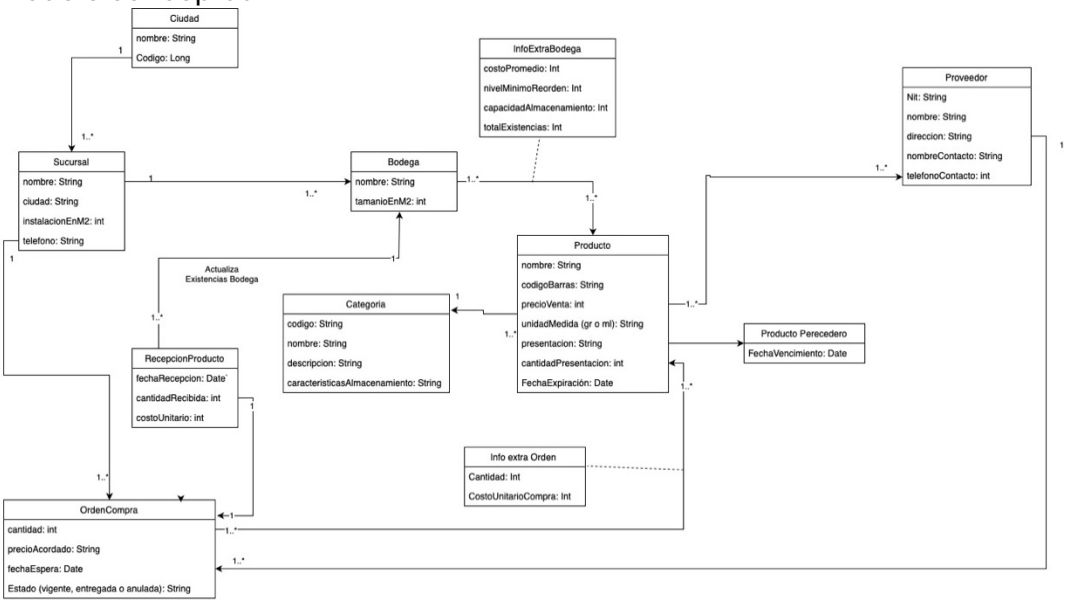
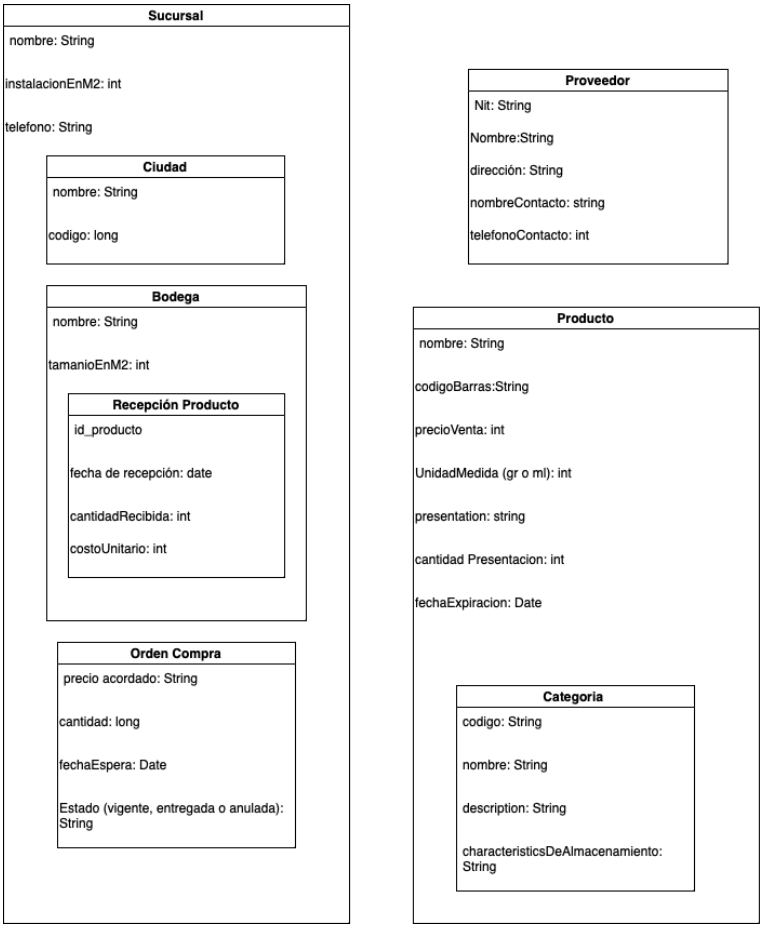
Proyecto 3 – implementación
Grupo 1

