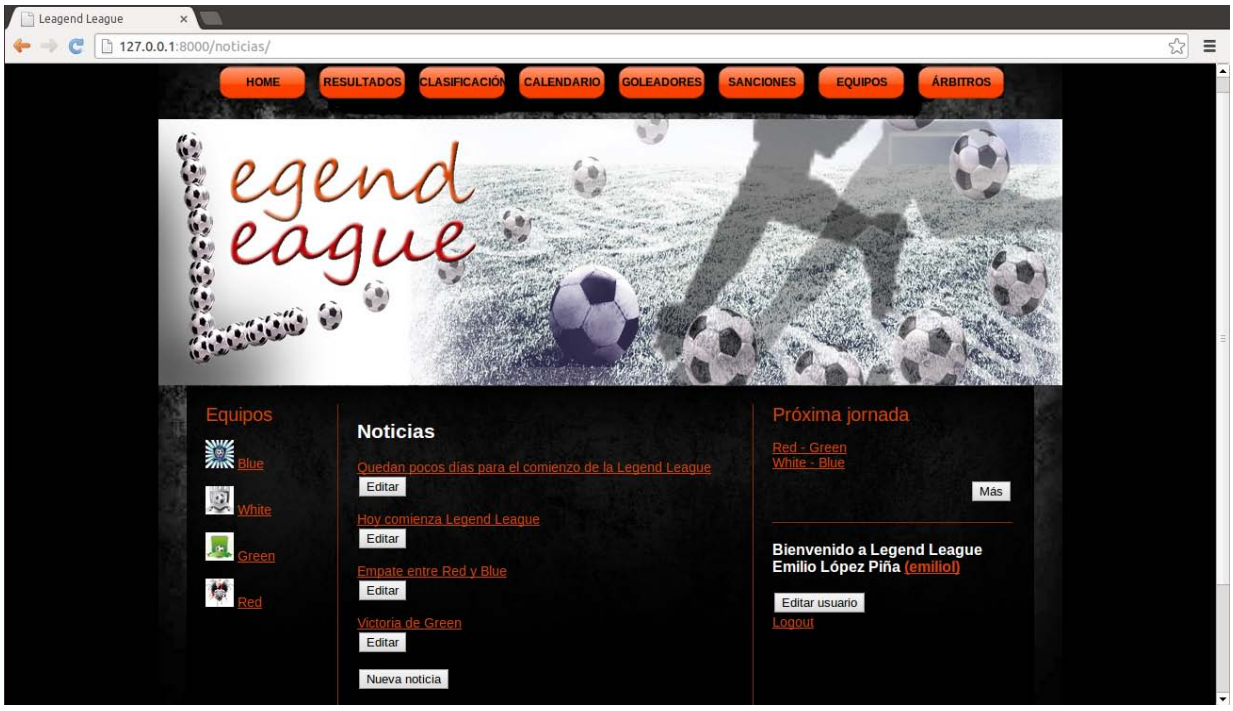
Para modificar la información de un determinado árbitro presionamos el botón “Editar” situado junto al nombre del árbitro en el listado. Aparecerá el siguiente formulario:
(Imagen formulario editar árbitro)
Para guardar los cambios presionamos el botón “Guardar” y el sistema nos redireccionará al listado de árbitros de la competición.

13.4 Eliminar árbitro

Si queremos eliminar del sistema a un árbitro, accedemos a la información del árbitro y presionamos el botón “Eliminar”. La aplicación nos llevará al listado de árbitros.

