Proyecto Final de Master en JAVA

Desarrollo de aplicación Web
CICE Escuela de nuevas tecnologías.

Alvaro Fernando Leiva Estrada
02/03/2021

INDICE

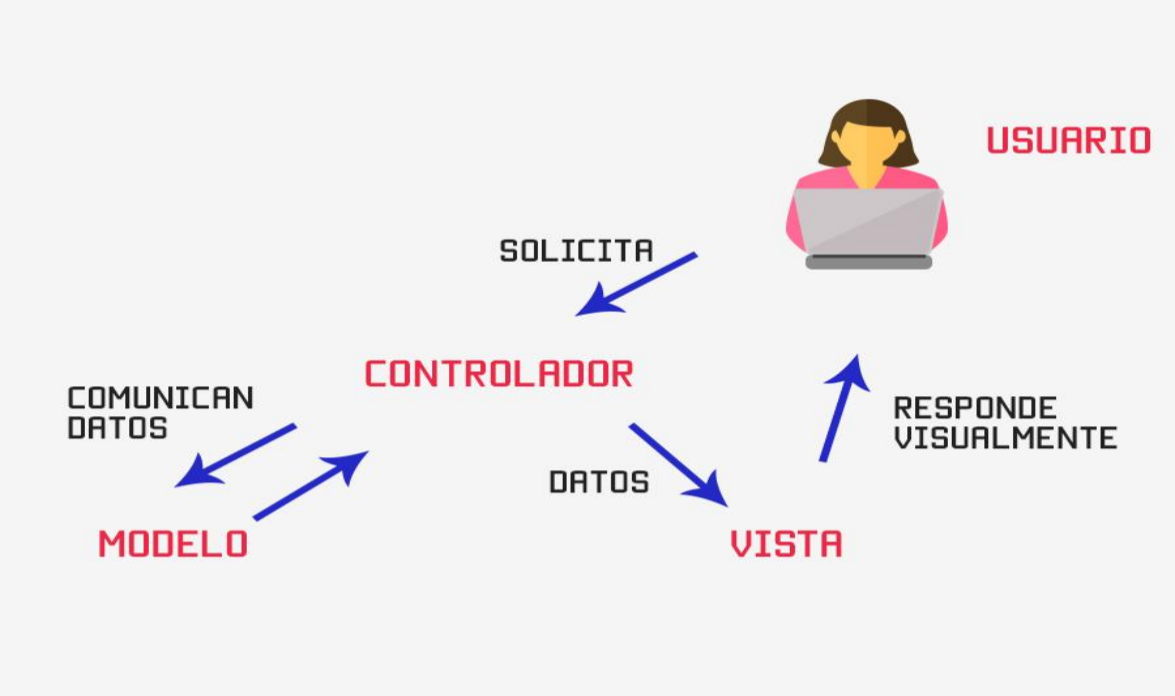
Introducción:

El proyecto final, esta basado en el desarrollo de una aplicación web para la gestión integral del sistema de almacén y ventas de una compañía. Para el cual se ha utilizado distintas herramientas, tanto en la parte del fronted como backend.
Se decidió para la parte del backend, utilizar java como lenguaje en la parte del servidor, al igual que se utilizo Spring Boot como framework o herramienta de apoyo al lenguaje JAVA, como gestor de base de datos de la aplicación se utilizo Mysql.
Por otra parte, del lado del cliente (fronted) se utilizo herramientas como BOOTSRAP, HTML y CSS, las cuales le han colaborado a darle un tomo mas amigable al sitio.

Arquitectura del Sistema Software:

Para la realización del proyecto, este se baso en la arquitectura MVC (Modelo,Vista,Controlador) esta arquitectura separa la lógica la de aplicación de la lógica de la vista en una aplicación.