1. Modelo conceptual



2. Diseño de base de datos



a. Análisis de la carga de trabajo (workload)
    a) Entidades y atributos:
        1. Ciudad: Atributos: nombre, código.
        2. Sucursal: Atributos: nombre, ciudad, instalaciónEnM2, teléfono.
        3. Bodega: Atributos: nombre, tamañoEnM2, sucursal.
        4. Producto:Atributos: nombre, códigoBarras, precioVenta, unidadMedida, presentación, cantidadPresentación, fechaExpiración, categoría.
        5. RecepciónProducto: Atributos: fechaRecepción, cantidadRecibida, costoUnitario.
        6. OrdenCompra: Atributos: cantidad, precioAcordado, fechaEspera, estado.
        7. Proveedor: Atributos: nit, nombre, dirección, nombreContacto, teléfonoContacto.
        8. Categoría:Atributos: código, nombre, descripción, característicasAlmacenamiento.
    b) Cuantificar entidades:
        1. Ciudad: 2 registros (Bogotá, Bucaramanga).
        2. Sucursal: 4 registros (Norte, Sur en Bucaramanga; Occidente, Oriente en Bogotá).
        3. Bodega: 4 registros (una por sucursal).
        4. Producto: Al menos 24 registros (3 productos por categoría, 8 categorías).
        5. RecepciónProducto: Variable (según operaciones).
        6. OrdenCompra: Variable (según operaciones).
        7. Proveedor: Al menos 5 registros.
        8. Categoría: 8 registros (perecederos, no perecederos, aseo, congelados, prendas de vestir, muebles, herramientas, electrodomésticos).
    c) Operaciones de lectura y escritura:

Entidad	Operaciones	Información requerida	Tipo
Ciudad	Consultar ciudades disponibles	Relación con sucursales y nombre	Lectura
Sucursal	Consultar sucursales por ciudad	Nombre, ciudad, relación con bodegas	lectura
Bodega	Consultar productos almacenados, capacidad	Productos, capacidad	lectura

Producto	Consultar productos por categoría o sucursal y actualizar detalles del producos	Detalles del producto, Nuevos datos (precio, existencias)	Lectura y escritura
RecepciónProducto	Consultar historial de recepciones	Cantidad, costo	Escritura
OrdenCompra	Consultar órdenes activas, crear nueva orden	Productos solicitados y productos y cantidades	Escritura y lectura
Proveedor	Consultar datos de proveedor	Datos de contacto	lectura
Categoría	Consultar categorías y productos relacionados	Clasificación	lectura

3. Cuantificación de las operaciones:

Entidad	**Operaciones**	**Información requerida**	**Tipo**	**Frecuencia**
Ciudad	Consultar ciudades disponibles	Relación con sucursales y nombre	Lectura	10
Sucursal	Consultar sucursales por ciudad	Nombre, ciudad, relación con bodegas	lectura	50
Bodega	Consultar productos almacenados, capacidad	Productos, capacidad	lectura	100
Producto	Consultar productos por categoría o sucursal y actualizar detalles del producto	Detalles del producto, Nuevos datos (precio, existencias)	Lectura y escritura	300
RecepciónProducto	Consultar historial de recepciones	Cantidad, costo	Escritura	200

