

TP N°2: API de E-Commerce



Idea: vamos a utilizar TypeScript con NodeJS para levantar un pequeño servidor local que atienda los requests a ciertas urls. Los datos del e-commerce se encuentran en una base de datos MySQL.

Rutas que tenemos que poder contestar:

- **GET /productos:** responde con el listado de productos. Esta ruta puede recibir los siguientes parámetros:
 - búsqueda: un string a buscar en el título del producto.
 - orden: ordena el listado según el valor del parámetro. Los valores posibles son:
 - precio: ordena precio de menor a mayor.

- calificación: ordena por calificación de los vendedores, de mayor a menor.
- usado: filtra el listado según el valor del parámetro:
 - 0: solamente trae los productos nuevos.
 - 1: solamente trae los productos usados.

Un ejemplo de una url podría ser

<http://localhost:3000/productos?busqueda=PS2&usado=1&orden=precio>

Debería retornar el JSON con los productos usados que tengan en su nombre “PS2” ordenados por precio.

- **GET /usuarios/{id_usuario}/fav:** devuelve el listado de productos favoritos de un usuario.
- **POST /usuarios/{id_usuario}/fav:** agrega un producto al listado de favoritos del usuario. El id_producto se pasa por parámetro.
- **DELETE /usuarios/{id_usuario}/fav:** elimina un producto del listado de favoritos del usuario. El id_producto se pasa por parámetro.
- **GET /usuarios/{id_usuario}/compras:** devuelve el listado de compras de un usuario.
- **POST /usuarios/{id_usuario}/compras:** genera una nueva compra si es posible. Requiere los siguiente parámetros:

- id_producto: el id del producto a adquirir.
- cantidad: la cantidad a adquirir del stock.
- **GET /usuarios/{id_usuario}/calificaciones:** trae el detalle de las calificaciones del usuario, tanto de comprador como de vendedor.
- **POST /usuarios/{id_usuario}/calificaciones:** permite calificar al usuario, ya sea por una compra o una venta. Requiere los siguiente parámetros:
 - id_calificante: id del usuario que quiere hacer la calificación.
 - id_operacion: el id de la compra/venta por la cual lo quiere calificar.

Para los grupos de 3: todas las tablas del modelo (producto, usuario, etc.) deben contar con su clase que a su vez debe proveer métodos para hacer las consultas. Ejemplos de uso:

```
// Busca el producto con id 3.  
Producto p = Producto.find(3);  
if (p != null) {  
    // Modifica el precio  
    p.precio = 200;  
    // Guarda los cambios en la base  
    p.save();  
}
```

```
/* Busca todos los productos usados con precio < 1000, ordenados de manera descendente*/
```

```
Set<Producto> productos = Producto  
    .where('precio', '<', '1000')  
    .where('usado', '=', '1')  
    .orderby('precio', 'DESC')  
    .get();
```

Métodos de interacción con la db requeridos para cada clase:

- find: busca el elemento por id, retorna el objeto si se encuentra, null en caso contrario.
- save: guarda el estado del objeto en la base.
- where: agrega una cláusula WHERE a la query, con los parámetros que reciba.
- orderby: agrega una cláusula ORDER BY a la query, con los parámetros que reciba.
- get: ejecuta una query en la base y retorna un conjunto con los objetos resultado.
- Links útiles:
 - [Ejemplo "Hello World" de Express](#)
 - [Paquete MySQL de NodeJS](#)