

# Sistema Integral de Gestión de Datos para EPNprende (PostgreSQL)

## Introducción

EPNprende es una aplicación de escritorio desarrollada para la Escuela Politécnica Nacional (EPN), cuyo objetivo es facilitar la gestión de productos y servicios de estudiantes y profesores emprendedores. A través de esta plataforma, los miembros de la comunidad politécnica pueden publicar, vender y comprar productos, interactuar entre sí y mantener un entorno controlado para sus actividades comerciales.

Para garantizar un almacenamiento seguro, estructurado y eficiente de toda la información, se implementa una base de datos relacional en PostgreSQL. Este proyecto desarrolla dicha base de datos, desde el diseño en 3FN hasta la implementación de funciones avanzadas como procedimientos, triggers, auditoría, respaldo y roles de seguridad.

## Objetivos

### Objetivo General

Diseñar e implementar una base de datos relacional en PostgreSQL que soporte de forma segura, eficiente y escalable la gestión de usuarios, productos, servicios, ofertas, notificaciones y auditoría para la aplicación EPNprende.

### Objetivos Específicos

- Modelar un esquema de datos en 3FN que incluya usuarios, perfiles, publicaciones, interacciones, ofertas y auditoría.  
- Implementar procedimientos, funciones y triggers que automaticen procesos clave y garanticen la integridad de los datos.  
- Definir roles de seguridad y aplicar cifrado para proteger datos sensibles.  
- Validar el rendimiento de la base con índices, consultas optimizadas y pruebas de carga.  
- Documentar respaldos, auditoría y mecanismos de protección contra SQL Injection.

## Alcance del Proyecto de Base de Datos

La base de datos abarca los módulos principales de la aplicación EPNprende: gestión de usuarios y perfiles, publicación de productos y servicios (con categorías, fotos, ofertas), interacciones (favoritos, comentarios), estadísticas (clicks y productos destacados) y un sistema de auditoría robusto. No incluye, en esta versión, pasarelas de pago ni integración con APIs externas.

## Requerimientos Técnicos del Proyecto

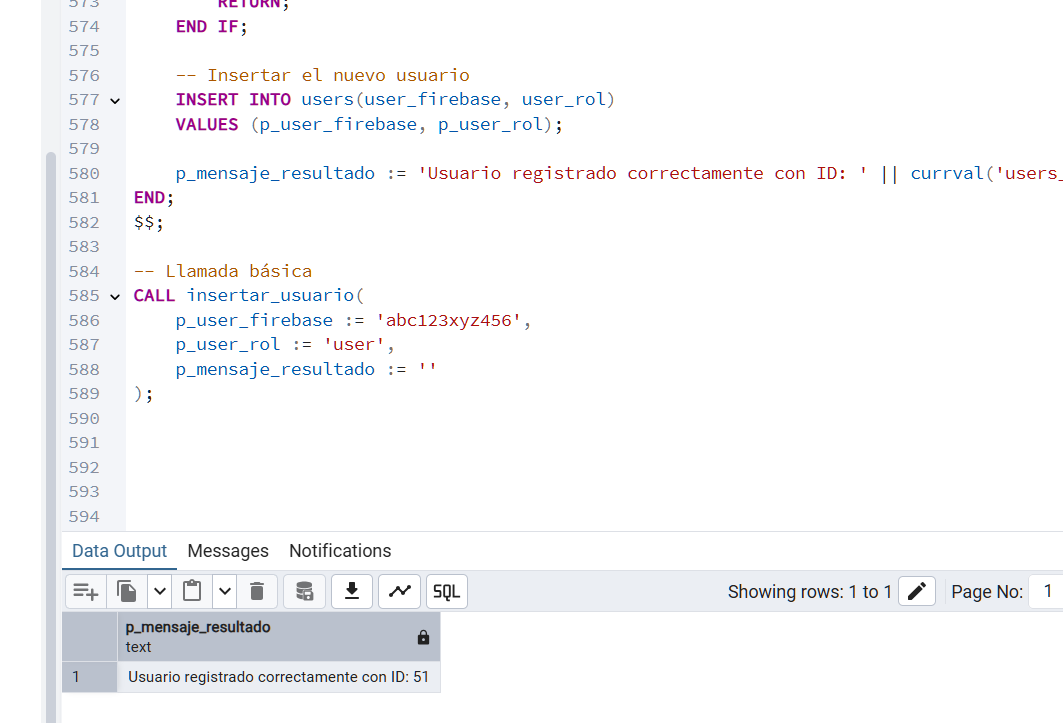
- Uso de restricciones NOT NULL, CHECK, DEFAULT, UNIQUE y claves foráneas con ON DELETE CASCADE/SET NULL según contexto.