OrdenCompra	Consultar órdenes activas, crear nueva orden	Productos solicitados y productos y cantidades	Escritura y lectura	150
Proveedor	Consultar datos de proveedor	Datos de contacto	lectura	20
Categoría	Consultar categorías y productos relacionados	Clasificación	lectura	50

a. Descripción de la colección y relaciones
   a) Lista de entidades con descripción
      - Ciudad: Información básica de las ciudades donde operan las sucursales.
      - Sucursal: Representa los puntos de operación de la empresa.
      - Bodega: Espacio físico asociado a una sucursal.
      - Producto: Bienes comercializados, con detalles sobre precio, presentación y almacenamiento.
      - RecepciónProducto: Registro de ingreso de productos a la bodega.
      - OrdenCompra: Solicitud de productos a proveedores.
      - Proveedor: Información de los proveedores asociados.
      - Categoría: Clasificación de productos.
   b) Relaciones y cardinalidades
      - Ciudad - Sucursal: 1 a muchos.
      - Sucursal - Bodega: 1 a 1.
      - Bodega - Producto: 1 a muchos.
      - Producto - Categoría: Muchos a 1.
      - Proveedor - Producto: Muchos a muchos.
      - Sucursal - OrdenCompra: 1 a muchos.
      - OrdenCompra - RecepciónProducto: 1 a muchos.
   c) Selección del esquema

Relación	**Esquema**	**Justificación**
Ciudad - Sucursal	Embebido	Bajo crecimiento de ciudades
Sucursal - Bodega	Embebido	Relación fija de 1:*
Bodega - Producto	Referenciado	Alta cardinalidad, uso compartido.
Producto - Categoría	Embebido	Baja complejidad, estático.
OrdenCompra - Producto	Referencia	Manejo independiente de datos

   d) Ejemplo de JSON: Sucursal – Bodega (embebido)

```
{
 "sucursal": {
  "nombre": "Sucursal Norte",
  "ciudad": "Bucaramanga",
  "bodega": {
   "nombre": "Bodega Principal",
   "tamañoEnM2": 500
  }
 }
}
```

b. Implementación MongoDB: para crear las colecciones principales podemos utilizar la consola y ejecutar este comando: use SuperAlpes;

```
// Crear la colección de productos
db.createCollection("productos", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["codigoBarras", "nombre", "precioVenta", "categoria"],
      properties: {
        codigoBarras: {
          bsonType: "int",
          description: "Debe ser un número entero único y es requerido"
        },
        nombre: {
          bsonType: "string",
          description: "Debe ser una cadena y es requerido"
        },
        precioVenta: {
          bsonType: "int",
          description: "Debe ser un número entero y es requerido"
        },
        unidadMedida: {
          bsonType: "int",
          description: "Debe ser un número entero"
        },
        presentacion: {
          bsonType: "string",
          description: "Debe ser una cadena"
        },
        cantidadPresentacion: {
          bsonType: "int",
          description: "Debe ser un número entero"
        },
```

```
            fechaVencimiento: {
              bsonType: "date",
              description: "Debe ser una fecha válida"
            },
            categoria: {
              bsonType: "array",
              items: {
                bsonType: "object",
                required: ["codigo", "nombre"],
                properties: {
                  codigo: {
                    bsonType: "int",
                    description: "Debe ser un número entero único y es
requerido"
                  },
                  nombre: {
                    bsonType: "string",
                    description: "Debe ser una cadena y es requerido"
                  },
                  descripcion: {
                    bsonType: "string",
                    description: "Debe ser una cadena"
                  },
                  caracteristicasDeAlmacenamiento: {
                    bsonType: "string",
                    description: "Debe ser una cadena"
                  }
                }
              }
            }
          }
        }
      }
    }
});

// Crear la colección de proveedores
db.createCollection("proveedor", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["nit", "nombre", "direccion", "nombreContacto",
"telefonoContacto"],
      properties: {
        nit: {
```

```
            bsonType: "int",
            description: "Debe ser un número entero único y es requerido"
          },
          nombre: {
            bsonType: "string",
            description: "Debe ser una cadena y es requerido"
          },
          direccion: {
            bsonType: "string",
            description: "Debe ser una cadena y es requerido"
          },
          nombreContacto: {
            bsonType: "string",
            description: "Debe ser una cadena y es requerido"
          },
          telefonoContacto: {
            bsonType: "long",
            description: "Debe ser un número entero largo y es requerido"
          }
        }
      }
    }
});

// Crear la colección de sucursales
db.createCollection("sucursal", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["id", "nombre", "direccion", "bodega"],
      properties: {
        id: {
          bsonType: "int",
          description: "Debe ser un número entero único y es requerido"
        },
        nombre: {
          bsonType: "string",
          description: "Debe ser una cadena y es requerido"
        },
        direccion: {
          bsonType: "string",
          description: "Debe ser una cadena y es requerido"
        },
        ciudad: {
```

```
            bsonType: "array",
            items: {
              bsonType: "object",
              required: ["codigo", "nombre"],
              properties: {
                codigo: {
                  bsonType: "int",
                  description: "Debe ser un número entero único y es
requerido"
                },
                nombre: {
                  bsonType: "string",
                  description: "Debe ser una cadena y es requerido"
                }
              }
            }
          },
          bodega: {
            bsonType: "array",
            items: {
              bsonType: "object",
              required: ["id", "nombre", "tamanioEnM2"],
              properties: {
                id: {
                  bsonType: "int",
                  description: "Debe ser un número entero único y es
requerido"
                },
                nombre: {
                  bsonType: "string",
                  description: "Debe ser una cadena y es requerido"
                },
                tamanioEnM2: {
                  bsonType: "int",
                  description: "Debe ser un número entero y es requerido"
                }
              }
            }
          }
        }
      }
    }
});
```

