Programación de Servicios y Procesos: Práctica de Docker Local

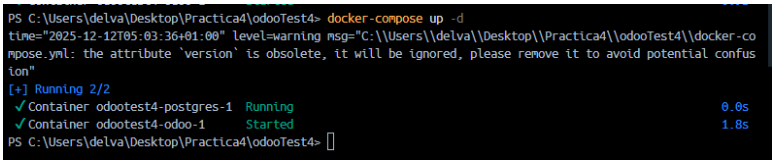
Álvaro Del Valle Fernández

February 2, 2026

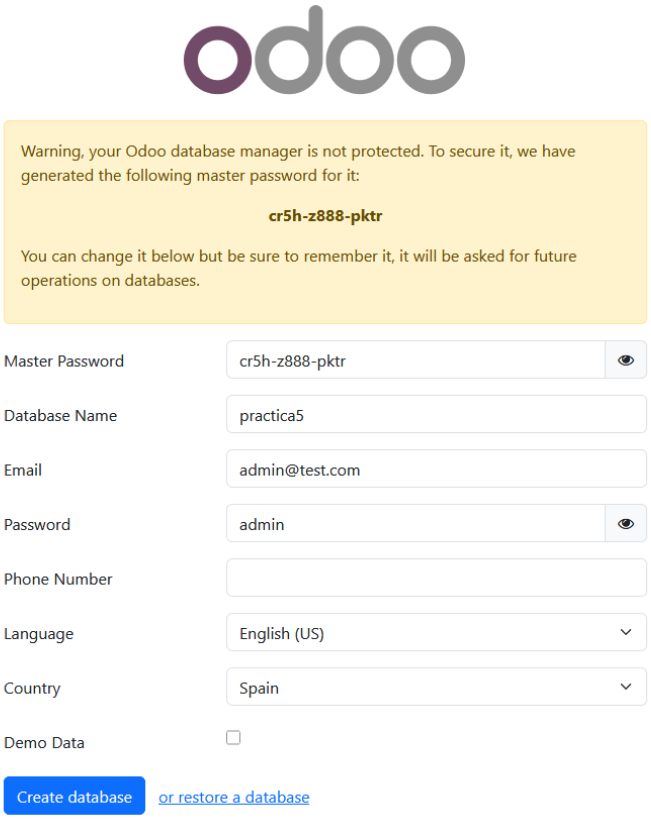
1 Introducción

En esta práctica completaré 4 ejercicios diferentes en Odoo usando como referencia los archivos del repositorio de git entregado.
Antes de comenzar con los ejercicios creare un odoo completamente nuevo en la version 18, realizando mas ajustes para no tener problemas constantes de nuevo.

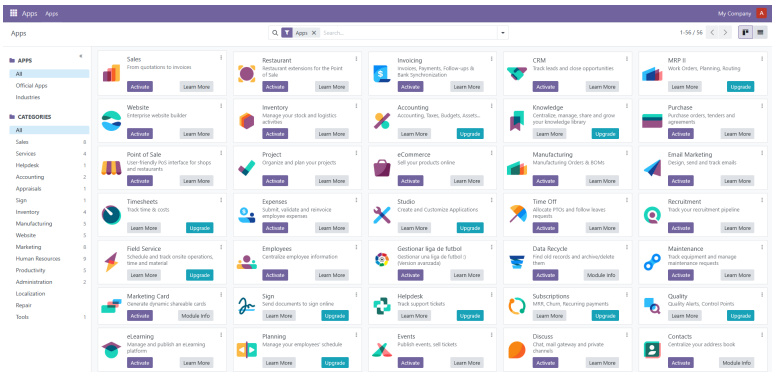
```
docker-compose up -d
```



Credenciales dentro de Odoo:



cr5h-z888-pktr



Cada vez que tengo que probar los cambios realizados, debo usar el comando en docker:

```
docker-compose restart odoo
docker-compose up -d
```

Cuando tengo algun error total de odoo uso este comando, el cual borra todos los volumenes y me permite crearlos de nuevo cuando se corrompen, cosa que es común al realizar cambios relacionados con manejo de datos.

```
docker volume prune -f
```

2 Actividad 01

2.1 Objetivo

Para realizar este ejercicio debo estudiar el modulo LigaFutbol y luego incluir las funciones de reglas, botones para goles de partidos, webcontroller para eliminar empates, crear inforemes PDF, Wizard para nuevos partidos y vista Graph.

2.2 Implementación

Usando de referencia base el archivo de EJ07 LigaFutbol, la estudiaremos y realizaremos los cambios necesarios.
Como es normal Odoo no funciona correctamente debido al tree en vez de list, por lo que modifico todos los tree de las views.
Tras esto analizo el primer punto.

2.3 01.1

Primero necesito modificar la puntuación de partidos.

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

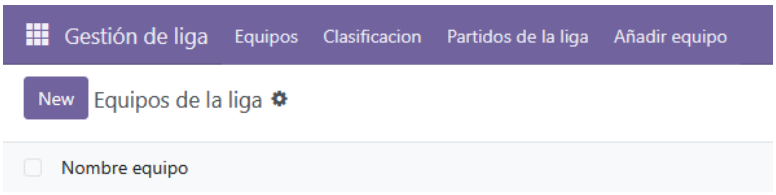
Modifico el metodo para sumar 4 putnos al ganador y -1 al perdedor.

```
    def actualizar_clasificacion(self):
        """
        Funcion para actualizar la clasificacion de todos los equipos
        """
#reinicio
        equipos = self.env['liga.equipo'].search([])
        for equipo in equipos:
            equipo.write({
                'victorias': 0,
                'empates': 0,
                'derrotas': 0,
                'goles_a_favor': 0,
                'goles_en_contra': 0,
                'puntos': 0,
            })

        partidos = self.env['liga.partido'].search([])
        for partido in partidos:
            goles_casa = partido.goles_casa or 0
            goles_fuera = partido.goles_fuera or 0
            diferencia = abs(goles_casa - goles_fuera)

 #Actualizar equipo local
            if partido.equipo_casa:
                equipo_casa = partido.equipo_casa
                nuevo_puntos_casa = equipo_casa.puntos
                nueva_victorias_casa = equipo_casa.victorias
                nueva_empates_casa = equipo_casa.empates
                nueva_derrotas_casa = equipo_casa.derrotas
                nuevo_goles_favor_casa = equipo_casa.goles_a_favor + goles_casa
                nuevo_goles_contra_casa = equipo_casa.goles_en_contra + goles_fu

                if goles_casa > goles_fuera:
                    nueva_victorias_casa += 1
                    if diferencia >= 4:
                        nuevo_puntos_casa += 4
                    else:
                        nuevo_puntos_casa += 3
                elif goles_casa < goles_fuera:
                    nueva_derrotas_casa += 1
                    if diferencia >= 4:
                        nuevo_puntos_casa -= 1
```

