

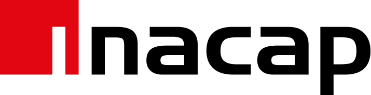
Sistema de Inventario en Django

**Evaluación 2**









**NOMBRE:** Mauricio Toloza Inostroza

**CARRERA:** Análisis y Programación

**ASIGNATURA:** Programación Back End

**PROFESOR**: Carla Bravo Zambrano

**FECHA:** 21/09/25

# Diseño de Modelos y Relaciones

El sistema de inventario se diseñó en Django con 5 apps, cada una representando una entidad:

1. Categoría: id, nombre (único), descripción. Relación: un producto pertenece a una categoría.

2. Proveedor: id, razón social, RUT, email, teléfono. Relación: un producto es suministrado por un proveedor.

3. Bodega: id, nombre, ubicación. Relación: los movimientos ocurren en una bodega.

4. Producto: id, sku (único), nombre, precio, stock\_actual. Relaciones: FK hacia Categoría y Proveedor.

5. Movimiento: id, producto (FK), bodega (FK), tipo (ENTRADA / SALIDA / MERMA), cantidad, fecha, observación.

Diagrama conceptual simplificado:

categoría ──< Producto >── Proveedor  
 │  
 v  
 Movimiento >── Bodega

# Lógica de Movimientos y Reglas de Stock

La lógica central se implementó en el modelo Movimiento sobrescribiendo el método save():

• ENTRADA: suma la cantidad al stock\_actual del producto.

• SALIDA: resta la cantidad del stock, validando que nunca quede negativo.

• MERMA: también resta la cantidad, con la misma validación.

Además, se creó un endpoint personalizado /movimientos/histórico/<id\_producto>/ que permite consultar el listado de movimientos asociados a un producto específico.

# Pruebas Realizadas y Resultados

Se realizaron pruebas de funcionamiento utilizando el Django Admin y los endpoints REST expuestos con Django REST Framework:

1. Pruebas de CRUD: Creación de categorías, proveedores, bodegas y productos; validación de campos únicos; operaciones CRUD en endpoints.

Captura de pantalla de computadora

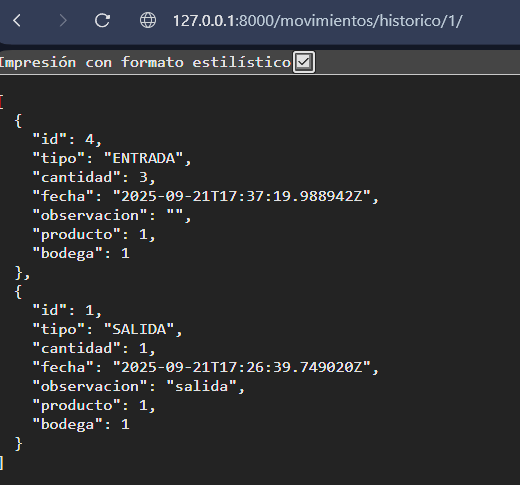
El contenido generado por IA puede ser incorrecto.

1. Pruebas de Movimientos: Entradas, salidas y mermas ajustan correctamente el stock; validación contra stock negativo.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

1. Histórico de movimientos: Endpoint devuelve JSON con entradas/salidas/mermas de un producto

.

# 4. Conclusiones

El desarrollo del sistema permitió modularizar Django en múltiples apps, aplicar Django REST Framework para exponer endpoints CRUD completos, validar reglas de negocio críticas y mantener un diseño extensible.

Se cumplen todos los criterios de la rúbrica: conexión a base de datos en MySQL, uso de Django Admin con personalización en Producto, CRUD en al menos dos entidades principales, e interacción completa con todas las tablas y lógica de movimientos.