

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

Sistemas de Gestión Empresarial

Cristian Fernández

30 de enero de 2026

Índice

1. Introducción	2
2. Actividad 01 – Modificación <i>EJ07-LigaFutbol</i>	3
2.1. Reglas de puntuación especiales	3
2.2. Botones para alterar los goles de los partidos	11
2.3. Web Controller para eliminar empates	17
2.4. Informe PDF por cada partido	21
2.5. Wizard para crear nuevos partidos	27
2.6. Vista Graph	32
3. Actividad 02 – Pruebas <i>EJ08-API-REST_Socio</i>	35
4. Actividad 03 – Bot Telegram API REST	36
5. Actividad 04 – Generación imágenes aleatorias Web Controller	37

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 órdenes 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:

```
addons > EJ07-LigaFutbol > views > liga_equipo.xml
  2 <odoo>
  3
  4     <record id='liga_equipo_action' model='ir.act.list'>
  5         <field name='name'>Equipos de la liga</field>
  6         <field name='res_model'>liga.equipo</field>
  7         <!-- Indicamos que este modelo se puede ver en kanban, tree y form -->
  8         <field name='view_mode'>tree,kanban,form,graph</field>
  9     </record>
 10
 11     <menuitem name="Gestión de liga" id="liga_base_menu" />
 12     <menuitem name="Equipos" id="liga_equipo_menu" parent="liga_base_menu" action="liga_equipo_action" />
 13
 14     <!-- Vista formulario -->
 15     <record id="liga_equipo_view_form" model="ir.ui.view">
 16         <field name="name">Formulario de equipo de futbol</field>
 17         <field name="model">liga.equipo</field>
 18         <field name="arch" type="xml">
 19             <form>
 20                 <group>
 21                     <group>
 22                         <field name="nombre" />
 23                         <field name="escudo" widget="image" class="oe_avatar" />
 24                         <field name="fecha_fundacion" />
 25                     </group>
 26                     <group>
 27                         <field name="descripcion" />
 28                     </group>
 29                 </group>
 30             </form>
 31         </field>
 32     </record>
 33
 34     <!-- Tree(list) View -->
 35     <record id="liga_equipo_view_tree" model="ir.ui.view">
 36         <field name="name" value="Lista de Equipos de la liga" />
 37         <field name="model" value="liga.equipo" />
 38         <field name="arch" type="xml">
 39             <tree>
```

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:

<input type="checkbox"/>	Nombre equipo	Escudo equipo	Fecha fundación
<input type="checkbox"/>	Atlético		26/04/1903
<input type="checkbox"/>	Barcelona		29/11/1899
<input type="checkbox"/>	Betis		12/09/1907
<input type="checkbox"/>	Celta		23/08/1923
<input type="checkbox"/>	Espanyol		28/10/1900
<input type="checkbox"/>	Girona		23/07/1930
<input type="checkbox"/>	Osasuna		24/10/1920
<input type="checkbox"/>	R. Madrid		06/03/1902
<input type="checkbox"/>	R. Sociedad		07/09/1909
<input type="checkbox"/>	Villarreal		10/03/1923

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

<input type="checkbox"/> Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A Favor	Goles En Contra	Victorias	Empates	Derrotas
	Barcelona	3	2	5	1	1	0	1
	Celta	3	1	1	0	1	0	0
	Girona	0	0	0	0	0	0	0
	Osasuna	0	0	0	0	0	0	0
	R. Sociedad	0	0	0	0	0	0	0
	Villarreal	0	0	0	0	0	0	0
	R. Madrid	0	1	0	5	0	0	1
	Betis	0	0	0	0	0	0	0
	Atlético	0	0	0	0	0	0	0
	Espanyol	0	0	0	0	0	0	0

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     puntos= fields.Integer( compute="_compute_puntos",default=0, store=True)
56
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
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106

for recordPartido in self.env['liga.partido'].search([]):
    # AÑADIMOS NUEVA VARIABLE PARA SIMPLIFICAR LOS IF
    goles_de_diferencia = abs(recordPartido.goles_casa - recordPartido.goles_fuera)

    #Si es el equipo de casa
    if recordPartido.equipo_casa.nombre==recordEquipo.nombre:

        #Miramos si es victoria o derrota
        if recordPartido.goles_casa>recordPartido.goles_fuera:
            recordEquipo.victorias=recordEquipo.victorias+1
            # APPLICAMOS LA LÓGICA NUEVA (VICTORIAS)
            if goles_de_diferencia >= 4:
                recordEquipo.puntos += 7
            else:
                recordEquipo.puntos += 3

        elif recordPartido.goles_casa<recordPartido.goles_fuera:
            recordEquipo.derrotas=recordEquipo.derrotas+1
            # APPLICAMOS LA LÓGICA NUEVA (DERROTAS)
            if goles_de_diferencia >= 4:
                recordEquipo.puntos -= 1
            else:
                pass

        else:
            recordEquipo.empates=recordEquipo.empates+1
            # AÑADIMOS PUNTUACIÓN ESTÁNDAR EN CASO DE EMPATE
            recordEquipo.puntos += 1

    #Sumamos goles a favor y en contra
    recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_casa
    recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_fuera

```

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
133     #Sumamos goles a favor y en contra
134     recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_fuera
135     recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_casa

```

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.

Gestión de liga		Equipos	Clasificación	Partidos de la liga	Añadir equipo		
Nuevo		Partidos de la liga					
<input type="text"/> Buscar...							
- Resultado -	Barcelona : 2	R. Madrid : 2	- Resultado -	Celta : 4	Atlético : 0		
- Resultado -	Celta : 5	Espanyol : 1	- Resultado -	Olasuna : 6	Celta : 2		
- Resultado -	Espanyol : 5	Atlético : 0					