El usuario envía la petición al controlador, este a su vez invoca al modelo, el cual tiene la tarea de hacer la petición de datos al gestor de BBDD que estemos utilizando, mediante sentencias (queries). El modelo luego retorna los datos suministrados por la BBDD al controlador, para que este proceda a enviar la información solicitado por el usuario a la vista, el cual es otro componente que forma esta arquitectura, una vez los datos en la vista esta renderiza las vistas al navegador, para que sean visibles al usuario.
En la siguiente imagen se puede visualizar el proceso de una petición de datos a una aplicación web.

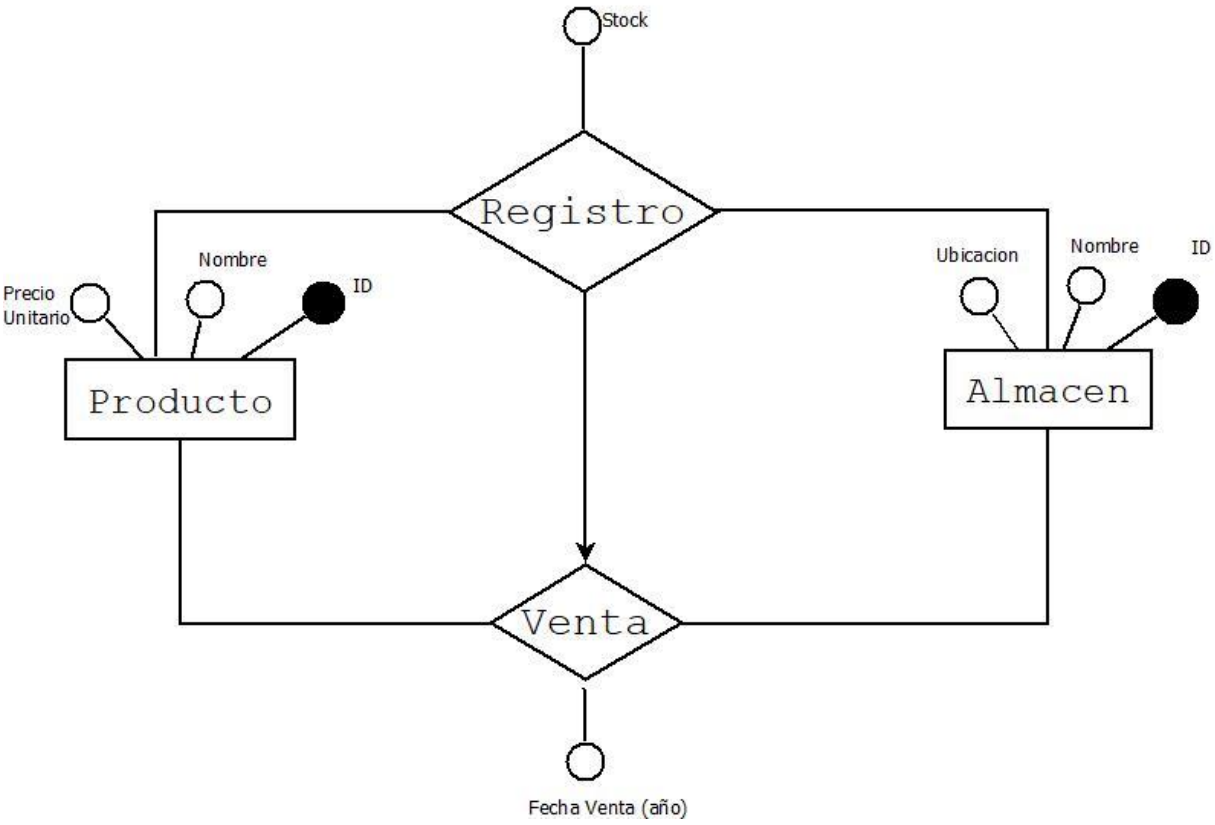


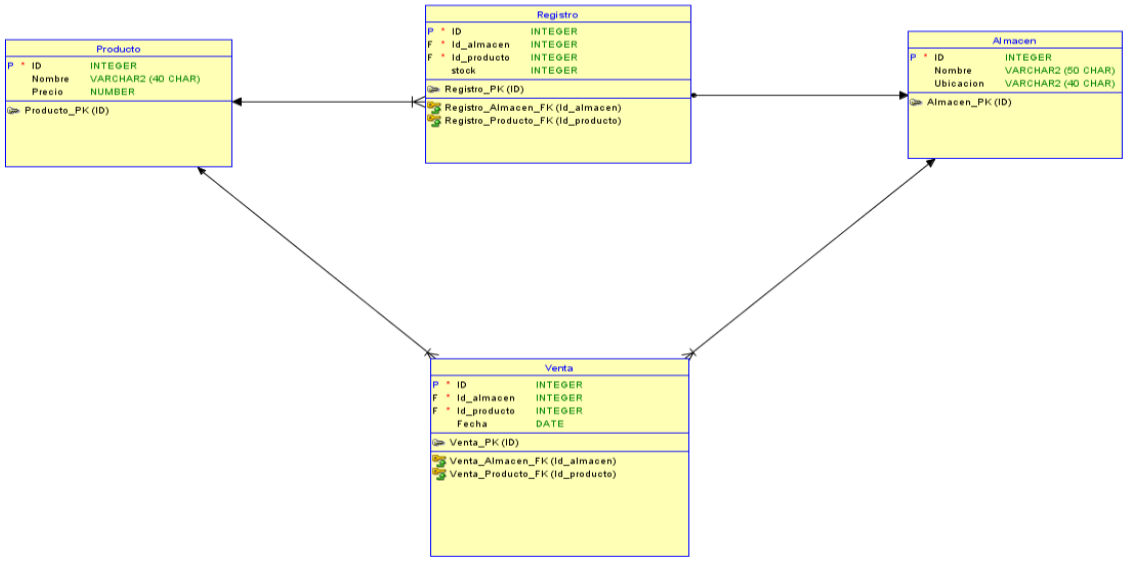
Desarrollo de aplicación para la gestión del sistema de almacén.

En la siguiente imagen se describe el diseño modelo entidad relación del sistema integral de almacén. Como bien indica el modelo, el cual es relativamente simple, al contener dos entidades, nos indica que entre estas dos relaciones existe una relación de Restricción de inclusión, es decir que por medio de este tipo de relación, el diseño se asegura que para un producto este en venta, este debe haber sido registrado en un almacén de no ser así, la venta del producto no podrá ser posible, ya que no se puede vender un producto del cual no se ha registrado.

