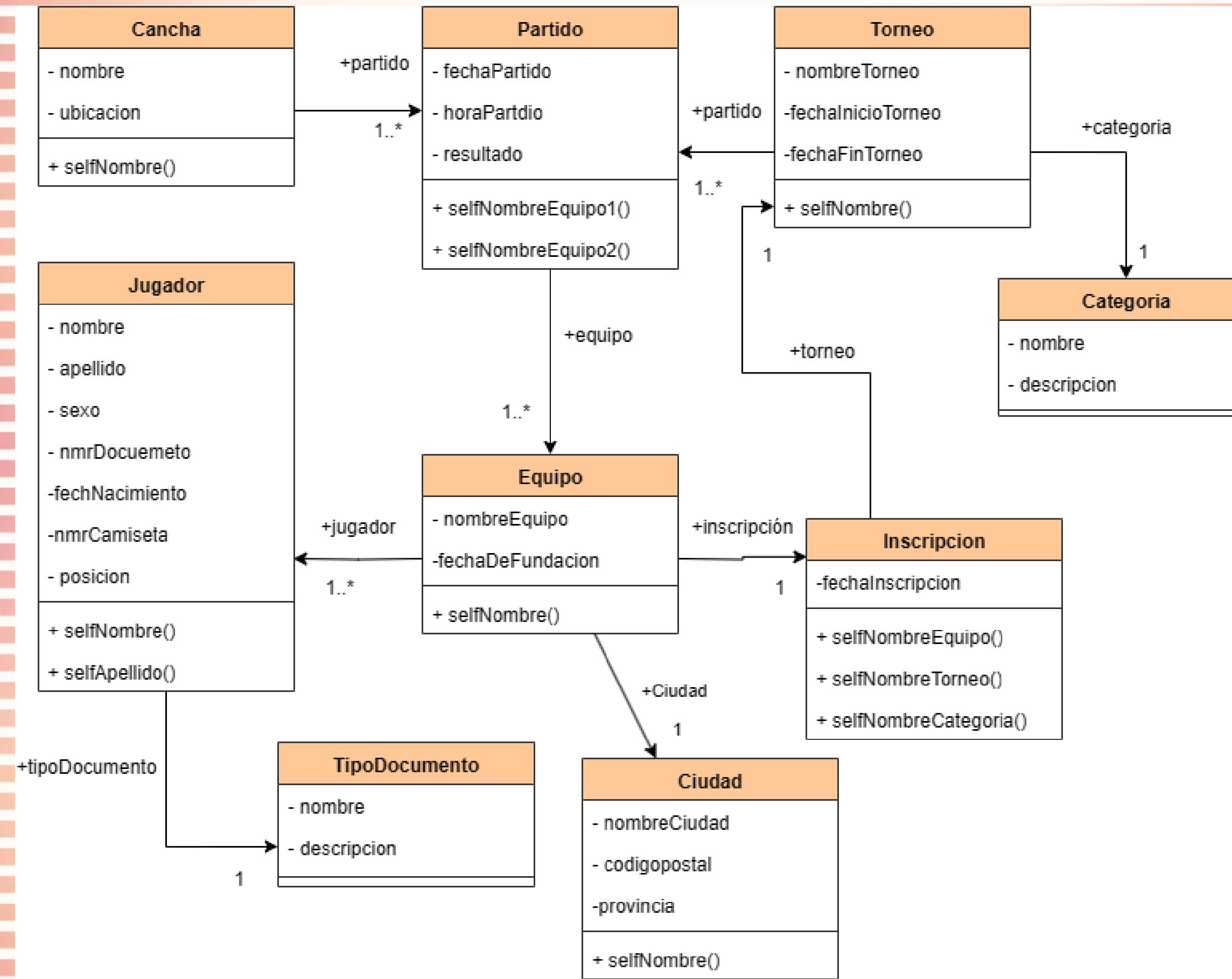


# Torneo de Fútbol

Grupo 3

# Diagrama de clases



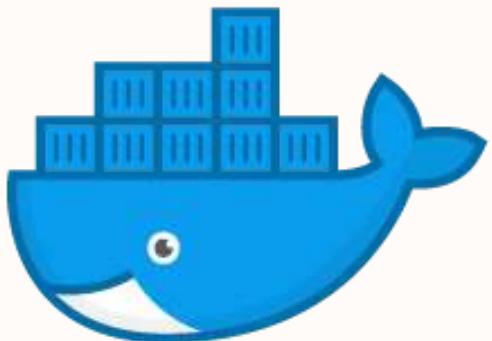
# Tecnologías Utilizadas

Relacional



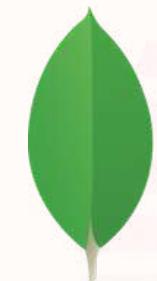
PostgreSQL

django



docker

Documental



mongoDB®

mongoengine



mongoDB Compass

# Django + PostgreSQL

Levantamos el proyecto mediante el archivo .init

user: admin  
password: admin



```
#!/bin/bash
#Ejecutar en la consola con ./init.ps1
#Es solo para Windows

#Baja contenedores, elimina volúmenes, imágenes huérfanas
docker compose down -v --remove-orphans --rmi all

#Crear migraciones y aplicarlas
docker compose run --rm manage makemigrations
docker compose run --rm manage migrate

#Levantar solo el backend en modo detached (background)
docker compose up backend -d

#Crear superusuario sin interacción
docker compose run --rm manage createsuperuser --noinput --username admin --email admin@example.com

#Establecer contraseña del superusuario (porque --noinput no la setea)
docker compose run --rm manage shell -c "from django.contrib.auth import get_user_model; User = get_user_model(); u=User.objects.get(username='admin'); u.set_password('admin'); u.save()"

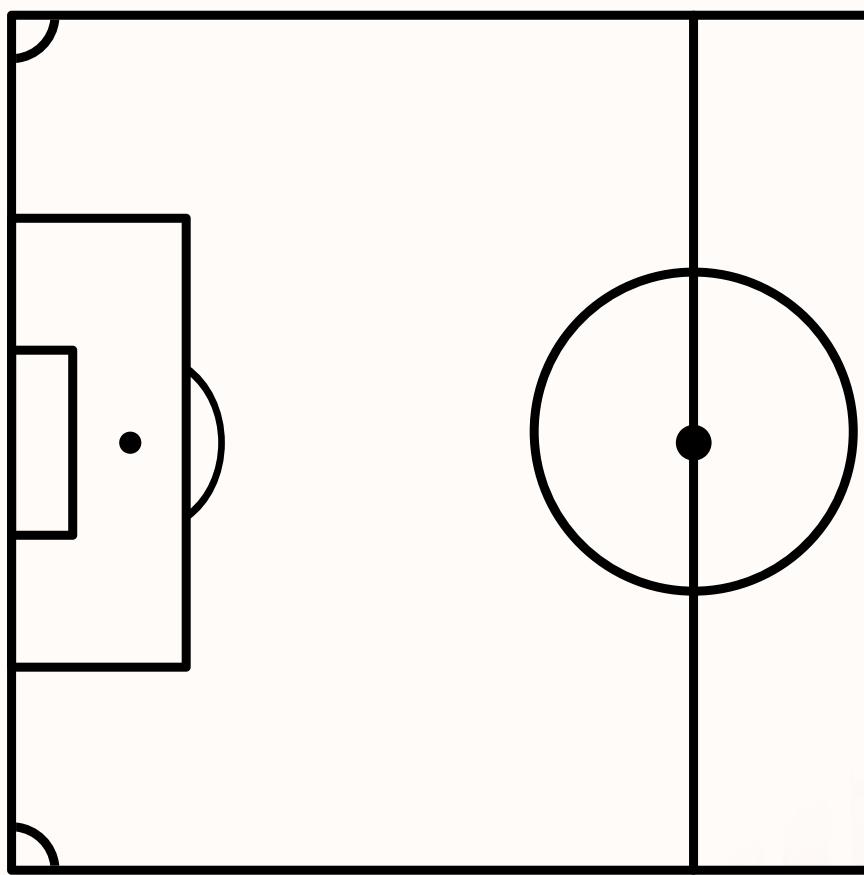
#Cargar datos iniciales (fixtures)
docker compose run --rm manage loaddata torneo/fixtures/initial_data.json
```

PS C:\Users\bianc\TORNEO-FUTBOL> .\init.ps1

La primera vez se puede hacer con el script de inicialización o comando por comando.