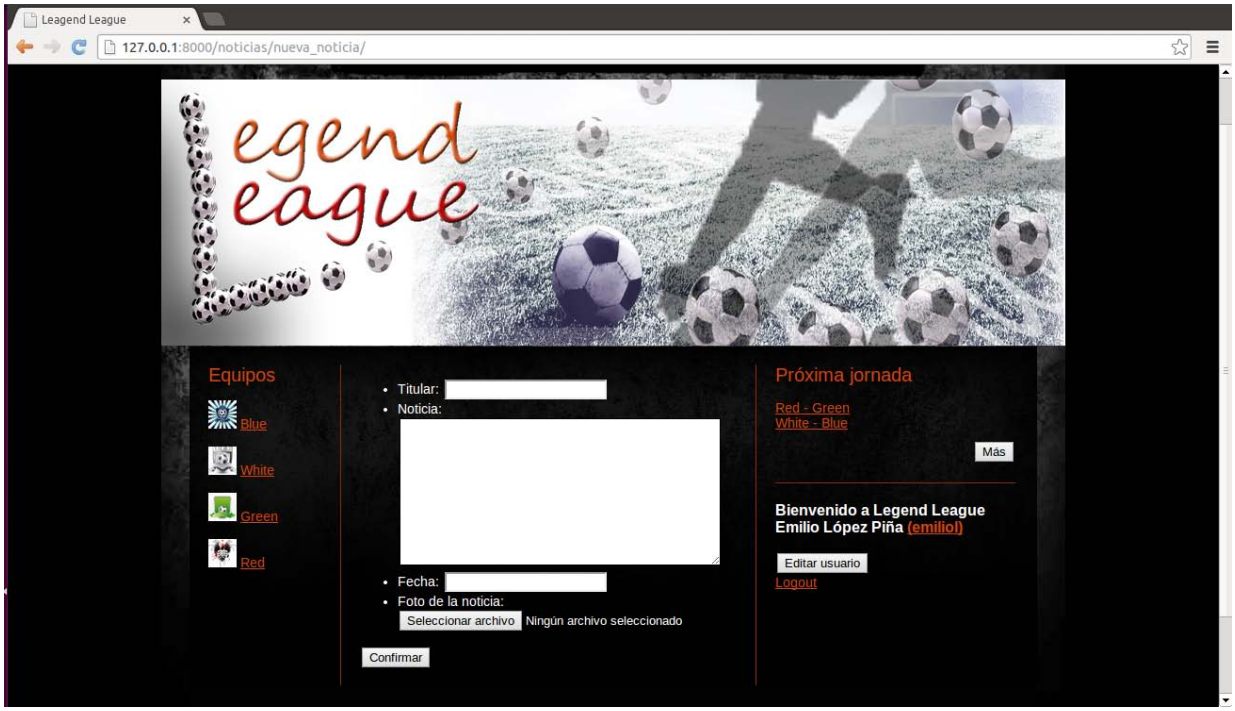
14. NOTICIAS

Para acceder al menú de noticias, debemos presionar el botón “Más” sitiado debajo de las últimas noticias que aparecen en la página inicial.



14.1 Añadir noticia

Una vez en el menú de noticias para añadir una presionamos el botón “Nueva noticia”. Veremos la siguiente imagen:



Rellenamos los siguientes campos:
-"Titular" de la noticia:
- "Noticia": descripción de la noticia.
- "Fecha" de publicación.
- "Imagen" del suceso.

14.2 Visualizar noticia

Para ver una determinada noticia completa presionamos el enlace de la misma del menú principal, el sistema nos llevará a la siguiente página:

Los campos son los mismos que los descritos en la sección anterior.

14.3 Editar noticia

Para cambiar la información de la noticia. Presionamos el botón “Editar” encontrado al lado del titular de la misma, la aplicación nos mostrará el siguiente formulario:
(Imagen editar noticia). Al presionar “Guardar” la web nos llevará al menú de noticias.

14.4 Eliminar noticia

Para eliminar una noticia debemos acceder a la información de la noticia y presionar el botón “Eliminar”. El sistema nos redireccionará al listado de noticias de la aplicación.

MANUAL DE TÉCNICO

El proyecto Leagend League, ha sido realizado mediante Django 1.5, consiste en una aplicación para la gestión de una liga deportiva, en este caso de fútbol.

1. CREACIÓN DEL PROYECTO

Para la creación del proyecto, se ha de introducir desde la línea de comando, cd al directorio en el que se desea almacenar el código del proyecto, y a continuación, ejecutar el siguiente comando:

django-admin.py startproject LegendLeague

Esto crea un directorio *LegendLeague* en el directorio actual. La localización del proyecto creado es la siguiente:

LeagendLeague/

manage.py

LeagendLeague/

__init__.py

settings.py

urls.py

wsgi.py

Estos archivos son:

– El exterior LeagendLeague/: directorio raíz que actúa como contenedor del proyecto.

– Manage.py: utilidad de lína de comandos que permite interactuar con este proyecto de Django de varias maneras.

– El interior /LeagendLeague: directorio real del paquete de Python para el proyecto. Su nombre es que se debe utilizar para importar cualquier información en su interior.

– LeagendLeague/__init__.py: archivo vacío que dice a Python que se debe considerar un paquete de Python.

– LeagendLeague/settings.py: opciones y configuraciones para el proyecto.

