Práctica 5

Módulos Odoo Controlador, Herencia y Web Controllers

Ángel Panadero Rodríguez.

2º Desarrollo de Aplicaciones multiplataforma.

1. Actividad 01 – Modificación del módulo EJ07-LigaFutbol
    1.1 Reglas de puntuación especiales.

    1.2 Botones para alterar los goles de los partidos.

    1.3 Web controller para eliminar empates.

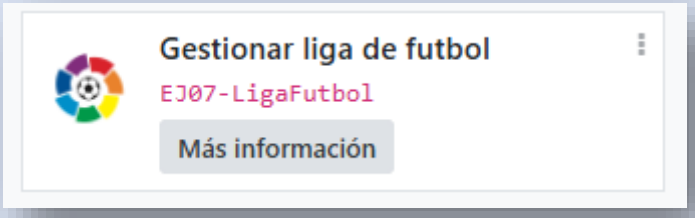
    1.4 Informe PDF por cada partido.

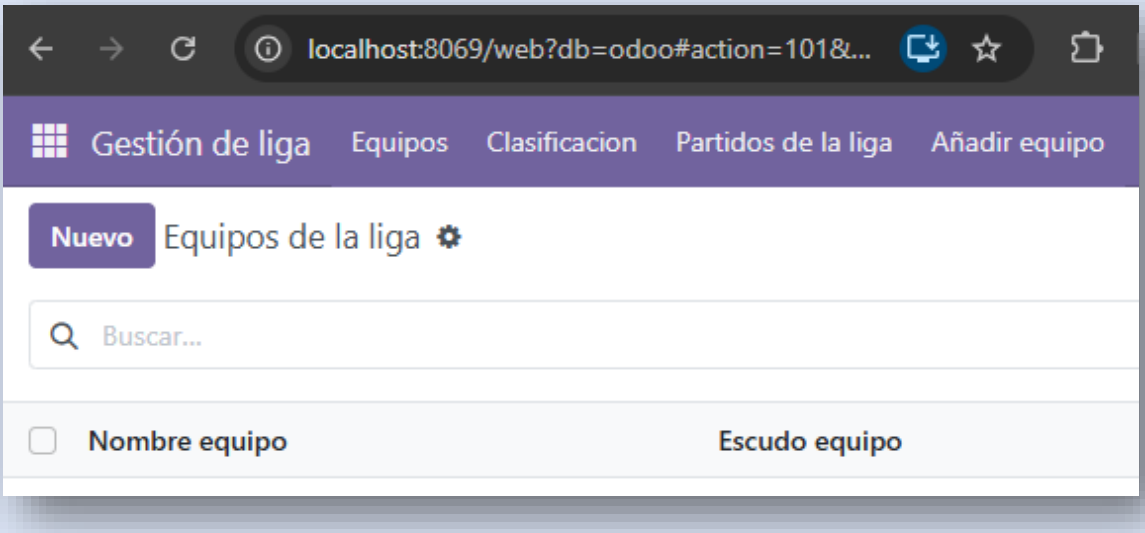
    1.5 Wizard para crear nuevos partidos.

    1.6 Vista Graph
2. Actividad 02 – Pruebas del API REST EJ08-API-REST_Socios
3. Actividad 03 – Bot de Telegram conectado a la API REST
4. Actividad 04 – Generación de imágenes aleatorias con Web Controller

Actividad 01 – Modificación del módulo EJ07-LigaFutbol

Para este punto, lo primero que he hecho ha sido insertar la carpeta del módulo dentro de la carpeta addons. Una vez cargados todos los datos me he ido a odoo con el modo desarrollado activado, he actualizado la lista de aplicaciones, busqué el módulo, lo instalé y comprobé que todo se visualizaba de forma correcta:





Tras esto, he hecho un commit a modo de imagen para poder volver atrás fácil y de forma cómoda.

1.1 Reglas de puntuación especiales.

Lo primero que he hecho ha sido localizar como se guardaban los puntos. Están definidos en liga_equipos.py de la siguiente manera:

```python
    puntos= fields.Integer( compute="_compute_puntos",default=0,
store=True)

    @api.depends('victorias','empates')
    def _compute_puntos(self):
        for record in self:
            record.puntos = record.victorias * 3 + record.empates
```

El objetivo es que si en un partido se sacan 4 goles o más de diferencia se le sumen 4 puntos al ganador y se le resten 1 al perdedor.

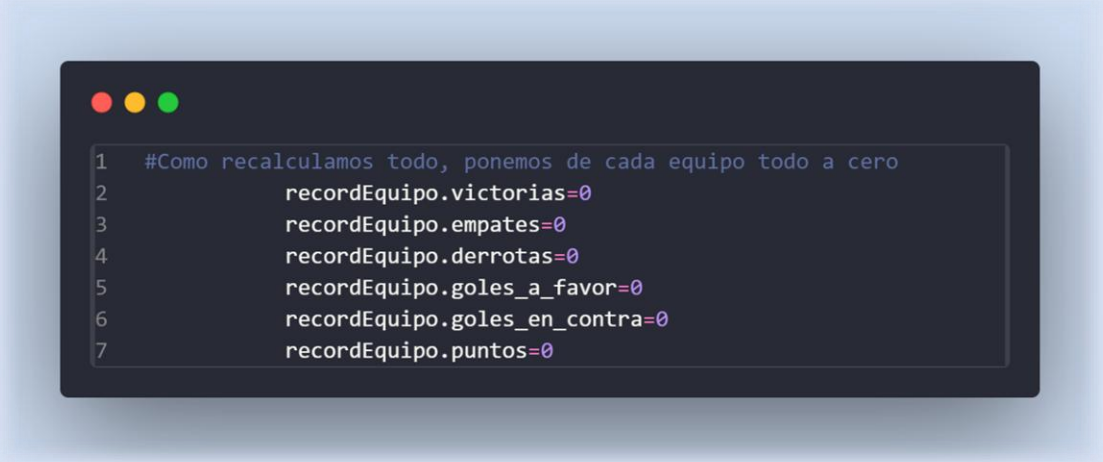
He convertido el campo de puntos a un campo "normal" como victorias, empates… Elimino lo de el compute y lo pongo como Integer normal quedando el siguiente resultado:

```python
puntos=fields.Integer(default=0)
```

Ahora debo aplicar la lógica para que los puntos sean alterados en caso de goleada.

Para ello, debo ir a la función que guarda los registros de un Equipo tras un partido. Esa función se encuentra en el archivo liga_partido.py, se llama actualizoRegistrosEquipos().

Ponemos que los puntos a 0 al comenzar igual que el resto de variables:



```
#Como recalculamos todo, ponemos de cada equipo todo a cero
            recordEquipo.victorias=0
            recordEquipo.empates=0
            recordEquipo.derrotas=0
            recordEquipo.goles_a_favor=0
            recordEquipo.goles_en_contra=0
            recordEquipo.puntos=0
```

Dentro del bucle que guarda los datos del partido he puesto lo siguiente antes de todo. En esta práctica voy a usar los comentarios para aprovechar mejor las imágenes por lo que habrá muchos puntos a lo largo de esta práctica que no comente en texto pero sí habré comentado en el código de las imágenes.

- Declaro la variable de los puntos de local y del visitante para luego poder diferenciar los sumatorios
- En caso de que sea empate, ambos se llevan un punto.
- En caso de que gane el local:
    - Si gana por 4 goles, sumo 4 puntos y resto 1 al visitante.
    - Sino sumo 3 puntos y no sumo ninguno al visitante, pero tampoco resto.
- Si gana el visitante:
    - Lo mismo que el local pero al revés.

```
    for recordPartido in self.env['liga.partido'].search([]):
                puntos_local=0
#declaro los puntos del local y visitante
                puntos_visitante=0
                #si hay empate
                if recordPartido.goles_casa==
recordPartido.goles_fuera:
                    puntos_local=1
                    puntos_visitante=1
                else: #no hay empate - calculo diferencia
                    diferencia = recordPartido.goles_casa -
 recordPartido.goles_fuera
                #local si...
                    if diferencia>0:
                        if diferencia>=4:
                            puntos_local=4
                            puntos_visitante= -1
                        else:
                            puntos_local=3
                            puntos_visitante=0
                #visitante sino...
                    else:
                        if diferencia<=-4:
                            puntos_local=-1
                            puntos_visitante=4
                        else:
                            puntos_visitante=3
                            puntos_local=0
```

Ahora se están aplicando los puntos en las variables definidas dentro de la función pero necesitamos hacer que se sumen / resten cuando toquen. Para ello, en el código nos vamos a la zona comentada como

```
1            #Si es el equipo de casa
2                 if recordPartido.equipo_casa.nombre==recordEquipo.nombre:
3                     recordEquipo.puntos = recordEquipo.puntos + puntos_local
4                     #Miramos si es victoria o derrota
5                     if recordPartido.goles_casa>recordPartido.goles_fuera:
6                         recordEquipo.victorias=recordEquipo.victorias+1
7                     elif recordPartido.goles_casa<recordPartido.goles_fuera:
8                         recordEquipo.derrotas=recordEquipo.derrotas+1
9                     else:
10                        recordEquipo.empates=recordEquipo.empates+1
11
12                    #Sumamos goles a favor y en contra
13                    recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_casa
14                    recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_fuera
15
16                #Si es el equipo de fuera
17                if recordPartido.equipo_fuera.nombre==recordEquipo.nombre:
18                    recordEquipo.puntos = recordEquipo.puntos + puntos_visitante
19                    #Miramos si es victoria o derrota
20                    if recordPartido.goles_casa<recordPartido.goles_fuera:
21                        recordEquipo.victorias=recordEquipo.victorias+1
22                    elif recordPartido.goles_casa>recordPartido.goles_fuera:
23                        recordEquipo.derrotas=recordEquipo.derrotas+1
24                    else:
25                        recordEquipo.empates=recordEquipo.empates+1
26
27                    #Sumamos goles a favor y en contra
28                    recordEquipo.goles_a_favor=recordEquipo.goles_a_favor+recordPartido.goles_fuera
29                    recordEquipo.goles_en_contra=recordEquipo.goles_en_contra+recordPartido.goles_casa
30
```

Guardo los puntos dentro de las variables como se puede comprobar en la línea 3 y 18 de la captura.

