Esquemas de validación:

Collection <<Sucursales>>:

```
{
  "$jsonSchema": {
    "bsonType": "object",
    "required": ["nombre", "direccion", "telefono", "ciudad", "bodegas"],
    "properties": {
      "nombre": {
        "bsonType": "string",
        "description": "El nombre de la sucursal es obligatorio y debe ser una cadena de texto."
      },
      "direccion": {
        "bsonType": "string",
        "description": "La dirección es obligatoria y debe ser una cadena de texto."
      },
      "telefono": {
        "bsonType": "string",
        "description": "El teléfono es obligatorio y debe ser una cadena de texto."
      },
      "ciudad": {
        "bsonType": "object",
        "required": ["nombre", "codigo"],
        "properties": {
          "nombre": {
            "bsonType": "string",
            "description": "El nombre de la ciudad es obligatorio y debe ser una cadena de texto."
          },
          "codigo": {
```

```
          "bsonType": "string",
          "description": "El código de la ciudad es obligatorio y debe ser una cadena de texto."
        }
      },
      "description": "La ciudad es obligatoria y debe ser un objeto con nombre y código."
    },
    "bodegas": {
      "bsonType": "array",
      "items": {
        "bsonType": "object",
        "required": ["nombre", "tamañoM2", "productos"],
        "properties": {
          "nombre": {
            "bsonType": "string",
            "description": "El nombre de la bodega es obligatorio y debe ser una cadena de texto."
          },
          "tamañoM2": {
            "bsonType": "double",
            "description": "El tamaño de la bodega en metros cuadrados es obligatorio y debe ser un número decimal."
          },
          "productos": {
            "bsonType": "array",
            "items": {
              "bsonType": "object",
              "required": ["nombre", "precio", "detallesEmpacado", "categoria"],
              "properties": {
                "nombre": {
```

```
            "bsonType": "string",
            "description": "El nombre del producto es obligatorio y debe ser una cadena de texto."
          },
          "precio": {
            "bsonType": "double",
            "description": "El precio del producto es obligatorio y debe ser un número decimal."
          },
          "detallesEmpacado": {
            "bsonType": "string",
            "description": "Los detalles del empaque son obligatorios y deben ser una cadena de texto."
          },
          "categoria": {
            "bsonType": "object",
            "required": ["codigo", "nombre", "descripcion"],
            "properties": {
              "codigo": {
                "bsonType": "string",
                "description": "El código de la categoría es obligatorio y debe ser una cadena de texto."
              },
              "nombre": {
                "bsonType": "string",
                "description": "El nombre de la categoría es obligatorio y debe ser una cadena de texto."
              },
              "descripcion": {
                "bsonType": "string",
                "description": "La descripción de la categoría es obligatoria y debe ser una cadena de texto."
```

```
                }
              },
                "description": "La categoría es obligatoria y debe ser un objeto con código, nombre y descripción."
            }
          },
            "description": "El producto es obligatorio y debe ser un objeto con nombre, precio, detallesEmpacado y categoría."
        },
          "description": "La lista de productos es obligatoria y debe contener objetos de productos."
      }
    },
      "description": "La bodega es obligatoria y debe ser un objeto con nombre, tamañoM2 y productos."
  },
    "description": "La lista de bodegas es obligatoria y debe contener objetos de bodegas."
  }
},
  "description": "La sucursal debe contener nombre, dirección, teléfono, ciudad y bodegas."
}
}
```

Collection <<Bodegas>>:

```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'nombre',
```

```
      'tamañoM2',
      'productos'
    ],
    properties: {
      nombre: {
        bsonType: 'string',
        description: 'El nombre es obligatorio y debe ser una cadena de texto.'
      },
      'tamañoM2': {
        bsonType: 'double',
        description: 'El tamaño de la bodega en metros cuadrados es obligatorio y debe ser un número decimal.'
      },
      productos: {
        bsonType: 'array',
        items: {
          bsonType: 'object',
          required: [
            'nombre',
            'precio',
            'detallesEmpacado',
            'categoria'
          ],
          properties: {
            nombre: {
              bsonType: 'string',
              description: 'El nombre del producto es obligatorio y debe ser una cadena de texto.'
            },
            precio: {
```

```
        bsonType: 'double',
        description: 'El precio del producto es obligatorio y debe ser un número decimal.'
      },
      detallesEmpacado: {
        bsonType: 'string',
        description: 'Los detalles del empaque son obligatorios y deben ser una cadena de texto.'
      },
      categoria: {
        bsonType: 'object',
        required: [
          'codigo',
          'nombre',
          'descripcion'
        ],
        properties: {
          codigo: {
            bsonType: 'string',
            description: 'El código de la categoría es obligatorio y debe ser una cadena de texto.'
          },
          nombre: {
            bsonType: 'string',
            description: 'El nombre de la categoría es obligatorio y debe ser una cadena de texto.'
          },
          descripcion: {
            bsonType: 'string',
            description: 'La descripción de la categoría es obligatoria y debe ser una cadena de texto.'
```

```
            }
          }
        }
      }
    }
  }
 }
}
```

Collection <<Productos>>:

```
{
 $jsonSchema: {
  bsonType: 'object',
  required: [
   'nombre',
   'precio',
   'detallesEmpacado',
   'categoria',
   'sucursalId'
  ],
  properties: {
   nombre: {
    bsonType: 'string',
    description: 'El nombre del producto es obligatorio y debe ser una cadena de texto.'
   },
   precio: {
    bsonType: 'double',
    description: 'El precio del producto es obligatorio y debe ser un número decimal.'
```

```
    },
    detallesEmpacado: {
     bsonType: 'string',
     description: 'Los detalles del empaque son obligatorios y deben ser una cadena de texto.'
    },
    categoria: {
     bsonType: 'object',
     required: [
      'codigo',
      'nombre',
      'descripcion'
     ],
     properties: {
      codigo: {
       bsonType: 'string',
       description: 'El código de la categoría es obligatorio y debe ser un número entero.'
      },
      nombre: {
       bsonType: 'string',
       description: 'El nombre de la categoría es obligatorio y debe ser una cadena de texto.'
      },
      descripcion: {
       bsonType: 'string',
       description: 'La descripción de la categoría es obligatoria y debe ser una cadena de texto.'
      }
     }
```

```
    },
    sucursalId: {
     bsonType: 'string',
     description: 'El ID de la sucursal es obligatorio y debe ser un número entero.'
    }
   }
 }
}
```

Collection <<Categorias>>:

```
db.runCommand({
collMod: "categorias",
validator: {
        $jsonSchema: {
                bsonType: "object",
                required: ["nombre", "descripcion"],
                properties: {
                        nombre: {
                        bsonType: "string",
                        description: "El nombre del producto es obligatorio y debe ser una cadena de texto."
        },
                        descripcion: {
                        bsonType: "string",
                        description: "La descripcion del producto es obligatoria y debe ser una cadena de texto."
}}}});
```

Collection <<Proveedor>>:

```
{
  $jsonSchema: {
```

```
    bsonType: 'object',
    required: [
      'nit',
      'nombre',
      'contacto',
      'productos'
    ],
    properties: {
      nit: {
        bsonType: 'string',
        description: 'El NIT del proveedor es obligatorio y debe ser un número entero.'
      },
      nombre: {
        bsonType: 'string',
        description: 'El nombre del proveedor es obligatorio y debe ser una cadena de texto.'
      },
      contacto: {
        bsonType: 'string',
        description: 'El contacto es obligatorio y debe ser una cadena de texto representando un número de teléfono.'
      },
      productos: {
        bsonType: 'array',
        description: 'La lista de productos es obligatoria.',
        items: {
          bsonType: 'object',
          required: [
            'nombre',
            'precio',
```

```
        'detallesEmpacado',
        'categoria'
      ],
      properties: {
        nombre: {
          bsonType: 'string',
          description: 'El nombre del producto es obligatorio y debe ser una cadena de texto.'
        },
        precio: {
          bsonType: 'double',
          description: 'El precio del producto es obligatorio y debe ser un número decimal.'
        },
        detallesEmpacado: {
          bsonType: 'string',
          description: 'Los detalles del empaque son obligatorios y deben ser una cadena de texto.'
        },
        categoria: {
          bsonType: 'object',
          description: 'La categoría es obligatoria y debe incluir los detalles de la misma.',
          required: [
            'codigo',
            'nombre',
            'descripcion'
          ],
          properties: {
            codigo: {
              bsonType: 'string',
```

```
            description: 'El código de la categoría es obligatorio y debe ser una cadena de texto.'
          },
          nombre: {
            bsonType: 'string',
            description: 'El nombre de la categoría es obligatorio y debe ser una cadena de texto.'
          },
          descripcion: {
            bsonType: 'string',
            description: 'La descripción de la categoría es obligatoria y debe ser una cadena de texto.'
          }
        }
      }
    }
  }
}
```

Collection <<OrdenesCompra>>:

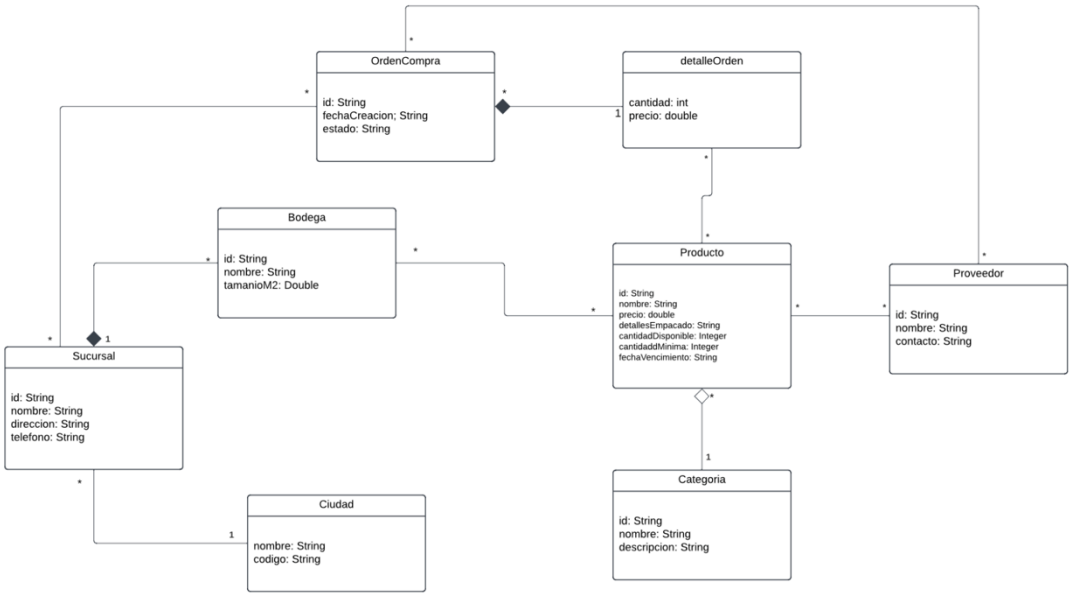
```
{
  $jsonSchema: {
    bsonType: 'object',
    required: [
      'sucursalId',
      'proveedorId',
      'detalle'
    ],
    properties: {
```

```
    sucursalId: {
      bsonType: 'string',
      description: 'El ID de la sucursal es obligatorio y debe ser de tipo string.'
    },
    proveedorId: {
      bsonType: 'string',
      description: 'El ID del proveedor es obligatorio y debe ser una cadena de texto.'
    },
    detalle: {
      bsonType: 'array',
      description: 'El detalle de la orden es obligatorio y debe ser un arreglo de elementos.',
      items: {
        bsonType: 'object',
        required: [
          'productoId',
          'cantidad',
          'precio'
        ],
        properties: {
          productoId: {
            bsonType: 'string',
            description: 'El ID del producto es obligatorio y debe ser de tipo string.'
          },
          cantidad: {
            bsonType: 'int',
            minimum: 1,
            description: 'La cantidad es obligatoria, debe ser un número entero y al menos 1.'
          },
```

```
            precio: {
              bsonType: 'double',
              minimum: 0,
              description: 'El precio es obligatorio, debe ser un número decimal y no negativo.'
            }
          }
        }
      }
    }
  }
}
```