Del mismo modo se hace referencia al siguiente escenario, en donde el producto se ha registrado, pero su venta es imposible debido al numero reducido de stock, inventariados por el almacén de esta manera, es conveniente tener un seguimiento de la cantidad de productos en inventario para cada almacén, por medio de un campo que llamaremos STOCK, el cual contendrá el numero exacto de productos en inventario.

Por otra parte, es deseable llevar el control de la fecha que se realiza cada venta, con lo cual a la relación venta se ha agregado un campo el cual, utilizamos para tener control de la fecha de venta de un producto.





Spring Data:

Para le creación de las tablas en la base de datos, se utilizará el modulo de Spring Data JPA, el cual nos facilitará la tarea de mappear pojos de Java con las necesarias anotaciones para crear las tablas necesarias para nuestro proyecto.

```
# CONFIGURACION DE SPRING DATA JPA
spring.jpa.generate-ddl=true
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create-drop
```

En la imagen de arriba, vemos la configuración necesaria para la creación de las tablas partiendo de clases de java.
En la primera línea de configuración encontramos el comando que genera a partir de un script insertara todos los registros en la base de datos.
Especificamos el tipo de dialecto que utilizaremos en Mysql.
La tercera línea indica, si, deseamos ver en la consola las sentencias de SQL que se ejecutaran, para el cual asignamos el valor de verdadero TRUE.
La cuarta línea indica la creación de las tablas al ejecutar el programa, y el droppeo de las mismas al finalizar.

