PROYECTO FINAL (Fase 1: Diseño y Configuración Inicia)

1. Toma de requerimientos

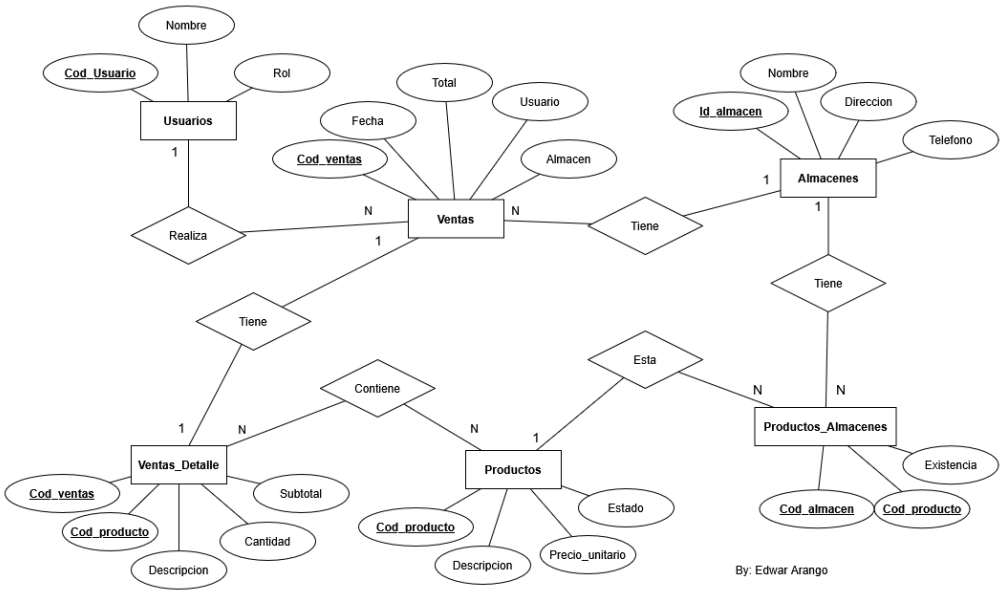
Diagramas E-R

El diseño de este modelo entidad-relación (ER) se realizó con el objetivo de representar de manera clara y organizada el funcionamiento de un sistema de ventas. Para ello, primero analizamos cómo interactúan los distintos elementos dentro del proceso de venta, desde los usuarios que realizan las transacciones hasta los productos que se venden y los almacenes donde se almacenan.

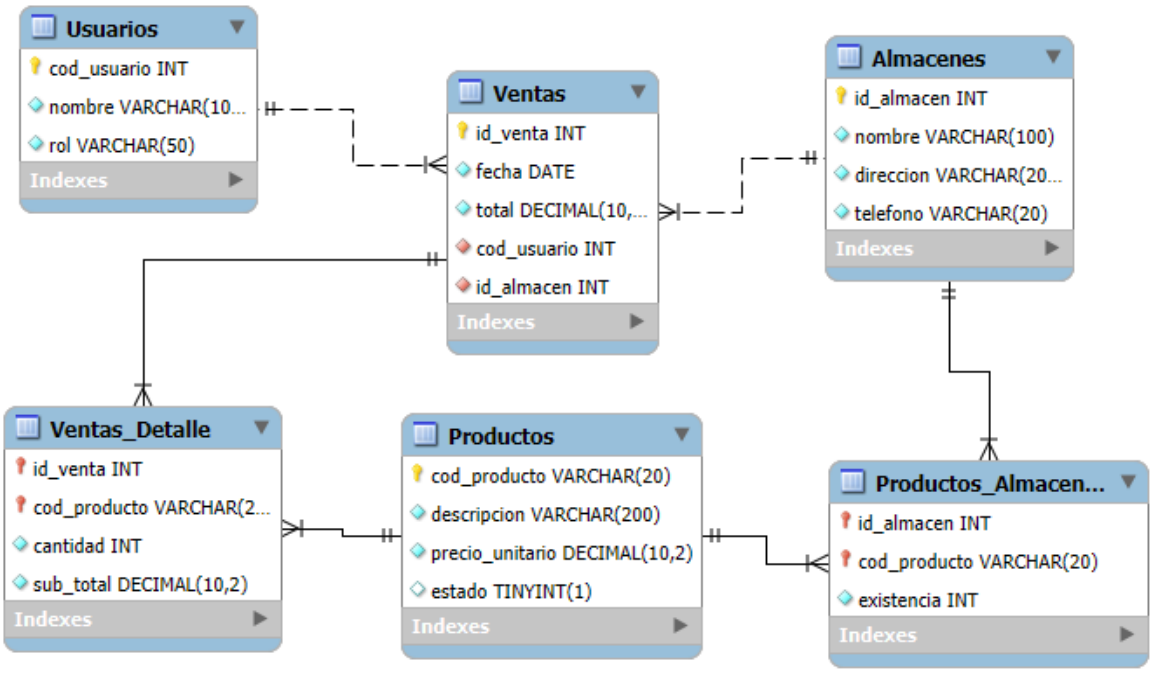
El primer paso fue identificar las entidades principales que forman parte del sistema. Se incluyeron los usuarios, que representan a las personas que realizan las ventas; las ventas, que registran cada transacción; los productos, que son los artículos disponibles para la venta; y los almacenes, que guardan el stock de productos. Además, se definieron entidades adicionales como Ventas_Detalle, que almacena la información específica de cada producto vendido dentro de una venta, y Productos_Almacenes, que permite controlar la cantidad de productos en cada almacén.