– LeagendLeague/urls.py: declaraciones URL para el proyecto.

– LeagendLeague/wsgi.py: punto de entrada para los servidores web compatible con WSGI para servir al proyecto.

1.1 Configuración

A continuación, es necesario editar LeagendLeague/settings.py. Nos disponemos a modificar lo siguiente:

- Declarar la ruta del proyecto para acceder con mayor facilidad a sus archivos:

import os
PROJECT_PATH=os.path.dirname(os.path.realpath(__file__))
NAME: el nombre de la base de datos.

USUARIO: nombre de usuario de base de datos.
CONTRASEÑA: contraseña de base de datos.
TIME_ZONE: establecer la zona horaria. El valor predeterminado es el uso horario central en ls EE.UU. (Chicago).
MEDIA_ROOT: directorio donde almacenar los archivos de imagen del proyecto (os.path.join(PROJECT_PATH,'media')
MEDIA_URL='/media/'
STATIC_URL='/static/'
STATICFILES_DIRS: directorio donde almacenar los archivos de estilos tipos css (os.path.join(PROJECT_PATH,'static')
TEMPLATE_DIRS: directorio que guarda las plantillas de la web (os.path.join(PROJECT_PATH,'templates').
INSTALLED_APPS: aplicaciones implementadas en el proyecto (principal,news).

Cada una de estas aplicaciones hacen uso de al menos una tabla de base de datos, por lo que debemos crear las tablas en la base de datos antes de poder utilizarlas. Para ello se debe ejecutar el siguiente comando:

python manage.py syncdb

El syncdb busca en INSTALLED_APPS y crea las tablas de base de datos necesarios de acuerdo a la configuración de la base de datos en LeagendLeague/settings.py. Veremos un mensaje para cada tabla de base de datos que crea, y se obtendrá un mensaje que pregunta si desea crear una cuenta de superusuario para el sistea de autenticación.

2. LA CREACIÓN DE MODELOS

Ahora que el entorno está configurado, ya está listo para empezar a hacer el trabajo.
En primer lugar se debe crear la aplicación en el mismo directorio que manage.py y escribir los siguiente:
python manage.py startapp principal
python manage.py startapp news

Esto va a crear un directorio *principal* y con newscon la siguiente estructura:
principal/
    __init__.py
    models.py
    tests.py
    views.py

Esta estructura albergará la aplicación principal del proyecto. En primer lugar para escribir una aplicación web de base de datos de Django es necesario definir sus modelos.

Antes de implementar los modelos es necesario definir las entidades, relaciones, atributos y el diagrama E/R.

2.1 Modelo conceptual de la aplicación “principal”:

- Definición de entidades, atributos y atributos identificadores:

Equipo: representa el objeto equipo. Se pueden considerar los atributos: escudo, nombre del equipo, nombre del estadio, adoro, ciudad, nombre del presidente, nombre del entrenador, numero de socios, puntos, partidos jugados, partidos ganados, partidos empatados, partidos perdidos, goles a favor, goles en contra, diferencia de goles y posicion. Considero como identificador de esta entidad nombre del equipo, ya que es único dentro del sistema.

Jugador: representa el objeto persona que milita en un equipo de la competición. Se pueden considerar los atributos: nombre, fecha de nacimiento, peso, estatura, posicion, cara, equipo, goles, tarjetas rojas, tarjetas amarillas, sancionado y lesionado. Considero como identificador de esta entidad el nombre del jugador.

Partido: representa el objeto encuentro que se disputa en una jornada entre dos equipos de fútbol. Se pueden considerar los atributos: nombre de equipo de casa, nombre de equipo de fuera, resultado de equipo de casa, resultado de equipo de fuera, fecha del partido y jornada. Este tipo de entidad es una entidad débil por identificación respecto a la entidad Equipo, ya que, un partido no puede identificarse si no se conocen los dos equipos que lo juegan. Considero como atributos identificadores la agregación de los atributos nombre de equipo de casa y nombre de equipo de fuera.

Arbitro: representa el objeto persona que aplica el reglamento en un partido entre dos equipos. Se pueden considerar los atributos: nombre, cara, colegio arbitral y numero de temporadas. Considero como identificador de esta entidad el nombre.

Jornada: representa el objeto jornada formada por un conjunto de partidos. Se pueden considerar los atributos: numero, fecha inicio jornada y jugada. Considero como identificador de esta entidad el número.

– Definición de relaciones, atributos, cardinalidad, grado de asociación y restricciones semánticas.