# .env.db

```
DATABASE_ENGINE=django.db.backends.postgresql
POSTGRES_HOST=db
POSTGRES_PORT=5432
POSTGRES_DB=torneo-futbol
POSTGRES_USER=postgres
PGUSER=${POSTGRES_USER}
POSTGRES_PASSWORD=postgres
LANG=es_AR.utf8
POSTGRES_INITDB_ARGS="--locale-provider=icu --icu-locale=es-AR --auth-local=trust"
```



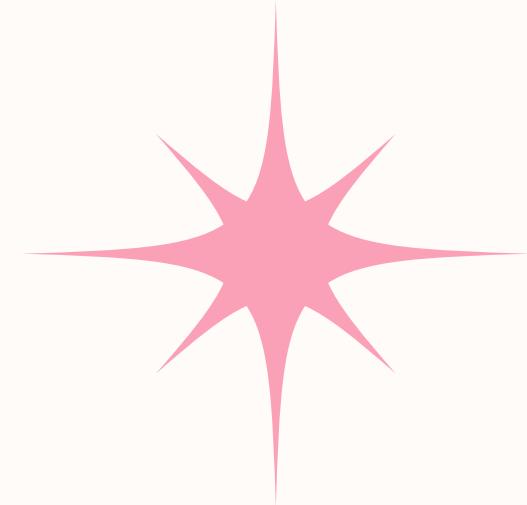
# docker-compose.yml

```
db:
  image: postgres:alpine
  env_file:
    - .env.db
  environment:
    - POSTGRES_INITDB_ARGS=--auth-host=md5 --auth-local=trust
  healthcheck:
    test: [ "CMD-SHELL", "pg_isready" ]
    interval: 10s
    timeout: 2s
    retries: 5
  volumes:
    - postgres-db:/var/lib/postgresql/data
  networks:
    - net
```

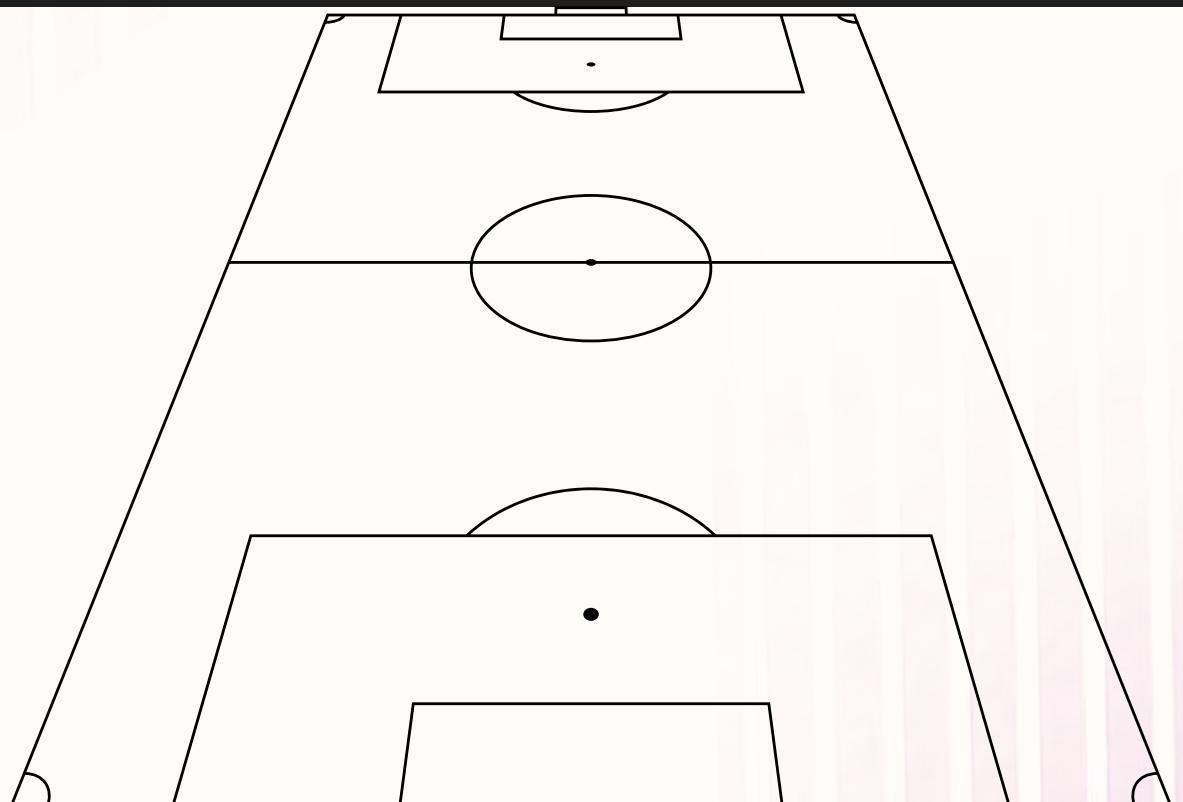
```
backend:
  build: .
  command: runserver 0.0.0.0:8000
  entrypoint: python3 manage.py
  env_file:
    - .env.db
  expose:
    - "8000"
  ports:
    - "8001:8000"
  volumes:
    - ./src:/code
  depends_on:
    db:
      condition: service_healthy
  networks:
    - net
```

# docker-compose.yml

```
generate:  
  build: .  
  user: root  
  command: /bin/sh -c 'mkdir src && django-admin startproject app src'  
  env_file:  
    - .env.db  
depends_on:  
  db:  
    condition: service_healthy  
volumes:  
  - ./code  
networks:  
  - net
```



```
manage:  
  build: .  
  entrypoint: python3 manage.py  
  env_file:  
    - .env.db  
volumes:  
  - ./src:/code  
depends_on:  
  db:  
    condition: service_healthy  
networks:  
  - net
```



# models.py - Equipo

```
class Equipo(models.Model):
    nombre = models.CharField(_('Nombre del equipo'), max_length=100)
    ciudad = models.ForeignKey(
        Ciudad,
        on_delete=models.PROTECT,
        verbose_name=_('Ciudad')
    )
    fechaDeFundacion = models.DateField(_('Fecha de fundación'))

    def __str__(self):
        return self.nombre

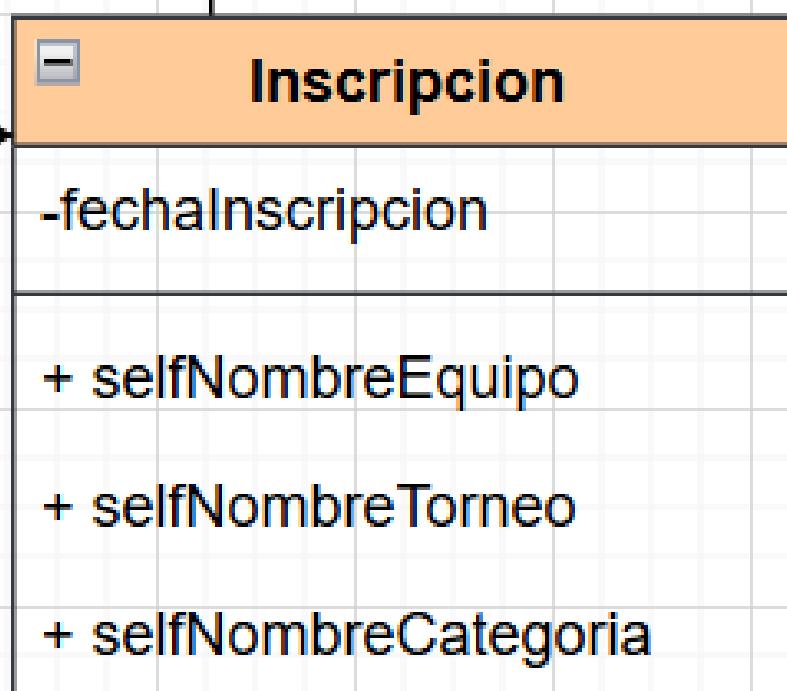
class Meta:
    verbose_name = 'Equipo'
    verbose_name_plural = 'Equipos'
```



