Documento de informe pruebas proyecto
Sistemas Transaccionales

Juan Pablo Reyes 202310852
Juan Diego Rodríguez 202221822
Daniela Munar 202311392

Balance plan de pruebas:

1. Pruebas generales de Inserción

Para los requerimientos que fueron llevados a cabo en esta entrega de proyecto, se lograron desarrollar con éxito todas las pruebas de funcionamiento que fueron propuestas desde un inicio. Para esta situación, se presenta el balance de las pruebas de inserción para cada una de las entidades del modelo relacional, a través de las cuáles opera el programa, cabe recalcar que, ya que son las pruebas de inserción, no aplicarían las relaciones que componen el sistema;

- ***Bodega:***

En el caso de bodega, la prueba al ser de inserción consistía en emplear el siguiente objeto de prueba, insertarlo, para que posteriormente se pudiera hallar en la base de datos de forma común y corriente.

```
{
  "nombre": "Bodega Chapinero",
  "tamano": 2000,
  "sucursal": {
    "id": 4
  }
}
```

Se ejecuta la prueba a través del controlador empleando el path correspondiente ({{BaseURL}}/bodegas/insert). A continuación, se imprime un mensaje descrito desde el controlador de la aplicación:

```
  @PostMapping("/insert")
  public ResponseEntity<String> insertBodega(@RequestBody Bodega bodega)
  {
    bodegaService.insertBodega(bodega);
```

```
    return ResponseEntity.status(HttpStatus.CREATED).body("La bodega '" + bodega.getNombre() + "' ha sido agregada correctamente");
```

En caso de que falle, bien sea por atributos o por valores no válidos, como el que es especificado en la prueba “InsertBodegaWithInvalidSucursalId” en la que ya que se presenta una relación con una sucursal, si esta no es definida de manera correcta, o en otras palabras, si su id no se haya dentro de la base de datos, debería de retornar un error de tipo 500:

```
{
  "nombre": "Bodega Chapinero",
  "tamano": 2000,
  "sucursal": {
    "id": 4545845783
  }
}
```

En la parte posterior se aprecia un ejemplo de una inserción que causaría un error

```
  public void insertBodega(Bodega bodega)
  {
    try
    {
      bodegaRepository.insertBodega(bodega.getNombre(), bodega.getTamano(), bodega.getSucursal().getId());
    }
    catch (Exception e)
    {
      throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Error al insertar la bodega");
    }
  }
```

El programa lanzaría una excepción como la mencionada, que no permite ingresar una bodega de forma correcta.

- ***Categoría:***

Para el caso de la categoría, es importante resaltar el hecho de que esta por el contrario de Bodega, es independiente de relaciones, por la que no existen objetos directamente arraigados a la misma.

Igualmente, en el body de la prueba se pretende ingresar un objeto que cuenta con los siguientes atributos y estructura:

```
{
  "nombre": "Muebles de Oficina",
  "descripcion": "Mobiliario diseñado específicamente para el uso en entornos de trabajo.",
  "caracteristicas": "Fabricados con materiales resistentes como madera y acero. Requieren montaje y espacio adecuado para su instalación."
}
```

Se ejecuta la prueba de forma normal haciendo uso del path:

```
{{BaseURL}} /categorias/insert
```

En caso de que el ingreso de la categoría sea exitoso se imprime un mensaje a través del controlador. Igualmente, cabe destacar que el posible error que podría ser generado al tratar de insertar una nueva categoría sería bien sea por la estructura que se emplee, o también por la mala definición de los atributos.

Para dicha situación el service lanzaría un error de tipo 500:

```
    public void insertCategoria(Categoria categoria)
    {
        try
        {
            categoriaRepository.insertCategoria(categoria.getNombre(), categoria.getDescripcion(), categoria.getCaracteristicas());
        }
        catch (Exception e)
        {
            throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Error al insertar la categoria");
        }
    }
```

- ***Ciudad:***

Para el caso de ciudad, la creación de un nuevo objeto de este tipo en la base de datos resulta mucho más simple, ya que además de su identificador normal, cuenta solamente con un atributo, que es su nombre.

Para eso mismo, se provee un ejemplo de un body que aplicaría para ser una ciudad:

```
{
  "nombre": "Cali"
```

```
}
```

Al ejecutarse la prueba, se emplea el path ({{BaseURL}}/ciudades/insert).

Posteriormente, se debería de imprimir un mensaje de ingreso existoso. Ahora bien, para que esta prueba en particular falle, se debería de ingresar mal el nombre del atributo, ya que de por si al ser un VARCHAR2, de gran tamaño, las restricciones en cuanto al valor que se le asigne son prácticamente nulas.

No obstante, en caso de que se presente el fallo comentado con anterioridad, el service lanzaría la siguiente excepción, evitando de esa forma la correcta creación de la nueva ciudad.

```
    public void insertCiudad(Ciudad ciudad)
    {
        try
        {
            ciudadRepository.insertCiudad(ciudad.getNombre());
        }
        catch (Exception e)
        {
            throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Error al insertar la
ciudad");
        }
    }
```

Se observa en la parte posterior el mensaje que debería lanzar el servicio llamado por el controlador en caso de presentar algún fallo en el ingreso de los datos.

- ***Producto:***

Para el caso de producto, nos encontramos con la entidad que posee la mayor cantidad de complejidades o restricciones a la hora de ser creado o insertado, esto debido al gran número de atributos.

