## Base de datos.

Se ha optado por un modelo relacional utilizando PostgreSQL 15.2, como se evidencia en el archivo docker-compose.yml. Este modelo es adecuado debido a las relaciones claras y bien definidas entre los productos, las órdenes y sus items.

### BD\_Order\_Management

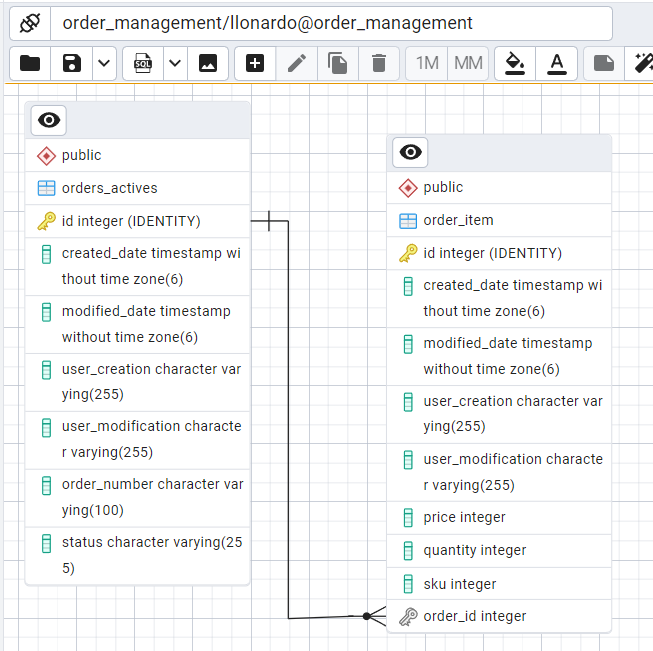


Ilustración 1 - BD\_Order\_Managment

**Tablas:**

* **orders\_actives:**
  + id: Clave primaria autoincremental.
  + order\_number: Número único de la orden.
  + status: Estado de la orden (e.g., "PENDING", "COMPLETED", "SHIPPED").
  + created\_date, modified\_date, user\_creation, user\_modification: Campos de auditoría para rastrear la creación y modificación de la orden.
* **order\_item:**
  + id: Clave primaria autoincremental.
  + quantity: Cantidad del producto en la orden.
  + price: Precio del producto en el momento de la orden.
  + sku: SKU del producto ordenado.
  + order\_id: Clave foránea que referencia orders\_actives.id, estableciendo la relación entre la orden y sus items.
  + created\_date, modified\_date, user\_creation, user\_modification: Campos de auditoría para rastrear la creación y modificación del item de la orden.

**Relación:**

* **Uno a Muchos:** Una orden (orders\_actives) puede tener muchos items (order\_item). Esta relación se implementa mediante la clave foránea order\_id en la tabla order\_item.

**Documentación de la Base de Datos del Microservicio de Órdenes**

**Contexto:**

Esta documentación describe el diseño de la base de datos para el microservicio de órdenes dentro de una aplicación de e-commerce con arquitectura de microservicios. El diseño se centra en la eficiencia, funcionalidad y escalabilidad del sistema de gestión de pedidos.

**Diseño de la Base de Datos:**

Se ha optado por un modelo relacional utilizando PostgreSQL 15.2, como se evidencia en el archivo docker-compose.yml. Este modelo es adecuado debido a las relaciones claras y bien definidas entre las órdenes y sus items.

**Tablas:**

* **orders\_actives:**
  + id: Clave primaria autoincremental.
  + order\_number: Número único de la orden.
  + status: Estado de la orden (e.g., "PENDING", "COMPLETED", "SHIPPED").
  + created\_date, modified\_date, user\_creation, user\_modification: Campos de auditoría para rastrear la creación y modificación de la orden.
* **order\_item:**
  + id: Clave primaria autoincremental.
  + quantity: Cantidad del producto en la orden.
  + price: Precio del producto en el momento de la orden.
  + sku: SKU del producto ordenado.
  + order\_id: Clave foránea que referencia orders\_actives.id, estableciendo la relación entre la orden y sus items.
  + created\_date, modified\_date, user\_creation, user\_modification: Campos de auditoría para rastrear la creación y modificación del item de la orden.