Generador de registros:

Como se solicita en el enunciado, se generarán distintos registros que luego serán, insertados en la base de datos.
Se solícita crear 10 almacenes, 100 productos y 1000 ventas.
Se utilizo el lenguaje de programación Python, para crear un script el cual generará todos los registros que luego serán usados para nuestra aplicación. En la imagen se muestra el script de creación de registros utilizado.

```python
import random

def productoGenerador(file):
    for x in range(1,100):

        producto=random.randint(1,100)
        nombre="'PRODUCTO_"+str(producto)+"'"
        precio = random.randint(1,100)
        line='insert into producto (nombre, precio_unitario) VALUES({},{});\n'.format(nombre,precio)
        file.write(line)

def ventasGenerador(file):
    for venta in range(1,1000):
        almacen=random.randint(1,10)
        producto=random.randint(1,100)
        year= random.randint(2015,2020)
        line='insert into ventas(almacen_id,ano_venta,producto_id) values ({},{},{});\n'.format(almacen,year,producto)
        file.write(line)


if __name__ == '__main__':
    #venta = open("ventas.txt", 'w')
    producto = open("productos.txt",'w')
    #ventasGenerador(venta);
    productoGenerador(producto)
    #venta.close();
    producto.close();
```

Web Service RESTful:

Como se solicita en el enunciado, la aplicación web proporciona cuatro distintos servicios, para todos aquellos que deseen consumir el API de la aplicación web del almacén.
Para verificar el correcto funcionamiento del API, se utilizo Postman una herramienta que se utiliza para el testing de API REST.

El primer **endpoint** al cual podemos acceder, es el de ventas totales el cual proporciona la cantidad exacta de ventas realizadas, es decir el total de productos vendidos por todos los almacenes y además proporciona el ingreso total en euros. La respuesta que obtendremos del API viene en formato JSON. La dirección URL es la siguiente: Localhost:8030/API/totalVentas

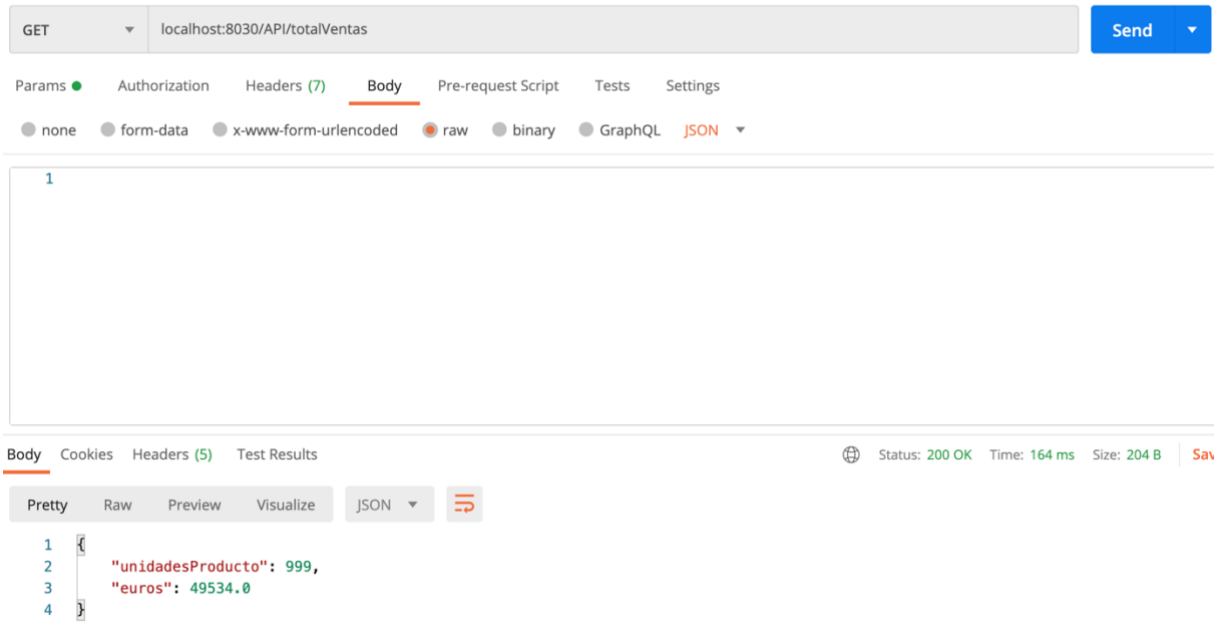
El segundo endpoint al cual podemos acceder, detalla la cantidad de ventas por almacén realizadas ordenado por facturación. La dirección URL es la siguiente:
Localhost:8030/API/ventasPorAlmacen