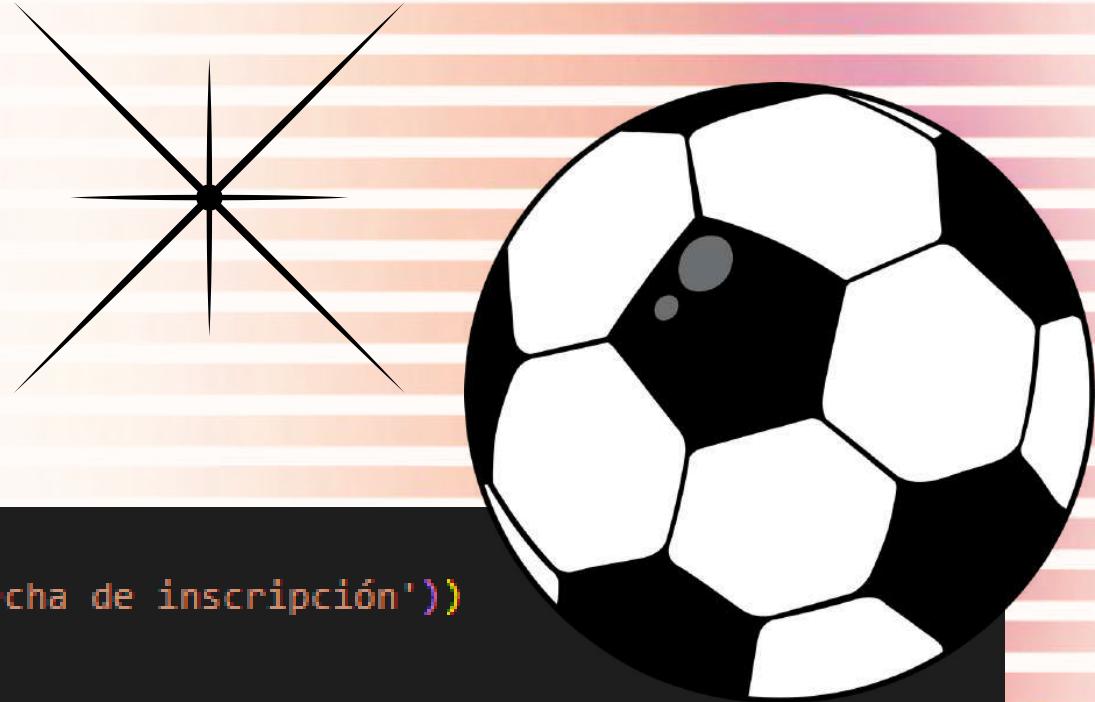
	NOMBRE DEL EQUIPO	CIUDAD	FECHA DE FUNDACIÓN
<input type="checkbox"/>	Club Atlético Independiente	Avellaneda	Jan. 1, 1905
<input type="checkbox"/>	Racing Club	Avellaneda	March 25, 1903
<input type="checkbox"/>	Club Atlético River Plate	Buenos Aires	May 25, 1901
<input type="checkbox"/>	Club Atlético Boca Juniors	Buenos Aires	April 3, 1905

# models.py - Inscripción

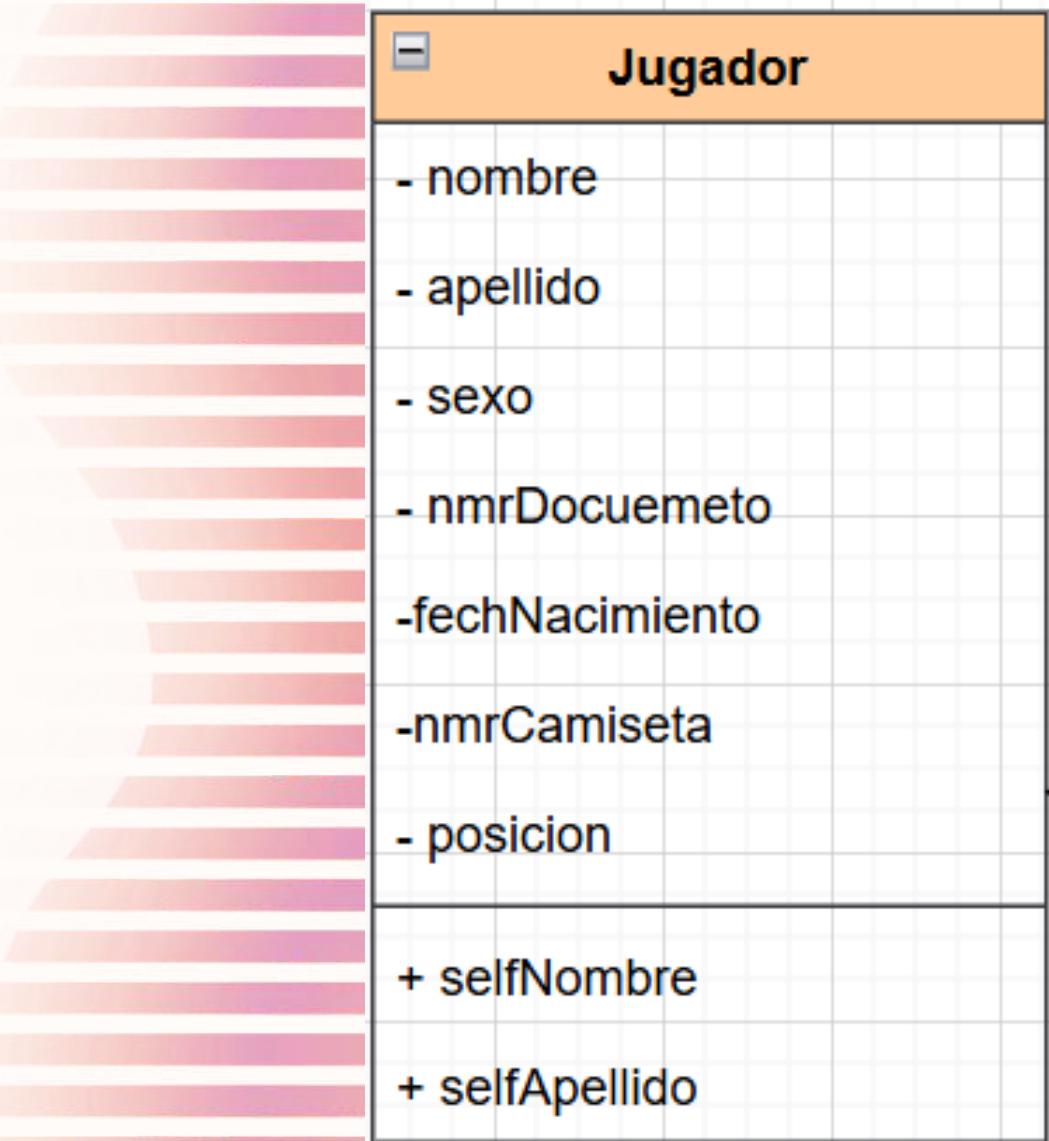
EQUIPO	TORNEO	FECHA DE INSCRIPCIÓN
Club Atlético Independiente	Liga Profesional Argentina 2025	June 4, 2025
Racing Club	Liga Profesional Argentina 2025	June 3, 2025
Club Atlético River Plate	Liga Profesional Argentina 2025	June 2, 2025
Club Atlético Boca Juniors	Liga Profesional Argentina 2025	June 1, 2025