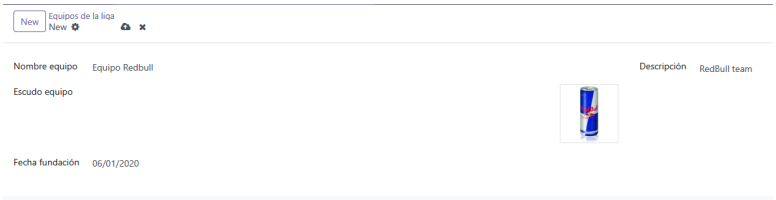
```
                else:
                    nueva_empates_casa += 1
                    nuevo_puntos_casa += 1

                equipo_casa.write({
                    'victorias': nueva_victorias_casa,
                    'empates': nueva_empates_casa,
                    'derrotas': nueva_derrotas_casa,
                    'goles_a_favor': nuevo_goles_favor_casa,
                    'goles_en_contra': nuevo_goles_contra_casa,
                    'puntos': nuevo_puntos_casa,
                })

#visitante
            if partido.equipo_fuera:
                equipo_fuera = partido.equipo_fuera
                nuevo_puntos_fuera = equipo_fuera.puntos
                nueva_victorias_fuera = equipo_fuera.victorias
                nueva_empates_fuera = equipo_fuera.empates
                nueva_derrotas_fuera = equipo_fuera.derrotas
                nuevo_goles_favor_fuera = equipo_fuera.goles_a_favor + goles_fue
                nuevo_goles_contra_fuera = equipo_fuera.goles_en_contra + goles_

                if goles_fuera > goles_casa:
                    nueva_victorias_fuera += 1
                    if diferencia >= 4:
                        nuevo_puntos_fuera += 4
                    else:
                        nuevo_puntos_fuera += 3
                elif goles_fuera < goles_casa:
                    nueva_derrotas_fuera += 1
                    if diferencia >= 4:
                        nuevo_puntos_fuera -= 1
                else:
                    nueva_empates_fuera += 1
                    nuevo_puntos_fuera += 1

                equipo_fuera.write({
                    'victorias': nueva_victorias_fuera,
                    'empates': nueva_empates_fuera,
                    'derrotas': nueva_derrotas_fuera,
                    'goles_a_favor': nuevo_goles_favor_fuera,
                    'goles_en_contra': nuevo_goles_contra_fuera,
                    'puntos': nuevo_puntos_fuera,
                })
```

```
        return True
```

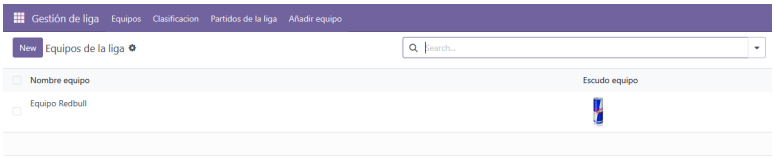
Ahora probare las modificaciones. Creo un equipo.



Equipo de prueba creado, utilizando todas sus opciones.

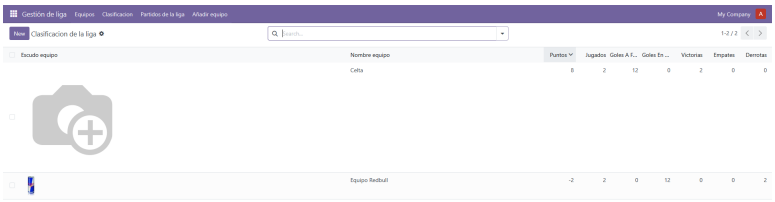


Aparece en equipos.



Creamos un partido de prueba

Tras los cambios realizados, el equipo obtiene los puntos correctamente.



Numerosos problemas con la actualizacion de los datos, principalmente por odoo, al no saber si se implementaron los cambios, si dieron error o si necesitaba añadir otro partido para que actualice los cambios anteriores.

2.4 01.2

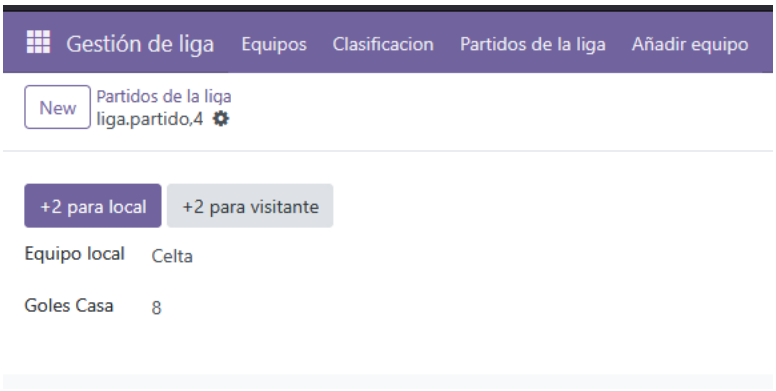
Ahora debo modificar los botones de partidos.
Por lo que deberia de modificar liga partido.xml para añadir botones.
Debo editar tambien la view para añadir el button.
Test de los botones en models.

```
def gollocal(self):
    for partido in self.search([]):
        partido.goles_casa += 2
    self.actualizoRegistrosEquipo()

def golvisit(self):
    for partido in self.search([]):
        partido.goles_fuera += 2
    self.actualizoRegistrosEquipo()
```

Views:

```
                <header>
                <button name="gollocal"
                        type="object"
                        string="+2 para local"
                        class="btn-primary"/>

                <button name="golvisit"
                        type="object"
                        string="+2 para visitante"
                        class="btn-secondary"/>
            </header>
```

Tras probar los botones, funcionan correctamente.