El tercer endpoint al cual podemos acceder, detalla las ventas realizadas por año. La dirección URL es la siguiente: localhost:8030/API/ventasPorFecha

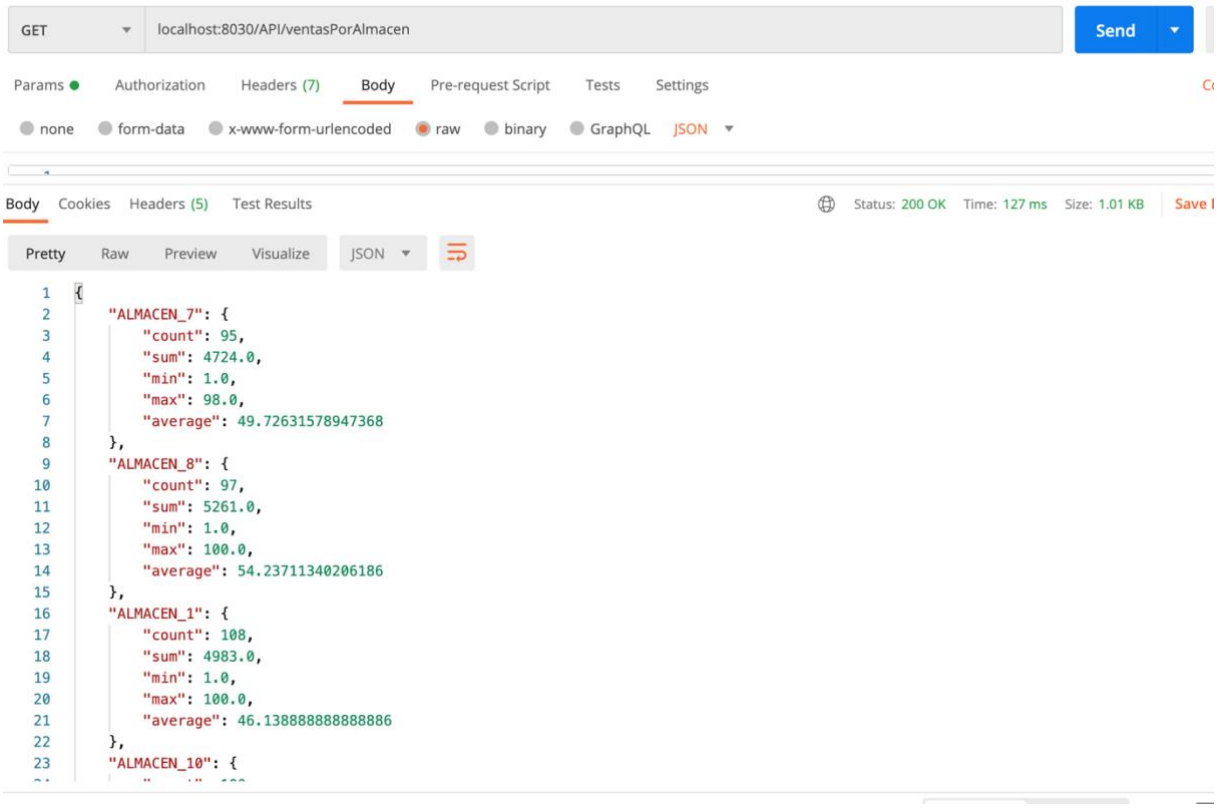
El cuarto endpoint al cual podemos acceder, detalla los 10 producto mas vendidos:
Localhost:8030/API/productosMasVendidos

Como se menciona anteriormente, Postman se utiliza en esta sección como herramienta de testing de la API, para verificar su correcto funcionamiento.