También se definieron los atributos clave de cada entidad, asegurando que el modelo permita almacenar toda la información necesaria para gestionar el sistema de manera eficiente.

El resultado es un diseño estructurado que permite administrar las ventas, los productos y su inventario en los distintos almacenes. Con este modelo, se garantiza que la información esté bien organizada y se pueda acceder fácilmente para realizar consultas, reportes o cualquier operación dentro del sistema.



Diseño lógico del sistema



2. Diseño de la base de datos

Creación de tablas

Según el diseño del diagrama ER se realizó un script para creación de tablas y sus relaciones, funciones, procedimientos y triggers necesarios para lograr la relación entre tablas para la gestión correcta de los datos, se adjuntará en un archivo sql (Script Proyecto Final DB gestion_productos).

Relaciones

Almacenes

- Ventas: id_almacen (ubicación de la venta).
- Productos_Almacenes: id_almacen (almacén donde se almacena el producto).

Usuarios

- Ventas: cod_usuario (quién realizó la venta).

Productos

- Ventas_Detalle: cod_producto (productos vendidos).
- Productos_Almacenes: cod_producto (productos almacenados).

Ventas

- Usuarios: cod_usuario (usuario que realizó la venta).
- Almacenes: id_almacen (almacén de origen).
- Ventas_Detalle: id_venta (detalles de la venta).

Ventas_Detalle

- Ventas: id_venta (venta asociada).
- Productos: cod_producto (producto vendido).

Productos_Almacenes

- Almacenes: id_almacen (almacén donde se almacena).
- Productos: cod_producto (producto almacenado).

Tabla de relaciones 1:N

Tabla Origen	Tabla Destino	Clave Foránea	Descripción
Usuarios	**Ventas**	cod_usuario	Un usuario puede realizar múltiples ventas.
Almacenes	**Ventas**	id_almacen	Un almacén puede ser origen de múltiples ventas.
Almacenes	**Productos_Almacenes**	id_almacen	Un almacén puede almacenar múltiples productos.
Productos	**Productos_Almacenes**	cod_producto	Un producto puede estar en múltiples almacenes.
Ventas	**Ventas_Detalle**	id_venta	Una venta puede incluir múltiples productos.
Productos	**Ventas_Detalle**	cod_producto	Un producto puede venderse en múltiples ventas.

Tabla de relaciones N:M

Tabla Intermedia	Tabla 1	Tabla 2	Clave Compuesta
Productos_Almacenes	**Almacenes**	**Productos**	id_almacen, cod_producto
Ventas_Detalle	**Ventas**	**Productos**	id_venta, cod_producto

Claves Foráneas:

- **Ventas**: cod_usuario y id_almacen **deben existir** en Usuarios y Almacenes.
- **Ventas_Detalle**: id_venta y cod_producto **deben existir** en Ventas y Productos.
- **Productos_Almacenes**: id_almacen y cod_producto **deben existir** en Almacenes y Productos.

Claves Compuestas:

- **Productos_Almacenes**: Combina id_almacen + cod_producto para evitar duplicados.
- **Ventas_Detalle**: Combina id_venta + cod_producto para evitar duplicados.

Funciones

F1 ObtenerCodigoUsuarioActual

Propósito: Obtener el código del usuario actual conectado a la base de datos.

Funcionamiento:

- Extrae el nombre del usuario:
  - Usa USER() para obtener el usuario actual (ej.: root@localhost)
  - SUBSTRING_INDEX extrae la parte antes del @ (ej.: root)
- Busca el código en la tabla Usuarios:
  - Realiza una consulta a la tabla Usuarios usando el nombre como filtro
  - Si no encuentra coincidencia, devuelve un mensaje de error

F2 VerificarProducto

Propósito: Verificar si un producto existe en la tabla Productos.

