{dev/talles}

django

Tipos de herencia para contenido polimórfico

Por: Ricardo Cuéllar

{dev/talles}

Herencia abstracta

Uso:

Cuando quieres compartir campos y métodos entre modelos sin crear una tabla en la base de datos para el modelo padre.

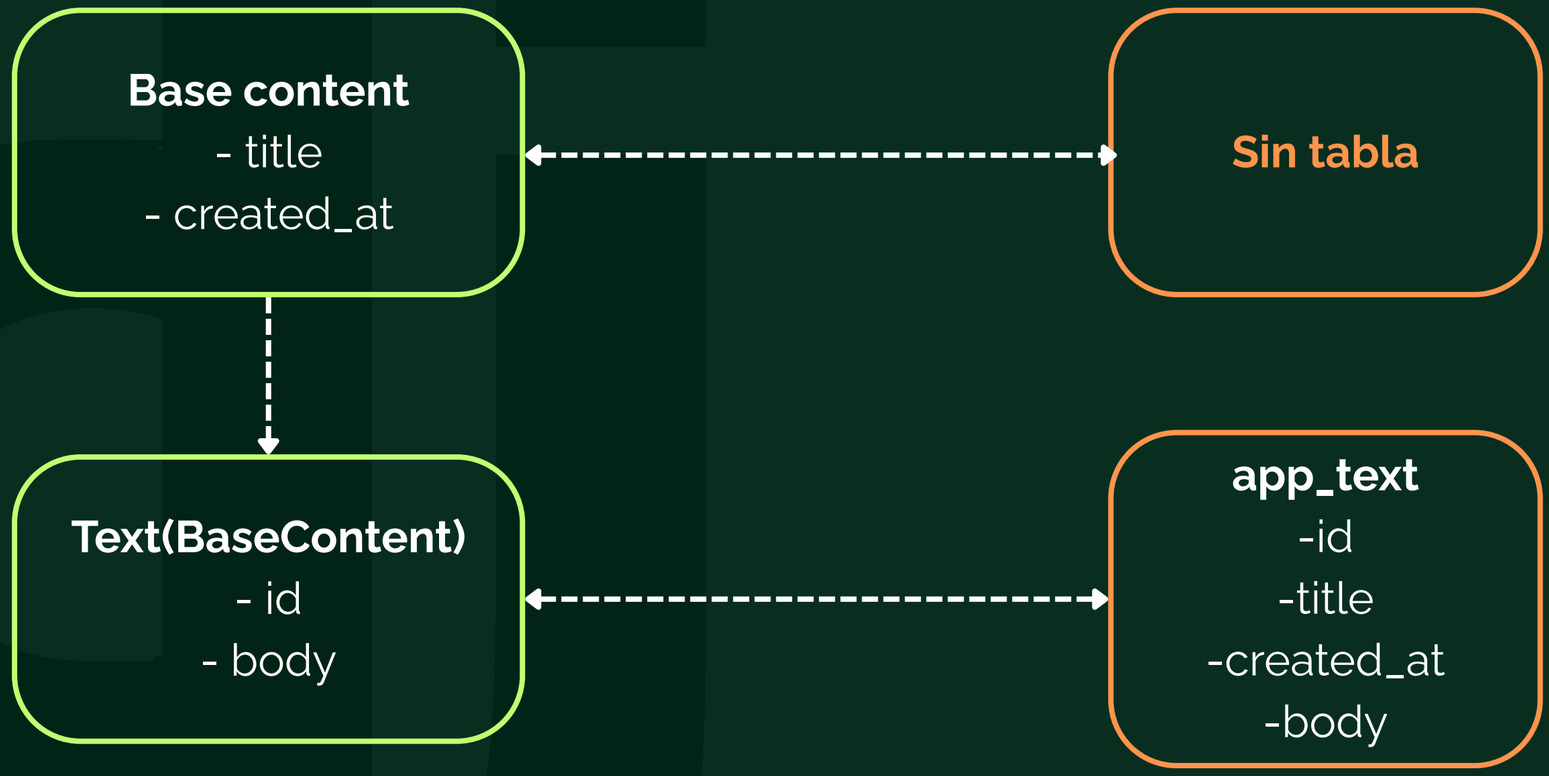
Ideal:

Para reutilizar código sin afectar la base de datos

```python
from django.db.import models

class BaseContent(models.Model):
    title = models.CharField(max_length=100)
    created_at = models.Datetimefield(auto_now_add = True)
    class Meta:
        abstract = True

class Text(BaseContent):
    body = models.TextField()
```

Modelos

Tablas Base de datos



{dev/talles}

Herencia Multi-tabla

Uso:

Cuando cada clase debe tener su propia tabla, pero quieres que compartan campos de una base.

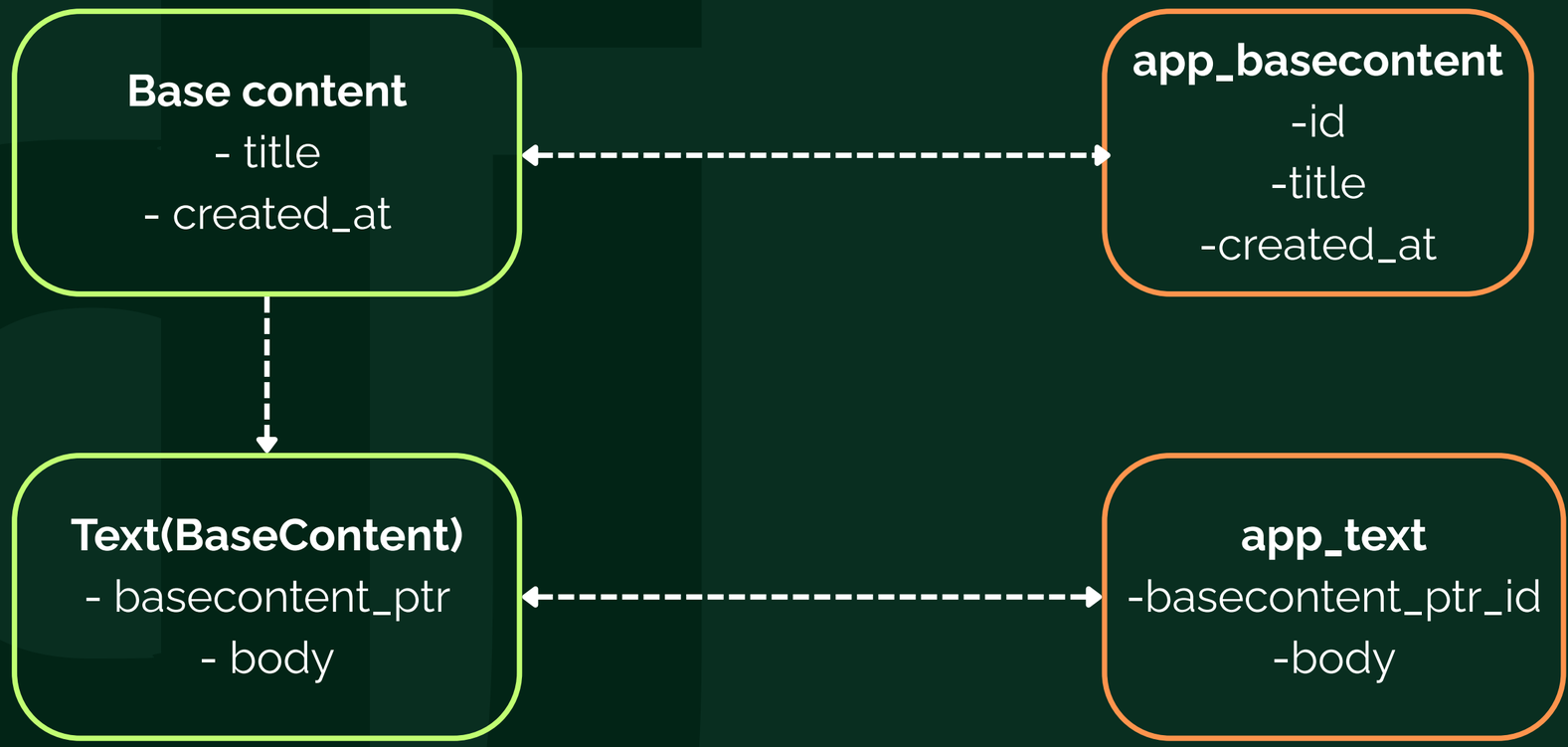
Ideal:

Para modelos jerárquicos con diferencias específicas por tipo.

```python
from django.db.import models

class BaseContent(models.Model):
    title = models.CharField(max_length=100)
    created_at = models.Datetimefield(auto_now_add = True)

class Text(BaseContent):
    body = models.TextField()
```

Modelos

Tablas Base de datos



{dev/talles}

Herencia Proxy

Uso:

Cuando quieres cambiar el comportamiento de un modelo sin cambiar su estructura, ni crear una nueva tabla.

