Práctica 5: Módulos Odoo Controlador, Herencia y Web Controllers.

Sistemas de Gestión Empresarial

Cristian Fernández

30 de enero de 2026

Índice

1. Introducción

El objetivo de esta práctica es aplicar los conocimientos adquiridos sobre vistas, wizards, informes y controladores web. Para ello, nos ocuparemos de la implementación de cuatro actividades que abarcan diferentes contextos:

- **EJ07-LigaFutbol** *4 puntos*
  - Se modificará el módulo para que incluya las siguientes funcionalidades:
    - Reglas de puntuación especiales. *1 punto*
    - Botones para alterar los goles de los partidos. *1 punto*
    - Web controller para eliminar empates. *0,5 punto*
    - Informe PDF por cada partido. *0,5 punto*
    - Wizard para crear nuevos partidos. *0,5 punto*
    - Vista Graph. *0,5 punto*
- **EJ08-API-REST_Socios** *1,5 puntos*
  - Se grabará un vídeo explicativo realizando operaciones CRUD.
- **Bot de Telegram** *3 puntos*
  - Se creará un bot de Telegram que escuchará ordenes enviadas por los usuarios.
- **Generación de imágenes aleatorias con Web Controller** *1,5 puntos*
  - Se creará un *Web Controller* que reciba parámetros, genere una imagen de píxeles aleatorios y la devuelva en Base64 o PNG.

Los ejemplos citados anteriormente se encuentran en el siguiente repositorio.
https://github.com/sergarb1/OdooModulosEjemplos

2. Actividad 01 – Modificación *EJ07-LigaFutbol*

2.1. Reglas de puntuación especiales

Se solicita modificar la lógica de puntuación de los partidos para que, en un partido con 4 o más goles de diferencia, el ganador se lleve 7 puntos y al perdedor se le reste 1.

Antes de comenzar será necesario modificar el código de las vistas para que el módulo se pueda instalar correctamente en Odoo. Deberemos sustituir las etiquetas <tree>por <list>tal como se ve en la siguiente imagen:

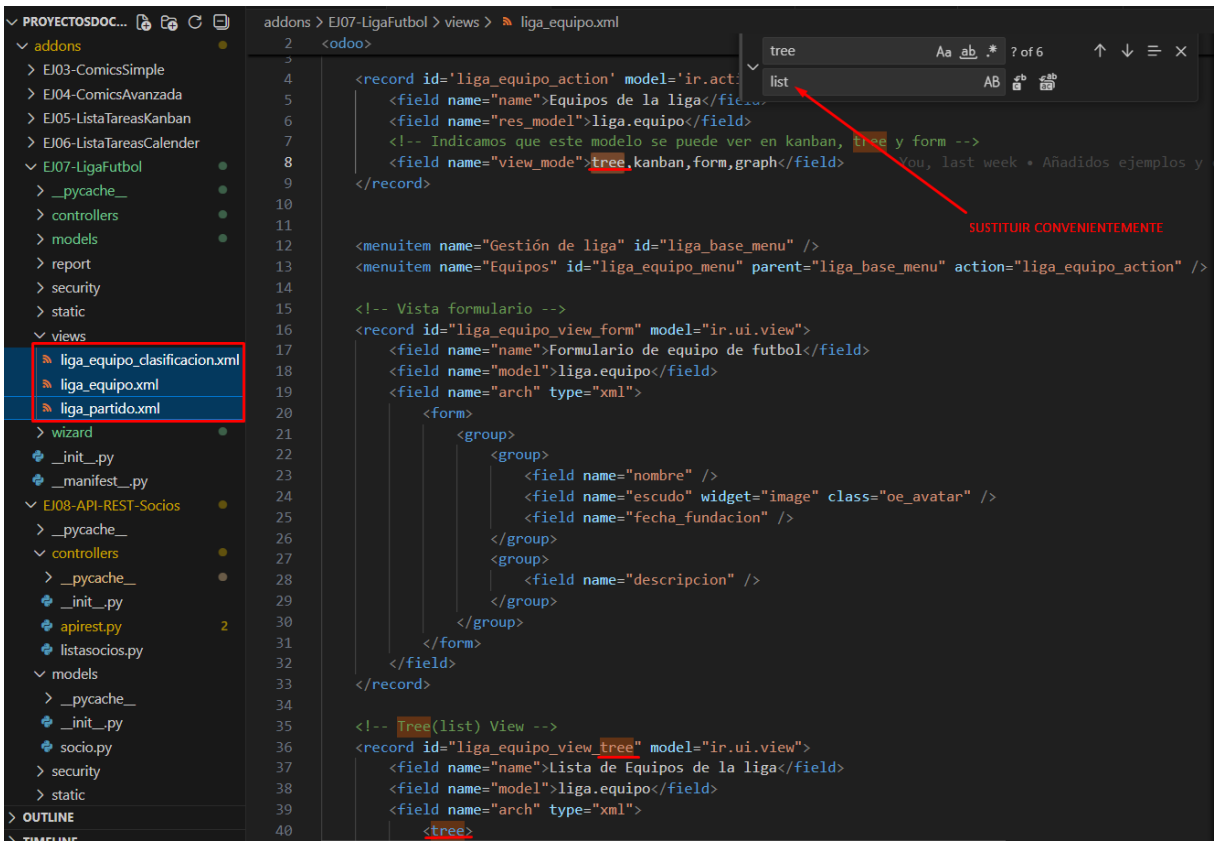


Figura 1: Aplicar estos cambios en las 3 vistas

Una vez instalado el módulo accedemos a él y vemos lo siguiente:

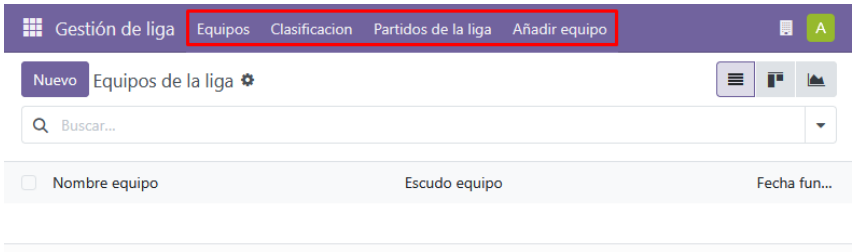


Figura 2: Las 4 vistas del módulo

Tras añadir los 10 primeros equipos de la primera división española nos quedará así:

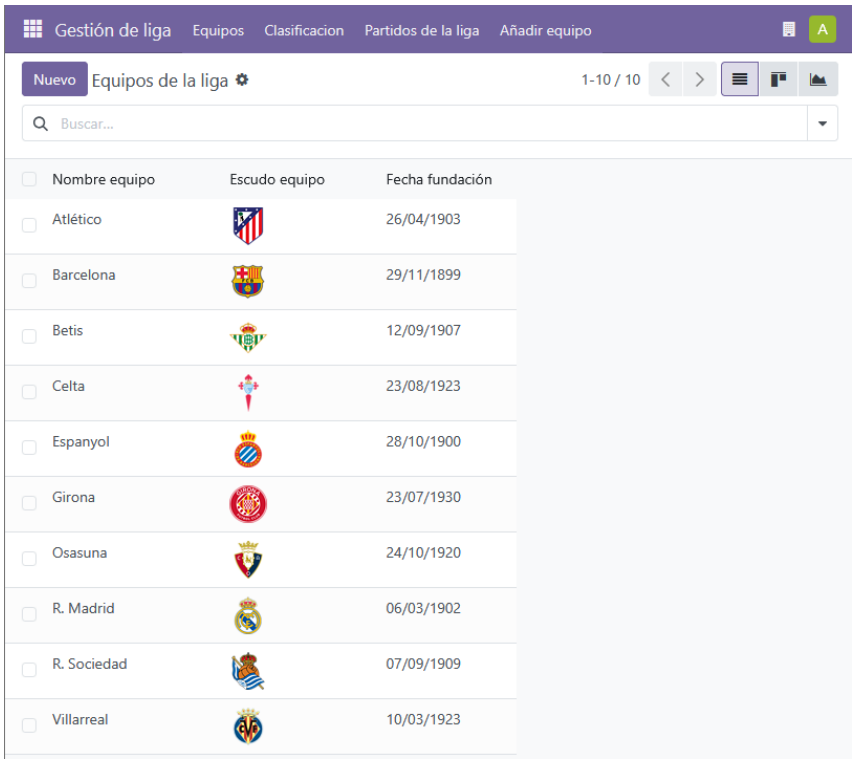


Figura 3: Podemos simular varios partidos de prueba...

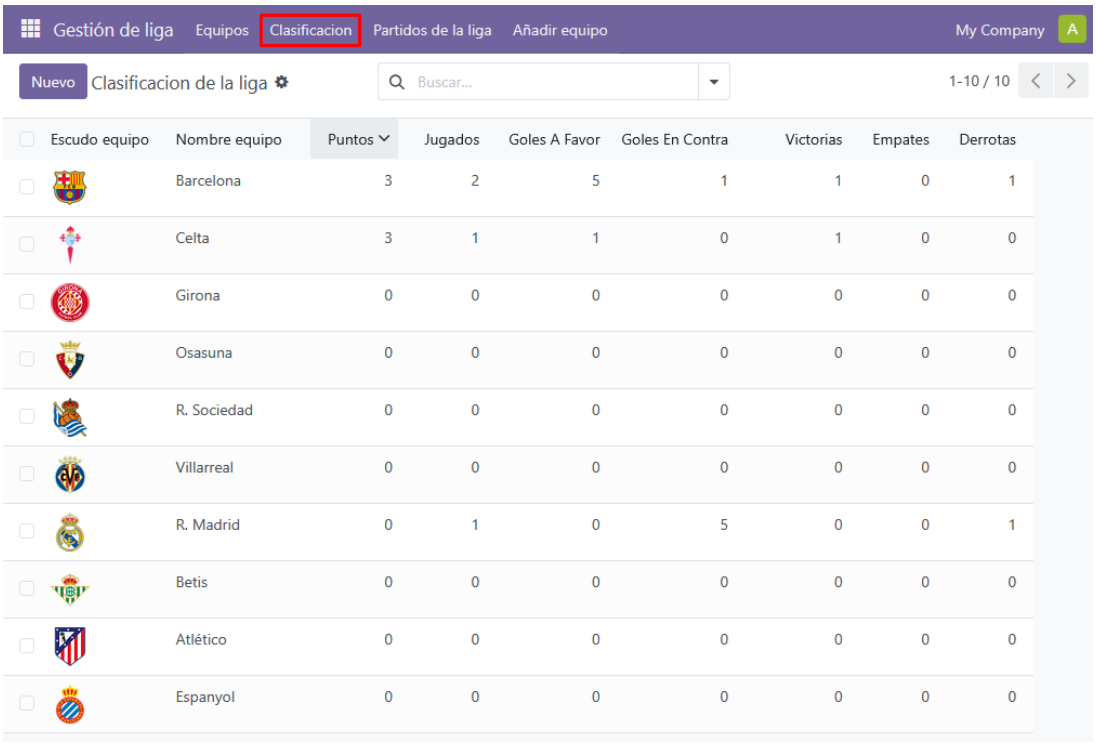


Figura 4: Para ver la puntuación genérica en la clasificación general

Ahora echémosle un vistazo al código del módulo. Básicamente nos encontramos con 2 modelos, *liga_partido.py* y *liga_equipo.py*. En este último se calcula la puntuación con un campo computado llamado puntos tal como se ve en la siguiente imagen:

```
55
56      puntos= fields.Integer( compute="_compute_puntos",default=0, store=True)
57
58      @api.depends('victorias','empates')
59      def _compute_puntos(self):
60          for record in self:
61              record.puntos = record.victorias * 3 + record.empates
62
```

Dada la lógica que queremos implementar para la puntuación vamos a convertir ese campo computado en uno normal, tal que así:

```
56      puntos = fields.Integer(default=0)
57
58      # @api.depends('victorias','empates')
59      # def _compute_puntos(self):
60      #     for record in self:
61      #         record.puntos = record.victorias * 3 + record.empates
62
```

Cambiamos de modelo y nos vamos a *liga_partido.py* y añadimos el campo creado en el paso anterior dentro de la función *actualizoRegistrosEquipo()*:

```
62      def actualizoRegistrosEquipo(self):
63          #Recorremos partidos y equipos
64          for recordEquipo in self.env['liga.equipo'].search([]):
65              #Como recalculamos todo, ponemos de cada equipo todo a cero
66              recordEquipo.victorias=0
67              recordEquipo.empates=0
68              recordEquipo.derrotas=0
69              recordEquipo.goles_a_favor=0
70              recordEquipo.goles_en_contra=0
71              recordEquipo.puntos=0 <-------- AÑADIMOS CAMPO PUNTOS
```

En la misma función deberemos modificar todo este código...

```
73              for recordPartido in self.env['liga.partido'].search([]):
74
75                  #Si es el equipo de casa
76                  if recordPartido.equipo_casa.nombre==recordEquipo.nombre:
77
78                      #Miramos si es victoria o derrota
79                      if recordPartido.goles_casa>recordPartido.goles_fuera:
80                          recordEquipo.victorias=recordEquipo.victorias+1
81                      elif recordPartido.goles_casa<recordPartido.goles_fuera:
82                          recordEquipo.derrotas=recordEquipo.derrotas+1
83                      else:
84                          recordEquipo.empates=recordEquipo.empates+1
85
86                      #Sumamos goles a favor y en contra
87                      recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_casa
88                      recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_fuera
89
90                  #Si es el equipo de fuera
91                  if recordPartido.equipo_fuera.nombre==recordEquipo.nombre:
92
93                      #Miramos si es victoria o derrota
94                      if recordPartido.goles_casa<recordPartido.goles_fuera:
95                          recordEquipo.victorias=recordEquipo.victorias+1
96                      elif recordPartido.goles_casa>recordPartido.goles_fuera:
97                          recordEquipo.derrotas=recordEquipo.derrotas+1
98                      else:
99                          recordEquipo.empates=recordEquipo.empates+1
100
101                     #Sumamos goles a favor y en contra
102                     recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_fuera
103                     recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_casa
```

Figura 5: Código original...

