Actividad 1.7 CRUD + DAO + Properties

Si se tienen muchos elementos en nuestro código que tiene valores constantes (rutas, separadores CSV, credenciales de base de datos claves de API, etc.) ¿investiga 4 formas distintas de gestionar estos elementos con valores constantes para un proyecto java "puro" (no es parte de un framework, ni carga dependencias como Maven/Gradle)?

Archivos .properties: Clave-valor externos estándar gestionados con la clase Properties de Java nativa.

Interfaces con constantes (public static final): Agrupación directa de valores estáticos en código mediante una interfaz de constantes.

Archivos JSON o YAML externos: Ficheros estructurados leídos mediante librerías externas ligeras (o mapeo manual básico).

Variables de entorno del sistema (System.getenv()): Lectura de parámetros directamente configurados en el sistema operativo o contenedor.

¿Qué ventajas tiene usar un archivo .properties frente a escribir valores directamente como constantes en el código fuente?

Desacoplamiento y mantenimiento: Permite modificar valores (como rutas o URLs) sin necesidad de recompilar el código fuente.

Flexibilidad por entorno: Facilita cambiar configuraciones fácilmente entre entornos de desarrollo, pruebas y producción sin tocar Java.

¿Qué tipo de valores son adecuados para guardarse en un archivo .properties, y cuáles no? ¿Por qué?

Adecuados: Rutas de archivos, URLs de conexión, timeouts, nombres de tablas o parámetros operativos no sensibles.

Inadecuados: Credenciales críticas, contraseñas en claro o claves API maestras, ya que el archivo está expuesto en texto plano.

¿Qué limitaciones tiene el uso de archivos .properties y qué otras alternativas modernas existen (por ejemplo, YAML, JSON, bases de datos o Spring @Value)?

Limitaciones: Carece de jerarquía compleja, tipos de datos avanzados (todo es texto) y validación nativa.

Alternativas: JSON y YAML ofrecen estructuras anidadas complejas; bases de datos o gestores de secretos centralizan y protegen la configuración.

¿Cómo cargar desde Java un archivo .properties y acceder a sus valores? ¿Qué ocurre si una clave no existe?

Carga y acceso: Se usa Properties props = new Properties(); props.load(new FileInputStream("ruta")); props.getProperty("clave");

Clave ausente: El método devuelve null a menos que se use una sobrecarga indicando un valor por defecto (props.getProperty("clave", "defecto")).