A continuación, se presenta un ejemplo de body que muestra el cómo debería de ser la estructura a tener en cuenta a la hora de ingresar un producto en la base de datos:

```
{
    "nombre": "Galletas de Avena",
    "costoBodega": 3500.0,
    "precioUnitario": 2500.0,
    "presentacion": "Paquete de 12 galletas.",
    "cantidad": 500,
```

```
  "unidadMedida": "gr",
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "fechaVencimiento": "2025-12-31",
  "codigoBarras": "z8z8z8z8z8",
  "categoria": {
    "id": 1
  }
}
```

Para la prueba puntualmente, se sigue de forma similar el para llevar la acción intencionada ({{BaseURL}}/productos/insert).

El controlador en conjunto con el servicio imprimirán un mensaje de éxito en caso de que no haya habido errores al ingresar los datos correspondientes, sin embargo, hay varios casos que harían que el programa lance un error de tipo. Estos se enumerarán a continuación con su respectiva causa de error:

1. Nombre, código de barras y/o presentación en blanco: un ejemplo de este error se presenta en la parte inferior.

```
{
  "nombre": "  ",
  "costoBodega": 3500.0,
  "precioUnitario": 5500.0,
  "presentacion": "   ",
  "cantidad": 500,
  "unidadMedida": "gr",
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "fechaVencimiento": "2025-12-31",
  "codigoBarras": "  ",
  "categoria": {
    "id": 1
  }
}
```

2. Costo de bodega, peso y/o volumen deben ser mayores a 0: un ejemplo se presenta a continuación

```
{
  "nombre": "Galletas de Avena",
  "costoBodega": 0.0,
  "precioUnitario": 5500.0,
  "presentacion": "Paquete de 12 galletas.",
  "cantidad": 500,
  "unidadMedida": "gr",
```

```
  "volumenEmpaque": 0.0,
  "pesoEmpaque": 0.0,
  "fechaVencimiento": "2025-12-31",
  "codigoBarras": "z8z8z8z8z8",
  "categoria": {
    "id": 1
  }
}
```

3. Precio unitario debe ser mayor al costo de bodega: un ejemplo se presenta a continuacion:

```
{
  "nombre": "Galletas de Avena",
  "costoBodega": 3500.0,
  "precioUnitario": 2500.0,
  "presentacion": "Paquete de 12 galletas.",
  "cantidad": 500,
  "unidadMedida": "gr",
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "fechaVencimiento": "2025-12-31",
  "codigoBarras": "z8z8z8z8z8",
  "categoria": {
    "id": 1
  }
}
```

4. La cantidad de la presentación del producto debe ser mayor a 0: un ejemplo de error se presenta a continuación.

```
{
  "nombre": "Galletas de Avena",
  "costoBodega": 3500.0,
  "precioUnitario": 5500.0,
  "presentacion": "Paquete de 12 galletas.",
  "cantidad": 0,
  "unidadMedida": "gr",
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "fechaVencimiento": "2025-12-31",
  "codigoBarras": "z8z8z8z8z8",
  "categoria": {
    "id": 1
  }
}
```

5. Los productos de la categoría perecederos deben tener fecha de vencimiento: un ejemplo de error se presenta a continuación.


```
{
  "nombre": "Galletas de Avena",
  "costoBodega": 3500.0,
  "precioUnitario": 5500.0,
  "presentacion": "Paquete de 12 galletas.",
  "cantidad": 500,
  "unidadMedida": "gr",
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "codigoBarras": "z8z8z8z8z8",
  "categoria": {
    "id": 1
  }
}
```


6. Los productos distintos a la categoría perecederos no deberían tener fecha de vencimiento: un ejemplo de error se muestra a continuación.


```
{
  "nombre": "Galletas de Avena",
  "costoBodega": 3500.0,
  "precioUnitario": 5500.0,
  "presentacion": "Paquete de 12 galletas.",
  "cantidad": 500,
  "unidadMedida": "gr",
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "codigoBarras": "z8z8z8z8z8",
  "categoria": {
    "id": 3535354
  }
}
```


7. La fecha de vencimiento del producto no puede ser anterior a la fecha actual: un ejemplo de error se muestra a continuación.


```
{
  "nombre": "Galletas de Avena",
  "costoBodega": 3500.0,
  "precioUnitario": 5500.0,
  "presentacion": "Paquete de 12 galletas.",
  "cantidad": 500,
  "unidadMedida": "gr",
```

```
  "volumenEmpaque": 50.0,
  "pesoEmpaque": 300.0,
  "fechaVencimiento": "2025-12-31",
  "codigoBarras": "  ",
  "categoria": {
    "id": 1
  }
}
```

En caso de que se llegase a incumplir alguna de las siguientes restricciones, se imprimiría un BAD REQUEST o INTERNAL SERVER ERROR.

- ***Proveedor:***

En el caso del proveedor, si bien existen diversos atributos, no existen muchos márgenes de error a la hora de ingresar uno.

