



Documentación Técnica: Módulos Odoo y Web Controllers

Autor: Pedro José Meixús Belsol

2 de febrero de 2026

Índice

1. Modificar módulo liga fútbol	2
1.1. Reglas de puntuación especiales	2
1.2. Botones para alterar los goles de los partidos	4
1.3. Web Controller para eliminar empates	6
1.4. Informe PDF por partido	8
1.5. Wizard para crear nuevos partidos	9
1.6. Vista Graph para goles de equipos locales	12
2. Bot de Telegram conectado a la API REST	14
2.1. Creación del bot	14
2.2. Creación y configuración de entorno para el bot	15
2.3. Desarrollo	16
2.4. Arranque y funcionamiento	18
3. Generación de imágenes aleatorias con Web Controllers	21
3.1. Instalaciones necesarias	21
3.2. Prueba Web Controller de código de barras	23
3.3. Creación de Web Controller para imágenes aleatorias	24
4. Problemas encontrados	26
5. Conclusión	27

1. Modificar módulo liga fútbol

Tomando como base el módulo EJ07LigaFutbol que se nos ha proporcionado, vamos a realizar diferentes modificaciones para ampliar su funcionalidad.

1.1. Reglas de puntuación especiales

Empezaremos por modificar las reglas de puntuación. Haremos que los partidos con 4 o más goles de diferencia otorguen 4 puntos al equipo ganador y -1 al perdedor. Para eso crearemos una función en el modelo de 'liga_partido.py' para que lo calcule, simplemente comparando la diferencia de goles de goles.



```
1  def actualizoRegistrosEquipo(self):
2      for recordEquipo in self.env['liga.equipo'].search([]):
3          recordEquipo.victorias = 0
4          recordEquipo.empates = 0
5          recordEquipo.derrotas = 0
6          recordEquipo.goles_a_favor = 0
7          recordEquipo.goles_en_contra = 0
8          recordEquipo.puntos = 0
9
10     for partido in self.env['liga.partido'].search([]):
11         # Local
12         local = partido.equipo_casa
13         visitante = partido.equipo_fuera
14         if not local or not visitante:
15             continue
16
17         # Goles
18         goles_local = partido.goles_casa
19         goles_vis = partido.goles_fuera
20         diferencia = abs(goles_local - goles_vis)
21
22         # Resultado local
23         if goles_local > goles_vis:
24             if diferencia >= 4:
25                 local.puntos += 4
26             else:
27                 local.puntos += 3
28                 local.victorias += 1
29                 visitante.derrotas += 1
30                 if diferencia >= 4:
31                     visitante.puntos -= 1
32             elif goles_local < goles_vis:
33                 if diferencia >= 4:
34                     visitante.puntos += 4
35                     local.puntos -= 1
36                 else:
37                     visitante.puntos += 3
38                     visitante.victorias += 1
39                     local.derrotas += 1
40             else:
41                 local.empates += 1
42                 visitante.empates += 1
43                 local.puntos += 1
44                 visitante.puntos += 1
45
46         # Sumar goles a favor y en contra
47         local.goles_a_favor += goles_local
48         local.goles_en_contra += goles_vis
49         visitante.goles_a_favor += goles_vis
50         visitante.goles_en_contra += goles_local
```

Podemos comprobarlo creando un partido con 4 goles de diferencia y viendo la tabla de clasificación. Ahora mismo esta es la tabla de clasificación:

Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A F...	Goles En ...	Victorias	Empates	Derrotas
	Barcelona	6	3	6	6	2	0	1
	Real Madrid	2	3	6	6	1	0	2

Vamos a añadir un partido en el que el equipo Real Madrid gana 6-0.

+2 goles equipos locales	+2 goles equipos visitantes	Imprimir PDF
Equipo local Real Madrid	Equipo visitante Barcelona	Goles Fuera 0

Y ahora la puntuación debería haber aumentando en 4 puntos para el Real Madrid y disminuido en 1 para el Barcelona.

Escudo equipo	Nombre equipo	Puntos	Jugados	Goles A F...	Goles En ...	Victorias	Empates	Derrotas
	Real Madrid	6	4	12	6	2	0	2
	Barcelona	5	4	6	12	2	0	2

1.2. Botones para alterar los goles de los partidos

Vamos a añadir dos botones en la vista de formulario de los partidos para aumentar o disminuir los goles de todos los equipos locales o visitantes. Tendremos que añadir dos funciones en el modelo de 'liga_partido.py' para que lo hagan.

```
● ● ●
1 def sumar_goles_locales(self):
2     for partido in self.env['liga.partido'].search([]):
3         partido.goles_casa += 2
4     self.actualizoRegistrosEquipo()
5
6 def sumar_goles_visitantes(self):
7     for partido in self.env['liga.partido'].search([]):
8         partido.goles_fuera += 2
9     self.actualizoRegistrosEquipo()
```

Vamos a probar si funciona, primero veremos los resultados de los partidos.

The screenshot shows the Odoo interface for managing league matches. The top navigation bar includes 'Gestión de liga', 'Equipos', 'Clasificación', 'Partidos de la liga', 'Añadir equipo', 'Crear Nuevo Partido', and 'My Company'. Below the navigation is a search bar with placeholder 'Buscar...' and a page indicator '1-4 / 4'. The main area displays four match results in boxes:

- Resultado -
Barcelona : 1
Real Madrid : 0
- Resultado -
Real Madrid : 6
Barcelona : 0
- Resultado -
Barcelona : 5
Real Madrid : 0
- Resultado -
Real Madrid : 6
Barcelona : 0

Ahora accedo a uno de ellos y pulso una vez cada botón.

The screenshot shows the Odoo interface for viewing a specific match. The top navigation bar includes 'Gestión de liga', 'Equipos', 'Clasificación', 'Partidos de la liga', 'Añadir equipo', 'Crear Nuevo Partido', and 'My Company'. Below the navigation is a search bar with placeholder 'Partidos de la liga' and a page indicator '1 / 4'. The main area shows the match details for 'liga.partido,24':

Equipo local	Barcelona	Equipo visitante	Real Madrid
Goles Casa	3	Goles Fuera	2

Below the table are three buttons: '+2 goles equipos locales', '+2 goles equipos visitantes', and 'Imprimir PDF'.

Y ahora volvemos a ver los resultados de los partidos para comprobar que se han modificado correctamente.

