

## Desafío - Tienda de Joyas

- Para realizar este desafío debes haber estudiado previamente todo el material disponible correspondiente a la unidad.
- Desarrollo desafío:
  - El desafío se debe desarrollar de manera grupal

### Descripción

La tienda de joyas My Precious Spa necesita cambiar su aplicación de escritorio por una moderna y dinámica. Para realizar esta tarea, contactó a un desarrollador Full Stack Developer que desarrolle la API REST de una aplicación cliente para satisfacer las necesidades puntuales de sus usuarios de una forma eficiente, mantenible y eficaz.

Deberás crear una API REST que permita:

1. Límite de recursos
2. Filtro de recursos por campos
3. Paginación
4. Ordenamiento
5. Estructura de datos HATEOAS

Para realizar este desafío necesitarás ejecutar el siguiente script *sql* en tu terminal **psql** para crear la base de datos y la tabla que utilizaremos:

```
CREATE DATABASE joyas;
\c joyas;

CREATE TABLE inventario (id SERIAL, nombre VARCHAR(50), categoria
VARCHAR(50), metal VARCHAR(50), precio INT, stock INT);

INSERT INTO inventario values
(DEFAULT, 'Collar Heart', 'collar', 'oro', 20000 , 2),
(DEFAULT, 'Collar History', 'collar', 'plata', 15000 , 5),
(DEFAULT, 'Aros Berry', 'aros', 'oro', 12000 , 10),
(DEFAULT, 'Aros Hook Blue', 'aros', 'oro', 25000 , 4),
(DEFAULT, 'Anillo Wish', 'aros', 'plata', 30000 , 4),
(DEFAULT, 'Anillo Cuarzo Greece', 'anillo', 'oro', 40000 , 2);
```

A continuación te mostramos imágenes de consultas HTTP realizadas a las rutas **GET** correspondientes a los requerimientos de este desafío:

- Consulta de joyas con cláusulas en estructura de datos HATEOAS:  
[localhost:3000/joyas?limits=3&page=2&order\\_by=stock\\_ASC](http://localhost:3000/joyas?limits=3&page=2&order_by=stock_ASC)

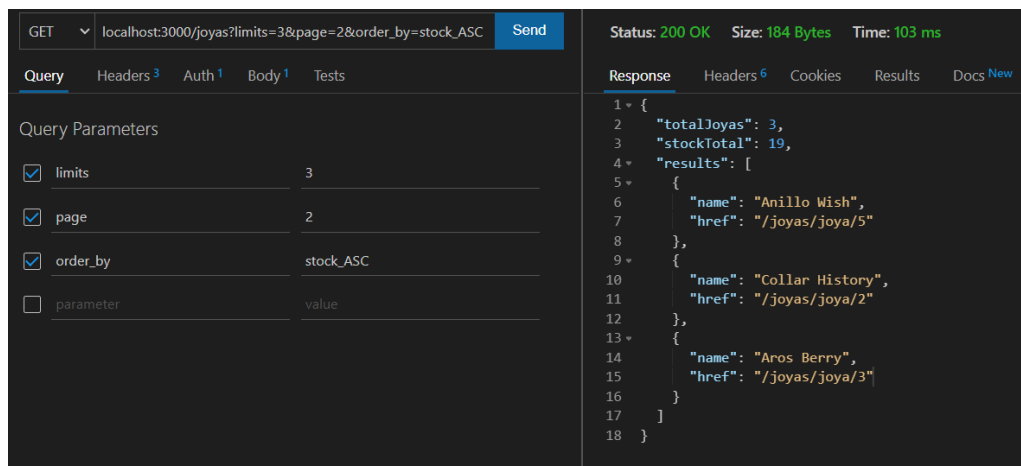


Imagen 1. Joyas en HATEOAS con cláusulas  
Fuente: Desafío Latam

- Filtrando las joyas por precio máximo, mínimo, categoría y metal:  
[http://localhost:3000/joyas/filtros?precio\\_min=25000&precio\\_max=30000&categoria=aros&metal=plata](http://localhost:3000/joyas/filtros?precio_min=25000&precio_max=30000&categoria=aros&metal=plata)

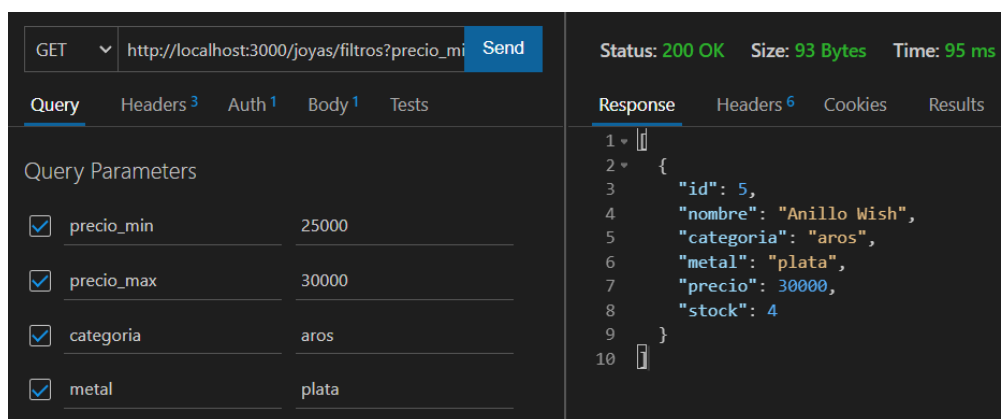



Imagen 2. Joyas filtradas por precios, categoría y metal  
Fuente: Desafío Latam

## Requerimientos

1. Crear una ruta **GET /joyas** que:
  - a. Devuelva la estructura HATEOAS de todas las joyas almacenadas en la base de datos (1.5 puntos)

- b. Reciba en la query string los parámetros (2 puntos):
      - i. **limits**: Limita la cantidad de joyas a devolver por página
      - ii. **page**: Define la página
      - iii. **order\_by**: Ordena las joyas según el valor de este parámetro, ejemplo: `stock_ASC`
  - 2. Crear una ruta **GET /joyas/filtros** que reciba los siguientes parámetros en la query string: (3.5 puntos)
    - a. **precio\_max**: Filtrar las joyas con un precio **mayor** al valor recibido
    - b. **precio\_min**: Filtrar las joyas con un precio **menor** al valor recibido.
    - c. **categoria**: Filtrar las joyas por la categoría
    - d. **metal**: Filtrar las joyas por la categoría
  - 3. Implementar middlewares para generar informes o reportes de alguna actividad o evento específico que ocurra en cada una de las rutas. (1 puntos)
-  Un ejemplo de reporte es crear un registro indicando a qué ruta se le realizó una consulta.
- 4. Usar **try catch** para capturar los posibles errores durante una consulta y la lógica de cada ruta creada. (1 puntos)
  - 5. Usar las consultas parametrizadas para evitar el SQL Injection en la consulta a la base de datos relacionada con la ruta **GET /joyas/filtros** (1 puntos)