Informe - P3

1. Modelo UML



2. Análisis de la carga de trabajo
   - Identificación entidades y atributos:

     > Bodega
     - id: id único de la bodega.
     - nombre: string nombre de la bodega
     - tamañoM2: double del tamaño en metros cuadrados
     - productos: array que contiene todos los productos de la bodega.

     > Sucursal
     - Id : id único de la sucursal.
     - nombre: string del nombre de la sucursal.
     - direccion: string de la dirección de la sucursal.
     - telefono: string del teléfono de la sucursal.
     - ciudad: entidad ciudad en la que se encuentra la sucursal.
     - bodegas: array que contiene todas las bodegas asociadas a la sucursal.

     > Ciudad
     - nombre: string con el nombre de la ciudad.
     - codigo: string con el código único de ciudad.

     > Orden compra
     - id: string del ID único de la orden de compra.
     - fechaCreacion: string con la fecha de creación de la orden.

- estado: string con el estado actual de la orden.
- sucursalId: string con el id de la sucursal asociada a la orden de compra y referencia un documento de la colección sucursal.
- proveedorId: string con el id del proveedor asociado a la orden de compra y referencia un documento de la colección proveedor.
- detalle: Array con los detalles de la orden.

> DetalleOrden
- productoId: string con el id del producto asociadi al detalle de la orden y referencia un documento de la colección producto.
- cantidad: int con la cantidad de productos.
- precio: double con el precio de cada producto individual.

> Proveedor
- id: ID único de cada proveedor.
- nombre: string con el nombre del proveedor.
- contacto: string con el numero/correo para contactar con el proveedor.
- productos: array con los documentos de productos asociados al proveedor.

> Producto
- id: ID único del producto.
- nombre: string del nombre del producto.
- precio: double con el precio del producto.
- detallesEmpacado: string con el detalle de empacado del producto.
- cantidadDisponible: int con la cantidad disponible de productos.
- cantidadMinima: int con la cantidad mínima requerida de productos.
- fechaVencimiento: string con la fecha de vencimiento del producto.
- categoriaId: string con el id que referencia de la colección categoría.
- sucursalId: string con el id que referencia de la colección sucursal.

> Categoria
- id: Id único de la categoría.
- nombre: string con el nombre de la categoría.
- descripción: string con la descripción de la categoría.

- Cuantificar las entidades
  - Bodega: Existen muchas ya que cada sucursal puede tener una o muchas.
  - Sucursal: La empresa cuenta con muchas sucursales.
  - Ciudad: Hay muchas porque la empresa se encuentra en diferentes ciudades.
  - OrdenCompra: Cada sucursal realiza varias órdenes de compra.

- DetalleOrde: Al haber muchas ordenes de compra también hay muchos detalles de orden.
- Proveedor: La empresa cuenta con muchos proveedores para surtir sus diferentes productos.
- Producto: Se cuenta con muchos diferentes productos.
- Categoría: Muchos productos requieren muchas categorías para ser organizados.

- Operaciones de lectura y escritura para cada entidad. Anexo A

Entities	Operations	Information needed	Type
Bodega	Obtener todas las bodegas.	None	Read
Bodega	Obtener una bodega en específico.	Bodega Id	Read
Bodega	Actualizar una bodega.	Bodega Id	Write
Bodega	Borrar una bodega.	Bodega Id	Write
Categoría	Obtener todas las Categoría.	None	Read
Categoría	Obtener una Categoría en específico.	Categoría id	Read
Categoría	Actualizar una Categoría.	Categoría id	Write
Categoría	Borrar una Categoría.	Categoría id	Write
Categoría	Insertar una categoría.	Categoria info	Write
Ciudad	Obtener todas las Ciudad.	None	Read
Ciudad	Obtener una Ciudad en específico.	Ciudad id	Read
Ciudad	Insertar una Ciudad.	Ciudad info	Write
Ciudad	Borrar una Ciudad.	Ciudad id	Write
OrdenCompra	Obtener todas las Orden Compra.	None	Read
OrdenCompra	Obtener una Orden Compra en específico.	Orden Compra id	Read
OrdenCompra	Insertar una Orden Compra.	Orden Compra info	Write
OrdenCompra	Borrar una Orden Compra.	Orden Compra id	Write
OrdenCompra	Actualizar una Orden Compra.	Orden Compra id	Write
Producto	Obtener todos los Producto.	None	Read
Producto	Obtener un Producto en específico.	Producto id	Read
Producto	Filtrar los productos.	Producto filter	Read
Producto	Crear producto.	Producto info	Write