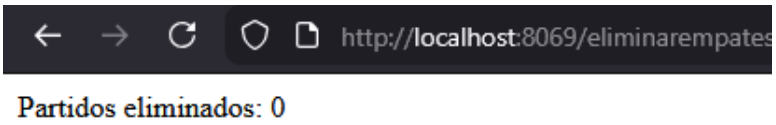


2.5 01.3

Debo eliminar empates.
Necesito crear un webcontroller nuevo por lo que creo este nuevo py.

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

class LigaController(http.Controller):
#link correcto  http://localhost:8069/eliminarempates
    @http.route('/eliminarempates', auth='public', type='http')
    def eliminar_empates(self):
            Partido = request.env['liga.partido'].sudo()
            partidos = Partido.search([])
            empates = partidos.filtered(lambda p: p.goles_casa == p.goles_fuera)
            total = len(empates)
            empates.unlink()
            return f"Partidos eliminados: {total}"
```

Ahora debo añadirlo al init.py (sin guion bajo en latex, si no explota)
El resultado es correcto al acceder a la url.



Me encontré con multiples errores 404 y 500 al intentar acceder a la url debido a problemas de odoo.

2.6 01.4

Crear informe en pdf.
Como podemos ver en la carpeta report, el xml ya cuenta con parte de los elementos para crear un informe en pdf.

```
<!-- Realmente, este es el informe, lo de arriba es la plantilla que utiliza
<report id="report_clasificacion" model="liga.equipo" string="Informe clasifi
```

Odoo deberia de crear automaticamente el informe al tener la plantilla y el informe definidos, pero como suele pasar Odoo no lo genera correctamente, al no mostrar la opcion de imprimir que deberia de aparecer de base
El py:

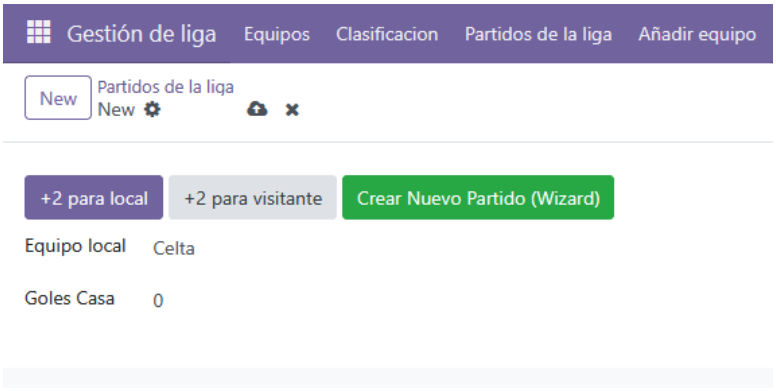
```
class LigaPartidoWizard(models.TransientModel):
_name = 'liga.partido.wizard'

equipo_casa = fields.Many2one('liga.equipo', required=True)
equipo_fuera = fields.Many2one('liga.equipo', required=True)
goles_casa = fields.Integer()
goles_fuera = fields.Integer()

def crear_partido(self):
    self.env['liga.partido'].create({
        'equipo_casa': self.equipo_casa.id,
        'equipo_fuera': self.equipo_fuera.id,
        'goles_casa': self.goles_casa,
        'goles_fuera': self.goles_fuera,
    })
```

El xml:

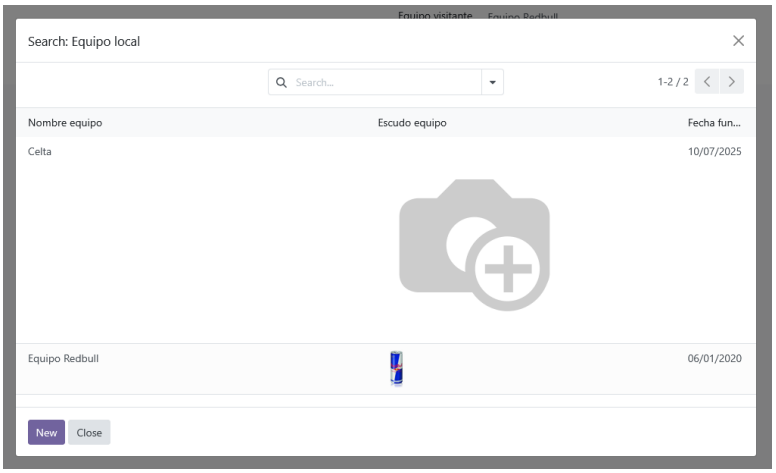
```
<form string="Nuevo Partido">
<group>
    <field name="equipo_casa"/>
    <field name="goles_casa"/>
    <field name="equipo_fuera"/>
    <field name="goles_fuera"/>
</group>
<footer>
    <button string="Crear" type="object" name="crear_partido" class="btn-pri
    <button string="Cancelar" special="cancel"/>
</footer>
</form>
```

2.7 01.5

Crear nuevos partidos mediante wizard.
Tenemos el docuemnto base wizard ligapartidowizard.py de referencia, ahora lo adaptamos a partidos.



Cree un boton debido a que odoo no queria mostrar el wizard en la vista de partidos automaticamente.
Este te permite crear nuevos partidos seleccionando ambos.



2.8 01.6

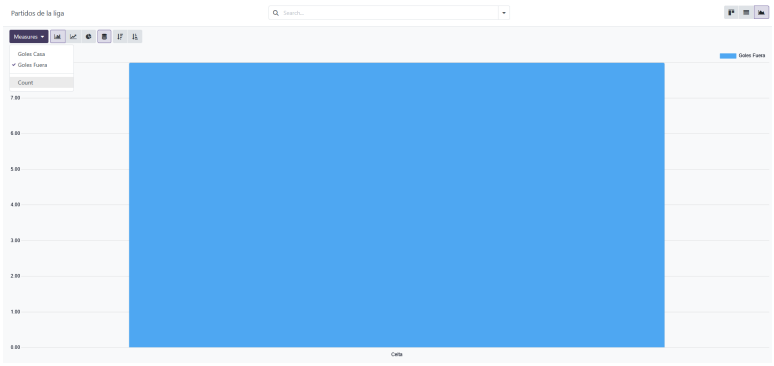
Por ultimo gráfica de estadisticas.
Esta consiste en generear una gráfica, ya contamos con un elemento de grafica en ligaequipo.xml, por lo que debemos adaptarlo sin mas.

```
    <record model="ir.ui.view" id="liga_equipo_view_graph">
    <field name="name">Puntos por equipo</field>
    <field name="model">liga.equipo</field>
    <field name="type">graph</field>
    <field name="arch" type="xml">
          Indicamos que cada fila es un equipo y que la medida que los valora
             Con esto conseguimos, por ejemplo, en un pie chart, que cada que
           <graph string="Puntos por equipo">
             <field name="nombre" group="True" type="row"/>
             <field name="puntos" group="True" type="measure"/>
          </graph>
      </field>
  </record>
```

En partidos:

```
  <record id="view_liga_partido_graph" model="ir.ui.view">
    <field name="name">liga.partido.graph</field>
    <field name="model">liga.partido</field>
    <field name="arch" type="xml">
        <graph string="Goles de equipos locales" type="bar">
            <field name="equipo_casa" type="row"/>

            <field name="goles_casa" type="measure"/>
        </graph>
    </field>
  </record>
```

Tras completar el resto de ajustes tal y como la otra visa, podemos ver las graficas en partidos con los datos actuales.



3 Actividad 02

3.1 Objetivo

En este ejercicio realizaré un video explicando el funcionamiento del modulo api rest socios, teniendo que usar las opciones crear, modificar,consulatar y eliminar.

Instalo Thunder CLient en Visual Studio, herramienta centrada en probar apis de forma sencilla

```
http://localhost:8069/
```

Crear: POST

```
http://localhost:8069/gestion/apirest/socio?data={"num_socio": 110, "nombre"
```

Consultar: get

```
http://localhost:8069/gestion/apirest/socio?data={"num_socio": 110 }
```

Modificar: put

```
http://localhost:8069/gestion/apirest/socio?data={"num_socio": 110, "nombre"
```

Consultar todo: get

```
http://localhost:8069/gestion/socio
```

Borrar: Delete

```
http://localhost:8069/gestion/apirest/socio?data={"num_socio": 110}
```

Link al video:
`https://youtu.be/VpNurlFFFbw`

4 Actividad 03

4.1 Objetivo

Crear un bot de telegram que realize ciertas ordenes mediante comandos.

4.2 Implementación

Primero analizo el codigo, veo que apirest es el controlador para los post,put,get y delete.
Listasocios.py se centra en devolver el json con los datos.
Socio.py es el modelo de datos.
Ahora debo crear el bot en telegram.
Documento telegramsociosbot. py:

```
    import json
import re
import requests
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, filters, C

TOKEN_TELEGRAM = "8268402944:AAHcpcgy6tD9enSmWuZ2H7gMim-8Yxxe7f8"
URL_API = "http://localhost:8069/gestion/apirest/socio"

def extraer_parametros(texto):
    resultado = {}
    patron = r'(\w+)="([^"]*)"'
    coincidencias = re.findall(patron, texto)

    for clave, valor in coincidencias:
        if clave == 'num_socio' and valor.isdigit():
            resultado[clave] = int(valor)
        else:
            resultado[clave] = valor

    return resultado

#/start
async def start(update: Update, context: ContextTypes.DEFAULT_TYPE):
    mensaje = (
        "COMANDOS DISPONIBLES:\n\n"
        "/crear nombre=\"nombre\",apellidos=\"apellidos\",num_socio=\"numero\"\n
        "/modificar nombre=\"nombre\",apellidos=\"apellidos\",num_socio=\"numero
        "/consultar num_socio=\"numero\"\n"
        "/borrar num_socio=\"numero\"\n\n"
        "EJEMPLOS:\n"
        "/crear nombre=\"Juan\",apellidos=\"Perez\",num_socio=\"101\"\n"
        "/consultar num_socio=\"101\""
    )
    await update.message.reply_text(mensaje)

#/crear
async def crear(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not context.args:
        await update.message.reply_text("Uso: /crear nombre=\"nombre\",apellidos
        return

    texto = ' '.join(context.args)
    params = extraer_parametros(texto)

    if not all(k in params for k in ['nombre', 'apellidos', 'num_socio']):
```

```
        await update.message.reply_text("Faltan parametros: nombre, apellidos, n
        return

    datos = {
        "num_socio": params['num_socio'],
        "nombre": params['nombre'],
        "apellidos": params['apellidos']
    }

    try:
        respuesta = requests.post(URL_API, json=datos, headers={'Content-Type':

        if respuesta.status_code == 200:
            socio = respuesta.json()[0]
            mensaje = f"Socio creado:\nNumero: {socio.get('num_socio')}\nNombre:
        else:
            mensaje = f"Error: {respuesta.status_code}"

    except Exception as e:
        mensaje = f"Error de conexion: {str(e)}"

    await update.message.reply_text(mensaje)

#/modificar
async def modificar(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not context.args:
        await update.message.reply_text("Uso: /modificar nombre=\"nombre\",apell
        return

    texto = ' '.join(context.args)
    params = extraer_parametros(texto)

    if 'num_socio' not in params:
        await update.message.reply_text("Falta parametro: num_socio")
        return

    datos = {
        "num_socio": params['num_socio'],
        "nombre": params.get('nombre', ''),
        "apellidos": params.get('apellidos', '')
    }

    try:
        respuesta = requests.put(URL_API, json=datos, headers={'Content-Type': '

        if respuesta.status_code == 200:
```

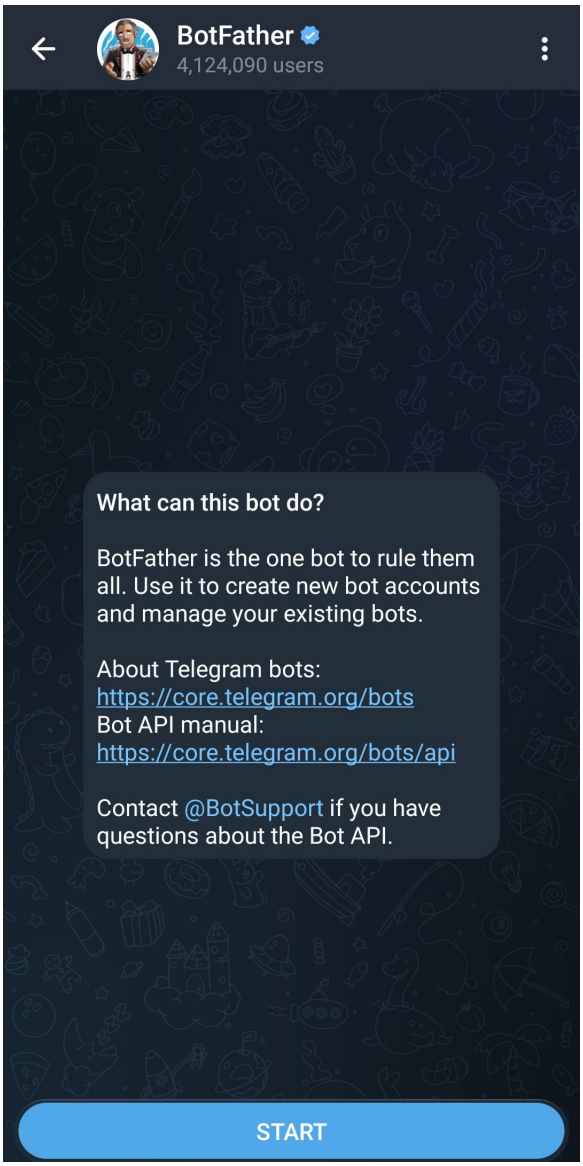
```
            socio = respuesta.json()[0]
            mensaje = f"Socio modificado:\nNumero: {socio.get('num_socio')}\nNom
        elif respuesta.status_code == 404:
            mensaje = f"Socio {params['num_socio']} no encontrado"
        else:
            mensaje = f"Error: {respuesta.status_code}"

    except Exception as e:
        mensaje = f"Error de conexion: {str(e)}"

    await update.message.reply_text(mensaje)

#/consultar
async def consultar(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not context.args:
        await update.message.reply_text("Uso: /consultar num_socio=\"numero\"")
        return

    texto = ' '.join(context.args)
    params = extraer_parametros(texto)

    if 'num_socio' not in params:
        await update.message.reply_text("Falta parametro: num_socio")
        return

    try:
        parametros_url = {'data': json.dumps({"num_socio": params['num_socio']})
        respuesta = requests.get(URL_API, params=parametros_url)

        if respuesta.status_code == 200:
            socio = respuesta.json()
            mensaje = f"Socio encontrado:\nNumero: {socio.get('num_socio')}\nNom
        elif respuesta.status_code == 404:
            mensaje = f"Socio {params['num_socio']} no encontrado"
        else:
            mensaje = f"Error: {respuesta.status_code}"

    except Exception as e:
        mensaje = f"Error de conexion: {str(e)}"

    await update.message.reply_text(mensaje)

#/borrar
async def borrar(update: Update, context: ContextTypes.DEFAULT_TYPE):
    if not context.args:
        await update.message.reply_text("Uso: /borrar num_socio=\"numero\"")
```

```
        return

    texto = ' '.join(context.args)
    params = extraer_parametros(texto)

    if 'num_socio' not in params:
        await update.message.reply_text("Falta parametro: num_socio")
        return

    try:
        parametros_url = {'data': json.dumps({"num_socio": params['num_socio']})
        respuesta = requests.delete(URL_API, params=parametros_url)

        if respuesta.status_code == 200:
            socio = respuesta.json()[0]
            mensaje = f"Socio eliminado:\nNumero: {socio.get('num_socio')}\nNomb
        elif respuesta.status_code == 404:
            mensaje = f"Socio {params['num_socio']} no encontrado"
        else:
            mensaje = f"Error: {respuesta.status_code}"

    except Exception as e:
        mensaje = f"Error de conexion: {str(e)}"

    await update.message.reply_text(mensaje)

async def mensaje_no_soportado(update: Update, context: ContextTypes.DEFAULT_TYP
    if update.message.text.startswith('/'):
        await update.message.reply_text("Orden no soportada")
    else:
        await update.message.reply_text("Envia un comando. Usa /start para ayuda

def main():
    print("Iniciando bot...")

    app = Application.builder().token(TOKEN_TELEGRAM).build()
    app.add_handler(CommandHandler("start", start))
    app.add_handler(CommandHandler("crear", crear))
    app.add_handler(CommandHandler("modificar", modificar))
    app.add_handler(CommandHandler("consultar", consultar))
    app.add_handler(CommandHandler("borrar", borrar))

    app.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, mensaje_no_s

    print("Bot listo. Presiona Ctrl+C para detener.")
    app.run_polling(allowed_updates=Update.ALL_TYPES)
```

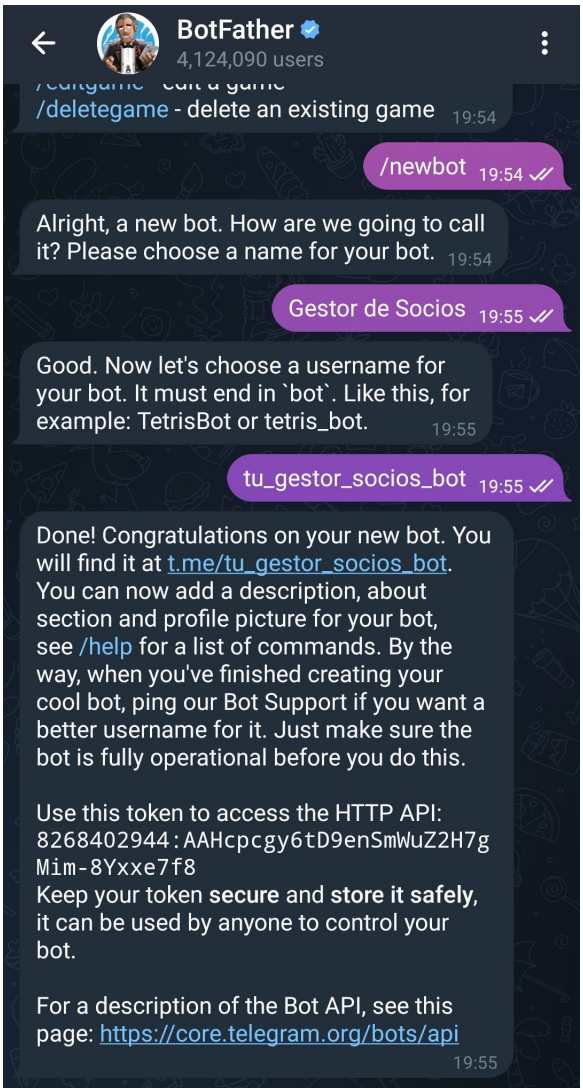
```
if __name__ == '__main__':
    main()
```

Busco el botfather en telegram y creo un nuevo bot.

Comienzo y creo un nuevo bot.



Tras configurarlo, ya esta creado correctamente.
Ahora debo obtener el token para usarlo en el codigo.



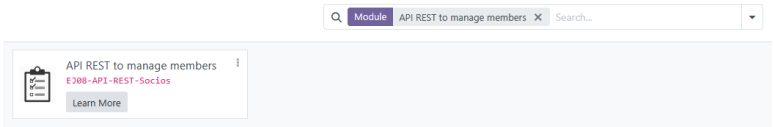
Tras iniciar el bot en telegram y obtener el token, ejecuto el comando para isntalar las dependencias.

```
PS C:\Users\delva\Desktop\Practica5\addons\EJ08-API-REST-Socios> python -m pip install python-telegram-bot==20.7 request
s
Collecting python-telegram-bot==20.7
  Downloading python_telegram_bot-20.7-py3-none-any.whl.metadata (15 kB)
Collecting requests
  Downloading requests-2.32.5-py3-none-any.whl.metadata (4.9 kB)
Collecting httpx~=0.25.2 (from python-telegram-bot==20.7)
  Downloading httpx-0.25.2-py3-none-any.whl.metadata (6.9 kB)
Collecting anyio (from httpx~=0.25.2->python-telegram-bot==20.7)
  Downloading anyio-4.12.1-py3-none-any.whl.metadata (4.3 kB)
Collecting certifi (from httpx~=0.25.2->python-telegram-bot==20.7)
  Downloading certifi-2026.1.4-py3-none-any.whl.metadata (2.5 kB)
Collecting httpcore==1.* (from httpx~=0.25.2->python-telegram-bot==20.7)
  Downloading httpcore-1.0.9-py3-none-any.whl.metadata (21 kB)
Collecting idna (from httpx~=0.25.2->python-telegram-bot==20.7)
  Downloading idna-3.11-py3-none-any.whl.metadata (8.4 kB)
Collecting sniffio (from httpx~=0.25.2->python-telegram-bot==20.7)
  Downloading sniffio-1.3.1-py3-none-any.whl.metadata (3.9 kB)
Collecting h11>=0.16 (from httpcore==1.*->httpx~=0.25.2->python-telegram-bot==20.7)
  Downloading h11-0.16.0-py3-none-any.whl.metadata (8.3 kB)
Collecting charset_normalizer<4,>=2 (from requests)
  Downloading charset_normalizer-3.4.4-cp314-cp314-win_amd64.whl.metadata (38 kB)
```

Ahora creo el neuvo archivo para el bot. fundamental añadir el token.

```
import json
import re
import requests
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, filters, C

TOKEN_TELEGRAM = "8268402944:AAHcpcgy6tD9enSmWuZ2H7gMim-8Yxxe7f8"
URL_API = "http://localhost:8069/gestion/apirest/socio"
```

Istalar en odoo:



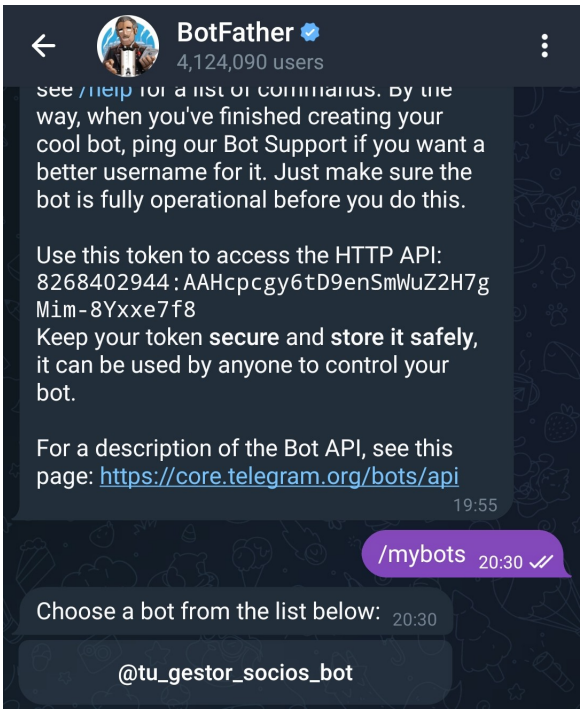
Error al añadir el bot, version no adecuada.

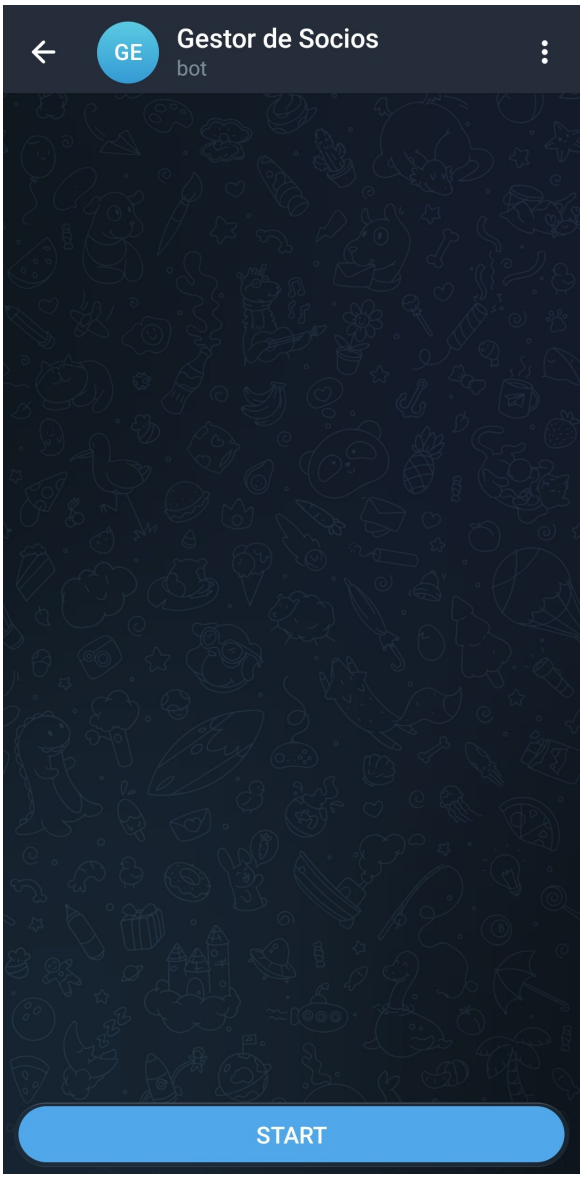
```
python -m pip uninstall python-telegram-bot
python -m pip install python-telegram-bot
python telegram_socios_bot.py
```

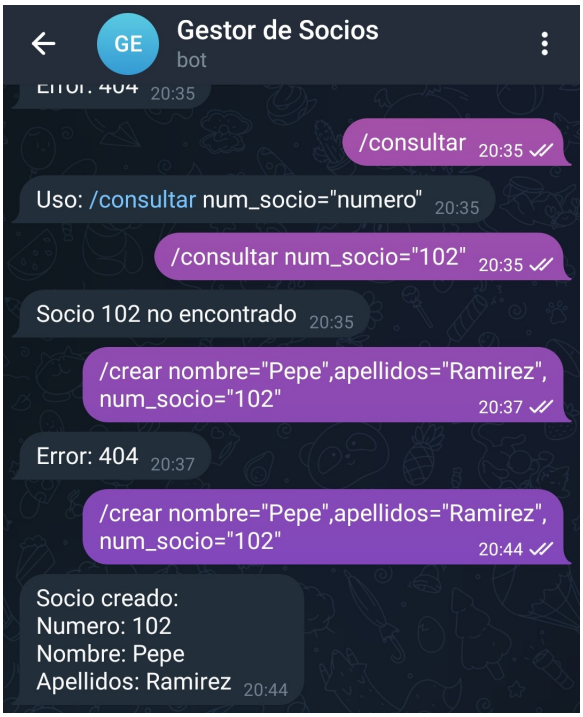
Tras esto:

```
[notice] To update, run: C:\Users\delva\AppData\Local\Python\pythoncore-3.14-64\python.exe -m pip install --upgrade pip
PS C:\Users\delva\Desktop\Practica5\addons\EJ08-API-REST-Socios> python telegram_socios_bot.py
Iniciando bot...
Bot listo. Presiona Ctrl+C para detener.
```

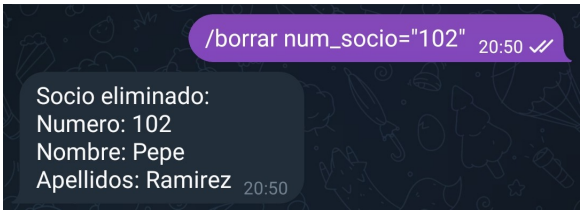
Ahora tras numerosos cambios:


BotFather
4,124,090 users
see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
Use this token to access the HTTP API:
8268402944:AAHcpcgy6tD9enSmWuZ2H7gMim-8Yxxe7f8
Keep your token secure and store it safely, it can be used by anyone to control your bot.
For a description of the Bot API, see this page: https://core.telegram.org/bots/api
19:55
/mybots 20:30
Choose a bot from the list below: 20:30
@tu_gestor_socios_bot


GE
Gestor de Socios
bot
START



Ya funciona correctamente permitiendo las funciones indicadas:



5 Actividad 04

5.1 Objetivo

Segun el ejemplo GenerarBarcode, crear un webcontroller que genere una imagen qr aleatoria.

5.2 Implementación

Para ello necesito crear un nuevo controller usando los elementos del ejemplo dado, este genera el elemento random al añadirle los parametros a la url.

```
    # -*- coding: utf-8 -*-
import base64
import io
import random
from PIL import Image
from odoo import http
from odoo.http import request, content_disposition

class RandomImageController(http.Controller):

    @http.route('/random_image', type='http', auth='public')
    def generate_random_image(self, width=100, height=100, **kwargs):
        """
        Genera una imagen con pixeles aleatorios
        Parametros:
        - width: ancho de la imagen (por defecto 100)
        - height: alto de la imagen (por defecto 100)

        Ejemplos:
        - http://localhost:8069/random_image?width=300&height=200
        - http://localhost:8069/random_image?width=500
        """
        try:
            width = int(width)
            height = int(height)
            if width <= 0 or height <= 0:
                return request.make_response(
                    'Error mas de 0.',
                    headers=[('Content-Type', 'text/plain')]
                )
            img = Image.new('RGB', (width, height), color='white')
            pixels = img.load()

            for i in range(width):
                for j in range(height):
                    red = random.randint(0, 255)
                    green = random.randint(0, 255)
                    blue = random.randint(0, 255)
                    pixels[i, j] = (red, green, blue)
            buffer = io.BytesIO()
            img.save(buffer, format='PNG')
            buffer.seek(0)

#PNG
            return request.make_response(
                buffer.getvalue(),
```

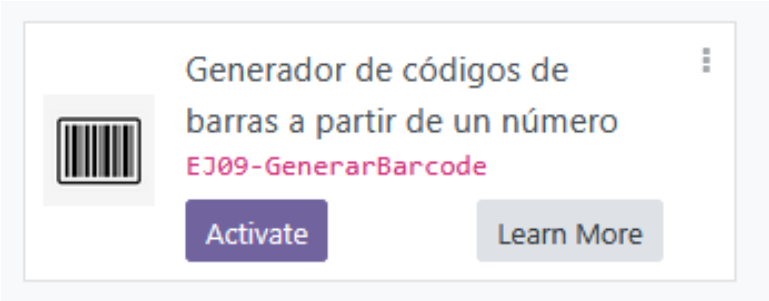
```
                headers=[
                    ('Content-Type', 'image/png'),
                    ('Content-Disposition', content_disposition('random_image.pn
                ]
            )

        except ValueError:
            return request.make_response(
                headers=[('Content-Type', 'text/plain')]
            )
        except Exception as e:
            return request.make_response(
                f'Error generando imagen: {str(e)}',
                headers=[('Content-Type', 'text/plain')]
            )

    @http.route('/random_image_base64', type='http', auth='public')
    def generate_random_image_base64(self, width=100, height=100, **kwargs):
        """
        - width: ancho de la imagen (por defecto 100)
        - height: alto de la imagen (por defecto 100)

        Ejemplo:
        - http://localhost:8069/random_image_base64?width=300&height=200
        """
        try:
            width = int(width)
            height = int(height)

            if width <= 0 or height <= 0:
                return request.make_response(
                    headers=[('Content-Type', 'application/json')]
                )

            img = Image.new('RGB', (width, height), color='white')
            pixels = img.load()

            for i in range(width):
                for j in range(height):
                    red = random.randint(0, 255)
                    green = random.randint(0, 255)
                    blue = random.randint(0, 255)
                    pixels[i, j] = (red, green, blue)
#64
            buffer = io.BytesIO()
            img.save(buffer, format='PNG')
```

```
            buffer.seek(0)
            img_base64 = base64.b64encode(buffer.getvalue()).decode('utf-8')

            return request.make_response(
                img_base64,
                headers=[('Content-Type', 'text/plain')]
            )

        except Exception as e:
            return request.make_response(
                f'{{"error": "{str(e)}"}}',
                headers=[('Content-Type', 'application/json')]
            )
```

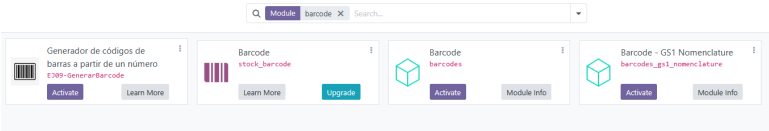
Instalar pillow acordarse

```
python -m pip install python-barcode Pillow
```

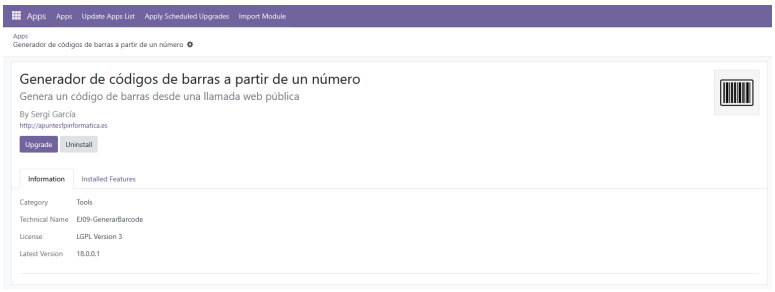
Instalar el modulo



Error hay que instalar los otros elementos del barcode antes:



Eliminar el elemento antiguo de ejemplo arrelgo el problema, al ejercutarlo claramente estaba abriendo el elemento antiguo de generador de codigo de barras en vez de la nueva aplicación



Usando la url ahora podemos crear una imagen aleatoria:

http://localhost:8069/random_image?width=400&height=300

6 Conclusión

Mi mayor problema al realizar estos ejercicios son los conflictos de versiones, teniendo problemas icluso lanzando alugnos de los modulos predeterminados.
Actualizar todo a la ultima version tambien me resulto inestable teniendo que usar versiones especificas que no tienen problemas con odoo 18.
Me encontre con numerosos 404 pero por suerte esta ver fueron mas intuitivos, pudiendo arreglarlos de forma mas sencilla.