A continuación se muestra un ejemplo de un body correcto si se pretende ingresar un nuevo proveedor, ello empleando el path ({{BaseURL}}/proveedores/insert):

```
{
  "nit": 1234567000,
  "nombre": "Proveedor de Tecnología",
  "direccion": "Calle 45 #10-20, Bogotá",
  "nombreContacto": "Juan Pérez",
  "telefonoContacto": 3152715537
}
```

Para este caso, como muchos anteriores, no existen muchas restricciones específicas que se deban de considerar a la hora de insertar un nuevo proveedor, sin embargo, en caso de que exista algún error de sintaxis o de tipos de datos no correspondidos en alguna casilla, es posible que se genere un error 500 de INTERNAL SERVER o BAD REQUEST.

- ***Sucursal:***

Para el caso de la sucursal, así como el caso anterior, no existen muchas restricciones altamente importantes a tener en cuenta a la hora de insertar un objeto de este tipo.

A continuación se presenta un ejemplo correcto de prueba empleando el path correspondiente ({{BaseURL}}/sucursales/insert):

```
{
   "nombre": "Sucursal Principal",
   "direccion": "Calle 123 #45-67, Bogotá",
   "telefono": 3216549870,
   "tamano": "5000 m2",
   "ciudad": {
      "id": 34
   }
}
```

En el caso de sucursal, así como en el de proveedor, si no se ingresan de forma correcta los datos que se pretenden manejar, es muy posible que se imprima un error de tipo 500 de INTERNAL SERVER o incluso BAD REQUEST.

- ***OrdenCompra:***

La presente clase o entidad como su nombre lo indica, refiere a la creación de las órdenes de compra que son realizadas por una sucursal y que se relaciona con una serie de productos.

A continuación, se hará ejemplificación de una orden de compra de correcta sintaxis y estructura:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-17",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```

Si se cumple exitosamente la petición, entonces a través del controlador y el servicio se imprimirá un mensaje de correcta inserción. Sin embargo, es bastante importante tener en cuenta una serie de restricciones que podría causar una falla en la creación de una orden de compra. Estos se mencionarán y se ejemplificarán:

1. La fecha de entrega de la orden de compra no puede ser anterior a la fecha actual. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-01-10",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```

2. La bodega de la orden de compra debe pertenecer a la sucursal correspondiente. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-10",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 3
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```


3. El proveedor correspondiente debe suministrar productos a la sucursal correspondiente. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-10",
        "proveedor": {
            "id": 4374573657
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```


4. La orden de compra necesita solicitar un producto al menos. Un ejemplo de error se muestra a continuación:

Aquellas restricciones mencionadas previamente hacen parte de la generalidad de la creación de la orden de compra, no obstante, ya que esta acción comprende un proceso de transacción, se deben de tener en cuenta las relaciones que resultan sustanciales. Estas mismas se enunciarán y enumerarán en la parte inferior.

Sobre los productos;

1. El proveedor tiene que suministrar el producto solicitado.

2. La cantidad solicitada del producto tiene que ser mayor a 0. No tiene sentido iniciar una orden de compra sin ningún producto para ofertar.

3. El precio del producto debe ser mayor o igual al precio unitario.

Sobre la bodega;

4. La bodega debe tener menos del 70% para de ese modo poder agregar más productos.

Sobre la relación entre el producto y la bodega;

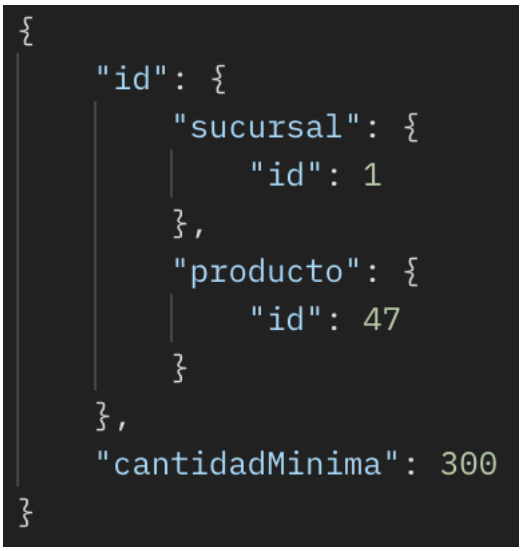
5. El producto tiene que estar registrado en la bodega.

6. La cantidad del producto solicitado no puede exceder la capacidad disponible en la bodega correspondiente.

Cualquier incumplimiento de las anteriores restricciones provocarán la generación del error BAD REQUEST, lanzado por el service a través del controlador.

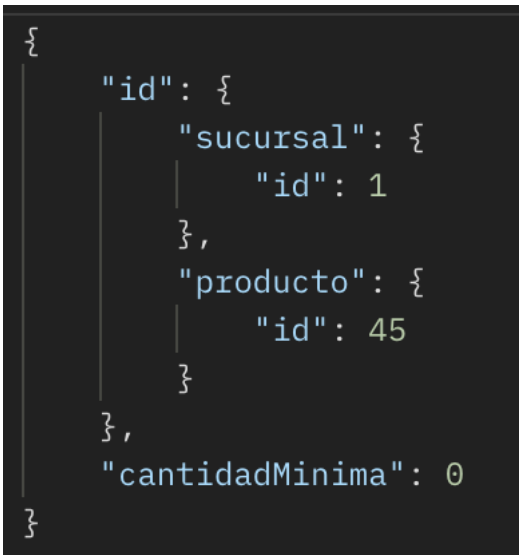
- ***ProductoSucursal:***

En producto sucursal nos encontramos igualmente con una relación que comprende diferentes restricciones y consideraciones para su correcto funcionamiento. Sin embargo, esto se torna más importante a la hora de hablar de la inserción. En la parte inferior se ejemplificará a través de un body un request apropiado para la acción intencionada:

