Tipo de Dato	Campo Django	Ejemplo / Parámetros útiles
🔢 Enteros	`models.IntegerField()`	`edad = models.PositiveIntegerField()`
💲 Decimal / Moneda	`models.DecimalField(max_digits=8)`	`precio = models.DecimalField(max_digits=10, decimal_places=2)`
🔤 Texto corto (cadena)	`models.CharField(max_length=100)`	`nombre = models.CharField(max_length=150, unique=True)`
📝 Texto largo	`models.TextField()`	`descripcion = models.TextField(blank=True)`
🔢 Flotante	`models.FloatField()`	`promedio = models.FloatField(default=0.0)`
🗓️ Fecha	`models.DateField()`	`fecha_nacimiento = models.DateField(null=True, blank=True)`
🕓 Fecha y hora	`models.DateTimeField()`	`creada_en = models.DateTimeField(auto_now_add=True)`
🕒 Solo hora	`models.TimeField()`	`hora_inicio = models.TimeField()`
✅ Booleano	`models.BooleanField()`	`activo = models.BooleanField(default=True)`
🧾 Correo electrónico	`models.EmailField()`	`correo = models.EmailField(unique=True)`
🌐 URL	`models.URLField()`	`sitio_web = models.URLField(null=True, blank=True)`
🧩 ForeignKey (relación muchos-a-uno)	`models.ForeignKey(Modelo, on_delete=models.CASCADE)`	`hotel = models.ForeignKey(Hotel, on_delete=models.CASCADE, related_name='habitaciones')`
🔗 OneToOneField (relación uno-a-uno)	`models.OneToOneField(Modelo, on_delete=models.CASCADE)`	`perfil = models.OneToOneField(Huesped, on_delete=models.CASCADE)`
♾️ ManyToManyField (muchos-a-muchos)	`models.ManyToManyField(Modelo, blank=True)`	`servicios = models.ManyToManyField(Servicio, blank=True, related_name='habitaciones')`
⚙️ Choices (opciones limitadas)	`models.CharField(choices=OPCIONES, max_length=2)`	`estado = models.CharField(max_length=1, choices=ESTADOS, default='P')`
💾 Texto libre	`models.TextField()`	`notas = models.TextField(blank=True)`

Parámetro	Descripción
null=True	Permite valores nulos en la base de datos
blank=True	Permite dejar el campo vacío en formularios
default=...	Valor por defecto
unique=True	Valor único en la tabla
choices=[(...)]	Lista de opciones válidas
related_name='...'	Nombre del reverso de la relación
on_delete=models.CASCADE	Borra los hijos si se borra el padre
auto_now_add=True	Guarda fecha/hora solo al crear

CREACIÓN DE MODELOS (SIN ATRIBUTOS)

```
#------------------------------ NombreModelo ----------------------------------------
class NombreModelo(models.Model):

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

CREACIÓN DE ATRIBUTO CHOICE

```
    ESTADOS = [
        ('P', 'Pendiente'),
        ('C', 'Confirmada'),
        ('F', 'Finalizada'),
        ('X', 'Cancelada'),
    ]
    estado = models.CharField(max_length=1, choices=ESTADOS, default='P')
```

VALIDATOR (NUMERO MINIMO Y MAXIMO)

```python
from django.core.validators import MinValueValidator, MaxValueValidator

nivel_prioridad = models.PositiveIntegerField(
        validators=[
            MinValueValidator(1),   # valor mínimo permitido
            MaxValueValidator(10)   # valor máximo permitido
        ],
        default=1
    )
```

1:1 / 1:N / N:N

En el **1:1** o **N:N** da igual donde pongas la relación pero en un **1:N** tienes que poner la relación en el modelo que que tenga la **N**

En las **N:N** si tu tabla intermedia tiene atributos, entonces tienes que indicar en la relación **ManyToMany** (de cualquiera de los dos modelos) el **through** para indicar el nombre de la nueva tabla intermedia que vas a crear. Y cuando crees el modelo de esa tabla intermedia le pones la **FK** de ambas tablas.

```python
#------------------------------ NOTIFICACION ----------------------------------------
class Notificacion(models.Model):
    # Relacion 1:N con Administrador (Envia)
    autor = models.ForeignKey(Administrador, on_delete=models.CASCADE, related_name='notificaciones_enviadas')
    # Relacion N:N con Usuario (Recibe)
    usuarios = models.ManyToManyField(Usuario, through='Recibe', related_name='notificaciones_recibidas')

    titulo = models.CharField(max_length=100)
    mensaje = models.TextField()
    fecha_envio = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.titulo

#------------------------------ RECIBE (Relacion) ----------------------------------------
class Recibe(models.Model):
    ESTADOS = [('PE', 'Pendiente'), ('EN', 'Enviada')]

    # FK de la relacion N:N con Usuario - Notificacion
    usuario = models.ForeignKey(Usuario, on_delete=models.CASCADE, related_name='recibos')
    notificacion = models.ForeignKey(Notificacion, on_delete=models.CASCADE, related_name='recibos')

    estado = models.CharField(max_length=2, choices=ESTADOS, default='PE')
    fecha_recepcion = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.usuario.nombre_usuario} recibe {self.notificacion.titulo}"
```

---

1. En el archivo **models.py** creamos los modelos
2. Creamos las migraciones con el comando : **python manage.py makemigrations**
3. Ejecutamos la migración para que se cree en la Base de Datos: **python manage.py migrate**

Para crear el super usuario para crear modelos en el panel de administración

python manage.py createsuperuser

Al archivo **admin.py** y añadimos lo siguiente:

```
from .models import MODELO
admin.site.register(MODELO)
...
```

---

python manage.py runserver

http://127.0.0.1:8000/admin/

---

python manage.py dumpdata --indent 4 > examen/fixtures/datos.json

Comando para guardar todos los datos en **fixtures/datos.json**