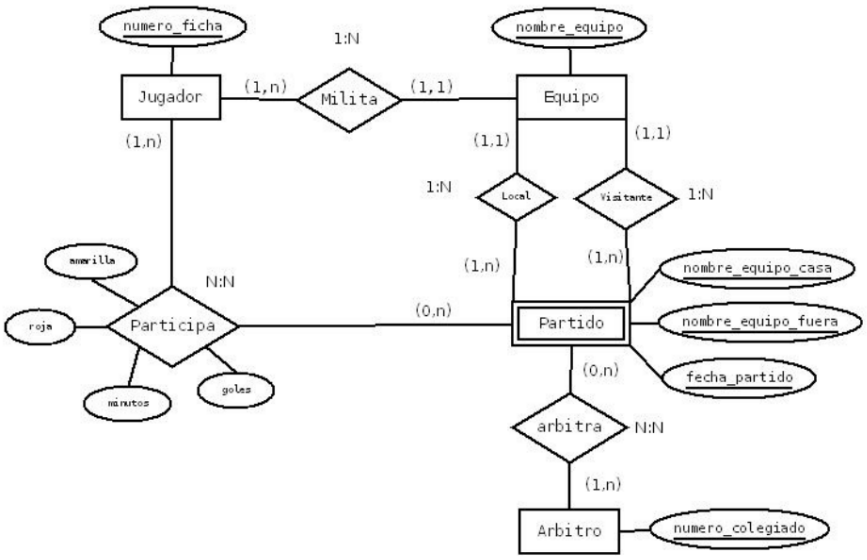
Participa: relaciona los diferentes partidos en los que juega un Jugador, siendo la relación de grado N:N, participando la entidad Jugador con cardinalidad (1,n) y la entidad Partido con cardinalidad (0,n), ya que, un jugador no siempre tiene que participar en un equipo, ya sea porque está sancionado o lesionado. Esta relación esta caracterizada por los atributos: jugador, partido, amarilla, roja, minutos, goles.

Arbitra: relaciona los diferentes partidos en los que arbitra un árbitro, siendo la relación de grado N:N, participando la entidad Arbitro con cardinalidad (1,n) y la entidad Partido con cardinalidad (0,n), ya que, un árbitro no siempre tiene que participar en un partido.

Milita: relaciona la entidad Jugador y Equipo, representando el equipo en el que juega cada uno de los jugadores, siendo la relación de grado 1:N, participando la entidad Jugador con cardinalidad (1,n) y la entidad Equipo con cardinalidad (1,1).

Pertenece: relaciona los diferentes partidos que forman una determinada jornada, siendo la relación de grado 1:N, participando la entidad Partido con cardinalidad (1,n) y la entidad Jornada con cardinalidad (1,N).

– **Esquema entidad-relación (E/R)**



(*) Para facilitar la lectura del esquema entidad-relación sólo indico las entidades, atributos identificadores y atributos que caracterizan una relación, el resto de atributos están indicados en el siguiente apartado.

– Aplicar al modelo conceptual las reglas (tablas, claves primarias, claves foráneas)

Equipo (nombre_equipo, escudo, nombre_estadio, aforo, anyo_fundacion, ciudad, nombre_presidente, nombre_entrenador, numero_socios, puntos, partidos_jugados, partidos_ganados, partidos_empatados, partidos_perdidos, goles_favor, goles_contra, diferencia_goles, posicion);
Jugador (nombre, fecha_nacimiento, peso, estatura, posicion, cara, equipo, goles, tarjetas_rojas, tarjetas_amarillas, sancionado, lesionado);
Arbitro (nombre, cara, colegio_arbitral, numero_temporadas);
Jornada (numero, fecha_inicio, terminada)
Partido (**nombre_equipo_casa, nombre_equipo_fuera**, resultado_equipo_casa, resultado_equipo_fuera, fecha_partido, jornada, arbitro);
Participa (**jugador, partido**, amarilla, roja, minutos, goles);
Arbitra (**colegiado, partido**);

