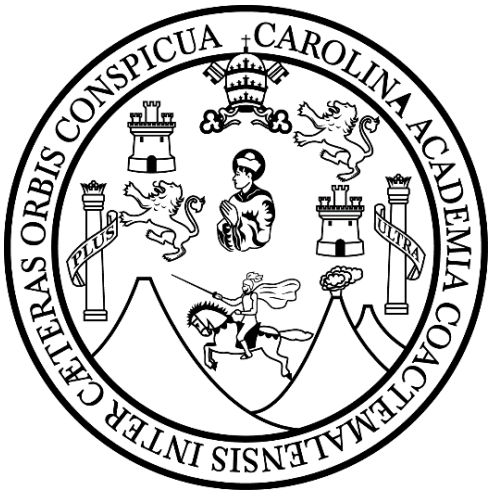
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE VACACIONES PRIMER SEMESTRE 2024



PROYECTO 2

DOCUMENTACIÓN DE CONSULTAS EN MONGO DB

GRUPO #2

AUX. JHONATHAN DANIEL TOCAY COTZOJAY

MARIANO FRANCISCO CAMPOSECO CAMPOSECO	**202030987**
LUIS NERY CIFUENTES RODAS	**202030482**
MANUEL ANTONIO ROJAS PAXTOR	**202030799**
DAVID ENRIQUE LUX BARRERA	**201931344**
ELVIS LIZANDRO AGUILAR TAX	**201930304**
ANGEL GEOVANY ARAGÓN PÉREZ	**201901055**

27 DE JUNIO DE 2024

TABLA DE CONTENIDO

1. CREACIÓN DE LA BASE DE DATOS Y COLECCIONES **1**

2. CONSULTAS DE LA TABLA “Autor” **1**

2.1. Crear autor 1

2.2. Obtener autor por ID 1

2.3. Eliminar autor 1

3. CONSULTAS DE LA TABLA “GeneroLibro” **2**

3.1. Crear genero 2

3.2. Obtener todos lo géneros 2

3.3. Obtener género por ID 2

3.4. Actualizar género 2

3.5. Eliminar género 3

4. CONSULTAS DE LA TABLA “Libro” **3**

4.1. Crear libro 3

4.2. Obtener todos los libros 4

4.3. Obtener libro por ID 4

4.4. Eliminar libro 4

4.5. Actualizar libro 4

4.6. Buscar libros 5

4.6.1. Buscar libros por título 5

4.6.2. Buscar libros por autor 5

4.6.3. Buscar libros por género 6

4.6.4. Buscar libros por rango de precio 6

4.6.5. Buscar libros por minimo de puntuacion 6

4.6.6. Buscar libros por múltiples parámetros de búsqueda. 7

5. CONSULTAS DE LA TABLA “Pedido” **7**

5.1. Crear pedido 7

5.1.1. Iniciando la transacción 7

5.1.2. Verificar y actualizar el stock de libros 7

5.1.3. Insertar un nuevo pedido 7
5.1.4. Actualizar las compras del usuario 7
5.1.5. Confirmar la transacción 8
5.2. Obtener todos los pedidos 8
5.3. Obtener pedido por ID 8
5.4. Obtener los libros más vendidos 8
5.5. Buscar pedidos 10
5.5.1. Buscar pedidos por usuario 10
5.5.2. Buscar pedidos por su estado 10
5.6. Actualizar el estado del pedido 11
6. CONSULTAS DE LA TABLA “Reseña” 11
6.1. Crear reseña 11
6.1.1. Iniciando la transacción 11
6.1.2. Verificar la reseña 11
6.1.3. Insertar la reseña 11
6.1.4. Cálculo de la nueva puntuación promedio del libro 12
6.1.5. Actualización de la puntuación promedio del libro 12
6.1.6. Confirmar la transacción 12
6.2. Obtener reseñas por libro 12
7. CONSULTAS DE LA TABLA “Usuario” 13
7.1. Registro de nuevo usuario 13
7.2. Obtener usuario por ID 13
7.3. Actualizar usuario 14
7.4. Buscar usuario por email (login) 14