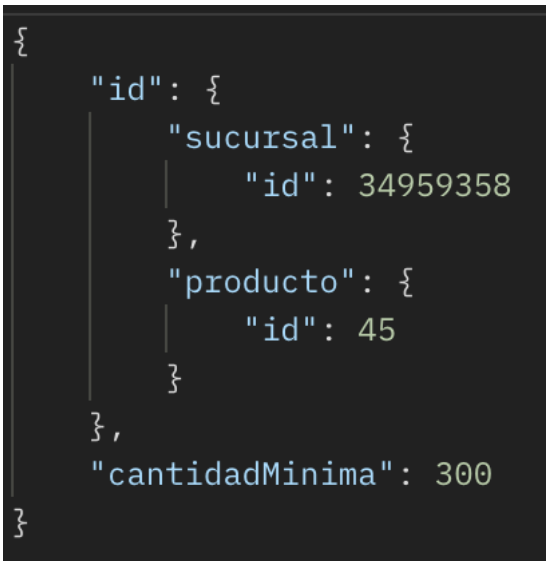


Ahora bien, teniendo en cuenta el hecho de que se trata de una relación entre dos entidades, es pertinente saber las restricciones a las que esta se encuentra arraigada. Estas se enumerarán y ejemplificarán:

1. La cantidad mínima del producto a registrar en la sucursal debe ser mayor a cero. Un ejemplo de error se muestra a continuación:



2. La sucursal debe estar registrada y debe tener bodegas asociadas para así completar la acción. Un ejemplo de error se muestra a continuación:



El no cumplimiento de las anteriores restricciones provocará la impresión de un error de tipo BAD REQUEST e incluso INTERNAL SERVER ERROR.

- ***ProveedorProducto:***

Sobre proveedor producto, a diferencia de los últimos dos análisis de pruebas, no presenta diferentes restricciones que deben considerarse, además de la estructura y

sintaxis a la hora de insertar un objeto, por lo que es un poco más simple de analizar. Igualmente, se presenta un body apropiado para la acción pretendida:

Cabe destacar nuevamente lo mencionado con anterioridad, y es el hecho de que, si bien no hay restricciones especificas a la hora de crear un nuevo objeto, en caso de que no se implementen de manera correcta los atributos o los tipos de datos que deben ser manejados, el programa lanzará un INTERNAL SERVER ERROR.

2. Pruebas de Requerimientos Funcionales

Sobre los requerimientos funcionales es relevante destacar el hecho de que fueron desarrollados con éxito, y que, gracias a la buena distribución de relaciones y organización dentro del proyecto, pudieron probarse empleando diferentes pruebas ya mencionadas en el primer apartado del documento.

De este modo, para este numeral del informe de pruebas, se tendrán en cuenta varias pruebas enunciadas previamente, pero también se evidenciarán unas nuevas, que, a diferencia de aquellas de inserción, **SÍ** poseen scripts, y por tanto serán señalados para el propósito inicial.

- ***Requerimiento Funcional 1***
  - ***Crear una ciudad***

En este caso particular, y como bien fue mencionado en la introducción de este segundo numeral, habrá casos de prueba que se repitan.

Para el caso de ciudad, la creación de un nuevo objeto de este tipo en la base de datos resulta mucho más simple, ya que además de su identificador normal, cuenta solamente con un atributo, que es su nombre.

Para eso mismo, se provee un ejemplo de un body que aplicaría para ser una ciudad:

```
{
  "nombre": "Cali"
}
```

Al ejecutarse la prueba, se emplea el path:

```
{{BaseURL}}/ciudades/insert
```

Posteriormente, se debería de imprimir un mensaje de ingreso existoso. Ahora bien, para que esta prueba en particular falle, se debería de ingresar mal el nombre del atributo, ya que de por si al ser un VARCHAR2, de gran tamaño, las restricciones en cuanto al valor que se le asigne son prácticamente nulas.

No obstante, en caso de que se presente el fallo comentado con anterioridad, el service lanzaría la siguiente excepción, evitando de esa forma la correcta creación de la nueva ciudad.

```
  public void insertCiudad(Ciudad ciudad)
  {
      try
      {
          ciudadRepository.insertCiudad(ciudad.getNombre());
      }
      catch (Exception e)
      {
          throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Error al insertar la
ciudad");
      }
  }
```

Se observa en la parte posterior el mensaje que debería lanzar el servicio llamado por el controlador en caso de presentar algún fallo en el ingreso de los datos.

- ***Requerimiento funcional 2***
  - ***Crear una sucursal***

Para el caso de la sucursal, así como el caso anterior, no existen muchas restricciones altamente importantes a tener en cuenta a la hora de insertar un objeto de este tipo.

A continuación se presenta un ejemplo correcto de prueba empleando el path correspondiente ({{BaseURL}}/sucursales/insert):

```
{
   "nombre": "Sucursal Principal",
   "direccion": "Calle 123 #45-67, Bogotá",
   "telefono": 3216549870,
   "tamano": "5000 m2",
   "ciudad": {
      "id": 34
   }
}
```

En el caso de sucursal, así como en el de proveedor, si no se ingresan de forma correcta los datos que se pretenden manejar, es muy posible que se imprima un error de tipo 500 de INTERNAL SERVER o incluso BAD REQUEST.