Estos conceptos están representados por las clases de Python simples. El archivo LeagendLeague/principal/models.py se ve así:

class Equipo(models.Model):
nombre_equipo=models.CharField(max_length=30,unique=True)
escudo = models.ImageField(upload_to='escudos', verbose_name='Foto del escudo', default='escudos/esc0.jpg')
nombre_estadio=models.CharField(max_length=50)
aforo=models.IntegerField()
anyo_fundacion=models.IntegerField()
ciudad=models.CharField(max_length=50)
nombre_presidente=models.CharField(max_length=50,unique=True)

```
        nombre_entrenador=models.CharField(max_length=50,unique=True)
        numero_socios=models.IntegerField()
        puntos=models.IntegerField(default=0)
        partidos_jugados=models.IntegerField(default=0)
        partidos_ganados=models.IntegerField(default=0)
        partidos_empatados=models.IntegerField(default=0)
        partidos_perdidos=models.IntegerField(default=0)
        goles_favor=models.IntegerField(default=0)
        goles_contra=models.IntegerField(default=0)
        diferencia_goles=models.IntegerField(default=0)
        posicion=models.IntegerField(unique=True)

        def __unicode__ (self):
                return self.nombre_equipo

class Jugador(models.Model):
        nombre=models.CharField(max_length=30,unique=True)
        fecha_nacimiento=models.DateField()
        peso=models.IntegerField()
        estatura=models.DecimalField(max_digits=3,decimal_places=2)
        posicion=models.CharField(max_length=3,choices=POSITION_CHOICES)
        cara = models.ImageField(upload_to='faces', verbose_name='Foto del jugador',
default='faces/desconocido.jpg')
        equipo=models.ForeignKey(Equipo,related_name='Equipo')
        goles=models.IntegerField()
        tarjetas_rojas=models.IntegerField()
        tarjetas_amarillas=models.IntegerField()
        sancionado=models.BooleanField()
        lesionado=models.BooleanField()

        def __unicode__ (self):
                return self.nombre

class Arbitro(models.Model):
        nombre=models.CharField(max_length=30,unique=True)
        cara = models.ImageField(upload_to='faces', verbose_name='Foto del arbitro',
default='faces/desconocido.jpg')
        colegio_arbitral=models.CharField(max_length=30)
        numero_temporadas=models.IntegerField()

        def __unicode__ (self):
                return self.nombre

class Jornada(models.Model):
        numero=models.CharField(max_length=2,unique=True)
        fecha_inicio=models.DateField()
        terminada=models.BooleanField()

        def __unicode__ (self):
                return self.numero
```

```
class Partido(models.Model):
    nombre_equipo_casa=models.ForeignKey(Equipo,related_name='Equipo_Casa')
    nombre_equipo_fuera=models.ForeignKey(Equipo,related_name='Equipo_Fuera')
    resultado_equipo_casa=models.IntegerField(null = True, blank = True)
    resultado_equipo_fuera=models.IntegerField(null = True, blank = True)
    fecha_partido=models.DateField()
    jornada=models.ForeignKey(Jornada,related_name='Jornada')
    arbitro=models.ForeignKey(Arbitro,related_name='Arbitra')

    def __unicode__ (self):
        return '%s - %s' % (self.nombre_equipo_casa,self.nombre_equipo_fuera)


class Participa(models.Model):
    jugador=models.ForeignKey(Jugador,related_name='Jugador')
    partido=models.ForeignKey(Partido,related_name='Partido participa')
    amarilla=models.IntegerField()
    roja=models.IntegerField()
    minutos=models.IntegerField()
    goles=models.IntegerField()

    def __unicode__ (self):
        return '%s %s' % (self.jugador,self.partido)
```

Cada entidad está representado por una clase. Cada modelo tiene una serie de atributos, cada uno de los cuales representa un campo de la base de datos en el modelo.

2.2 Modelo conceptual de la aplicación “principal”:

- Definición de entidades, atributos y atributos identificadores:

New: representa el objeto noticia. Se pueden considerar los atributos: titular, noticia, fecha e imagen. Considero como identificador des entidad “titular”.

Estos conceptos están representados por las clases de Python simples. El archivo LeagendLeague/news/models.py se ve así:

```
class New(models.Model):
    titular=models.CharField(max_length=200)
    noticia = models.TextField()
    fecha = models.DateField()
    imagen = models.ImageField(upload_to='noticia', verbose_name='Foto de la noticia')
    def __unicode__(self):
        return self.titular
```

2.3 Activación de modelos

Se debe incluír el nombre de la aplicación *principal* en el settings.py en el campo INSTALLED_APPS. Ahora vamos a ejecutar el comando:

python manage.py principal sql

De nuevo, ejecutar syncdb para crear los modelos de cuadro en la base de datos:
python manage.py syncdb
El comando syncdb ejecuta el SQL desde *sql* en su base de datos para todas las aplicaciones que aún no existen en su base de datos. Esto crea todas las tablas, los datos iniciales y los indices para cualquier aplicación que se ha añadido al proyecto desde la última vez que se ejecutó syncdb.

