Challenge Backend

A continuación, presentamos los requerimientos de un pequeño gestor de catálogo, que deberás desarrollar a partir de una base de datos dada.

Stack tecnológico:

Utilizar Strapi para resolver todo el challenge es un Plus. En su defecto podés utilizar:

- NodeJs.
- Express (o cualquier FW liviano).
- Swagger (para documentar y ejecutar).
- o Collections de Postman (Con Documentación).
- No utilizar ORMs. Usar Queries de MySQL.
- Evitar usar paquetes de NPM.

MATERIALES:

Con la Base de datos entregada en el siguiente diagrama, deberás generar los siguientes endpoints. https://dbdiagram.io/d/DER_Producto-Pedido-68be2d6c61a46d388edf432e

* Desde este link podrás descargar la estructura de la DB. Y con los JSON podrás poblar de datos la DB.

TE PEDIMOS:

Los siguientes endpoints que representan un gestor de catálogos.

GET /search: buscador de productos. Debe buscar por SKU y/o descripción corta y larga. Además, el resultado debe permitir ordenamiento por fecha de creación, precio, categoría y tener paginador.

GET /product: listado de productos. Se beben mostrar sólo los productos que tengan cantidad mayor a 0, estado igual a 3, que tengan imagen y precio mayor a 0 o distinto de NULL. También debe poder ser ordenado por fecha de creación, precio, categoría y tener paginador.

GET /product/{SLUG}: detalle de un producto. Debe formar su propia URL según un SLUG (a partir de su descripción larga). Además, algunos productos relacionados con algún criterio de similitud (no importa cuál).

GET /order: este endpoint debe permitir la búsqueda de ordenes por su ID de pedido, cliente (CUIT) y también por fecha de creación.

Por ej: /orders?created_at_min=2024-06-01T23:59:59&created_at_max=2025-09-10T23:59:59

POST /order: es la ORDEN/PEDIDO de venta en sí misma. Es el producto que compra un cliente. Debe tener un incremental de número de orden. Los productos que tengan 0 (cero) en cantidad de stock, no se debe permitir comprar.

Cada una de las respuestas de los endpoints, en caso de error, debe cumplir con un standard de código, mensaje y descripción. Por ej:

```
{
  "code": 400,
  "message": "Validation Error",
  "description": "Out of Stock"
}
```

ENTREGABLE:

Entregar un repositorio (preferentemente Github).

El Challenge se debe poder ejecutar. Podes hostearlo o dockerizarlo (siempre con README para su correcta ejecución). Si no puede ser ejecutado fácilmente, no se tendrá en cuenta.

Criterios que vamos a tener en cuenta:

Correctitud funcional

- Que cumpla con todos los requisitos planteados en el enunciado.
- Manejo de casos borde (ejemplo: entradas inválidas, datos vacíos, límites altos de registros).

Diseño de arquitectura

- Separación clara de capas (controllers, services, repositorios).
- Uso adecuado de patrones de diseño (ej. Repository, Factory, DTOs, etc. si aplica).
- Claridad en la estructura de carpetas y modularización.

Seguridad

- Manejo de inputs (validaciones, sanitización).
- Uso de autenticación/autorización (aunque sea básico).

Manejo de datos

- Modelado correcto de base de datos.

- Uso eficiente de queries (evitar N+1, uso de índices, joins necesarios).

Pruebas

- Tests unitarios y/o de integración.
- Cobertura mínima.
- Mocking de dependencias externas.

Documentación

- README claro con instrucciones de instalación, uso y endpoints.
- Ejemplos de request/response.

Mantenibilidad

- Código legible, con comentarios cuando es necesario.
- Nombres de variables, funciones y clases expresivos.
- Uso de linters y formateadores.

Muchas gracias!