The screenshot shows a list of four recent football match results:

- Barcelona : 3
Real Madrid : 2
- Real Madrid : 8
Barcelona : 2
- Real Madrid : 8
Barcelona : 2
- Barcelona : 7
Real Madrid : 2

Para probar que la clasificación se recalcula, voy a sumarle dos veces a los visitantes y al ver la clasificación podemos ver que ha cambiado.

The screenshot shows the updated classification table:

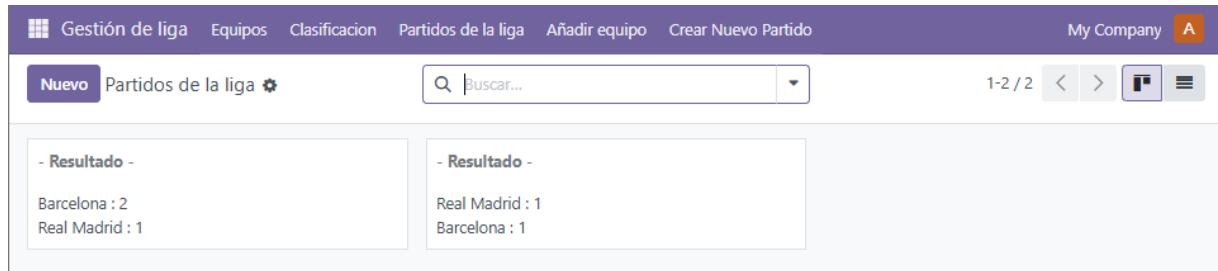
Nombre equipo	Puntos	Jugados	Goles A F...	Goles En ...	Victorias	Empates	Derrotas
Real Madrid	9	4	28	22	3	0	1
Barcelona	3	4	22	28	1	0	3

1.3. Web Controller para eliminar empates

Vamos a añadir un Web Controller que nos permita eliminar todos los partidos que hayan terminado en empate. Primero añadiremos un '@http.route' con una función que buscara los partidos terminados en empate y los eliminará.

```
1  @http.route('/eliminarempates', type='http', auth='none')
2  def eliminar_empates(self):
3      # Buscamos todos los partidos que hayan terminado en empate
4      partidos_empate = request.env['liga.partido'].sudo().search([
5          ('goles_casa', '=', 'goles_fuera')
6      ])
7
8      # Contamos cuántos hay antes de eliminar
9      total_eliminados = len(partidos_empate)
10
11     # Eliminamos los partidos encontrados
12     partidos_empate.unlink()
13
14     # Devolvemos el número de partidos eliminados como texto
15     return f"Se han eliminado {total_eliminados} partidos en empate."
```

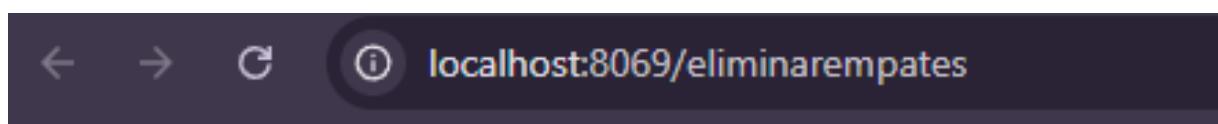
Ahora crearemos un partido que termine en empate, si no tenemos alguno lo crearemos como en mi caso.



The screenshot shows a web-based management system for a league. The top navigation bar includes links for 'Gestión de liga', 'Equipos', 'Clasificación', 'Partidos de la liga', 'Añadir equipo', and 'Crear Nuevo Partido'. On the right, there's a 'My Company' section with a user icon. Below the navigation is a search bar with placeholder 'buscar...' and a dropdown arrow. To the right of the search bar are pagination controls showing '1-2 / 2' and icons for navigating between pages. The main content area displays two match results in separate boxes:

- Resultado -
Barcelona : 2
Real Madrid : 1
- Resultado -
Real Madrid : 1
Barcelona : 1

Teniendo uno por lo menos iremos a la URL del Web Controller que hemos creado ('<http://localhost:8069/eliminarempates>') y eliminará automáticamente los partidos en empate.



Se han eliminado 1 partidos en empate.

Y si volvemos a la vista de partidos podemos ver que el partido en empate ha sido eliminado.



Nuevo

Partidos de la liga

Buscar...

- Resultado -

Barcelona : 2

Real Madrid : 1

1.4. Informe PDF por partido

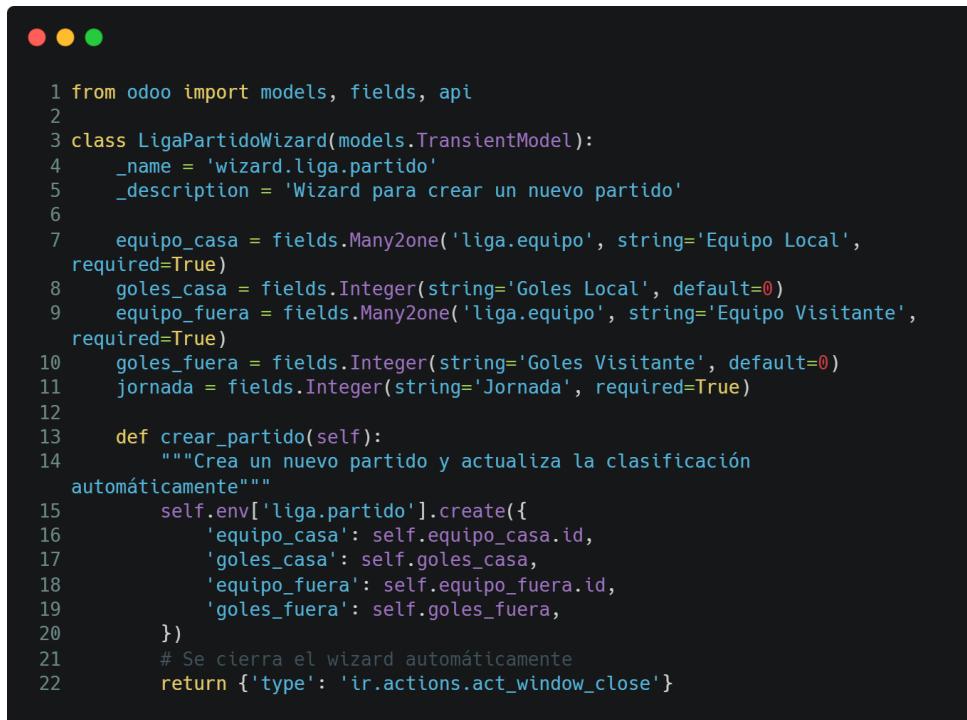
No he conseguido hacer funcionar esta parte.