Es de señanar la adición de __unicode__() al modelo para la representación de la información de forma más lógica y porque las representaciones de objetos se utiliza a lo largo de administración generada automáticamente de Django.

2.4 Activar el sitio de administración

El sitio de administración de Django no está activado por defecto. Para activar el sitio de administración para su instalación hay que hacer lo siguiente:
– Eliminar el comentario de “django.contrib.admin” en “INSTALLED_APPS”.
– Ejecutar python manage.py syncdb, ya que se ha agregado una nueva aplicación para INSTALLED_APPS.
– Editar LegendLeague/urls.py y descomentar las tres líneas que hacen referencia al admin.

El sitio de administración de Django lee los metadatos de su modelo para proporcionar una interfaz potente y lista para producción que los productores de contenido pueden usar para comenzar a agregar contenido al sitio, una vez que se ejecuta el siguiente comando:

python manage.py runserver

3. VIEWS, URLS Y FORMS

Cada una de las vistas del proyecto cumple una función determinada y tiene una plantilla específica. Las aplicación *principal* del proyecto está formada por las siguientes vistas:
– Página “index”: muestra el menú principal de la aplicación, últimas noticias y resultados.
– “userLogin”: realiza la función de iniciar sesión, muestra un formulario para tal acción.
– “userDetail”: muestra toda la información del perfil de un determinado usuario.
– “editarUsuario”: muestra un formulario para modificar la información de un determinado usuario.
– “eliminarUsuario”: borra todos los atributos de un usuario de la base de datos.
– “JornadaList”: muestra la lista de jornadas que forman el calendario de la competición.
– “NuevaJornada”: página cuya función es mostrar un formulario para añadir una nueva jornada al calendario.
– “JornadaDetail”: muestra los partidos que forman una determinada jornada.
– “EditJornada”: muestra un formulario similar al de “NuevaJornada” para modificar la información anteriormente inscrita.
– “JornadaDelete”: borra una determinada dada por la url.
– “ProximaJornada”: muestra la jornada de valor mínimo que aún no se haya disputado.

– “NuevoPartido”: añade un partido a una jornada encontrada en el sistema.
– “PartidoDetail”: muestra todos los detalles de un determinado partido de la jornada.
– “EditPartido”: muestra un formulario para modificar la información almacenada sobre un partido.
– “PartidoDelete”: borra los datos de un partido de la base de datos.
– “NuevoJugador”: muestra un formulario para añadir un nuevo jugador a la plantilla de un equipo previamente registrado en la aplicación.
– “EditarJugador”: muestra un formulario para modificar la información almacenada sobre un jugador cualquiera.
– “JugadorDelete”: borra un jugador del sistema.
– “Clasificación”: vista que muestra el listado de equipos registrados según un orden establecido por una puntuación asignada en función de su rendimiento en la competición.
– “Goleadores”: muestra todos los jugadores que hayan marcado al menos un gol en lo que va de competición.
– “Sanciones”: muestra la lista de jugadores que han recibido al menos una tarjeta.
– “EquipoList”: página que muestra un listado de los equipos registrados en la base de datos.
– “EquipoDetail”: muestra los detalles de un determinado equipo (datos del club, plantilla...).
– “JugadorDetail”: informa de todos los detalles de un jugador.
– “ArbitroList”: muestra la lista de los arbitros que pitan los partidos de la competición.
– “ArbitroDetail”: muestra la información principal de un determinado árbitro.
– “NuevoArbitro”: muestra un formulario con el que rellenar la información para añadir un árbitro al sistema.
– “ParticipaDetail”: muestra los detalles de la actuación de un jugador en un partido.
– “ParticipaNuevo”: crea una participación.
– “ParticipaEditar”: modifica una participación mediante un formulario.
– “ParticipaDelete”: borra una participación de un jugador en un partido.
– “EditarArbitro”: muestra un formulario similar al de “NuevoArbitro” con el fin de modificar la información añadida previamente.
– “Logout”: finaliza la sesión del usuario que haya iniciado sesión previamente.
– “NuevoUsuario”: muestra un formulario para añadir un usuario al sistema.
– “NuevoEquipo”: añade un nuevo equipo a la base de datos.
– “edit_team”: modifica la información de un determinado equipo.