- ***Requerimiento funcional 3***
  - ***Crear y borrar una bodega***

En el caso de bodega, la prueba al ser de inserción consistía en emplear el siguiente objeto de prueba, insertarlo, para que posteriormente se pudiera hallar en la base de datos de forma común y corriente.

```
{
   "nombre": "Bodega Chapinero",
   "tamano": 2000,
   "sucursal": {
      "id": 4
   }
}
```

Se ejecuta la prueba a través del controlador empleando el path correspondiente ({{BaseURL}}/bodegas/insert). A continuación, se imprime un mensaje descrito desde el controlador de la aplicación:

```
   @PostMapping("/insert")
   public ResponseEntity<String> insertBodega(@RequestBody Bodega bodega)
   {
      bodegaService.insertBodega(bodega);
      return ResponseEntity.status(HttpStatus.CREATED).body("La bodega '" + bodega.getNombre() + "' ha
sido agregada correctamente");
```

En caso de que falle, bien sea por atributos o por valores no válidos, como el que es especificado en la prueba “InsertBodegaWithInvalidSucursalId” en la que ya que se presenta una relación con una sucursal, si esta no es definida de manera correcta, o en otras palabras, si su id no se haya dentro de la base de datos, debería de retornar un error de tipo 500:

```
{
  "nombre": "Bodega Chapinero",
  "tamano": 2000,
  "sucursal": {
    "id": 4545845783
  }
}
```

En la parte posterior se aprecia un ejemplo de una inserción que causaría un error

```
  public void insertBodega(Bodega bodega)
  {
      try
      {
          bodegaRepository.insertBodega(bodega.getNombre(), bodega.getTamano(),
bodega.getSucursal().getId());
      }
      catch (Exception e)
      {
          throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Error al insertar la
bodega");
      }
  }
```

El programa lanzaría una excepción como la mencionada, que no permite ingresar una bodega de forma correcta.

- ***Borrar una bodega***

Para el caso de eliminación de la bodega, dentro de las pruebas que fueron implementadas no fue necesario el uso de un script o body. Únicamente se empleó el path correcto;

```
{{BaseURL}}/bodegas/13/delete
```

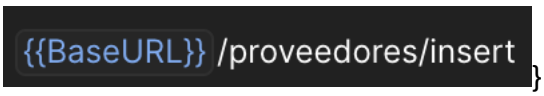
Para su uso correcto se debe de ingresar el id de una bodega existente que se quiera eliminar. De ese mismo modo, existe la restricción general que es el ingreso de un id no existente, un ejemplo de ello sería el path…

Si se llegase a cumplir esta restricción se generaría un error de BAD REQUEST, ya que no se habría hallado el objeto a eliminar.

- ***Requerimiento funcional 4***
  - ***Crear un proveedor***

En el caso del proveedor, si bien existen diversos atributos, no existen muchos márgenes de error a la hora de ingresar uno.

A continuación se muestra un ejemplo de un body correcto si se pretende ingresar un nuevo proveedor, ello empleando el path:

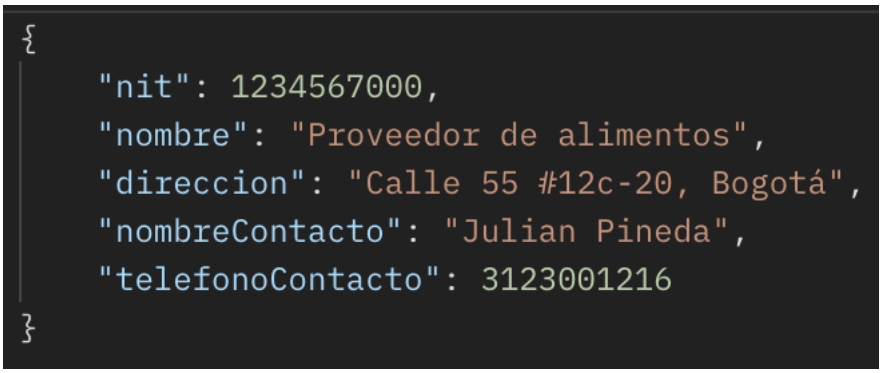


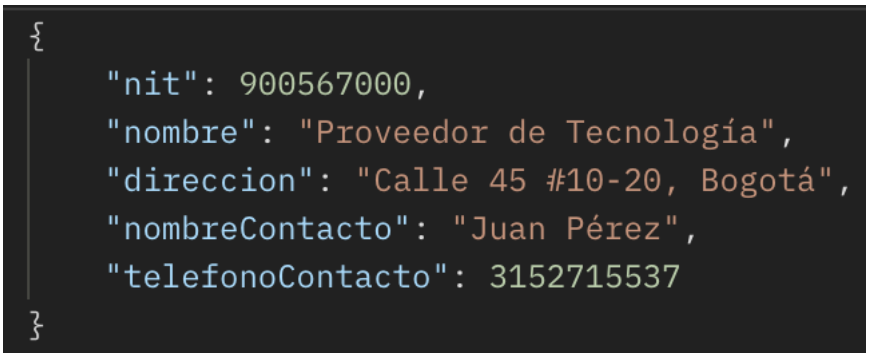

```
{
  "nit": 1234567000,
  "nombre": "Proveedor de Tecnología",
  "direccion": "Calle 45 #10-20, Bogotá",
  "nombreContacto": "Juan Pérez",
  "telefonoContacto": 3152715537
}
```

Para este caso, como muchos anteriores, no existen muchas restricciones específicas que se deban de considerar a la hora de insertar un nuevo proveedor, sin embargo, en caso de que exista algún error de sintaxis o de tipos de datos no correspondidos en alguna casilla, es posible que se genere un error 500 de INTERNAL SERVER o BAD REQUEST.