Luego he guardado los archivos y hecho un commit.

He actualizado la lista de aplicaciones y simplemente asegurado de que no me devolvía ningún error. Al ejecutar un partido donde haya una diferencia superior, efectivamente hace el cálculo.

Partidos:

Clasificación:

1.2 Botones para alterar los goles de los partidos.

En este punto debemos crear un botón para sumar 2 goles al equipo local y otro botón para sumar 2 goles al equipo visitante.

Lo que he hecho ha sido generar dos métodos en el archivo liga_partido.py

Los métodos que he creado lucen así:

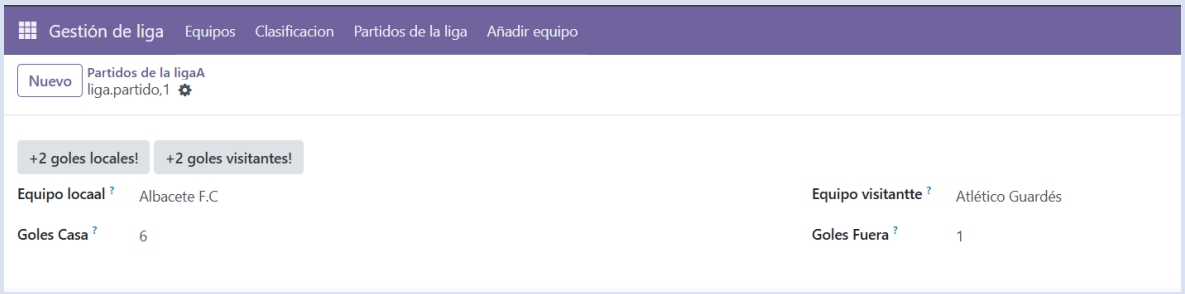
```python
#+2 goles
def sumar_dos_locales(self):
    partidos = self.env['liga.partido'].search([])
    for partido in partidos:
#saca cada partido para poder hacerlo sobre todos los registros
        partido.goles_casa = partido.goles_casa + 2 #le suma 2 goles al equipo
    self.actualizoRegistrosEquipo()
def sumar_dos_visitantes(self):
    partidos = self.env['liga.partido'].search([])
    for partido in partidos:
        partido.goles_fuera = partido.goles_fuera + 2
    self.actualizoRegistrosEquipo()
```

He comentado como funciona.

Con ello, he tenido que adjuntar la siguiente modificación a la vista liga_partido.xml:

```xml
<header>
            <button name="sumar_dos_locales"
                type="object"
                string="+2 goles locales!"/>
            <button name="sumar_dos_visitantes"
                type="object"
                string="+2 goles visitantes!"/>
            </header>
```

En la interfaz lucen así:



En este punto, la vista no se me actualizaba. Lo que suelo hacer para actualizar todo cuando hago algún cambio es:

Docker compose restart web

Actualizar lista de aplicaciones (vista odoo modo desarrollador)

Pero esta vez no ha sido suficiente. Tras probar diversas cosas me salvó el comando:

odoo -d [db] -u EJ07-LigaFutbol --stop-after-init

Sigo sin conocer a que se debió esto, odoo no dio localizado que el archivo vista había sufrido cambios.

Por eso puedes ver nombres que quizás no cuadran tanto en la captura anterior (son intentos frustrados de cambiar algo en la vista)

He hecho un push a mi repositorio con los cambios y avanzado al siguiente punto.

1.3 Web controller para eliminar empates.

Para este punto vamos a crear un controller HTTP para eliminar todos los partidos empate de la lista de partidos.

Para ello, lo primero que debemos hacer será entrar en nuestro archivo main.py (del módulo LigaFutbol) y meteremos el controlador al cual he modificado dandole el siguiente formato en Python:

```python
@http.route('/eliminarempates', type='http', auth='none')
def eliminarEmpates(self, **kw):
    partidos = request.env['liga.partido'].sudo().search([])

    empates = []
    for p in partidos:
        if p.goles_casa == p.goles_fuera:
            empates.append(p)

    eliminados = len(empates)
    for p in empates:
        p.unlink()

    return "Partidos eliminados: %s" % eliminados
```

(Ya había otra ruta de controlador antes, la he conservado ya que es necesaria para el funcionamiento del módulo) Este fragmento lo que hace es:

- Busca todos los partidos.
- Guarda en una variable solo los partidos empatados comparando los goles de fuera con los de casa.
- Guarda todos los partidos que debe eliminar, los cuenta para imprimir cuantos borró y borra cada uno de la lista de partidos.

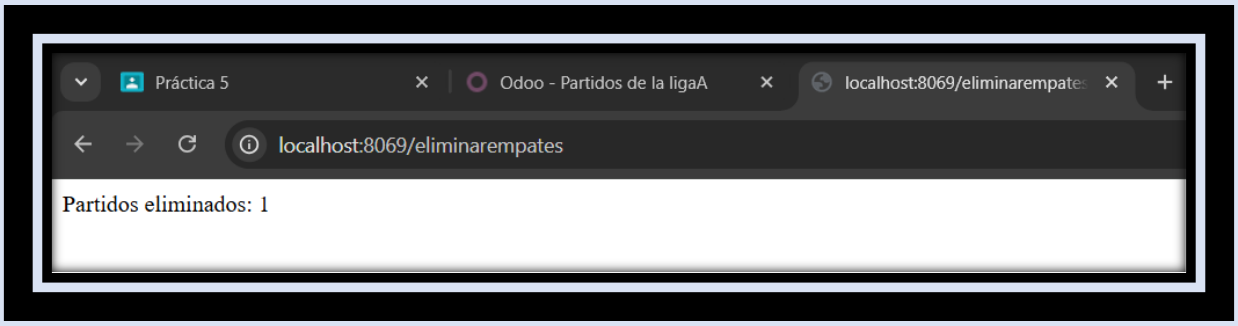
Como ya había una ruta de controlador antes, supongo que no habrá que configurar nada a mayores por lo que me limito a reiniciar y actualizar.

He creado un partido empatado:

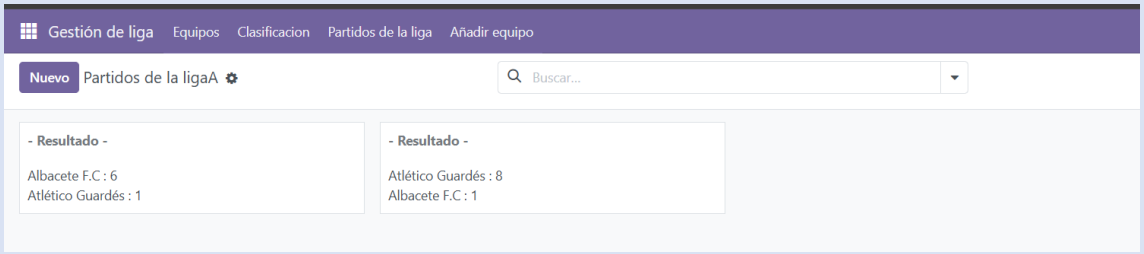
En teoría debería eliminarse accediendo a la ruta:

http://localhost:8069/eliminarempates

Accedemos:



Efectivamente se ha eliminado de forma correcta:



Tras esto, como siempre, he subido un push a mi repositorio con los cambios aplicados. Paso al siguiente punto.

1.4 Informe PDF por cada partido.

Para este punto lo que vamos a hacer será crear un report en xml. Esto sirve para coger los datos que seleccionemos e imprimirlos con una plantilla QWeb, esto imita un formato predefinido para la consulta de datos.

Lo primero que haremos será crear en la carpeta de /report un nuevo archivo (ya hay uno que ignoraremos). Yo le he llamado liga_partido_informe.xml pero por lo que veo lo normal es ponerle "report" al final.