*Revisar en la sección de problemas encontrados al final del documento.

1.5. Wizard para crear nuevos partidos

Vamos a crear un Wizard que nos permita crear nuevos partidos de una forma más sencilla. Para eso crearemos dos archivos nuevos, uno para el modelo del Wizard y otro para la vista dentro de la carpeta 'wizard'.

En el modelo crearemos los campos necesarios (y sus relaciones) para crear un partido y una función que lo cree.



```
 1 from odoo import models, fields, api
 2
 3 class LigaPartidoWizard(models.TransientModel):
 4     _name = 'wizard.liga.partido'
 5     _description = 'Wizard para crear un nuevo partido'
 6
 7     equipo_casa = fields.Many2one('liga.equipo', string='Equipo Local',
 8                                   required=True)
 9     goles_casa = fields.Integer(string='Goles Local', default=0)
10     equipo_fuera = fields.Many2one('liga.equipo', string='Equipo Visitante',
11                                    required=True)
12     goles_fuera = fields.Integer(string='Goles Visitante', default=0)
13     jornada = fields.Integer(string='Jornada', required=True)
14
15     def crear_partido(self):
16         """Crea un nuevo partido y actualiza la clasificación
17         automáticamente"""
18         self.env['liga.partido'].create({
19             'equipo_casa': self.equipo_casa.id,
20             'goles_casa': self.goles_casa,
21             'equipo_fuera': self.equipo_fuera.id,
22             'goles_fuera': self.goles_fuera,
23         })
24         # Se cierra el wizard automáticamente
25     return {'type': 'ir.actions.act_window_close'}
```

Y en la vista crearemos el formulario para el Wizard.

```

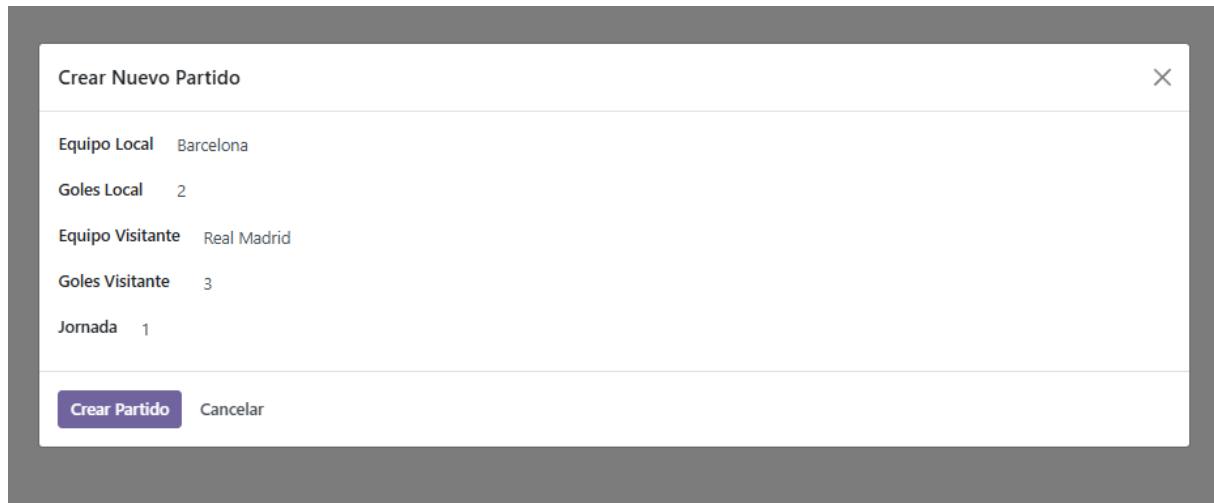
1 <?xml version="1.0" encoding="utf-8"?>
2 <ooodo>
3
4     <!-- Vista del wizard -->
5     <record id='liga_partido_wizard_form' model='ir.ui.view'>
6         <field name='name'>Wizard para introducir un partido</field>
7         <field name='model'>wizard.liga.partido</field>
8         <field name='arch' type='xml'>
9             <form string="Introducir un nuevo partido">
10            <sheet>
11                <group>
12                    <field name='equipo_casa'/>
13                    <field name='goles_casa'/>
14                </group>
15                <group>
16                    <field name='equipo_fuera'/>
17                    <field name='goles_fuera'/>
18                </group>
19                <group>
20                    <field name='jornada'/>
21                </group>
22            </sheet>
23            <footer>
24                <button string='Crear Partido' name='crear_partido'
25 class='btn-primary' type='object'/>
26                <button string='Cancelar' class='btn-default'
27 special='cancel'/>
28            </footer>
29        </form>
30    </field>
31 </record>
32
33     <!-- Acción del wizard -->
34     <record id="action_wizard_liga_partido" model="ir.actions.act_window">
35         <field name="name">Crear Nuevo Partido</field>
36         <field name="res_model">wizard.liga.partido</field>
37         <field name="view_mode">form</field>
38         <field name="target">new</field>
39     </record>
40
41     <!-- Menúitem -->
42     <menuitem id="menu_wizard_liga_partido"
43         parent="liga_base_menu"
44         action="action_wizard_liga_partido"
45         sequence="20"
46         name="Crear Nuevo Partido"/>
47 </ooodo>

```

Con esto ya hecho, añadiremos la ruta del xml al '`__manifest__.py`' para que Odoo los reconozca. Y ahora veremos un nuevo botón en el menú superior para abrir el Wizard.

The screenshot shows the Odoo web interface. At the top, there is a purple navigation bar with several menu items: 'Gestión de liga', 'Equipos', 'Clasificación', 'Partidos de la liga', 'Añadir equipo', and 'Crear Nuevo Partido'. The 'Crear Nuevo Partido' button is circled in red. Below the navigation bar, there is a search bar with the placeholder 'Buscar...' and a dropdown arrow. On the left, there is a blue button labeled 'Nuevo' and another button labeled 'Partidos de la liga' with a gear icon. The main content area has a white background and contains a box with the heading '- Resultado -'. Inside this box, the text 'Barcelona : 2' and 'Real Madrid : 1' is displayed.

Al pulsarlo se abrirá el Wizard que hemos creado y podremos añadir un nuevo partido.



1.6. Vista Graph para goles de equipos locales

Vamos a añadir una vista Graph que nos muestre los goles totales de los equipos locales.