  - ***Actualizar un proveedor***

En este caso particular, se propondrá un body apto para la correcta actualización de un proveedor, para lo mismo es importante tener en cuenta el hecho de que hay ciertos valores en ocasiones que no están disponibles para su cambio. Para claridad de lo anterior se propondrá un body adecuado para dicha explicación:





En este caso, el primer body es aquel que fue agregado inicialmente, para posteriormente llamar a la actualización del proveedor mediante el body del segundo. Se empleo el siguiente path:

Finalmente, para la prueba como tal se hace uso del segundo body y la única restricción o posibilidad de error es el llamado a la función hacia un objeto que no existe.

- ***Requerimiento funcional 5***
  - ***Crear una categoría***

Para el caso de la categoría, es importante resaltar el hecho de que esta por el contrario de Bodega, es independiente de relaciones, por la que no existen objetos directamente arraigados a la misma.

Igualmente, en el body de la prueba se pretende ingresar un objeto que cuenta con los siguientes atributos y estructura:

```
{
  "nombre": "Muebles de Oficina",
  "descripcion": "Mobiliario diseñado específicamente para el uso en entornos de trabajo.",
  "caracteristicas": "Fabricados con materiales resistentes como madera y acero. Requieren montaje y espacio adecuado para su instalación."
}
```

Se ejecuta la prueba de forma normal haciendo uso del path:

```
{{BaseURL}} /categorias/insert
```

En caso de que el ingreso de la categoría sea exitoso se imprime un mensaje a través del controlador. Igualmente, cabe destacar que el posible error que podría ser generado al tratar de insertar una nueva categoría sería bien sea por la estructura que se emplee, o también por la mala definición de los atributos.

Para dicha situación el service lanzaría un error de tipo 500:

```
    public void insertCategoria(Categoria categoria)
    {
        try
        {
            categoriaRepository.insertCategoria(categoria.getNombre(), categoria.getDescripcion(), categoria.getCaracteristicas());
        }
        catch (Exception e)
        {
            throw new ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Error al insertar la categoria");
        }
    }
```

- ***Leer una categoría***

Para la lectura de una categoría en particular lo que se hizo fue hacer el llamado a la función del service mediante el siguiente path:

```
{{BaseURL}} /categorias/1/find
```

Igualmente, cabe destacar que para la elaboración de esta prueba únicamente existe como restricción el mal ingreso del id de la categoría.

Finalmente, a diferencia de las demás pruebas realizadas hasta el momento, esta requirió el siguiente script para la revisión de su funcionamiento:

```
var template = `
<style type="text/css">
  body {
    font-family: Arial, sans-serif;
  }
  .tftable {
    font-size:12px;
    color:#333333;
    width:100%;
    border-width: 1px;
    border-color: #d9d9d9;
    border-collapse: collapse;
    margin-bottom: 20px;
  }
  .tftable th {
    font-size:12px;
    background-color:#f3f3f3;
    padding: 8px;
    border-style: solid;
    border-color: #d9d9d9;
    text-align:left;
    text-transform: none;
    color: #333;
  }
  .tftable tr {
    background-color:#ffffff;
  }
  .tftable td {
    font-size:12px;
    border-width: 1px;
    padding: 8px;
    border-style: solid;
    border-color: #d9d9d9;
    text-align: left;
  }
  .tftable tr:hover {
    background-color:#f5f5f5;
    cursor: pointer;
  }
  .tftable td, .tftable th {
    border-radius: 0;
  }
  h1 {
    color: #333;
```

```
    font-size: 18px;
    text-align: center;
    margin-bottom: 20px;
  }
</style>

<h1>Detalles de la Categoría</h1>
<table class="tftable" border="1">
  <tr>
    <th>id</th>
    <td>{{response.id}}</td>
  </tr>
  <tr>
    <th>nombre</th>
    <td>{{response.nombre}}</td>
  </tr>
  <tr>
    <th>descripción</th>
    <td>{{response.descripcion}}</td>
  </tr>
  <tr>
    <th>características</th>
    <td>{{response.caracteristicas}}</td>
  </tr>
</table>
`;

function constructVisualizerPayload() {
  return {response: pm.response.json()};
}

pm.visualizer.set(template, constructVisualizerPayload());
```

- ***Requerimiento funcional 6***
  - ***Crear un producto***

Dada la longitud de la prueba, en el primer apartado de pruebas de inserción se encuentra la prueba completa para esta primera parte del requerimiento.

  - ***Leer un producto***

Para la lectura de un producto, se hace el llamado a la función a través del siguiente path:

```
{{BaseURL}} /productos/4/find
```

Igualmente, así como el caso anterior, en esta parte el requerimiento es sustancial ingresar de manera correcta el id del producto que se pretende encontrar, pues esta es la única manera en la que la búsqueda podría fallar.