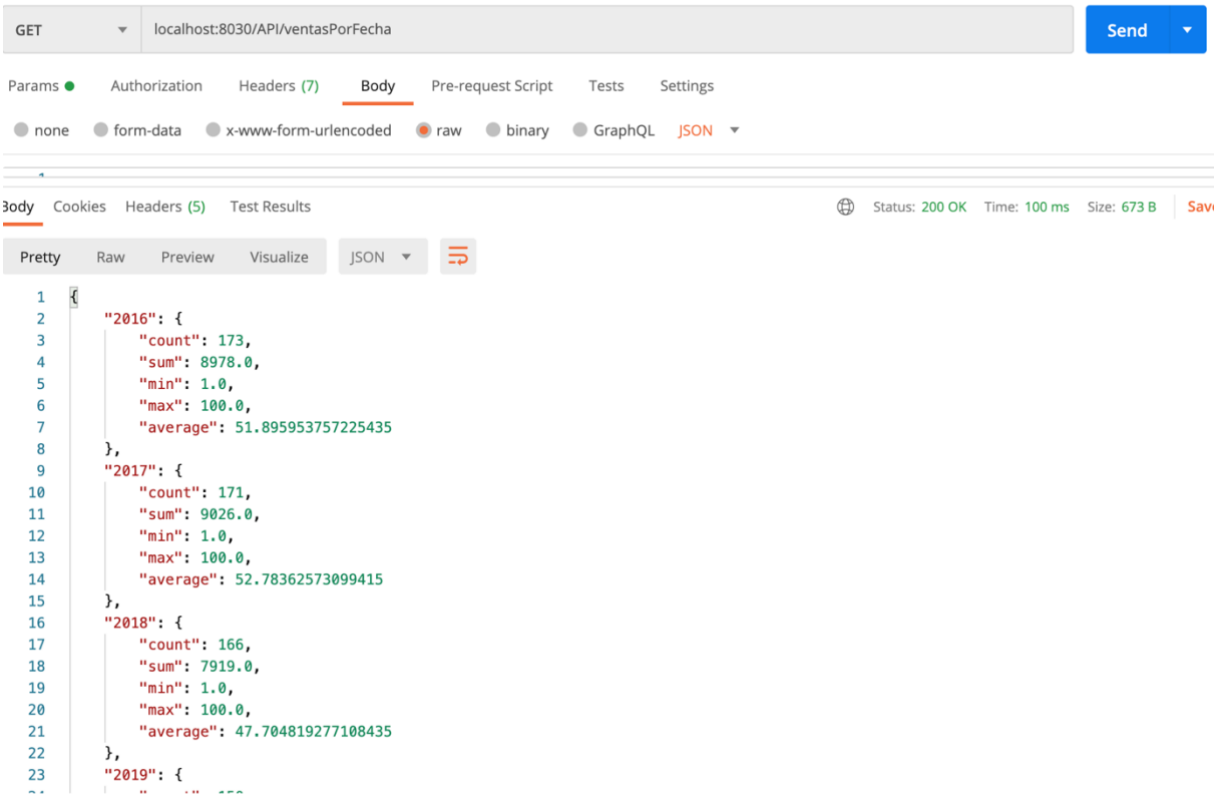
En la imagen de abajo, hacemos una petición a la API almacenes, para conocer el total de ventas realizadas por todos los almacenes