1. CREACIÓN DE LA BASE DE DATOS Y COLECCIONES

```
use BookStore
db.createCollection("Autor");
db.createCollection("GeneroLibro");
db.createCollection("Libro");
db.createCollection("Pedido");
db.createCollection("Reseña");
db.createCollection("Usuario");
```

2. CONSULTAS DE LA TABLA "Autor"

2.1. Crear autor

```
db.autors.insertOne({
    nombre: "Nombre del Autor",
    biografia: "Biografía del Autor",
    foto_url: "URL de la foto"
});
```

Inserta un nuevo documento en la colección autors con los campos nombre, biografia y foto_url.

2.2. Obtener autor por ID

```
db.autors.findOne({ _id: ObjectId(id) })
    .populate({
        path: 'libros',
            select: 'titulo descripcion fecha_publicacion cantidad_stock
puntuacion_promedio precio imagen_url',
        populate: { path: 'genero_id', select: 'nombre' }
    });
```

Busca un único documento en la colección autors que coincida con el id proporcionado. Con populate se llenan los campos relacionados en la colección libros y los subcampos de genero_id.

2.3. Eliminar autor

```
const session = db.getMongo().startSession();
session.startTransaction();

db.autors.updateOne(
    { _id: ObjectId(id) },
    { $set: { disponibilidad: false } },
    { session }
);
```

```
db.libros.updateMany(
    { autor_id: ObjectId(id) },
    { $set: { disponibilidad: false } },
    { session }
);

session.commitTransaction();
session.endSession();
```

Utilizando una transacción en mongoDB, cambia el estado de un autor a no disponible (que es equivalente a ser eliminado), al mismo tiempo también se colocan como no disponibles todos los libros pertenecientes a ese autor.

3. CONSULTAS DE LA TABLA "GeneroLibro"

3.1. Crear genero

Primero se verifica si ya existe un género con ese nombre

```
db.generolibros.findOne({ nombre: "Nombre del Género" });
```

En caso de no encontrar una coincidencia, se procede a insertar el nuevo genero

```
db.generolibros.insertOne({ nombre: "Nombre del Género" });
```

3.2. Obtener todos lo géneros

```
db.generolibros.find({});
```

Busca todos los documentos en la colección generolibros.

3.3. Obtener género por ID

```
db.generolibros.findOne({ _id: ObjectId(id) });
```

Busca un único documento en la colección generolibros que coincida con el id proporcionado.

3.4. Actualizar género

```
db.generolibros.updateOne(
    { _id: ObjectId(id) },
    { $set: { nombre: "Nuevo Nombre del Género" } }
);
```

```

```

Actualiza el campo nombre del documento que coincide con el id proporcionado.

3.5. Eliminar género

```
db.generolibros.deleteOne({ _id: ObjectId(id) });
```

Elimina un documento en la colección generolibros que coincida con el id proporcionado.

4. CONSULTAS DE LA TABLA "Libro"

4.1. Crear libro

```
const session = db.getMongo().startSession();
session.startTransaction();

db.libros.insertOne({
    titulo: "Título del Libro",
    autor_id: ObjectId("ID del Autor"),
    descripcion: "Descripción del Libro",
    genero_id: ObjectId("ID del Género"),
    fecha_publicacion: new Date("Fecha de Publicación"),
    disponibilidad: true,
    cantidad_stock: 10,
    precio: 20.00,
    imagen_url: "URL de la Imagen"
}, { session });

db.autors.updateOne(
    { _id: ObjectId("ID del Autor") },
    { $push: { libros: ObjectId("ID del Libro") } },
    { session }
);

session.commitTransaction();
session.endSession();
```

Utilizando transacciones, se inserta un nuevo documento en la colección "libros" con insertOne y se actualiza el autor en la colección "autors" con updateOne para agregarle el nuevo libro. Si todo es exitoso, se confirma con commitTransaction; si hay errores, se aborta con abortTransaction.

