
**INSTITUTO POLITÉCNICO NACIONAL**

**UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS**

**ACADEMIA DE TELEMÁTICA**


**BASES DE DATOS DISTRIBUÍDAS – 3TV2**

# *Proyecto Final*
# *AdventureWords*


Acosta Villa Cristian Abraham
Arvizu González Abraham
Enriquez Melendez Jesús
García Acosta Abraham
Monzon Mogollán Daniela
Rosas Palacios Alan


13 de diciembre de 2021
Ciudad de México

# Planes de Ejecución

Los esquemas con los que trabaja nuestro equipo están alojados en 2 instancias de AWS, nuestra API (programada en TypeScript) esta alojada localmente y esta consume datos de las instancias y los mapea utilizando un ORM, todo esto gracias a un paquete llamado TypeORM.

Las consultas son implementadas con este modelo ORM para crear la API y los resultados de cada una de ellas son mostrados con ayuda de una aplicación llamada POSTMAN (esta app nos permite consumir los datos de un api), finalmente los datos son mostraos en formato json.

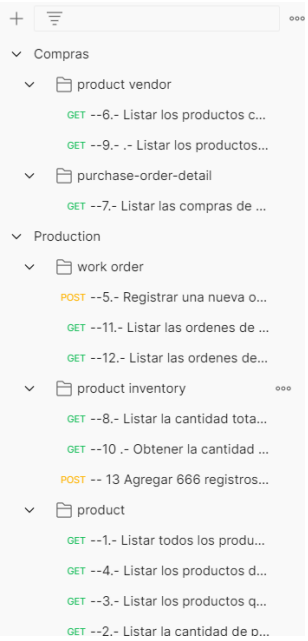
Cabe aclarar que nuestra aplicación se ejecuta localmente para ser utilizada a través de peticiones HTTP, puesta en el puerto 3000, así de esta manera podemos acceder al resultado o en su defecto inserción de datos.

En el caso de las operaciones de lectura/escritura, gracias a POSTMAN podemos simular un escenario de tipo formulario (durante el desarrollo se podra observar mejor esto).

Ejecución de las consultas:

Ahora mostraremos como se obtiene el resultado de las consultas través del servicio web POSTMAT.

Primero se mostrará el enlace del servicio http que se le envía a POSTMAN y posteriormente su ejecución.



1.- Listar todos los productos que tengan un precio de venta mayor o igual a "n" dólares.

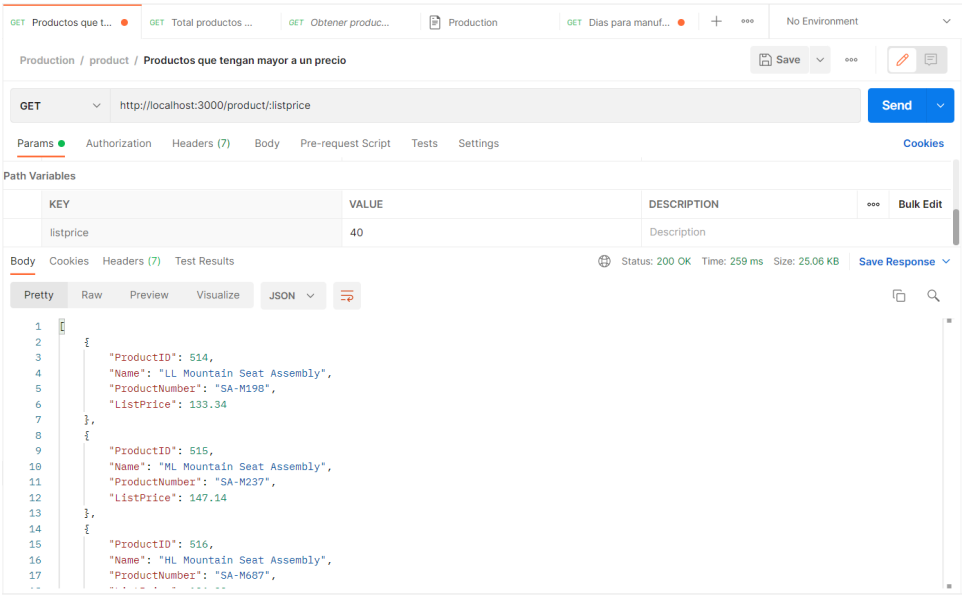
Originalmente la consulta era estática, es decir tenía el mismo propósito, pero solo se podían mostrar los productos con un precio de venta mayor o igual a 40.