Dentro he definido lo siguiente:

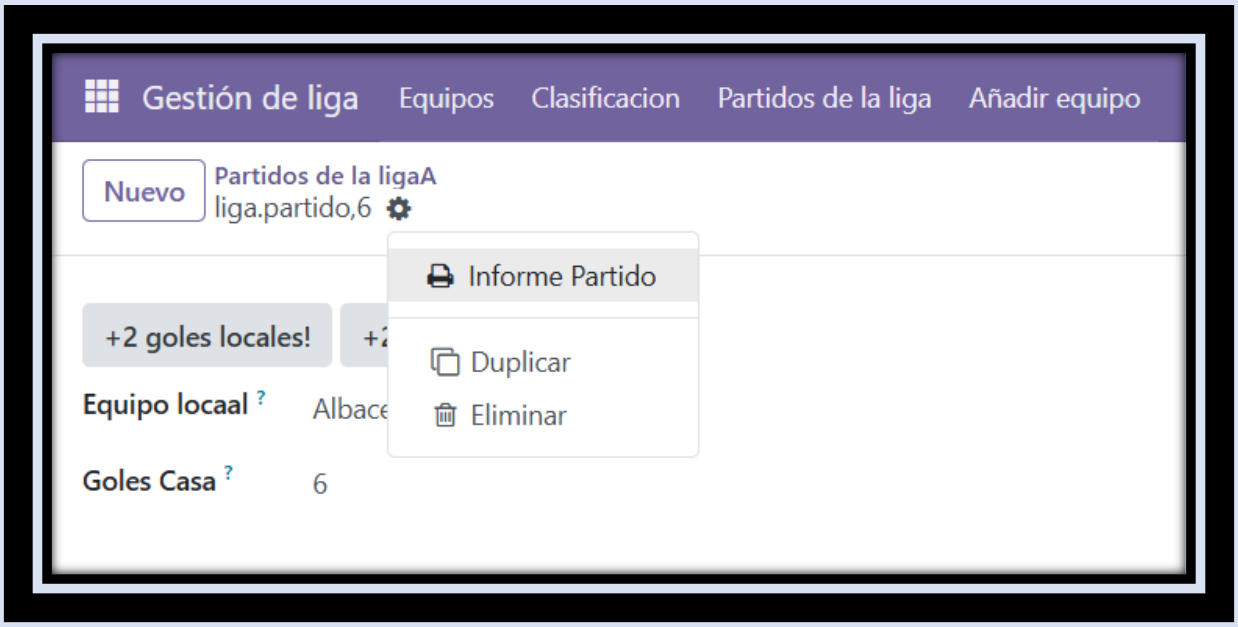
```xml
<?xml version="1.0" encoding="utf-8"?>
<odoo>
    <report
        id="liga_partido_informe"
        model="liga.partido"
        string="Informe Partido"
        report_type="qweb-pdf"
        name="EJ07-LigaFutbol.liga_partido_informe_template"
        file="EJ07-LigaFutbol.liga_partido_informe_template"
    />
    <template id="liga_partido_informe_template">
        <t t-call="web.html_container">
            <t t-call="web.external_layout">
                <div class="page">
                    <h2>Informe de Partido</h2>

                    <t t-foreach="docs" t-as="p">
                        <p><b>Equipo local:</b> <span t-esc="p.equipo_casa.nombre"/></p>
                        <p><b>Goles local:</b> <span t-esc="p.goles_casa"/></p>

                        <p><b>Equipo visitante:</b> <span t-esc="p.equipo_fuera.nombre"/></p>
                        <p><b>Goles visitante:</b> <span t-esc="p.goles_fuera"/></p>

                        <p><b>Resultado:</b>
                            <span t-esc="p.goles_casa"/> - <span t-esc="p.goles_fuera"/>
                        </p>
                    </t>

                </div>
            </t>
        </t>
    </template>
</odoo>

```

Con el formato del informe ya definido , lo siguiente que haremos es que odoo cargue nuestro archivo añadiendo esta línea dentro del manifest en el data:

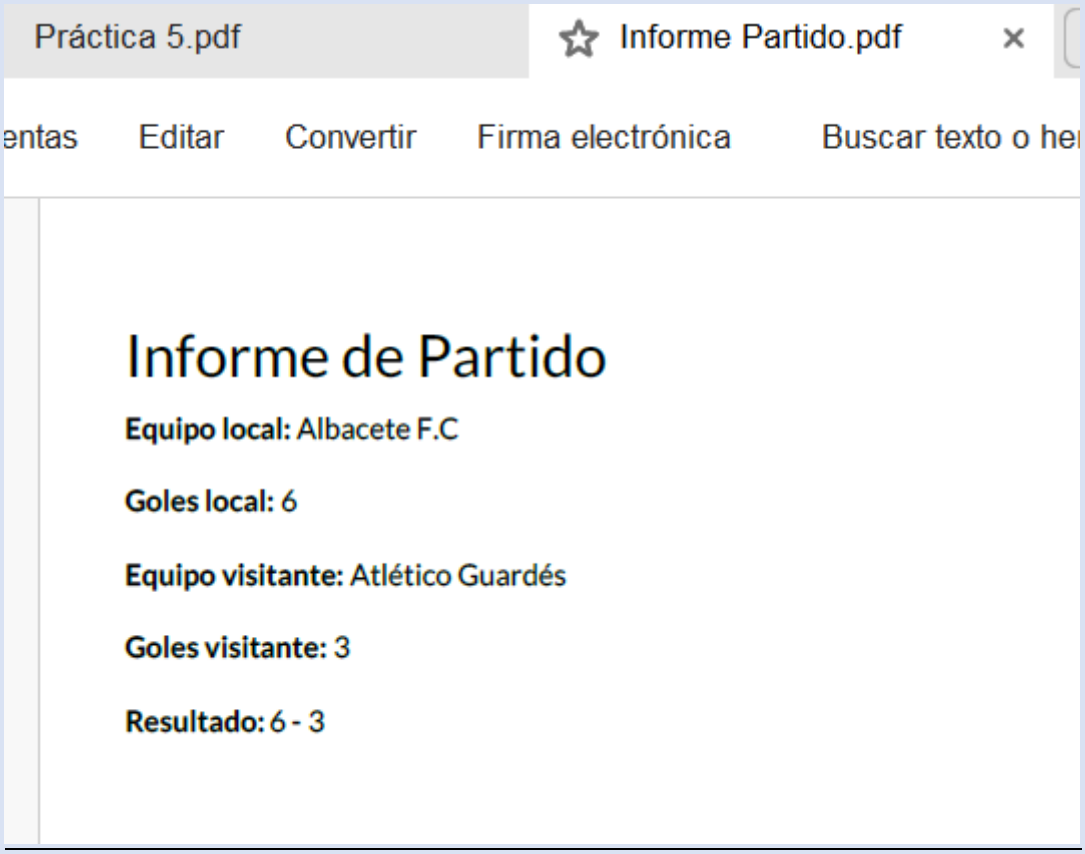
'report/liga_partido_informe.xml'

Ahora nos aparecerá una opción de “Imprimir”-“informe partido”

Vamos a probarlo:



Tenemos la opción de Informe Partido. Si hacemos click, adjuntará a nuestra carpeta de descargas un pdf con el siguiente formato:



Está claro que podemos poner más formal / estético el Pdf.

"Pusheamos" y a otro tema.

1.5 Wizard para crear nuevos partidos.

Para ello, lo primero que he hecho ha sido añadir el campo de la clase LigaPartido

jornada = fields.Integer(*string*='Jornada')

Dentro de la carpeta /wizard he creado el archivo liga_partido_wizard.py (imitando esta vez el formato del otro archivo) y liga_partido_wizard.xml

Tras esto, debemos agregar al manifest la ruta del archivo:

'wizard/liga_partido_wizard.xml'

Los archivos tienen el siguiente formato, esta es la vista:

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

    <record id="liga_partido_wizard_form" model="ir.ui.view">
        <field name="name">Wizard para introducir un Partido</field>
        <field name="model">liga.partido.wizard</field>
        <field name="arch" type="xml">
            <form string="Introducir datos de un partido">
                <sheet>
                    <group>
                        <field name="jornada"/>
                    </group>
                    <group>
                        <field name="equipo_casa"/>
                        <field name="goles_casa"/>
                    </group>
                    <group>
                        <field name="equipo_fuera"/>
                        <field name="goles_fuera"/>
                    </group>
                </sheet>
                <footer>
                    <button string="Añadir" name="add_liga_partido" class="btn-primary" type="object"/>
                    <button string="Cancelar" class="btn-default" special="cancel"/>
                </footer>
            </form>
        </field>
    </record>

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

    <menuitem id="menu_wizard_liga_partido"
              name="Añadir partido"
              parent="liga_partido_menu"
              action="action_wizard_liga_partido"
              sequence="10"/>

</odoo>

```

Inspirado en la vista que ya trae el ejemplo

```python
from odoo import models, fields

class LigaPartidoWizard(models.TransientModel):
    _name = 'liga.partido.wizard'

    equipo_casa = fields.Many2one('liga.equipo', string='Equipo local', required=True)
    equipo_fuera = fields.Many2one('liga.equipo', string='Equipo visitante', required=True)
    goles_casa = fields.Integer(string='Goles local', default=0)
    goles_fuera = fields.Integer(string='Goles visitante', default=0)
    jornada = fields.Integer(string='Jornada', default=1)