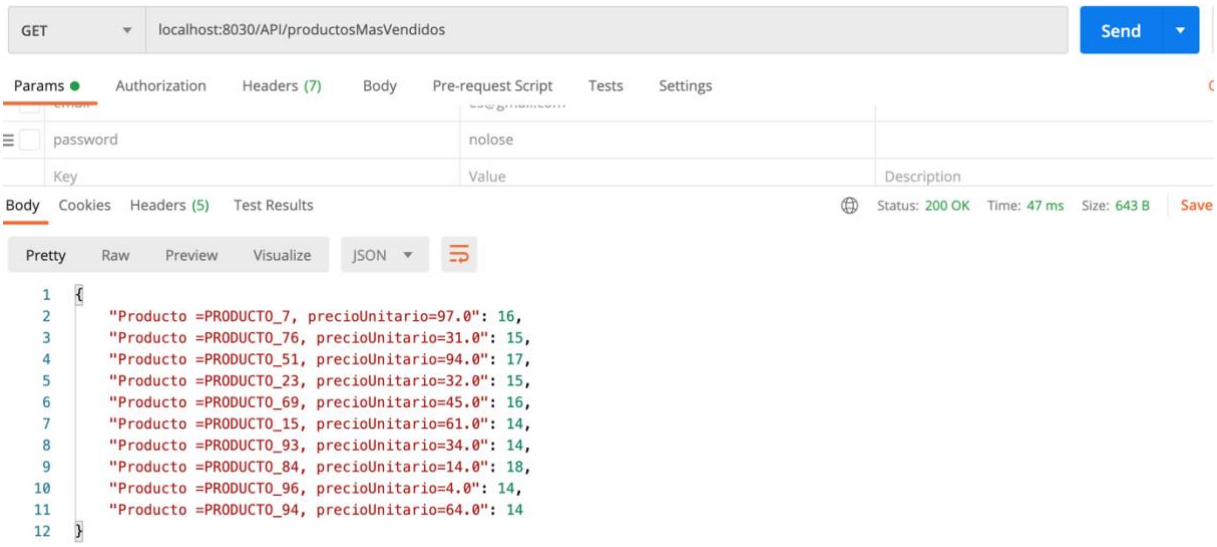
En la imagen de abajo, hacemos una petición a la API almacenes, para conocer el total de ventas por almacén.



En la imagen de abajo, hacemos una petición a la API almacenes, con el fin de conocer el total de ventas por año de todos los almacenes.



En la imagen de abajo, hacemos una petición a la API almacenes, con el fin de conocer los 10 productos mas vendidos.



Manual de Usuario:

Como gestor de base de datos se utilizo MySQL, por lo tanto, es necesario haber creado un schema con el nombre de “pfmDB”

```
# CONFIGURACION MYSQL DATASOURCE (MySQL 8.0)
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost/pfmDB?useSSL=false&serverTimezone=UTC
spring.datasource.username= ?
spring.datasource.password= ?
```

Seguidamente como se muestra en la imagen, se deberá remplazar el signo “?” en el campo “username” por el nombre de la base de datos, comúnmente este suele ser “root”.
Luego remplazamos el campo “password” por la contraseña que utilicé para ingresar a su workbench de Mysql.
De esta manera tendrá ya configurado la BBDD para el proyecto.

Seguidamente, se podrá vía el puerto 8030, ingresar a la aplicación web la cual tendrá una vista igual a la imagen que se muestra abajo.

Como se observa en la imagen, la aplicación vendrá equipada con el funcionamiento requerido. Como pantalla principal se despliega los distintos almacenes con su ubicación.

Además de esto contara con acciones como “productos” el cual, al pulsarlo, desplegara todos los productos que se han registrados previamente en dicho almacén. Al desplegarse la pantalla de los productos disponibles en el almacén tendremos un botón de “compra “ para realizar una venta al cliente.

HOME **LISTADO DE PRODUCTOS**

#	Nombre	Precio Unitario	Numero de almacen	Comprar Producto
30	PRODUCTO_96	4.0	1	Comprar
50	PRODUCTO_91	31.0	1	Comprar
32	PRODUCTO_58	63.0	1	Comprar
52	PRODUCTO_93	34.0	1	Comprar
31	PRODUCTO_38	22.0	1	Comprar
51	PRODUCTO_7	1.0	1	Comprar
33	PRODUCTO_84	37.0	1	Comprar
53	PRODUCTO_58	58.0	1	Comprar
34	PRODUCTO_51	75.0	1	Comprar
54	PRODUCTO_88	34.0	1	Comprar

El botón de “Ventas realizadas” desplegara todas las ventas realizadas en ese almacén. En esta pantalla dispondremos de un buscador por año fecha de ventas realizadas.