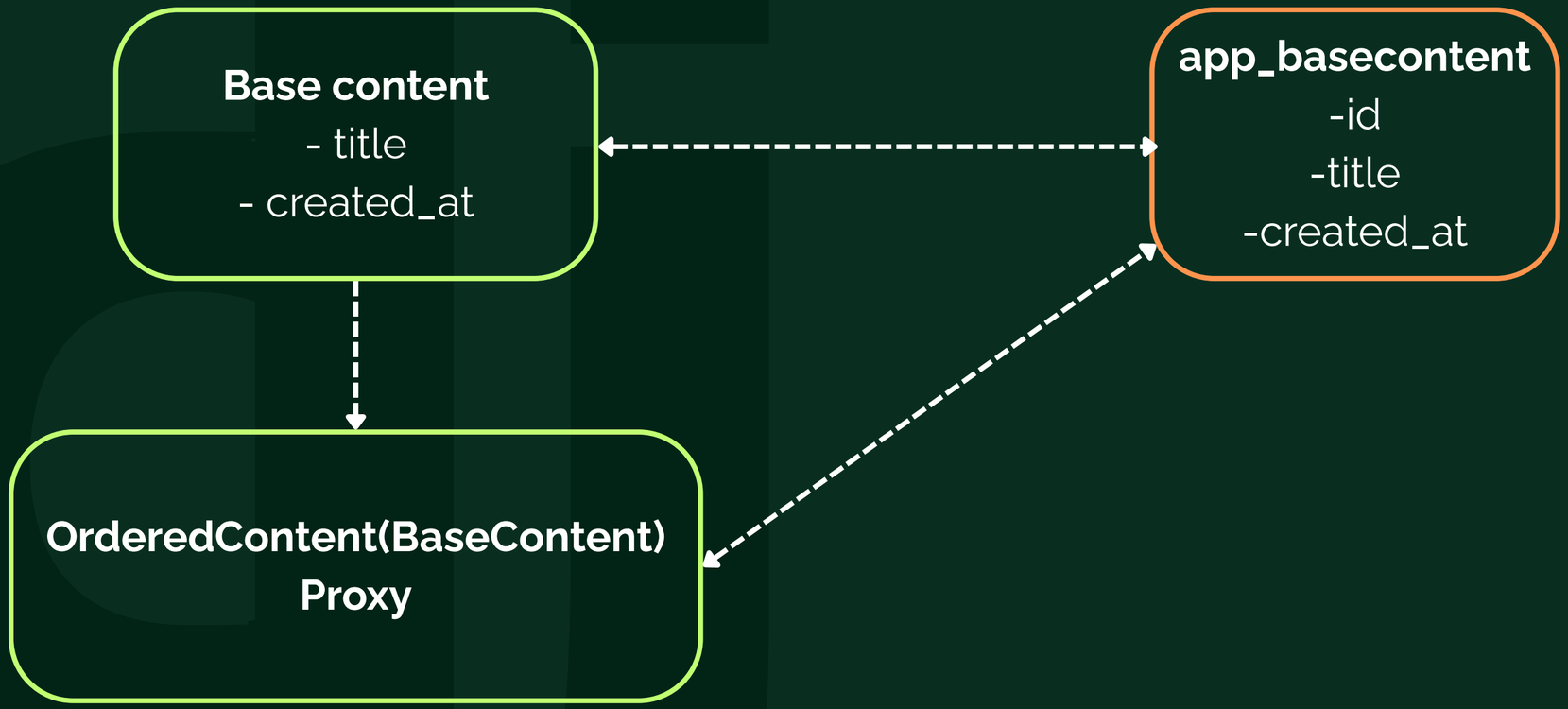
Ideal:

Para agregar ordenamientos, métodos nuevos o personalizaciones de lógica.

```python
from django.db import models

class BaseContent(models.Model):
    title = models.CharField(max_length=100)
    created_at = models.Datetimefield(auto_now_add = True)

class OrderedContent(BaseContent):
        class Meta:
                proxy = True
                ordering = ['-created_at']

        def created_delta(self):
                return timezone.now() - self.created_at
```

Modelos

Tablas Base de datos



{dev/talles}

¡Vamos a darle!