Finalmente, se hizo uso del siguiente script para la revisión del correcto funcionamiento del request.

```
var template = `
<style type="text/css">
  body {
    font-family: Arial, sans-serif;
  }
  .tftable {
    font-size:12px;
    color:#333333;
    width:100%;
    border-width: 1px;
    border-color: #d9d9d9;
    border-collapse: collapse;
    margin-bottom: 20px;
  }
  .tftable th {
    font-size:12px;
    background-color:#f3f3f3;
    border-width: 1px;
    padding: 8px;
    border-style: solid;
    border-color: #d9d9d9;
    text-align:left;
    text-transform: none;
    color: #333;
  }
  .tftable tr {
    background-color:#ffffff;
  }
  .tftable td {
    font-size:12px;
    border-width: 1px;
    padding: 8px;
    border-style: solid;
    border-color: #d9d9d9;
    text-align: left;
  }
  .tftable tr:hover {
```

```
    background-color:#f5f5f5;
    cursor: pointer;
  }
  .tftable td, .tftable th {
    border-radius: 0;
  }
  h1 {
    color: #333;
    font-size: 18px;
    text-align: center;
    margin-bottom: 20px;
  }
</style>

<h1>Detalles del Producto</h1>
<table class="tftable" border="1">
  <tr>
    <th>id</th>
    <td>{{response.id}}</td>
  </tr>
  <tr>
    <th>nombre</th>
    <td>{{response.nombre}}</td>
  </tr>
  <tr>
    <th>costo Bodega</th>
    <td>{{response.costoBodega}}</td>
  </tr>
  <tr>
    <th>precio Unitario</th>
    <td>{{response.precioUnitario}}</td>
  </tr>
  <tr>
    <th>presentación</th>
    <td>{{response.presentacion}}</td>
  </tr>
  <tr>
    <th>cantidad</th>
    <td>{{response.cantidad}}</td>
  </tr>
  <tr>
    <th>unidad Medida</th>
    <td>{{response.unidadMedida}}</td>
  </tr>
  <tr>
    <th>volumen Empaque</th>
    <td>{{response.volumenEmpaque}}</td>
  </tr>
```

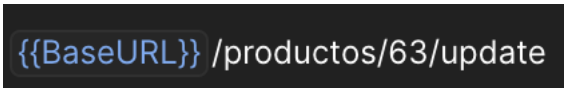
```
  <tr>
    <th>peso Empaque</th>
    <td>{{response.pesoEmpaque}}</td>
  </tr>
  <tr>
    <th>fecha de Vencimiento</th>
    <td>{{response.fechaVencimiento}}</td>
  </tr>
  <tr>
    <th>código de Barras</th>
    <td>{{response.codigoBarras}}</td>
  </tr>
  <tr>
    <th>id Categoría</th>
    <td>{{response.categoria.id}}</td> <!-- Solo mostramos el ID de la categoría -->
  </tr>
</table>
`;

function constructVisualizerPayload() {
  return {response: pm.response.json()};
}

pm.visualizer.set(template, constructVisualizerPayload());
```

- ***Actualizar un producto***

Para la correcta actualización de cualquier producto es sumamente importante tener en cuenta el path implementado, así como ciertas restricciones importantes.



A continuación, se presentarán una serie de restricciones o ejemplos de error en los cuales la prueba falla. Para dicho propósito, se enseñará un producto correcto de ejemplo que será modificado:

```
{
    "nombre": "Galletas de Avena",
    "costoBodega": 3500.0,
    "precioUnitario": 5500.0,
    "presentacion": "Paquete de 12 galletas.",
    "cantidad": 500,
    "unidadMedida": "gr",
    "volumenEmpaque": 50.0,
    "pesoEmpaque": 300.0,
    "fechaVencimiento": "2025-12-31",
    "codigoBarras": "z8z8z8z8z8",
    "categoria": {
        "id": 1
    }
}
```


1. Actualización de un producto cuyo id es inexistente.

```
{{BaseURL}}/productos/8045486/update
```

2. Actualización de un producto con categoría incorrecta.

```
{
    "nombre": "Galletas de Avena",
    "costoBodega": 3500.0,
    "precioUnitario": 5500.0,
    "presentacion": "Paquete de 12 galletas.",
    "cantidad": 500,
    "unidadMedida": "gr",
    "volumenEmpaque": 50.0,
    "pesoEmpaque": 300.0,
    "fechaVencimiento": "2025-12-31",
    "categoria": {
        "id": 2
    }
}
```


3. Actualización de un producto con fecha de vencimiento incorrecta.


```
{
    "nombre": "Galletas de Avena",
    "costoBodega": 3500.0,
    "precioUnitario": 5500.0,
    "presentacion": "Paquete de 12 galletas.",
    "cantidad": 500,
    "unidadMedida": "gr",
    "volumenEmpaque": 50.0,
    "pesoEmpaque": 300.0,
    "fechaVencimiento": "2023-12-31",
    "categoria": {
        "id": 1
    }
}
```


- ***Requerimiento funcional 7***
    - ***Creación de una orden de compra***

La presente clase o entidad como su nombre lo indica, refiere a la creación de las órdenes de compra que son realizadas por una sucursal y que se relaciona con una serie de productos.