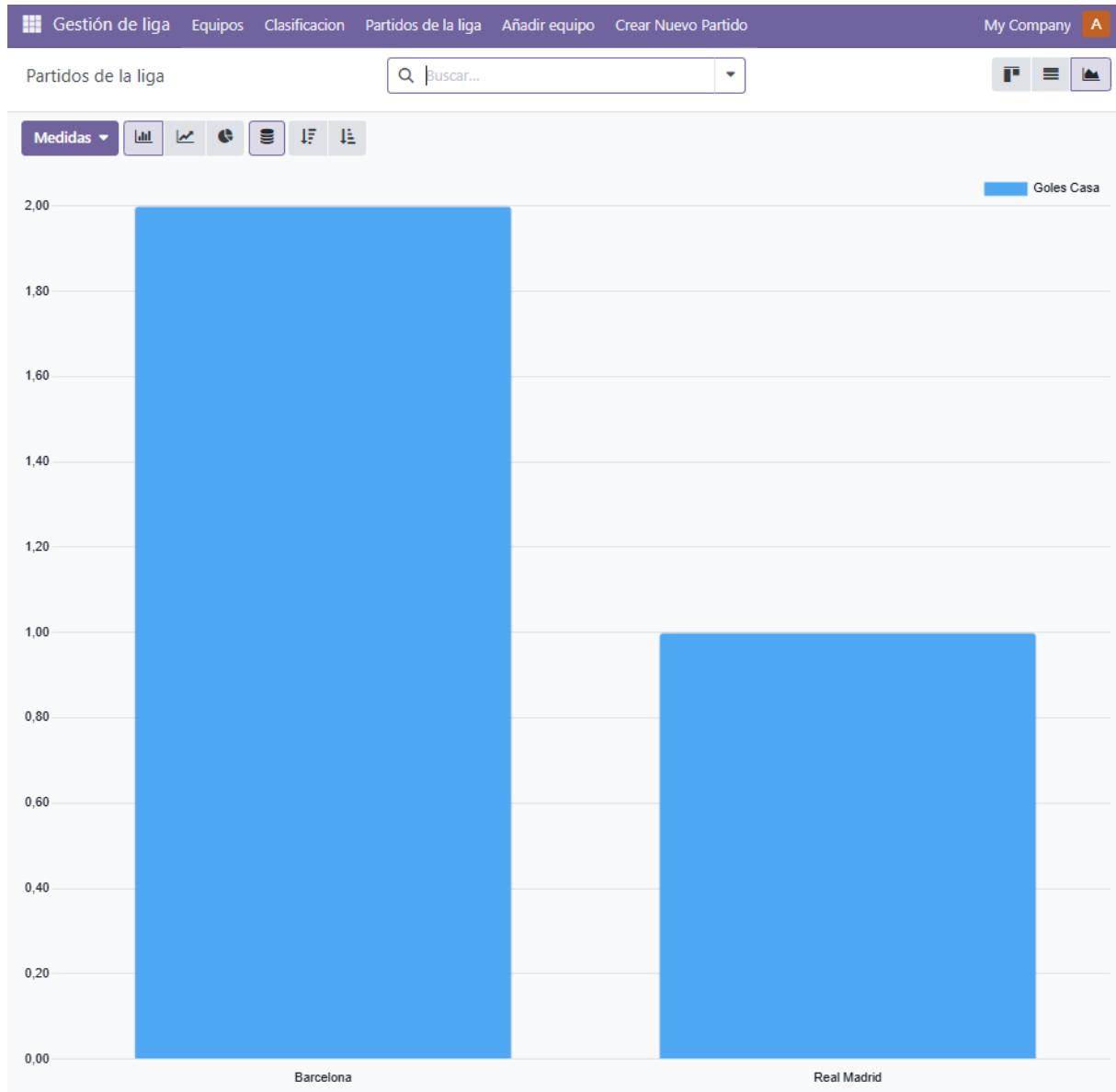
Para esto he añadido la vista Graph en el archivo XML de 'liga_partido.xml'.

```
● ● ●
1 <record id="liga_partido_view_graph" model="ir.ui.view">
2   <field name="name">Goles locales por equipo</field>
3   <field name="model">liga.partido</field>
4   <field name="type">graph</field>
5   <field name="arch" type="xml">
6     <graph string="Goles locales por equipo">
7       <field name="equipo_casa" type="row"/>
8       <field name="goles_casa" type="measure"/>
9     </graph>
10   </field>
11 </record>
```

Y también he añadido la acción para que se una de las vistas posibles.

```
● ● ●
1 <field name="view_mode">kanban,list,form,graph</field>
```

Ahora si vamos a la vista de partidos podemos seleccionar la vista Graph y ver los goles totales de los equipos locales.