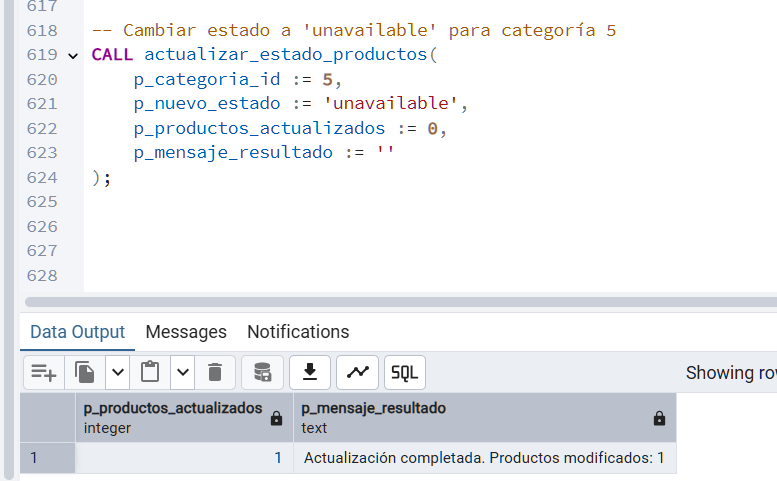
**- Creación de al menos 5 procedimientos (inserción validada, actualizaciones masivas, eliminaciones seguras, reportes, facturación (solo si aplica).**

**Creación de los procedimientos almacenados en las tablas más críticas del sistema (users, products, profiles, offers y reports).**

-- Tabla de usuarios: Procedimiento para insertar un usuario con validaciones.



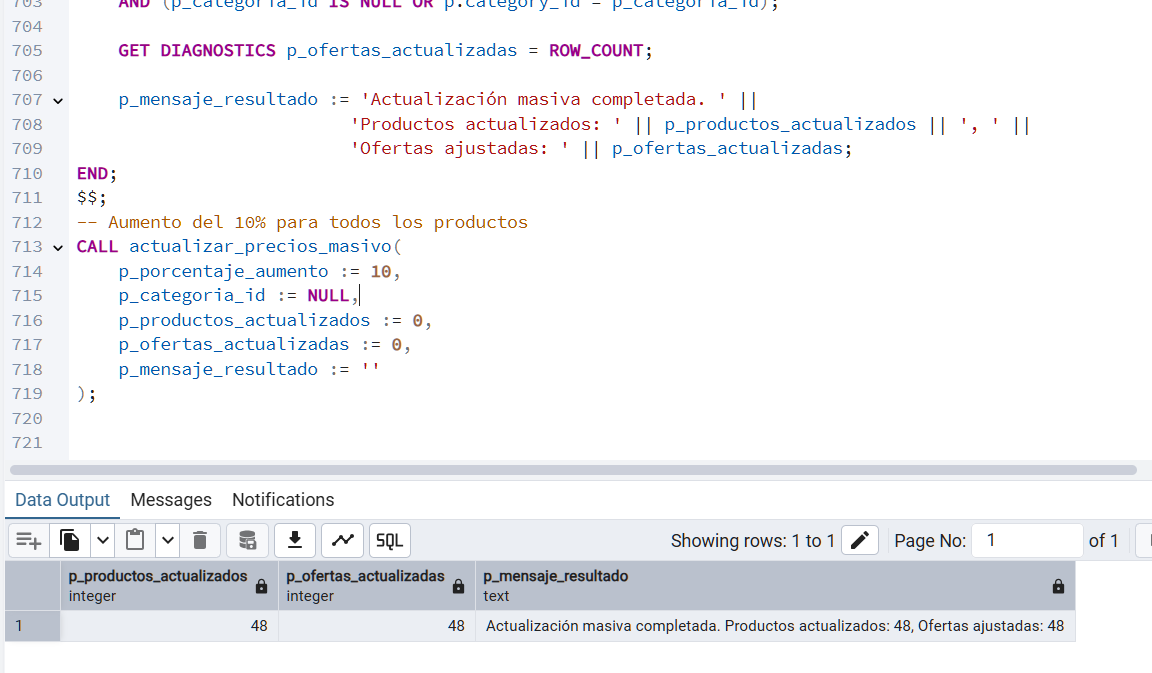
-- Tabla de productos: Procedimiento para actualizar estado de productos.



