

FASE I — BUSINESS (RUTAS ABIERTAS / SIN AUTH)

Aprendizaje-Express-ISS-06

Feature ProductType (Tipos de Producto): Replicación Consciente de la Arquitectura por Capas sin Claves Foráneas

Propósito del ISS-06 e Integración Arquitectónica

Objetivo y Replicación del Patrón

- **Ensamblaje del 2º Feature:** Demuestra que la arquitectura en 5 capas aprendida en Clientes se sostiene de forma modular.
- **Entidad Sin FKs:** ProductType representa el catálogo de familias/categorías de producto sin dependencias externas.
- **Rutas Abiertas:** Expone las operaciones CRUD sobre /api/tipos-producto en modalidad SIN AUTH.
- **DoD Completo:** Cubre Modelo, DTOs, Repository, Service, Controller, Routes, Seeder con Faker y Swagger.

Flujo de Datos por Capas

1. Cliente HTTP envía petición a /api/tipos-producto
2. ProductTypeRoutes dirige la solicitud al Controller
3. ProductTypeController valida DTOs e invoca el Service
4. ProductTypeService aplica reglas de negocio y llama al Repository
5. ProductTypesRepository interactúa con Sequelize (ProductTypeModel)
6. BD persiste cambios en la tabla product_types

Anatomía de src/features/business/product-types/

product-type.model.ts

Modelo Sequelize con campos id, name, description, status, timestamps.

dto/

Contratos de entrada/salida: Create, Update, Patch y ProductTypeResponse.

product-types.repository.ts

Acceso directo a datos mediante Sequelize ORM (findAll, findById, create, update, delete).

product-types.service.ts

Lógica de negocio: validación de duplicados, borrado lógico (status = 'inactive') e impresiones.

product-types.controller.ts

Manejo de Express req/res derivando de BaseController con AppError.

product-types.routes.ts

Definición del router de Express mapeando los verbos HTTP.

Capa de Datos: Modelo Sequelize y Sanitización DTO

Modelo Sequelize (product_types)

- **id:** INTEGER, Primary Key, Auto Increment.
- **name:** STRING(100), NOT NULL, Unique.
- **description:** TEXT, Nullable.
- **status:** ENUM('active', 'inactive'), Default 'active'.
- **createdAt / updatedAt:** DATE, Timestamps automáticos.

DTOs (Data Transfer Objects)

- **CreateProductTypeDto:** name (requerido), description (opcional).
- **UpdateProductTypeDto:** Reemplazo total de propiedades.
- **PatchProductTypeDto:** Actualización parcial de campos.
- **ProductTypeResponseDto:** Formateador estándar para la salida JSON.
- **Mapper toProductTypeResponse:** Garantiza un contrato limpio en la API.

Capa de Negocio: Repositorio y Servicio



ProductTypesRepository

- **findAll():** Retorna lista de tipos de producto.
- **findById(id):** Busca un tipo de producto por su ID.
- **findByName(name):** Comprueba existencia por nombre.
- **create(data):** Persiste un nuevo tipo de producto.
- **update(id, data):** Actualiza un registro existente.
- **deletePhysical(id):** Eliminación física de la BD.



ProductTypesService & Reglas

- **Control de Duplicados:** Valida que no exista otro tipo de producto con el mismo nombre (409 Conflict).
- **Borrado Lógico (Soft Delete):** El método deleteLogical marca status = 'inactive' en lugar de ejecutar DELETE.
- **Excepciones de Dominio:** Lanza AppError con código HTTP adecuado (400, 404, 409).
- **Aislamiento de Infraestructura:** No conoce req/res de Express ni Sequelize directamente.

Integración Global y Extensión de Infraestructura

Los 4 Puntos de Extensión en la Aplicación

- 1. Central de Rutas (src/routes/index.ts):** Importa `productTypeRoutes` y lo monta bajo el prefijo `/api/tipos-producto`.
- 2. Instanciación en App (src/config/index.ts):** Garantiza que la inicialización de rutas incluya las rutas de tipos de producto.
- 3. Runner de Seeders (src/database/seeder/index.ts):** Registra `productTypesSeeder` dentro de la secuencia de siembra de datos.
- 4. Swagger Registry (src/swagger/index.ts):** Fusiona `productTypesSwaggerSpec` en la especificación global OpenAPI 3.0.3.

Seeder con Faker y Documentación Swagger



product-types.seeder.ts

- **Generación con Faker:** Utiliza @faker-js/faker para crear categorías de producto realistas (e.g., Electrónica, Ropa).
- **Conteo Dinámico:** Lee la configuración de counts.ts (Default: 10, CLI: --product-types=N).
- **Idempotencia:** Limpia la tabla product_types antes de sembrar nuevas instancias.
- **Ejecución Independiente:** Escribe directamente mediante el modelo Sequelize fuera del servidor HTTP.



product-types.swagger.ts

- **Tag Dedicado:** Agrupa las operaciones bajo 'Tipos de Producto (SIN AUTH)'.
- **Documentación de Endpoints:** Define /api/tipos-producto (GET, POST) y /api/tipos-producto/{id} (GET, PUT, PATCH, DELETE).
- **Esquemas de Entradas y Salidas:** Documenta las propiedades y respuestas HTTP 200, 201, 400, 404, 409.
- **Integración Transparente:** Visible en /api/docs y /api/docs.json.

Criterios de Cierre GATE y Siguiente Hito

✓ Checklist de Cierre (GATE ISS-06)

- ✓ `npx tsc --noEmit` finaliza con 0 errores de compilación.
- ✓ Tabla `product_types` creada físicamente en la BD.
- ✓ Script `npm run db:seed` puebla la tabla correctamente.
- ✓ Rutas `/api/tipos-producto` expuestas y probadas en `http/`.
- ✓ Documentación en Swagger UI accesible en `/api/docs`.



¿Qué habilita el ISS-06?

- **ISS-07 (Feature Product + FK):** Construcción del CRUD de productos vinculados mediante la clave foránea `product_type_id`.
- **Validación entre Features:** `ProductsService` utilizará `ProductTypesRepository` para verificar existencia y estado activo de la categoría.
- **Relaciones en Sequelize:** Declaración explícita de `associations` (`belongsTo` / `hasMany`) entre entidades de negocio.