4.2. Obtener todos los libros

```
db.libros.find({ disponibilidad: true })
    .populate('autor_id', 'nombre biografia foto_url')
    .populate('genero_id', 'nombre');
```

Busca todos los documentos en la colección libros donde la disponibilidad sea true (es decir que existan). Se utiliza populate el cual llena los campos relacionados en las colecciones autor_id y genero_id con los campos especificados.

4.3. Obtener libro por ID

```
db.libros.findOne({ _id: ObjectId(id) })
    .populate('autor_id', 'nombre biografia foto_url')
    .populate('genero_id', 'nombre');
```

Busca un único documento en la colección libros que coincida con el id proporcionado. Se utiliza populate el cual llena los campos relacionados en las colecciones autor_id y genero_id con los campos especificados.

4.4. Eliminar libro

```
db.libros.updateOne(
    { _id: ObjectId(id) },
    { $set: { disponibilidad: false } }
);
```

Actualiza el campo disponibilidad del documento en la colección libros que coincida con el id proporcionado, estableciéndolo en false.

4.5. Actualizar libro

```
db.libros.updateOne(
    { _id: ObjectId(id) },
    {
        $set: {
            titulo: "Nuevo Título",
            autor_id: ObjectId("Nuevo ID del Autor"),
            descripcion: "Nueva Descripción",
            genero_id: ObjectId("Nuevo ID del Género"),
            fecha_publicacion: new Date("Nueva Fecha de Publicación"),
            disponibilidad: true,
```

```
            cantidad_stock: 20,
            precio: 25.00,
            imagen_url: "Nueva URL de la Imagen"
        }
    }
);
```

Actualiza los campos especificados del libro que coincidan con el id proporcionado.

4.6. Buscar libros

4.6.1. Buscar libros por título

```
db.libros.find({
    titulo: { $regex: "título buscado", $options: 'i' },
    disponibilidad: true
})
.populate('autor_id', 'nombre biografia foto_url')
.populate('genero_id', 'nombre');
```

Busca libros cuyo título coincide con el texto proporcionado, sin importar mayúsculas o minúsculas, y que estén disponibles. Además, usa populate para incluir los detalles del autor y del género.

4.6.2. Buscar libros por autor

```
const autores = db.autors.find({ nombre: { $regex: "nombre del autor", $options: 'i' } }).select('_id');
const autorIds = autores.map(autor => autor._id);

db.libros.find({
    autor_id: { $in: autorIds },
    disponibilidad: true
})
.populate('autor_id', 'nombre biografia foto_url')
.populate('genero_id', 'nombre');
```

Esta consulta primero busca los IDs de autores cuyo nombre coincida con el texto proporcionado, sin importar mayúsculas o minúsculas. Luego, busca libros de esos autores que estén disponibles y utiliza populate para incluir detalles del autor y del género.

4.6.3. Buscar libros por género

```
const genero = db.generolibros.findOne({ nombre: { $regex: "nombre del género", $options: 'i' } });

db.libros.find({
    genero_id: genero._id,
    disponibilidad: true
})
.populate('autor_id', 'nombre biografia foto_url')
.populate('genero_id', 'nombre');
```

Esta consulta primero busca el ID del género cuyo nombre coincida con el texto proporcionado, sin importar mayúsculas o minúsculas. Luego, busca libros de ese género que estén disponibles y usa populate para incluir detalles del autor y del género.

4.6.4. Buscar libros por rango de precio

```
db.libros.find({
    precio: { $gte: minimo, $lte: maximo },
    disponibilidad: true
})
.populate('autor_id', 'nombre biografia foto_url')
.populate('genero_id', 'nombre');
```

Estas consultas buscan libros que estén disponibles y cuyo precio esté dentro del rango especificado (mínimo, máximo). Utilizan populate para incluir detalles del autor y del género.