Figura 8: Crearemos nuevos partidos de prueba

Gestión de liga		Equipos	Clasificación	Partidos de la liga	Añadir equipo			
Nuevo		Clasificación de la liga						
<input type="text"/> Buscar...								
<input type="checkbox"/> Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A F...	Goles En ...	Victorias	Empates	Derrotas
<input type="checkbox"/>	 Celta	13	3	11	7	2	0	1
<input type="checkbox"/>	 Osasuna	7	1	6	2	1	0	0
<input type="checkbox"/>	 Espanyol	6	2	6	5	1	0	1
<input type="checkbox"/>	 Barcelona	1	1	2	2	0	1	0
<input type="checkbox"/>	 R. Madrid	1	1	2	2	0	1	0
<input type="checkbox"/>	 Betis	0	0	0	0	0	0	0
<input type="checkbox"/>	 Girona	0	0	0	0	0	0	0
<input type="checkbox"/>	 R. Sociedad	0	0	0	0	0	0	0
<input type="checkbox"/>	 Villarreal	0	0	0	0	0	0	0
<input type="checkbox"/>	 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.

```
addons > EJ07-LigaFutbol > models > liga_partido.py > ...
6   class LigaPartido(models.Model):
154     #Sobreescribo el metodo crear
155     @api.model
156     def create(self, values):
157       #hago lo normal del metodo create
158       result = super().create(values)
159       #Añado esto: llamo a la funcion que actualiza la clasificacion
160       self.actualizoRegistrosEquipo()
161       #hago lo normal del metodo create
162       return result
163
164     #NUEVAS FUNCIONES PARA LA BOTONERA DE GOLES
165     def sumar_goles_equipos_locales(self):
166       # BUSCAMOS TODOS LOS PARTIDOS
167       partidos = self.search([])
168
169       for partido in partidos:
170         partido.goles_casa += 2
171
172       # ACTUALIZAMOS CLASIFICACIÓN
173       self.actualizoRegistrosEquipo()
174
175     def sumar_goles_equipos_visitantes(self):
176       # BUSCAMOS TODOS LOS PARTIDOS
177       partidos = self.search([])
178
179       for partido in partidos:
180         partido.goles_fuera += 2
181
182       # ACTUALIZAMOS CLASIFICACIÓN
183       self.actualizoRegistrosEquipo()
184 |
```

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 obligatoriamente 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             </list>
62         </field>

```

```
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)

Equipo local	Goles Casa	Equipo visitante	Goles Fuera
<input type="checkbox"/> Barcelona	6	R. Madrid	13
<input type="checkbox"/> Celta	8	Atlético	6
<input type="checkbox"/> Celta	9	Espanyol	7
<input type="checkbox"/> Osasuna	10	Celta	8
<input type="checkbox"/> Espanyol	9	Atlético	6

Figura 12: Vista Lista

Probemos la funcionalidad de los botones:

The screenshot shows a web-based application for managing a league. At the top, there's a navigation bar with links like 'Gestión de liga', 'Equipos', 'Clasificación', 'Partidos de la liga', and 'Añadir equipo'. Below the navigation is a toolbar with buttons for 'Nuevo', '+2 goles equipos locales' (which has a red arrow pointing to it), '+2 goles equipos visitantes', and 'Partidos de la liga'. There's also a search bar and some pagination controls ('1-5 / 5'). A red box highlights the 'Partidos de la liga' button. In the main area, there are several cards representing match results. One card for 'Barcelona : 6' and 'R. Madrid : 13' is visible. A modal dialog box titled 'Confirmación' is overlaid on the page, asking 'Esta acción sumará 2 goles a todos los equipos locales. ¿Estás seguro/a?'. It has two buttons: 'Aceptar' (highlighted with a red arrow) and 'Cancelar'.

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

This screenshot shows the same application interface after the action from Figure 13 has been completed. The match results have been updated. The first card now shows 'Barcelona : 8' and 'R. Madrid : 13'. The second card shows 'Celta : 11' and 'Espanyol : 7'. The third card shows 'Osasuna : 12' and 'Celta : 8'. The other cards are partially visible.

Figura 14: El resultado es satisfactorio

The screenshot shows a table of football match results. A red arrow points from the '+2 goles equipos locales' button in the top navigation bar to the 'Aceptar' button in a modal dialog box. Another red arrow points from the 'Aceptar' button in the dialog to the text inside it.

Equipo local	Goles Casa	Equipo visitante	Goles Fu...
<input type="checkbox"/> Barcelona	8	R. Madrid	13
<input type="checkbox"/> Celta	10	Atlético	6
<input type="checkbox"/> Celta	11	Espanyol	7
<input type="checkbox"/> Osasuna	12	Celta	8
<input type="checkbox"/> Espanyol	11	Atlético	6

Confirmación

Esta acción sumará 2 goles a todos los equipos visitantes. ¿estás seguro/a?

Aceptar Cancelar

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

The screenshot shows the same table of match results after the update. The 'Aceptar' button was clicked, and the 'Goles Fu...' column has been updated to reflect the new total goals. A red arrow points from the 'Aceptar' button in the previous dialog to the 'Goles Fu...' column in this table.

Equipo local	Goles Casa	Equipo visitante	Goles Fu...
<input type="checkbox"/> Barcelona	8	R. Madrid	15
<input type="checkbox"/> Celta	10	Atlético	8
<input type="checkbox"/> Celta	11	Espanyol	9
<input type="checkbox"/> Osasuna	12	Celta	10
<input type="checkbox"/> Espanyol	11	Atlético	8

Figura 16: El resultado es satisfactorio

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

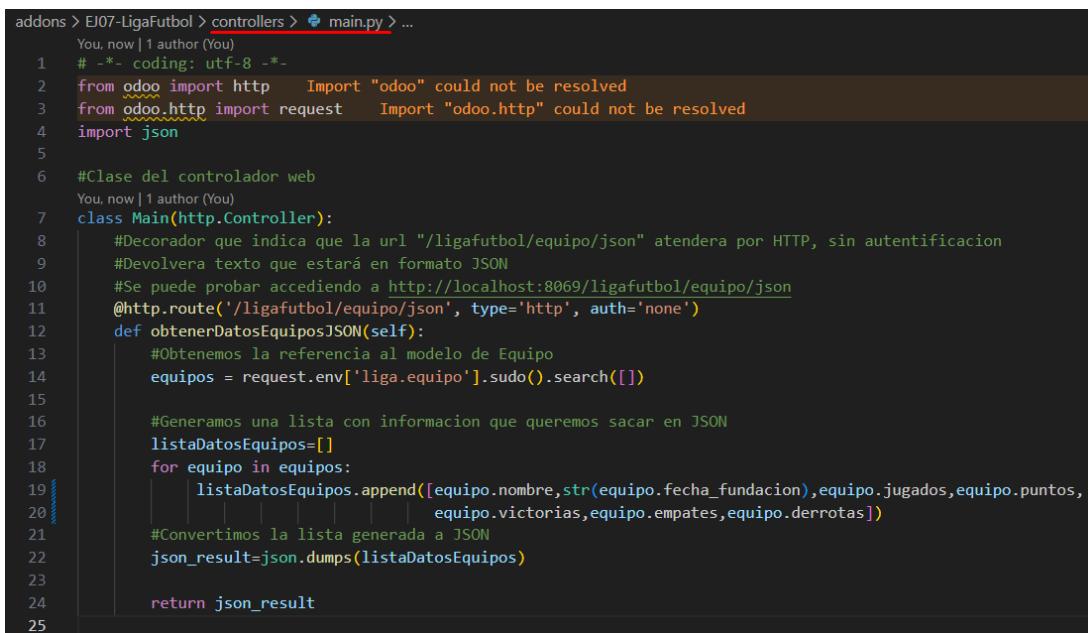
<input type="checkbox"/> Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A Favor	Goles En Contra	Victorias	Empates	Derrotas
<input type="checkbox"/>	R. Madrid	7	1	15	8	1	0	0
<input type="checkbox"/>	Celta	6	3	31	29	2	0	1
<input type="checkbox"/>	Olasuna	3	1	12	10	1	0	0
<input type="checkbox"/>	Espanyol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