    def add_liga_partido(self):
        ligaPartidoModel = self.env['liga.partido']
        for wiz in self:
            ligaPartidoModel.create({
                'equipo_casa': wiz.equipo_casa.id,
                'equipo_fuera': wiz.equipo_fuera.id,
                'goles_casa': wiz.goles_casa,
                'goles_fuera': wiz.goles_fuera,
                'jornada': wiz.jornada,
            })

```

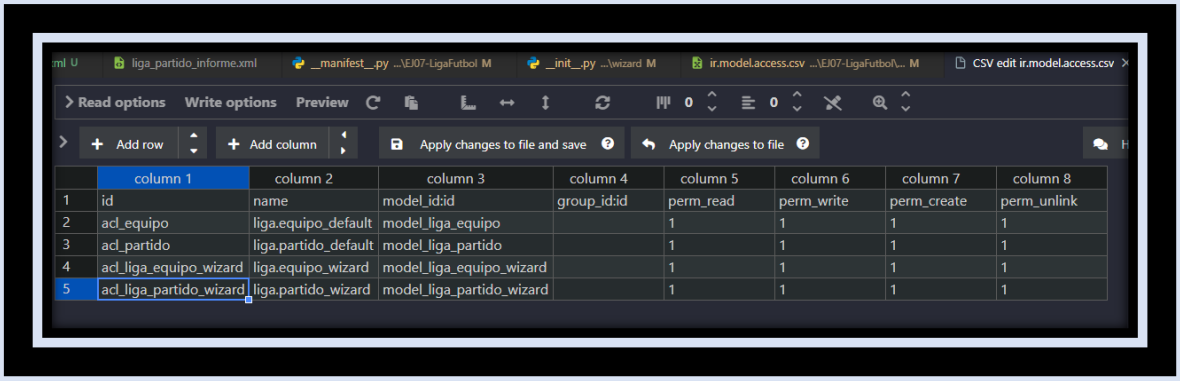
Código de la lógica.

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

    <record id="liga_partido_wizard_form" model="ir.ui.view">
        <field name="name">Wizard para introducir un Partido</field>
        <field name="model">liga.partido.wizard</field>
        <field name="arch" type="xml">
            <form string="Introducir datos de un partido">
                <sheet>
                    <group>
                        <field name="jornada"/>
                    </group>
                    <group>
                        <field name="equipo_casa"/>
                        <field name="goles_casa"/>
                    </group>
                    <group>
                        <field name="equipo_fuera"/>
                        <field name="goles_fuera"/>
                    </group>
                </sheet>
                <footer>
                    <button string="Añadir" name="add_liga_partido" class="btn-primary" type="object"/>
                    <button string="Cancelar" class="btn-default" special="cancel"/>
                </footer>
            </form>
        </field>
    </record>

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

    <menuitem id="menu_wizard_liga_partido"
              name="Añadir partido"
              parent="liga_partido_menu"
              action="action_wizard_liga_partido"
              sequence="10"/>

</odoo>

```

Ahora actualizamos el proyecto y accedemos a el.

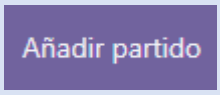
En este punto comenzó a fallarme. Con la ayuda de un compañero di sacado que debía dar permisos al wizard. Para ello, como en la práctica anterior, debía acceder al archivo de ir.access.model.csv

Con la ayuda de una extensión de vsc logro ver de forma más sencilla el csv. Al final un .csv es una hoja de cálculo. Le doy permisos al wizard:



	column 1	column 2	column 3	column 4	column 5	column 6	column 7	column 8
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

Ahora sí, comprobamos:



Podremos añadir partidos a través del wizard.

1.6 Vista Graph

Para poder añadir una vista de gráfico, lo primero que haremos será declarar que esa vista existe al principio de nuestro liga_partido.xml:

<field name="view_mode">kanban,tree,form,graph</field>

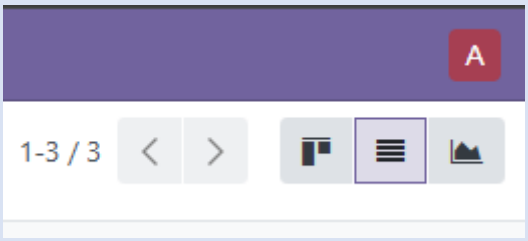
(LÍNEA GRAPH)

El archivo de liga_equipo.xml ya incorpora una vista en gráfico por lo que he copiado un poco el formato añadiendo esto a mi código:

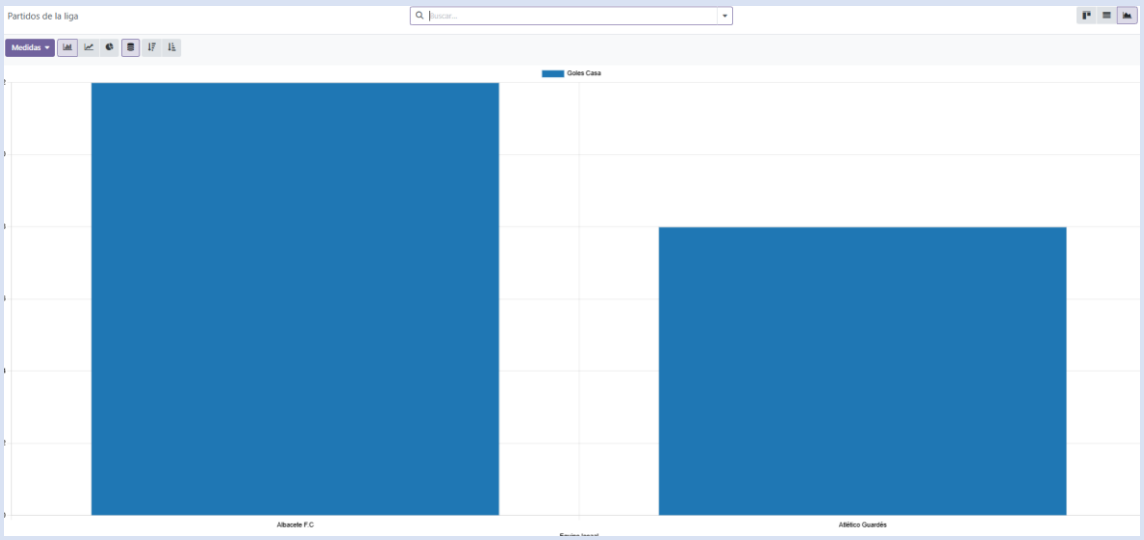
```xml
<record id="liga_partido_view_graph" model="ir.ui.view">
    <field name="name">Goles locales por equipo</field>
    <field name="model">liga.partido</field>
    <field name="arch" type="xml">
        <graph string="Goles marcados por equipos locales" type="bar">
            <field name="equipo_casa" type="row"/>
            <field name="goles_casa" type="measure"/>
        </graph>
    </field>
</record>
```

- Agrupamos por equipo local
- Sumamos goles locales

Ahora, debemos actualizar el contenedor y comprobar la vista:



A la derecha nos aparece el icono del gráfico. Si accedemos a el:



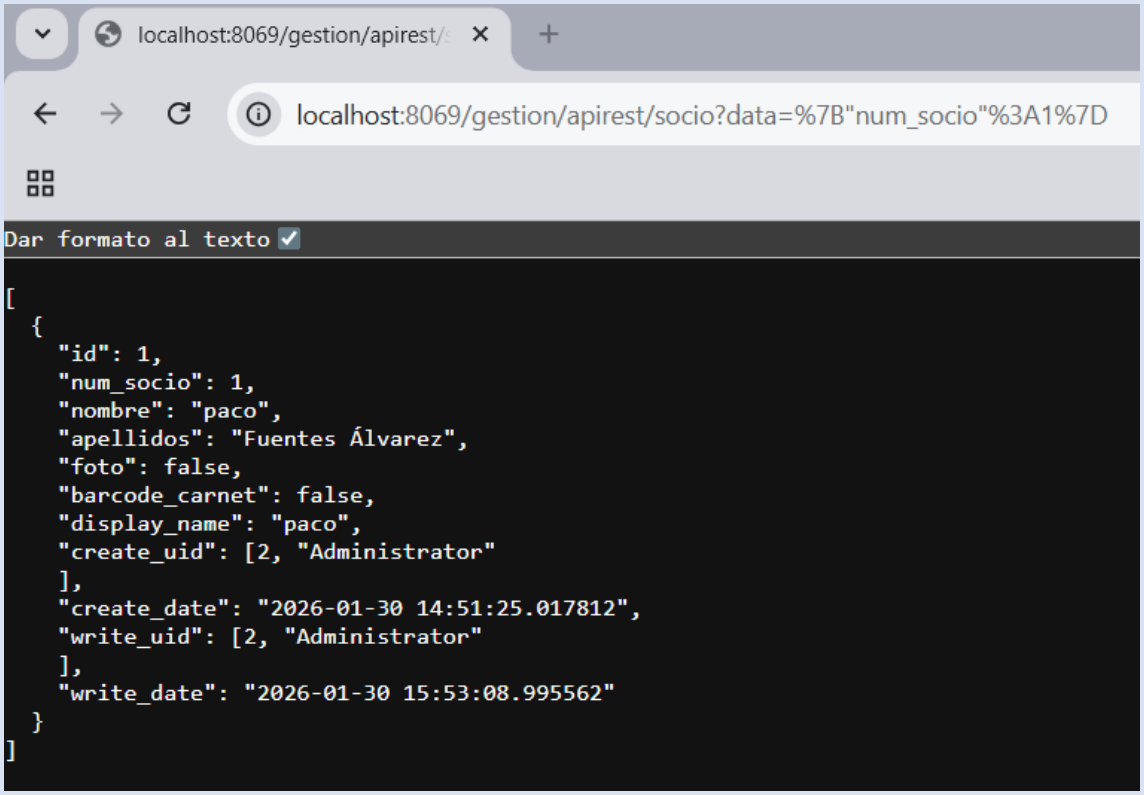
Yo solo tengo dos equipos pero los datos que muestra concuerdan con la realidad.