VENTAS REALIZADAS

HOME 0 Buscador

#	Nombre de almacen	Nombre de producto	Año de venta
9	ALMACEN_1	PRODUCTO_42	2020
12	ALMACEN_1	PRODUCTO_99	2019
16	ALMACEN_1	PRODUCTO_88	2020
21	ALMACEN_1	PRODUCTO_76	2019
24	ALMACEN_1	PRODUCTO_63	2017
28	ALMACEN_1	PRODUCTO_96	2020
47	ALMACEN_1	PRODUCTO_90	2019
69	ALMACEN_1	PRODUCTO_100	2017
77	ALMACEN_1	PRODUCTO_62	2016
78	ALMACEN_1	PRODUCTO_91	2018
111	ALMACEN_1	PRODUCTO_89	2015

Por ultimo se dispone de un botón “Adquirir” el cual listara todos los productos disponibles para agregar al almacén ya sea si para incrementar el producto o bien registrar un nuevo producto.

Como se solicita en las especificaciones del enunciado, al momento de seleccionar la opción de rotura en stock, se despliega todos los almacenes con la cantidad de productos que se encuentran por debajo de las 5 unidades.

Para cada almacén con productos por debajo del umbral mínimo, se desea conocer cuales son los productos, por la cual se habilito la opción de mostrar todos los productos, presionando el botón de “mostrar producto”.

Productos en rotura

HOME

Producto	Cantidad
PRODUCTO_84	2
PRODUCTO_98	4
PRODUCTO_89	2
PRODUCTO_79	2
PRODUCTO_87	2
PRODUCTO_31	2
PRODUCTO_64	2
PRODUCTO_84	2
PRODUCTO_11	2
PRODUCTO_12	2
PRODUCTO_10	2

Conclusión
Conocimientos Adquiridos:

JAVA 8:

Para la realización de varias de las consultas contra la base de datos, se introdujeron varios de los conceptos aprendidos y nuevas funcionalidades que traía la versión 8 de java. Entre ellas la introducción de lambdas, las cuales han sido de beneficio, por su forma compacta y menos verbosa de hacer las cosas.

```
public ProductoAlmacen obtenerProductoAlmacen(Long idAlmacen,Long idProducto) {

    List<ProductoAlmacen> lista = productoAlmacenRepostory
            .findAll()
            .stream()
            .filter(e -> e.getIdAlmacen().getId() == idAlmacen && e.getIdProducto().getId() == idProducto)
            .collect(Collectors.toList());
    ProductoAlmacen productoAlmacen = (lista.isEmpty())?null:lista.get(0);
    return productoAlmacen;
}
```

En la imagen de arriba, se aprecia el uso de la librería stream, que de igual forma fue introducida en la versión 8 de java. De esta manera Java agrega un nuevo paradigma de programación, el paradigma funcional todo esto a partir de la versión 8.

En esta imagen se demuestra el beneficio que se obtiene al trabajar con la versión 8 de Java. El método devuelve los 10 productos mas vendidos, de forma elegante y legible se logra extraer la información que se solicita de la BBDD.

```
@GetMapping(value = "/productosMasVendidos")
public ResponseEntity<Map<Producto,Long>> productosMasVendidos(){
    Map<Producto, Long> productos = ventaServiceImp
            .listar()
            .stream()
            .collect(Collectors.groupingBy(Venta::getProducto,Collectors.counting()))
            .entrySet()
            .stream()
            .sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()))
            .limit(maxProductos)
            .collect(Collectors.toMap(Map.Entry :: getKey,Map.Entry::getValue));

    return ResponseEntity.ok(productos);
}
```