c. Esquemas de validación: en el script anterior incluimos los esquemas de validación

4. Documentación de RF y RFC
    a. RF1 Sucursal:
        a) Descripción general: Este código implementa la funcionalidad para crear una sucursal, almacenando información básica como su nombre, tamaño en metros cuadrados, dirección y la ciudad a la que pertenece.
        b) Relación con la ciudad: Cada sucursal está asociada a una única ciudad, representada como un objeto embebido dentro de la sucursal.
        c) API REST: El endpoint /sucursal/save/new permite registrar una nueva sucursal en la base de datos MongoDB a través de un POST request con los datos correspondientes.
    b. RF2 Bodega:
        a) Descripción general: Este código permite la creación y eliminación de bodegas asociadas a una sucursal específica, considerando que una sucursal puede contener una o más bodegas.
        b) Crear una bodega: El endpoint POST /{sucursalId}/bodega agrega una nueva bodega a una sucursal existente, actualizando su lista de bodegas en la base de datos.
        c) Eliminar una bodega: El endpoint DELETE /{sucursalId}/bodega/{bodegaId} elimina una bodega específica de una sucursal, actualizando la información almacenada.
    c. RF3 Proveedores:
        a) Descripción general: Este código permite la gestión de proveedores en la base de datos, específicamente la creación y actualización de información asociada a cada proveedor.
        b) Crear un proveedor: El endpoint POST /proveedor/save/new registra un nuevo proveedor con información como NIT, nombre, dirección, nombre de contacto y teléfono de contacto.
        c) Actualizar un proveedor: El endpoint PUT /proveedor/update/{nit} permite modificar los datos de un proveedor identificado por su NIT, actualizando detalles como dirección y nombre de contacto.
    d. RF4 Categoría
        a) Descripción general: Este código gestiona la creación y consulta de categorías embebidas dentro de productos, asegurando que cada categoría contenga información como código, nombre, descripción y características de almacenamiento.
        b) Crear una categoría: El endpoint POST /{id}/categoria permite agregar una nueva categoría a un producto existente, identificándolo mediante su código de barras (id).

c) Consultar categorías: Los endpoints GET /categoria/codigo/{codigo} y GET /categoria/nombre/{nombre} permiten buscar productos asociados a una categoría específica por su código o nombre, devolviendo los productos correspondientes.