Producto	Actualizar un producto.	Producto id + producto info	Write
Producto	Eliminar un producto.	Producto id	Write
Proveedor	Obtener todas las Proveedor.	None	Read
Proveedor	Obtener un Proveedor en específico.	Proveedor id	Read
Proveedor	Actualizar un Proveedor.	Proveedor id + proveedor info	Write
Proveedor	Borrar un Proveedor.	Proveedor id	Write
Proveedor	Insertar un Proveedor.	Proveedor info	Write
Recepcionproducto	Insertar un Recepcionproducto	Recepcionproducto info	Write
Recepcionproducto	Obtener todas las Recepcionproducto	None	Read
Sucursal	Obtener todas las Sucursal	None	Read
Sucursal	Crear sucursal	Sucursal info	Write
Sucursal	Obtener una Sucursal en especifico	Sucursal id	Read
Sucursal	Actualizar Sucursal	Sucursal id + Sucursal info	Write
Sucursal	Borrar Sucursal	Sucursal id	Write
Sucursal	Obtener el inventario de la Sucursal	Sucursal id	Read

- Cuantifiquen las operaciones de lectura y escritura para cada entidad. Anexo B

Entities	Operations	Information needed	Type	Rate
Bodega	Obtener todas las bodegas.	None	Read	1000/sec
Bodega	Obtener una bodega en específico.	Bodega Id	Read	1000/sec
Bodega	Actualizar una bodega.	Bodega Id	Write	100/sec
Bodega	Borrar una bodega.	Bodega Id	Write	1000/sec
Categoría	Obtener todas las Categoría.	None	Read	1000/sec
Categoría	Obtener una Categoría en específico.	Categoría id	Read	1000/sec

Categoría	Actualizar una Categoría.	Categoría id	Write	10/min
Categoría	Borrar una Categoría.	Categoría id	Write	1000/sec
Categoría	Insertar una categoría.	Categoria info	Write	10/min
Ciudad	Obtener todas las Ciudad.	None	Read	1000/sec
Ciudad	Obtener una Ciudad en específico.	Ciudad id	Read	1/min
Ciudad	Insertar una Ciudad.	Ciudad info	Write	1/min
Ciudad	Borrar una Ciudad.	Ciudad id	Write	1/week
OrdenCompra	Obtener todas las Orden Compra.	None	Read	1000/sec
OrdenCompra	Obtener una Orden Compra en específico.	Orden Compra id	Read	10/sec
OrdenCompra	Insertar una Orden Compra.	Orden Compra info	Write	1/min
OrdenCompra	Borrar una Orden Compra.	Orden Compra id	Write	1000/sec
OrdenCompra	Actualizar una Orden Compra.	Orden Compra id	Write	10/min
Producto	Obtener todos los Producto.	None	Read	1000/sec
Producto	Obtener un Producto en específico.	Producto id	Read	1000/sec
Producto	Filtrar los productos.	Producto filter	Read	1000/sec
Producto	Crear producto.	Producto info	Write	1/min
Producto	Actualizar un producto.	Producto id + producto info	Write	1000/sec
Producto	Eliminar un producto.	Producto id	Write	1000/sec
Proveedor	Obtener todas las Proveedor.	None	Read	1000/sec
Proveedor	Obtener un Proveedor en específico.	Proveedor id	Read	1000/sec
Proveedor	Actualizar un Proveedor.	Proveedor id + proveedor info	Write	1/min

Proveedor	Borrar un Proveedor.	Proveedor id	Write	1000/sec
Proveedor	Insertar un Proveedor.	Proveedor info	Write	1000/sec
Recepcionproducto	Insertar un Recepcionproducto	Recepcionproducto info	Write	10/min
Recepcionproducto	Obtener todas las Recepcionproducto	None	Read	1000/sec
Sucursal	Obtener todas las Sucursal	None	Read	1000/sec
Sucursal	Crear sucursal	Sucursal info	Write	10/min
Sucursal	Obtener una Sucursal en especifico	Sucursal id	Read	1000/sec
Sucursal	Actualizar Sucursal	Sucursal id + Sucursal info	Write	1000/sec
Sucursal	Borrar Sucursal	Sucursal id	Write	1/min
Sucursal	Obtener el inventario de la Sucursal	Sucursal id	Read	1/sec

3. Descripción las colecciones de datos

    - La lista de entidades con la descripción de cada una de ellas.

A continuación, se presenta un listado con las colecciones desarrolladas en MongoDB. Para esto se presenta una pequeña descripción para cada una de ellas y las colecciones que se encuentran embebidas dentro de estas. Cabe aclarar que en próximos puntos se describirán de mejor manera las características de las colecciones embebidas y se explicará cuáles colecciones se encuentran referenciadas.