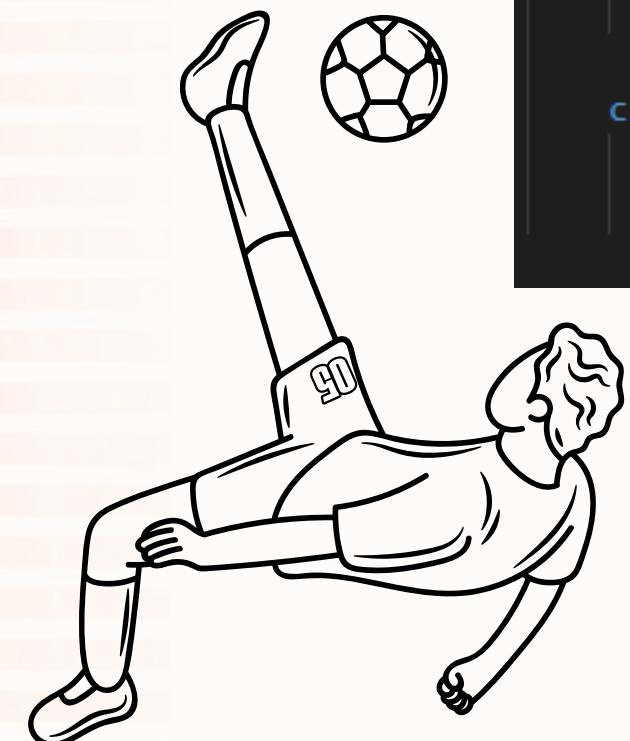
```
class Inscripcion(models.Model):  
    fechaInscripcion = models.DateField(_('Fecha de inscripción'))  
  
    equipo = models.ForeignKey(  
        Equipo,  
        on_delete=models.CASCADE,  
        verbose_name=_('Equipo'))  
    categoria = models.ForeignKey(  
        Categoria,  
        on_delete=models.PROTECT,  
        verbose_name=_('Categoría'),  
        default=1 # Categoría por defecto temporal)  
    torneo = models.ForeignKey(  
        Torneo,  
        on_delete=models.CASCADE,  
        verbose_name=_('Torneo'))  
  
    def __str__(self):  
        return f'{self.equipo.nombre} - {self.torneo.nombre} ({self.categoria.nombre})'  
  
class Meta:  
    verbose_name = 'Inscripción'  
    verbose_name_plural = 'Inscripciones'
```



# models.py - Jugador



	NOMBRE	APELLIDO	EQUIPO	NÚMERO DE DOCUMENTO	POSICIÓN
□	Leandro	Fernández	Club Atlético Independiente	41234567	Delantero
□	Juan Fernando	Quintero	Racing Club	26789123	Mediocampista
□	Julián	Álvarez	Club Atlético River Plate	44123456	Delantero
□	Franco	Armani	Club Atlético River Plate	31987654	Arquero
□	Marcos	Rojo	Club Atlético Boca Juniors	32456789	Defensor
□	Edinson	Cavani	Club Atlético Boca Juniors	35123456	Delantero



```
class Jugador(models.Model):
    nombre = models.CharField(_('Nombre'), max_length=100)
    apellido = models.CharField(_('Apellido'), max_length=100)
    sexo = models.CharField(_('Sexo'), max_length=10)
    numeroDocumento = models.BigIntegerField(_('Número de documento'))
    fechaNacimiento = models.DateField(_('Fecha de nacimiento'))
    nroCamiseta = models.IntegerField(_('Número de camiseta'))
    posicion = models.CharField(_('Posición'), max_length=50)

    tipoDocumento = models.ForeignKey(
        TipoDocumento,
        on_delete=models.PROTECT,
        verbose_name=_('Tipo de documento')
    )
    equipo = models.ForeignKey(
        Equipo,
        on_delete=models.CASCADE,
        verbose_name=_('Equipo')
    )

    def __str__(self):
        return f'{self.nombre} {self.apellido}'

    class Meta:
        verbose_name = 'Jugador'
        verbose_name_plural = 'Jugadores'
```

# models.py - Partido

Partido	
-	fechaPartido
-	horaPartdio
-	resultado
+	selfNombreEquipo1
+	selfNombreEquipo2



```
class Partido(models.Model):
    fechaPartido = models.DateField(_('Fecha del partido'))
    horaPartido = models.TimeField(_('Hora del partido'))
    resultado = models.CharField(_('Resultado'), max_length=50, blank=True)

    equipo1 = models.ForeignKey(
        Equipo,
        on_delete=models.CASCADE,
        related_name='partidos_equipo1',
        verbose_name=_('Equipo 1')
    )
    equipo2 = models.ForeignKey(
        Equipo,
        on_delete=models.CASCADE,
        related_name='partidos_equipo2',
        verbose_name=_('Equipo 2')
    )
    cancha = models.ForeignKey(
        Cancha,
        on_delete=models.PROTECT,
        verbose_name=_('Cancha')
    )
    torneo = models.ForeignKey(
        Torneo,
        on_delete=models.CASCADE,
        verbose_name=_('Torneo')
    )

    def __str__(self):
        return f'{self.equipo1.nombre} vs {self.equipo2.nombre}'

    class Meta:
        verbose_name = 'Partido'
        verbose_name_plural = 'Partidos'
        ordering = ['fechaPartido', 'horaPartido']
```

	FECHA DEL PARTIDO	HORA DEL PARTIDO	EQUIPO 1	EQUIPO 2
	July 15, 2025	9 p.m.	Club Atlético Boca Juniors	Club Atlético F
	July 20, 2025	7 p.m.	Racing Club	Club Atlético I



# admin.py

```
@admin.register(Torneo)
class TorneoAdmin(admin.ModelAdmin):
    list_display = ('nombre', 'fechaInicio', 'fechaFin', 'categoria', 'total_partidos')
    list_filter = ['categoria']
    search_fields = ['nombre']
    inlines = [InscripcionInline, PartidoInline]
    ordering = ['fechaInicio']

    def total_partidos(self, obj):
        return obj.partido_set.count()
    total_partidos.short_description = 'Cantidad de partidos'
```

```
class PartidoInline(admin.TabularInline):
    model = Partido
    extra = 0
    verbose_name = "Partido del torneo"
    verbose_name_plural = "Partidos del torneo"
```

PARTIDOS DEL TORNEO					
FECHA DEL PARTIDO	HORA DEL PARTIDO	RESULTADO	EQUIPO 1	EQUIPO 2	CANCHAS
Club Atlético Boca Juniors vs Club Atlético River Plate 2025-07-15	21:00:00	2-1	Club Atlético Boca Juniors	Club Atlético River Plate	La Bombonera
Today	Now	Note: You are 3 hours behind server time.			
Racing Club vs Club Atlético Independiente 2025-07-20	19:00:00	1-0	Racing Club	Club Atlético Independiente	El Monumental
Today	Now	Note: You are 3 hours behind server time.			
<a href="#">+ Add another Partido del torneo</a>					
<a href="#">SAVE</a>	<a href="#">Save and add another</a>	<a href="#">Save and continue editing</a>			
<a href="#">Delete</a>					

## Change Torneo

### Liga Profesional Argentina 2025

Nombre del torneo:

Fecha de inicio:  Today |   
Note: You are 3 hours behind server time.

Fecha de fin:  Today |   
Note: You are 3 hours behind server time.

