Trabajo en Clase (Comunicación de Servicios)

Ejercicio: Plataforma Distribuida de Gestión de Inventario con Microservicios y Pyro4

La empresa **TecnoMarket**, dedicada a la venta de productos electrónicos, quiere modernizar su sistema de inventarios para soportar:

- crecimiento del negocio,
- múltiples sucursales,
- accesos concurrentes,
- actualización en tiempo real de stock.

Actualmente toda la información se maneja en una base de datos centralizada y un sistema monolítico, lo que genera:

- lentitud,
- cuellos de botella en horas pico,
- inconsistencias en el inventario,
- caídas cuando varios usuarios actualizan el stock simultáneamente.

Los directivos quieren migrar hacia una **arquitectura de microservicios**, donde cada funcionalidad esté desacoplada.

Para comenzar este proceso, solicitan la implementación de un prototipo sencillo en el cual se desarrolle un **microservicio de inventario distribuido usando Pyro4**.

Objetivo general del ejercicio

Desarrollar un **microservicio de Inventario** que funcione como un **objeto remoto Pyro4**, persistiendo los datos en un archivo (JSON, CSV o SQLite), y que permita a varios clientes consultar y modificar el inventario de forma segura.

Componentes del sistema

1. Microservicio de Inventario (servidor Pyro4)

Debe actuar como un **servicio independiente**, ofreciendo las siguientes operaciones remotas:

✔ Métodos obligatorios

1. **agregar_producto(id, nombre, stock)**
    - Inserta un nuevo producto.
    - Si ya existe, debe rechazarlo.
2. **obtener_producto(id)**
    - Devuelve la información del producto.
3. **actualizar_stock(id, nuevo_stock)**
    - Modifica el stock del producto.
    - Debe registrar la modificación en la persistencia.
4. **listar_productos()**
    - Devuelve todos los productos almacenados.
5. **eliminar_producto(id)**
    - Elimina un producto del inventario.

Requisitos técnicos

- El inventario debe persistirse usando **JSON o SQLite**.
- El servicio debe tener **carga desde persistencia al iniciar** y **guardado automático después de cada cambio**.
- Debe estar protegido con **threading.Lock()** para evitar race conditions.

- Debe registrarse en el Name Server con el nombre:inventario.servicio

Cliente de Inventario (client.py)

Los estudiantes deben crear un cliente que:

✔ Funciones

- Se conecte al microservicio vía `Pyro4.Proxy("PYRONAME:inventario.servicio")`.
- Ofrezca un menú como:

1. Agregar producto

2. Obtener producto

3. Actualizar stock

4. Listar productos

5. Eliminar producto

0. Salir

Envíe las solicitudes al microservicio y muestre la respuesta.

Maneje errores como:

- producto no encontrado,
- entradas no válidas,
- desconexiones.

Persistencia obligatoria

Debe usarse una de las siguientes opciones:

Opción A: Archivo JSON

- Archivo `inventario.json` que guarda una lista de productos.
- Al iniciar, si el archivo no existe, se genera vacío.

Opción B: SQLite

- Tabla `productos(id TEXT PRIMARY KEY, nombre TEXT, stock INTEGER)`.

Los estudiantes son libres de elegir, pero deben justificar su elección.

Concurrencia obligatoria

- El servidor debe usar `threading.Lock()` para garantizar que lecturas y escrituras al inventario sean seguras.
- Los estudiantes deben demostrar que su microservicio soporta acceso concurrente.

Producto final esperado

Los estudiantes deben entregar:

1. **server.py** — microservicio Pyro4 con persistencia
2. **client.py** — cliente de consola
3. Archivo de **persistencia** (`inventario.json` o `.db`)
4. **Explicación breve** (1 página) del funcionamiento y arquitectura del microservicio
5. Ejemplo de prueba con múltiples clientes concurrentes (pueden ser terminales o threads)