Con esta nueva implementación se crea dinamismo con la variable “Precio de lista”.

La sintaxis se comopone como una URL, el host donde se encuentra la aplicacion es en el dispositivo local, por eso se pone localhost, en el puerto 3000. La ruta define el metodo que sera ejecutado en nuestra API, en esta caso, actua sobre la tabla product y el parametro dinamico de esta consulta se especifica con el siguiente slash. Entonces, para recuperar los datos de esta operacion tenemos que:

***localhost:3000/product/n***

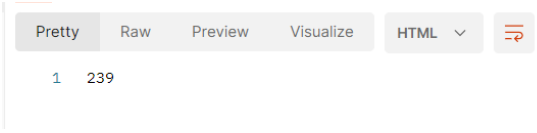
Donde n es el valor de venta mayores o iguales de los productos que queremos recuperar




2.- Listar la cantidad de productos que se venden para ensamblar en casa

En el caso de esta consulta, no se requiere enviar parametros y por ende el formato para recuperar los datos de esta consulta solo especifica el nombre de la tabla. En la programacion puede verse que el metodo para implementar esta consulta no tiene entradas. Entonces para recuperar los datos de esta operacion tenemos que:

***localhost:3000/product***



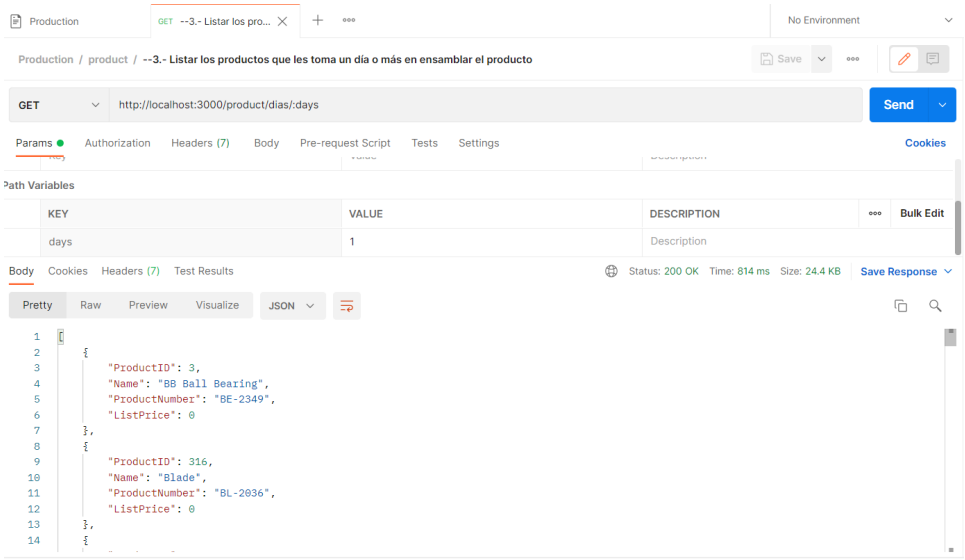
3.- Listar los productos que les toma un día o más en ensamblar el producto

Para esta operacion de consulta despues de especificar la ruta de la tabla product, se especifica que vamos a trabajar sobre el atributo de dias (haciendo referencia a la consulta 3) para posteriormente especificar un numero de dias n.

Entonces para recuperar los datos de esta operacion tenemos que:

***localhost:3000/product/dias/n***

Donde “n” son los dias que se toman para ensamblar un producto.