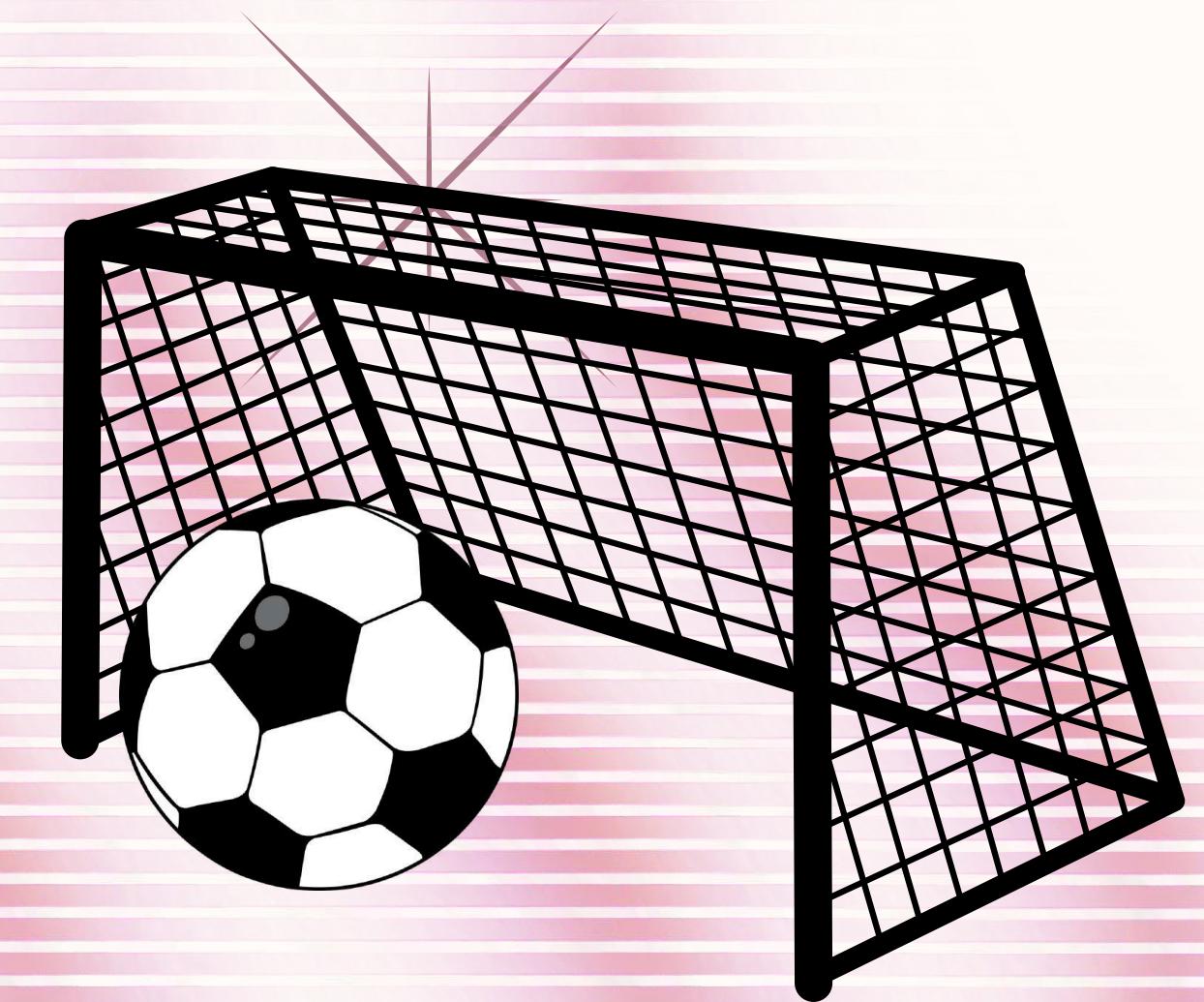
Categoría:

INSCRIPCIONES DE EQUIPOS					
FECHA DE INSCRIPCIÓN	EQUIPO	CATEGORÍA	DELETE?		
Club Atlético Boca Juniors - Liga Profesional Argentina 2025 (PRIMERA DIVISIÓN) 2025-06-01	Club Atlético Boca Juniors	PRIMERA DIVISIÓN	<input type="checkbox"/>		
Club Atlético River Plate - Liga Profesional Argentina 2025 (PRIMERA DIVISIÓN) 2025-06-02	Club Atlético River Plate	PRIMERA DIVISIÓN	<input type="checkbox"/>		
Racing Club - Liga Profesional Argentina 2025 (PRIMERA DIVISIÓN) 2025-06-03	Racing Club	PRIMERA DIVISIÓN	<input type="checkbox"/>		
Club Atlético Independiente - Liga Profesional Argentina 2025 (PRIMERA DIVISIÓN) 2025-06-04	Club Atlético Independiente	PRIMERA DIVISIÓN	<input type="checkbox"/>		

# Mapeo ORM

```
# Clases del modelo
class TipoDocumento(NombreAbstract): ...
class Ciudad(models.Model): ...
class Equipo(models.Model): ...
class Jugador(models.Model): ...
class Cancha(models.Model): ...
class Categoria(NombreAbstract): ...
class Torneo(models.Model): ...
class Inscripcion(models.Model): ...
class Partido(models.Model): ...

#Definido el models.py
```



```
PS C:\Users\Usuario\Downloads\torneo_futbol-mongo> docker-compose exec postgres psql -U postgres -d postgres -c "SELECT schemaname
, tablename, tableowner FROM pg_tables WHERE schemaname = 'public' AND tablename LIKE 'torneo_%' ORDER BY tablename;"
```

schemaname	tablename	tableowner
public	torneo_cancha	postgres
public	torneo_categoria	postgres
public	torneo_ciudad	postgres
public	torneo_equipo	postgres
public	torneo_inscripcion	postgres
public	torneo_jugador	postgres
public	torneo_partido	postgres
public	torneo_tipodocumento	postgres
public	torneo_torneo	postgres

(9 rows)

# Insertando datos desde DJANGO

## Creamos un nuevo Equipo

### Add Equipo

Nombre del equipo: Club Atlético San Lorenzo de Almagro

Ciudad: Buenos Aires    

Fecha de fundación: 1908-04-01 Today |   
Note: You are 3 hours behind server time.

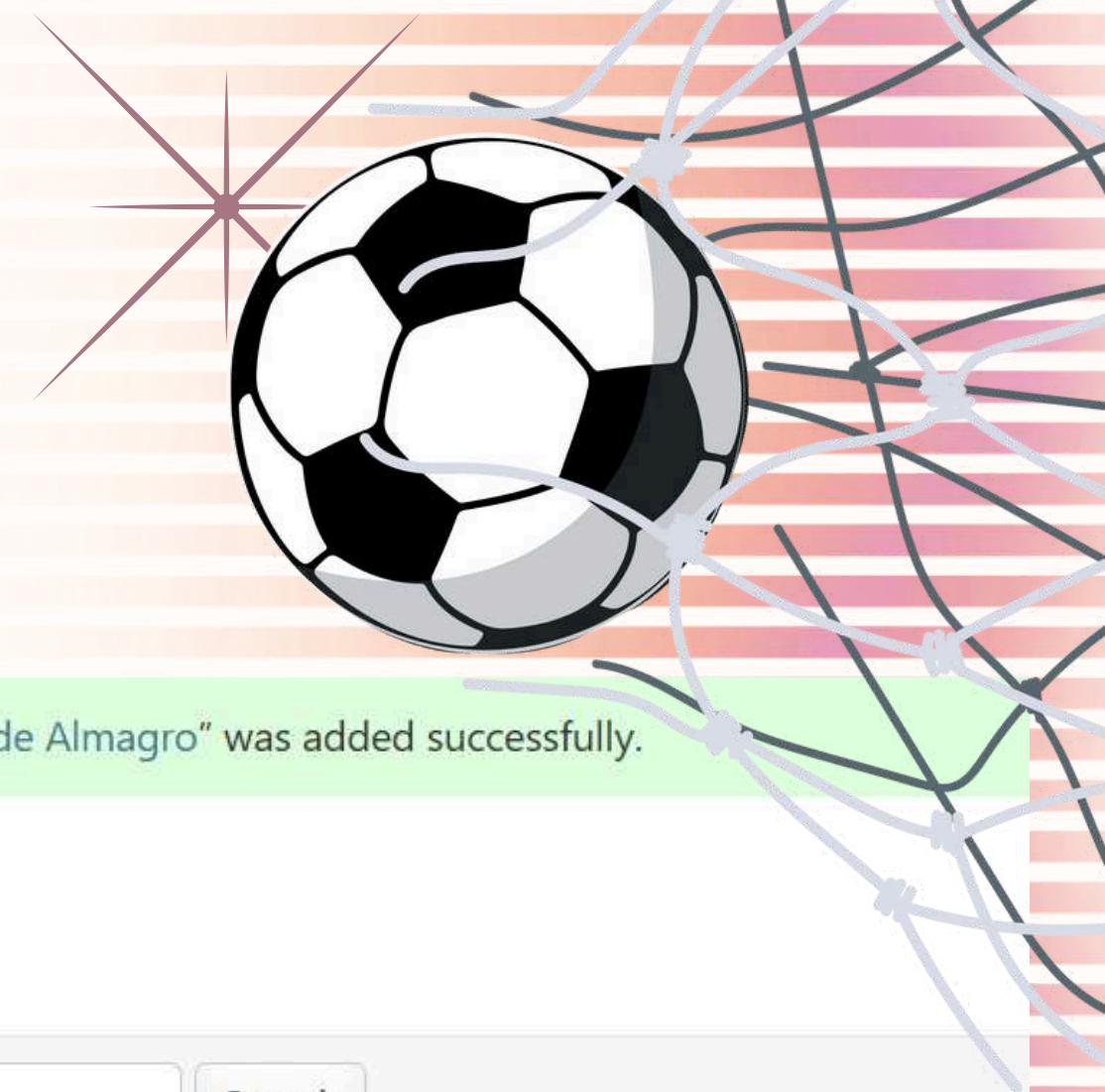
**SAVE** **Save and add another** **Save and continue editing**

 The Equipo "Club Atlético San Lorenzo de Almagro" was added successfully.