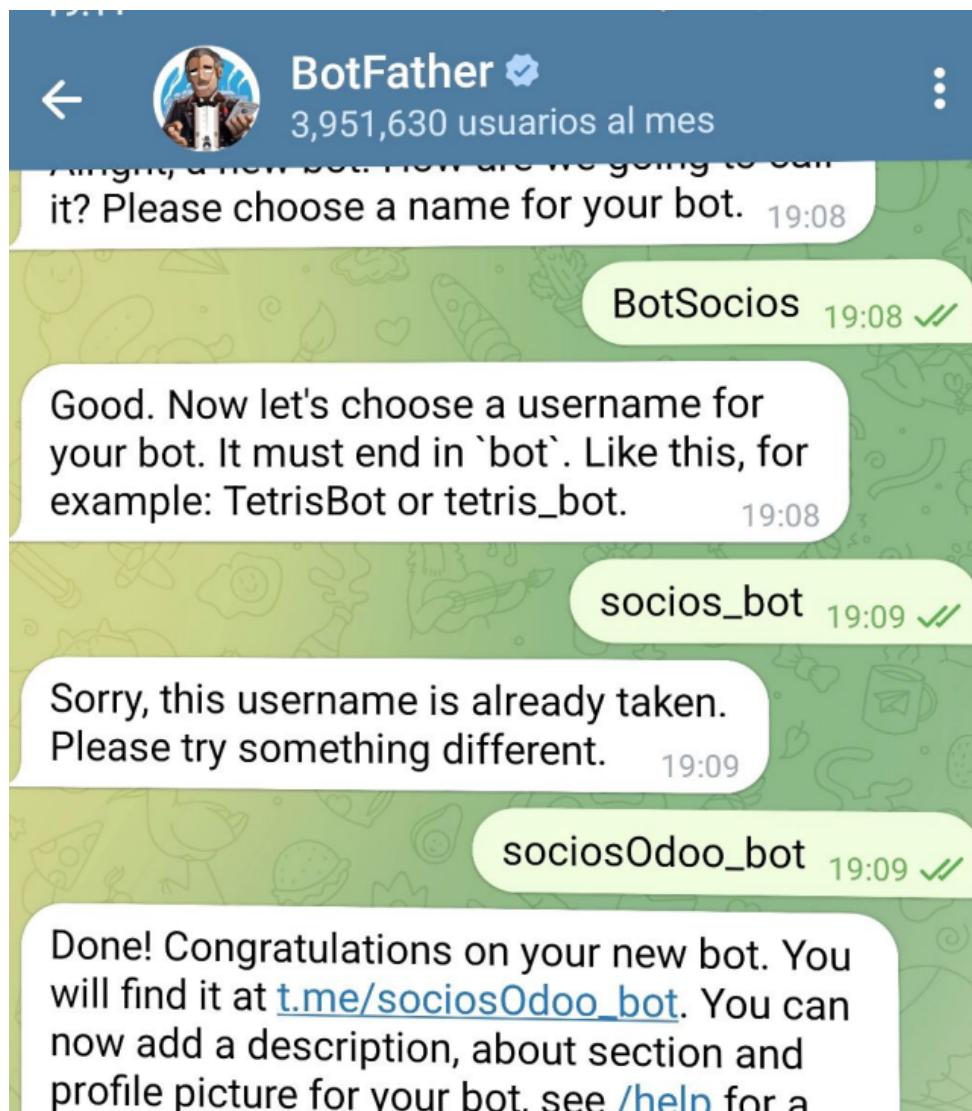


2. Bot de Telegram conectado a la API REST

En esta actividad vamos a crear un bot de Telegram que se conecte a la API REST que creamos en la actividad anterior. Usaremos la librería 'python-telegram-bot' para crear el bot y la librería 'requests' para hacer las peticiones HTTP a la API REST. Dependiendo de la orden que se envíe se realizarán diferentes peticiones (GET, POST, PUT o DELETE) a la API, Odoo procesara esa solicitud y el bot devolverá el resultado al usuario.

2.1. Creación del bot

Primero de nada, debemos crear el propio bot en telegram. Para eso, abrimos Telegram y buscamos el bot 'BotFather'. Le enviamos el comando '/newbot' y seguimos las instrucciones para crear nuestro bot.



Nos dará un token que guardaremos y usaremos para conectar nuestro bot con la API de Telegram.

2.2. Creación y configuración de entorno para el bot

Ahora crearemos un entorno virtual para nuestro bot con el siguiente comando.

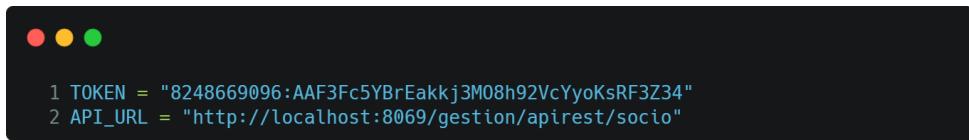
```
PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\bot> python -m venv venv_bot
```

Accedemos al entorno virtual e instalaremos las librerías necesarias con pip(el python-telegram-bot y requests).

```
PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\bot> .\venv_bot\Scripts\activate  
(venv_bot) PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\bot> pip install python-telegram-bot requests
```

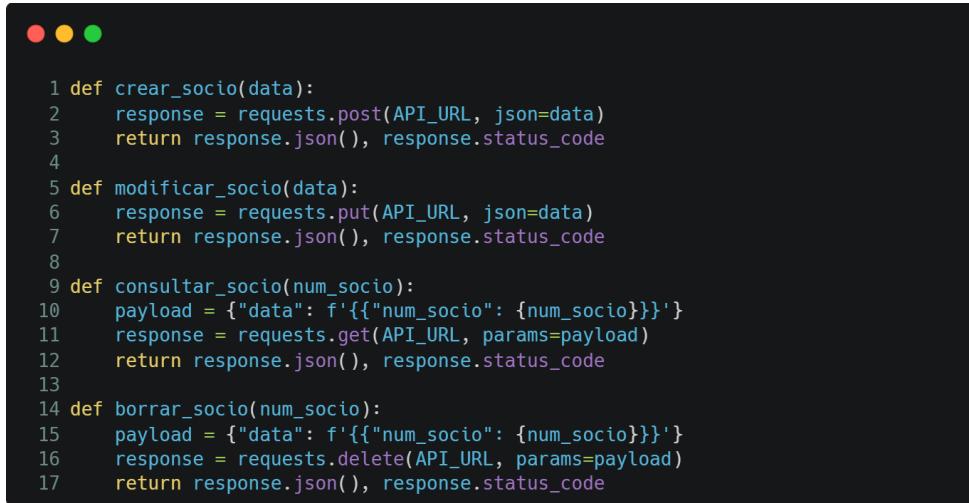
2.3. Desarrollo

Posteriormente crearemos un archivo 'bot_socios.py' y añadiremos el código necesario para crear el bot. Aquí es donde deberemos usar por un lado el token que nos dio BotFather y por otro la URL de nuestra API REST.



```
1 TOKEN = "8248669096:AAF3Fc5YBrEakkj3M08h92VcYyoKsRF3Z34"
2 API_URL = "http://localhost:8069/gestion/apirest/socio"
```

También deberemos indicar las funciones para interactuar con la API REST (crear, modificar, consultar y borrar socios).



```
1 def crear_socio(data):
2     response = requests.post(API_URL, json=data)
3     return response.json(), response.status_code
4
5 def modificar_socio(data):
6     response = requests.put(API_URL, json=data)
7     return response.json(), response.status_code
8
9 def consultar_socio(num_socio):
10    payload = {"data": f'{{"num_socio": {num_socio}}}'}
11    response = requests.get(API_URL, params=payload)
12    return response.json(), response.status_code
13
14 def borrar_socio(num_socio):
15    payload = {"data": f'{{"num_socio": {num_socio}}}'}
16    response = requests.delete(API_URL, params=payload)
17    return response.json(), response.status_code
```

