

ISS-05 — ProductType y Product.

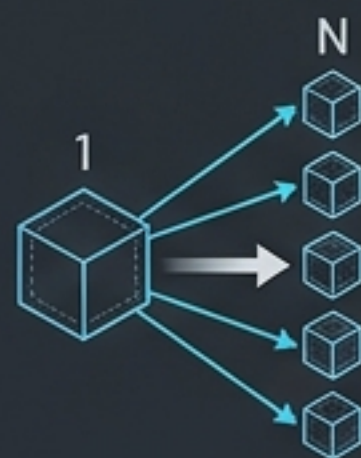
Modelado del catálogo, integridad referencial y restricciones de negocio en Django.





Objetivo

Crear la jerarquía base del catálogo de inventario.



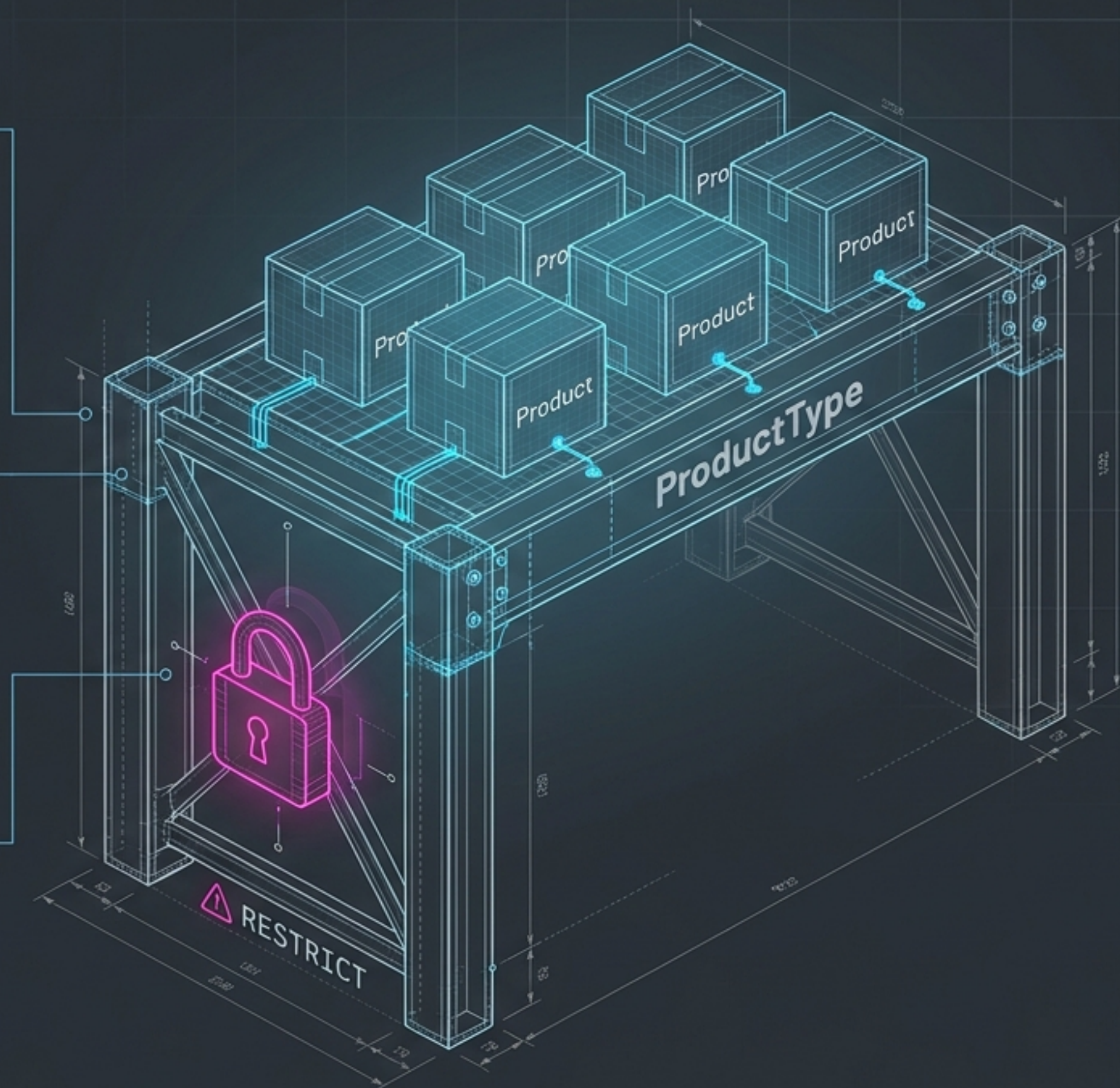
Regla de Oro

Todo **Product** pertenece obligatoriamente a un **ProductType** (Relación 1 a N).



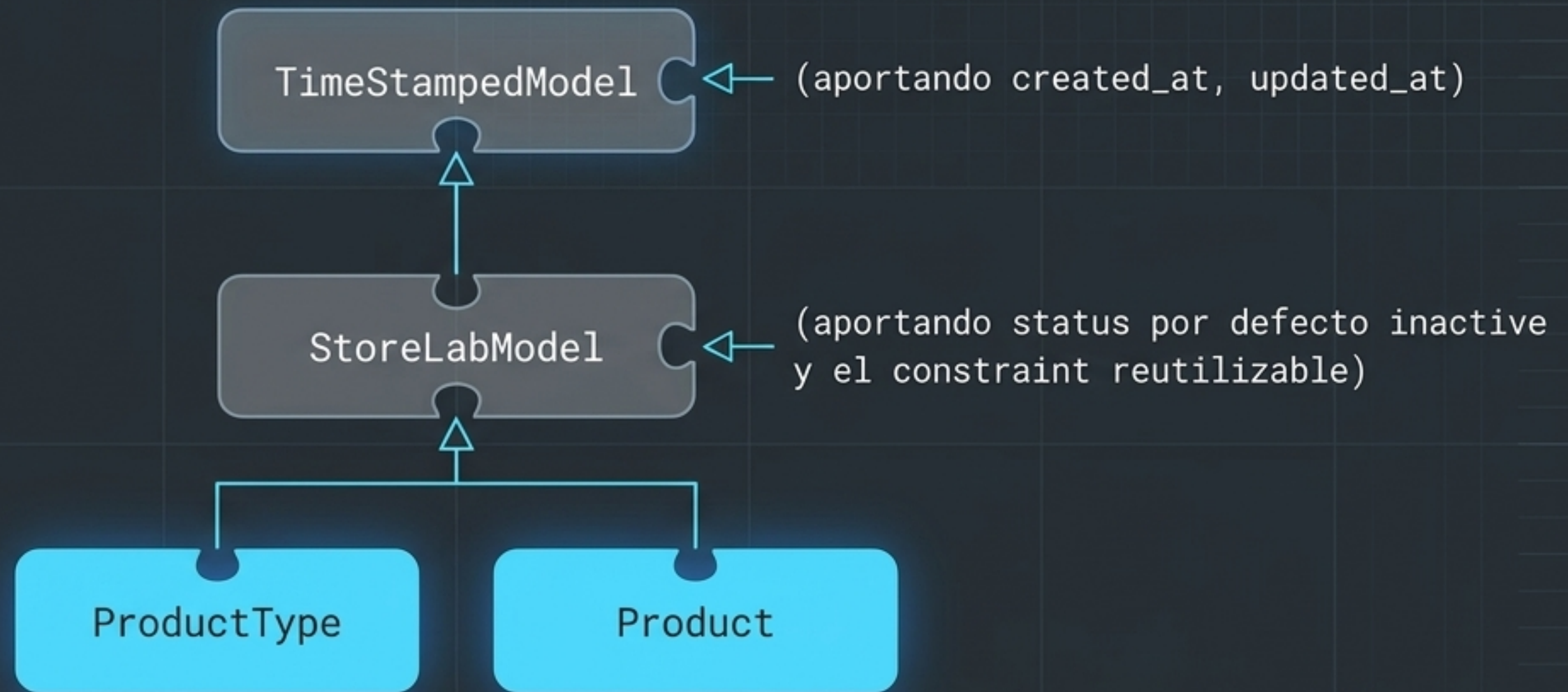
Protección de Datos

Implementar una restricción estricta de borrado (**RESTRICT**) para evitar orfandad o destrucción accidental de datos en cascada.



Los Cimientos: ¿Sobre qué estamos construyendo?

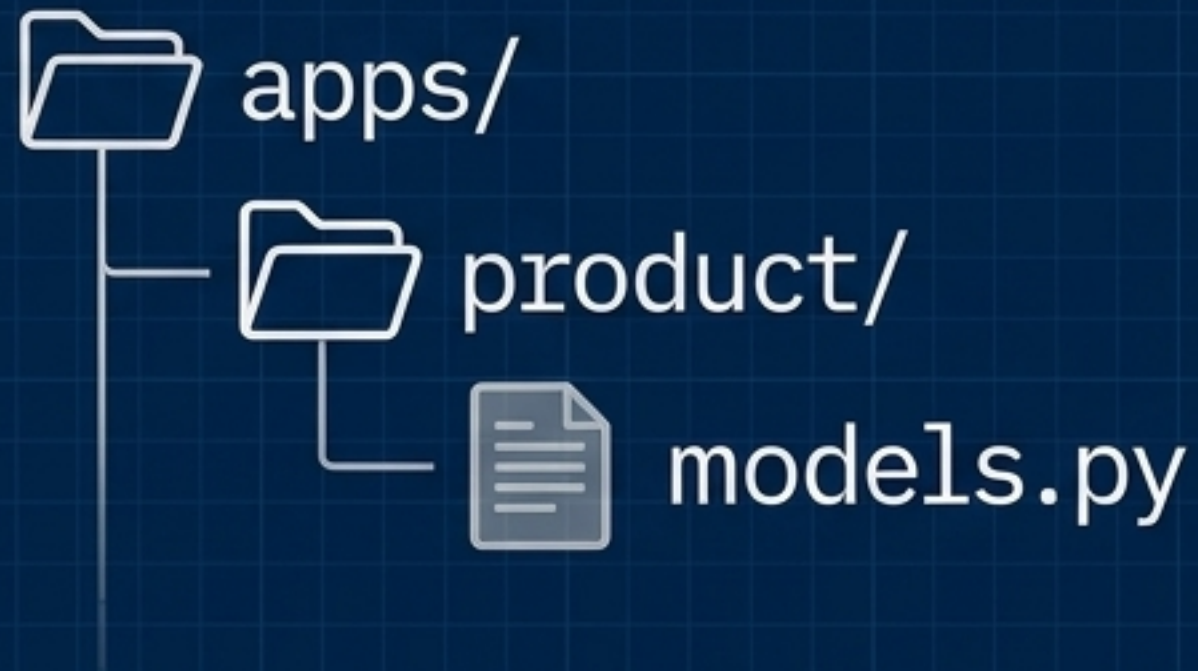
El ISS-04 nos dejó la base abstracta perfecta en `apps/common/models.py`. Nuestros modelos de negocio heredarán auditoría y control de estado sin reescribir código.



Evolución del Módulo product

Pasamos de un módulo de catálogo vacío a la implementación completa de de las entidades base en el ORM.

Antes



Después



Anatomía de ProductType: La Categoría



Herencia

Al heredar de StoreLabModel, obtiene status, created_at y updated_at gratis.



Validación de Integridad

name es único y exige mínimo 2 caracteres.

```
class ProductType(StoreLabModel):  
    name = models.CharField(max_length=100, unique=True, validators=[MinLengthValidator(2)])  
    description = models.CharField(max_length=255, null=True, blank=True)  
  
    class Meta:  
        db_table = 'product_types'  
        constraints = [StoreLabModel.status_constraint('product_types_status_valid')]
```



Nombre Físico

Fuerza el nombre en la base de datos relacional.



Restricción de Estado

Aplica el CheckConstraint heredado para asegurar un estado válido.

Anatomía de Product: El Ítem Inventariable

El Corazón Relacional

Conecta directamente con ProductType.

Reglas Financieras

price es decimal e impide precios negativos mediante MinValueValidator(0).

```
class Product(StoreLabModel):  
    product_type = models.ForeignKey(ProductType, on_delete=models.RESTRICT, db_column='product_type_id')  
    price = models.DecimalField(max_digits=12, decimal_places=2, validators=[MinValueValidator(Decimal('0'))])  
    stock = models.PositiveIntegerField(default=0)  
    # ... Meta con constraints products_status_valid y product_price_non_neg
```

Inventario Base

stock arranca en 0 por defecto y usa PositiveIntegerField.

Doble Escudo en BD

El Meta contiene CheckConstraints duales: estado válido y product_price_non_neg.

Atando los Nudos: Integridad Referencial

```
product_type = models.ForeignKey(ProductType,  
on_delete=models.RESTRICT,  
db_column='productry_type_id')
```



Parámetro 1

ProductType -> Define la tabla padre.



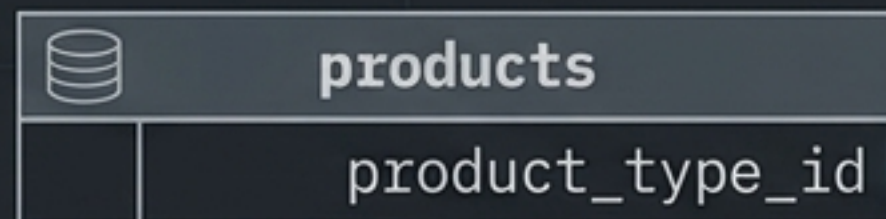
Parámetro 2

on_delete=models.RESTRICT
-> La regla de la física (Muro de contención). Lanza **RestrictedError** si se intenta romper.

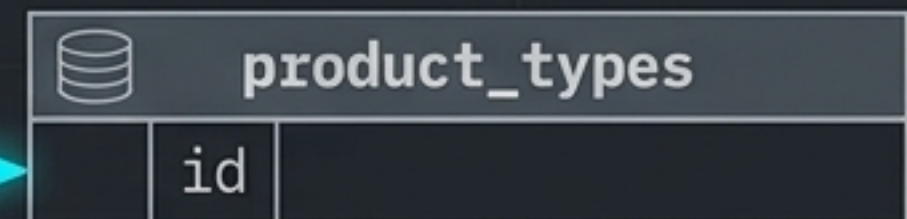


Parámetro 3

db_column='product_type_id'
-> Fuerza el nombre exacto de la columna física en SQL, evitando sufijos automáticos impredecibles.



Relational Bond



El peligro de borrar: ¿Por qué RESTRICT?

Política	Comportamiento	Veredicto
CASCADE 	Borra la categoría y elimina silenciosamente todos los productos asociados.	Peligro de negocio catastrófico en un e-commerce.
SET_NULL 	Borra la categoría y deja los productos 'huérfanos' (con FK en NULL).	Rompe nuestra regla de negocio de FK obligatoria (AC-05-01).
RESTRICT 	Impide el borrado del padre y lanza una excepción.	Nuestro héroe. Seguridad total referencial.

Materializando los Planos (CLI & BD)

El ORM traduce nuestras clases a esquemas físicos fuertemente tipados en el motor de base de datos resultante.



Zonas de Peligro: Errores Frecuentes

Usar CASCADE en lugar de RESTRICT



Destruye datos históricos masivamente.

Omitir MinValueValidator(0) en price



Permitiría la inserción de precios negativos en la lógica de la aplicación.

Olvidar db_column='product_type_id'



Genera columnas físicas con sufijos erróneos en la base de datos SQL.

No validar la longitud mínima de nombres



Permite la creación de productos llamados 'A'.

El Contrato: Criterios de Aceptación



AC-05-01

La FK es obligatoria (No acepta valores nulos).



AC-05-02

El borrado de un padre con hijos falla lanzando explícitamente **RestrictedError**.



AC-05-03

Unicidad garantizada en el nombre del producto.



AC-05-04

Valores por defecto aplicados correctamente (status arranca en inactive, stock arranca en 0).

Certificación del ISS-05 (GATE)



Estado de Auditoría Técnica: Estructura Sólida

EVI-05-01	Migración product.0001_initial generada y aplicada exitosamente en el esquema físico.	 PASS
EVI-05-02	Test automatizado captura exitosamente la excepción RestrictedError al intentar un borrado ilegal.	 PASS

Preparación para la Defensa (Q&A)



¿Por qué heredamos de `StoreLabModel` en lugar de utilizar `models.Model` directamente para estas entidades?



¿Qué ocurriría **catastróficamente** en su base de datos si utilizara **CASCADE** y un administrador borra un tipo de producto por error?



¿Por qué las **validaciones** como **MinValueValidator** deben ir en el modelo y no dejarse exclusivamente para la **validación de la vista** o el frontend?

Preparados para el escrutinio técnico. Listos para avanzar al ISS-06.