Select Equipo to change

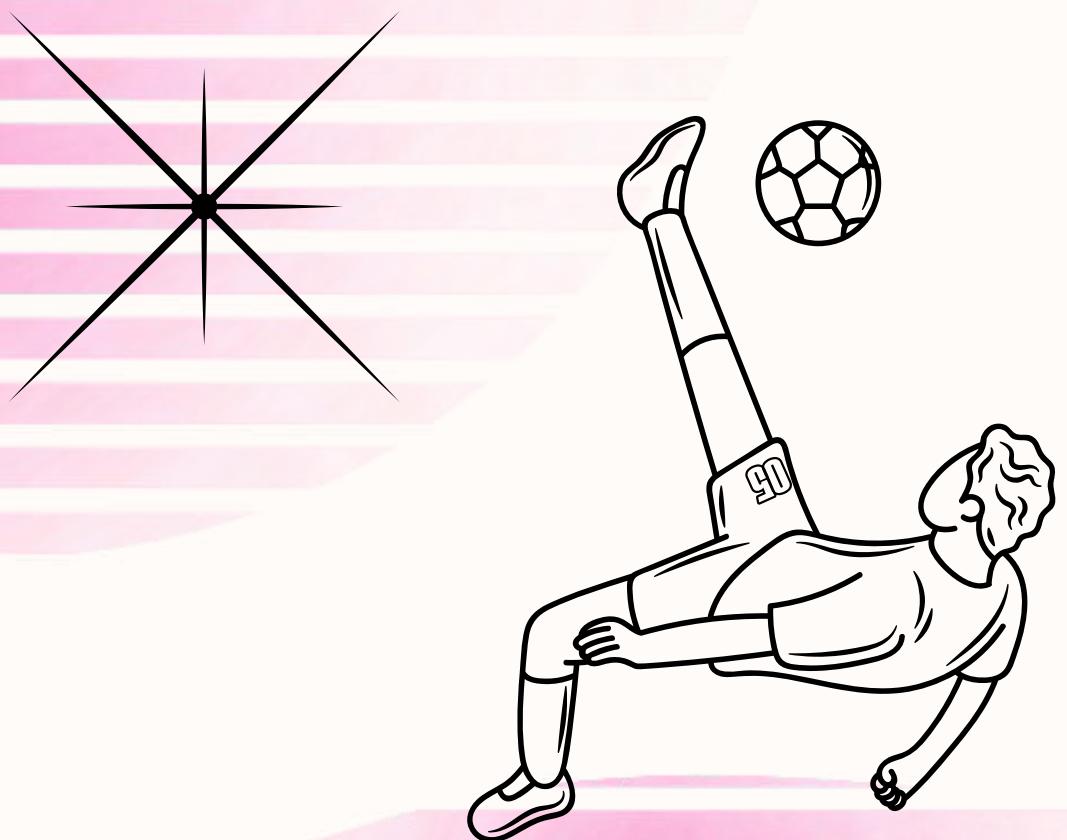
  Search

Action:	NOMBRE DEL EQUIPO	CIUDAD	FECHA DE FUNDACIÓN
<input type="checkbox"/>	Club Atlético San Lorenzo de Almagro	Buenos Aires	April 1, 1908
<input type="checkbox"/>	Club Atlético Independiente	Avellaneda	Jan. 1, 1905
<input type="checkbox"/>	Racing Club	Avellaneda	March 25, 1903
<input type="checkbox"/>	Club Atlético River Plate	Buenos Aires	May 25, 1901
<input type="checkbox"/>	Club Atlético Boca Juniors	Buenos Aires	April 3, 1905



# Insertando datos desde DJANGO

## Creamos un nuevo Jugador



### Add Jugador

Nombre: Juan Cruz

Apellido: Rattalino

Sexo: Masculino

Número de documento: 47123456

Fecha de nacimiento: 2004/06/23 | Today | Note: You are 3 hours behind server time.

Número de camiseta: 40

Posición: Mediocampista

Tipo de documento: DNI |

Equipo: Club Atlético San Lorenzo de Almagro |

**SAVE** **Save and add another** **Save and continue editing**

The Jugador "Juan Cruz Rattalino" was added successfully.

Select Jugador to change

Search

Action: ----- Go 0 of 7 selected

<input type="checkbox"/>	NOMBRE	APELLIDO	EQUIPO	NÚMERO DE DOCUMENTO	POSICIÓN
<input type="checkbox"/>	Juan Cruz	Rattalino	Club Atlético San Lorenzo de Almagro	47123456	Mediocampista
<input type="checkbox"/>	Leandro	Fernández	Club Atlético Independiente	41234567	Delantero
<input type="checkbox"/>	Juan Fernando	Quintero	Racing Club	26789123	Mediocampista
<input type="checkbox"/>	Julián	Álvarez	Club Atlético River Plate	44123456	Delantero
<input type="checkbox"/>	Franco	Armani	Club Atlético River Plate	31987654	Arquero
<input type="checkbox"/>	Marcos	Rojo	Club Atlético Boca Juniors	32456789	Defensor
<input type="checkbox"/>	Edinson	Cavani	Club Atlético Boca Juniors	35123456	Delantero
7 Jugadores					

# Insertando datos desde DJANGO

Verificamos en la bd

```
SELECT id, nombre, ciudad  
FROM torneo_equipo  
ORDER BY id;
```

id	nombre	ciudad
1	Club Atlético Boca Juniors	Buenos Aires
2	Club Atlético River Plate	Buenos Aires
3	Racing Club	Avellaneda
4	Club Atlético Independiente	Avellaneda
6	Club Atlético San Lorenzo de Almagro	Buenos Aires
(5 rows)		

```
SELECT j.nombre, j.apellido, j.posicion, e.nombre as equipo  
FROM torneo_jugador j  
JOIN torneo_equipo e ON j.equipo_id = e.id  
WHERE e.id = 6;
```

nombre	apellido	posicion	equipo
Juan Cruz	Rattalino	Mediocampista	Club Atlético San Lorenzo de Almagro
(1 row)			



# Django + MongoDB + PostgreSQL

```
services:  
  > Run Service  
    docker-compose.yml  
  
  postgres:  
    image: postgres:15-alpine  
    container_name: postgres  
    restart: unless-stopped  
    env_file:  
      - .env.db  
    volumes:  
      - postgres-data:/var/lib/postgresql/data  
    ports:  
      - "5433:5432"  
    networks:  
      - net  
  
  > Run Service  
  mongo:  
    image: mongo:7  
    container_name: mongo  
    restart: unless-stopped  
    environment:  
      MONGO_INITDB_DATABASE: torneo  
      MONGO_INITDB_ROOT_USERNAME: root  
      MONGO_INITDB_ROOT_PASSWORD: example  
    volumes:  
      - mongo-data:/data/db  
    ports:  
      - "27019:27017"  
    networks:  
      - net
```

```
# PostgreSQL  
DATABASE_ENGINE=django.db.backends.postgresql  
POSTGRES_DB=postgres  
POSTGRES_USER=postgres  
POSTGRES_PASSWORD=postgres  
POSTGRES_HOST=postgres  
POSTGRES_PORT=5432  
POSTGRES_INITDB_ARGS=--auth-host=md5 --auth-local=trust  
LANG=es_AR.utf8  
  
# MongoDB  
MONGO_DB=torneo  
MONGO_URI=mongodb://root@example@mongo:27017/torneo?authSource=admin  
MONGO_URI_LOCAL=mongodb://root@example@localhost:27019/torneo?authSource=admin  
MONGO_USER=root  
MONGO_PASS=example  
MONGO_HOST=mongo  
MONGO_PORT=27017  
  
# Django  
SECRET_KEY=clave-insegura-para-dev-torneo  
DEBUG=True  
ALLOWED_HOSTS=*
```

```
# Conexión MongoDB  
try:  
    # Usar las variables de entorno para la conexión  
    mongo_uri = os.getenv("MONGO_URI", "mongodb://root@example@mongo:27017/torneo?authSource=admin")  
  
    connect(  
        host=mongo_uri,  
        alias='default'  
    )  
    print("Conexión con MongoDB correctamente establecida")  
    print(f"URI utilizada: {mongo_uri}")  
except Exception as e:  
    print(f"Error al conectar a MongoDB: {e}")
```

```
Django>=4.2  
psycopg[binary]>=3.1  
mongoengine>=0.29  
gunicorn  
requirements.txt
```