4.6.5. Buscar libros por minimo de puntuacion

```
db.libros.find({
    puntuacion_promedio: { $gte: minimo },
    disponibilidad: true
})
.populate('autor_id', 'nombre biografia foto_url')
.populate('genero_id', 'nombre');
```

Busca libros que estén disponibles y que tengan una puntuación promedio mayor o igual a la especificada. También utiliza populate para incluir detalles del autor y del género.

4.6.6. Buscar libros por múltiples parámetros de búsqueda.

Para obtener los libros con múltiples parámetros de búsqueda, lo único que se hace es robustecer la query agregando más parámetros a objetos que se le envía a find. De esta manera se pueden obtener libros por título, autor, género, rango de precio y mínimo de puntuación. Todo esto bajo las querys que fueron mostradas de manera individual.

5. CONSULTAS DE LA TABLA "Pedido"

5.1. Crear pedido

La creacion de un nuevo pedido es una operacion en donde se requiere de multiples consultas dentro de mongoDB. Dichas consultas estan seccionadas de la siguiente manera para lograr reflejar un nuevo pedido con toda la logica del negocio dado:

5.1.1. Iniciando la transacción

```
const session = db.getMongo().startSession();
session.startTransaction();
```

5.1.2. Verificar y actualizar el stock de libros

```
const libro = db.libros.findOne({ _id: ObjectId(libro_id) });
if (libro.cantidad_stock < cantidad) {
    throw new Error("Not enough stock");
}
libro.cantidad_stock -= cantidad;
db.libros.updateOne({ _id: ObjectId(libro_id) }, { $set: { cantidad_stock:
libro.cantidad_stock } }, { session });
```

5.1.3. Insertar un nuevo pedido

```
db.pedidos.insertOne({
    usuario_id: ObjectId(usuario_id),
    libros: [/* array de libros */],
    direccion_envio: "dirección de envío",
    metodo_pago: "método de pago"
}, { session });
```

5.1.4. Actualizar las compras del usuario

```
db.usuarios.updateOne(
    { _id: ObjectId(usuario_id) },
    { $push: { compras: ObjectId(pedido_id) } },
    { session }
);
```

5.1.5. Confirmar la transacción

```
session.commitTransaction();
session.endSession();
```

5.2. Obtener todos los pedidos

```
db.pedidos.find({})
    .populate({
        path: 'libros.libro_id',
         populate: { path: 'autor_id genero_id', select: 'nombre biografia
foto_url' }
    })
    .populate({
        path: 'usuario_id',
        select: 'nombre apellido email telefono direccion'
    });
```

Esta consulta busca todos los pedidos. Usa populate para incluir los detalles de los libros, autores y géneros relacionados, así como los detalles del usuario que hizo el pedido.

5.3. Obtener pedido por ID

```
db.pedidos.findOne({ _id: ObjectId(id) })
    .populate({
        path: 'libros.libro_id',
         populate: { path: 'autor_id genero_id', select: 'nombre biografia
foto_url' }
    })
    .populate({
        path: 'usuario_id',
        select: 'nombre apellido email telefono direccion'
    });
```

Esta consulta busca un pedido de acuerdo a un ID dado. Usa populate para incluir los detalles de los libros, autores y géneros relacionados, así como los detalles del usuario que hizo el pedido.