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)
1  # -*- coding: utf-8 -*-
2  from odoo import http      Import "odoo" could not be resolved
3  from odoo.http import request    Import "odoo.http" could not be resolved
4  import json
5
6  #Clase del controlador web
7  You, now | 1 author (You)
8  class Main(http.Controller):
9      #Decorador que indica que la url "/ligafutbol/equipo/json" atenderá por HTTP, sin autenticación
10     #Devolverá texto que estará en formato JSON
11     #Se puede probar accediendo a http://localhost:8069/ligafutbol/equipo/json
12     @http.route('/ligafutbol/equipo/json', type='http', auth='none')
13     def obtenerDatosEquiposJSON(self):
14         #Obtenemos la referencia al modelo de Equipo
15         equipos = request.env['liga.equipo'].sudo().search([])
16
17         #Generamos una lista con información que queremos sacar en JSON
18         listaDatosEquipos=[]
19         for equipo in equipos:
20             listaDatosEquipos.append([equipo.nombre,str(equipo.fecha_fundacion),equipo.jugados,equipo.puntos,
21                                     equipo.victorias,equipo.empates,equipo.derrotas])
22         #Convertimos la lista generada a JSON
23         json_result=json.dumps(listaDatosEquipos)
24
25     return json_result
```

Figura 18: Web Controller por defecto

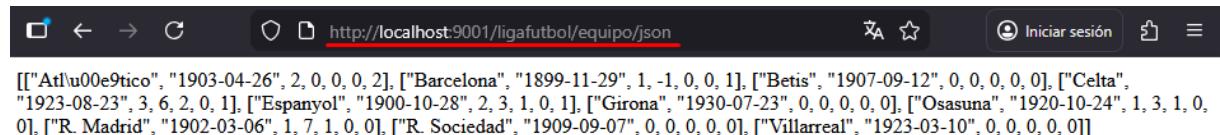


Figura 19: Comprobamos que efectivamente funciona

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:

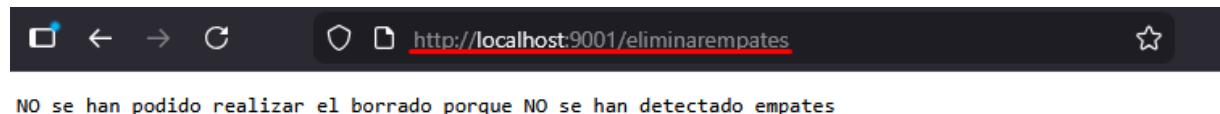


Figura 20: Este resultado es correcto

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

The screenshot shows a web interface for managing a league. The top navigation bar includes tabs for 'Gestión de liga', 'Equipos', 'Clasificación', and 'Partidos de la liga' (which is highlighted with a red box). Below the navigation is a search bar and a toolbar with icons for 'Nuevo', '+2 goles equipos locales', '+2 goles equipos visitantes', and 'Partidos de la liga'. The main content area displays seven match results in boxes:

- 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 -
Girona : 0
R. Sociedad : 0
- Resultado -
Betis : 2
Espanyol : 2

Figura 21: Creamos 2 empates de prueba

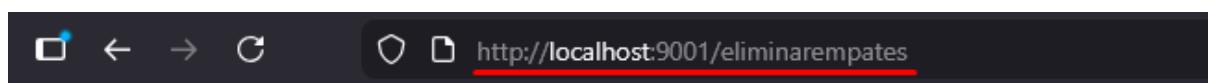


Figura 22: Ahora el resultado es el correcto tras la eliminación de empates

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

<input type="checkbox"/> 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)
1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3
4      <!-- Ejemplo de informe asociado a equipo (se genera al darle aimprimir
5          desde la vista Form de un equipo) -->
6      <template id="report_clasificacion_view">
7          <!-- Aquí la estructura a seguir -->
8          <t t-call="web.html_container">
9              <t t-foreach="docs" t-as="doc">
10                 <t t-call="web.internal_layout">
11                     <div class="page">
12                         <!-- Cabecera con el nombre de equipo -->
13                         <h2 t-field="doc.nombre" />
14                         <!-- Datos de los partidos jugados por el equipo -->
15                         <p>
16                             Partidos jugados:
17                             <span t-field="doc.jugados" />
18                         </p><p>
19                             Partidos ganados:
20                             <span t-field="doc.victorias" />
21                         </p><p>
22                             Partidos empatados:
23                             <span t-field="doc.empates" />
24                         </p><p>
25                             Partidos perdidos:
26                             <span t-field="doc.derrotas" />
27                         </p><p>
28                             Puntos obtenidos:
29                             <span t-field="doc.puntos" />
30                         </p>
31                     </div>
32                 </t>
33             </t>
34         </t>
35     </template>
36
37     <!-- Realmente, este es el informe, lo de arriba es la plantilla que utilizará el informe -->
38     <report id="report_clasificacion" model="liga.equipo" string="Informe clasificación de cada equipo"
39         name="EJ07-LigaFutbol.report_clasificacion_view" file="EJ07-LigaFutbol.report_clasificacion_view"
40         report_type="qweb-pdf" />
41
42
43 </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:

```
addons > EJ07-LigaFutbol > __manifest__.py
2  {
10     #Indicamos que es una aplicación
11     'application': True,
12     'author': "Sergi García",
13     'website': "http://apuntesfpinformatica.es",
14     'category': 'Tools',
15     'version': '0.1',
16     'depends': ['base'],
17
18     'data': [
19
20         #Estos dos primeros ficheros:
21         #1) El primero indica grupo de seguridad basado en rol
22         #2) El segundo indica la política de acceso del modelo
23         #Mas información en https://www.odoo.com/documentation/17.0/es/developer/howtos/rdtraining/05_securityintro.html
24         #Y en www.odoo.yenthevg.com/creating-security-groups-odoo/
25         #'security/groups.xml',
26         #'security/ir.model.access.csv',
27
28         #Aqui distintas vistas de equipo (vistas diferentes, mismo modelo)
29         'views/liga_equipo.xml',
30         'views/liga_equipo_clasificacion.xml',
31         #Vista a un informe
32         'report/liga_equipo_clasificacion_report.xml',
33         'report/liga_partido_report.xml', ←
34         #Aqui vista sobre los partidos
35         'views/liga_partido.xml',
36         #Añadimos un Wizard para introducir equipos
37         'wizard/liga_equipo_wizard.xml'
38
39     ],
40     # Fichero con data de demo si se inicializa la base de datos con "demo data" (No incluido en ejemplo)
41     # 'demo': [
42     #     'demo.xml'
43     # ],
44
45 }
```

¡AVISO! ES IMPORTANTE QUE LA PLANTILLA VAYA ANTES QUE LA VISTA PORQUE DE LO CONTRARIO PUEDE HABER CONFLICTOS

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
1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3
4      <record id="action_report_liga_partido" model="ir.actions.report"> ← ESTE ID SERVIRÁ PARA ENLAZAR CON
5          <field name="name">Acta del Partido</field> EL BOTÓN DE LA VISTA
6          <field name="model">liga.partido</field>
7          <field name="report_type">qweb-pdf</field>
8          <field name="report_name">EJ07-LigaFutbol.report_partido_liga_template</field> ← ESPECIFICAR NOMBRE MÓDULO
9          <field name="report_file">EJ07-LigaFutbol.report_partido_liga_template</field> Y ID TEMPLATE
10         <field name="print_report_name">'Partido - %s vs %s' % (object.equipo_casa.nombre,
11                                         object.equipo_fuera.nombre)</field>
12         <field name="binding_model_id" ref="model_liga_partido"/>
13         <field name="binding_type">report</field>
14     </record>
15
16     <template id="report_partido_liga_template"> ←
17         <t t-call="web.html_container">
18             <t t-foreach="docs" t-as="doc">
19                 <t t-call="web.external_layout">
20                     <div class="page">
21                         <div class="text-center" style="border-bottom: 2px solid black; margin-bottom: 20px;">
22                             <h2>Acta Oficial de Partido</h2>
23                         </div>
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

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

```

24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

```

<h3 style="margin-bottom: 10px;" t-field="doc.equipo_casa.nombre"></h3> <div style="font-size: 48px; font-weight: bold;"> </div> <p style="color: #666; font-size: 12px;">LOCAL</p>	VS	<h3 style="margin-bottom: 10px;" t-field="doc.equipo_fuera.nombre"></h3> <div style="font-size: 48px; font-weight: bold;"> </div> <p style="color: #666; font-size: 12px;">VISITANTE</p>
--	---	--

Figura 25: Esta parte mostrará los equipos y el resultado

```

48         <div style="width: 100%; margin-top: 30px;">
49             <table class="table table-sm" style="width: 100%; border-top: 1px solid #dee2e6;">
50                 <thead>
51                     <tr style="background-color: #f8f9fa;">
52                         <th style="padding: 8px; text-align: left; border-bottom: 2px solid #333;">Detalle</th>
53                         <th style="padding: 8px; text-align: right; border-bottom: 2px solid #333;">Resultado</th>
54                     </tr>
55                 </thead>
56                 <tbody>
57                     <tr>
58                         <td style="padding: 8px; border-bottom: 1px solid #eee;">Estado del encuentro</td>
59                         <td style="padding: 8px; text-align: right; border-bottom: 1px solid #eee;">
60                             <span class="badge badge-secondary" style="text-transform: uppercase;">Finalizado</span>
61                         </td>
62                     </tr>
63                     <tr>
64                         <td style="padding: 8px; border-bottom: 1px solid #eee;">Diferencia de goles</td>
65                         <td style="padding: 8px; text-align: right; border-bottom: 1px solid #eee; font-weight: bold;">
66                             <t t-esc="abs(doc.goles_casa - doc.goles_fuera)" />
67                         </td>
68                     </tr>
69                 </tbody>
70             </table>
71         </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
2   <odoo>
14     <!-- VISTA DE FORMULARIO -->
15     <record id="liga_partido_view_form" model="ir.ui.view">
16         <field name="name">Formulario Partidos</field>
17         <field name="model">liga.partido</field>
18         <field name="arch" type="xml">
19             <form>
20                 <button name="%(%(EJ07-LigaFutbol.action_report_liga_partido)d" string="Generar PDF" type="action" class="btn-primary"/>
21             <group>
22                 <group>
23                     <field name="equipo_casa" />
24                     <field name="goles_casa" />
25                 </group>
26
27                 <group>
28                     <field name="equipo_fuera" />
29                     <field name="goles_fuera" />
30                 </group>
31             </form>
32         </field>
33     </record>
34

```

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 ESTRÍCTAMENTE 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:

Gestión de liga

Nuevo +2 goles equipos locales +2 goles equipos visitantes Partidos de la liga Añadir equipo

1-5 / 5 < >

Buscar...

- Resultado -	- Resultado -
Barcelona : 8 R. Madrid : 15	Celta : 10 Atlético : 8
- Resultado -	- Resultado -
Celta : 11 Espanyol : 9	Osasuna : 12 Celta : 10
- Resultado -	
Espanyol : 11 Atlético : 8	

Figura 28: Clicamos en un partido

Gestión de liga

Nuevo Partidos de la liga liga.partido.9

2 / 5 < >

Generar PDF	Equipo local Celta
	Goles Casa 10
	Equipo visitante Atlético
	Goles Fueras 8

Figura 29: Clicamos en el nuevo botón

Acta Oficial de Partido

Acta Oficial de Partido

Celta **10** VS **Atlético** **8**

LOCAL VISITANTE

Detalle	Resultado
Estado del encuentro	FINALIZADO
Diferencia de goles	2

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.

The screenshot shows the Odoo 'Gestión de liga' (League Management) module. At the top, there are tabs for 'Gestión de liga', 'Equipos', 'Clasificación', 'Partidos de la liga', and 'Añadir equipo'. Below the tabs, there are buttons for 'Nuevo', '+2 goles equipos locales', '+2 goles equipos visitantes', and 'Partidos de la liga'. A status bar indicates '5 seleccionado' (5 selected). An 'Imprimir' (Print) button is highlighted with a red arrow. To its right is an 'Acciones' (Actions) button. The main area displays a table of matches:

Equipo local	Goles Casa	Equipo visitante	Goles Fu...
Barcelona	8	R. Madrid	15
Celta	10	Atlético	8
Celta	11	Espanyol	9
Osasuna	12	Celta	10
Espanyol	11	Atlético	8

The screenshot shows a PDF viewer displaying a generated PDF document. The document is titled 'Acta Oficial de Partido'. It features a header with the teams: 'Barcelona' and 'R. Madrid'. The score is listed as '8 VS 15'. Below the score, it says 'LOCAL' under Barcelona and 'VISITANTE' under R. Madrid. A table at the bottom provides details: 'Estado del encuentro' (FINALIZADO) and 'Diferencia de goles' (7). On the left side of the PDF viewer, a vertical list of other generated PDF documents is visible, with one item highlighted by a red box.

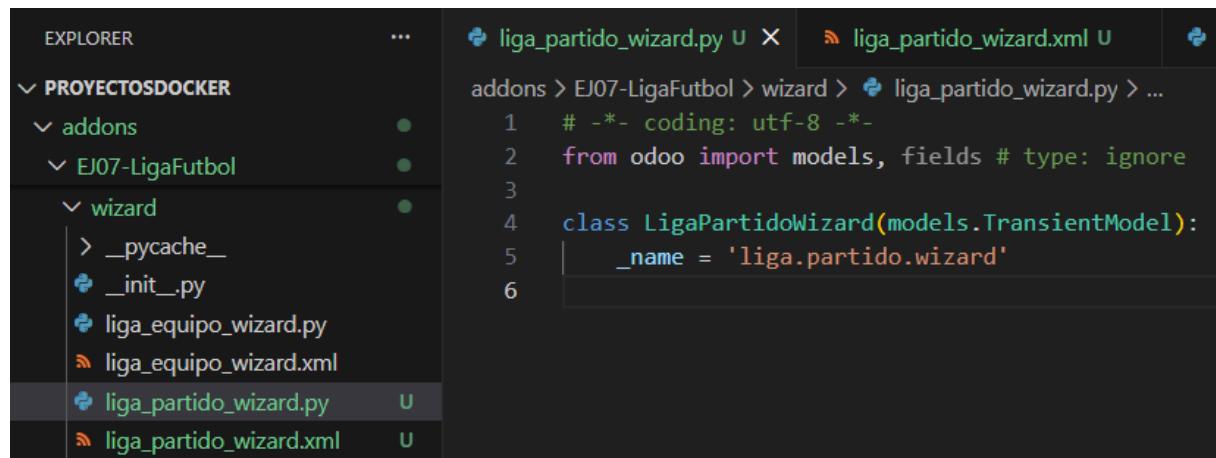
Figura 31: ¿El resultado? Un PDF con todos los partidos

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*.

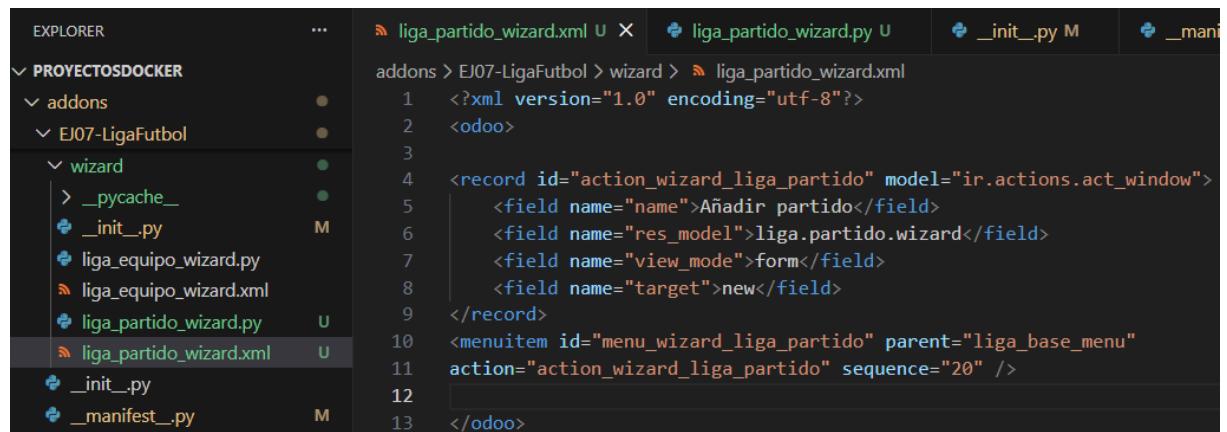
2.5. Wizard para crear nuevos partidos

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`.