Por este otro en el que añadiremos la nueva lógica de puntuación, tal como se muestran en las capturas de la página siguiente:

```
73              for recordPartido in self.env['liga.partido'].search([]):
74
75                  # AÑADIMOS NUEVA VARIABLE PARA SIMPLIFICAR LOS IF
76                  goles_de_diferencia = abs(recordPartido.goles_casa - recordPartido.goles_fuera)
77
78                  #Si es el equipo de casa
79                  if recordPartido.equipo_casa.nombre==recordEquipo.nombre:
80
81                      #Miramos si es victoria o derrota
82                      if recordPartido.goles_casa>recordPartido.goles_fuera:
83                          recordEquipo.victorias=recordEquipo.victorias+1
84                          # APLICAMOS LA LÓGICA NUEVA (VICTORIAS)
85                          if goles_de_diferencia >= 4:
86                              recordEquipo.puntos += 7
87                          else:
88                              recordEquipo.puntos += 3
89
90                      elif recordPartido.goles_casa<recordPartido.goles_fuera:
91                          recordEquipo.derrotas=recordEquipo.derrotas+1
92                          # APLICAMOS LA LÓGICA NUEVA (DERROTAS)
93                          if goles_de_diferencia >= 4:
94                              recordEquipo.puntos -= 1
95                          else:
96                              pass
97
98                      else:
99                          recordEquipo.empates=recordEquipo.empates+1
100                         # AÑADIMOS PUNTUACIÓN ESTÁNDAR EN CASO DE EMPATE
101                         recordEquipo.puntos += 1
102
103                     #Sumamos goles a favor y en contra
104                     recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_casa
105                     recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_fuera
106
```

Figura 6: Código para equipo local

```
107                 #Si es el equipo de fuera
108                 if recordPartido.equipo_fuera.nombre==recordEquipo.nombre:
109
110                     #Miramos si es victoria o derrota
111                     if recordPartido.goles_casa<recordPartido.goles_fuera:
112                         recordEquipo.victorias=recordEquipo.victorias+1
113                         # APLICAMOS LA LÓGICA NUEVA (VICTORIAS)
114                         if goles_de_diferencia >= 4:
115                             recordEquipo.puntos += 7
116                         else:
117                             recordEquipo.puntos += 3
118
119                     elif recordPartido.goles_casa>recordPartido.goles_fuera:
120                         recordEquipo.derrotas=recordEquipo.derrotas+1
121                         # APLICAMOS LA LÓGICA NUEVA (DERROTAS)
122                         if goles_de_diferencia >= 4:
123                             recordEquipo.puntos -= 1
124                         else:
125                             pass
126
127                     else:
128                         recordEquipo.empates=recordEquipo.empates+1
129                         # AÑADIMOS PUNTUACIÓN ESTÁNDAR EN CASO DE EMPATE
130                         recordEquipo.puntos += 1
131
132                     #Sumamos goles a favor y en contra
133                     recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_fuera
134                     recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_casa
135
```

Figura 7: Código para equipo visitante

Es necesario matizar que, según esta lógica, cuando un partido se resuelve con una diferencia de goles igual o superior a 4, el equipo perdedor recibirá una puntuación negativa de -1. Esto quiere decir que puede terminar la temporada con puntuación negativa.

Ahora echemos un vistazo al módulo de Odoo para ver si se han efectuado correctamente las modificaciones. Sobra decir que es necesario reiniciar el contenedor para que se apliquen los cambios.

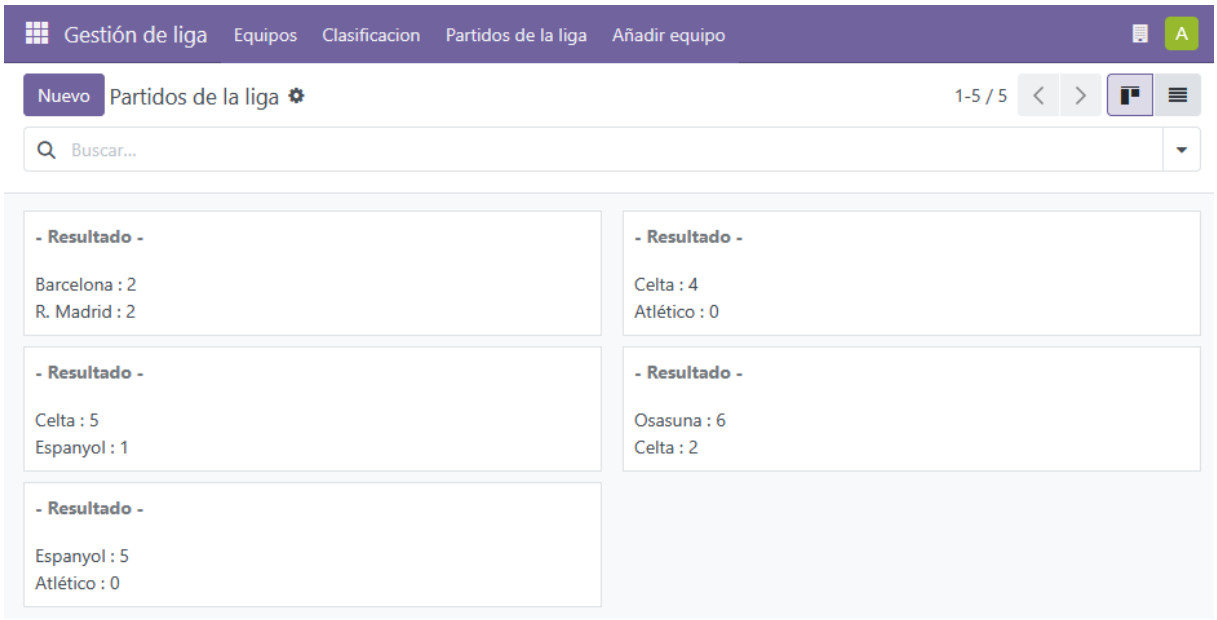
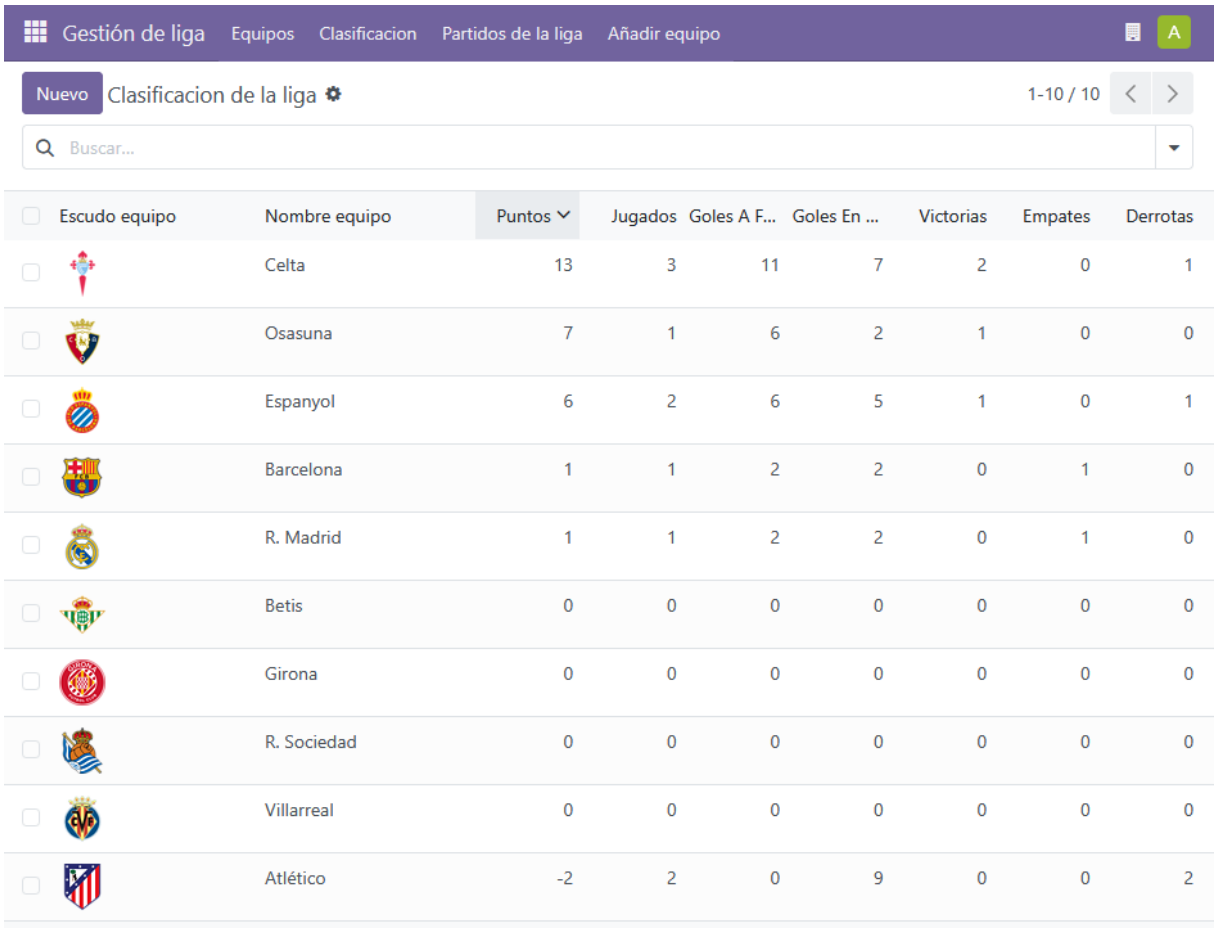


Figura 8: Crearemos nuevos partidos de prueba



Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A F...	Goles En ...	Victorias	Empates	Derrotas
	Celta	13	3	11	7	2	0	1
	Osasuna	7	1	6	2	1	0	0
	Espanyol	6	2	6	5	1	0	1
	Barcelona	1	1	2	2	0	1	0
	R. Madrid	1	1	2	2	0	1	0
	Betis	0	0	0	0	0	0	0
	Girona	0	0	0	0	0	0	0
	R. Sociedad	0	0	0	0	0	0	0
	Villarreal	0	0	0	0	0	0	0
	Atlético	-2	2	0	9	0	0	2

Figura 9: Comprobamos que el sistema de puntuación se aplica correctamente

2.2. Botones para alterar los goles de los partidos

Se nos pide modificar la vista de los partidos para añadir 2 nuevos botones:

- **Botón "+2 goles equipos locales"** Sumará 2 goles a todos los equipos locales.
- **Botón "+2 goles equipos visitantes"** Sumará 2 goles a todos los equipos visitantes.

Tras cada acción la clasificación deberá actualizarse.

Lo primero que haremos será añadir 2 funciones de actualización en el modelo correspondiente (*liga_partido.py*) que posteriormente asignaremos a los botones de la vista.

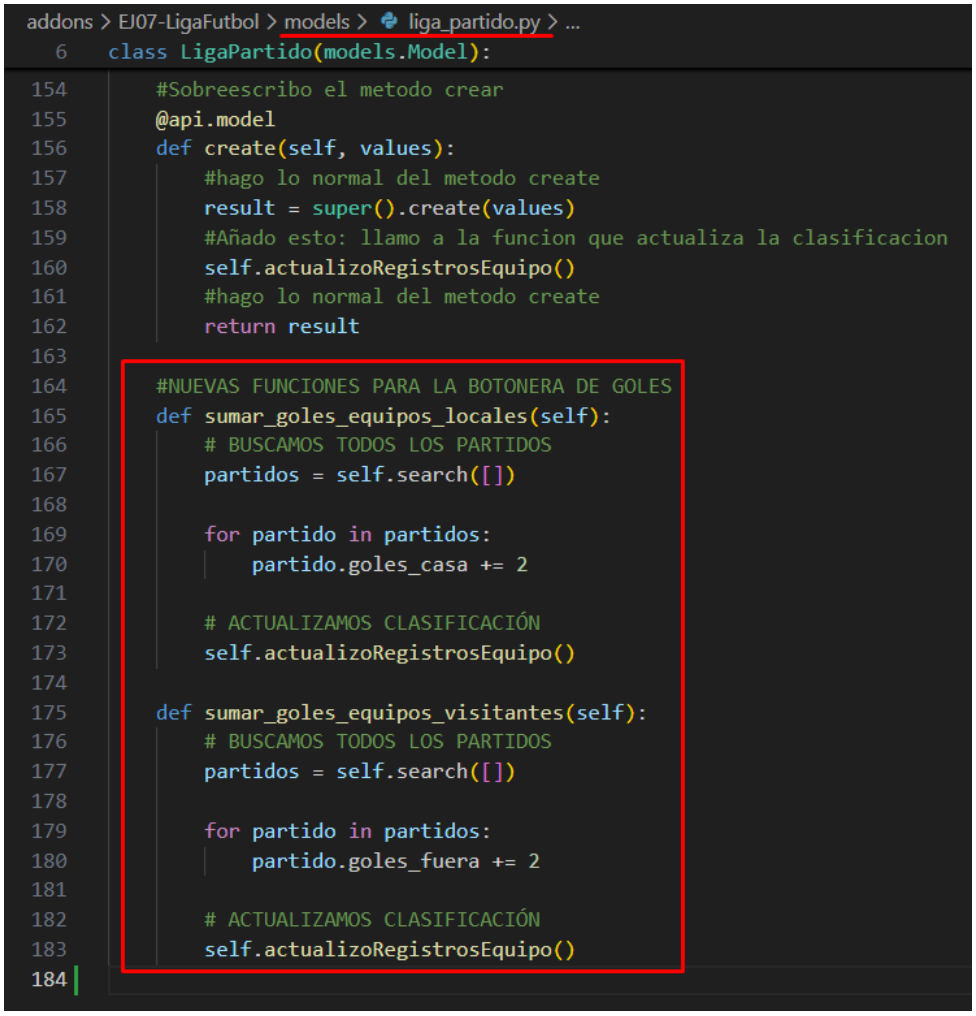


Figura 10: Recorremos array, aplicamos +2 a la propiedad correspondiente y actualizamos

Ahora toca modificar la vista *liga_partido.xml* para añadir los nuevos botones. En esta vista hay 2 formas de visualizar los datos, una en modo Lista y la otra en modo Kanban (la que está por defecto). Para hacer correctamente la implementación solicitada deberemos añadir la nueva botonera en la cabecera de ambas etiquetas, forzando su visualización (si no la forzamos deberemos seleccionar obligatoriamene algún partido para que se visualicen los botones), tal como se ve en las siguientes capturas:

addons > EJ07-LigaFutbol > views > liga_partido.xml

```
2  <odoo>
36     <!-- Vista list -->¡¡ LISTA !!
37     <record id="liga_partido_view_list" model="ir.ui.view">
38         <field name="name">Lista de partidos de la liga</field>
39         <field name="model">liga.partido</field>
40         <field name="arch" type="xml">
41             <list>
42                 <header>
43                     <button name="sumar_goles_equipos_locales"
44                         string="+2 goles equipos locales"
45                         type="object"
46                         display="always"          ¡¡IMPORTANTE!
47                         class="oe_highlight"
48                         confirm="Esta acción sumará 2 goles a todos los equipos locales, ¿estás seguro/a?"/>
49                     <button name="sumar_goles_equipos_visitantes"
50                         string="+2 goles equipos visitantes"
51                         type="object"
52                         display="always"          ¡¡IMPORTANTE!
53                         class="oe_highlight"
54                         confirm="Esta acción sumará 2 goles a todos los equipos visitantes, ¿estás seguro/a?"/>
55                 </header>
56                 <!-- Indicamos que atributos usaremos al hacer la vista list -->
57                 <field name="equipo_casa" />
58                 <field name="goles_casa" />
59                 <field name="equipo_fuera" />
60                 <field name="goles_fuera" />
61
62             </list>
```

addons > EJ07-LigaFutbol > views > liga_partido.xml

```
2  <odoo>
67     <!-- Vista Kanban -->¡¡ KANBAN !!
68     <record id="liga_partido_view_kanban" model="ir.ui.view">
69         <field name="name">Lista de partidos de la liga</field>
70         <field name="model">liga.partido</field>
71         <field name="arch" type="xml">
72             <!-- Agrupamos por el atributo "parent_id"-->
73             <kanban>
74                 <header>
75                     <button name="sumar_goles_equipos_locales"
76                         string="+2 goles equipos locales"
77                         type="object"
78                         display="always"          ¡¡IMPORTANTE!
79                         class="oe_highlight"
80                         confirm="Esta acción sumará 2 goles a todos los equipos locales, ¿estás seguro/a?"/>
81                     <button name="sumar_goles_equipos_visitantes"
82                         string="+2 goles equipos visitantes"
83                         type="object"
84                         display="always"          ¡¡IMPORTANTE!
85                         class="oe_highlight"
86                         confirm="Esta acción sumará 2 goles a todos los equipos visitantes, ¿estás seguro/a?"/>
87                 </header>
```

Llegados a este punto sólo nos falta comprobar que los cambios se han realizado correctamente. Visualicemos el módulo de Odoo para comprobarlo.