-- Tabla perfiles: Procedimiento para eliminar perfil y sus dependencias (profiles)



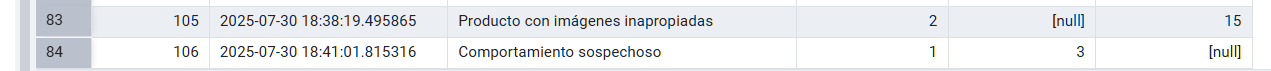
-- Procedimiento para actualización masiva de precios Tablas: (products + offers)



-- Tabla reports: Inserta un nuevo reporte, asegurando que el perfil que lo genera existe y que se reporta al menos un perfil o producto, cumpliendo con las restricciones de integridad definidas.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

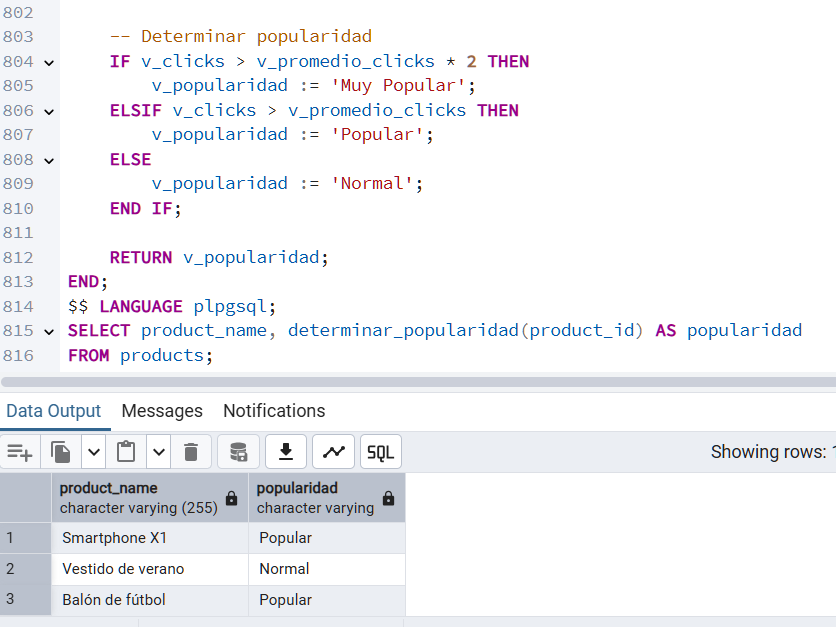


**- Implementación de 3 funciones (cálculos de métricas, descuentos, estados de usuario).**

-- Función para calcular el descuento promedio de ofertas.



-- Función para determinar la Popularidad de un producto.



-- Función para Validar una nueva oferta.

Texto

El contenido generado por IA puede ser incorrecto.

**- Implementación de 3 triggers (auditoría de cambios, control de stock, notificaciones).**

**- Definición de roles y privilegios (admin\_db, auditor\_db, operador\_db, usuario\_final).**

El diseño de roles y privilegios en la base de datos EPNprendeDB se ha estructurado para garantizar seguridad, eficiencia y segregación de funciones. Se implementaron cuatro roles principales con distintos niveles de acceso, asignados a esquemas específicos para controlar el alcance de las operaciones que cada usuario puede realizar. Esta arquitectura sigue el principio de mínimo privilegio, otorgando solo los permisos estrictamente necesarios para cada función.