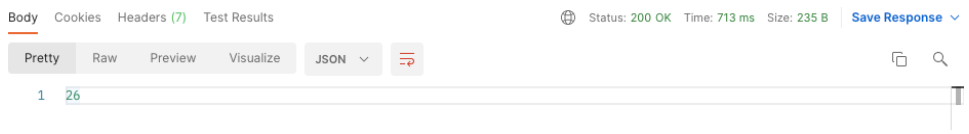

4.- Listar la productos de cierto color

Esta consulta como la primera tambien se adapto para que tuviera dinamismo.

Al igual que el caso anterior se especifica que vamos a trabajar sobre el atributo de product y despues sobre el atributo de color; La siguiente ruta a especificar de la misma forma se refiere a un parametro, que debe ser valido (Los podemos consultar en el archivo procut.type.ts) para entregar el dato correctamente. Entonces para recuperar el dato de esta operacion tenemos que:

***localhost:3000/product/color/nom***

Donde “nom” se refiere al nombre del color (en ingles).

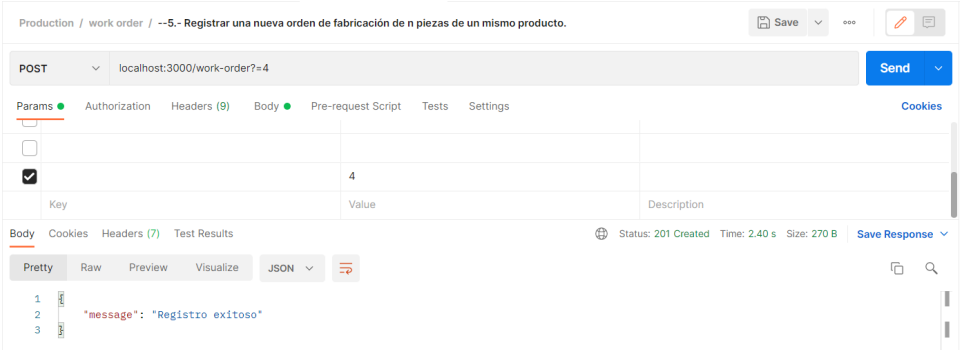



5.- Registrar una nueva orden de fabricación de n piezas de un mismo producto.

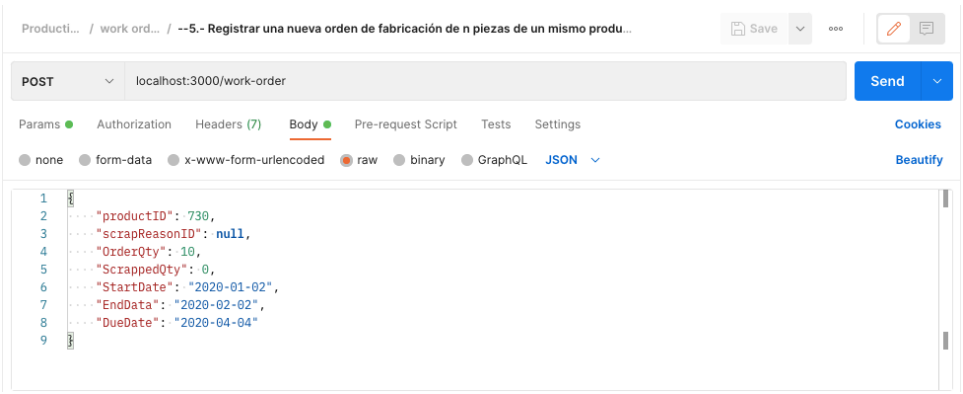
Esta operacion consta de 2 partes, primero la ruta que define este metodo, esta dada como:

***localhost:3000/work-order***

Ahora bien, desde alguna aplicacion montada en un servidor que consuma los servicios de esta API, se crea un formulario para poder insertar los datos solicitados, la misama API cuida la integridad referencial y los errores que pudieran llegar a pasar al momento de insertar.