Las funciones anteriores se usarán en la función principal del bot (handler), que se encargará de manejar los mensajes recibidos y llamar a las funciones correspondientes según el comando recibido.

```

1  async def manejar_mensaje(update: Update, context: ContextTypes.DEFAULT_TYPE):
2      texto = update.message.text.strip()
3      chat_id = update.message.chat_id
4
5      if texto.lower().startswith("crear"):
6          try:
7              params = texto[6:]
8              data = {k.strip(): int(v) if k.strip() == "num_socio" else v.strip()
9                      for k, v in (x.split("=") for x in params.split(","))}
10             res, status = crear_socio(data)
11             await context.bot.send_message(chat_id, f"Respuesta Crear
12 ({status}): {res}")
12         except Exception as e:
13             await context.bot.send_message(chat_id, f"Error al crear: {e}")
14         return
15
16     elif texto.lower().startswith("modificar"):
17         try:
18             params = texto[10:]
19             data = {k.strip(): int(v) if k.strip() == "num_socio" else v.strip()
20                     for k, v in (x.split("=") for x in params.split(","))}
21             res, status = modificar_socio(data)
22             await context.bot.send_message(chat_id, f"Respuesta Modificar
23 ({status}): {res}")
23         except Exception as e:
24             await context.bot.send_message(chat_id, f"Error al modificar: {e}")
25         return
26
27     elif texto.lower().startswith("consultar"):
28         try:
29             params = texto[9:]
30             num_socio = int(params.split("=")[1])
31             res, status = consultar_socio(num_socio)
32             await context.bot.send_message(chat_id, f"Respuesta Consultar
33 ({status}): {res}")
33         except Exception as e:
34             await context.bot.send_message(chat_id, f"Error al consultar: {e}")
35         return
36
37     elif texto.lower().startswith("borrar"):
38         try:
39             params = texto[7:]
40             num_socio = int(params.split("=")[1])
41             res, status = borrar_socio(num_socio)
42             await context.bot.send_message(chat_id, f"Respuesta Borrar
43 ({status}): {res}")
43         except Exception as e:
44             await context.bot.send_message(chat_id, f"Error al borrar: {e}")
45         return
46
47     else:
48         await context.bot.send_message(chat_id, "Orden no soportada")

```

2.4. Arranque y funcionamiento

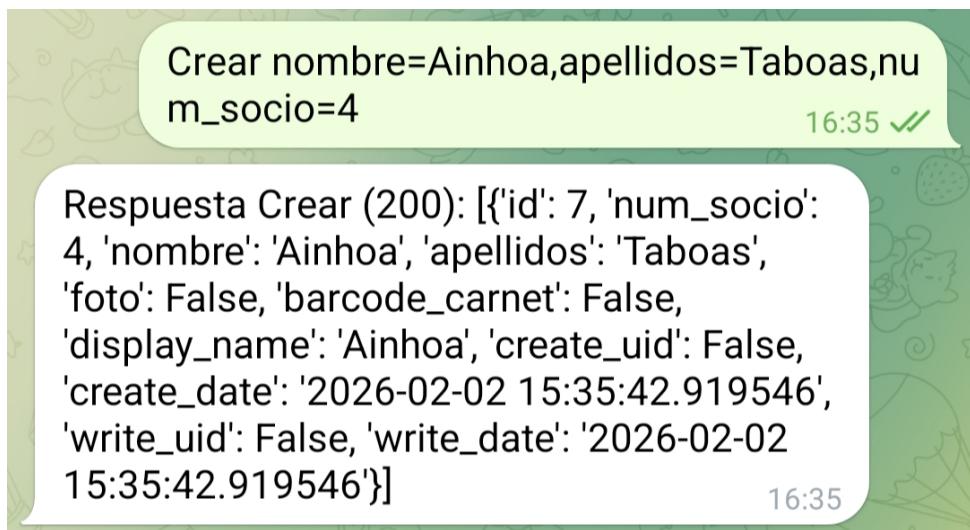
Con todo esto hecho podemos iniciar nuestro bot con el siguiente comando.

```
● PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\bot> .\venv_bot\Scripts\activate
❖ (venv_bot) PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\bot>
```

Desde dentro usaremos el comando 'python' junto con el nombre del archivo para iniciarla y si todo esta correcto iniciará sin problema.

```
❖ (venv_bot) PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\bot> python .\bot_socios.py
Bot de socios iniciado...
```

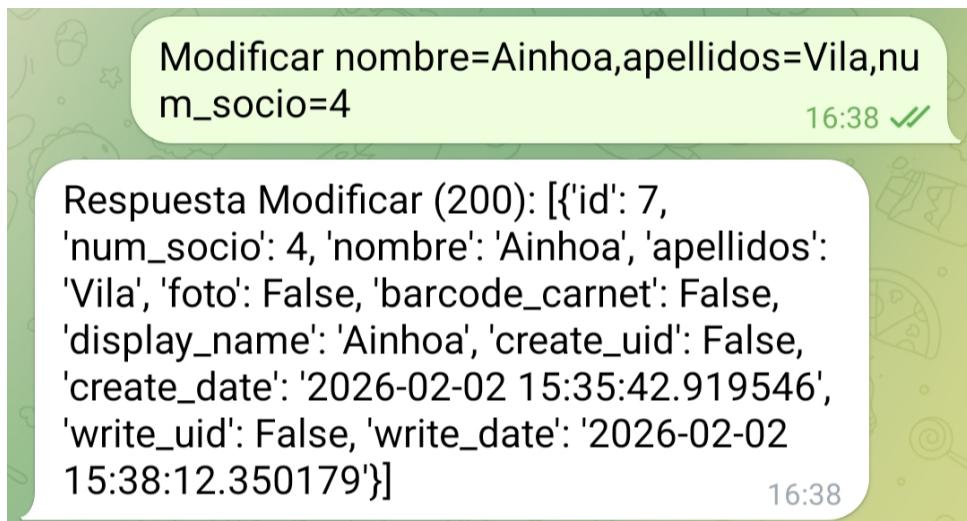
Ahora nos quedará probar los comandos uno por uno desde Telegram y ver si los cambios se reflejan en Odoo. Empecemos creando un socio.



Y comprobamos en Odoo que se ha creado correctamente.