- **ordenesCompra**: Esta colección cuenta con la información más importante de la orden generada luego de que un cliente realiza una compra. Cuenta con tres atributos, los cuales son: _id, estado y fechaCreacion. Por otra parte, la entidad “detalle” se encuentra embebida dentro de ordenesCompra, y cuenta con la información detallada de la orden. Sus atributos son: id, cantidad y precio.
- **Productos**: Esta colección presenta la información pertinente de los productos vendidos en SuperAndes, para ello cuenta con los siguientes atributos: id, nombre, precio, detallesEmpacado. Por otra parte, la entidad “categoría” se encuentra embebida dentro del producto y cuenta con los siguientes atributos: codigo, nombre y descripción.
- **Bodegas**: esta colección cuenta con la información pertinente de las diversas bodegas que tiene SuperAndes. Para poder describir esta información, esta

colección cuenta con los siguientes atributos: id, nombre, tamañoM2. En adición, la colección “productos” que fue descrita anteriormente, se encuentra embebida dentro de cada instancia de la colección “bodegas”.

- **Proveedores:** la colección “proveedores” presenta la información de cada proveedor aliado de SuperAndes. Para poder presentar estos datos presenta los siguientes atributos: id, nombre coma contacto. Además, la colección “productos” se encuentra embebida dentro de esta colección.
- **Sucursal:** la colección “sucursales” cuenta con la información importante de las sucursales que son de SuperAndes. Esta información se basa en los siguientes atributos: id, nombre, dirección y teléfono. Por otro lado, la colección “bodegas” se encuentra embebida dentro de esta sucursal. Y, la colección “ciudad”, la cual cuenta con los atributos: nombre y código, también se encuentra embebida en esta colección.

    - Las relaciones entre entidades y su cardinalidad (uno a uno, uno a muchos, o muchos a muchos).

En la siguiente tabla se presentan las diferentes relaciones y cardinalidades presentes entre los diferentes pares de entidades desarrollados en el UML y en las colecciones.

Entidad 1	Entidad 2	Relación	Cardinalidad	Explicación
Sucursal	Ciudad	Asociación simple	Uno a muchos	Una sucursal está asociada a una ciudad. Sin embargo, no depende de ella. Por otra parte, una ciudad puede tener varias sucursales, pero una sucursal está asociada a una ciudad especifica.
Sucursal	Bodega	Composición	Uno a muchos	Las bodegas forman parte de las sucursales, si una sucursal es eliminada, las bodegas asociadas se verán afectadas. Por otro lado, una sucursal puede tener varias bodegas, pero una bodega está asociada a una sucursal.
Bodega	Producto	Asociación simple	muchos a muchos	Los productos son almacenados en bodegas, pero no dependen directamente de las bodegas para existir.

				Respecto a la cardinalidad, las bodegas almacenan diversos productos y los productos pueden estar en diferentes bodegas.
Producto	Categoría	Agregación	Uno a muchos	Una categoría agrupa diversos productos, sin embargo, no las contiene estrictamente. Sobre la cardinalidad, es uno a muchos debido a que un producto solo tiene una categoría, pero una categoría puede agrupar muchos productos.
ordenCompra	detalleOrden	Composición	Uno a muchos	Los detalles de una orden hacen parte de una orden de compra, es decir, si se elimina una orden de compra, se eliminarán sus detalles. Por otra parte. La cardinalidad es uno a muchos, debido a que cada detalle pertenece a una orden de compra, pero una orden puede tener diversos detalles dependiendo de los productos comprados.
ordenCompra	Proveedores	Asociación simple	muchos a muchos	Las ordenes de compran están asociadas a sucursales, pero no dependen directamente de las sucursales para existir. Respecto a la cardinalidad, las sucursales pueden estar relacionadas con diversas órdenes de compra y viceversa
ordenCompra	Sucursal	Asociación simple	muchos a muchos	Las ordenes de compran están asociadas a diversos proveedores que brindan diversos productos, pero no dependen directamente de los proveedores para existir.

				Respecto a la cardinalidad, los proveedores pueden estar relacionados con diversas órdenes de compra y viceversa
Producto	Proveedor	Asociación simple	muchos a muchos	Los proveedores pueden tener asociados diversos productos, sin embargo, no depende su existencia de los productos, ni viceversa. Es decir, si los productos son eliminados, los proveedores no lo serán. Por otro lado, un proveedor puede tener asociados diversos productos, y los productos pueden ser distribuidos por muchos proveedores.
Producto	detalleOrden	Asociación simple	muchos a muchos	Los productos comprados son descritos en los detalles de orden, pero no dependen directamente de los detalles de orden para existir. Por otra parte, los detalles de orden pueden contar con la información de diversos productos que hayan sido comprados, y los productos pueden ser parte de diversos detalles.

- El análisis de selección de esquema de asociación (referenciado o embebido) para cada relación entre entidades. Para ello use la tabla de análisis vista en clase, la cual se retoma en el anexo C, junto con los resultados del análisis de la carga de trabajo (workload), descrita antes.

A continuación, se presentan las diversas asociaciones desarrolladas para las entidades con su respectiva explicación. Además, se presentan algunos parámetros o preguntas que permiten visualizar de mejor manera la decisión que se tomó de embeber o referenciar las diversas entidades, Para ello, las respuestas de estas preguntas se encuentran resaltadas de color amarillo.

