



DeRevés – Diagrama de Datos (versión inicial)

Este documento describe el **modelo de datos conceptual** de DeRevés.

Sirve como base para definir los `models.py` en las distintas apps de Django.

⚠ Nota: es una primera versión para arrancar el desarrollo.

A medida que avancemos, iremos ajustando campos, índices y relaciones.

1. Convenciones generales

- Todas las tablas tienen:
 - `id` (PK, entero autoincremental).
 - Campos de auditoría básicos:
 - `creado_en`, `actualizado_en`.
- `Usuario` se basa en el modelo de Django (`auth_user`) o un `AbstractUser` custom.
- Nombres de modelos en **singular** (ej.: `Complejo`, `Cancha`, `Reserva`).

2. Visión general de entidades principales

A nivel alto, los bloques son:

- **Cuentas y perfiles:** usuarios, jugadores, dueños.
- **Complejos y canchas:** estructura física.
- **Reservas y partidos:** uso de canchas y organización de juegos.
- **Torneos:** campeonatos, categorías y resultados.
- **Red social:** relaciones entre jugadores y contenido social.
- **Valoraciones:** opiniones y pelotitas.
- **Publicidades y sponsors:** monetización global/local.
- **Sitio público:** información pública de cada complejo.

PROF

3. Módulo cuentas

3.1. Usuario

Usamos el usuario de Django (`User`) extendido con perfiles.

Modelo: `Usuario` (hereda de `AbstractUser` o `OneToOne` con `User`)

Campos extra sugeridos:

- `tipo_usuario` (choices): `JUGADOR`, `DUENIO`, `ORGANIZADOR`, `ADMIN_SISTEMA`
- `telefono`
- `foto_perfil` (opcional)

Notas importantes de identificación y login:

- **dni** (string, único, opcional/según país) — documento nacional de identidad para identificación real. Debe considerarse dato sensible; validar formato localmente y/o mediante verificación adicional si se requiere (ej.: verificación manual o servicios externos).
- **nombre_real** (string) — nombre completo real del usuario. Útil para comunicación, facturación y confianza en la plataforma. Puede exigirse en roles **DUENIO** o para pagos/contratos.
- **username** — el sistema puede soportar un **username** público (alias) para login y menciones; puede ser opcional si se usa el correo/Gmail para login.

Login con cuentas Gmail (Google OAuth2):

- Soporte recomendado: integrar OAuth2 (ej.: **django-allauth** o **social-auth-app-django**).
- Campos relacionados que podemos almacenar en **Usuario**:
 - **google_oauth2_id** (string, opcional) — id del usuario en Google para enlazar cuentas.
 - **email_verified** (bool) — si el proveedor (Google) verificó el email.
 - **provider** (choices) — **local**, **google**, **facebook**, etc. (útil para UX y soporte).
- Flujo sugerido:
 1. El usuario inicia sesión con Google.
 2. Si no existe usuario con ese email, crear **Usuario** con **email** y **google_oauth2_id** y marcar **email_verified=True**.
 3. Si existe un usuario local con el mismo email, pedir al usuario que vincule la cuenta (evitar sobrescribir datos sin consentimiento).
 4. Al permitir login sólo por Gmail, documentar en la UI que el email de Google será la identidad principal.

Privacidad y cumplimiento:

- El **dni** es un dato personal sensible en muchos países. Definir políticas de retención y acceso (quién puede ver DNI). Considerar cifrado en la base de datos (campo encriptado) o almacenamiento en un servicio con mayor control de acceso.
- Registrar consentimiento explícito si se recolecta DNI y para qué fines (facturación, verificación de identidad, etc.).
- Al usar Google OAuth, cumplir con las políticas de Google y mostrar la política de privacidad adecuada.

PROF

3.2. PerfilJugador

Modelo: PerfilJugador

- **usuario** (OneToOne → **Usuario**)
- **alias**
- **localidad**
- **estado_juego** (choices): **RECREATIVO**, **COMPETITIVO**
- **categoria_nivel** (FK → **CategoríaNivel**)
- **fecha_nacimiento** (opcional)
- **biografia** (texto corto)
- Estadísticas agregadas (podrían calcularse o denormalizarse):
 - **total_partidos**

- `total_torneos`
- `puntos_ranking`
- `pelotitas` (o estrellas acumuladas)

Relaciones:

- 1:1 con `Usuario`
 - 1:N con `ParticipantePartido`
 - 1:N con `InscripcionTorneo`
 - 1:N con `Valoracion`*
 - N:N con otros jugadores (a través de `Seguidor`)
-

3.3. PerfilDueno / Administrador de Complejo

Modelo: `PerfilDueno`

- `usuario` (OneToOne → `Usuario`)
- `nombre_comercial` (opcional)
- `telefono_contacto`
- `es_organizador_torneos` (bool)

Relaciones:

- 1:N con `Complejo`

En un futuro se puede unificar en un modelo `Perfil` con roles, pero para iniciar esta separación es clara.

4. Módulo complejos

4.1. Complejo

—
PROF

Modelo: `Complejo`

- `dueno` (FK → `PerfilDueno`)
- `nombre`
- `descripcion`
- `direccion`
- `localidad`
- `latitud`, `longitud` (opcional)
- `telefono`
- `email`
- `sitio_web` (opcional)
- `slug` (para URL)
- `subdominio` (ej.: `puntoyrev.es`, para `puntoyrev.es.dereves.ar`)
- `logo` (imagen)
- `activo` (bool)

Campos y consideraciones para geolocalización (Google Maps):

- **latitud, longitud** (float) — coordenadas en WGS84. Recomendado: llenar siempre para buenas búsquedas y mapas.
- **google_place_id** (string, opcional) — ID de Place de Google Maps si se obtiene (útil para enlaces directos, actualizaciones y place details).
- **direccion_formateada** (string) — dirección tal como la devuelve la API de Google (o servicio de geocoding), para mostrar en la ficha pública.
- **google_maps_url** (string, opcional) — url pública de Google Maps al lugar (para abrir en app/web).

Notas de captura y uso:

- Durante el alta/edición del complejo, sugerir autocompletar dirección con Places API para asegurar consistencia y obtener **place_id, direccion_formateada** y **lat/lng**.
- **latitud** y **longitud** deben usarse para búsquedas por proximidad, filtros por radio y para centrar mapas en la UI.
- Si no se usan servicios de Google, documentar la fuente de geocoding (OpenStreetMap / Nominatim) y conservar **direccion_formateada** para coherencia.
- Privacidad: la localización del complejo es pública por diseño (página pública), pero no almacenar datos sensibles adicionales.

Relaciones:

- 1:N con **Cancha**
- 1:N con **ServicioComplejo**
- 1:N con **Torneo**
- 1:N con **ValoracionComplejo**
- 1:N con **SponsorComplejo**
- 1:N con **PublicidadLocal** (si la separamos de globales)

4.2. Cancha

PROF

Modelo: Cancha

- **complejo** (FK → **Complejo**)
- **nombre** (ej.: "Cancha 1", "Central")
- **deporte** (choices): **PADEL**, **FUTBOL5**, **TENIS**, **FUTBOL_TENIS**, etc.
- **tipo_superficie** (texto/choices)
- **techada** (bool)
- **iluminacion** (bool)
- **precio_base** (Decimal)
- **duracion_turno_minutos** (int)
- **activo** (bool)

Relaciones:

- 1:N con **Reserva**

- 1:N con **Partido**
 - 1:N con **EncuentroTorneo**
 - 1:N con **ValoracionCancha**
-

4.3. ServicioComplejo

Modelo: **ServicioComplejo**

- **complejo** (FK → **Complejo**)
- **tipo_servicio** (choices o FK → **TipoServicio**):
 - BUFET, PARRILLA, AGUA_CALIENTE, WIFI, SALON, ESTACIONAMIENTO, etc.
- **descripcion** (opcional)

Alternativa: tabla **TipoServicio** si queremos parametrizar.

5. Módulo reservas

5.1. Reserva

Modelo: **Reserva**

- **cancha** (FK → **Cancha**)
- **jugador_principal** (FK → **PerfilJugador**)
(quien realiza la reserva; los demás pueden ser libres)
- **fecha** (date)
- **hora_inicio** (time / datetime)
- **hora_fin** (time / datetime)
- **precio** (Decimal)
- **estado** (choices): PENDIENTE, CONFIRMADA, CANCELADA, NO_ASISTIO, COMPLETADA
- **metodo_pago** (FK → **MetodoPago**, opcional)
- **pagado** (bool)
- **observaciones** (texto corto)

—
PROF

Relaciones:

- 1:N con **CheckIn** (si queremos registrar asistencia separada)
 - Podríamos relacionar con **Partido** si la reserva se usa para un partido social.
-

5.2. MetodoPago (opcional, MVP simple)

Modelo: **MetodoPago**

- **nombre** (ej.: EFECTIVO, TRANSFERENCIA, TARJETA, MP)
-

6. Módulo partidos

6.1. Partido

Modelo: Partido

- `reserva` (FK → Reserva, opcional pero recomendado)
- `creador` (FK → PerfilJugador)
- `cancha` (FK → Cancha)
(podría inferirse de `reserva`, pero lo dejamos explícito para flexibilidad)
- `tipo_partido` (choices): ABIERTO, CERRADO
- `nivel_sugerido` (FK → CategoriaNivel, opcional)
- `descripcion`
- `max_jugadores` (int)
- `estado` (choices): ABIERTO, LLENO, JUGADO, CANCELADO
- `es_amistoso` (bool)

Relaciones:

- 1:N con ParticipantePartido
-

6.2. ParticipantePartido

Modelo: ParticipantePartido

- `partido` (FK → Partido)
- `jugador` (FK → PerfilJugador)
- `rol` (choices): CREADOR, INVITADO, SUMADO
- `equipo` (int o char, ej.: 1, 2)
- `asistio` (bool)
- `marcador` (opcional, para guardar juegos/sets)

Esta tabla es el N:M entre jugador y partido.

PROF

7. Módulo torneos

7.1. CategoriaNivel

Modelo: CategoriaNivel

- `nombre` (ej.: "8va", "7ma", "6ta")
 - `orden` (int, para ranking de dificultad)
 - `descripcion` (opcional)
-

7.2. CategoriaEdad

Modelo: CategoriaEdad

- `nombre` (ej.: "Infantil", "Juvenil", "Adulto", "Senior")

- `edad_min` (opcional)
 - `edad_max` (opcional)
-

7.3. CategoriaGenero

Modelo: CategoriaGenero

- `nombre` (ej.: "Masculino", "Femenino", "Mixto")
-

7.4. Torneo

Modelo: Torneo

- `complejo` (FK → Complejo)
- `organizador` (FK → PerfilDueno o Usuario organizador)
- `nombre`
- `descripcion`
- `tipo_torneo` (choices): AMERICANO, LIGUILLA, ELIMINATORIA, MIXTO, TORNEO_LARGO
- `deporte` (choices)
- `fecha_inicio`
- `fecha_fin` (opcional)
- `inscripcion_desde` (datetime)
- `inscripcion_hasta` (datetime)
- `costo_inscripcion` (Decimal)
- `estado` (choices): BORRADOR, INSCRIPCION_ABIERTA, EN_CURSO, FINALIZADO, CANCELADO

Relaciones:

- 1:N con TorneoCategoria
- 1:N con InscripcionTorneo
- 1:N con EncuentroTorneo
- 1:N con PublicidadTorneo (si la diferenciamos)

—
PROF

7.5. TorneoCategoria

Modelo: TorneoCategoria

Relaciona un torneo con sus categorías:

- `torneo` (FK → Torneo)
 - `categoria_nivel` (FK → CategoriaNivel, opcional)
 - `categoria_edad` (FK → CategoriaEdad, opcional)
 - `categoria_genero` (FK → CategoriaGenero, opcional)
 - `cupos` (int, opcional)
-

7.6. InscripcionTorneo

Modelo: IncripcionTorneo

- **torneo** (FK → Torneo)
 - **torneo_categoria** (FK → TorneoCategoria, opcional)
 - **jugador_1** (FK → PerfilJugador)
 - **jugador_2** (FK → PerfilJugador, opcional, ej.: torneos singles vs dobles)
 - **fecha_inscripcion**
 - **estado** (choices): PENDIENTE, CONFIRMADA, CANCELADA
 - **observaciones**
-

7.7. EncuentroTorneo

Modelo: EncuentroTorneo

- **torneo** (FK → Torneo)
 - **ronda** (texto o int, ej.: "Grupo A - Fecha 1", "Cuartos", "Final")
 - **cancha** (FK → Cancha, opcional)
 - **fecha_hora** (datetime, opcional)
 - **inscripcion_local** (FK → IncripcionTorneo)
 - **inscripcion_visitante** (FK → IncripcionTorneo)
 - **estado** (choices): PENDIENTE, PROGRAMADO, JUGADO, W.O.
 - **resultado_texto** (resumen tipo "6-3 / 6-4")
-

7.8. ResultadoEncuentro (opcional, si queremos sets detallados)

Modelo: SetEncuentro

- **encuentro** (FK → EncuentroTorneo)
 - **numero_set** (int)
 - **games_local** (int)
 - **games_visitante** (int)
-

PROF

8. Módulo social

8.1. Seguidor

Modelo: Seguidor

Relación similar a “seguir” en redes sociales.

- **seguidor** (FK → PerfilJugador)
- **seguido** (FK → PerfilJugador)
- **creado_en**

Esto es un N:M jugador ↔ jugador.

8.2. Publicacion

Modelo: **Publicacion**

- **autor** (FK → **PerfilJugador**)
 - **texto**
 - **imagen** (opcional)
 - **tipo** (choices): **LOGRO, TORNEO, PARTIDO, GENERAL**
 - **visible_publico** (bool)
-

8.3. Comentario

Modelo: **Comentario**

- **publicacion** (FK → **Publicacion**)
 - **autor** (FK → **PerfilJugador**)
 - **texto**
-

8.4. Reaccion

Modelo: **Reaccion**

- **publicacion** (FK → **Publicacion**)
 - **autor** (FK → **PerfilJugador**)
 - **tipo** (choices): **LIKE, ME_GUSTA, FUEGO**, etc. (podemos simplificar en MVP)
-

8.5. Logro

Modelo: **Logro**

—
PROF

- **codigo** (string único, ej.: **CAMPEON_7MA**)
 - **nombre**
 - **descripcion**
 - **icono** (opcional)
-

8.6. LogroJugador

Modelo: **LogroJugador**

- **logro** (FK → **Logro**)
 - **jugador** (FK → **PerfilJugador**)
 - **fecha_obtenido**
-

9. Módulo **valoraciones**

9.1. ValoracionComplejo

Modelo: ValoracionComplejo

- `complejo` (FK → `Complejo`)
 - `jugador` (FK → `PerfilJugador`)
 - `puntaje` (int, 1–5 pelotitas)
 - `comentario`
 - `fecha`
-

9.2. ValoracionCancha

Modelo: ValoracionCancha

- `cancha` (FK → `Cancha`)
 - `jugador` (FK → `PerfilJugador`)
 - `puntaje` (int, 1–5)
 - `comentario`
 - `fecha`
-

10. Módulo `publicidades / finanzas`

10.1. Sponsor

Modelo: Sponsor

- `nombre`
 - `descripcion`
 - `logo`
 - `sitio_web`
 - `contacto_email`
 - `contacto_telefono`
 - `es_global` (bool)
(si es `True` lo maneja exclusivamente Mantis)
-

PROF

10.2. CampaniaPublicidad

Modelo: CampaniaPublicidad

- `sponsor` (FK → `Sponsor`)
 - `nombre`
 - `descripcion`
 - `fecha_inicio`
 - `fecha_fin`
 - `segmento_deporte` (opcional, ej.: solo pádel)
 - `segmento_region` (opcional)
 - `activo` (bool)
-

10.3. UbicacionPublicidad

Modelo: **UbicacionPublicidad**

Define dónde puede aparecer un banner:

- **codigo** (ej.: HOME_GLOBAL, SUBDOMINIO_COMPLEJO, PANEL_JUGADOR, PAGINA_TORNEO)
 - **descripcion**
-

10.4. Publicidad

Modelo: **Publicidad**

Instancia de una campaña en un lugar determinado.

- **campania** (FK → CampaniaPublicidad)
 - **ubicacion** (FK → UbicacionPublicidad)
 - **complejo** (FK → Complejo, opcional → si es local)
 - **imagen** (banner)
 - **url_destino**
 - **prioridad** (int)
 - **activo** (bool)
-

10.5. ImpresionPublicidad (opcional, para métricas)

Modelo: **ImpresionPublicidad**

- **publicidad** (FK → Publicidad)
 - **usuario** (FK → Usuario, opcional)
 - **ip**
 - **user_agent**
 - **es_click** (bool)
 - **fecha_hora**
-

PROF

10.6. SponsorComplejo (para sponsors locales)

Modelo: **SponsorComplejo**

- **sponsor** (FK → Sponsor)
 - **complejo** (FK → Complejo)
 - **descripcion_acuerdo** (texto)
 - **fecha_inicio**
 - **fecha_fin** (opcional)
-

11. Módulo **sitio_publico**

En general reutiliza modelos de **Complejo**, **Cancha**, **Torneo**, **Valoracion**.

Puede tener una configuración específica:

Modelo: **ConfiguracionSitioComplejo**

- **complejo** (FK → **Complejo**)
 - **color_primario**
 - **color_secundario**
 - **mostrar_torneos** (bool)
 - **mostrar_valoraciones** (bool)
-

12. Relación general resumida (texto)

- Un **Usuario** tiene un **PerfilJugador** o **PerfilDueño** (o ambos).
 - Un **PerfilDueño** administra uno o varios **Complejos**.
 - Un **Complejo** tiene muchas **Canchas** y muchos **Torneos**.
 - Una **Cancha** tiene muchas **Reservas** y puede participar en **EncuentrosTorneo** y **Partidos**.
 - Una **Reserva** puede estar asociada a un **Partido**.
 - Un **Partido** tiene muchos **ParticipantePartido** (jugadores).
 - Un **Torneo** tiene muchas **Inscripciones** y muchos **Encuentros**.
 - Los **Jugadores** pueden seguirse entre sí (**Seguidor**) y publicar contenido (**Publicacion**).
 - Los **Jugadores** valoran **Complejos** y **Canchas** (**Valoracion***).
 - Un **Sponsor** puede tener muchas **Campañas**, y estas se muestran como **Publicidades** en diferentes **Ubicaciones**, opcionalmente asociadas a un **Complejo** (global vs local).
-

✓ Con esto ya tenemos una base sólida para que Copilot nos ayude a ir generando los **models.py** por app (**cuentas/models.py**, **complejos/models.py**, etc.).

Después podemos ir simplificando o extendiendo según el MVP que elijamos.