**SERIE II:**

**¿Qué ventajas tiene usar JPA con respecto a una conexión manual a base de datos usando JDBC puro?**
**Menciona al menos dos.**

La verdad es que trabajar directamente con JDBC puede volverse tedioso rápidamente mas si somos principiantes . Imagínemenos tener que escribir todas las consultas SQL a mano, gestionar conexiones, cerrar recursos... es mucho código repetitivo y propenso a errores. Ahí es donde JPA marca la diferencia de una forma muy sutil.
Lo primero que notáremos al usar JPA es que trabajamos con objetos normales de Java - tus entidades - en lugar de andar con Strings de SQL por todos lados. Simplemente anotámos una clase con @Entity y JPA se encarga de mapearla a una tabla. Para guardar un objeto, solo necesitamos entityManager.persist(objeto). ¡Mucho más limpio que armar un INSERT manual con JDBC! Otra cosa que hay que agradecer es cómo JPA maneja las transacciones automáticamente. Con JDBC, tenemos que recordar hacer commit() y rollback() vos mismo. En JPA, con una simple anotación @Transactional ya está todo cubierto. Y ni hablar del cache que trae incorporado - evita que hagas consultas repetidas a la base sin darte cuenta.

**Explica brevemente qué papel cumple el archivo persistence.xml (o la configuración application.properties) en una aplicación Java con JPA. ¿Qué tipo de información contiene y por qué es**
**importante?**

El persistence.xml (o application.properties en Spring) es como el manual de instrucciones que le decimos a JPA cómo conectarse y comportarse. Acá es donde le indiquemos:

- Dónde está la base de datos (esa URL que siempre empieza con jdbc:mysql...)
- Qué usuario y contraseña usar
- Qué "dialecto" de SQL hablar (porque cada base de datos tiene sus particularidades)
- Si queremos que JPA cree o actualice las tablas automáticamente

Lo bueno es que toda esta configuración queda en un solo lugar. Si mañana cambiámos de MySQL a PostgreSQL, solo tenemos que modificar este archivo. O si estamos probando en local y después pasás a producción, simplemente ajustás las propiedades sin tocar el código. En proyectos reales, este archivo se vuelve super importante porque centraliza todo lo que JPA necesita saber para funcionar correctamente. Sin él, sería como tener un auto sin llave: por más motor que tenga, no vas a poder hacerlo andar. Por eso es muy