Actividad 02 – Pruebas del API REST EJ08-API-REST_Socios

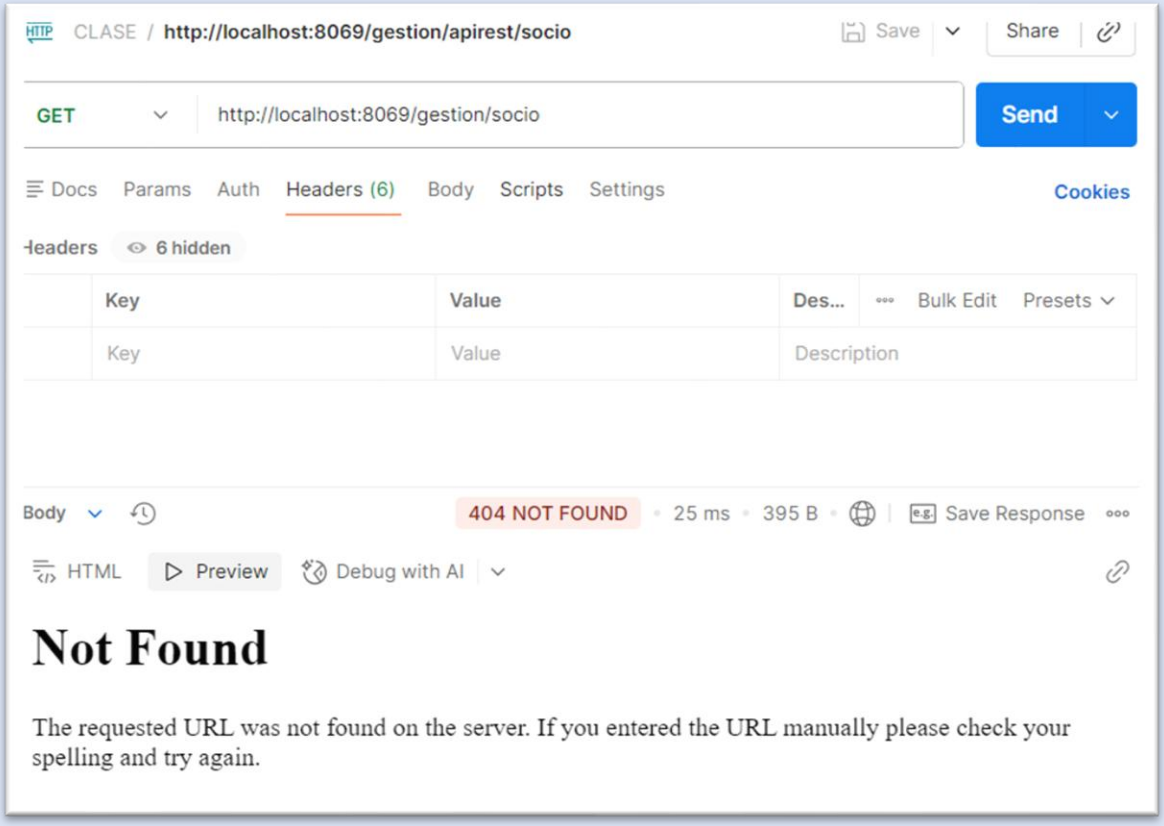
He activado el módulo. Lo he lanzado en mi plataforma POSTMAN, la cual llevo usando todas las semanas en los últimos 4 meses en este dispositivo.

- Configurado de forma “default”, sin cosas raras.
- Equipo no limitado y totalmente liberado.

Pese a múltiples intentos y horas perdidas en diferentes días y puestos de trabajo, no he logrado solucionar el siguiente error: Los enpoint no devuelven nada en aplicaciones como Postman pero SI devuelve correctamente el resultado en navegador:

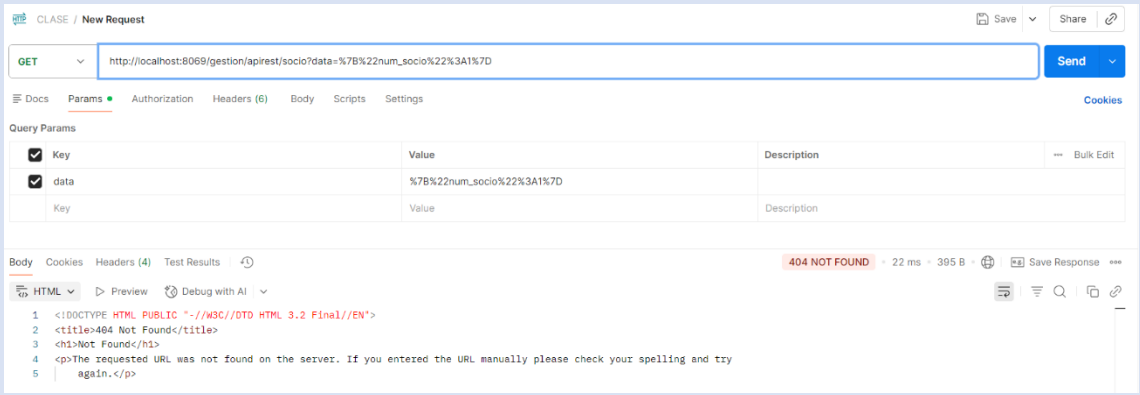


Al lanzar cualquier endpoint. Me devolvía un error como el siguiente:

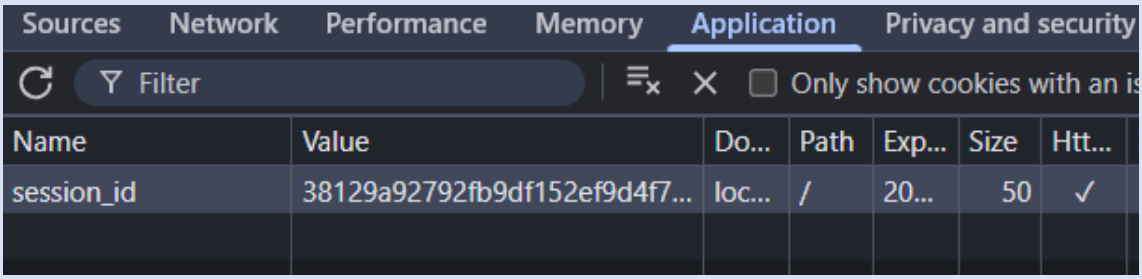


He probado:

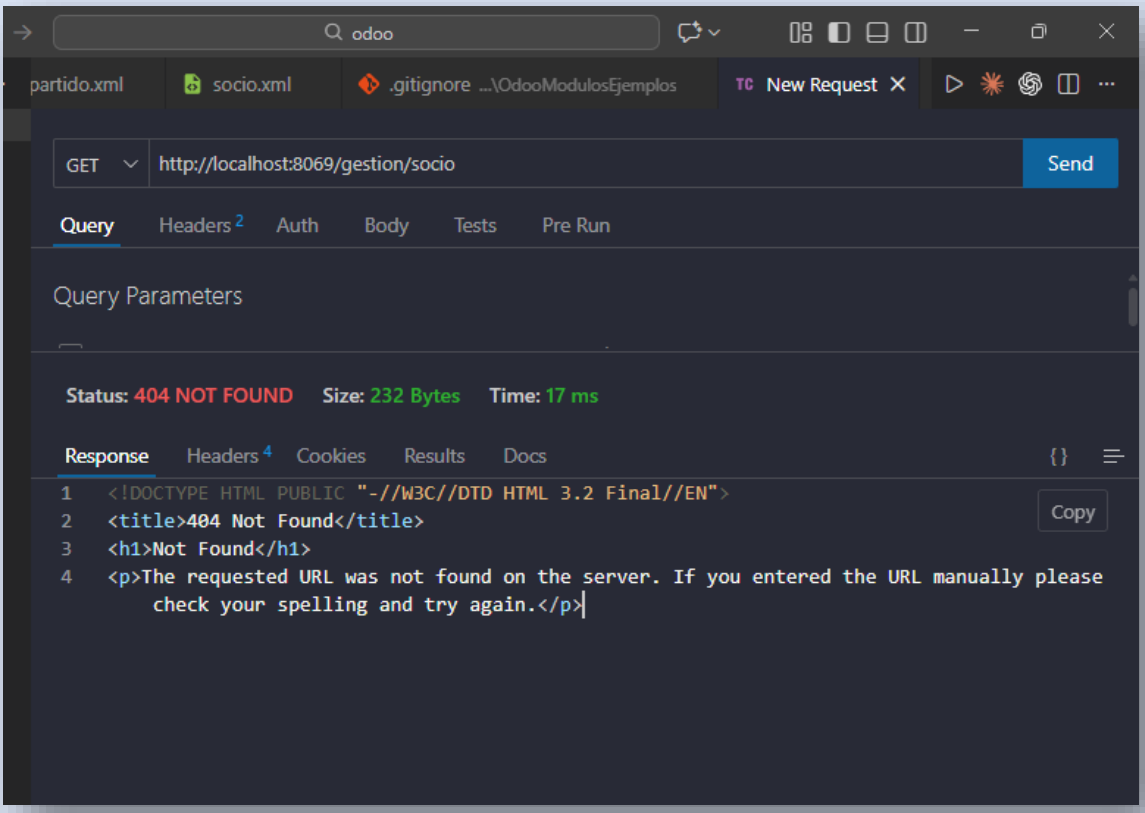
- Encodear los parámetros que pasamos en el endpoint:



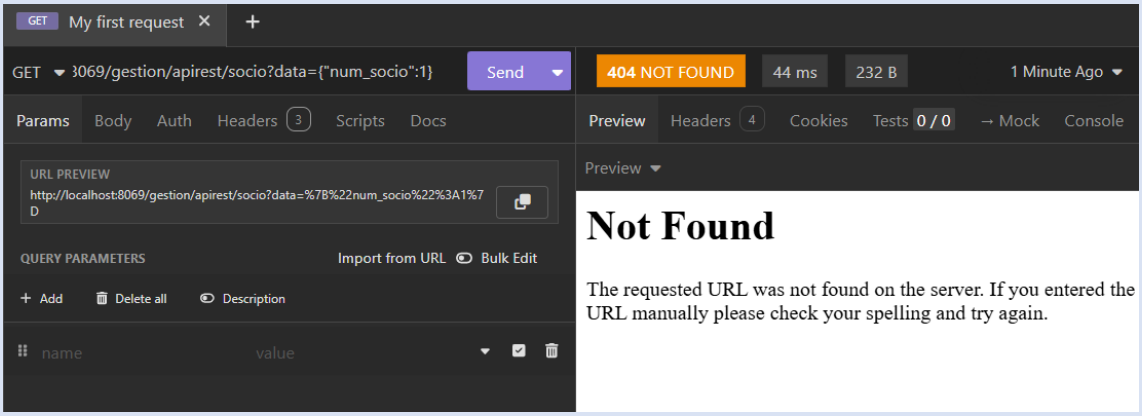
- Pasarle la cookie del session_id del navegador:



- Probar la extensión Thunder para VSC



- Probar Insomnia en su prueba pro gratuita de 14 días con unas configuraciones que me proporciono una IA basándose en mi error:



- He probado a concretar la base de datos en el endpoint de forma fallida
- Múltiples pruebas fallidas un poco más “light”

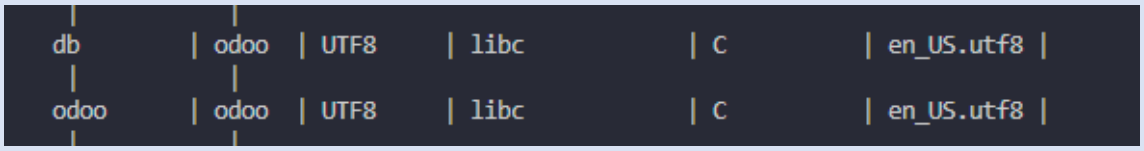
Llevo 3-4 días intentando solucionar este problema de forma fallida.

