

ISS-13 — DJANGO ADMIN

Administración y Exploración de las Tablas de Negocio

Configuración avanzada de ModelAdmin, inlines de solo lectura, autocompletado y protección de integridad transaccional.

¿Qué construiremos en el ISS-13?

Objetivos del Panel de Administración

Exploración Completa: Habilitar las 5 tablas del dominio (Client, ProductType, Product, Sale, ProductSale) en /admin/.

Gestión de Catálogo y Clientes: Formularios agrupados con fieldsets, list_display, list_filter, search_fields y list_editable.

Protección de Ventas: Modelos de venta (Sale y ProductSale) en solo lectura (readonly_fields), impidiendo creación o eliminación directa.

Optimización ORM: Uso de select_related y autocomplete_fields para evitar consultas N+1 y mejorar el rendimiento UI.

Estructura del Panel Admin

ClientAdmin: Lista editable de estados, filtro lateral y búsqueda multi-campo.

ProductTypeAdmin: Búsqueda por nombre/descripción (base para autocomplete).

ProductAdmin: Edición rápida de precio/stock, filtro por tipo y autocompletado.

SaleAdmin: Ficha inmutable con jerarquía de fecha y ProductSaleInline.

ProductSaleAdmin: Ficha inmutable con enlaces a venta y producto.

Punto de partida y Creación del Superusuario

Superusuario y Modelo de Seguridad

Comando CLI: `python manage.py createsuperuser`

`StoreUserManager.create_superuser` fuerza `status=ACTIVE` (`is_active=True`, `is_staff=True`, `is_superuser=True`). Si el superusuario quedara con `status=INACTIVE`, la regla `is_active = (status == RecordStatus.ACTIVE)` le impediría iniciar sesión en `/admin/`.

Integración previa en Settings y URLs

INSTALLED_APPS: `django.contrib.admin` ya está incluido en `config/settings/__init__.py`.

URLconf Global: `path('admin/', admin.site.urls)` ya está registrado en `config/urls.py` desde ISS-00.

Autenticación Admin: El admin de Django utiliza sesiones y cookies CSRF. No utiliza tokens JWT ni cabeceras Bearer.

Configuración de ClientAdmin (apps/client/admin.py)

Clase ClientAdmin

```
@admin.register(Client)
class ClientAdmin(admin.ModelAdmin):
    list_display = ("name", "email", "phone", "address", "status", "created_at")
    list_filter = ("status",)
    search_fields = ("name", "email", "phone", "address")
    list_editable = ("status",)
    ordering = ("name", "email")
    readonly_fields = ("created_at", "updated_at")
    fieldsets = (
        ("Información del cliente", {"fields": ("name", "email", "phone", "address")}),
        ("Estado", {"fields": ("status",)}),
        ("Auditoría", {"fields": ("created_at", "updated_at"), "classes": ("collapse",)}),
    )
```

Catálogo: ProductTypeAdmin y ProductAdmin (apps/product/admin.py)

ProductTypeAdmin

```
@admin.register(ProductType)
class ProductTypeAdmin(admin.ModelAdmin):
    list_display = ("name", "status", "created_at")
    list_filter = ("status",)
    search_fields = ("name", "description")
    list_editable = ("status",)
    ordering = ("name",)
    readonly_fields = ("created_at", "updated_at")
    fieldsets = (
        (None, {"fields": ("name", "description",
"status")}),
        ("Auditoría", {"fields": ("created_at",
"updated_at"), "classes": ("collapse",)}),
    )
```

ProductAdmin y Autocompletado

```
@admin.register(Product)
class ProductAdmin(admin.ModelAdmin):
    list_display = ("name", "product_type", "price", "stock",
"status")
    list_filter = ("status", "product_type")
    search_fields = ("name", "description")
    list_editable = ("price", "stock", "status")
    autocomplete_fields = ("product_type",)
    readonly_fields = ("created_at", "updated_at")

    def get_queryset(self, request):
        return
super().get_queryset(request).select_related("product_type")
```

Ventas Inmutables: SaleAdmin e Inlines (apps/sale/admin.py)

ProductSaleInline (Lectura)

```
class ProductSaleInline(admin.TabularInline):
    model = ProductSale
    extra = 0
    can_delete = False
    show_change_link = True
    fields = ("product", "quantity", "unit_price",
"line_total", "status")
    readonly_fields = ("product", "quantity", "unit_price",
"line_total", "status")

    def has_add_permission(self, request, obj=None):
        return False
```

SaleAdmin (Protección de Integridad)

```
@admin.register(Sale)
class SaleAdmin(admin.ModelAdmin):
    list_display = ("id", "client", "sale_date", "total",
"status")
    date_hierarchy = "sale_date"
    readonly_fields = ("client", "sale_date", "subtotal",
"tax", "discounts", "total", "status")
    inlines = (ProductSaleInline,)

    def has_add_permission(self, request):
        return False

    def has_delete_permission(self, request, obj=None):
        return False

    def get_queryset(self, request):
        return
    super().get_queryset(request).select_related("client")
```

Reglas de Arquitectura e Invariantes del Admin

Principios de Diseño del Panel Admin

- **No Crear Ventas desde Admin:** La venta exige lógica de negocio compleja (bloqueo pesimista, precio histórico, descuento de stock). Permitir crearlas en Admin omitiría esa transacción.
- **No Borrar Ventas desde Admin:** `has_delete_permission = False` evita borrados físicos que alteren la contabilidad e inventario.
- **Optimización ORM Mandatory:** `get_queryset` con `select_related('client')` en `SaleAdmin` y `select_related('product_type')` en `ProductAdmin` evita consultas N+1 al renderizar listas.
- **Requisito de Autocompletado:** `autocomplete_fields = ('product_type',)` en `ProductAdmin` exige que `ProductTypeAdmin` defina `search_fields`; de lo contrario Django lanza un `SystemCheckError`.

Criterios de Aceptación, Evidencias y GATE

Compuerta de Calidad (GATE)

CRITERIO	VERIFICACIÓN	EVIDENCIA	RESULTADO
AC-13-01	Client, ProductType, Product, Sale y ProductSale registrados	revisión admin.py	PASS
AC-13-02	Formulario de cliente agrupa campos y no pide password	client/admin.py	PASS
AC-13-03	Producto lista price y stock, no marca ni quantity	product/admin.py	PASS
AC-13-04	Venta no se agrega ni borra en admin, inline no crea líneas	sale/admin.py	PASS
AC-13-05	python manage.py check sin observaciones	EVI-13-01 (PASS)	PASS