Funcionamiento:

- Consulta directa:
  - Usa EXISTS para verificar la existencia del producto
  - Retorna TRUE si existe, FALSE si no

F3 CalcularSubtotal

Propósito: Calcular el subtotal de un producto multiplicando su precio por la cantidad.

Funcionamiento:

- Validación de producto:
  - Verifica que el producto exista en Productos
  - Si no existe, retorna NULL
- Cálculo:
  - Obtiene el precio unitario del producto
  - Multiplica por la cantidad y retorna el resultado

Procedimientos

SP-1 Agregar Producto
SP-2 ActualizarProducto
SP-3 EliminarProducto
SP-4 AgregarProductoAlmacen
SP-5 ExistenciaProducto
SP-6 InsertarVenta
SP-7 InsertarDetalleVenta
SP-8 VerificarDuplicidadProductos

Triggers en MySQL.

T-1 ActualizarExistencias (Trigger)

Propósito: Reducir la existencia del producto en el almacén después de cada venta.

Funcionamiento:

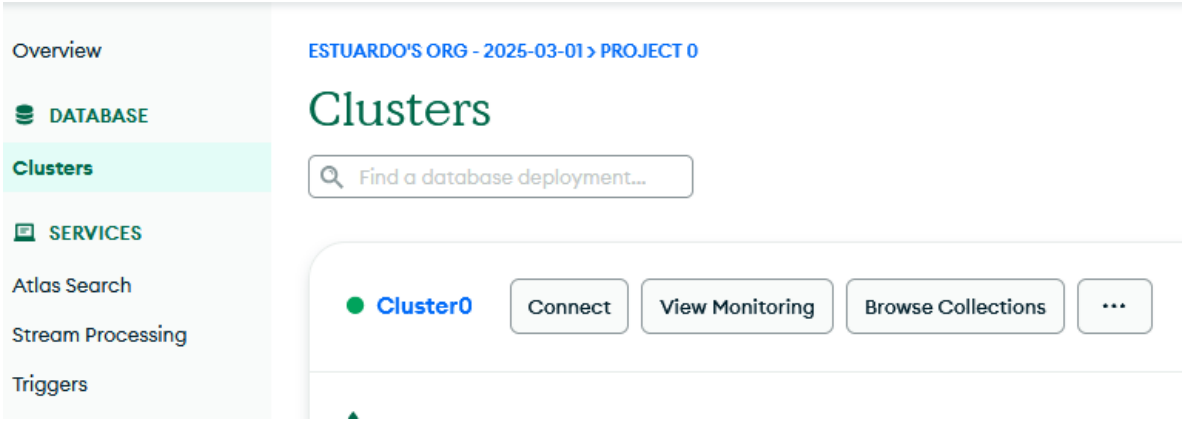
- Disparador automático:
  - Se ejecuta después de cada inserción en Ventas_Detalle
- Actualización:
  - Resta la cantidad vendida de la existencia en Productos_Almacenes
  - Usa el almacén asociado a la venta

Configuración de bases de datos en MongoDB

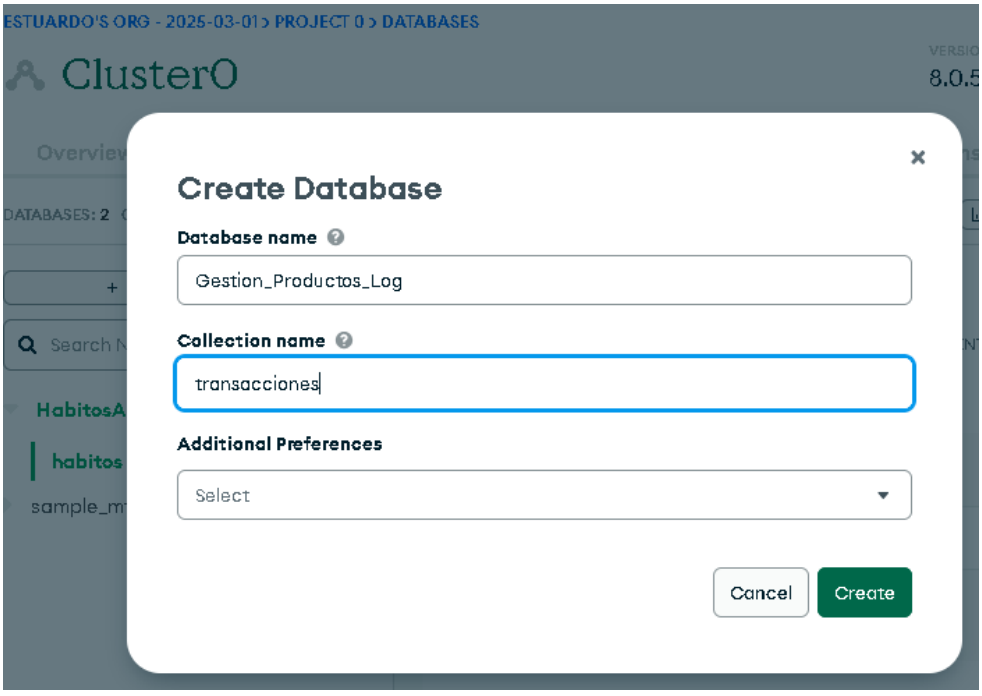
Se configuro una base de datos NoSQL en Altas MongoDB para guardar registros históricos, como un tipo de log, para ello se realizaron los siguientes pasos:

Nota: se utilizó un clúster que se había configura anteriores para otro curso, en el que se realizó la configuración inicial.

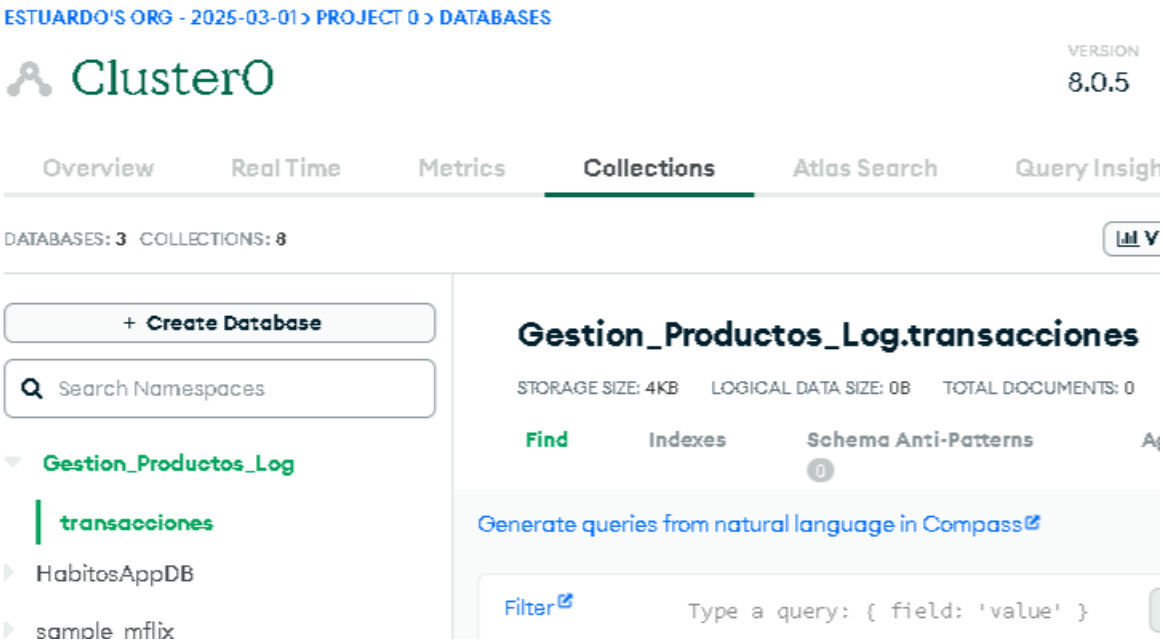
Se inició sesión en Atlas MongoD, en el apartado de Clusters y en Ver colecciones



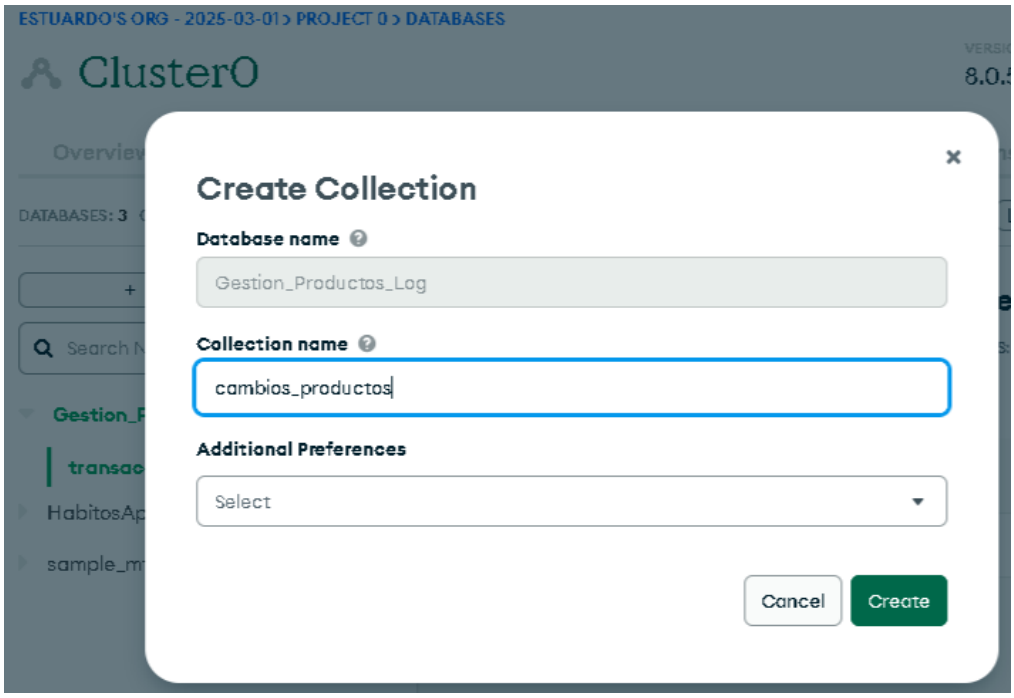
Podemos ver que aún no teníamos la base de datos, procedemos a crearla con el nombre de Gestion_Productos_Log y una colección transacciones



