Universidad Intercontinental de la Empresa

Práctica 1

Unidad I
Fecha de entrega: 26/09/25 14:59 (hora límite)
Entregables: Fichero .zip conteniendo los notebooks P1.1, P1.2 y la carpeta del proyecto P1.3 con todos los scripts y ficheros necesarios para su ejecución.

Objetivo

Tras la realización de las partes preliminares de la Práctica 1 (los notebooks P1.1 y P1.2), deberéis poner en práctica los conceptos básicos de Python y POO.

El objetivo de esta práctica es diseñar e implementar un conjunto de clases en Python que representen los elementos básicos de una tienda online: productos, usuarios y pedidos. Esta primera versión del sistema se ejecutará en consola, pero debe estar diseñada de manera que pueda evolucionar hacia una API REST con FastAPI en prácticas posteriores.

Clase producto (2p)

En primer lugar, tendrás que crear la clase **Producto**, que servirá como clase padre para todos los productos de la tienda. Un producto debe tener un identificador único que se genere automáticamente en el momento de su creación, además de un nombre, un precio y una cantidad de stock disponible. La clase debe incluir métodos que permitan consultar si hay suficiente stock de un producto en función de una cantidad solicitada, así como un método para actualizar el stock cuando se realicen compras o reposiciones.

La clase Producto tendrá además un método __str__() que devuelva en texto las características principales del producto. Este método deberá ser sobrescrito en las clases hijas para que cada tipo de producto pueda ofrecer una descripción adaptada a sus propios atributos.

A partir de esta clase base, deberás implementar al menos dos subclases que amplíen la información de los productos. Una de ellas será la clase **ProductoElectronico**, que además de los atributos heredados, tendrá un campo adicional que indique los meses de garantía. Otra será la clase **ProductoRopa**, que incorporará la talla y el color. Recuerda que, en ambas subclases, el método __str__() debe ser sobrescrito para incluir la información extra en el texto mostrado al usuario (es decir, incluyendo la información de la garantía, o la talla y color).

Clase usuario (2p)

En paralelo, tendrás que diseñar la clase **Usuario**, que representa a cualquier persona que interactúe con la tienda. Un usuario debe tener un identificador único, un nombre y un correo electrónico. Esta clase incluirá un método is_admin() que, en la versión por defecto, devolverá siempre False. A partir de aquí, deberás crear dos subclases. La primera será la clase **Cliente**, que heredará de Usuario e incluirá como atributo adicional

la dirección postal. La segunda será la clase **Administrador**, que representará a los gestores de la tienda. Esta subclase debe sobrescribir el método is_admin() para que devuelva True, reflejando así el rol diferenciado de este tipo de usuario.

Clase pedido (2p)

Para completar el modelo de la tienda, deberás implementar la clase **Pedido**, que estará vinculada a un usuario de tipo cliente y a una lista de productos con sus cantidades correspondientes. Cada pedido debe tener también un identificador único, una fecha de pedido y debe ser capaz de calcular el importe total de la compra en función de los precios y las cantidades solicitadas. El método __str__() deberá devolver un resumen legible del pedido, incluyendo el nombre del cliente y el total de la compra.

Servicios (3p)

Con estas clases, deberás ser capaz de representar de forma estructurada los elementos básicos de una tienda online, garantizando que las operaciones fundamentales —como comprobar stock, registrar un usuario o generar un pedido— puedan realizarse desde código.

Para ello, además de las clases del modelo, deberás implementar una capa de **servicios** que gestione las operaciones de la tienda de manera centralizada. Para ello, se creará la clase **TiendaService**, que actuará como intermediario entre los usuarios y los productos.

La clase TiendaService deberá proporcionar métodos que permitan registrar un nuevo usuario indicando su tipo (cliente o administrador) y sus atributos básicos; añadir productos de diferentes categorías al inventario; eliminar un producto a partir de su identificador; listar todos los productos disponibles en la tienda y realizar un pedido verificando primero que el usuario que lo solicita existe y que hay stock suficiente de los productos solicitados. En caso afirmativo, el servicio deberá generar un objeto Pedido, descontar las unidades correspondientes del inventario y devolver el pedido ya creado. Además, deberá incluirse un método capaz de listar todos los pedidos por orden de fecha de un usuario en base a su id.

Main (1p)

Finalmente, se deberá implementar el archivo main.py, que actuará como punto de entrada del programa y servirá para probar manualmente el funcionamiento del sistema. En este archivo, se debe crear una instancia de TiendaService, registrar al menos tres clientes y un administrador, crear cinco productos de diferentes categorías y añadirlos al inventario. A continuación, tendrán que listar los productos para comprobar que el inventario se ha llenado correctamente, simular tres pedidos realizados por distintos clientes con varios productos y comprobar el histórico de pedidos de un cliente. El resultado de este pedido deberá mostrarse en consola, tanto el resumen del pedido como el stock actualizado de los productos.

Estructura

En base a las clases definidas anteriormente, el proyecto tendrá la siguiente estructura:

Tienda_online/

- models/
  - __init.py__
  - Producto.py
  - Usuario.py
  - Pedido.py
- Services/
  - __init.py__
  - Tienda_service.py
- main.py

Restricciones y recomendaciones

- Usar **tipado de datos** en atributos y métodos (Python type hints).
- Usar uuid o random para generar identificadores únicos.
- Mantener el código modular: cada clase principal en un archivo independiente dentro de models/.
- Usar convenciones de nombres de Python (PEP8).
- Documentar el código con docstrings simples.