```
liga_partido_wizard.py U X liga_partido_wizard.xml U
addons > EJ07-LigaFutbol > wizard > liga_partido_wizard.py > ...
1  # -*- coding: utf-8 -*-
2  from odoo import models, fields # type: ignore
3
4  class LigaPartidoWizard(models.TransientModel):
5      _name = 'liga.partido.wizard'
6
```



```
liga_partido_wizard.xml U X liga_partido_wizard.py U __init__.py M __manifest__.py M
addons > EJ07-LigaFutbol > wizard > liga_partido_wizard.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <odoo>
3
4 <record id="action_wizard_liga_partido" model="ir.actions.act_window">
5     <field name="name">Añadir partido</field>
6     <field name="res_model">liga.partido.wizard</field>
7     <field name="view_mode">form</field>
8     <field name="target">new</field>
9 </record>
10 <menuitem id="menu_wizard_liga_partido" parent="liga_base_menu"
11     action="action_wizard_liga_partido" sequence="20" />
12
13 </odoo>
```

En las siguiente página los enlaces:

EXPLORER ...

PROYECTOSDOCKER

- addons
 - EJ07-LigaFutbol
 - wizard
 - _pycache_
 - _init_.py M
 - liga_equipo_wizard.py
 - liga_equipo_wizard.xml
 - liga_partido_wizard.py U
 - liga_partido_wizard.xml U

addons > EJ07-LigaFutbol > wizard > _init_.py

```
You, 17 seconds ago | 1 author (You)
1 # -*- coding: utf-8 -*-
2 from . import liga_equipo_wizard
3 from . import liga_partido_wizard
4
```

AÑADIMOS EL NUEVO MODELO

EXPLORER ...

PROYECTOSDOCKER

- addons
 - EJ07-LigaFutbol
 - wizard
 - _pycache_
 - _init_.py M
 - liga_equipo_wizard.py
 - liga_equipo_wizard.xml
 - liga_partido_wizard.py U
 - liga_partido_wizard.xml U
 - _init_.py
 - _manifest_.py M

addons > EJ07-LigaFutbol > _manifest_.py

```
2 {
18     'data': [
19
20
21         #Estos dos primeros ficheros:
22         #1) El primero indica grupo de seguridad basado en rol
23         #2) El segundo indica la politica de acceso del modelo
24         #Mas informacion en https://www.odoo.com/documentation/17.0/es/develop/security/groups.xml',
25         #'security/groups.xml',
26         'security/ir.model.access.csv',
27
28
29         #Aqui distintas vistas de equipo (vistas diferentes, mismo modelo)
30         'views/liga_equipo.xml',
31         'views/liga_equipo_clasificacion.xml',
32         #Vista a un informe
33         'report/liga_equipo_clasificacion_report.xml',
34         'report/liga_partido_report.xml',
35         #Aqui vista sobre los partidos
36         'views/liga_partido.xml',
37         #Añadimos un Wizard para introducir equipos
38         'wizard/liga_equipo_wizard.xml',
39         'wizard/liga_partido_wizard.xml' AÑADIMOS LA NUEVA VISTA
40
41     ],
42
```

Por último los permisos:

EXPLORER ...

PROYECTOSDOCKER

- addons
 - EJ07-LigaFutbol
 - security
 - groups.xml
 - ir.model.access.csv M

addons > EJ07-LigaFutbol > security > ir.model.access.csv