Hoy, sábado, tras 3 horas buscando recursos en Google y preguntando cosas a la IA, me enorgullece (de verdad que me enorgullece) escribir que **HE SOLUCIONADO EL PROBLEMA.**

¿Cuál era el problema?

Odoo permite trabajar con diversas bases de datos, pudiendo intercambiar de forma fácil y cómoda a través de su interfaz.

En su día, cree una base de datos a modo de prueba para un par de cosillas que quería hacer y no me salieron. No le he dado importancia, deje esa base de datos ahí y seguí trabajando en la otra:



La de arriba fue la primera que cree, para pruebas. La de abajo "odoo" es la que uso ahora, donde tengo instalado el módulo con la API para gestionar socios.

Postman hace una petición general a un puerto. No concreta una base de datos sino que comprueba si ese puerto devuelve "algo".

En resumen: **NO SABE QUE BASE DE DATOS USAR**

Pero… en el navegador si funciona. Correcto, funciona en el navegador debido a que tengo la sesión iniciada Y , la api ya sabe a que base de datos apuntar gracias a las cookies.

¿Cómo lo he solucionado?

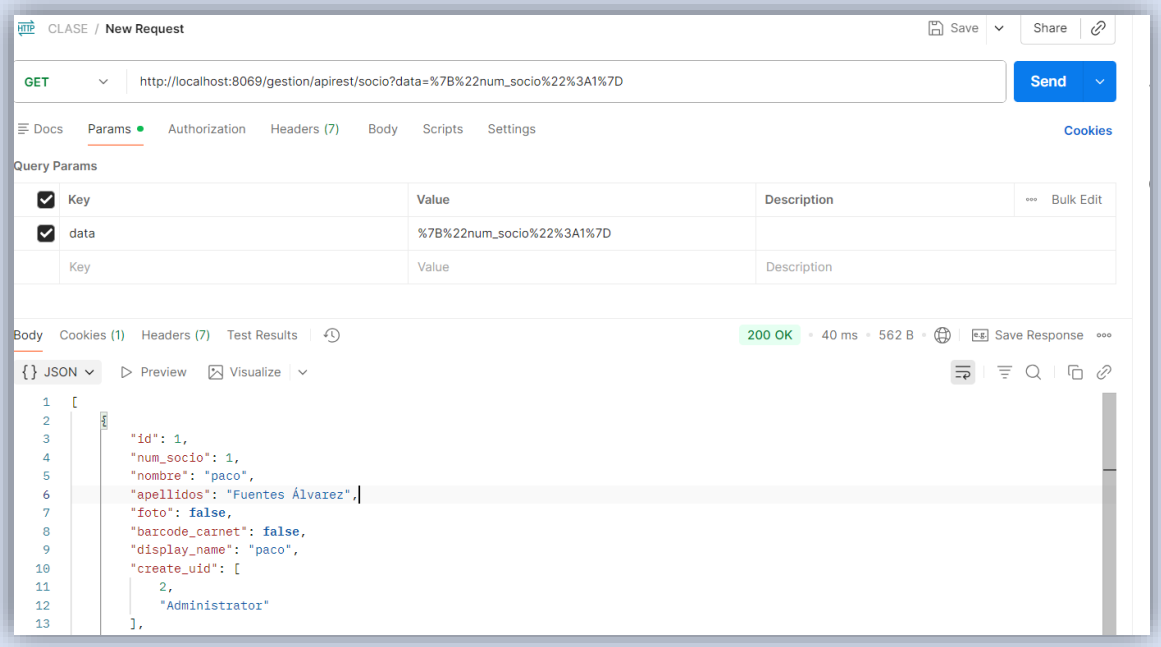
He consultado si había alguna manera de limitar la base de datos a la que debe apuntar odoo, sí, existe.

Son estas 3 simples líneas:

```
db_name = odoo
list_db = False
dbfilter = ^odoo$
```

- Apunta solo a la base de datos “odoo”
- No listes ninguna base de datos más a no ser que comience por odoo

Tras ponerlas en mi archivo de configuración odoo.conf y probar la API en postman (esperando que, como TODO el resto que probé no fuese a funcionar):



Hay algo que tengo claro desde hace tiempo pero esta práctica me ha vuelto a enseñar que es importante no frustrarse con los errores, es realmente de valor saber parar antes de llegar a la frustración y tomarse un respiro antes de continuar.

El saber resolver vale mucho.

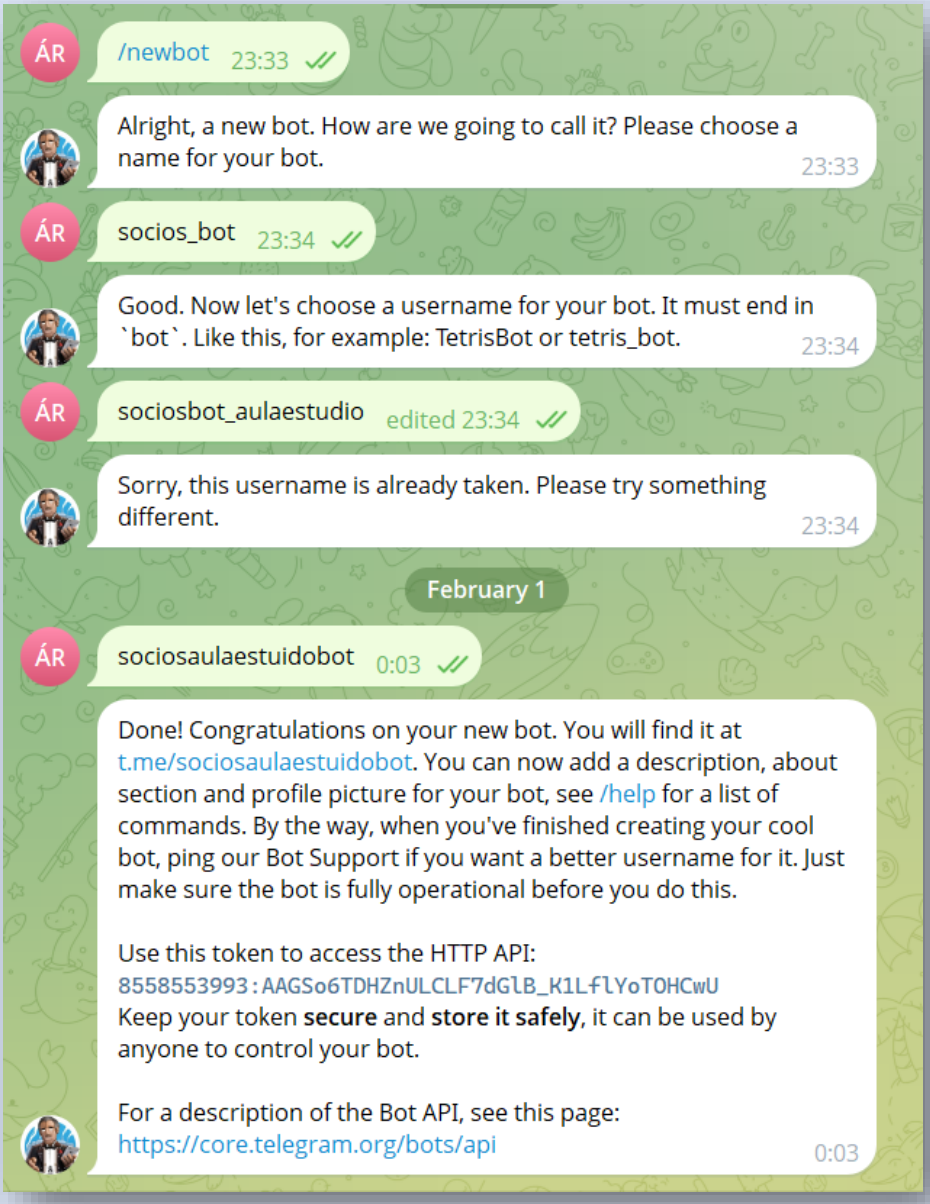
A raíz de esto me he colocado en mi habitación un poste personalizado de las 3 líneas de odoo.conf que me han salvado el fin de semana, lo he compartido con mis compañeros, puede parecer que esto es un poco absurdo y obvio pero esto le estaba pasando a MUCHOS de mis compañeros, he compartido la resolución con ellos. A continuación, te dejo con el vídeo:

Vídeo demostración del módulo API REST y explicación del problema.