Las aplicación *news* del proyecto está formada por las siguientes vistas:
– “NoticiaList”: muestra las noticias de la competición identificadas por la fecha de publicación y el titular.
– “NoticiaDetail”: muestra los campos de la clase New declarada anteriormente en el models.py.
– “nuevaNoticia”: llama a un formulario para crear una noticia.
– “editarNoticia”: muestra el mismo formulario de antes con el fin de modificar los datos de una determinada noticia.
– “eliminarNoticia”: elimina una noticia de la base de datos.

A continuación vamos una vista con su respectiva urls de cada tipo de las implementadas, ya que son muchas las que se han incluído en el proyecto y su diseño e implementación son muy parecidos.

3.1 Vista inicio

Para abrir la primera vista abrimos el archivo principal/views.py y escribimos lo siguiente:

```
from django.shortcuts import render,redirect

def inicio(request):
        return render(request,'index.html',context)
```

La vista llama al archivo 'index.html' encontrado en el archivo previamente definido para tal fin en el settings.py.

Para llamar a la vista, hay que asignarla a una URL, para ello hay que tener un archivo llamado urls.py. Dicho archivo incluirá el siguiente código:

```
from django.conf.urls import patterns, include, url

urlpatterns = patterns('',
        url(r'^$','principal.views.inicio')
)
```

Como se ve en la url la vista inicio se encuentra en el archivo views.py de la aplicación “principal”.

3.2 Vista crear usuario

Para crear la vista para crear un nuevo usuario realizamos la siguiente llama en el urls.py:
url(r’^principal/users/nuevo_usuario/$’,’principal.views.NuevoUsuario’,name=’new_user’)

Con esta línea hacemos uso de la vista “NuevoUsuario”:

```
Def NuevoUsuario(request):
   List=Equipo.objects.all()
   jornada_id=Jornada.objects.filter(terminada=False).annotate(Max(‘numero’))
  partidos=Partido.objects.filter(jornada=jornada_id)
 if request.method==’POST’:
   form=MyUserForm(request.POST)
  if form.is_valid():
     form.save()
   return redirect(‘/’)
else:
   form=MyUserForm()
context={‘form’:form,’equipoList’:list,’partidoList’:partidos}
return render(request,’newuser.html’,context)
```

En las tres primeras líneas de la función se recogen los objetos de la base de datos para imprimir tanto la lista de equipos como los próximos encuentros de la competición (esta información aparece en la mayoría de las vistas). En la siguientes líneas se llama al formulario “MyUserForm” encontrado en el archivo forms.py de la aplicación. Si la información que el usuario le pasa al formulario es correcta, se guarda y el sistema redirecciona a la raíz de la aplicación web. Toda esta información se presenta en el documento “newuser.html” encontrado en la carpeta “plantillas” del proyecto. En esta vista se hacen una serie de llamadas a la base de datos para obtener las características más importantes de un partido, se guardan en distintas variables y éstas se pasan al fichero “partido.html” para ser mostradas en la plantilla.

El formulario incluído en el archivo forms.py es el siguiente:

```
class MyUserForm(UserCreationForm):
    """A form for creating new users. Includes all the required fields, plus a repeated password."""
    password1 = forms.CharField(label='Password', widget=forms.PasswordInput)
    password2 = forms.CharField(label='Password confirmation', widget=forms.PasswordInput)

    class Meta:
        model = User
        fields = ('first_name', 'last_name', 'username', 'email')

    def clean_username(self):
        username = self.cleaned_data["username"]
        try:
            User.objects.get(username = username)
        except User.DoesNotExist:
            return username
        raise forms.ValidationError("A user with this user name already exists.")

    def clean_password2(self):
        password1 = self.cleaned_data.get("password1", "")
        password2 = self.cleaned_data["password2"]
        if password1 != password2:
            raise forms.ValidationError("The two password fields did not match.")
        return password2

    def save(self, commit=True):
        user = super(UserCreationForm, self).save(commit=False)
        user.set_password(self.cleaned_data["password1"])
        user.is_staff = False
        user.is_superuser = False
        if commit:
            user.save()
        return user
```

3.3 Vista iniciar sesión

Para diseñar la opción de iniciar sesión sigo los siguientes pasos:
En primer lugar, añado la siguiente línea url que hace uso de la vista “userLogin”

```
url(r'^users/login$','principal.views.userLogin',name='login')
```