**Roles y sus Funcionalidades**

Rol Administrador (admin\_db)

Este rol tiene privilegios completos sobre toda la base de datos, incluyendo todos los esquemas (config, operaciones, auditoria y publico). Puede crear, modificar y eliminar cualquier objeto, así como gestionar otros roles. Es el único con capacidad para alterar la estructura de la base de datos o realizar operaciones sensibles. Su configuración incluye una ruta de búsqueda optimizada que prioriza los esquemas administrativos.

**Rol Auditor (auditor\_db)**

Diseñado exclusivamente para actividades de monitoreo, este rol tiene permisos de solo lectura sobre el esquema de auditoría. Puede consultar los registros de logs y tablas de trazabilidad, pero no tiene acceso a modificar datos ni a visualizar información en otros esquemas. Esta restricción asegura que las actividades de auditoría sean transparentes, pero no interferentes.

**Rol Operador (operador\_db)**

Dirigido al personal operativo, este rol puede realizar operaciones básicas CRUD (Crear, Leer, Actualizar) en el esquema de operaciones, pero específicamente se le ha revocado el permiso DELETE para prevenir eliminaciones accidentales o malintencionadas. Se ha implementado un límite de conexiones simultáneas para evitar sobrecargas, y su ruta de búsqueda está configurada para priorizar el esquema operaciones.

**Rol Usuario Final (usuario\_final)**

Con el acceso más restrictivo, este rol solo puede realizar consultas de lectura (SELECT) en el esquema publico, donde residen los datos destinados a visualización general. No tiene permisos para acceder a esquemas sensibles ni para ejecutar operaciones de escritura. Al igual que el rol operador, tiene un límite de conexiones para gestionar recursos.

**Configuraciones Clave en la base de datos**

**Límite de Conexiones Simultáneas (CONNECTION LIMIT)**

Esta configuración restringe el número máximo de conexiones que cada rol puede establecer simultáneamente con la base de datos. Por ejemplo, el operador\_db tiene un límite de 20 conexiones, mientras que el usuario\_final está limitado a 10. Esto previene que un usuario agote los recursos del servidor mediante múltiples conexiones, ya sea por error o por actividades maliciosas. Cuando se alcanza el límite, cualquier intento adicional de conexión será rechazado automáticamente.

**Ruta de Búsqueda (search\_path)**

La ruta de búsqueda define el orden en que PostgreSQL busca objetos cuando no se especifica explícitamente el esquema. Para el usuario\_final, por ejemplo, está configurada para buscar primero en el esquema público y luego en public. Esto permite que las consultas puedan omitir el nombre del esquema (ej: "SELECT \* FROM productos" en lugar de "SELECT \* FROM publico.productos"), mejorando la usabilidad. Además, al controlar esta ruta, se asegura que los usuarios accedan prioritariamente a los esquemas designados para su rol, reduciendo riesgos de acceso accidental a datos sensibles.

**Estructura de Esquemas y su Relación con los Roles**

La base de datos organiza los objetos en esquemas especializados:

* **config**: Parámetros del sistema, accesible solo por admin\_db.
* **operaciones**: Datos transaccionales del negocio, accesible por operador\_db y admin\_db.
* **auditoria**: Registros de seguimiento, accesible por auditor\_db y admin\_db.
* **publico**: Datos de consulta pública, accesible por usuario\_final y admin\_db.

Esta segmentación permite un aislamiento lógico de los datos según su sensibilidad y uso. Adicionalmente, se aplicaron restricciones explícitas para evitar que roles no autorizados accedan a esquemas fuera de su alcance.

**Beneficios de la Implementación**

**Seguridad**

* **Segregación estricta** de funciones mediante roles especializados.
* **Acceso mínimo necesario**, reduciendo superficies de ataque.
* **Protección contra sobrecargas** mediante límites de conexión.

**Rendimiento**

* **Optimización de consultas** mediante rutas de búsqueda personalizadas.
* **Distribución controlada** de recursos del servidor.