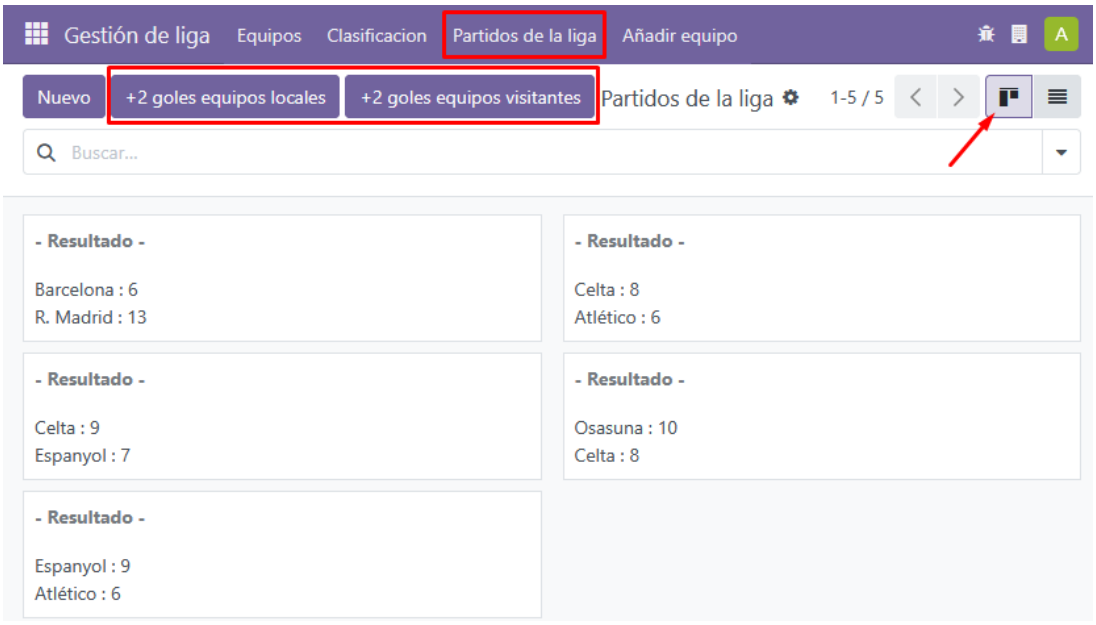


Figura 11: Vista Kanban (por defecto)

Figura 12: Vista Lista

Probemos la funcionalidad de los botones:

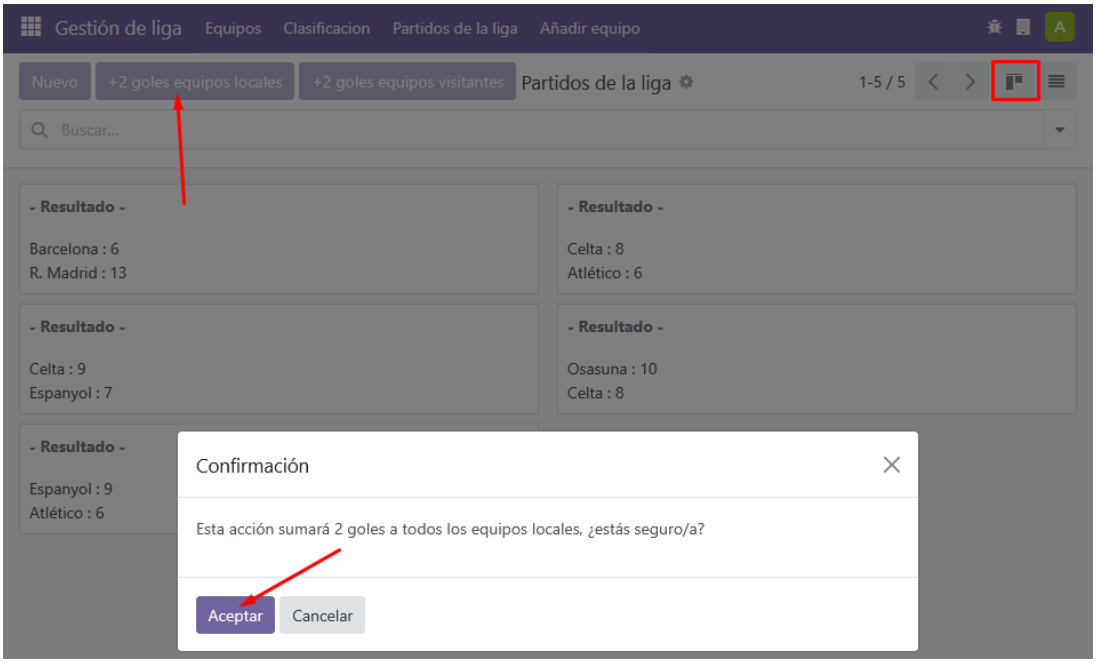


Figura 13: Añadimos 2 goles a todos los equipos locales en Vista Kanban

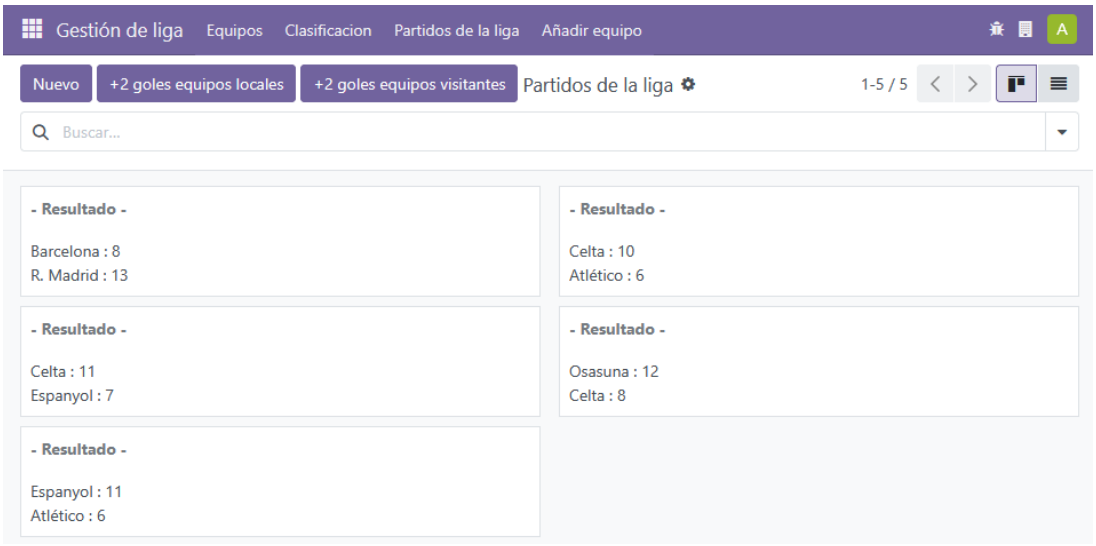


Figura 14: El resultado es satisfactorio

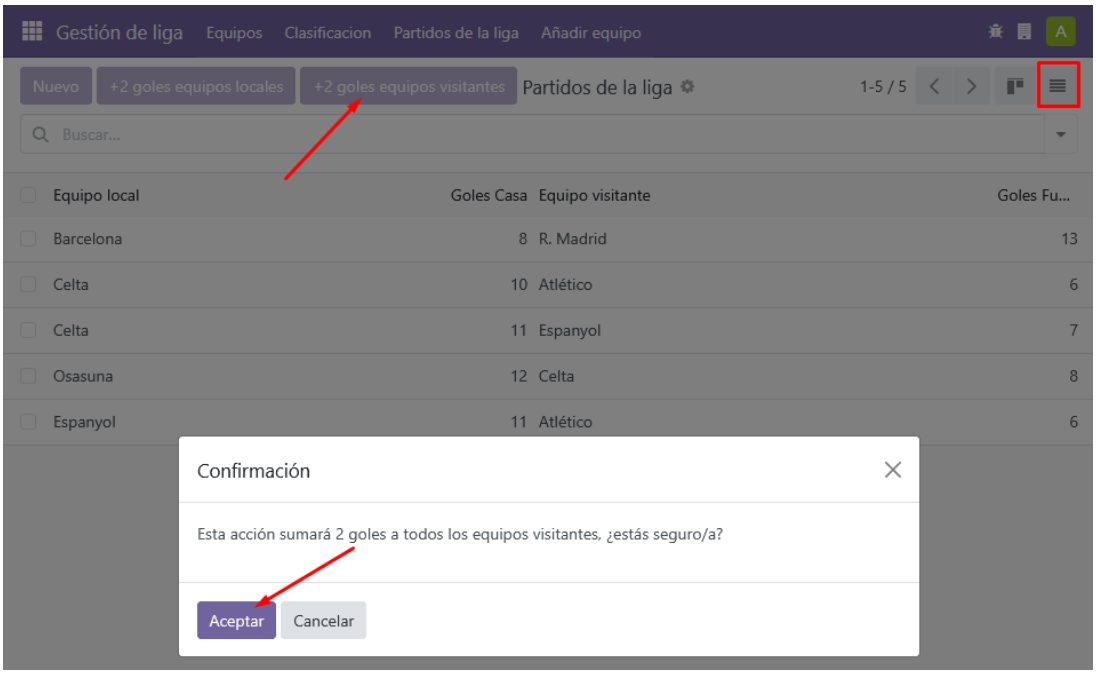


Figura 15: Añadimos 2 goles a todos los equipos locales en Vista Lista

Figura 16: El resultado es satisfactorio

Por último, vayamos a la Vista de Clasificación para comprobar que los datos se han actualizado correctamente.

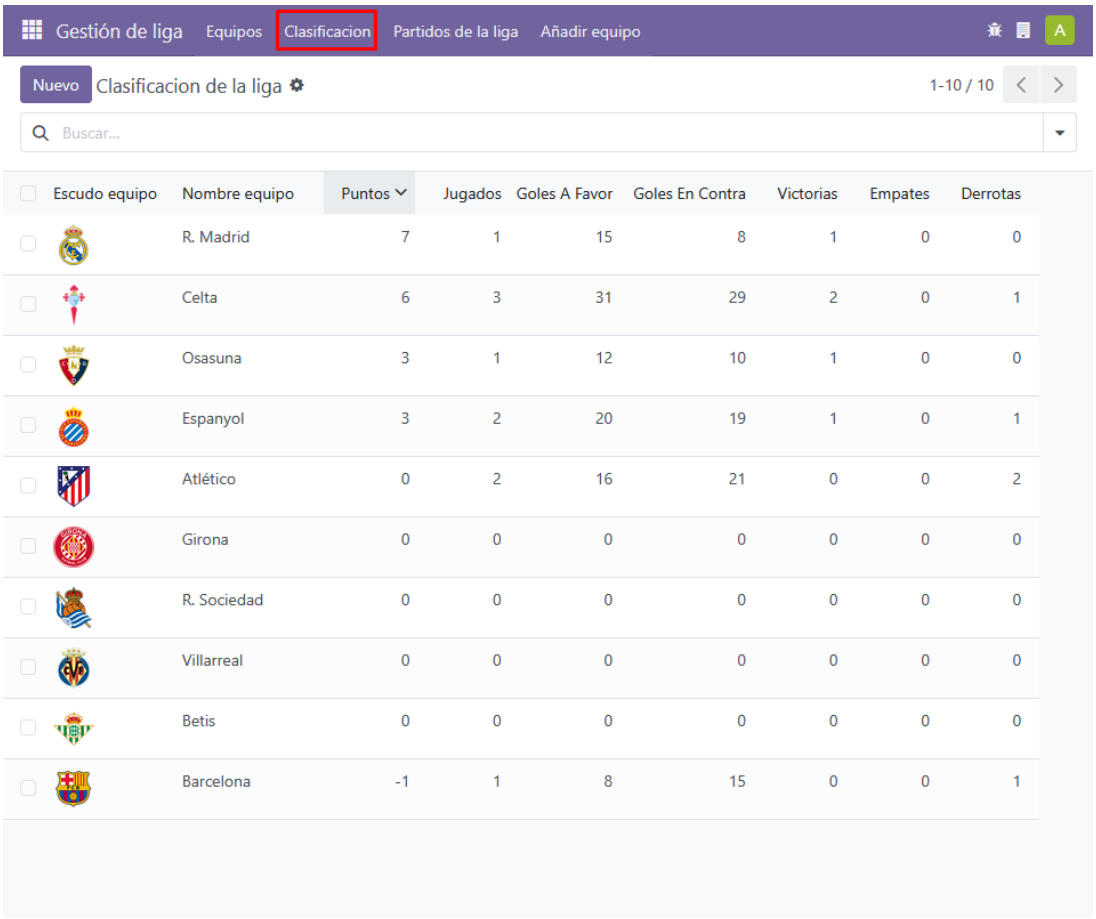


Figura 17: Actualización realizada satisfactoriamente

2.3. Web Controller para eliminar empates

Se nos solicita crear un Web Controller que elimine todos los partidos que hayan terminado en empate y que devuelva el número de partidos eliminados. La url designada para dicha petición será *http://localhost:**puerto**/eliminarempates* (en este caso, debido a la configuración de Docker, el puerto será el 9001).

Lo primero que haremos será echar un vistazo a la clase *main.py* ubicada en la carpeta *controllers*. Aquí crearemos una nueva función en la que, al acceder a la url requerida, efectuará la lógica solicitada. Veamos el código:

```
addons > EJ07-LigaFutbol > controllers > main.py > ...
You, now | 1 author (You)
# -*- coding: utf-8 -*-
from odoo import http    Import "odoo" could not be resolved
from odoo.http import request    Import "odoo.http" could not be resolved
import json

#Clase del controlador web
You, now | 1 author (You)
class Main(http.Controller):
    #Decorador que indica que la url "/ligafutbol/equipo/json" atendera por HTTP, sin autentificacion
    #Devolvera texto que estará en formato JSON
    #Se puede probar accediendo a http://localhost:8069/ligafutbol/equipo/json
    @http.route('/ligafutbol/equipo/json', type='http', auth='none')
    def obtenerDatosEquiposJSON(self):
        #Obtenemos la referencia al modelo de Equipo
        equipos = request.env['liga.equipo'].sudo().search([])

        #Generamos una lista con informacion que queremos sacar en JSON
        listaDatosEquipos=[]
        for equipo in equipos:
            listaDatosEquipos.append([equipo.nombre,str(equipo.fecha_fundacion),equipo.jugados,equipo.puntos,
                                      equipo.victorias,equipo.empates,equipo.derrotas])
        #Convertimos la lista generada a JSON
        json_result=json.dumps(listaDatosEquipos)

        return json_result

```

Figura 18: Web Controller por defecto

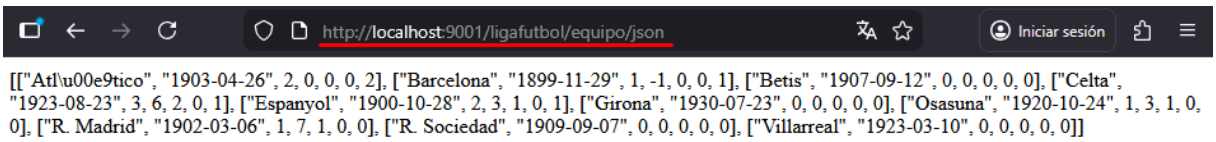


Figura 19: Comprobamos que efectivamente funciona

Creamos la nueva función solicitada, quedaría tal como se muestra en la siguiente captura:

addons > EJ07-LigaFutbol > controllers > main.py > ...

```
7   class Main(http.Controller):
26      # NUESTRO WEB CONTROLLER PARA ELIMINAR EMPATES
27      # http://localhost:9001/eliminarempates
28      @http.route('/eliminarempates', type='http', auth='none')
29      def eliminarPartidosConEmpate(self):
30          # OBTENEMOS TODOS LOS REGISTROS DE LA TABLA DE LA BD
31          partidos = request.env['liga.partido'].sudo().search([])
32
33          # LISTA TEMPORAL EN LA QUE GUARDAREMOS EMPATES EXISTENTES
34          listaEmpates = []
35
36          # RECORREMOS TODOS LOS REGISTROS
37          for partido in partidos:
38              # COMPROBAMOS SI HAY EMPATES Y LOS AÑADIMOS A LA LISTA
39              if partido.goles_casa == partido.goles_fuera:
40                  listaEmpates.append(partido)
41
42          # VARIABLE TEMPORAL DE TOTAL EMPATES, NECESARIA PORQUE DESPUÉS DEL BORRADO len(listaEmpates) SIEMPRE SERÁ 0
43          empatesTotales = len(listaEmpates)
44
45          # SI HAY EMPATES ELIMINAMOS LOS REGISTROS Y MOSTRAMOS LA RESPUESTA SOLICITADA
46          if empatesTotales > 0:
47              for empate in listaEmpates:
48                  empate.unlink()
49
50              respuesta = "El total de empates eliminados ha sido de: " + str(empatesTotales)
51          # SI NO HAY EMPATES LO INDICAREMOS TAMBIÉN
52          else:
53              respuesta = "NO se han podido realizar el borrado porque NO se han detectado empates"
54
55          # DEVOLVEMOS LA RESPUESTA ESPECIFICANDO QUE ES TEXTO PLANO PARA EVITAR POSIBLES ERRORES
56          return request.make_response(respuesta, [('Content-Type', 'text/plain')])
```

Haremos una prueba que nos debería devolver un resultado negativo ya que nos tenemos almacenados partidos con empate:

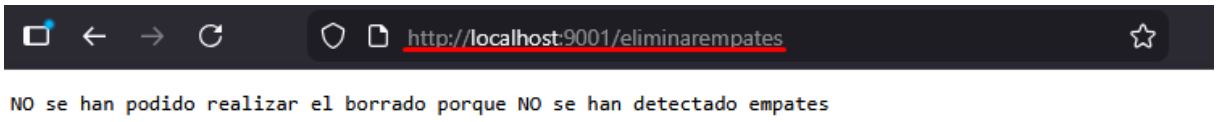


Figura 20: Este resultado es correcto

Como queremos comprobar un resultado positivo crearemos unos empates y volveremos a hacer la petición, tal como se muestra en las siguientes imágenes:

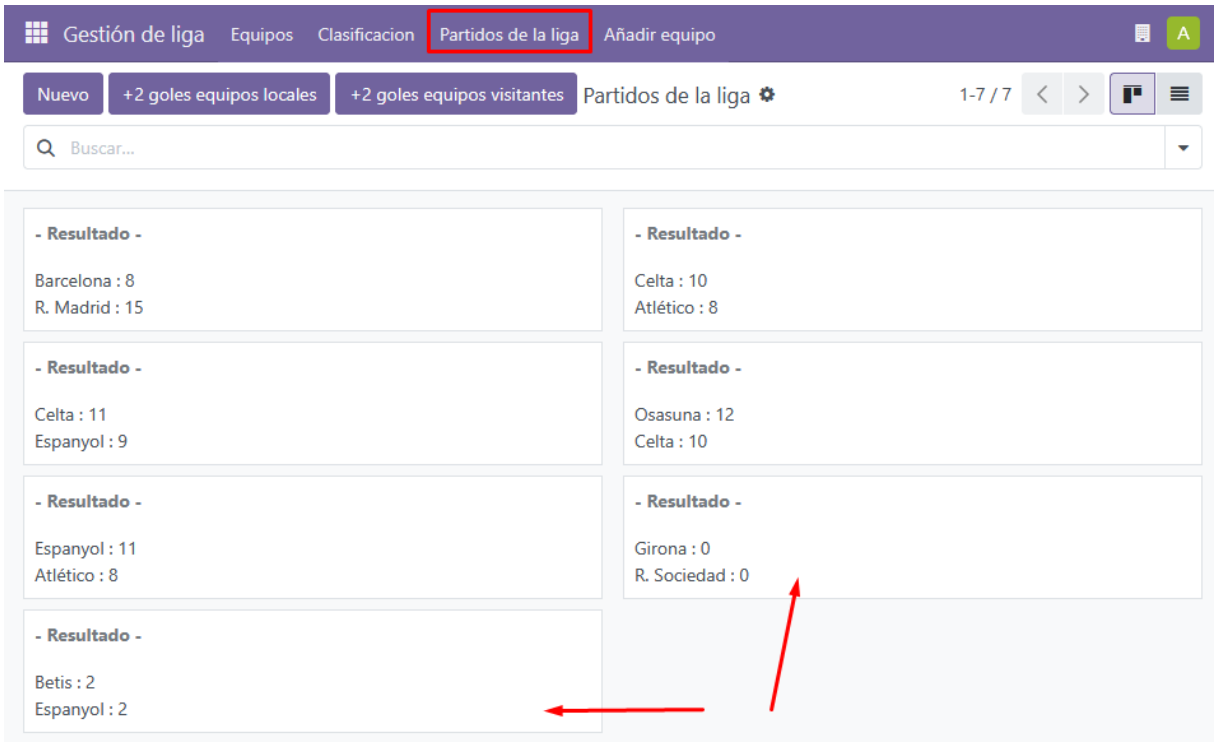


Figura 21: Creamos 2 empates de prueba

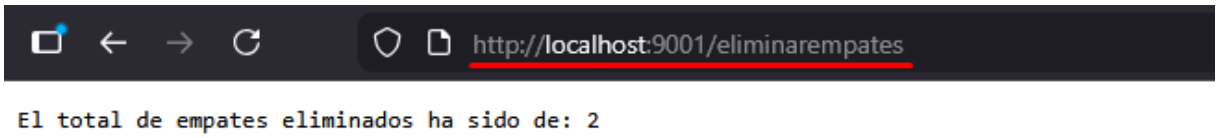
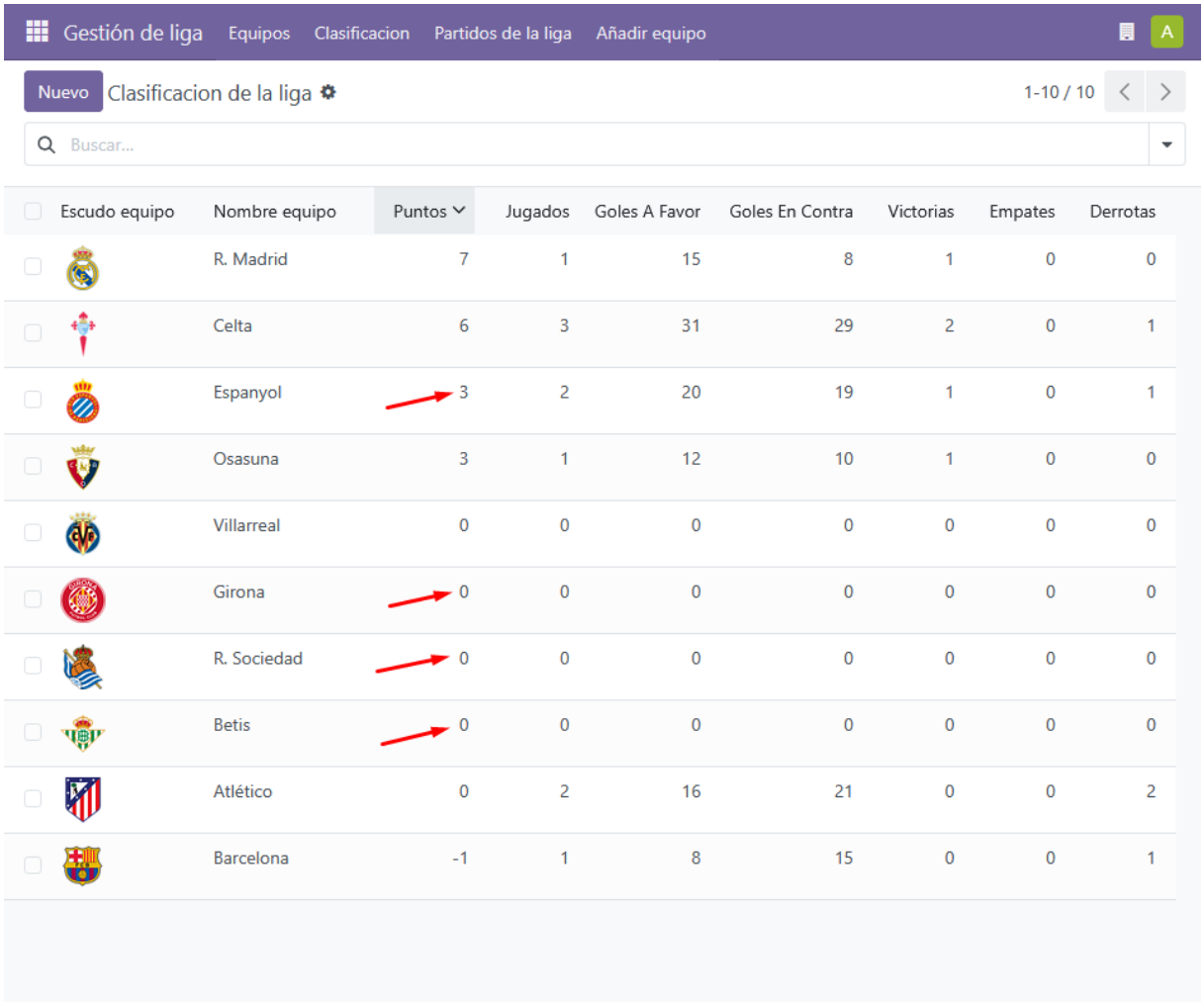


Figura 22: Ahora el resultado es el correcto tras la eliminación de empates

Finalmente vamos a comprobar la clasificación para verificar que las puntuaciones se han actualizado correctamente tras los borrados:



Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A Favor	Goles En Contra	Victorias	Empates	Derrotas
	R. Madrid	7	1	15	8	1	0	0
	Celta	6	3	31	29	2	0	1
	Espanyol	3	2	20	19	1	0	1
	Osasuna	3	1	12	10	1	0	0
	Villarreal	0	0	0	0	0	0	0
	Girona	0	0	0	0	0	0	0
	R. Sociedad	0	0	0	0	0	0	0
	Betis	0	0	0	0	0	0	0
	Atlético	0	2	16	21	0	0	2
	Barcelona	-1	1	8	15	0	0	1

Figura 23: Puntuaciones actualizadas satisfactoriamente

No hay duda... este Web Controller funciona a las mil maravillas.

2.4. Informe PDF por cada partido

Se nos pide generar un informe PDF que muestre los datos de un partido. Para ello crearemos una plantilla visual dentro de la carpeta *report* (que está relacionada con la generación de documentos impresos) y un botón dentro de la vista de partidos para que el usuario pueda generar el documento.

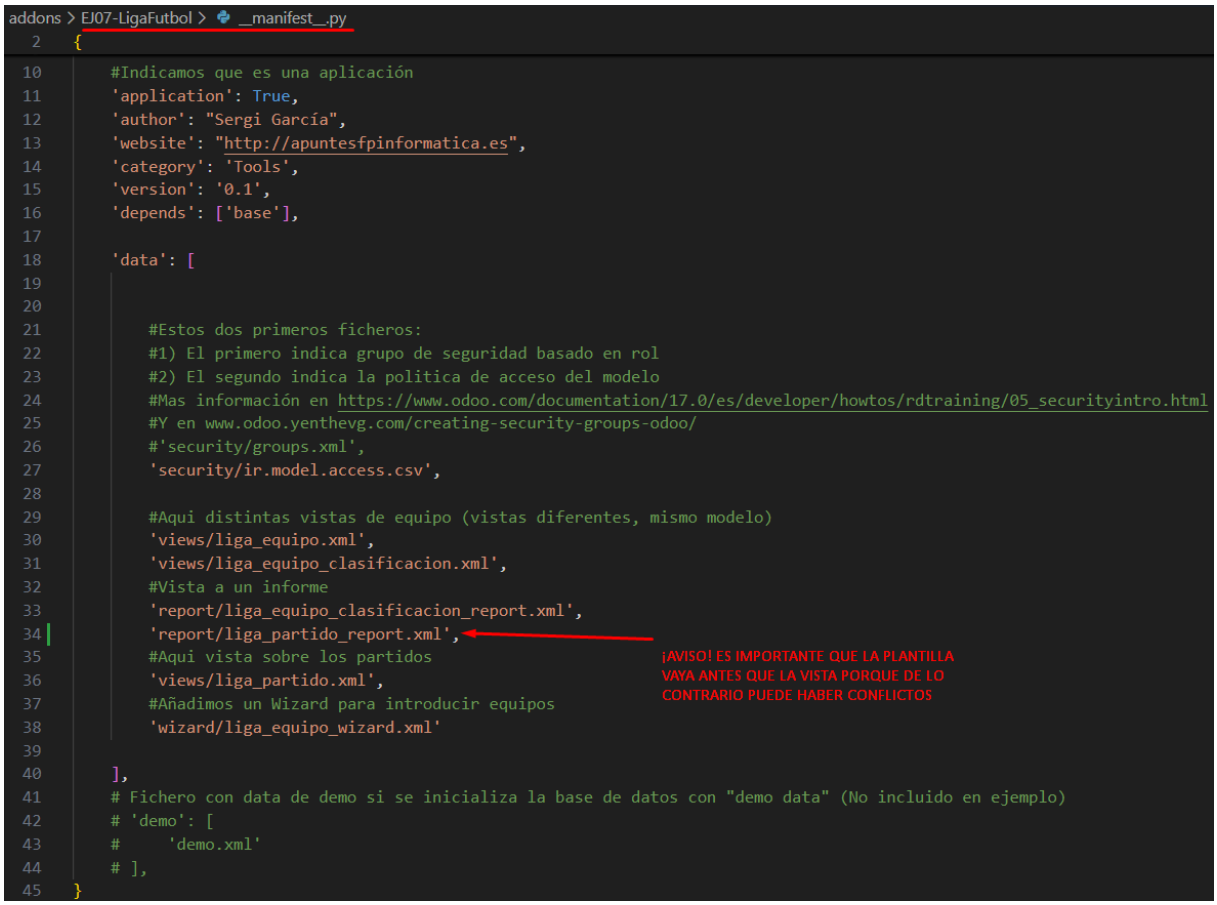
Echémosle un vistazo a la plantilla que ya está creada:

addons > EJ07-LigaFutbol > report > liga_equipo_clasificacion_report.xml

You, now | 1 author (You)

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>

    <!-- Ejemplo de informe asociado a equipo (se genera al darle  aimprimir
        desde la vista Form de un equipo) -->
    <template id="report_clasificacion_view">
        <!-- Aqui la estructura a seguir -->
        <t t-call="web.html_container">
            <t t-foreach="docs" t-as="doc">
                <t t-call="web.internal_layout">
                    <div class="page">
                        <!-- Cabecera con el nombre de equipo -->
                        <h2 t-field="doc.nombre" />
                        <!-- Datos de los partidos jugados por el equipo -->
                        <p>
                            Partidos jugados:
                            <span t-field="doc.jugados" />
                        </p><p>
                            Partidos ganados:
                            <span t-field="doc.victorias" />
                        </p><p>
                            Partidos empatados:
                            <span t-field="doc.empates" />
                        </p><p>
                            Partidos perdidos:
                            <span t-field="doc.derrotas" />
                        </p><p>
                            Puntos obtenidos:
                            <span t-field="doc.puntos" />
                        </p>
                    </div>
                </t>
            </t>
        </t>
    </template>

    <!-- Realmente, este es el informe, lo de arriba es la plantilla que utilizara el informe -->
    <report id="report_clasificacion" model="liga.equipo" string="Informe clasificacion de cada equipo"
    name="EJ07-LigaFutbol.report_clasificacion_view" file="EJ07-LigaFutbol.report_clasificacion_view"
    report_type="qweb-pdf" />


</odoo>
```

Basándonos en esa plantilla crearemos una nueva desde cero que cumpla con los requisitos solicitados. La llamaremos *liga_partido_report.xml* y la ubicaremos dentro de la carpeta *report*. Será necesario enlazarla dentro del archivo *__manifest__.py* del proyecto, tal como se muestra en la siguiente captura:



La creación de la plantilla es problemática ya que **Odoo 18** y la librería **wkhtmltopdf**, ya que existen ciertas incompatibilidades. Odoo 18 usa Bootstrap 5, que basa casi todo su diseño en Flexbox y CSS Grid. Sin embargo, wkhtmltopdf (el motor que genera el PDF) utiliza un motor de renderizado basado en una versión muy antigua de WebKit (similar a un navegador de hace 10 años). Por las dificultades de crear una plantilla decente me he visto obligado a generar gran parte de la parte visual con asistencia de la IA.

Veamos por partes cómo ha quedado el código de la plantilla en las siguientes capturas:

addons > EJ07-LigaFutbol > report > liga_partido_report.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>

    <record id="action_report_liga_partido" model="ir.actions.report">
        <field name="name">Acta del Partido</field>
        <field name="model">liga.partido</field>
        <field name="report_type">qweb-pdf</field>
        <field name="report_name">EJ07-LigaFutbol.report_partido_liga_template</field>
        <field name="report_file">EJ07-LigaFutbol.report_partido_liga_template</field>
        <field name="print_report_name">'Partido - %s vs %s' % (object.equipo_casa.nombre, object.equipo_fuera.nombre)</field>
        <field name="binding_model_id" ref="model_liga_partido"/>
        <field name="binding_type">report</field>
    </record>

    <template id="report_partido_liga_template">
        <t t-call="web.html_container">
            <t t-foreach="docs" t-as="doc">
                <t t-call="web.external_layout">
                    <div class="page">
                        <div class="text-center" style="border-bottom: 2px solid black; margin-bottom: 20px;">
                            <h2>Acta Oficial de Partido</h2>
                        </div>
```

ESTE ID SERVIRÁ PARA ENLAZAR CON EL BOTÓN DE LA VISTA

ESPECIFICAR NOMBRE MÓDULO Y ID TEMPLATE

Figura 24: La acción (orden de generar informe) e inicio de la plantilla

```xml
                        <table style="width: 100%; text-align: center; border-collapse: collapse; margin-top: 0px;">
                            <tr>
                                <td style="width: 40%; vertical-align: top;">
                                    <h3 t-field="doc.equipo_casa.nombre" style="margin-bottom: 10px;"/>
                                    <div style="font-size: 48px; font-weight: bold;">
                                        <span t-field="doc.goles_casa"/>
                                    </div>
                                    <p style="color: #666; font-size: 12px;">LOCAL</p>
                                </td>

                                <td style="width: 20%; vertical-align: middle;">
                                    <span style="font-size: 24px; font-weight: 300; color: #999;">VS</span>
                                </td>

                                <td style="width: 40%; vertical-align: top;">
                                    <h3 t-field="doc.equipo_fuera.nombre" style="margin-bottom: 10px;"/>
                                    <div style="font-size: 48px; font-weight: bold;">
                                        <span t-field="doc.goles_fuera"/>
                                    </div>
                                    <p style="color: #666; font-size: 12px;">VISITANTE</p>
                                </td>
                            </tr>
                        </table>
```

Figura 25: Esta parte mostrará los equipos y el resultado

```
<div style="width: 100%; margin-top: 30px;">
    <table class="table table-sm" style="width: 100%; border-top: 1px solid #dee2e6;">
        <thead>
            <tr style="background-color: #f8f9fa;">
                <th style="padding: 8px; text-align: left; border-bottom: 2px solid #333;">Detalle</th>
                <th style="padding: 8px; text-align: right; border-bottom: 2px solid #333;">Resultado</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td style="padding: 8px; border-bottom: 1px solid #eee;">Estado del encuentro</td>
                <td style="padding: 8px; text-align: right; border-bottom: 1px solid #eee;">
                    <span class="badge badge-secondary" style="text-transform: uppercase;">Finalizado</span>
                </td>
            </tr>
            <tr>
                <td style="padding: 8px; border-bottom: 1px solid #eee;">Diferencia de goles</td>
                <td style="padding: 8px; text-align: right; border-bottom: 1px solid #eee; font-weight: bold;">
                    <t t-esc="abs(doc.goles_casa - doc.goles_fuera)"/>
                </td>
            </tr>
        </tbody>
    </table>
</div>
```

Figura 26: Esta parte mostrará los detalles del partido

Por último falta añadir el botón en la vista *liga_partido.xml* tal como se ve en la siguiente captura:

addons > EJ07-LigaFutbol > views > liga_partido.xml

```
<odoo>
    <!-- VISTA DE FORMULARIO -->
    <record id="liga_partido_view_form" model="ir.ui.view">
        <field name="name">Formulario Partidos</field>
        <field name="model">liga.partido</field>
        <field name="arch" type="xml">
            <form>
                <button name="%(EJ07-LigaFutbol.action_report_liga_partido)d" string="Generar PDF" type="action" class="btn-primary"/>
                <group>
                    <group>
                        <field name="equipo_casa" />
                        <field name="goles_casa" />
                    </group>

                    <group>
                        <field name="equipo_fuera" />
                        <field name="goles_fuera" />
                    </group>
                </group>
            </form>
        </field>
    </record>
```

ESTE FORMATO DE CADENA "%(..)D" PRIMERO INDICA QUE ES UNA ID DE LA BASE DE DATOS Y SEGUNDO PRECISA QUE ES UN NÚMERO ENTERO. ¡¡ES ESTRICTAMENTE NECESARIO DAR ESTE FORMATO!!

Figura 27: Esta será nuestro botón, se llamará *"Generar PDF"*

Sólo nos falta comprobar si funciona. Cabe recordar que será necesario reiniciar el contenedor y actualizar el módulo desde Odoo. Veamos los resultados en las siguientes capturas:

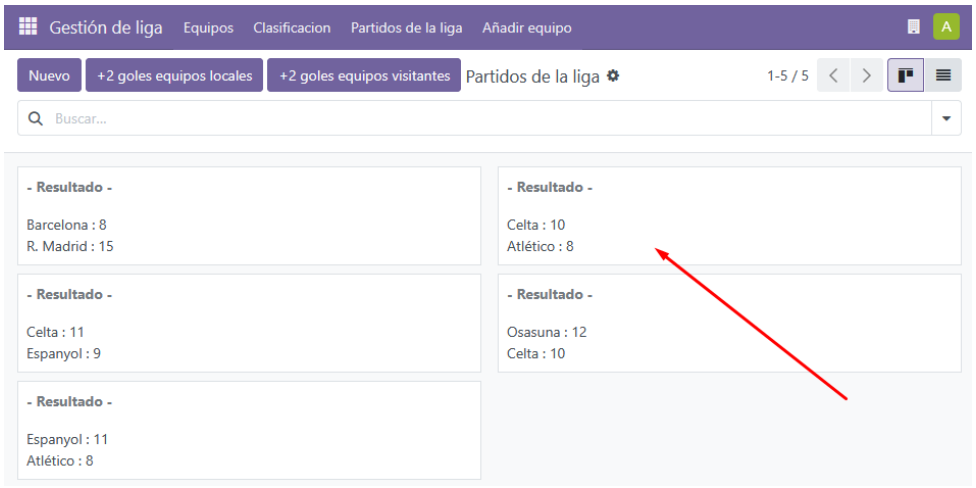


Figura 28: Clicamos en un partido

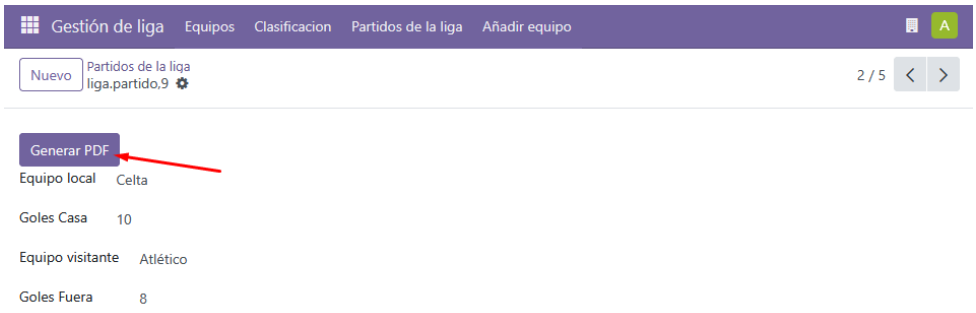


Figura 29: Clicamos en el nuevo botón

Figura 30: Tras unos segundos se generará el siguiente archivo PDF

Si quisiésemos imprimir todos los partidos en un único archivo deberemos cargar la vista Lista, seleccionar todos los partidos y darle a imprimir.

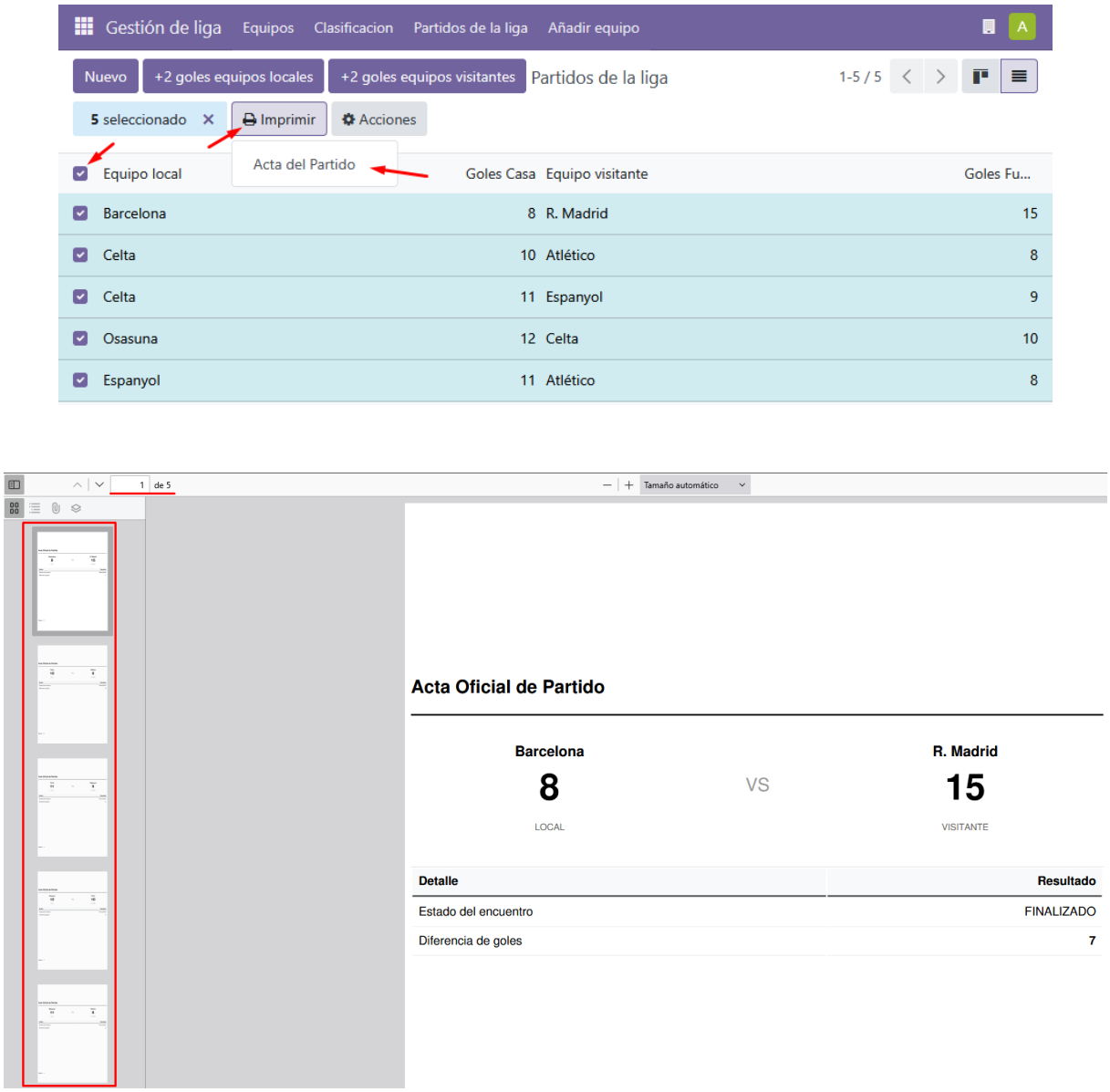


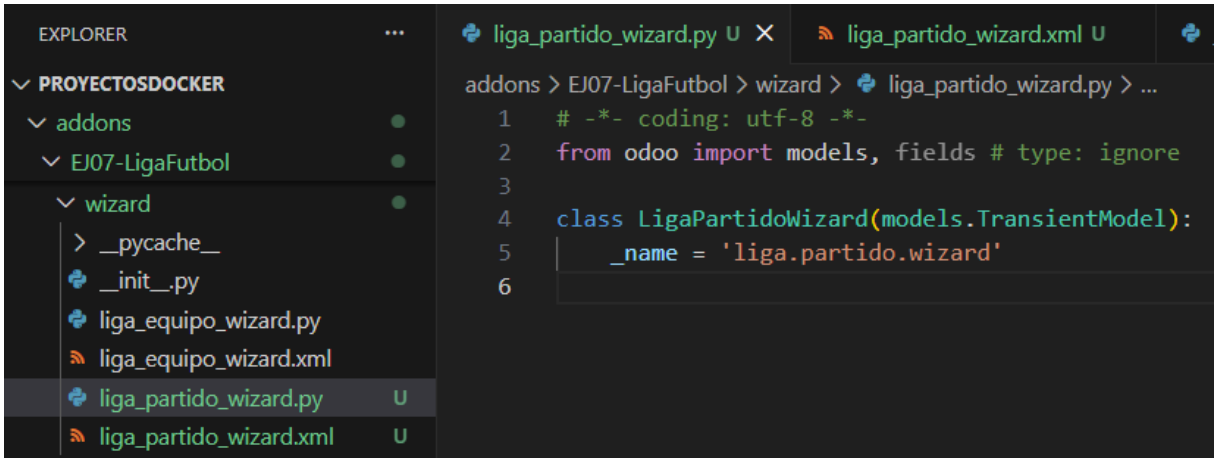
Figura 31: ¿El resultado? Un PDF con todos los partidos

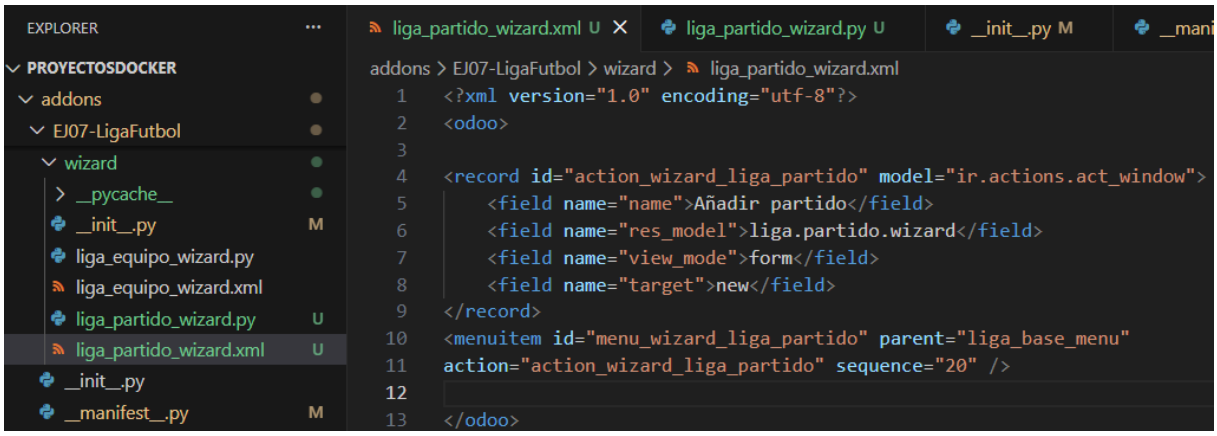
Es un quebradero de cabeza muy elevado crear una plantilla decente si no se tienen en cuenta todas las dificultades existentes actualmente entre Odoo y la herramienta externa *wkhtmltopdf*.

2.5. Wizard para crear nuevos partidos

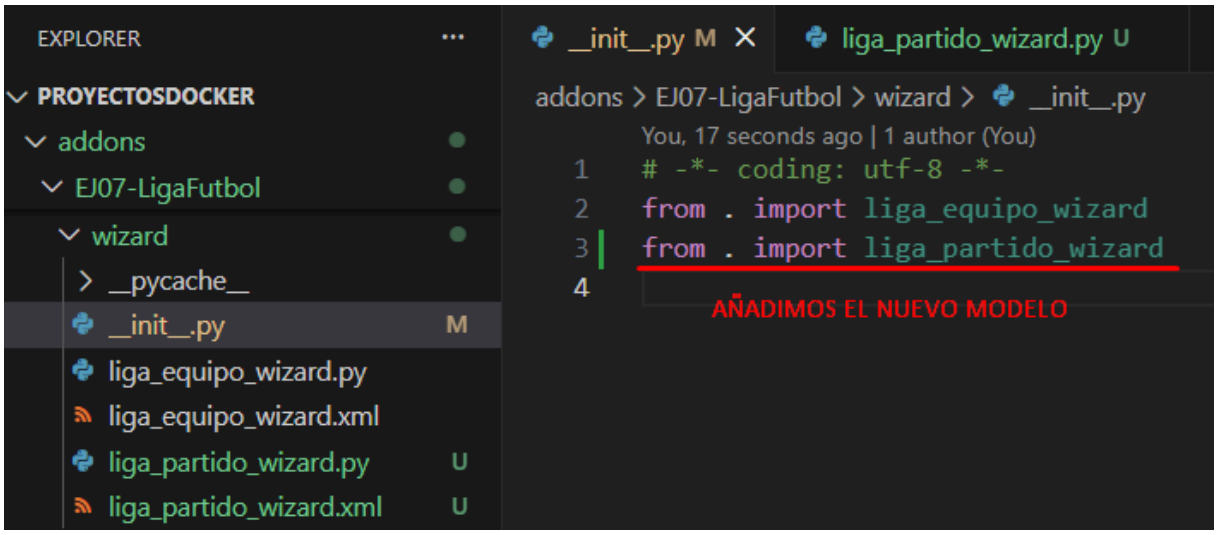
Se solicita crear un Wizard que permita introducir equipo local, equipo visitante, goles y un nuevo campo jornada. Obviamente esto generará un nuevo registro en la tabla liga_partido de la base de datos. Para crear correctamente el Wizard tendremos que crear un modelo y una vista, enlazarlos en el archivo *__init__.py* ubicado en la carpeta *wizard* del proyecto y en el *__manifest__.py* en la raíz del proyecto respectivamente, y sin olvidarnos de proporcionar permisos al nuevo modelo creado en el archivo *ir.model.access.csv*.

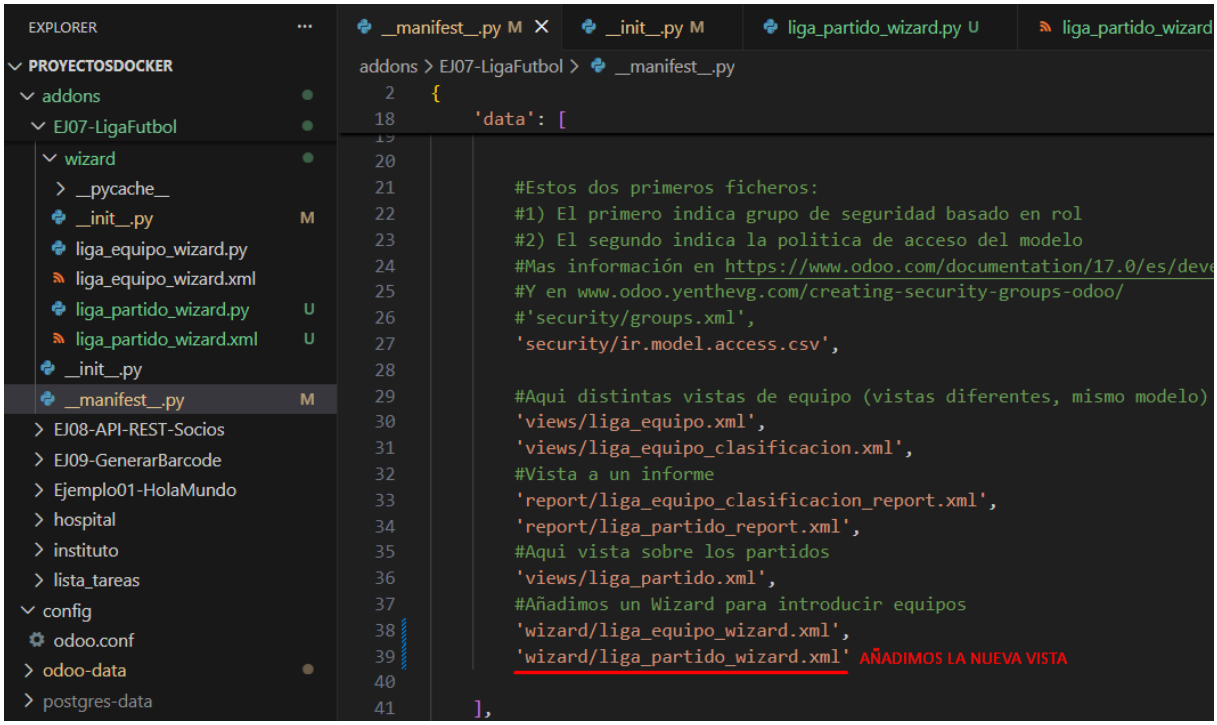
Empecemos por lo más tedioso. Primero crearemos el modelo y la vista únicamente con el esqueleto dentro de la carpeta wizard.



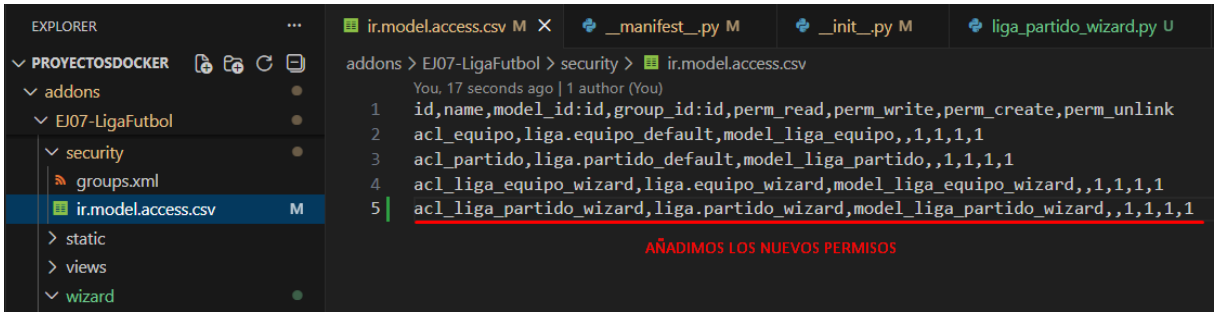


En las siguiente página los enlaces:





Por último los permisos:



Configuración inicial del Wizard completada. Tras comprobar que la pestaña "Añadir partido" aparece en nuestro módulo de Odoo podemos empezar a escribir el código del modelo.

addons > EJ07-LigaFutbol > wizard > liga_partido_wizard.py > ...

```python
# -*- coding: utf-8 -*-
from odoo import models, fields # type: ignore

class LigaPartidoWizard(models.TransientModel):
    _name = 'liga.partido.wizard'
    _description = 'Wizard para registrar un partido'

    # CAMPOS DEL MODELO QUE USAREMOS EN EL WIZARD
    equipo_casa = fields.Many2one('liga.equipo', string='Equipo Local', required=True)
    equipo_fuera = fields.Many2one('liga.equipo', string='Equipo Visitante', required=True)
    goles_casa = fields.Integer('Goles Local', default=0)
    goles_fuera = fields.Integer('Goles Visitante', default=0)
    jornada = fields.Integer('Jornada', default=1, required=False)

    def crear_partido(self):
        # REFERENCIA DEL MODELO DESTINO
        modeloLigaPartido = self.env['liga.partido']
        # RECORREMOS tODO EL MODELO
        for wiz in self:
            # POR CADA ELEMENTO SE CREARÁ UN REGISTRO (SÓLO 1 EN ESTE CASO)
            modeloLigaPartido.create({
                'equipo_casa': wiz.equipo_casa.id,
                'equipo_fuera': wiz.equipo_fuera.id,
                'goles_casa': wiz.goles_casa,
                'goles_fuera': wiz.goles_fuera,
                'jornada': wiz.jornada
            })

        # CIERRA EL WIZARD Y REFRESCA LA VISTA ACTUAL
        return {'type': 'ir.actions.client', 'tag': 'reload'}
```

LOS .id SIEMPRE SON OBLIGATORIOS EN LOS CAMPOS Many2One. SI NO SE ESPECIFICAN ODOO LANZARÁ UN ERROR AL INTENTAR REGISTRAR EL PARTIDO.

Es importante que el nombre de los campos de este Wizard coincidan con los nombres de los campos del modelo original, en este caso *liga_partido.py*, en el que tendremos que añadir el nuevo campo jornada.

addons > EJ07-LigaFutbol > models > liga_partido.py >

```python
class LigaPartido(models.Model):
    #Goles equipo de casa
    goles_fuera= fields.Integer()

    # AÑADIMOS UN NUEVO CAMPO JORNADA
    jornada = fields.Integer()
```

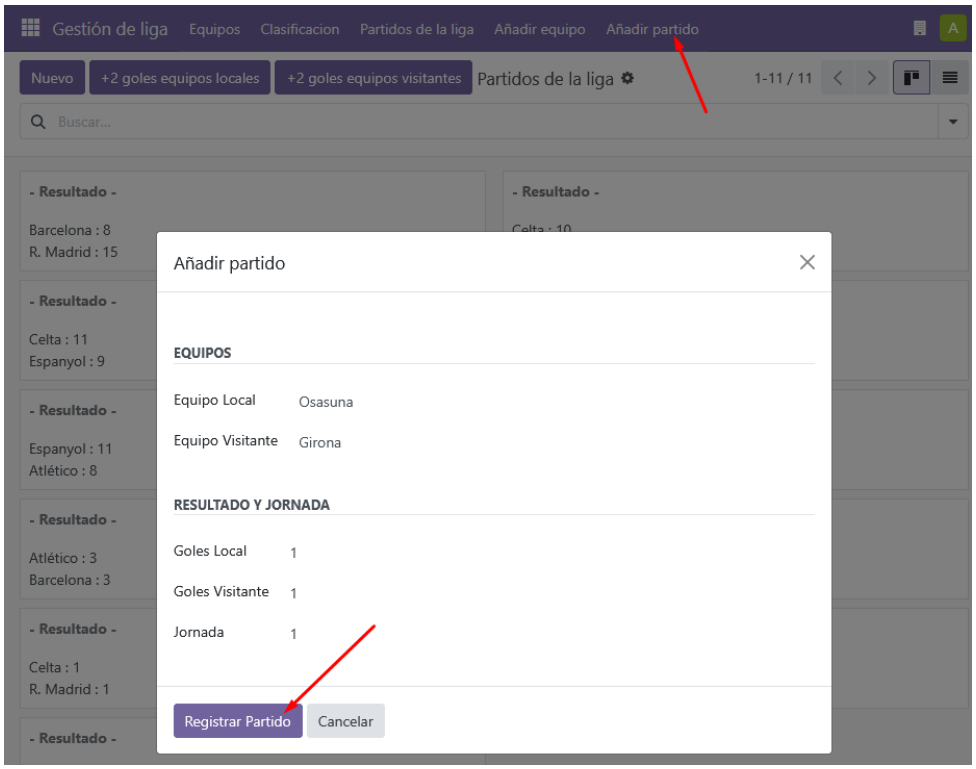
Ahora echemos un vistazo a cómo debe quedar la vista:

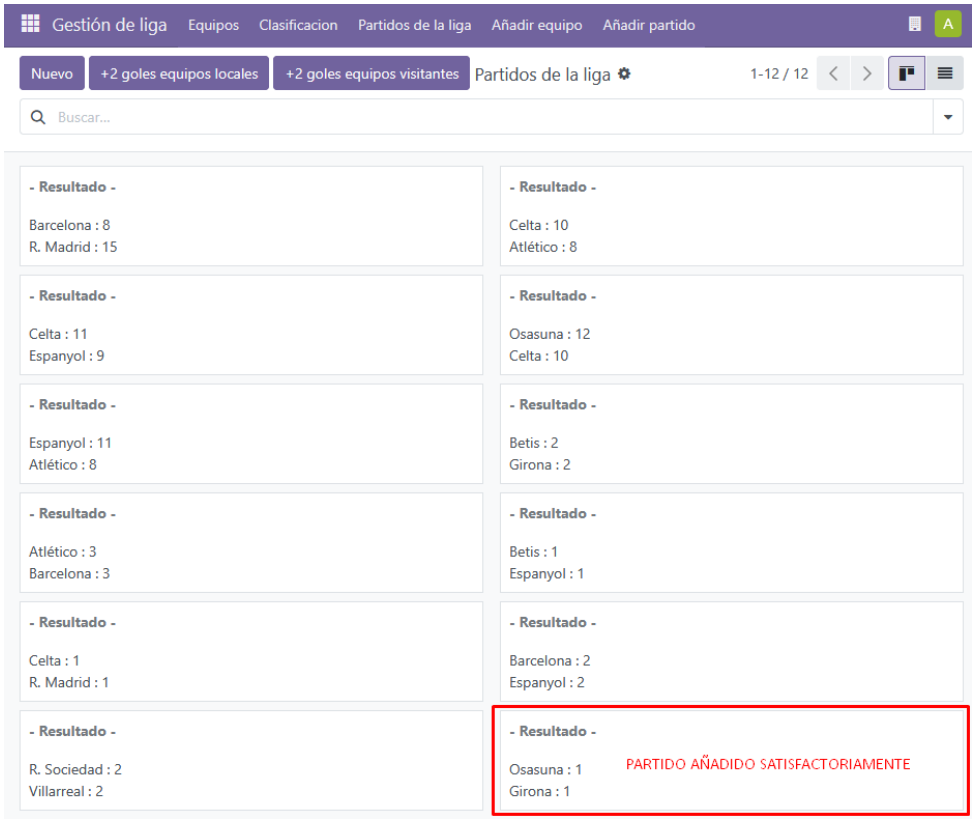
addons > EJ07-LigaFutbol > wizard > liga_partido_wizard.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>

<record id='liga_partido_wizard_form' model='ir.ui.view'>
    <field name='name'>Añadir Partido</field>
    <field name='model'>liga.partido.wizard</field>
    <field name='arch' type='xml'>
        <form>
            <group>
                <group string="Equipos">
                    <field name='equipo_casa'/>
                    <field name='equipo_fuera'/>
                </group>
                <group string="Resultado y Jornada">
                    <field name='goles_casa'/>
                    <field name='goles_fuera'/>
                    <field name='jornada'/>
                </group>
            </group>
            <footer>
                <button string='Registrar Partido' name='crear_partido' class='btn-primary' type='object'/>
                <button string='Cancelar' class='btn-secondary' special='cancel'/>
            </footer>
        </form>
    </field>
</record>

<record id="action_wizard_liga_partido" model="ir.actions.act_window">
    <field name="name">Añadir partido</field>
    <field name="res_model">liga.partido.wizard</field>
    <field name="view_mode">form</field>
    <field name="target">new</field>
</record>

<menuitem   id="menu_wizard_liga_partido"
            parent="liga_base_menu"
            action="action_wizard_liga_partido"
            sequence="20" />
</odoo>
```

Tenemos todo lo necesario para hacer pruebas. Como bien sabemos, toca reiniciar contenedores y actualizar módulos dentro de Odoo.

En las siguientes capturas observaremos el Wizard en funcionamiento:


Gestión de liga
Equipos
Clasificacion
Partidos de la liga
Añadir equipo
Añadir partido
Nuevo
+2 goles equipos locales
+2 goles equipos visitantes
Partidos de la liga
1-11 / 11
Buscar...
- Resultado -
Barcelona : 8
R. Madrid : 15
- Resultado -
Celta : 11
Espanyol : 9
- Resultado -
Espanyol : 11
Atlético : 8
- Resultado -
Atlético : 3
Barcelona : 3
- Resultado -
Celta : 1
R. Madrid : 1
- Resultado -
Añadir partido
EQUIPOS
Equipo Local
Osasuna
Equipo Visitante
Girona
RESULTADO Y JORNADA
Goles Local
1
Goles Visitante
1
Jornada
1
Registrar Partido
Cancelar


Gestión de liga
Equipos
Clasificacion
Partidos de la liga
Añadir equipo
Añadir partido
Nuevo
+2 goles equipos locales
+2 goles equipos visitantes
Partidos de la liga
1-12 / 12
Buscar...
- Resultado -
Barcelona : 8
R. Madrid : 15
- Resultado -
Celta : 10
Atlético : 8
- Resultado -
Celta : 11
Espanyol : 9
- Resultado -
Osasuna : 12
Celta : 10
- Resultado -
Espanyol : 11
Atlético : 8
- Resultado -
Betis : 2
Girona : 2
- Resultado -
Atlético : 3
Barcelona : 3
- Resultado -
Betis : 1
Espanyol : 1
- Resultado -
Celta : 1
R. Madrid : 1
- Resultado -
Barcelona : 2
Espanyol : 2
- Resultado -
R. Sociedad : 2
Villarreal : 2
- Resultado -
Osasuna : 1
Girona : 1
PARTIDO AÑADIDO SATISFACTORIAMENTE

2.6. Vista Graph

Se nos solicita implementar una vista gráfica en la vista partidos que muestre estadísticas de los goles marcados por los equipos locales. Basándonos en el Graph que ya está creado en la vista *liga_equipo.xml*, vamos a adaptarlo en la vista *liga_partido.xml*.

Veamos cómo queda el nuevo código añadido:

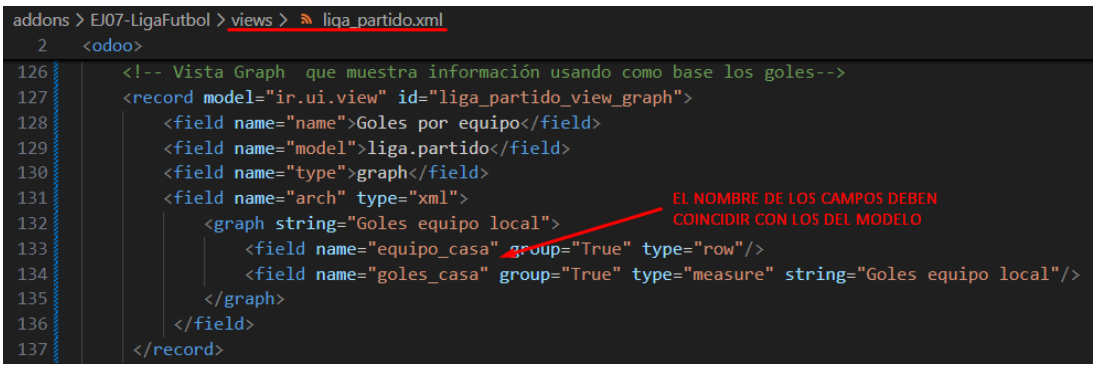


Figura 32: Recordad añadir graph en el *view_mode*

Comprobremos si funciona:

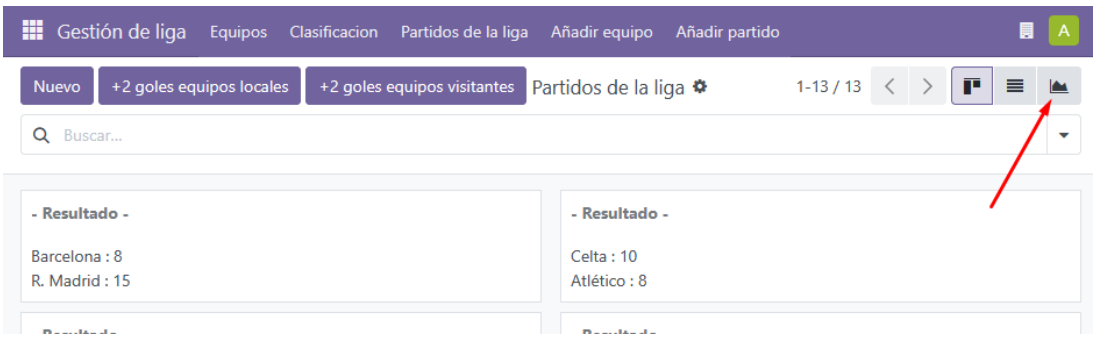
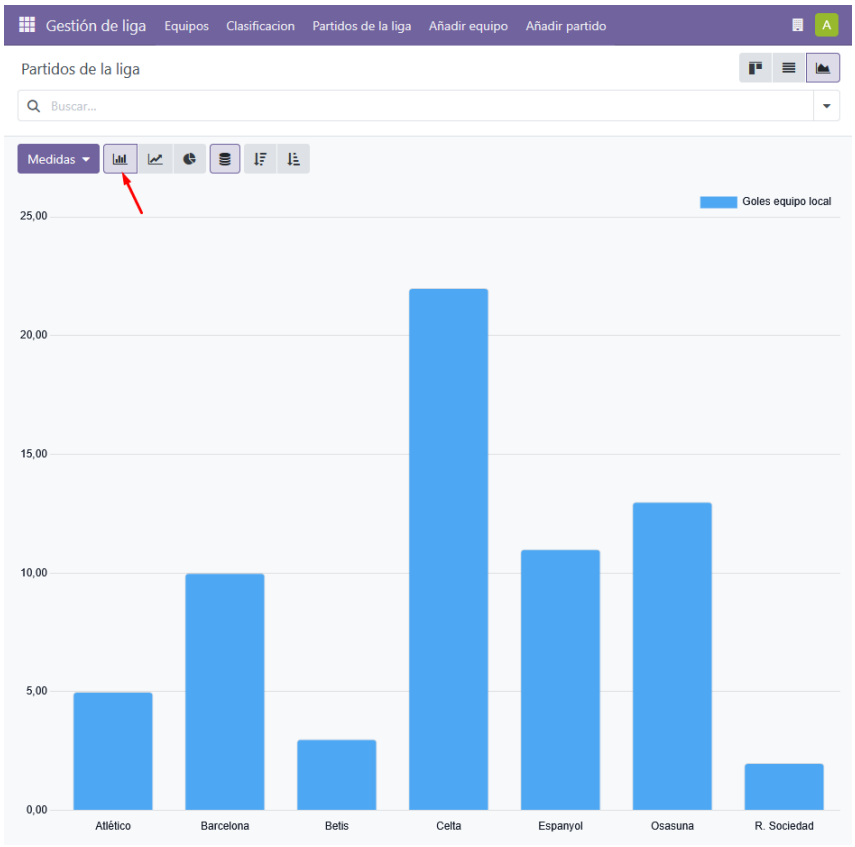
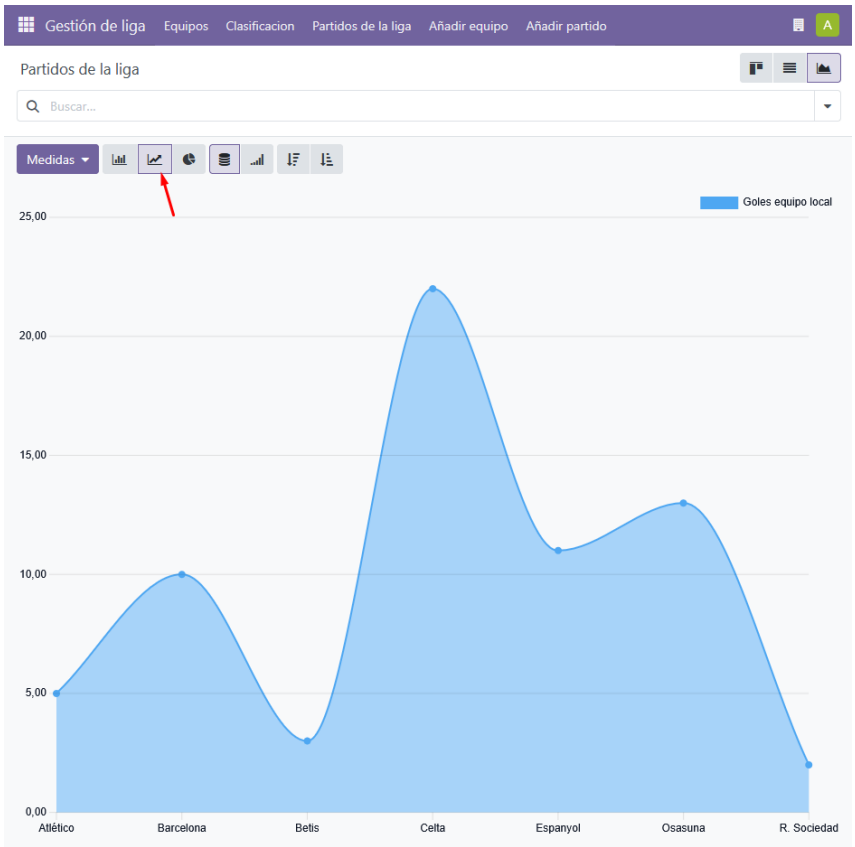
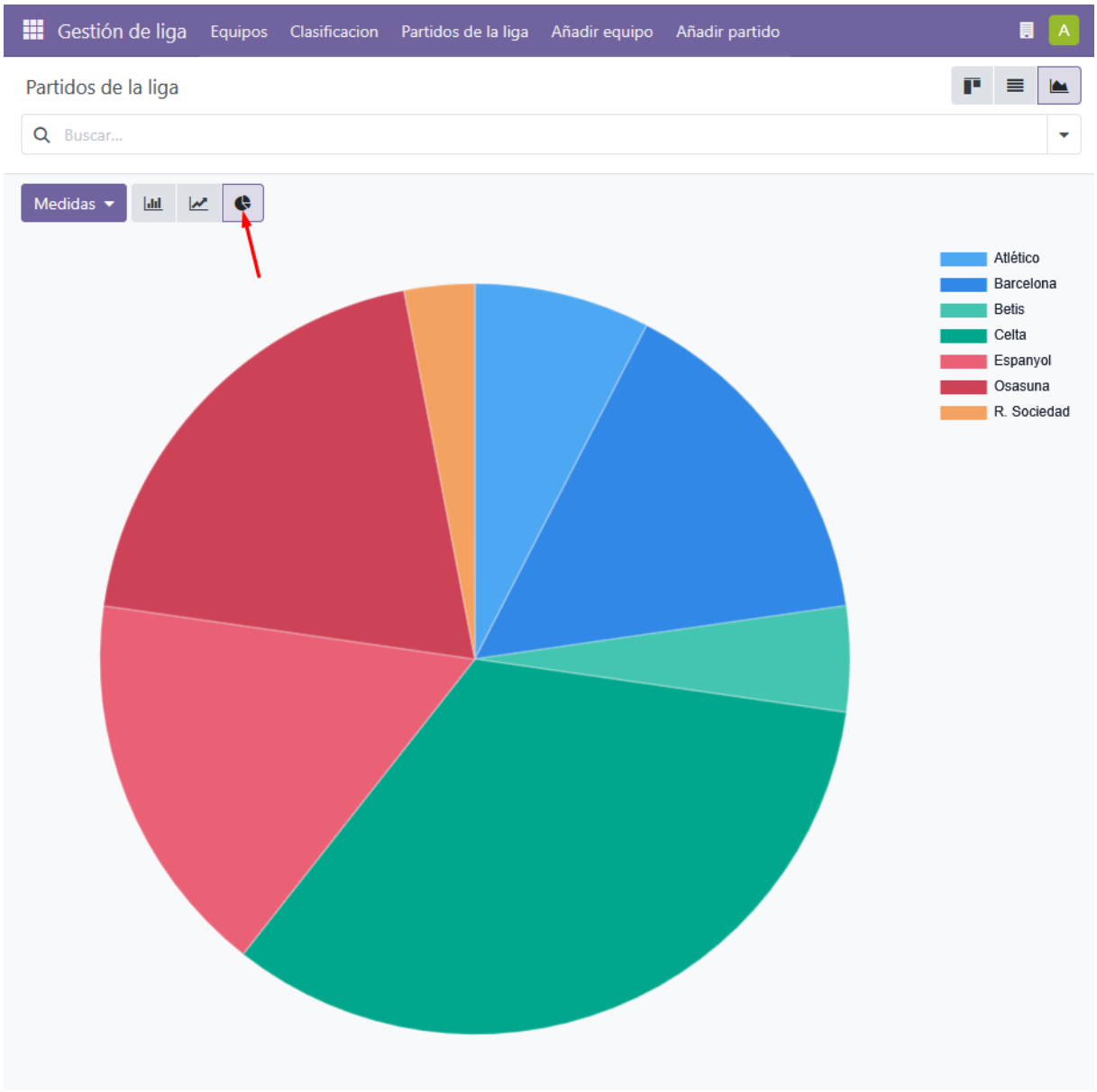


Figura 33: Aparece la nueva vista Graph, clicamos en ella


Gestión de liga
Equipos
Clasificacion
Partidos de la liga
Añadir equipo
Añadir partido
Partidos de la liga
Buscar...
Medidas
Goles equipo local
25,00
20,00
15,00
10,00
5,00
0,00
Atlético
Barcelona
Betis
Celta
Espanyol
Osasuna
R. Sociedad


Gestión de liga
Equipos
Clasificacion
Partidos de la liga
Añadir equipo
Añadir partido
Partidos de la liga
Buscar...
Medidas
Goles equipo local
25,00
20,00
15,00
10,00
5,00
0,00
Atlético
Barcelona
Betis
Celta
Espanyol
Osasuna
R. Sociedad



Como hemos podido apreciar en las capturas anteriores, los distintos modos de mostrar los datos (gráfica de barras, gráfica de líneas y gráfica circular) funcionan correctamente.

La implementación de la vista gráfica en la vista de partidos ha sido satisfactoria.

3. Actividad 02 – Pruebas *EJ08-API-REST_Socio*

Para la realización de esta actividad he empleado las siguientes herramientas:

- ***Kdenlive:*** es un potente editor de video no lineal, libre y de código abierto, basado en el framework MLT. Funciona en Linux, Windows y macOS, permitiendo la edición multipista de audio y video con múltiples efectos, transiciones y herramientas de corrección de color, ideal tanto para usuarios principiantes como profesionales.
- ***Luvvoice:*** es una plataforma de inteligencia artificial diseñada para convertir texto en voz (Text-to-Speech) de forma realista y natural. Es una herramienta muy utilizada por creadores de contenido para narrar videos de YouTube o TikTok, así como por educadores para crear audiolibros o material didáctico.

El vídeo creado se divide en dos partes bien diferenciadas. En la primera parte se muestra el código de la API donde se puede apreciar la url en la que realizar las peticiones y el formato en el que deben enviarse los datos. En la segunda parte se muestran las peticiones http (POST, GET, PUT y DELETE) hechas con el programa ***Postman***.

La duración del vídeo no debe exceder de los 3 minutos (en este caso dura aproximadamente 2 minutos y medio).

El archivo *Actividad 2.mp4* está ubicado en la carpeta *practica5* del proyecto.

4. Actividad 03 – Bot Telegram API REST

Se nos solicita crear un bot de Telegram (configurado con BotFather) que escuche ordenes enviadas por el usuario que apunten a la API REST del módulo "EJ08-API-REST-Socios".

Las órdenes que debe soportar el bot serán:

- **Crear**, nombre="nombre", apellidos="apellidos", num_socio="numerosocio"
- **Modificar**, nombre="nombre", apellidos="apellidos", num_socio="numerosocio"
- **Consultar**, num_socio="numerosocio"
- **Borrar**, num_socio="numerosocio"

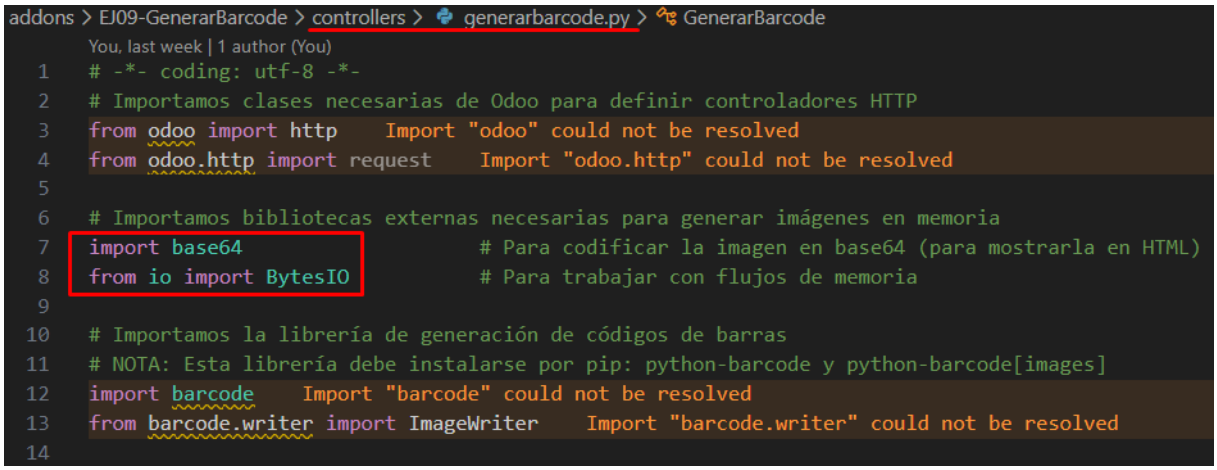
Si no recibe ninguna de las órdenes anteriores deberá devolver la respuesta "Orden no soportada". Lo primero que haremos será descargarnos las librerías necesarias en la máquina real.

5. Actividad 04 – Generación imágenes aleatorias Web Controller

Se nos pide crear un Web Controller que:

- Utilice la librería **Pillow**
- Reciba como parámetros el ancho y alto de una imagen.
- Genere una imagen compuesta por píxeles aleatorios.
- Devuelva la imagen codificada en Base64 o directamente como binario PNG.
- Se recomienda ojear el módulo "*EJ09-GenerarBarcode*"

Tal como nos recomienda el enunciado vamos a ojear el archivo *generarbarcode.py* dentro de la carpeta *controllers* del módulo *EJ09-GenerarBarcode*:



Podemos ver en la "preparación del entorno"la importación de 2 herramientas clave: **BytesIO** y **base64**. La primera permite almacenar un archivo en la memoria RAM de forma temporal en vez de en el disco duro del servidor, evitando una posible saturación en el supuesto de que cientos o miles de usuarios intenten generar dicho archivo al mismo tiempo. La segunda se encarga de traducir los datos binarios de un archivo y convertirlos a una cadena de caracteres seguros, lo cual beneficia su procesado en el lenguaje HTML.

addons > EJ09-GenerarBarcode > controllers > generarbarcode.py > ...

```python
class GenerarBarcode(http.Controller):

    '''
    Este método permite generar un código de barras EAN-13 desde una URL,
    sin necesidad de autenticación. Ideal para integración con otras apps o para pruebas.

    Ejemplo de uso:
    http://localhost:8069/generador/123456789012

    Mostrará una imagen HTML con el código de barras generado.
    '''

    # Ruta expuesta públicamente (auth='public'), sin restricciones CORS (cors='*')
    @http.route('/generador/<codigo>', auth='public', cors='*', type='http')
    def crearBarcode(self, codigo, **kw):
        # Obtenemos la clase del tipo de código de barras que vamos a usar (EAN13)
        EAN = barcode.get_barcode_class('ean13')

        # Aseguramos que el código tenga 12 dígitos rellenando con ceros a la izquierda
        # NOTA: El EAN-13 requiere 12 dígitos + 1 dígito de control (calculado automáticamente)
        codigo_ean = str(codigo).zfill(12)

        # Creamos el objeto del código de barras, pasándole el escritor de imágenes
        ean = EAN(codigo_ean, writer=ImageWriter())

        # Creamos un flujo en memoria para guardar la imagen generada
        fp = BytesIO()

        # Escribimos la imagen del código de barras en el flujo
        ean.write(fp)

        # Codificamos el flujo de bytes en base64 para poder usarlo en HTML
        img_str = base64.b64encode(fp.getvalue()).decode("utf-8")

        # Devolvemos una pequeña vista HTML que muestra la imagen generada
        return '<div><img src="' + img_str + '"/></div>'
```

El código de esta función es bastante autoexplicativo. Primero genera un código de barras, luego lo almacena en un buffer temporal, después transforma todo el binario a texto y finalmente lo muestra en formato HTML.

Tomando este código como inspiración vamos a crear un nuevo Web Controller que haga exactamente lo mismo, solo que en vez de generar un código de barras generará la imagen de píxeles aleatorios que nos han solicitado. Por gusto crearé el controlador dentro del módulo *EJ07-LigaFutbol* (ya que ha sido el más utilizado en toda esta práctica) dentro de su respectiva carpeta *controllers* (importante referenciarlo en su respectivo __init__.py). El nuevo archivo se llamará *imagen_aleatoria.py*.

addons > EJ07-LigaFutbol > controllers > imagen_aleatoria.py > ...

```python
# -*- coding: utf-8 -*-
from odoo import http # type: ignore
from odoo.http import request # type: ignore

import base64
from io import BytesIO

from PIL import Image # type: ignore
import random

class ImagenAleatoria(http.Controller):

    @http.route('/imagen/<int:ancho>/<int:alto>', auth='public', type='http')
    def crearImagen(self, ancho, alto):
        # CREAMOS IMAGEN VACÍA EN MODO RGB
        imagen = Image.new('RGB', (ancho, alto))
        pixeles = imagen.load()

        # CREAMOS BUCLES ANIDADOS PARA RECORRER CADA PÍXEL DE LA IMAGEN,
        # CREANDO UN COLOR DE FORMA ALEATORIA EN CADA ITERACIÓN
        for x in range(ancho):
            for y in range(alto):
                r = random.randint(0, 255)
                g = random.randint(0, 255)
                b = random.randint(0, 255)
                pixeles[x, y] = (r, g, b)

        # UTILIZAMOS BytesIO COMO BUFFER EN RAM
        buffer = BytesIO()

        # GUARDAMOS LA IMAGEN Pillow EN EL BUFFER COMO PNG
        imagen.save(buffer, format="PNG")

        # TRADUCIMOS EL CONTENIDO DEL BUFFER A base64
        imagenB64 = base64.b64encode(buffer.getvalue()).decode("utf-8")

        # DEVOLVEMOS EL HTML MOSTRANDO LA IMAGEN RECIÉN CREADA
        return '''<div><h2>PÍXELES ALEATORIOS(%dx%d)</h2>
            <img src="%s"/></div>''' % (ancho, alto, imagenB64)
```

LIBRERÍA PILLOW YA INCLUÍDA EN LA IMAGEN DE ODOO PARA DOCKER.
SI FUESE NECESARIO INSTALARLA, ENTRAR AL CONTENEDOR MEDIANTE TERMINAL Y ESCRIBIR "pip install pillow"

De forma muy similar al código de ejemplo anterior, importamos las librerías necesarias (en este caso la librerías clave son base64, BytesIO y especialmente Pillow) y creamos una función (Web Controller) que genere una imagen, la llene de píxeles aleatorios, la almacene temporalmente en RAM, la traduzca a cadena de caracteres y la muestre por pantalla en la ruta especificada, que en este caso sería *http://localhost:9001/imagen/'ancho'/'alto'*, siendo 'ancho' y 'alto' 2 números enteros.

A continuación una muestra del funcionamiento del Web Controller. Le diremos al controlador que nos devuelva una imagen de 500 píxeles de ancho y 500 píxeles de alto generada con píxeles de colores aleatorios.