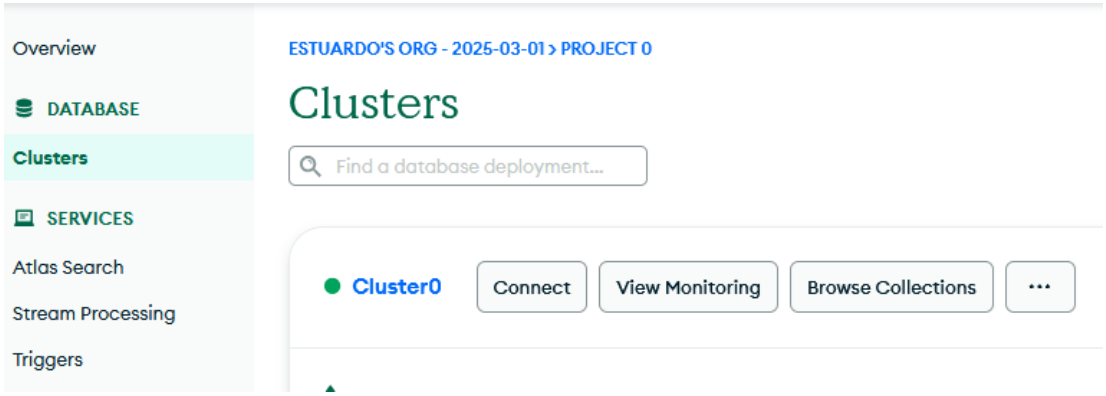
Ahora nos aparece la Base de datos con la colección



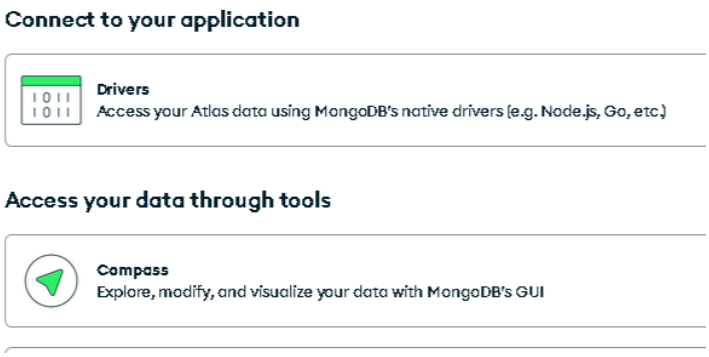
Procedemos a crear la otra colección con el nombre de cambios_productos



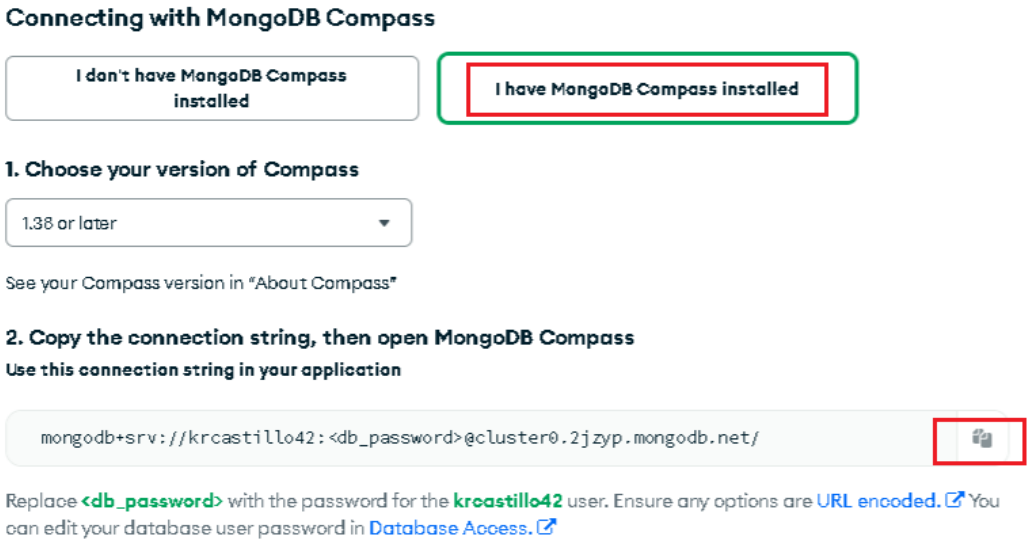
Regresamos a la opción de Clúster y hacemos clic en Connect