Para el caso de recrear el formulario, POSTMAN nos ayuda visualmente para editar los campos necesarios que se ven involucrados en esta operacion INSERT (POST).

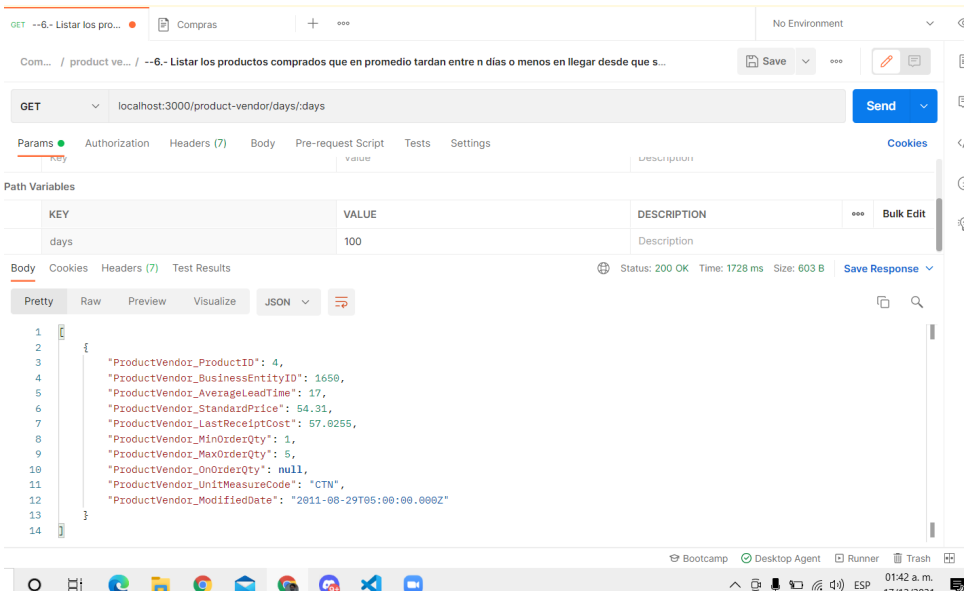



6.- Listar los productos comprados por AdventureWorks que en promedio tardan entre n días o menos en llegar desde que se realizó el pedido de compra.

La ruta que define a este metodo se estructura como:

***localhost:3000//product-vendor/days/n***

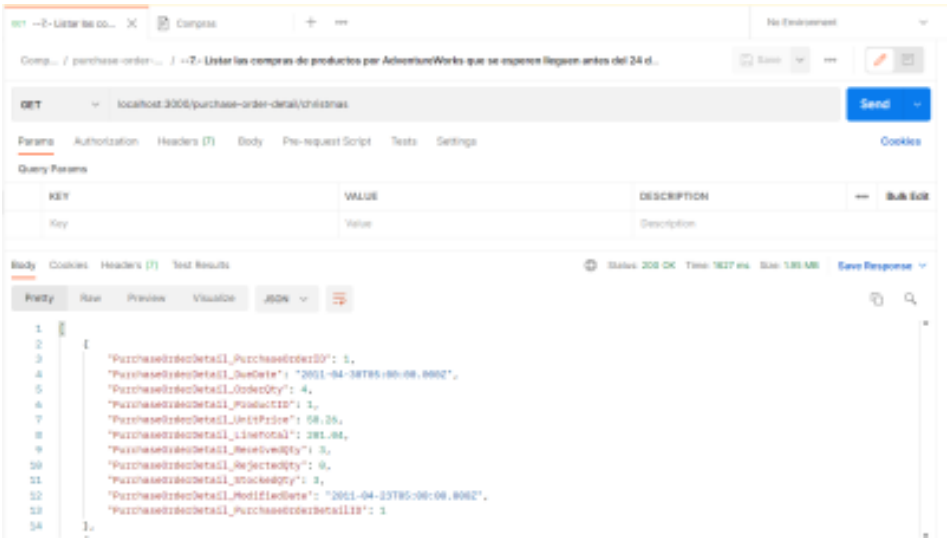
Donde “n" es el numero de dias o menos en llegar desde que se realizo el pedido de compra.