**Mantenibilidad**

* **Organización clara** de objetos por esquemas.
* **Facilidad de auditoría** gracias a permisos.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

**Credenciales especificadas por fines educativos**

1. Rol Administrador
   * 1. Clave: EPNprende@Admin2025!
2. Rol Auditor
   * 1. Clave: EPNprende@Auditor2025\*
3. Rol Operador
   * 1. Clave: EPNprendeOperador@EPN2025%
4. Rol usuario
   * 1. Clave: EPNprende@Usuario2025#

**- Uso de pgcrypto para cifrado de contraseñas y correos, con digest (SHA256) y cifrado simétrico.**

**Proceso de Cifrado de IDs de Firebase en PostgreSQL**

PostgreSQL ofrece la extensión pgcrypto para operaciones criptográficas, permitiendo:

* Hash seguro (SHA-256) para contraseñas de los usuarios (id de los usuarios de firebase).

Cifrado de IDs de Firebase en la Tabla users: Esta acción será realizada en el esquema de auditoría ya que esta sección, está diseñada para manejar datos sensibles y operaciones críticas. Por defecto solo es accesible por admin\_db y auditor\_db (según tu configuración previa en la sección de roles y privilegios).

**1. Objetivo del Script**

Implementar un sistema seguro de almacenamiento para los IDs de Firebase mediante cifrado AES, eliminando los datos sensibles en texto plano y asegurando su accesibilidad solo para roles autorizados.

**2. Actividades Realizadas**

**A. Preparación del Entorno**

**Creación del esquema**operaciones

* + Se estableció con permisos exclusivos para admin\_db y acceso limitado para operador\_db.

**Activación de la extensión**pgcrypto

* + Requerida para funciones de cifrado avanzado (pgp\_sym\_encrypt/decrypt).

**Configuración de permisos en el esquema**auditoria

* + Se restringió el acceso a PUBLIC y se otorgó acceso exclusivo a admin\_db.

**B. Modificación de la Estructura de Datos**

**Adición de columnas para cifrado**

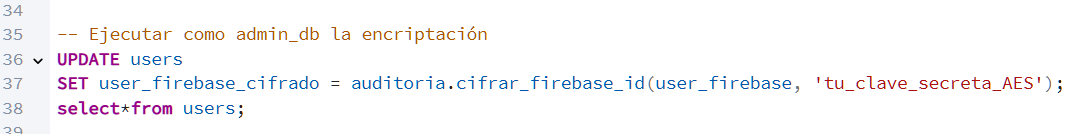
* + user\_firebase\_cifrado (BYTEA): Almacena los IDs cifrados con AES.
  + user\_firebase\_iv (BYTEA): Reservada para IV (no utilizada finalmente).

**Función de cifrado**auditoria.cifrar\_firebase\_id

* + Recibe texto plano y una clave, devuelve el contenido cifrado.
  + Configurada como SECURITY DEFINER para ejecutarse con privilegios de administrador.

**C. Proceso de Cifrado**

**Actualización masiva de datos en la tabla**



Todos los IDs se cifraron usando una clave simétrica.

**Verificación de resultados**

* + Consulta SELECT \* FROM users para validar la transformación.

Tabla

El contenido generado por IA puede ser incorrecto.



**D. Gestión de Seguridad Post-Cifrado**

**Backup de datos originales**

* + Se creó la tabla auditoria.backup\_firebase\_ids para preservar los IDs en texto plano (con fines de auditoría/reversión).

**Eliminación de datos sensibles**

* + Se eliminó la restricción UNIQUE en user\_firebase.
  + Se eliminó la columna user\_firebase (texto plano).
  + La columna cifrada se renombró a user\_firebase para mantener compatibilidad.

**Eliminación del IV no utilizado**

* + Se verificó que user\_firebase\_iv estaba vacía (NULL) y se eliminó.

**Flujo del Proceso**

Diagrama

El contenido generado por IA puede ser incorrecto.

**Medidas de Seguridad Implementadas**

* **Cifrado AES-256**: A través de pgp\_sym\_encrypt.
* **Backup en esquema restringido**: Solo accesible por admin\_db.
* **Eliminación de datos redundantes**: Minimiza superficies de ataque.

