

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

Sistemas de Gestión Empresarial

Cristian Fernández

28 de enero de 2026

# Índice

<b>1. Introducción</b>	<b>2</b>
<b>2. Actividad 01 – Modificación <i>EJ07-LigaFutbol</i></b>	<b>3</b>
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 . . . . .	12
2.4. Informe PDF por cada partido . . . . .	13
2.5. Wizard para crear nuevos partidos . . . . .	14
2.6. Vista Graph . . . . .	15
<b>3. Actividad 02 – Pruebas <i>EJ08-API-REST_Socio</i></b>	<b>16</b>
<b>4. Actividad 03 – Bot Telegram API REST</b>	<b>17</b>
<b>5. Actividad 04 – Generación imágenes aleatorias Web Controller</b>	<b>18</b>

# 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		
<a href="#">Nuevo</a>		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			
<a href="#">Nuevo</a>		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**

01

### **2.3. Web Controller para eliminar empates**

01

## **2.4. Informe PDF por cada partido**

01

## **2.5. Wizard para crear nuevos partidos**

01

## 2.6. Vista Graph

01

### 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**

03

## **5. Actividad 04 – Generación imágenes aleatorias Web Controller**

04