7.- Listar las compras de productos por AdventureWorks que se esperen lleguen antes del 24 de diciembre del 2014;

Debido a la dificultad que pudo haber generado ORM con el formato de las fechas (como condicional) esta colsulta se definio de manera estatica, esto quiere decir que la ruta que se especifica manda a llamar un metodo que realiza la operacion. Entonces tenemos que:

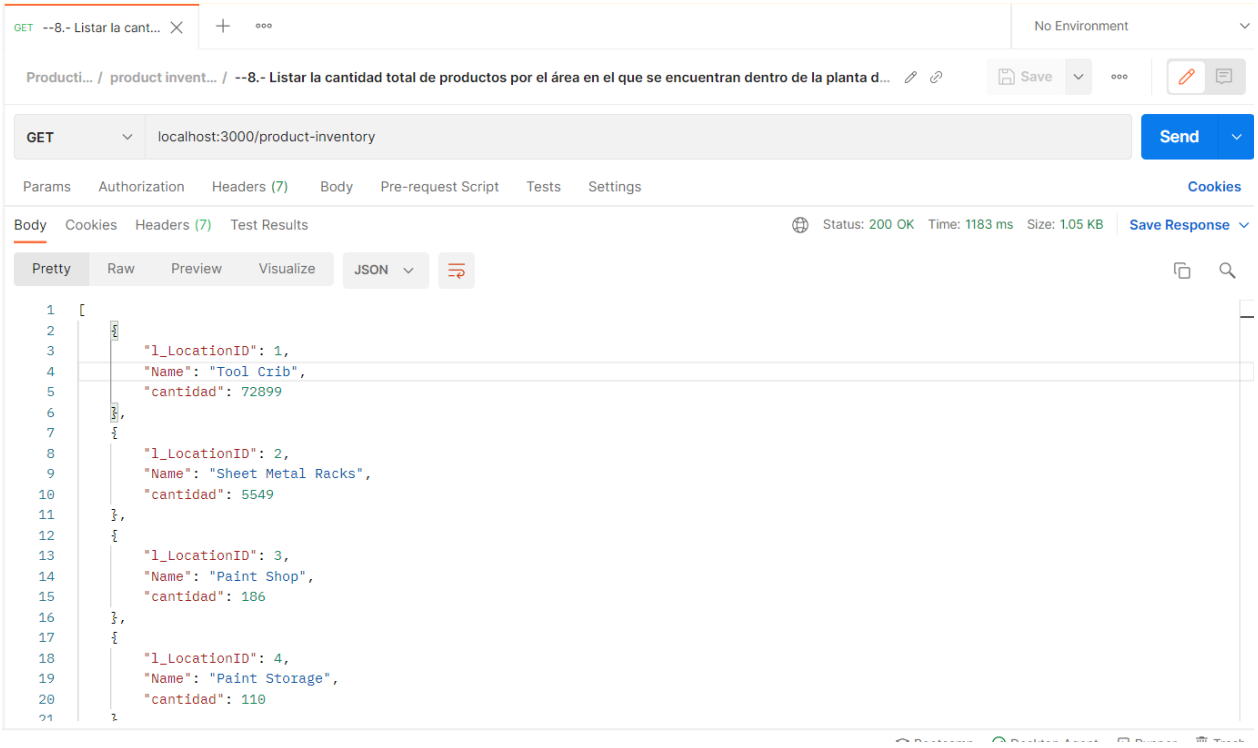
***localhost:3000//purchase-order-detail/christmas***



8.- Listar la cantidad total de productos por el área en el que se encuentran dentro de la planta de producción.