**Notas Clave**

* **El IV no se usó**: PostgreSQL lo gestiona internamente en pgp\_sym\_encrypt.
* **El esquema auditorio** centraliza todas las operaciones sensibles.
* **El rol**operador\_db solo tiene acceso a datos cifrados, no a funciones de descifrado.

**Esquemas en la Base de Datos**

**1. Esquema**operaciones

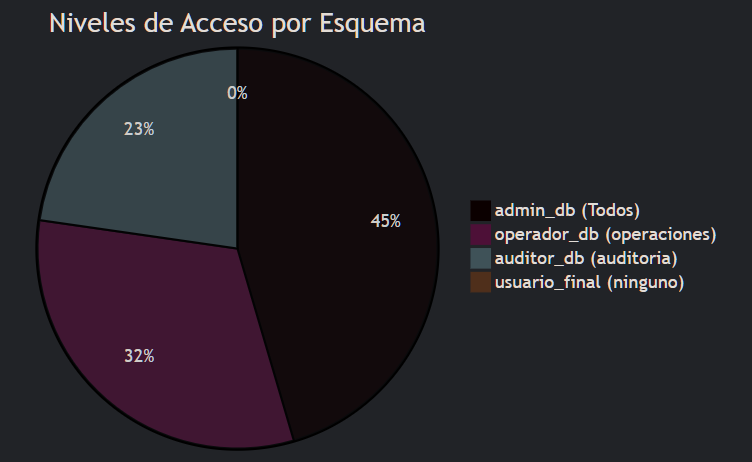
* **Propósito**: Contiene todas las tablas y datos transaccionales del negocio (ej: información de usuarios, productos, ventas).
* **Acceso**:
  + neondb\_owner: Permisos completos (CREAR/MODIFICAR/ELIMINAR).
  + operador\_db: Solo USAGE + permisos CRUD limitados (sin DELETE).

**2. Esquema**auditoria

* **Propósito**: Almacena registros de seguridad y cambios críticos:
  + Logs de cifrado (backup\_firebase\_ids).
  + Funciones de encriptación (cifrar\_firebase\_id).
* **Acceso**:
  + Solo admin\_db y auditor\_db (este último con permisos de solo lectura).

**3. Esquema**public**(por defecto)**

* **Propósito**:
  + Objetos compartidos o temporales.
  + Acceso restringido, solo permitido a al administrador y el auditor.



**Configuración SSL/TLS para PostgreSQL en Windows**

**- Creación de índices simples y compuestos para optimizar consultas.  
- Evidencia de respaldo y restauración (en caliente y frío).  
- Simulación y mitigación de ataques de SQL Injection mediante consultas preparadas y validaciones.**

## Modelo Conceptual y Lógico

El modelo de datos se basa en un DER con relaciones 1:N y N:M, cubriendo usuarios, perfiles, productos, categorías, ofertas,favoritos, comentarios, estadísticas y auditoría. El modelo relacional asegura integridad referencial completa, con claves foráneas en todas las relaciones.