e. RF5 Producto
   a) Descripción general: Este código permite gestionar productos, asegurando que cada uno esté asociado a una categoría existente, y soporta operaciones de creación, lectura y actualización.
   b) Crear un producto: El endpoint POST /productos/new/save registra un nuevo producto, asociándolo a una o más categorías, e incluye detalles como precio, presentación y fecha de vencimiento.
   c) Leer y actualizar productos: Se permite consultar productos por código o nombre, devolviendo toda su información incluyendo las categorías asociadas, y actualizar detalles del producto mediante su código.
f. RF6 Y RF7 Orden de compra:
   a) Crear una Orden de Compra
      1. Descripción general: Permite registrar una nueva orden de compra asociada a una sucursal, con un encabezado y detalle. La orden de compra incluye la fecha de creación (automática), el proveedor, la sucursal, la fecha esperada de entrega y los productos a comprar.
      2. Detalles del encabezado:
         a) Fecha de creación: Se asigna automáticamente como la fecha actual del sistema.
         b) Sucursal: Asociada por el ID proporcionado.
         c) Proveedor: Seleccionado mediante el ID proporcionado.
         d) Fecha esperada de entrega: Especificada por el usuario.
      3. Detalles del pedido: Lista de productos con sus respectivas cantidades y precios unitarios. Estado inicial de la orden: "vigente".
      4. Implementación en la API: Endpoint: POST /ordencompra/{sucursalId}/save. Procesa el encabezado y detalle para crear y guardar la orden de compra asociada a una sucursal.
   b) Leer una Orden de Compra
      1. Descripción general: Recupera información completa de una orden de compra, incluyendo el encabezado y el detalle, con base en su ID.
      2. Detalles devueltos:
         a) Encabezado: Incluye fecha de creación, sucursal, proveedor, y fecha esperada de entrega.

b) Detalle: Lista de productos con cantidades, precios y otros detalles asociados.

3. Implementación en la API: Endpoint: GET /ordencompra/{sucursalId}/ordencompra/{ordenCompraId}. Busca la orden dentro de las órdenes de compra asociadas a una sucursal específica.

g. RFC1:
   a) Descripción

Este requerimiento permite obtener la información detallada de productos que cumplen con criterios específicos ingresados por el usuario, incluyendo el rango de precio, fecha de vencimiento, categoría, o disponibilidad en una sucursal específica.

   b) Detalles

1. Parámetros de búsqueda:
   - Rango de precios: Productos dentro de un precio mínimo y máximo especificado.
   - Fecha de vencimiento: Productos cuya fecha de vencimiento sea antes o después de una fecha dada.
   - Categoría: Productos que pertenezcan a una categoría específica.
   - Sucursal: Productos disponibles en una sucursal específica.
2. Implementación:
   - Endpoint para buscar productos:
     - URL: POST /productos/filtrar
     - Entrada: JSON con los parámetros de filtro (precioMin, precioMax, fechaVencimiento, categoría, idSucursal).
     - Salida: Lista de productos que cumplen los criterios.
   - Método: El controlador aplica los filtros y devuelve los productos que coincidan con los criterios.

h. RFC2:
   a) Descripción general: Este código implementa funcionalidades para gestionar el inventario de productos en las bodegas de una sucursal, generando un reporte detallado o actualizando los productos en las bodegas.
   b) Obtener inventario: El endpoint GET /sucursal/{sucursalId}/inventario genera un reporte por bodega, mostrando el nombre del producto, cantidad disponible, cantidad mínima requerida, y el costo promedio.
   c) Añadir productos a una bodega: El endpoint POST /sucursal/{sucursalId}/bodega/{bodegaId}/recepcion permite registrar una recepción de producto en una bodega específica de la sucursal, actualizando la información del inventario.

5. Escenarios de prueba en postman: los escenarios fueron creados teniendo en cuenta los datos que ya estaban en la base de datos y las URL dadas en el controller. Este es un ejemplo de una implementación de un caso de prueba en postman del requerimiento funcional 1: