Carrito de compra

La idea es crear una aplicación muy sencilla para que la use un frontal. Esta aplicación debe tener las siguientes características:

1. Listado de productos

La aplicación utiliza dos productos distintos: gorras y camisetas. De las gorras se necesita tener su color principal, colores secundarios, color del logo y la marca. En cuanto a las camisetas se necesita su color principal, colores secundarios, talla, marca, composición (porcentajes de algunos de los siguientes materiales: algodón, lino, cáñamo, poliéster, nylon, lana y seda), tallaje (hombre, mujer o unisex) y si tiene mangas o no. De ambos tipos de producto se necesita además la fecha de inclusión en el catálogo, la URL a la foto del mismo, el precio por unidad, el stock inicial (propiedad que no se debe poder modificar una vez creado el producto) y el stock actual. Además, añadir un campo que muestre en una única frase descriptiva (uniendo los valores como tú prefieras): el tipo de producto + marca + colores + año inclusión al catálogo + talla + composición (estos dos últimos solo para camisetas).

Este primer endpoint solamente devolverá un listado de todos los productos de ambos tipos que haya en la BBDD ordenados por tipo de producto (primero gorras y después camisetas) y dentro de cada uno por fecha de inclusión en el catálogo (más recientes primero). Se debe devolver toda la información de cada tipo de producto a excepción del stock inicial, que no hay que devolverlo (solo el actual).

2. CRUD de producto

Se deberá crear el CRUD de producto para que el front haga uso de él. En cuanto a la eliminación de productos, es importante que se haga un borrado lógico ya que de lo contrario provocaría error si el producto existe en algún carrito comprado o pendiente por comprar. Los productos borrados del catálogo no deben ser devueltos en el listado de productos (punto 1) pero sí al ver el carrito (punto 4). También es importante poder saber cuándo se ha borrado un determinado producto.

3. Añadir producto al carrito

Una vez obtenido el listado de productos, el front debe tener la posibilidad de añadirlos al carrito.

Este segundo endpoint recibirá el producto a añadir y lo añadirá al carrito junto con la cantidad de dicho producto que se quiere añadir (por defecto la cantidad es una unidad). Cantidades negativas restarán elementos del carrito. El stock deberá ser actualizado en cuanto el producto entre al carrito.

Hay que tener en cuenta que el carrito no existirá antes de añadirse el primer producto, por lo que habrá que crearlo. Como no se va a implementar ningún modelo de usuario, asociaremos el carrito al día actual y su estado (comprado o no). Solo podrá existir un carrito pendiente de comprar por día. Si en el día de hoy ya existe un carrito pendiente de compra, usamos ese; si no, lo creamos. En el momento en que un carrito es comprado debe ser posible volver a crear otro en el mismo día.

4. Ver el carrito

El front en cualquier momento podrá ver el contenido del carrito (incluso antes de añadir un producto, en cuyo caso deberá verlo vacío).

Este endpoint simplemente devolverá una lista de los productos que hayan sido añadidos y el sumatorio del total de los productos añadidos. En esta ocasión, para los productos, solo hace falta que se devuelva su ID, descripción, URL de la foto, cantidad y precio por unidad.

5. Comprar

Este último endpoint va a recibir un formulario con los datos personales del cliente: nombre, apellidos, dirección postal, email y teléfono. Simularemos la compra enviando al cliente un email con el resumen de su compra.

6. Actualización de stock

En un proyecto con el alcance tan pequeño de esta prueba no haría falta ya que, si no hay errores, solamente hay un servicio que modifica el stock de productos y por tanto siempre debería ser correcto, pero vamos a simular un proceso que en sistemas más grandes sí se hace. Para evitar desajustes en el stock de productos, debemos crear un proceso automático que calcule y actualice el stock de los productos cada hora.

7. Testing

Añade los tests que creas conveniente al código que escribas.

Puntos a tener en cuenta

- El proyecto se debe desarrollar en Python usando el framework Django.
- Sube la app a tu cuenta de GitHub.
- El proyecto se debe poder levantar en cualquier máquina sin dependencias de ningún tipo, por lo que se pide dockerizarlo.
- Utiliza cualquier solución (incluir la colección Postman que hayas usado, incluir Swagger, etc.) que nos facilite poder probar la aplicación.
- Tendrás que hacer una carga inicial de productos (5 gorras y 5 camisetas con un stock inicial de cada producto de 10 unidades). Las demás propiedades de cada uno rellénalas a tu gusto. Como recomendación, crea un fixture que se importe en la BBDD al levantar la app controlando que no se dupliquen los productos la sucesivas veces que la levantes.
- Para el envío de email, ya que no vamos a implementar un modelo de usuario para esta prueba ni pedimos que configures un servidor de envío de emails puedes utilizar como EMAIL_BACKEND la escritura de emails por consola, aunque siempre puedes usar otro backend para el envío (se valorará).
- El contenido o formato de dicho email es libre y no afecta a hacer mejor o peor la prueba. Créalo como más te guste (en texto plano ya es perfectamente válido).
- No te preocupes si no has tocado alguna cosa que se pide. La idea de esta prueba es ver cómo te desenvuelves y resuelves problemas.
- Siéntete libre de preguntar cualquier cosa que no quede clara sobre la prueba.