5.4. Obtener los libros más vendidos

```
db.pedidos.aggregate([
    { $unwind: "$libros" },
    {
        $group: {
```

```
            _id: "$libros.libro_id",
            totalCantidad: { $sum: "$libros.cantidad" }
        }
    },
    { $sort: { totalCantidad: -1 } },
    { $limit: 5 },
    {
        $lookup: {
            from: "libros",
            localField: "_id",
            foreignField: "_id",
            as: "libro"
        }
    },
    { $unwind: "$libro" },
    {
        $lookup: {
            from: "autors",
            localField: "libro.autor_id",
            foreignField: "_id",
            as: "autor"
        }
    },
    { $unwind: "$autor" },
    {
        $lookup: {
            from: "generolibros",
            localField: "libro.genero_id",
            foreignField: "_id",
            as: "genero"
        }
    },
    { $unwind: "$genero" },
    {
        $project: {
            _id: "$libro._id",
            titulo: "$libro.titulo",
            autor: "$autor.nombre",
            descripcion: "$libro.descripcion",
            genero: "$genero.nombre",
```

```
            fecha_publicacion: "$libro.fecha_publicacion",
            puntuacion_promedio: "$libro.puntuacion_promedio",
            imagen_url: "$libro.imagen_url",
            totalCantidad: 1
        }
    }
]);
```

Esta consulta agrega los pedidos para contar la cantidad total vendida de cada libro, ordena los libros por cantidad en orden descendente, y luego utiliza lookup para obtener los detalles de los libros, autores y géneros relacionados, mostrando los 5 libros más vendidos.

5.5. Buscar pedidos

5.5.1. Buscar pedidos por usuario

```
db.pedidos.find({ usuario_id: ObjectId(usuario_id) })
    .populate({
        path: 'libros.libro_id',
         populate: { path: 'autor_id genero_id', select: 'nombre biografia
foto_url' }
    })
    .populate({
        path: 'usuario_id',
        select: 'nombre apellido email telefono direccion'
    });
```

Busca pedidos específicos según el usuario que los realizó. Usa populate para incluir detalles de los libros, autores y géneros relacionados, así como detalles del usuario.

5.5.2. Buscar pedidos por su estado

```
db.pedidos.find({ estado: "estado del pedido" })
    .populate({
        path: 'libros.libro_id',
         populate: { path: 'autor_id genero_id', select: 'nombre biografia
foto_url' }
    })
    .populate({
        path: 'usuario_id',
        select: 'nombre apellido email telefono direccion'
    });
```

Busca pedidos específicos según el estado del pedido. Usa populate para incluir detalles de los libros, autores y géneros relacionados, así como detalles del usuario.

5.6. Actualizar el estado del pedido

```
db.pedidos.updateOne(
    { _id: ObjectId(id) },
    {
        $set: {
            estado: "nuevo estado",
            fecha_envio: estado === 'Enviado' ? new Date() : null,
            fecha_entrega: estado === 'Entregado' ? new Date() : null
        }
    }
);
```

Actualiza el estado de un pedido específico, y si el estado es "Enviado" o "Entregado", también actualiza las fechas correspondientes.

6. CONSULTAS DE LA TABLA "Reseña"

6.1. Crear reseña

Crear una reseña es un procedimiento complejo que consta de varias consultas. Estas consultas primero verifican si el usuario ya ha reseñado el libro, luego insertan una nueva reseña y calculan la nueva puntuación promedio del libro. Finalmente, actualizan la puntuación promedio del libro en una transacción para asegurar la consistencia de los datos.

6.1.1. Iniciando la transacción

```
const session = db.getMongo().startSession();
session.startTransaction();
```

6.1.2. Verificar la reseña

```
const existingReview = db.reseñas.findOne({ libro_id: ObjectId(libro_id),
usuario_id: ObjectId(usuario_id) });
if (existingReview) {
    throw new Error('You have already reviewed this book');
}
```

6.1.3. Insertar la reseña

```
db.reseñas.insertOne({
```

```
    libro_id: ObjectId(libro_id),
    usuario_id: ObjectId(usuario_id),
    comentario: "comentario de la reseña",
    puntuacion: puntuacion
}, { session });
```

6.1.4. Cálculo de la nueva puntuación promedio del libro

```
const reviews = db.reseñas.find({ libro_id: ObjectId(libro_id)
}).toArray();
const totalReviews = reviews.length;
const totalPuntuacion = reviews.reduce((sum, review) => sum +
review.puntuacion, 0);
const puntuacionPromedio = totalPuntuacion / totalReviews;
```