Actividad 03 – Bot de Telegram conectado a la API REST

Para esta práctica lo primero que haré será crear un bot de telegram. Para ello, como siempre, haremos uso de botfather en Telegram. Buscaremos bot Father y comenzamos con la configuración.

En el chat, colocamos /newbot y comenzamos la configuración:

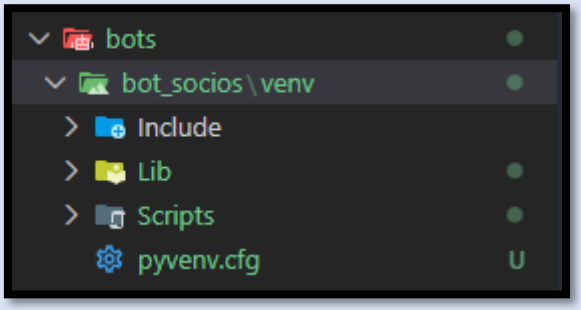


Dio la casualidad de que alguien ya ha intentado crear sociosbot_aulaestudio, casualidad.

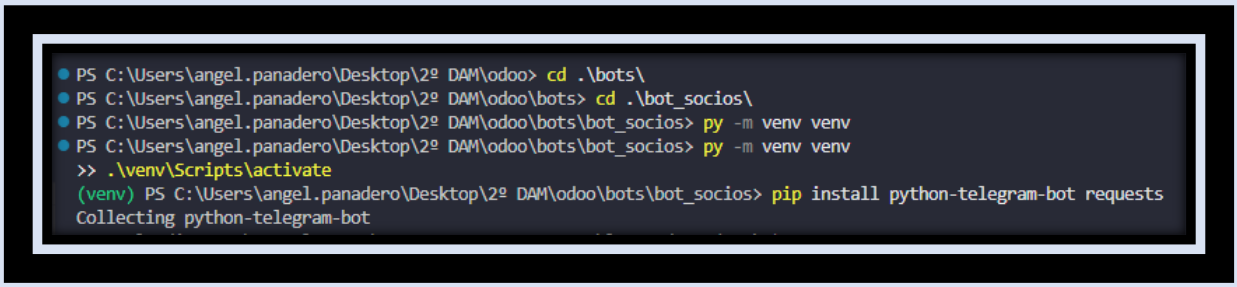
El mensaje de success nos proporciona el token del bot. Lo guardaremos.

Lo siguiente que he hecho ha sido preparar el entorno. Para ello, en una subcarpeta de nuestro proyecto, instalaremos la librería

Python-telegram-bot



Correcta estructuración de directorio.



Comandos que he usado.

Es la librería más usada para estas cosas en Python.

Con todo preparado, pasamos a la acción.

El bot necesita poder:

- Crear
- Modificar
- Consultar
- Borrar

Lo que está claro es que debemos buscar una manera de decirle al bot que interprete comandos concretos lanzando peticiones a nuestra API.

Para ello, lo que he hecho ha sido crear un archivo que gestione esto. Dento de la carpeta de bot_socios, donde tengo mi librería importada, creo el archivo bot_socios.py

El objetivo será el de gestionar las llamadas a través del chat.

```python
TOKEN = "8558553993:AAGSo6TDHZnULCLF7dGlB_K1LflYoTOHCwU"
ODOO_BASE = "http://localhost:8069"
API = f"{ODOO_BASE}/gestion/apirest/socio"

```

Variables de :

- TOKEN del bot
- Ruta de odoo
- API, estructura del endpoint que es siempre igual.

```python
def comando(texto: str):
    if not texto:
        return None, None

    partes = [p.strip() for p in texto.split(",") if p.strip()]
    if not partes:
        return None, None

    accion = partes[0].lower()

    pares = re.findall(r'(\w+)\s*=\s*"([^"]*)"', texto)
    datos = {k: v for k, v in pares}

    if "num_socio" in datos:
        try:
            datos["num_socio"] = int(datos["num_socio"])
        except ValueError:
            return accion, {"error": "num_socio debe ser número"}

    return accion, datos

```

Función que interpreta los mensajes del usuario buscando un comando.

Definimos que espera mensajes tipo Crear, Consultar, Borrar

- Si el mensaje esta vacio lo rechaza (línea 3)
- Divide el texto por comas, quita espacios, sino queda nada que se pueda interpretar comando, RECHAZA
- Guarda acción
- Solo se queda con coincidencias del patron como nombre, apellidos, numsocio
- Devuelve la lista.

Funciones odoo de la API

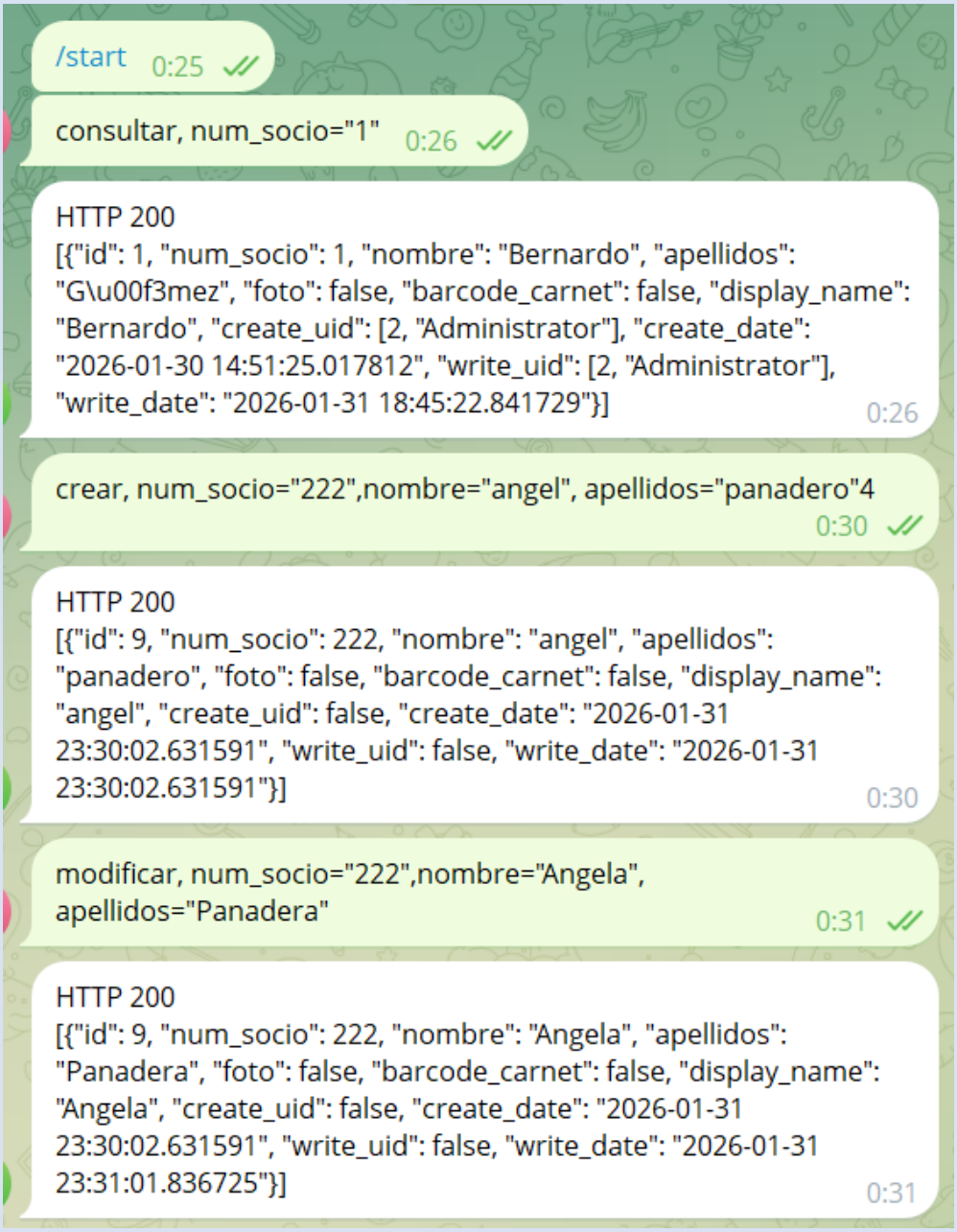
```python
def odoo_crear(datos):
    r = requests.post(API, json=datos, timeout=10)
    return r.status_code, r.text

def odoo_modificar(datos):
    r = requests.put(API, json=datos, timeout=10)
    return r.status_code, r.text

def odoo_consultar(num_socio):
    params = {"data": json.dumps({"num_socio": num_socio})}
    r = requests.get(API, params=params, timeout=10)
    return r.status_code, r.text

def odoo_borrar(num_socio):
    params = {"data": json.dumps({"num_socio": num_socio})}
    r = requests.delete(API, params=params, timeout=10)
    return r.status_code, r.text

async def lector(update: Update, context: ContextTypes.DEFAULT_TYPE):
    texto = (update.message.text or "").strip()
    accion, datos = comando(texto)

    if accion is None:
        await update.message.reply_text("Orden no soportada")
        return

    if isinstance(datos, dict) and datos.get("error"):
        await update.message.reply_text(f"Error: {datos['error']}")
        return

    try:
        if accion == "crear":
            if not all(k in datos for k in ("num_socio", "nombre", "apellidos")):
                await update.message.reply_text(
                    'Faltan datos. Ej: Crear, nombre="A",apellidos="B", num_socio="1"'
                )
                return
            code, body = odoo_crear(datos)
            await update.message.reply_text(f"HTTP {code}\n{body}")

        elif accion == "modificar":
            if not all(k in datos for k in ("num_socio", "nombre", "apellidos")):
                await update.message.reply_text(
                    'Faltan datos. Ej: Modificar, nombre="A",apellidos="B", num_socio="1"'
                )
                return
            code, body = odoo_modificar(datos)
            await update.message.reply_text(f"HTTP {code}\n{body}")

        elif accion == "consultar":
            if "num_socio" not in datos:
                await update.message.reply_text('Faltan datos. Ej: Consultar, num_socio="1"')
                return
            code, body = odoo_consultar(datos["num_socio"])
            await update.message.reply_text(f"HTTP {code}\n{body}")

        elif accion == "borrar":
            if "num_socio" not in datos:
                await update.message.reply_text('Faltan datos. Ej: Borrar, num_socio="1"')
                return
            code, body = odoo_borrar(datos["num_socio"])
            await update.message.reply_text(f"HTTP {code}\n{body}")

        else:
            await update.message.reply_text("Orden no soportada")

    except requests.exceptions.RequestException as e:
        await update.message.reply_text(f"Error llamando a Odoo: {e}")
```

Declaro/defino todas los posibles funciones y defino lo que espera + si debe llevar body para la llamada o no.