A continuación, se hará ejemplificación de una orden de compra de correcta sintaxis y estructura:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-17",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```

Si se cumple exitosamente la petición, entonces a través del controlador y el servicio se imprimirá un mensaje de correcta inserción. Sin embargo, es bastante importante

tener en cuenta una serie de restricciones que podría causar una falla en la creación de una orden de compra. Estos se mencionarán y se ejemplificarán:

5. La fecha de entrega de la orden de compra no puede ser anterior a la fecha actual. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-01-10",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```

6. La bodega de la orden de compra debe pertenecer a la sucursal correspondiente. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-10",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 3
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```

7. El proveedor correspondiente debe suministrar productos a la sucursal correspondiente. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-10",
        "proveedor": {
            "id": 4374573657
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": [
        {
            "idProducto": 1,
            "cantidad": 100,
            "precio": 9000.00
        },
        {
            "idProducto": 2,
            "cantidad": 50,
            "precio": 5000.00
        }
    ]
}
```


8. La orden de compra necesita solicitar un producto al menos. Un ejemplo de error se muestra a continuación:

```
{
    "ordenCompra": {
        "fechaEntrega": "2024-10-10",
        "proveedor": {
            "id": 1
        },
        "sucursal": {
            "id": 2
        },
        "bodega": {
            "id": 2
        }
    },
    "productosOrden": []
}
```

Aquellas restricciones mencionadas previamente hacen parte de la generalidad de la creación de la orden de compra, no obstante, ya que esta acción comprende un proceso de transacción, se deben de tener en cuenta las relaciones que resultan sustanciales. Estas mismas se enunciarán y enumerarán en la parte inferior.

Sobre los productos;

7. El proveedor tiene que suministrar el producto solicitado.

8. La cantidad solicitada del producto tiene que ser mayor a 0. No tiene sentido iniciar una orden de compra sin ningún producto para ofertar.

9. El precio del producto debe ser mayor o igual al precio unitario.

Sobre la bodega;

10. La bodega debe tener menos del 70% para de ese modo poder agregar más productos.

Sobre la relación entre el producto y la bodega;

11. El producto tiene que estar registrado en la bodega.

12. La cantidad del producto solicitado no puede exceder la capacidad disponible en la bodega correspondiente.

Cualquier incumplimiento de las anteriores restricciones provocarán la generación del error BAD REQUEST, lanzado por el service a través del controlador.

- ***Requerimiento funcional 8***
  - ***Actualizar orden de compra cambiando su estado***

Para este requerimiento, únicamente se tiene que llamar al siguiente path de actualización para ejecutar la acción deseada:

```
{{BaseURL}} /ordenescompra/188/update
```

Cabe destacar que, para el propósito descrito, se tiene que enviar un body con la información que quiere ser cambiada, en este caso, sería el estado. A continuación, se presenta el body que sería el adecuado:

```
{
    "estado": "Anulada"
}
```

- ***Requerimiento funcional 9***
  - ***Mostrar todas las órdenes de compra***

Para este requerimiento solamente se llama al path correspondiente para poder realizar la acción deseada.

```
{{BaseURL}} /ordenescompra
```

Para este requerimiento, no se requiere ningún body, eso sí, se hizo uso del siguiente script para revisar el correcto funcionamiento del request:

```
var template = `
<style type="text/css">
  body {
    font-family: Arial, sans-serif;
  }
```

```
  .tftable {
    font-size:12px;
    color:#333333;
    width:100%;
    border-width: 1px;
    border-color: #d9d9d9;
    border-collapse: collapse;
    margin-bottom: 20px;
  }
  .tftable th {
    font-size:12px;
    background-color:#f3f3f3;
    padding: 8px;
    border-style: solid;
    border-color: #d9d9d9;
    text-align:left;
    text-transform: none;
    color: #333;
  }
  .tftable tr {
    background-color:#ffffff;
  }
  .tftable td {
    font-size:12px;
    border-width: 1px;
    padding: 8px;
    border-style: solid;
    border-color: #d9d9d9;
    text-align: left;
  }
  .tftable tr:hover {
    background-color:#f5f5f5;
    cursor: pointer;
  }
  .tftable td, .tftable th {
    border-radius: 0;
  }
  h1 {
    color: #333;
    font-size: 18px;
    text-align: center;
    margin-bottom: 20px;
  }
</style>

<h1>Listado de órdenes de compra</h1>
<table class="tftable" border="1">
  <tr>
```

```
    <th>id</th>
    <th>fecha Creación</th>
    <th>fecha Entrega</th>
    <th>fecha Recepción</th>
    <th>estado</th>
    <th>id Proveedor</th>
    <th>id Sucursal</th>
    <th>id Bodega</th>
  </tr>

  {{#each response}}
    <tr>
      <td>{{id}}</td>
      <td>{{fechaCreacion}}</td>
      <td>{{fechaEntrega}}</td>
      <td>{{fechaRecepcion}}</td>
      <td>{{estado}}</td>
      <td>{{proveedor.id}}</td>
      <td>{{sucursal.id}}</td>
      <td>{{bodega.id}}</td>
    </tr>
  {{/each}}

</table>
`;

function constructVisualizerPayload() {
  return {response: pm.response.json()};
}

pm.visualizer.set(template, constructVisualizerPayload());
```