# Modificación esquemas

//models.py

```
# Equipo que participa en Los torneos
class Equipo(models.Model):
    nombre = models.CharField(_('Nombre del equipo'), max_length=100)
    ciudad = models.ForeignKey(
        Ciudad,
        on_delete=models.PROTECT,
        verbose_name=_('Ciudad')
    )
    fechaDeFundacion = models.DateField(_('Fecha de fundación'))

    def __str__(self):
        return self.nombre

class Meta:
    verbose_name = 'Equipo'
    verbose_name_plural = 'Equipos'
```



//models\_mongo.py

```
# Equipo que participa en Los torneos
class Equipo(Document):
    postgres_id = IntField(required=True, unique=True)
    nombre = StringField(required=True, max_length=100)
    ciudad = ReferenceField(Ciudad, reverse_delete_rule=CASCADE)
    ciudad_nombre = StringField()
    fechaDeFundacion = DateField(required=True)

    def __str__(self):
        return self.nombre
```



# Sincronización automática con signals.py

```
from django.db.models.signals import post_save, post_delete
```

```
# =====
# EQUIPO
# =====
@receiver(post_save, sender=Equipo)
def sync_equipo(sender, instance, **kwargs):
    try:
        ciudad_doc = CiudadDoc.objects(postgres_id=instance.ciudad.id).first()
        if ciudad_doc:
            EquipoDoc.objects(postgres_id=instance.id).update_one(
                set_nombre=instance.nombre,
                set_ciudad=ciudad_doc,
                set_ciudad_nombre=instance.ciudad.nombreCiudad if instance.ciudad else "",
                set_fechaDeFundacion=instance.fechaDeFundacion,
                upsert=True
            )
    except Exception as e:
        print(f"[ERROR EN SYNC_EQUIPO] {e}")

@receiver(post_delete, sender=Equipo)
def delete_equipo(sender, instance, **kwargs):
    try:
        EquipoDoc.objects(postgres_id=instance.id).delete()
    except Exception as e:
        print(f"[ERROR EN DELETE_EQUIPO] {e}")
```

Usamos las señales **post\_save** y **post\_delete** de Django. Cada vez que se guarda o elimina un objeto (como un Jugador o un Torneo), se replica o borra en MongoDB.

Usamos **MongoEngine** como **ODM**, sin acceder manualmente a la base.



# Script inicial de carga y sincronización de datos

```
load_and_sync.py > ...
# src/Load_and_sync.py

import os
import django
from django.core.management import call_command

# Configurar entorno Django
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'app.settings')
django.setup()

# Importar modelos
from torneo.models import Ciudad, TipoDocumento, Categoria, Equipo, Jugador, Cancha, Torneo, Inscripcion, Partido

print("⬇️ Cargando datos en PostgreSQL desde el fixture...")
call_command('loaddata', 'torneo/fixtures/initial_data.json')

print("⌚ Sincronizando datos en MongoDB vía signals...")
modelos = [
    Ciudad,
    TipoDocumento,
    Categoria,
    Equipo,
    Jugador,
    Cancha,
    Torneo,
    Inscripcion,
    Partido
]

for modelo in modelos:
    nombre_modelo = modelo.__name__
    objetos = modelo.objects.all()
    print(f" • {nombre_modelo}: {objetos.count()} objetos")
    for obj in objetos:
        obj.save() # Esto dispara los signals definidos en signals.py

print("✅ Sincronización completada.")
```

Con este script llamado **load\_and\_sync.py** podemos hacer una carga inicial de datos que hará que los mismos estén disponibles y sincronizados tanto en la base de datos Postgres como en la de Mongo

# Levantar el entorno híbrido

## Opción 1: script de inicialización ejecutando .\init.ps1

```
# Detener y eliminar contenedores, volúmenes e imágenes huérfanas
docker-compose down -v --remove-orphans --rmi all

# Construir imágenes y levantar contenedores postgres, mongo y backend en segundo plano
docker-compose up --build -d postgres mongo backend

# Esperar 10 segundos para asegurar que los servicios estén disponibles
Start-Sleep -Seconds 10

# Generar migraciones a partir de los modelos Django
docker-compose run --rm manage makemigrations

# Aplicar las migraciones a la base de datos PostgreSQL
docker-compose run --rm manage migrate

# Crear superusuario admin sin interacción (usuario: admin, email: admin@example.com)
docker-compose run --rm manage createsuperuser --noinput --username admin --email admin@example.com

# Establecer contraseña 'admin' para el superusuario creado
docker-compose run --rm manage shell -c >
"from django.contrib.auth import get_user_model; User = get_user_model(); u, created = User.objects.get_or_create(username='admin',
defaults={'email': 'admin@example.com'}); u.set_password('admin'); u.save()"

# Cargar datos iniciales desde fixture JSON
docker-compose run --rm manage loaddata initial_data

# Ejecutar script para cargar y sincronizar datos con MongoDB
docker-compose exec backend python load_and_sync.py

# Mensaje final para informar que el proceso terminó
echo "El proyecto se levantó correctamente. Podes acceder al admin en http://localhost:8000/admin con usuario 'admin' y contraseña 'admin'."
```

**Solo la  
primera  
vez**



# Levantar el entorno híbrido

## Opción 2: manualmente con docker-compose

- Levantamos el proyecto

`docker-compose up --build -d`

```
PS C:\Users\Usuario\Downloads\torneo_futbol-mongo> docker-compose up --build -d
Compose can now delegate builds to bake for better performance.
To do so, set COMPOSE_BAKE=true.
[+] Building 32.8s (19/23)
=> [manage internal] load build definition from Dockerfile
=> => transferring dockerfile: 914B
=> [backend internal] load build definition from Dockerfile
=> => transferring dockerfile: 914B
-
✓ backend                                Built
✓ manage                                  Built
✓ Container postgres                      Started
✓ Container mongo                         Started
✓ Container django-manage                 Started
✓ Container django-backend                Started
PS C:\Users\Usuario\Downloads\torneo_futbol-mongo>
```



# Levantar el entorno híbrido

- Hacemos las migraciones

```
docker-compose run --rm manage makemigrations  
docker-compose run --rm manage migrate
```

```
PS C:\Users\Usuario\Downloads\torneo_futbol-mongo> docker-compose run --rm manage makemigrations  
[+] Creating 2/2  
  ✓ Container mongo    Running          0.0s  
●  ✓ Container postgres Running          0.0s  
Conexión con MongoDB correctamente establecida  
● URI utilizada: mongodb://root@example@mongo:27017/torneo?authSource=admin  
No changes detected  
PS C:\Users\Usuario\Downloads\torneo_futbol-mongo> docker-compose run --rm manage migrate  
[+] Creating 2/2  
  ✓ Container mongo    Running          0.0s  
  ✓ Container postgres Running          0.0s  
Conexión con MongoDB correctamente establecida  
URI utilizada: mongodb://root@example@mongo:27017/torneo?authSource=admin  
Operations to perform:  
  Apply all migrations: admin, auth, contenttypes, sessions, torneo  
Running migrations:  
  No migrations to apply.
```



# Levantar el entorno híbrido

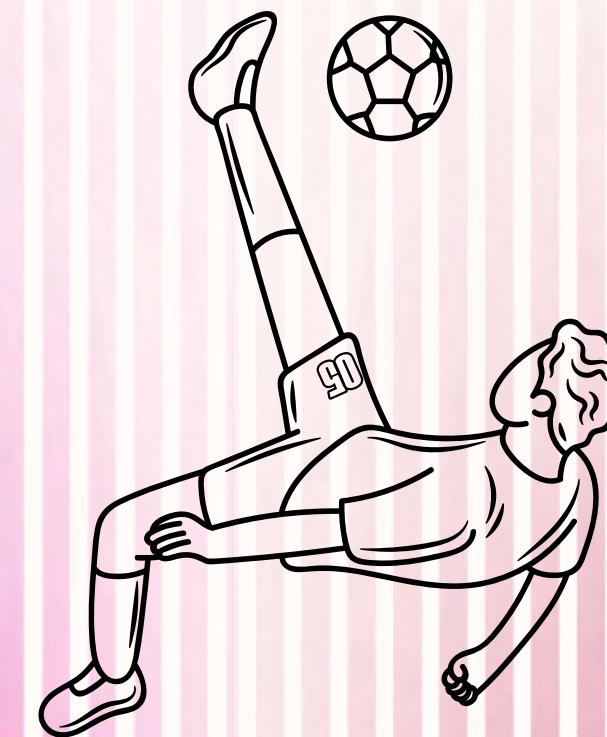
- Creamos el superusuario

`docker-compose run --rm manage createsuperuser`

```
[+] Creating 2/2
✓ Container mongo    Running
✓ Container postgres  Running
Conexión con MongoDB correctamente establecida
URI utilizada: mongodb://root:example@mongo:27017/torneo?authSource=admin
Username (leave blank to use 'root'): admin2
Email address: admin2@mail.com
Password:
Password (again):
The password is too similar to the username.
This password is too common.
Bypass password validation and create user anyway? [y/N]: y
Superuser created successfully.
```

- Carga de datos inicial

`docker-compose exec backend bash`  
`python load_and_sync.py`



```
PS C:\Users\Usuario\Downloads\torneo_futbol-mongo> docker-compose exec backend bash
6a5ec9688397:/code# python load_and_sync.py
Conexión con MongoDB correctamente establecida
URI utilizada: mongodb://root:example@mongo:27017/torneo?authSource=admin
🕒 Cargando datos en PostgreSQL desde el fixture...
Installed 27 object(s) from 1 fixture(s)
🕒 Sincronizando datos en MongoDB vía signals...
  • Ciudad: 2 objetos
  • TipoDocumento: 2 objetos
  • Categoria: 3 objetos
  • Equipo: 5 objetos
  • Jugador: 7 objetos
  • Cancha: 2 objetos
  • Torneo: 2 objetos
  • Inscripcion: 4 objetos
  • Partido: 2 objetos
✅ Sincronización completada.
```

localhost:27019

- admin
- config
- local
- torneo
  - cancha
  - categoria
  - ciudad
  - equipo
  - inscripcion
  - jugador
  - partido
  - tipo\_documento
  - torneo
- ...

<a href="#">+ ADD DATA</a>	<a href="#">EXPORT DATA</a>	<a href="#">UPDATE</a>	<a href="#">DELETE</a>
<pre>_id: ObjectId('68748cb821b174e48e3a26c1') postgres_id : 3 ciudad : ObjectId('68748cb721b174e48e3a26ab') ciudad_nombre : "Avellaneda" fechaDeFundacion : 1903-03-25T00:00:00.000+00:00 nombre : "Racing Club"</pre>			
<pre>_id: ObjectId('68748cb821b174e48e3a26c4') postgres_id : 4 ciudad : ObjectId('68748cb721b174e48e3a26ab') ciudad_nombre : "Avellaneda" fechaDeFundacion : 1905-01-01T00:00:00.000+00:00 nombre : "Club Atlético Independiente"</pre>			
<pre>_id: ObjectId('687491be21b174e48e3a2ad9') postgres_id : 5 ciudad : ObjectId('68748cb721b174e48e3a26a9') ciudad_nombre : "Buenos Aires" fechaDeFundacion : 1908-04-01T00:00:00.000+00:00 nombre : "Club Atlético San Lorenzo de Almagro"</pre>			

**TORNEO**

CANCHAS	<a href="#">+ Add</a>
Canchas	<a href="#">+ Add</a>
CATEGORIAS	<a href="#">+ Add</a>
Categorías	<a href="#">+ Add</a>
CIUDADES	<a href="#">+ Add</a>
Ciudades	<a href="#">+ Add</a>
EQUIPOS	<a href="#">+ Add</a>
Equipos	<a href="#">+ Add</a>
INSCRIPCIONES	<a href="#">+ Add</a>
Inscripciones	<a href="#">+ Add</a>
JUGADORES	<a href="#">+ Add</a>
Jugadores	<a href="#">+ Add</a>
PARTIDOS	<a href="#">+ Add</a>
Partidos	<a href="#">+ Add</a>
TIPOS DE DOCUMENTO	<a href="#">+ Add</a>
Tipos de documento	<a href="#">+ Add</a>
TORNEOS	<a href="#">+ Add</a>
Torneos	<a href="#">+ Add</a>

<input type="checkbox"/>	NOMBRE DEL EQUIPO	CIUDAD	FECHA DE FUNDACIÓN
<input type="checkbox"/>	Club Atlético San Lorenzo de Almagro	Buenos Aires	April 1, 1908
<input type="checkbox"/>	Club Atlético Independiente	Avellaneda	Jan. 1, 1905
<input type="checkbox"/>	Racing Club	Avellaneda	March 25, 1903
<input type="checkbox"/>	Club Atlético River Plate	Buenos Aires	May 25, 1901
<input type="checkbox"/>	Club Atlético Boca Juniors	Buenos Aires	April 3, 1905

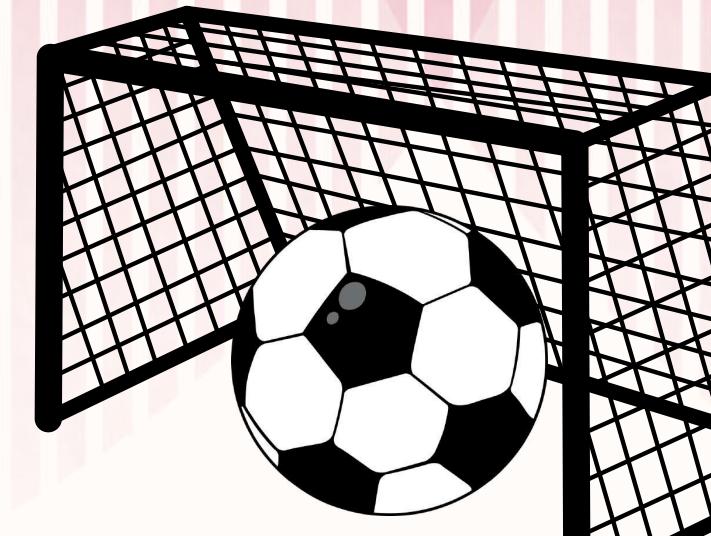
5 Equipos

# DB en mongo con Mongo Compass

mongodb://root:example@localhost:27019/torneo?authSource=admin

# Postgres + Django en puerto 8000

<http://localhost:8000/admin/>



# Django + PostgreSQL

[https://github.com/Bianll26/torneo\\_futbol-django](https://github.com/Bianll26/torneo_futbol-django)



# Django + MongoDB + PostgreSQL

[https://github.com/Bianll26/torneo\\_futbol-mongo](https://github.com/Bianll26/torneo_futbol-mongo)



## ACLARACIÓN

El repositorio de Django + Postgres quedó como una prueba inicial de cómo arrancamos el proyecto únicamente con postgres. Luego, en el de Django + MongoDB + Postgres es donde realmente se hizo la unificación de todo y quedó como proyecto principal.

**¡GRACIAS POR  
SU ATENCIÓN!**