La función userLogin incluída en el views.py es la siguiente:

```
Def userLogin(request):
    List=Equipo.objects.all()
    jornada_id=Jornada.objects.filter(terminada=False).annotate(Max('numero'))
   partidos=Partido.objects.filter(jornada=jornada_id)
   if request.method=='POST':
```

```
    form=AuthenticationForm(request.POST,request.FILES)
   if form.is_valid:
       user=request.POST['username']
      passwd=request.POST['password']
     access=authenticate(username=user,password=passwd)
    if access is not None:
      if access.is_active:
         login(request,access)
        return redirect('/')
     else:
       return render(request,'inactive.html')
   else:
     return render(request.'nouser.html')
else:
   form=AuthenticationForm()
context={'form':form,'equipoList':list,'partidoList':partidos}
return render(request,'login.html',context)
```

En esta vista se hace uso de un formulario ya predefinido como es AuthenticationForm y sólo es necesario declararlo en la cabecera del fichero. En este caso si los datos no se encuentran se llama al archivo "nouser.html" con un mensaje de error. Y en el caso de que el usuario no esté activo se llamará al archivo "inactive.html".

3.4 Vista detalles de un partido

Un ejemplo de vista para ver la descripción de los atributos de un determinado objeto previamente creado es la de "PartidoDetail". Para ver los atributos incluidos en esta vista previamente se ha de haber creado un partido mediante la vista "NuevoPartido" que llama al formulario PartidoForm.

A la vista "PartidoDetail" se le asigna la siguiente url:

```
url(r`^principal/calendario/jornada(?P<jornada_id>\d+)/(?Ppartido_id>\d+)$','principal.views.
PartidoDetail)
```

La vista "PartidoDetail" es la siguiente:

```
Def PartidoDetail(request,jornada_id,partido_id):
   list=Equipo.objects.all()
   jornada_id=Jornada.objects.filter(terminada=False).annotate(Max('numero))
   partidos=Partido.objects.filter(jornada=jornada_id)
   goles=Participa.objects.filter(partido=partido_id,goles__gt=0)
   amarillas=Participa.objects.filter(partido=partido_id,amarilla__gt=0)
   rojas=Participa.objects.filter(partido=partido_id,roja__gt=0)
   partido=Partido.objects.get(pk=partido_id)
   estadio=Partido.objects.get(id=1).nombre_equipo_casa.nombre_estadio
   equipos=Equipo.objects.all()

context={'golesList':goles,'amarillasList':amarillas,'rojasList':rojas,'partido':partido,'estadio':es
tadio,'equipoList':equipos,'partidoList':partidos}
return render(request,'partido.html',context)
```

3.5 Vista editar árbitro

Un ejemplo de vista para editar información es la de EditarArbitro. La línea url que se añade es la siguiente:

url(r'^principal/árbitros/(?P<arbitro_id>\d+)/edit/$','principal.views.EditarArbitro)

La función "EditarArbitro" es la siguiente:

```
def EditarArbitro(request,arbitro_id):
    list=Equipo.objects.all()
    jornada_id=Jornada.objects.filter(terminada=False).annotate(Max('numero'))
    partidos=Partido.objects.filter(jornada=jornada_id)
    arbitro=get_object_or_404(Arbitro,pk=arbitro_id)
    if reques.method=='POST':
        form=ArbitroForm(request.POST,request.FILES,instance=arbitro)
        if form.is_valid():
            form.save()
            return HttpResponseRedirect("/principal/arbitros/")
    else:
        form=ArbitroForm(instance=arbitro)

context={'form':form,'equipoList':list,'partidoList':partidos}
return render(request,'artibro.html',context)
```

Como se puede apreciar esta vista es similar a la de crear, excepto por el uso de la variable "instance" para guardar la información del objeto que se edita.

3.6 Vista borrado

Un ejemplo de borrado es la vista "eliminarJugador" que borra de la base de datos una noticia antes creada.

En primer lugar, se crea la url:

url(r'^noticias/(?P<noticia_id>\d+)/eliminar_noticia/$','news.views.eliminarNoticia')

La vista "eliminarJugador" es la siguiente:

```
def JugadorDelete(request,equipo_id,jugador_id):
    jugador=get_object_or_404(Jugador,pk=jugador_id)
    jugador.delete()
    return redirect('principal/equipos/%'%equipo_id)
```

En esta vista guarda el jugador con el id dado por el url en la variable "jugador" y se borra llamando a la función "delete()" tras borrar, la aplicación redirecciona a la página del equipo.