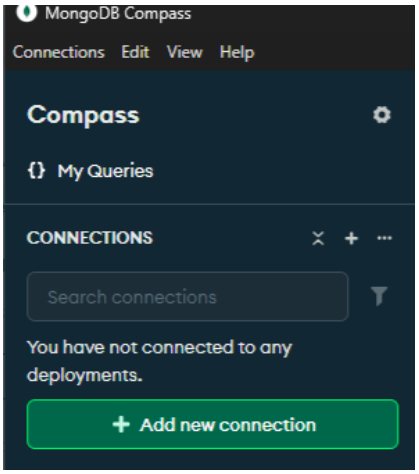
Seleccionamos la opción de compas



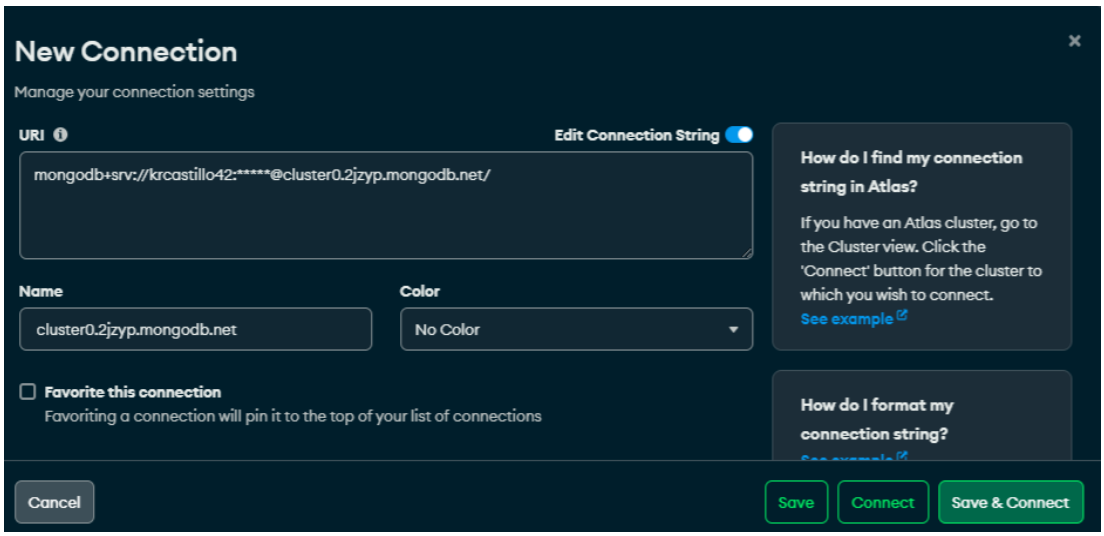
Seleccionamos la opción de que ya tenemos Compas Instalado y copiamos el enlace que nos da.



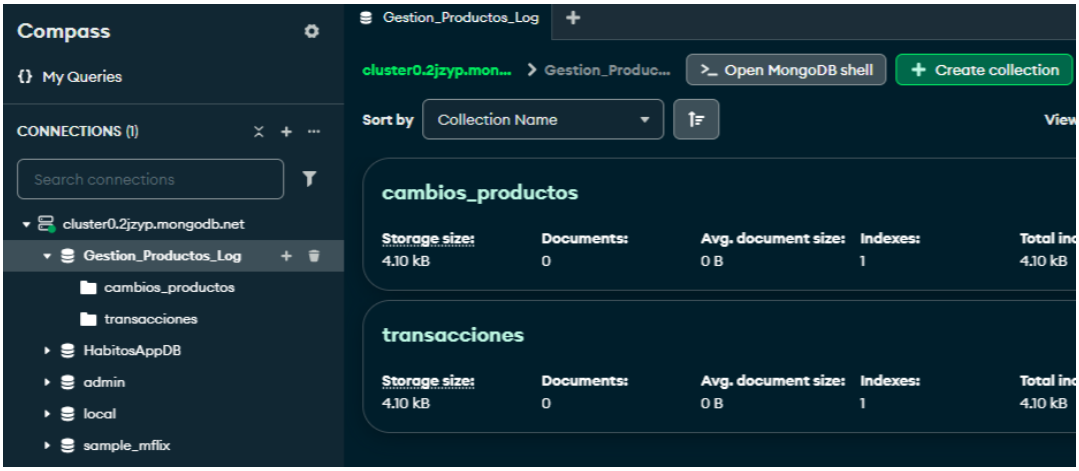
Abrimos Mongo Compas, y damos clic en opción agregar conexión