```
You, 17 seconds ago | 1 author (You)
1 id,name,model_id:id,group_id:id,perm_read,perm_write,perm_create,perm_unlink
2 acl_equipo,liga.equipo_default,model_liga_equipo,,1,1,1,1
3 acl_partido,liga.partido_default,model_liga_partido,,1,1,1,1
4 acl_liga_equipo_wizard,liga.equipo_wizard,model_liga_equipo_wizard,,1,1,1,1
5 acl_liga_partido_wizard,liga.partido_wizard,model_liga_partido_wizard,,1,1,1,1
```

AÑADIMOS LOS NUEVOS 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 > ...
1  # -*- coding: utf-8 -*-
2  from odoo import models, fields # type: ignore
3
4  class LigaPartidoWizard(models.TransientModel):
5      _name = 'liga.partido.wizard'
6      _description = 'Wizard para registrar un partido'
7
8      # CAMPOS DEL MODELO QUE USAREMOS EN EL WIZARD
9      equipo_casa = fields.Many2one('liga.equipo', string='Equipo Local', required=True)
10     equipo_fuera = fields.Many2one('liga.equipo', string='Equipo Visitante', required=True)
11     goles_casa = fields.Integer('Goles Local', default=0)
12     goles_fuera = fields.Integer('Goles Visitante', default=0)
13     jornada = fields.Integer('Jornada', default=1, required=False)
14
15     def crear_partido(self):
16         # REFERENCIA DEL MODELO DESTINO
17         modeloLigaPartido = self.env['liga.partido']
18         # RECORREMOS TODO EL MODELO
19         for wiz in self:
20             # POR CADA ELEMENTO SE CREARÁ UN REGISTRO (SÓLO 1 EN ESTE CASO)
21             modeloLigaPartido.create({
22                 'equipo_casa': wiz.equipo_casa.id, ← LOS .id SIEMPRE SON OBLIGATORIOS
23                 'equipo_fuera': wiz.equipo_fuera.id, ← EN LOS CAMPOS Many2One.
24                 'goles_casa': wiz.goles_casa, ← SI NO SE ESPECIFICAN ODOO LANZARÁ
25                 'goles_fuera': wiz.goles_fuera, ← UN ERROR AL INTENTAR REGISTRAR
26                 'jornada': wiz.jornada ← EL PARTIDO.
27             })
28
29             # CIERRA EL WIZARD Y REFRESCA LA VISTA ACTUAL
30             return {'type': 'ir.actions.client', 'tag': 'reload'}
```

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 >
6  class LigaPartido(models.Model):
33     #Goles equipo de casa
34     goles_fuera= fields.Integer()
35
36     # AÑADIMOS UN NUEVO CAMPO JORNADA
37     jornada = fields.Integer()
```

Ahora echemos un vistazo a cómo debe quedar la vista:

```
addons > EJ07-LigaFutbol > wizard > liga_partido_wizard.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <odoo>
3
4  <record id='liga_partido_wizard_form' model='ir.ui.view'>
5      <field name='name'>Añadir Partido</field>
6      <field name='model'>liga.partido.wizard</field>
7      <field name='arch' type='xml'>
8          <form>
9              <group>
10                  <group string="Equipos">
11                      <field name='equipo_casa'/>
12                      <field name='equipo_fuera'/>
13                  </group>
14                  <group string="Resultado y Jornada">
15                      <field name='goles_casa'/>
16                      <field name='goles_fuera'/>
17                      <field name='jornada'/>
18                  </group>
19              </group>
20              <footer>
21                  <button string='Registrar Partido' name='crear_partido' class='btn-primary' type='object'/>
22                  <button string='Cancelar' class='btn-secondary' special='cancel'/>
23              </footer>
24          </form>
25      </field>
26  </record>
27
28  <record id="action_wizard_liga_partido" model="ir.actions.act_window">
29      <field name="name">Añadir partido</field>
30      <field name="res_model">liga.partido.wizard</field>
31      <field name="view_mode">form</field>
32      <field name="target">new</field>
33  </record>
34
35  <menuitem id="menu_wizard_liga_partido"
36          parent="liga_base_menu"
37          action="action_wizard_liga_partido"
38          sequence="20" />
39 </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

Añadir partido

Nuevo +2 goles equipos locales +2 goles equipos visitantes Partidos de la liga

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 -

Osasuna : 1
Girona : 1

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

Añadir partido

Nuevo +2 goles equipos locales +2 goles equipos visitantes Partidos de la liga

Buscar...

- Resultado -

Barcelona : 8
R. Madrid : 15

- Resultado -

Celta : 10
Atlético : 8

- Resultado -

Osasuna : 12
Celta : 10

- Resultado -

Betis : 2
Girona : 2

- Resultado -

Betis : 1
Espanyol : 1

- Resultado -

Barcelona : 2
Espanyol : 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:

```
addons > EJ07-LigaFutbol > views > liga_partido.xml
  2 <odoo>
126    <!-- Vista Graph que muestra información usando como base los goles-->
127    <record model="ir.ui.view" id="liga_partido_view_graph">
128        <field name="name">Goles por equipo</field>
129        <field name="model">liga.partido</field>
130        <field name="type">graph</field>
131        <field name="arch" type="xml">
132            <graph string="Goles equipo local"> EL NOMBRE DE LOS CAMPOS DEBEN
133                <field name="equipo_casa" group="True" type="row"/> COINCIDIR CON LOS DEL MODELO
134                <field name="goles_casa" group="True" type="measure" string="Goles equipo local"/>
135            </graph>
136        </field>
137    </record>
```

Figura 32: Recordad añadir graph en el *view_mode*

Comprobaremos si funciona:

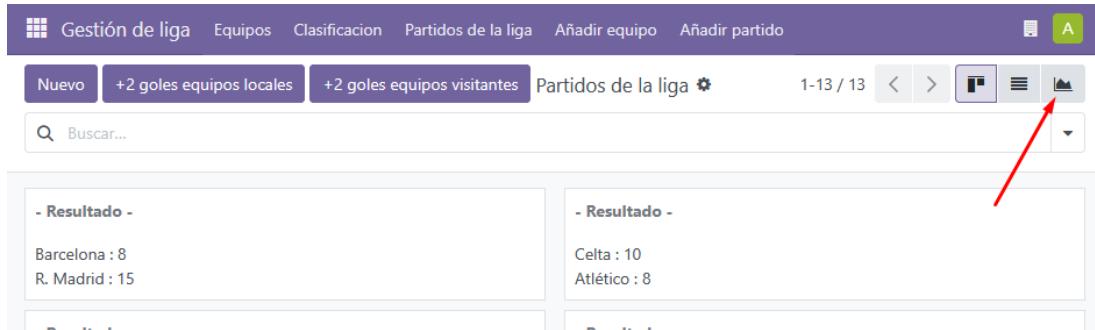
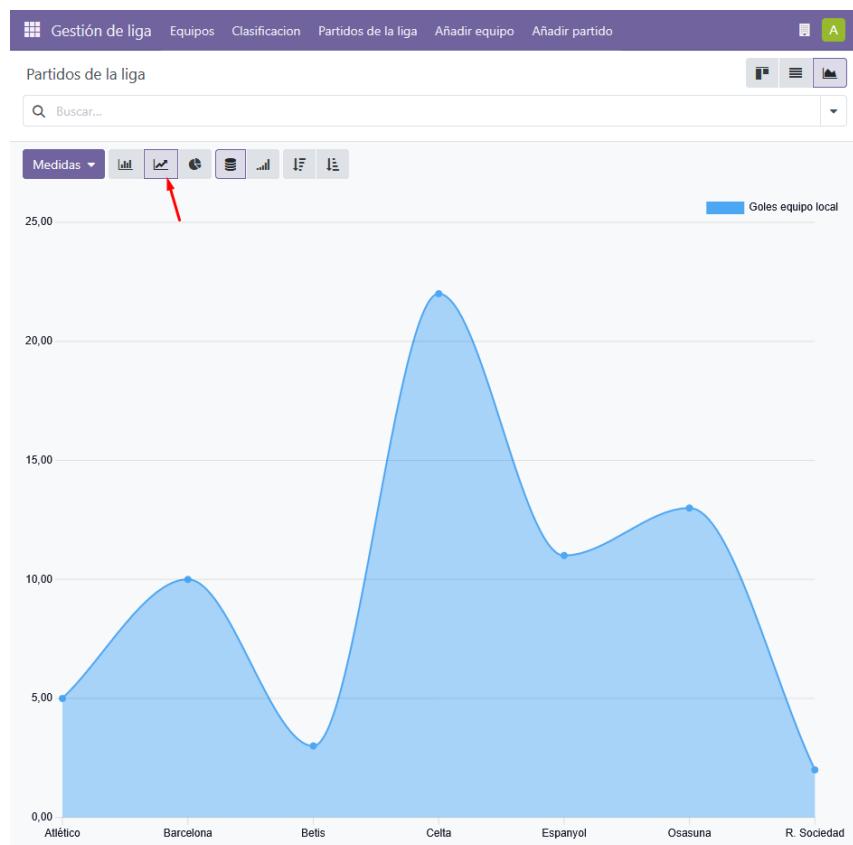
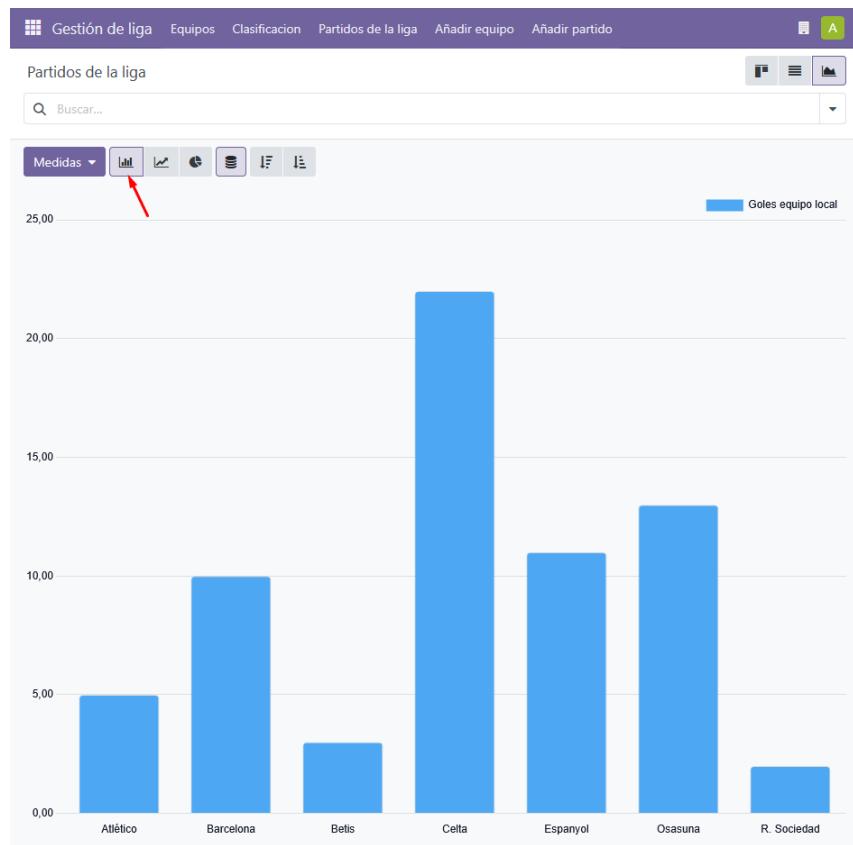
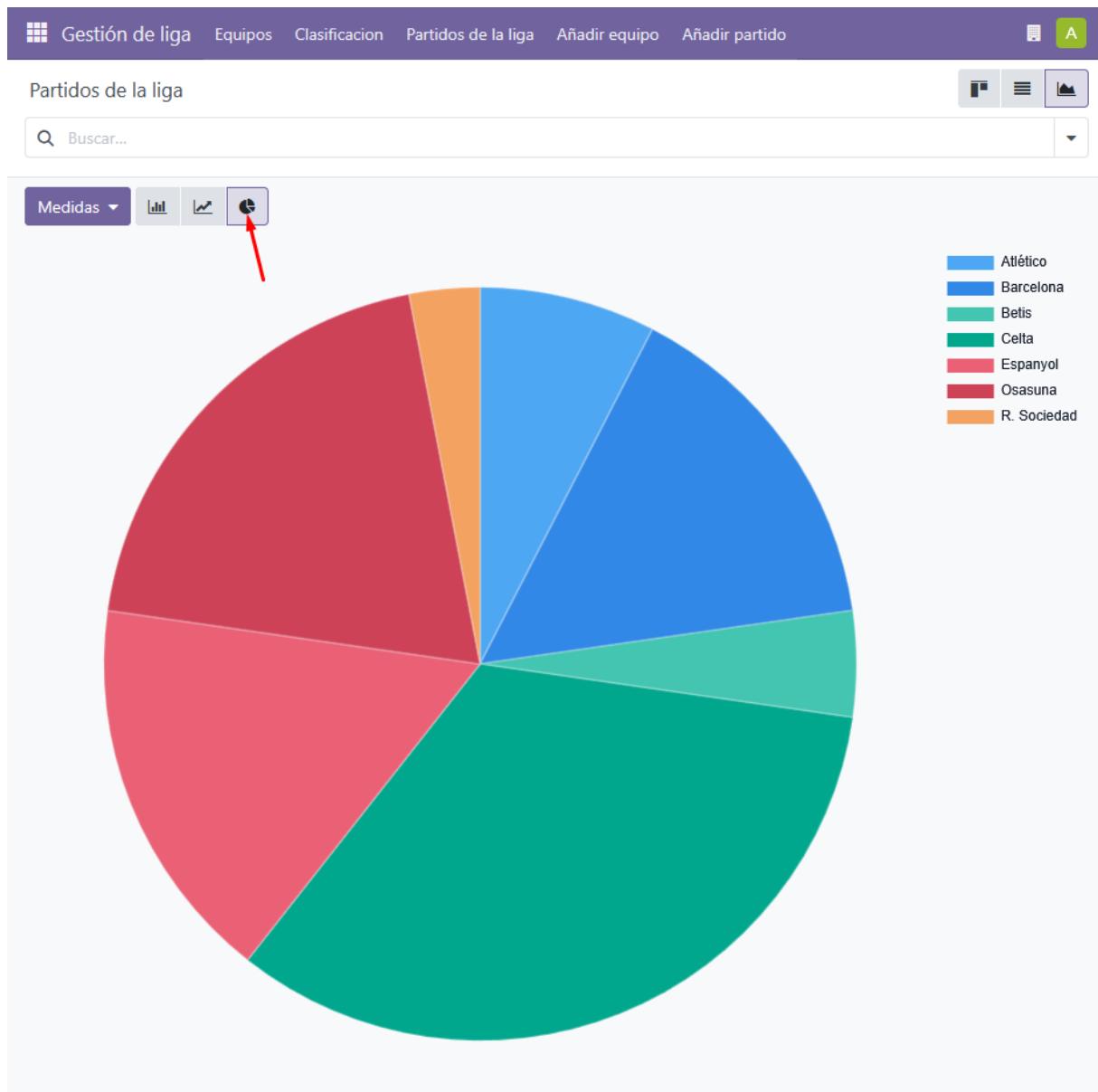


