1. Diseño e implementación del patrón DAO

Responsabilidad única

El DAO (Data Access Object) centraliza todas las operaciones CRUD (crear, leer, actualizar, eliminar) contra la tabla `gatos`. Su única responsabilidad es traducir llamadas de la capa de negocio a consultas SQL y devolver objetos de dominio (`GatoVO`) o listas de estos, sin involucrarse en la lógica de presentación ni de negocio.

Desacoplamiento

La clase `GatoDAO` actúa como frontera entre la aplicación y la base de datos. Ni la vista ni los controladores conocen detalles de JDBC o de la estructura de la tabla; simplemente solicitan a `GatoDAO` que “consulte el gato con ID X” o “liste todos los gatos”, recibiendo de vuelta objetos ya poblados. Esto permite, por ejemplo, cambiar a otro motor de base de datos o reemplazar JDBC por JPA: bastaría con modificar o sustituir la implementación de `GatoDAO`, sin tocar el resto del código.

Gestión de recursos

Cada método de `GatoDAO` obtiene la conexión mediante `ConexionBD`, ejecuta la consulta o actualización y luego “desconecta” (pone a null la referencia). Así se garantiza que no queden conexiones abiertas, aunque en la práctica se recomendaría un cierre explícito de `ResultSet`, `Statement` y `Connection` para manejar correctamente excepciones.

Tratamiento de excepciones

Las operaciones en `GatoDAO` capturan o lanzan `SQLException`, permitiendo a la capa superior

(controlador o servicio) decidir cómo notificar errores al usuario o qué estrategia de reintento aplicar.

---

2. Diseño e implementación del patrón Singleton

Única instancia de la conexión

Para evitar crear múltiples conexiones costosas a la base de datos, la clase `ConexionBD` debe implementarse como Singleton: sólo una instancia gestiona internamente el objeto `Connection`. De esta forma, todos los DAOs comparten la misma conexión, reduciendo el overhead de abrir/cerrar conexiones continuamente y asegurando un punto único de configuración (URL, usuario, contraseña).

Control de acceso global

Al exponer un método estático `getInstance()` (o en este caso un único `getConnection()` tras inicializar internamente la instancia), se ofrece un punto de acceso global y controlado. No es posible crear otro objeto `ConexionBD` directamente, lo que previene inconsistencias o parámetros de conexión divergentes.

Configuración centralizada

Los métodos estáticos para ajustar URL, usuario y contraseña funcionan sólo sobre la única instancia. Así, antes de que la app haga cualquier operación, se establece la configuración una vez (por ejemplo, al arrancar el sistema), y todos los DAOs leerán esos valores.

---

3. Aplicación práctica conjunta

Al iniciar la aplicación, se configura primero la conexión:

Se invocan los setters de URL, usuario y contraseña en `ConexionBD`.

La primera petición de `getConnection()` crea la conexión real.

En cada operación de datos, los controladores o la capa de negocio:

Instancian un `GatoVO` con los datos o vacío para consulta.

Llaman a `GatoDAO.consultarGato(...)`, `insertarGato(...)`, etc.

Reciben el objeto o el resultado y actualizan la vista.

Beneficios obtenidos:

Mantenibilidad: la lógica de acceso a datos está aislada.

Reusabilidad: si surge otra entidad (por ejemplo, DueñoDAO), reutiliza el mismo patrón y la misma conexión.

Consistencia: toda la aplicación usa la misma configuración de conexión y el mismo pool de recursos.