Pegamos el enlace de conexión que copiamos en la opción de conexión y damos guardar y conectar.



Y ya tendremos acceso a La base de datos desde Mongo Compas, podemos ver la Base de datos con sus colecciones.



Estructura de las colecciones de Base de Datos Gestion_Productos_Log

C-1 Transacciones

- usuario: Identificador único del usuario (ej.: usuario123).
- fecha: Fecha de la transacción en formato ISO.
- tipo: Tipo de transacción (ej.: venta, compra).

Formato JSON

```
{
 "usuario": "usuario123",
  "fecha": ISODate("2025-03-16T18:00:00.000Z"),
  "tipo": "venta"
}
```

C-2 Cambios_productos

- usuario: Identificador del usuario que realizó el cambio.
- tipo: Tipo de cambio (ej.: actualización, eliminación, creación).
- codigo_producto: Código del producto afectado.
- fecha: Fecha del cambio en formato ISO.

Formato JSON

```
{
  "usuario": "usuario123",
  "tipo": "actualización",
  "codigo_producto": "P00001",
  "fecha": ISODate("2025-03-16T18:00:00.000Z")
}
```

3. Implementación básica de funciones

Registro, Actualización y Eliminación de Productos

Registro de Productos

El proceso de registro de productos en la base de datos gestion_productos se realiza a través del procedimiento almacenado **SP-1 Agregar Producto** este procedimiento genera un código de producto automático en formato P00001, asegurando que cada producto tenga un identificador único.

Parámetros:

- p_descripcion: Nombre del producto

- p_precio_unitario: Precio del producto
- p_estado: Estado del producto (booleano)

Funcionamiento:

- Generación de código:
  - Extrae el número máximo de los códigos existentes (ej.: si hay "P00005", extrae 5)
  - Incrementa en 1 y formatea como P00006 usando LPAD
  - Verifica si el código generado ya existe
- Validación:
  - Si el código existe, muestra mensaje de error
  - Si no existe, inserta el producto y muestra confirmación

Ejemplo de Uso:

CALL AgregarProducto('Jugo de naranja', 25.00, 1);

Actualización de Productos

Para modificar un producto existente en la base de datos, se utiliza el procedimiento almacenado **SP-2 ActualizarProducto**

Parámetros:

- p_cod_producto: Código único del producto
- p_descripcion, p_precio_unitario, p_estado: Nuevos valores

Funcionamiento:

- Actualiza directamente los campos especificados en la tabla Productos

Ejemplo de Uso:

CALL ActualizarProducto('P00021', 'Jugo de naranja', 20.00, 0);

Eliminación de Productos

La eliminación de un producto se maneja con el procedimiento almacenado **SP-3 EliminarProducto**

Parámetros:

- p_cod_producto: Código del producto a eliminar

Funcionamiento:

- Validación de existencia:
  - Verifica si el producto existe
- Eliminación en cascada:
  - Elimina registros en Ventas_Detalle (detalles de ventas)
  - Elimina registros en Productos_Almacenes (inventario)
  - Elimina el producto de Productos
- Mensajes:
  - Confirma eliminación o muestra error si no existe

Ejemplo de Uso:
CALL EliminarProducto('P00021');

Gestión de inventario

La gestión del inventario se encarga de controlar las existencias de productos en los distintos almacenes, permitiendo agregar productos a almacenes, modificar su cantidad y consultar el estado general del inventario.

Agregar Productos a Almacenes

El procedimiento almacenado **SP-4 AgregarProductoAlmacen** permite asociar un producto a un almacén específico y definir su cantidad inicial.