Gestión de socios - Socios		My Company A	
Nuevo	Listado de socios	Buscar...	1-4 / 4 < > [grid] [list]
<input type="checkbox"/>	Número ... Nombre	Apellidos	
<input type="checkbox"/>	4 Ainhoa	Taboas	
<input type="checkbox"/>	2 Juan	Pérez	
<input type="checkbox"/>	3 Pablo	Millor	
<input type="checkbox"/>	1 Pepe	Barreiro	

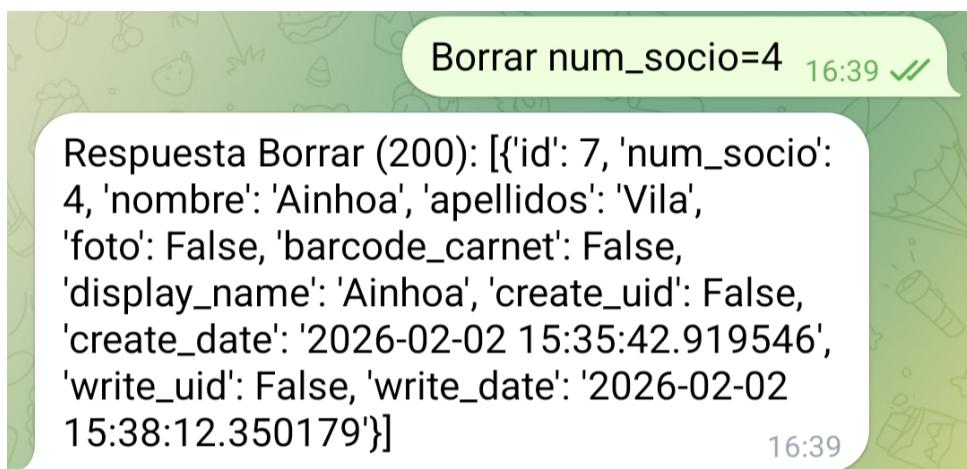
Probemos a modificarlo.



Revisamos en Odoo que el apellido ha sido modificado.

Gestión de socios			Socios	My Company A	
<input type="button" value="Nuevo"/>	Listado de socios	Buscar...		1-4 / 4	< >
<input type="checkbox"/>	Número ...	Nombre	Apellidos		
<input type="checkbox"/>	4	Ainhoa	Vila		
<input type="checkbox"/>	2	Juan	Pérez		
<input type="checkbox"/>	3	Pablo	Millor		
<input type="checkbox"/>	1	Pepe	Barreiro		

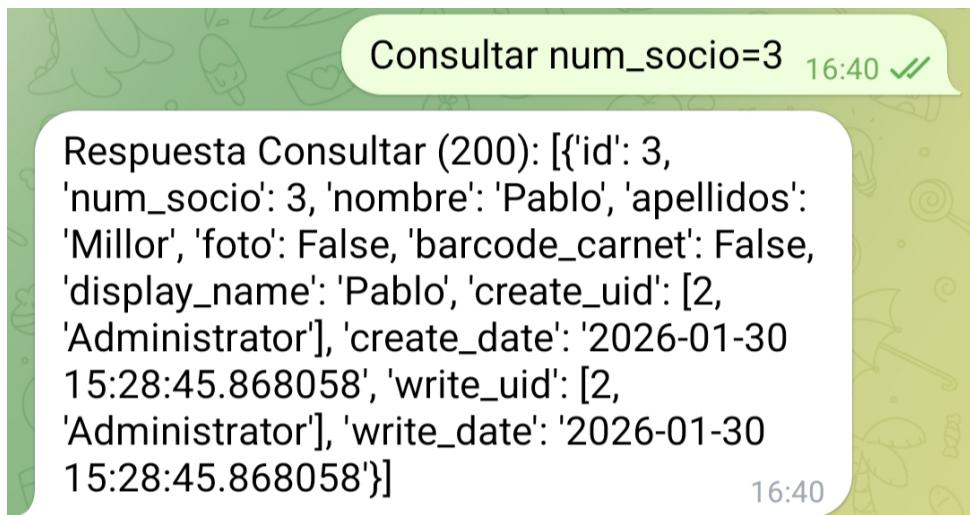
Ahora borraremos el socio que hemos creado.



Y comprobamos en Odoo que ha sido eliminado.

Gestión de socios			Socios	My Company A	
Nuevo		Listado de socios		Buscar...	1-3 / 3
	Número ...	Nombre	Apellidos		
<input type="checkbox"/>	2	Juan	Pérez		
<input type="checkbox"/>	3	Pablo	Millor		
<input type="checkbox"/>	1	Pepe	Barreiro		

Para terminar las operaciones probaremos a consultar un socio (en este caso el que tenga el numero de socio 3).



Por último, comprobamos que al mandar cualquier otro mensaje el bot responde 'Orden no soportada'.



Si todo ha funcionado correctamente, nuestro bot estará terminado.

3. Generación de imágenes aleatorias con Web Controllers

Para esta parte de la práctica, vamos a crear un Web Controller que genere imágenes aleatorias tomando como base el módulo 'EJ09-GenerarBarcode' proporcionado.

3.1. Instalaciones necesarias

Primero deberemos acceder a nuestro contenedor de Odoo, para buscarlo usaremos el comando 'docker ps'.

```
PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\Docker> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
3379579c7956 odoo:18.0 "/usr/bin/odoo -c /e..." 6 weeks ago Up 2 seconds 0.0.0.0:8069->8069/tcp, [::]:8069->8069/tcp odoo18
a0c46bad56df postgres:15 "docker-entrypoint.s..." 6 weeks ago Up 3 seconds 0.0.0.0:5432->5432/tcp, [::]:5432->5432/tcp odoo18_db
```

Esto nos mostrará los contenedores activos, buscaremos el ID del contenedor de Odoo y accederemos a él con el siguiente comando.

```
PS C:\Users\pmeibel\Desktop\laburo\SGE\ERP_Odoo\Docker> docker exec -u root -it 3379579c7956 bash
```

Una vez dentro usaremos el comando 'pip list' para ver las librerías instaladas.

```
root@3379579c7956:/# pip list
WARNING: Skipping /usr/lib/python3.12/dist-packages
          Package           Version
-----
asn1crypto            1.5.1
attrs                 23.2.0
Babel                2.10.3
beautifulsoup4        4.12.3
cached-property       1.5.2
cbor2                5.6.2
```

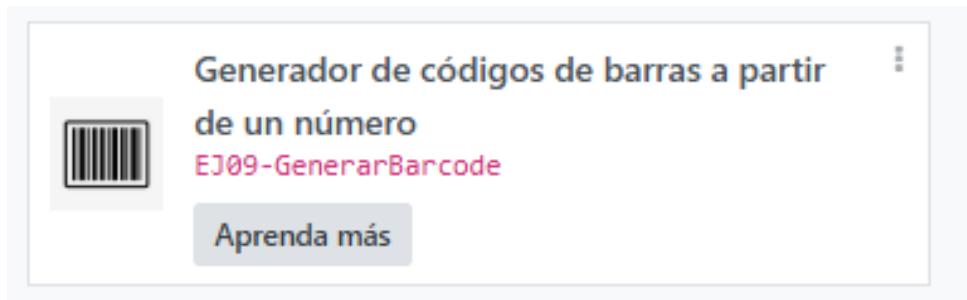
En cualquier caso, haremos un 'apt-get update' para actualizar lo necesario.

```
root@3379579c7956:/# apt-get update
Get:1 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:2 http://archive.ubuntu.com/ubuntu noble InRelease [256 kB]
Get:3 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [34.8 kB]
Get:4 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1776 kB]
Get:5 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [1194 kB]
Get:6 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [3008 kB]
Get:7 http://archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:8 http://archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:9 http://archive.ubuntu.com/ubuntu noble/restricted amd64 Packages [117 kB]
```

