

# ISS-09 — CATÁLOGO CON MODELVIEWSET

StoreLab REST API • Abstracción `CatalogViewSet`, Enrutamiento y Suite de Pruebas

## ■ ESTRUCTURA & ARQUITECTURA

- **¿Qué construimos?:** Endpoints CRUD para `ProductType` y `Product`.
- **CatalogViewSet (Clase Base):** Agrupa ``permission_classes = [AllowAny]``, ``lookup_value_converter = 'int'``, filtrado ``?status=``` y baja lógica ``perform_destroy``.
- **Optimización ORM:** ``ProductViewSet`` usa ``queryset = Product.objects.select_related('product_type')`` para evitar consultas N+1 en SQL.

## ■■ REGLAS DE NEGOCIO Y BORDE

- **Borde HTTP camelCase:** Recibe ``productTypeId`` y responde con ``productTypeName`` sin alterar el modelo Python.
- **Baja Lógica con DELETE:** La acción HTTP DELETE cambia ``status = RecordStatus.INACTIVE`` en lugar de borrar la fila física.
- **Integridad Referencial:** Intento de borrado directo de un ``ProductType`` con productos en BD lanza ``RestrictedError``.

## ■ FLUJO DE EJECUCIÓN DEL CATÁLOGO

1. **Cliente HTTP:** Envía request a ``/api/product-types/`` o ``/api/products/`` en formato JSON camelCase.
2. **SimpleRouter:** Mapea la petición hacia ``ProductTypeViewSet`` o ``ProductViewSet`` derivando las URLs REST.
3. **CatalogViewSet:** Filtra opcionalmente por ``?status=```, ejecuta validaciones en Serializer y procesa la baja lógica.
4. **Base de Datos:** Aplica restricciones CheckConstraint (``products_status_valid``, ``product_price_non_neg``) e inmutabilidad.

| Endpoint HTTP                     | Métodos            | Acción DRF                        | Efecto / Comportamiento                                                                       |
|-----------------------------------|--------------------|-----------------------------------|-----------------------------------------------------------------------------------------------|
| <code>/api/product-types/</code>  | GET, POST          | list, create                      | Consulta tipos de producto (filtro <code>?status=``</code> ) y crea nuevos.                   |
| <code>/api/product-types//</code> | GET, PATCH, DELETE | retrieve, partial_update, destroy | Detalle, actualización y baja lógica ( <code>`status=inactive`</code> ).                      |
| <code>/api/products/</code>       | GET, POST          | list, create                      | Catálogo completo con <code>`productTypeName`</code> e ingesta <code>`productTypeId`</code> . |
| <code>/api/products//</code>      | GET, PATCH, DELETE | retrieve, partial_update, destroy | Detalle, edición parcial (ej: stock) y baja lógica.                                           |

| Criterio de Aceptación                | Mecanismo de Verificación                                  | Evidencia | Resultado |
|---------------------------------------|------------------------------------------------------------|-----------|-----------|
| AC-09-01: Alta de producto activo     | POST <code>/api/products/</code> con stock y status active | EVI-09-01 | PASS      |
| AC-09-02: Nombre duplicado rechaza    | POST duplicado responde HTTP 400 Bad Request               | EVI-09-01 | PASS      |
| AC-09-03: Defaults inactivo y stock 0 | POST omitiendo status y stock asigna defaults              | EVI-09-01 | PASS      |