Función de lector, lo que se ejecuta cada vez que alguien interactua con el bot. Esta función es async por que la librería de telgram funciona en modo asíncrono. A continuación describo lo que hace el código en orden:

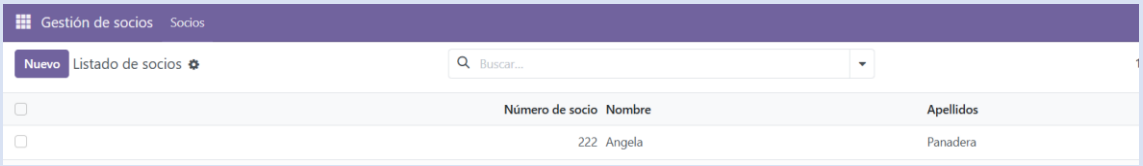
- Update.message.text es el texto real que mandaron y llama a la función de lector para parsearlo.
- Si no entiende la orden, pasa, "Orden no soportada".
- Si hubo error de parseo, devuelve el literal del error.
- Para decidir que operación hacer, existe un bloque try que lee le primer comando que le pasan.
- Hago pequeñas validaciones de campos como mínimos que debes pasar.

Comprobación del correcto funcionamiento con el bot.

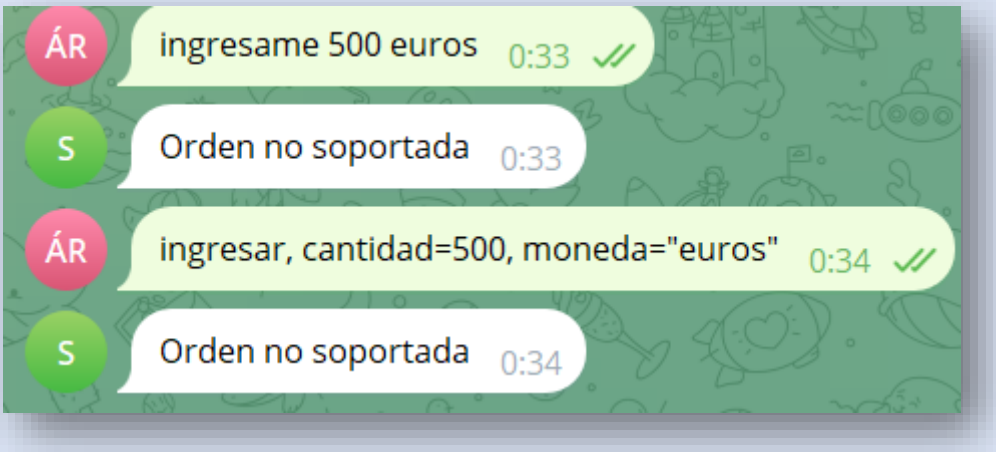


Llamadas y como le pasamos los datos. Nos devuelve igual que en postman.

Usuario creado y modificado:



Lamentablemente, si le pido otras cosas, le da igual:



Punto terminado, hacemos commit y pasamos a lo siguiente.

5. Actividad 04 – Generación de imágenes aleatorias con Web Controller

Para esta actividad, voy a aprovechar en su totalidad la estructura entera del módulo 09, voy a crear un solo archivo a mayores dentro de la carpeta de controladores que en lugar de generar un código de barras, generar una imagen totalmente random.

Lo primero que he hecho ha sido instalar la libreri ade PIL dentro del contenedor ejecutando:

docker compose exec web bash -lc "pip3 install Pillow"

Ya estaba instalado.

Tras esto, procedemos a crear en el módulo de generación de códigos de barras, dentro de la carpeta de controllers un nuevo archivo controllers/imagen_aleatoria.py que, en mi caso luce así:

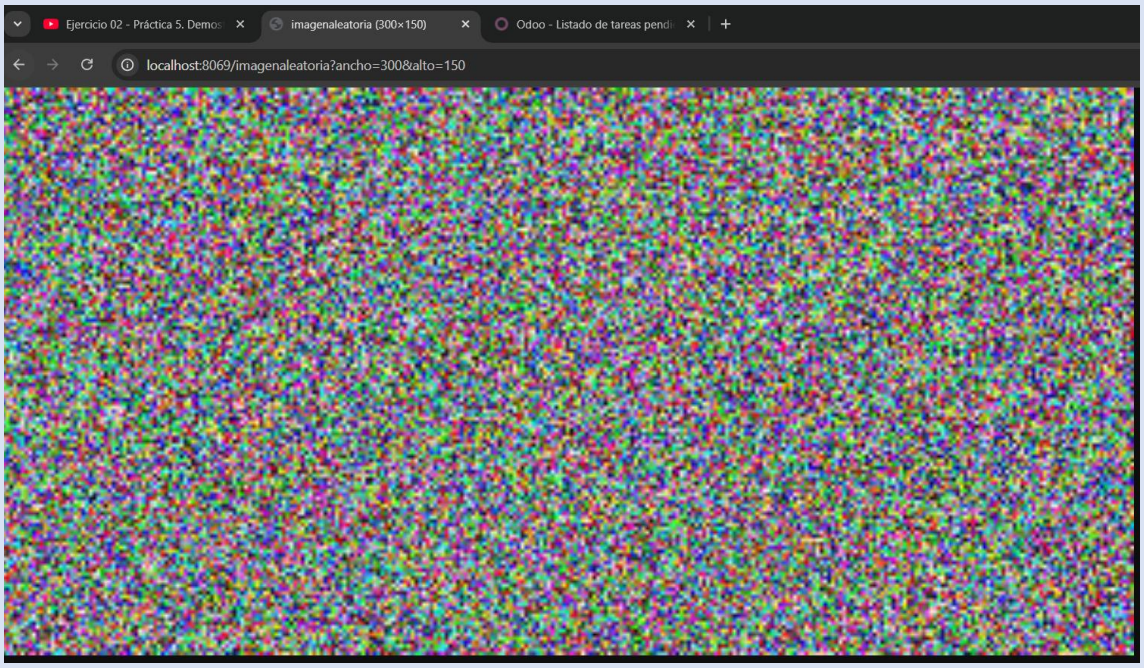
```python
from odoo import http
from odoo.http import request

import base64
from io import BytesIO
import os

from PIL import Image  # Pillow
# /imagenaleatoria?ancho=200&alto=100
# /imagenaleatoria?ancho=200&alto=100&modo=base64

class ImagenAleatoria(http.Controller):

    @http.route('/imagenaleatoria', auth='public', cors='*', type='http')
    def imagen_aleatoria(self, **kw):

        try:
            ancho = int(kw.get("ancho", 200))
            alto = int(kw.get("alto", 200))
        except ValueError:
            return http.Response("Parámetros ancho/alto inválidos", status=400)

        if ancho < 1 or alto < 1 or ancho > 2000 or alto > 2000:
            return http.Response("tamaño hasta 200", status=400)

        data = os.urandom(ancho * alto * 3)
        img = Image.frombytes("RGB", (ancho, alto), data)

        fp = BytesIO()
        img.save(fp, format="PNG")
        png_bytes = fp.getvalue()

        modo = (kw.get("modo", "png") or "").lower()

        if modo == "base64":
            img_str = base64.b64encode(png_bytes).decode("utf-8")
            html = f'<div><img src="{img_str}"/></div>'
            return html

        return request.make_response(
            png_bytes,
            headers=[("Content-Type", "image/png")]
        )

```

- Lee los parámetros de la query que le pasamos buscando el alto y el ancho
- No interpretará nunca un tamaño fuera de rango (hasta 200)
- Por cada byte de un pixel genera un patron con los 3 colores de forma aleatoria. Por lo que ví, por cada pixel hay 3 bytes.
- Lo pasa a png
- Lo devuelve al usuario interpretando el binario.
- He comentado las rutas que debería ejecutar.

Dicho esto, le decimos al init.py que debe cargar también nuestro archivo:

```
1  # -*- coding: utf-8 -*-
2  from . import generarbarcode
3  from . import imagen_aleatoria
```

Y lanzamos la ruta en el navegador:



Se ha generado una imagen de 300x150 totalmente aleatoria.

Tras esto, he finalizado la práctica. Aún que en mi opinión no haya sido de las más complejas, me ha llevado mucho tiempo debido al error de la actividad 02 que por cabezonería traté de solucionar.

Ángel Panadero Rodríguez.