Parámetros:

- p_id_almacen: Identificador del almacén
- p_cod_producto: Código del producto
- p_existencia: Cantidad inicial del producto en el almacén

Funcionamiento:

- Validaciones previas:
  - Verifica que el almacén y el producto existan
  - Chequea que el producto no esté ya registrado en ese almacén
- Acciones:
  - Si todo es válido, inserta el registro en Productos_Almacenes
  - Muestra mensaje de éxito o error según el caso

Ejemplo de Uso:

CALL AgregarProductoAlmacen(1, 'P00020', 5);

Modificar Existencias en Almacenes

El procedimiento almacenado **SP-5 ExistenciaProducto** permite actualizar la cantidad de un producto en un almacén.

Parámetros:

- p_id_almacen: Identificador del almacén
- p_cod_producto: Código del producto
- p_nueva_existencia: Nueva cantidad del producto

Funcionamiento:

- Validaciones previas:
  - Verifica que el almacén y el producto existan
  - Chequea que el producto esté registrado en ese almacén
- Acciones:
  - Si todo es válido, actualiza la existencia en Productos_Almacenes
  - Muestra mensaje de éxito o error según el caso

Ejemplo de Uso:

CALL ExistenciaProducto(1, 'P00020', 10);

Generar Venta

El procedimiento almacenado **SP-6 InsertarVenta** permite generar el encabezado de una venta en un almacén.

Parámetros:

- p_id_almacen: Almacén asociado a la venta

Funcionamiento:

- Datos básicos:
  - Obtiene fecha actual con CURDATE()
  - Recupera código de usuario actual usando ObtenerCodigoUsuarioActual()
- Insertar venta:
  - Registra la venta en Ventas con total inicial en 0
  - Usa LAST_INSERT_ID() para obtener el ID generado

Generar Detalle Venta

El procedimiento almacenado **SP-7 InsertarDetalleVenta** agregar productos a un encabezado de venta existente.

Parámetros:

- p_id_venta: ID de la venta a modificar
- p_cod_producto: Producto a agregar
- p_cantidad: Cantidad del producto

Funcionamiento:

1. Validaciones:
    - Verifica que la venta sea la más reciente del usuario
    - Chequea existencia del producto con VerificarProducto()
    - Verifica disponibilidad en el almacén asociado
2. Cálculos:
    - Usa CalcularSubtotal() para calcular el costo
    - Actualiza el total de la venta

Verificación de duplicidad

El procedimiento almacenado **SP-8 VerificarDuplicidadProductos** permite validar la duplicidad de registros en los productos.

Funcionamiento:

- Consulta subordinada:
    - Agrupa por descripción y cuenta cuántas tienen más de 1 registro
- Resultado:
    - Si hay duplicados, muestra código y descripción de los productos repetidos
    - Si no hay duplicados, muestra mensaje informativo

Usuarios y Roles

Usuarios

La base de datos almacena información sobre los usuarios en la tabla Usuarios, incluyendo:

- cod_usuario: Código único del usuario.
- usuario: Nombre de usuario.
- password: Contraseña encriptada.
- role: Tipo de usuario (administrador u operador).
- fecha_creacion: Fecha en la que se creó la cuenta.

Roles de Usuario

Administrador

- Tiene acceso total a la base de datos.
- Puede crear, modificar y eliminar usuarios.
- Gestiona productos, almacenes y ventas.

Operador

- Puede registrar y consultar ventas.
- Accede a la información de productos y almacenes.
- No puede modificar usuarios ni cambiar configuraciones del sistema.

Consultas básicas para reportes generales

A continuación, se describen los reportes generados en la base de datos gestion_productos.

1. Reporte de productos disponibles

- Muestra todos los productos registrados con su código, descripción, precio unitario y estado.

2. Inventario por Almacén

- Presenta el inventario actual de cada almacén con los productos y su cantidad disponible.

3. Detalles de una Venta Específica

- Muestra los productos vendidos en una venta particular, indicando cantidad y subtotal.

4. Ventas por Usuario

- Resume la cantidad de ventas realizadas por cada usuario y el monto total generado.

5. Ventas por Almacén

- Indica el total de ventas y el monto total generado en cada almacén.

6. Usuarios Registrados

- Muestra la lista de usuarios con sus roles y la fecha de creación de la cuenta.

7. Existencia de un Producto Específico

- Consulta el stock disponible de un producto en todos los almacenes.

Consulta de Inventario:

El sistema proporciona una vista llamada “ReporteInventarioGeneral”, que muestra el stock total de cada producto en todos los almacenes junto con su valor total.

*SELECT * FROM ReporteInventarioGeneral;*

Resultados Esperados:

Codigo	Nombre	StockActual	PrecioUnitario	ValorTotal
P00001	Arroz Blanco	50	10.00	500.00
P00002	Frijoles Negros	30	12.00	360.00

Esta funcionalidad permite un control efectivo del inventario, asegurando la disponibilidad de productos y facilitando la toma de decisiones en la gestión de almacenes.