1. Bodegas – Productos: la colección productos se encuentra embebida dentro de la colección bodegas. Debido a que los productos hacen parte de la

información necesaria de bodegas. Y tener esa colección embebida permite realizar consultas de una manera más eficaz. Teniendo en cuenta que “Bodegas” es más consultadas, que “productos”, se decidió embeber “productos”, para así disminuir la cantidad de consultas que se deben realizar.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Atomicidad de Consultas	¿La aplicación consulta las piezas de información juntas?	Sí	No
Complejidad de Actualización	¿Las piezas de información se actualizan juntas?	Sí	No
Archivo	¿Deben las piezas de información ser archivadas al mismo tiempo?	Sí	No

2. ordenesCompra - Proveedores: información de los proveedores se encuentra referenciada dentro de la colección órdenes de compra. Debido a que la duplicación de los datos de proveedores no proporciona suficientes ventajas al momento de consultar información- También debido a que la cardinalidad entre estas dos colecciones es de muchos a muchos. Teniendo en cuenta que existen muchos proveedores y muchas órdenes de compra es más adecuado referenciar la información de proveedores para disminuir el consumo en la memoria.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Atomicidad de Consultas	¿La aplicación consulta las piezas de información juntas?	Sí	No
Complejidad de Actualización	¿Las piezas de información se actualizan juntas?	Sí	No

Archivo	¿Deben las piezas de información ser archivadas al mismo tiempo?	Sí	No
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) en el lado hijo de la relación?	No	Sí
Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Tamaño del Documento	¿El tamaño combinado de las piezas de información consume demasiada memoria o ancho de banda para la aplicación?	No	Sí
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí
Carga de Trabajo	¿Las piezas de información se escriben en momentos diferentes en una carga de trabajo intensiva de escritura?	No	Sí
Individualidad	Para el lado hijo de la relación, ¿pueden las piezas existir por sí solas sin un padre?	No	Sí

3. ordenesCompra - Sucursal: La información de las sucursales se encuentra referenciada en la colección de órdenes de compra. Esto debido a que la colección “sucursales” presenta información de otras entidades, lo cual hace que la información sea extensa dentro de esta colección. Es por lo anterior que, si no se referenciara, sería más complicado realizar consultas de información específica, debido a que tanto sucursales como órdenes de compra presentan una gran cantidad de datos. También teniendo en cuenta que las sucursales presentan información de otras entidades podría generarse una dificultad al momento de hacer consultas de las órdenes que no tengan que ver con las entidades presentes en las sucursales. Debido a esto, se decidió referenciar esta información, para simplificar el modelo desarrollado.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Atomicidad de Consultas	¿La aplicación consulta las piezas de información juntas?	Sí	No
Complejidad de Actualización	¿Las piezas de información se actualizan juntas?	Sí	No

Archivo	¿Deben las piezas de información ser archivadas al mismo tiempo?	Sí	No
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) en el lado hijo de la relación?	No	Sí
Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Tamaño del Documento	¿El tamaño combinado de las piezas de información consume demasiada memoria o ancho de banda para la aplicación?	No	Sí
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí
Carga de Trabajo	¿Las piezas de información se escriben en momentos diferentes en una carga de trabajo intensiva de escritura?	No	Sí
Individualidad	Para el lado hijo de la relación, ¿pueden las piezas existir por sí solas sin un padre?	No	Sí

4. ordenesCompra - Detalle: La colección de detalles se encuentra embebida dentro de órdenes de compra, ya que existe una relación de contenencia entre ambas. Esto debido a que, si no existieran las órdenes de compra, no existirían a su vez los detalles de estas órdenes. También el embeber esta información permite simplificar las consultas y el modelo, debido a que, al consultar una orden, se necesitan la cantidad de productos y otros detalles que se encuentran en la otra colección.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Atomicidad de Consultas	¿La aplicación consulta las piezas de información juntas?	Sí	No
Complejidad de Actualización	¿Las piezas de información se actualizan juntas?	Sí	No
Archivo	¿Deben las piezas de información ser archivadas al mismo tiempo?	Sí	No
Cardinalidad	¿Hay una alta cardinalidad (actual o creciente) en el lado hijo de la relación?	No	Sí

Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí
Carga de Trabajo	¿Las piezas de información se escriben en momentos diferentes en una carga de trabajo intensiva de escritura?	No	Sí
Individualidad	Para el lado hijo de la relación, ¿pueden las piezas existir por sí solas sin un padre?	No	Sí

5. Productos – Categoría: La colección categoría se encuentra embebida dentro de la colección productos. Debido a que se realizan más consultas de productos que de las categorías. Por lo tanto, se puede simplificar el modelo y el proceso de consultas, teniendo embebida la información de la Categoría. Teniendo en cuenta que los productos tienen información más relevante de la organización, resulta más práctico., al momento de hacer los requerimientos, tener la información de la categoría en un segundo plano, ya que no es recurrentemente necesitada para los requerimientos de la organización.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Tamaño del Documento	¿El tamaño combinado de las piezas de información consume demasiada memoria o ancho de banda para la aplicación?	No	Sí
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí

6. Proveedores – Productos: Sabiendo que los proveedores pueden manejar una gran lista de productos, se decidió embeber la información de productos dentro de la primera colección mencionada. Teniendo en cuenta la relevancia que tienen los proveedores dentro del proceso de almacenamiento de productos en las sucursales y dentro de los procesos de compra en las órdenes de

compra y su relevancia dentro de los requerimientos, al embeber se simplifica las consultas.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Tamaño del Documento	¿El tamaño combinado de las piezas de información consume demasiada memoria o ancho de banda para la aplicación?	No	Sí
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí

7) Sucursal – Ciudad: La información de ciudad se encuentra embebida dentro de la colección “sucursal”, debido a que la información de la ciudad sólo es requerida cuando se consulta la información de las sucursales de la organización. Por lo tanto, es innecesario tener una colección independiente de las ciudades, ya que dentro de los requerimientos no se realizan consultas únicamente de las ciudades. Al embeber la información de la ciudad se vuelve más fácil consultar la. información total de las sucursales, ya que directamente se podría encontrar la información de la ciudad en la cual se encuentra la sucursal. Por otra parte, sabiendo que las sucursales son una entidad que es recurrentemente consultada, es necesario que en esta se encuentre la mayor cantidad de información de otras colecciones.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Atomicidad de Consultas	¿La aplicación consulta las piezas de información juntas?	Sí	No
Archivo	¿Deben las piezas de información ser archivadas al mismo tiempo?	Sí	No
Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Tamaño del Documento	¿El tamaño combinado de las piezas de información consume demasiada	No	Sí

	memoria o ancho de banda para la aplicación?		
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí

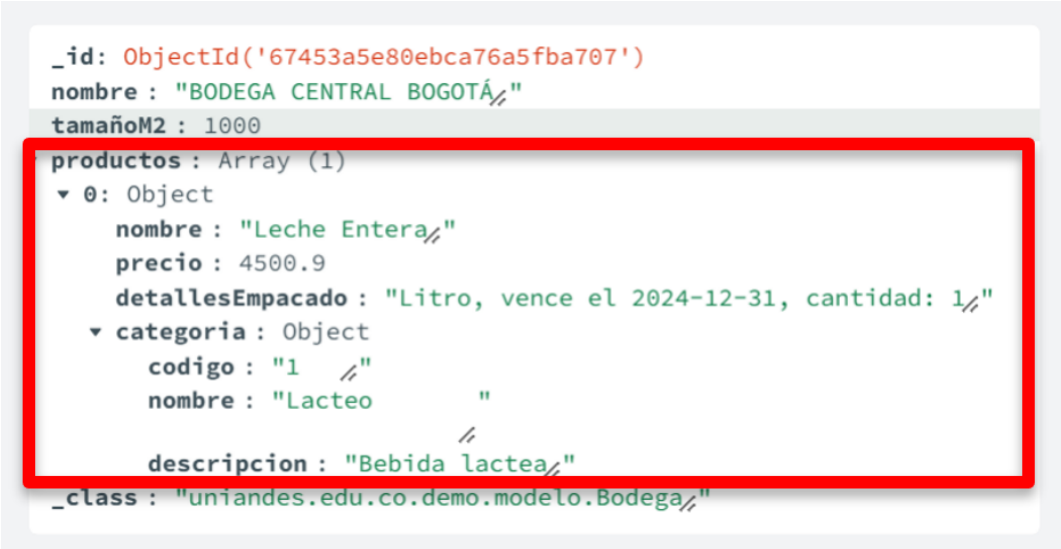
8) Sucursal – Bodega: La información de bodega se encuentra embebida dentro de las sucursales, ya que las sucursales están compuestas de diversas bodegas, por lo que al tener embebida esta información, las consultas de las sucursales se vuelven óptimas, ya que se presentaría la información de todas las bodegas que hacen parte de la sucursal. Por otra parte, teniendo en cuenta que los requerimientos requieren buscar la información de las sucursales y otra información respecto a lo que se almacena dentro de estas bodegas, al tener toda esa información contenida dentro de sucursal se pueden encontrar los datos necesarios sin la necesidad de hacer muchas consultas.

Nombre de la Directriz	Pregunta	Embeber	Referenciar
Simplicidad	¿Mantener juntas las piezas de información llevaría a un modelo de datos y código más simple?	Sí	No
Cohesión	¿Las piezas de información tienen una relación de 'tiene-un', 'contiene' o similar?	Sí	No
Duplicación de Datos	¿Sería muy complicado manejar y no deseable la duplicación de datos?	No	Sí
Tamaño del Documento	¿El tamaño combinado de las piezas de información consume demasiada memoria o ancho de banda para la aplicación?	No	Sí
Crecimiento del Documento	¿La pieza incrustada crecerá sin límites?	No	Sí

- Descripción grafica usando Json de cada relación entre entidades en donde presente un ejemplo de datos junto con el esquema de asociación usado (referenciado o embebido). En el anexo D

A continuación, se presentan las gráficas para representar la información embebida y referenciada de las diferentes relaciones existentes entre las entidades. Para el caso de la información que fue embebida, se presenta en un recuadro rojo la colección que está embebida en la otra. Por otro lado, para la información que está referenciada se usa un recuadro rojo para marcar el id que se utilizó para referenciar la colección y la información referente a la colección referenciada.