Ahora instalaremos las librerías que vamos a necesitar con los siguiente comandos.

```
root@3379579c7956:/# apt-get install -y python3-pip python3-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Hecho esto, reiniciamos el contenedor para que los cambios tengan efecto. Y activamos el módulo en Odoo.



3.2. Prueba Web Controller de código de barras

Ahora podemos probar el Web Controller del código de barras que viene por defecto accediendo a la URL '<http://localhost:8069/generador/123456789012>'.



3.3. Creación de Web Controller para imágenes aleatorias

Vamos ahora entonces a añadir nuestro propio Web Controller para generar imágenes aleatorias. Añadiremos un archivo en mi caso 'generarImagen.py' y añadiremos el siguiente código, que generará pixeles de colores aleatorios con el alto y ancho que indiquemos.

La imagen se genera en formato PNG y se devolverá como respuesta al acceder a la URL.

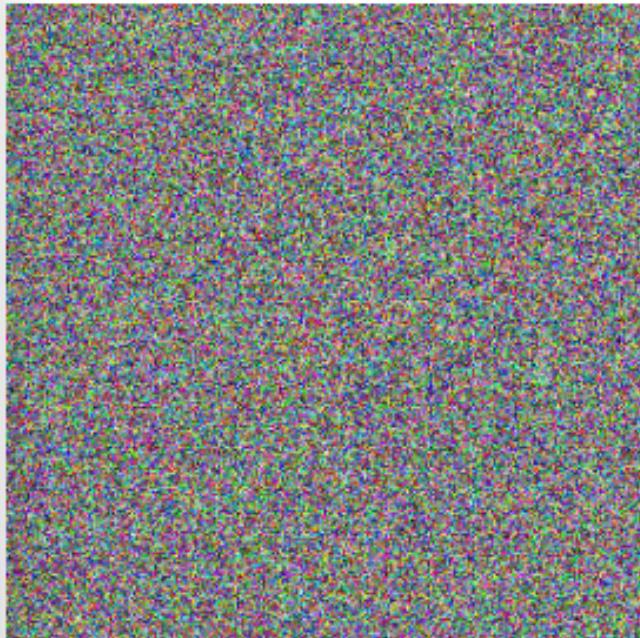
```
 1 # -*- coding: utf-8 -*-
 2 from odoo import http
 3 from odoo.http import request
 4 from PIL import Image
 5 from io import BytesIO
 6 import base64
 7 import random
 8
 9 class GenerarImagenAleatoria(http.Controller):
10
11     @http.route('/imagen/aleatoria', auth='public', cors='*', type='http')
12     def crearImagen(self, ancho=200, alto=200, formato='html', **kw):
13         # Convertimos parámetros a enteros
14         try:
15             ancho = int(ancho)
16             alto = int(alto)
17         except ValueError:
18             return "Parámetros inválidos. Usa enteros para ancho y alto."
19
20         # Creamos imagen RGB
21         img = Image.new('RGB', (ancho, alto))
22
23         # Generamos pixeles aleatorios
24         pixels = [(random.randint(0,255), random.randint(0,255),
25                    random.randint(0,255))
26                    for _ in range(ancho * alto)] 
27         img.putdata(pixels)
28
29         buffer = BytesIO()
30         img.save(buffer, format='PNG')
31         buffer.seek(0)
32
33         if formato.lower() == 'png':
34             # Devuelve la imagen directamente como PNG
35             return request.make_response(
36                 buffer.getvalue(),
37                 headers=[('Content-Type', 'image/png')])
38         else:
39             # Devuelve HTML con la imagen en Base64
40             img_str = base64.b64encode(buffer.getvalue()).decode('utf-8')
41             html = f'''
42             <div>
43                 <h3>Imagen aleatoria {ancho}x{alto}</h3>
44                 
45             </div>
46             ...
47         return html
```

Con esto hecho, actualizamos el módulo y probamos el Web Controller accediendo a la URL 'http://localhost:8069/imagen/aleatoria?ancho=300&alto=300' (indicando el alto y ancho deseados).



localhost:8069/imagen/aleatoria?ancho=300&alto=300

Imagen aleatoria 300x300



4. Problemas encontrados

Durante la realización de esta práctica he tenido algunos problemas básicos como sintaxis de Python o permisos mal configurados.

*Pero el principal problema lo he tenido en el apartado 4 de la primera actividad, el informe PDF para los partidos. Al intentar generar el PDF, me da un error de 'External ID not found'.

```
File "/usr/lib/python3/dist-packages/odoo/addons/base/models/ir_model.py", line 2246, in _xmlid_lookup
    raise ValueError('External ID not found in the system: %s' % xmlid)
ValueError: External ID not found in the system: EJ07-LigaFutbol.action_report_partido

The above server error caused the following client error:
RPC_ERROR: Odoo Server Error
RPC_ERROR
    at makeErrorFromResponse (http://localhost:8069/web/assets/4b4350f/web.assets_web.min.js:3159:163)
    at XMLHttpRequest.<anonymous> (http://localhost:8069/web/assets/4b4350f/web.assets_web.min.js:3164:13)
```

Cerrar

Por la experiencia que tengo con Odoo y lo que he investigado, debería ser un problema con la referencia (que la hubiera escrito mal) o que no haya declarado el XML en el manifest y Odoo no pueda verlo, pero aún revisandolo varias veces no he conseguido encontrar el error, por lo que no he podido completar esa parte de la práctica.

Algunos problemas menores que he tenido también fueron con la creación del bot de Telegram, la recálculo de las puntuaciones de los partidos y los botones para alterar los goles, pero no especialmente destacables

5. Conclusión

Esta práctica me ha servido para desenvolverme más en la creación y modificación de módulos en Odoo, la creación de Web Controllers, la interacción de una API REST con otro servicio diferente (en este caso, el bot de Telegram).