Este si es un metodo estatico sin necesidad de implementar dinamismo, pues, utiliza una categorizacion de todos los datos de la tabla. La tabla en cuestio es product inventory y para llevar a cabo esta consulta se tiene la siguiente dirección:

***localhost:3000//product-inventory***

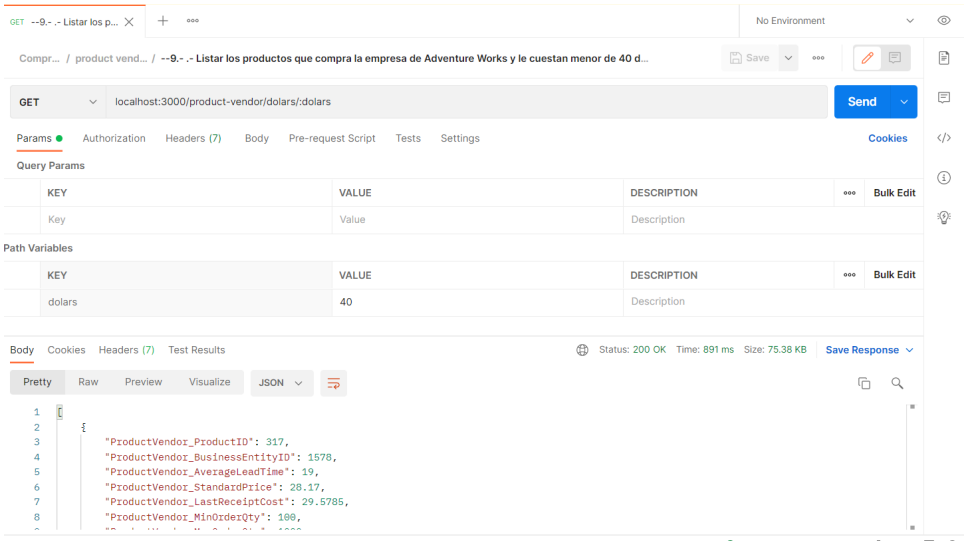


9.- Listar los productos que compra la empresa de Adventure Works y le cuestan menor de n dolares.

Al igual que algunas de las de las operaciones de lectura descritas en este documento, este metodo se define a traves de una ruta que acepta un parametro que le da dinamismo a la consulta en general. En este caso tenemos lo siguiente:

***localhost:3000//product-vendor/dolars/n***

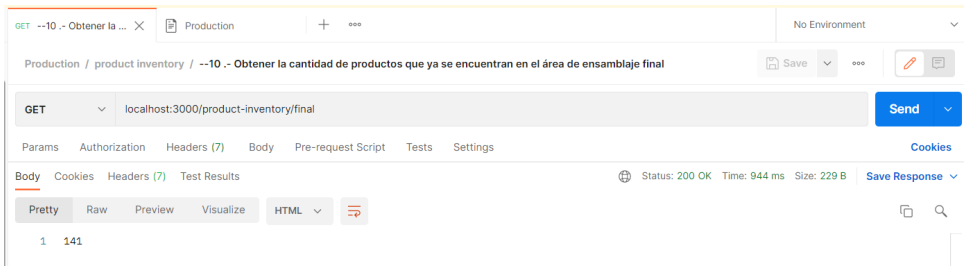
Donde n es la cantidad de dolares que le cuesta la pieza o producto a AdventureWorks




10 .- Obtener la cantidad de productos que ya se encuentran en el área de ensamblaje final.

Al especificar el area de ensamblaje final como parametro pensando que el usuario no tiene conocimiento de otras areas o no tiene permiso de conocerlo, este metodo se ejecuta solo especificando la direccion:

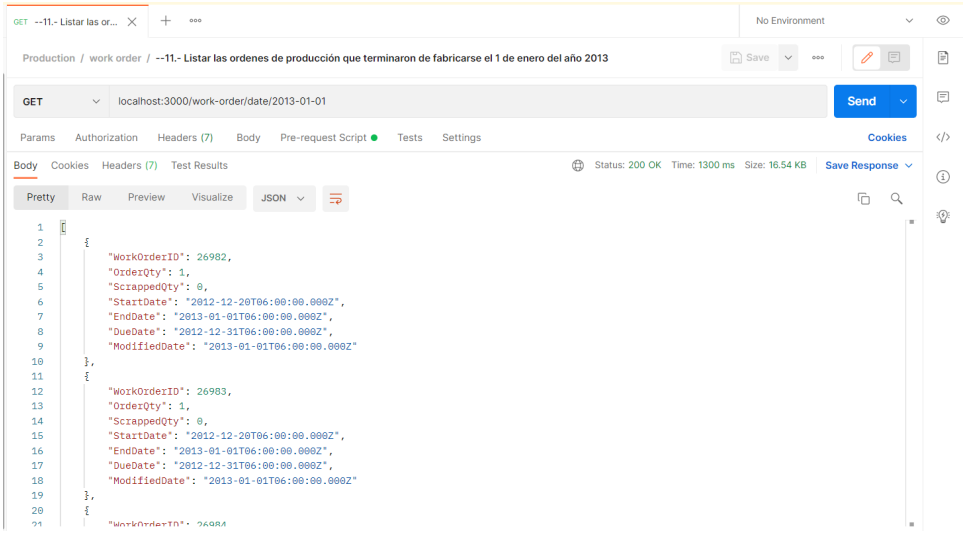
***localhost:3000//product-inventory/final***




11.- Listar las ordenes de producción que terminaron de fabricarse el 1 de enero del año 2013

Al igual que en la operacion 7, los datos son obtenidos especificando solo la ruta:

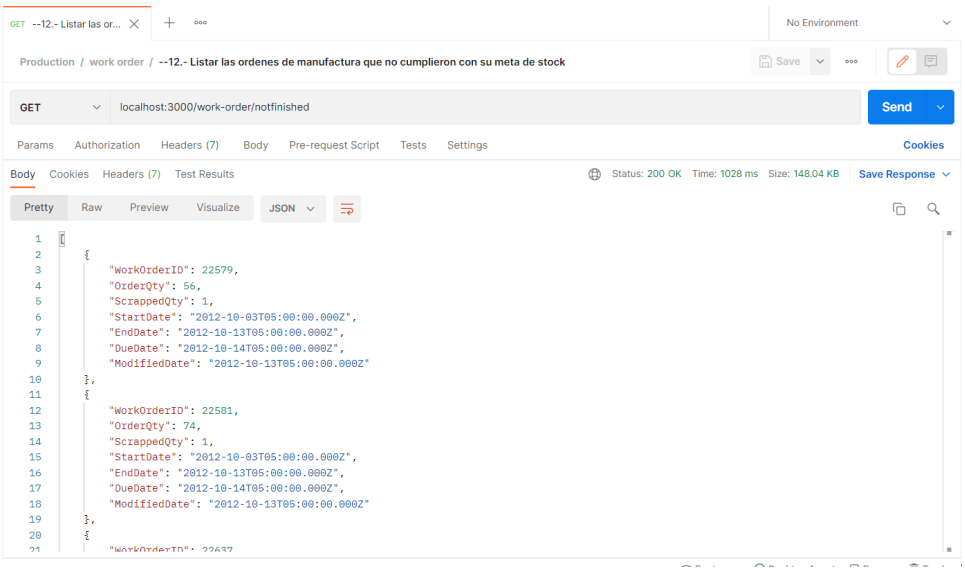
***localhost:3000//work-order/date/2013-01-01***




12.- Listar las ordenes de manufactura que no cumplieron con su meta de stock.

Solo se especifica la ruta para recuperar los datos:

***localhost:3000//work-order/notfinished***




13 .- Agregar 666 registros a un estante tipo 4 a un estante R en un compartimiento de almacenamiento 100.

Al igual que en la operacion 5 de este documento, esta es una operacion insert que se invoca con la ruta:

***localhost:3000//product-invetory***

De la misma manera se tiene que editar un “formulario” con ayuda de POSTMAN