Figura 33: Aparece la nueva vista Graph, clicamos en ella





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 órdenes 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*:

```
addons > EJ09-GenerarBarcode > controllers > 🐸 generarbarcode.py > 📁 GenerarBarcode
You, last week | 1 author (You)
1 # -*- coding: utf-8 -*-
2 # Importamos clases necesarias de Odoo para definir controladores HTTP
3 from odoo import http    Import "odoo" could not be resolved
4 from odoo.http import request    Import "odoo.http" could not be resolved
5
6 # Importamos bibliotecas externas necesarias para generar imágenes en memoria
7 import base64                # Para codificar la imagen en base64 (para mostrarla en HTML)
8 from io import BytesIO          # Para trabajar con flujos de memoria
9
10 # Importamos la librería de generación de códigos de barras
11 # NOTA: Esta librería debe instalarse por pip: python-barcode y python-barcode[images]
12 import barcode    Import "barcode" could not be resolved
13 from barcode.writer import ImageWriter    Import "barcode.writer" could not be resolved
14
```

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 > ...
17 class GenerarBarcode(http.Controller):
18     ...
19     ...
20     Este método permite generar un código de barras EAN-13 desde una URL,
21     sin necesidad de autenticación. Ideal para integración con otras apps o para pruebas.
22
23     Ejemplo de uso:
24     http://localhost:8069/generador/123456789012
25
26     Mostrará una imagen HTML con el código de barras generado.
27     ...
28
29     # Ruta expuesta públicamente (auth='public'), sin restricciones CORS (cors='*')
30     @http.route('/generador/<codigo>', auth='public', cors='*', type='http')
31     def crearBarcode(self, codigo, **kw):
32         # Obtenemos la clase del tipo de código de barras que vamos a usar (EAN13)
33         EAN = barcode.get_barcode_class('ean13')
34
35         # Aseguramos que el código tenga 12 dígitos rellenando con ceros a la izquierda
36         # NOTA: El EAN-13 requiere 12 dígitos + 1 dígito de control (calculado automáticamente)
37         codigo_ean = str(codigo).zfill(12)
38
39         # Creamos el objeto del código de barras, pasándole el escritor de imágenes
40         ean = EAN(codigo_ean, writer=ImageWriter())
41
42         # Creamos un flujo en memoria para guardar la imagen generada
43         fp = BytesIO()
44
45         # Escribimos la imagen del código de barras en el flujo
46         ean.write(fp)
47
48         # Codificamos el flujo de bytes en base64 para poder usarlo en HTML
49         img_str = base64.b64encode(fp.getvalue()).decode("utf-8")
50
51         # Devolvemos una pequeña vista HTML que muestra la imagen generada
52         return '<div></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 > ...
1  # -*- coding: utf-8 -*-
2  from odoo import http # type: ignore
3  from odoo.http import request # type: ignore
4
5  import base64
6  from io import BytesIO
7
8  from PIL import Image # type: ignore LIBRERÍA PILLOW YA INCLUÍDA EN LA IMAGEN DE ODOO
9  import random          PARA DOCKER.
10
11 class ImagenAleatoria(http.Controller):
12
13     @http.route('/imagen/<int:ancho>/<int:alto>', auth='public', type='http')
14     def crearImagen(self, ancho, alto):
15         # CREAMOS IMAGEN VACÍA EN MODO RGB
16         imagen = Image.new('RGB', (ancho, alto))
17         pixeles = imagen.load()
18
19         # CREAMOS BUCLES ANIDADOS PARA RECORRER CADA PÍXEL DE LA IMAGEN,
20         # CREANDO UN COLOR DE FORMA ALEATORIA EN CADA ITERACIÓN
21         for x in range(ancho):
22             for y in range(alto):
23                 r = random.randint(0, 255)
24                 g = random.randint(0, 255)
25                 b = random.randint(0, 255)
26                 pixeles[x, y] = (r, g, b)
27
28         # UTILIZAMOS BytesIO COMO BUFFER EN RAM
29         buffer = BytesIO()
30
31         # GUARDAMOS LA IMAGEN Pillow EN EL BUFFER COMO PNG
32         imagen.save(buffer, format="PNG")
33
34         # TRADUCIMOS EL CONTENIDO DEL BUFFER A base64
35         imagenB64 = base64.b64encode(buffer.getvalue()).decode("utf-8")
36
37         # DEVOLVEMOS EL HTML MOSTRANDO LA IMAGEN RECIÉN CREADA
38         return '''<div><h2>PÍXELES ALEATORIOS(%dx%d)</h2>
39             </div>''' % (ancho, alto, imagenB64)

```

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.