6.1.5. Actualización de la puntuación promedio del libro

```
db.libros.updateOne(
    { _id: ObjectId(libro_id) },
    { $set: { puntuacion_promedio: puntuacionPromedio } },
    { session }
);
```

6.1.6. Confirmar la transacción

```
session.commitTransaction();
session.endSession();
```

6.2. Obtener reseñas por libro

```
db.reseñas.find({ libro_id: ObjectId(id) })
    .populate({
        path: 'usuario_id',
        select: 'nombre apellido correo fecha_registro'
    })
    .select('-libro_id');
```

Esta consulta busca todas las reseñas asociadas a un libro específico, identificado por su libro_id. Utiliza populate para incluir los detalles del usuario que hizo la reseña y excluye el campo libro_id de los resultados.

7. CONSULTAS DE LA TABLA "Usuario"

7.1. Registro de nuevo usuario

```
db.usuarios.insertOne({
    nombre: "Nombre del Usuario",
    apellido: "Apellido del Usuario",
    email: "email@example.com",
    telefono: "Número de Teléfono",
    direccion: "Dirección del Usuario",
    fecha_nacimiento: new Date("Fecha de Nacimiento"),
    contrasenia: "Contraseña Encriptada",
    metodo_pago: "Método de Pago",
    rol: "Rol del Usuario"
});
```

Esta consulta inserta un nuevo documento en la colección usuarios con los campos proporcionados en el cuerpo de la solicitud. La contraseña debe estar encriptada antes de ser almacenada.

7.2. Obtener usuario por ID

```
db.usuarios.findOne({ _id: ObjectId(id) })
    .select('-contrasenia')
    .populate({
        path: 'compras',
        model: 'pedidos',
               select: 'direccion_envio metodo_pago estado fecha_pedido
fecha_envio fecha_entrega libros',
        populate: {
            path: 'libros.libro_id',
                          select: 'titulo descripcion fecha_publicacion
puntuacion_promedio imagen_url autor_id genero_id',
            populate: [
                {
                    path: 'autor_id',
                    select: 'nombre biografia foto_url'
                },
                {
                    path: 'genero_id',
                    select: 'nombre'
                }
            ]
```

```
        }
    });
```

Esta consulta busca un usuario específico por su id, excluyendo el campo contrasenia de los resultados. Además, utiliza populate para incluir los detalles de las compras del usuario, así como los detalles de los libros, autores y géneros relacionados.

7.3. Actualizar usuario

```
db.usuarios.updateOne(
    { _id: ObjectId(id) },
    {
        $set: {
            nombre: "Nuevo Nombre",
            apellido: "Nuevo Apellido",
            email: "nuevoemail@example.com",
            telefono: "Nuevo Teléfono",
            direccion: "Nueva Dirección",
            metodo_pago: "Nuevo Método de Pago",
            fecha_nacimiento: new Date("Fecha de Nacimiento"),
            contrasenia: "nueva_contrasenia_hash"

        }
    }
);
```

Esta consulta actualiza los campos del usuario que se proporcionan en el cuerpo de la solicitud. Utiliza updateOne para buscar el usuario por su id y establece los nuevos valores para los campos presentes en la solicitud. Solo los campos que están presentes en el cuerpo de la solicitud serán actualizados.

7.4. Buscar usuario por email (login)

```
db.usuarios.findOne(
    { email: "email@example.com" },
    { email: 1, contrasenia: 1 }
);
```

Esta consulta busca un usuario por su email y devuelve solo los campos email y contrasenia. La selección de campos se realiza mediante el segundo argumento de findOne, donde 1 indica que el campo debe ser incluido en el resultado. Luego, en la lógica de la aplicación, se compara la contraseña proporcionada con la almacenada. Si coinciden, se genera un token JWT con la información del usuario.