- Bodega – Productos



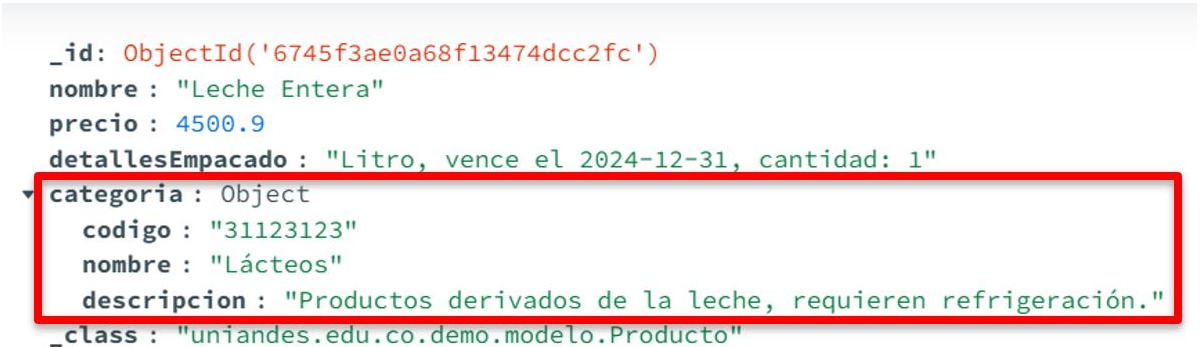
- OrdenesCompra – Detalle

- Detalle – Producto

- OrdenesCompra – Proveedores

OrdenesCompra – Sucursal

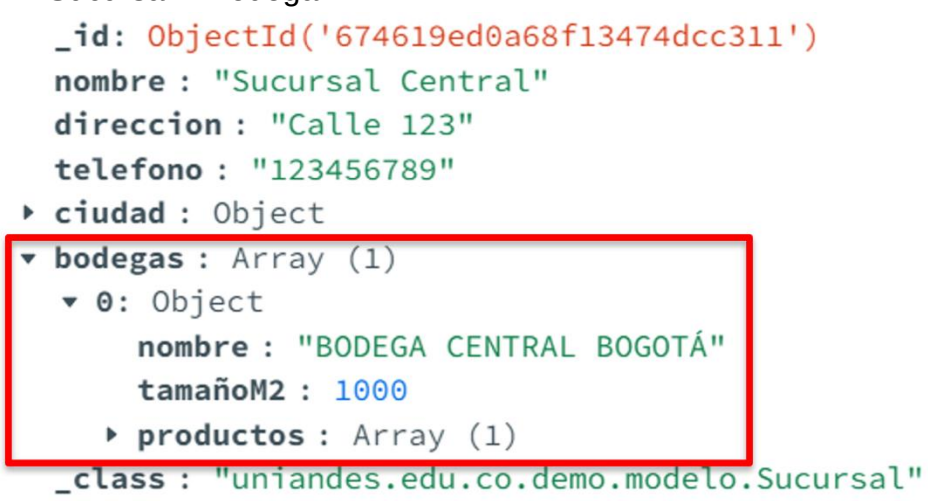
Productos – Categoría



- Proveedores – Productos

- Sucursal – Ciudad

- Sucursal – Bodega



4. Documentación de los requerimientos funcionales

Algunos de los scripts que se usan para insertar documentos que contienen errores se dan a continuación:

- Colección de Bodegas:

```
db.bodegas.insertOne({
  nombre: 12345,
  tamañoM2: "mil",
  productos: [
   {
    nombre: "Leche Entera",
    precio: "cuatro mil",
    detallesEmpacado: 123,
    categoria: {
     codigo: 1,
     nombre: 123,
     descripcion: true
    }
```

```
    }
  ]
});
```

Esta inserción falla porque categoria debe tener una descripción de tipo de cadena de texto.

- Colección de productos:

```
db.productos.insertOne({ precio: 4500.9, detallesEmpacado: "Litro, vence el 2024-12-31, cantidad: 1", categoria: { codigo: "1", nombre: "Lácteo", descripcion: "Bebida láctea" } });
```

La inserción falla por un campo faltante, “nombre” is missing.

- Colección de Sucursales:

```
db.sucursales.insertOne({
  nombre: "Sucursal Central",
  direccion: "Calle 123",
  telefono: 123456789,
  ciudad: {
    nombre: "Bogotá",

::contentReference[oaicite:0]{index=0}
```

La inserción falla porque no se define correctamente los parametros de ciudad.

- Colección de Categoria:

```
db.categorias.insertOne({
  codigo: 123,
  nombre: 456,
  descripcion: 789
});
```

- El campo `codigo` debe ser una cadena de texto, no un número.
- El campo `nombre` debe ser una cadena de texto, no un número.
- El campo `descripcion` debe ser una cadena de texto, no un número.

- Colección proveedores:

```
db.proveedores.insertOne({
  nit: "ABC123",
  nombre: 456,
  contacto: 789,
  productos: [
    {
```

```
    nombre: 123,
    precio: "mil",
    detallesEmpacado: 456,
    categoria: {
     codigo: 1,
     nombre: 2,
     descripcion: 3
    }
   }
 ]
});
```

- El campo `nit` debe ser un número entero, no una cadena de texto.
- El campo `nombre` debe ser una cadena de texto, no un número.
- El campo `contacto` debe ser una cadena de texto, no un número.
- El campo `nombre` del producto debe ser una cadena de texto, no un número.
- El campo `precio` del producto debe ser un número decimal, no una cadena de texto.

Colección de ordenes de compra

```
db.ordenesCompra.insertOne({
 sucursalId: "123",
 proveedorId: "ABC",
 detalle: [
  {
   productoId: "456",
   cantidad: "diez",
   precio: "cien"
  }
 ]
});
```

Los errores se dan porque:

El campo `sucursalId` debe ser de tipo ObjectId, no una cadena de texto.