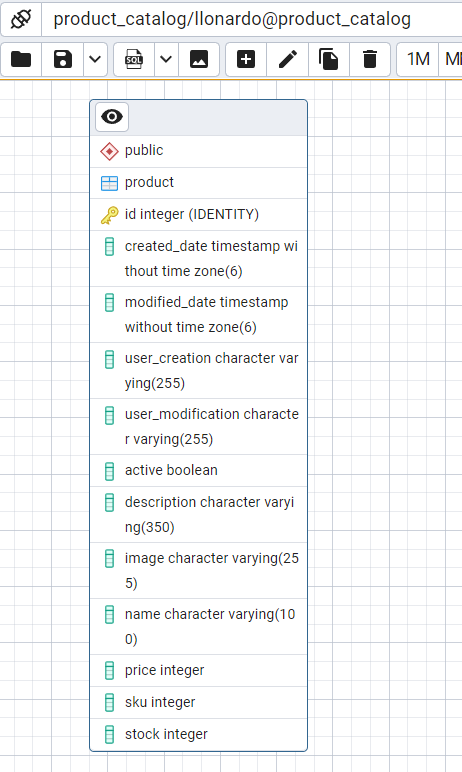
**Relación:**

* **Uno a Muchos:** Una orden (orders\_actives) puede tener muchos items (order\_item). Esta relación se implementa mediante la clave foránea order\_id en la tabla order\_item.

**Justificación del Diseño:**

* **Eficiencia:**
  + El modelo relacional es eficiente para consultas que involucran relaciones entre órdenes e items.
  + Los índices en las claves primarias y foráneas optimizan las búsquedas y uniones.
* **Funcionalidad:**
  + El diseño captura la información esencial para la gestión de órdenes, incluyendo el estado, los items, y la información de auditoría.
  + La relación uno a muchos permite una gestión flexible de los items de la orden.
* **Escalabilidad:**
  + **Microservicios:** Al separar la gestión de órdenes en un microservicio con su propia base de datos, se facilita la escalabilidad horizontal. Se pueden desplegar múltiples instancias del microservicio para manejar un mayor volumen de pedidos.
  + **Sharding:** En el futuro, si el volumen de datos crece significativamente, se puede implementar sharding en la tabla orders\_actives para distribuir la carga en múltiples bases de datos.
  + **Docker:** El uso de Docker facilita el despliegue y la escalabilidad del microservicio.

### Product\_Catalog



**Tabla:**

* **product:**
  + id: Clave primaria autoincremental.
  + name: Nombre del producto.
  + description: Descripción detallada del producto.
  + price: Precio del producto.
  + stock: Cantidad disponible en inventario.
  + image: URL de la imagen del producto.
  + sku: Número único que identifica al producto (SKU).
  + active: Indicador booleano que señala si el producto está activo en el catálogo.
  + created\_date, modified\_date, user\_creation, user\_modification: Campos de auditoría para rastrear la creación y modificación del producto.

**Justificación del Diseño:**

* **Eficiencia:**
  + El modelo relacional permite un almacenamiento eficiente de la información de los productos y sus atributos.
  + Los índices en la clave primaria (id) y en el SKU (sku) optimizan las búsquedas. Y permite exponer un codigo único del producto sin comprometer la información de la BD.
  + El campo active permite controlar la visibilidad del producto en el catálogo sin necesidad de eliminarlo de la base de datos.

### 2.3 Consideraciones generales

* EL uso de Docker se pensó para facilita el despliegue y la escalabilidad de los microservicios y la BD, permitiendo una gestión eficiente de los contenedores.
* Para el microservicio de Órdenes, se puede implementar un mecanismo de bloqueo optimista para manejar la concurrencia al actualizar las órdenes, previniendo conflictos y asegurando la consistencia de los datos.
* Ambos esquemas de bases de datos están normalizados para evitar la redundancia de datos y asegurar la integridad de los mismos.
* Se pueden crear índices en las columnas clave de cada tabla para optimizar las consultas. En el microservicio de Órdenes, se recomienda indexar order\_id y sku en la tabla order\_item. En el microservicio de Catálogo de Productos, se deben indexar columnas como name y sku.
* Si el volumen de datos crece significativamente, se puede implementar sharding en de diferentes formas. En la tabla product para distribuir la carga en múltiples bases de datos, basándose en categorías de productos o rangos de SKU. Para las tablas de ordenes se puede fragmentar por el estado de las órdenes.