Diagrama

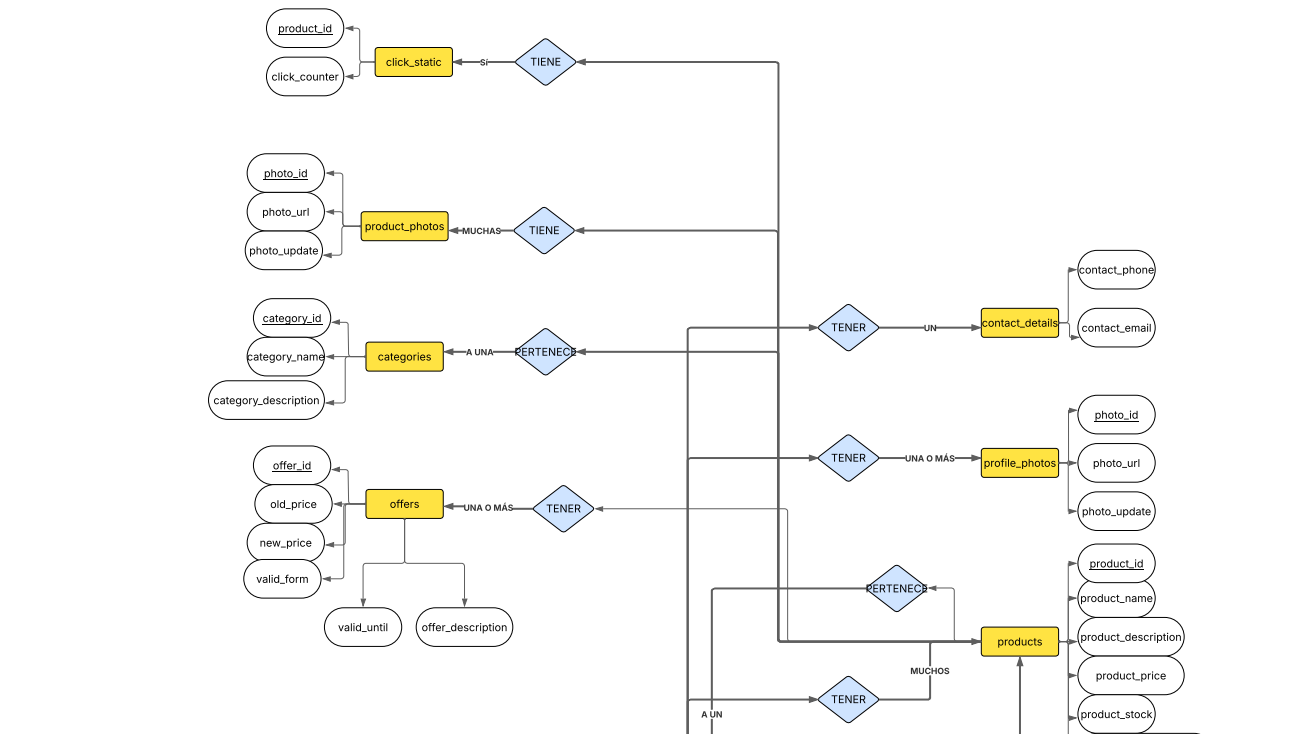
El contenido generado por IA puede ser incorrecto.

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

## Normalización (3FN)

Todas las tablas fueron diseñadas siguiendo la tercera forma normal (3FN), eliminando redundancias y asegurando que cada atributo dependa únicamente de la clave primaria. Por ejemplo, los datos de contacto se gestionan en una tabla separada relacionada 1:1 con perfiles, y las fotos se dividen en fotos\_perfil y fotos\_producto para mantener integridad referencial.

## Implementación SQL

Se incluye los scripts completos que contienen:  
- Definición de tablas con restricciones y relaciones.  
- Procedimientos almacenados (inserciones validadas, reportes, transacciones controladas).  
- Funciones (cálculos de métricas, descuentos, estados).  
- Triggers para auditoría y automatización de procesos.  
- Roles y privilegios definidos para control de acceso.

## Validación y Pruebas

Se realizaron pruebas para validar:  
- Rendimiento de consultas antes y después de aplicar índices.  
- Seguridad ante intentos de SQL Injection.  
- Respaldo y restauración de la base en caliente y frío.  
- Auditoría de cambios en registros clave mediante triggers.

## Conclusiones y Recomendaciones

La base de datos diseñada para EPNprende en PostgreSQL cumple con los requerimientos técnicos exigidos: está normalizada, es segura, mantiene integridad referencial, soporta automatización mediante procedimientos y triggers, y fue validada en rendimiento y seguridad. Se recomienda a futuro:  
- Integrar métricas avanzadas con herramientas de análisis (Power BI o Tableau).  
- Incluir pasarelas de pago para ampliar funcionalidad.  
- Implementar un sistema de copias de seguridad automatizado.

- Mejora en el proceso y definición de la estructura del sistema de la base datos.

- Definir requerimientos iniciales para el sistema en el que se va a usar la base de datos.

- Hacer un análisis previo antes de realizar pruebas de rendimiento y seguridad.