# **Patrón de Diseño DAO**

El patrón de diseño DAO (Data Access Object) se utiliza para separar la lógica de acceso a datos de una aplicación de la lógica de negocio y la interfaz de usuario. Su principal objetivo es proporcionar una abstracción entre el código de la aplicación y la fuente de datos subyacente, como una base de datos, un servicio web, un archivo, etc. Aquí se describen algunas de las razones clave por las que se utiliza el patrón DAO:

Separación de responsabilidades: El patrón DAO ayuda a separar las preocupaciones de la aplicación. La lógica de acceso a datos se separa de la lógica de negocio y la interfaz de usuario, lo que facilita el mantenimiento y la escalabilidad.

Reutilización de código: Al encapsular todas las operaciones de acceso a datos en un conjunto de clases DAO, puedes reutilizar estas clases en diferentes partes de tu aplicación. Esto evita la duplicación de código y promueve la coherencia en el acceso a datos.

Flexibilidad y adaptabilidad: Si necesitas cambiar la fuente de datos (por ejemplo, pasar de una base de datos relacional a una base de datos NoSQL), puedes hacerlo sin afectar el resto de la aplicación. Solo necesitas cambiar la implementación del DAO.

Seguridad: El patrón DAO puede ayudar a mejorar la seguridad de tu aplicación al permitir el uso de técnicas como la validación de datos y la prevención de ataques de inyección de SQL en un solo lugar, el DAO.

Pruebas unitarias: Al tener una capa de acceso a datos separada, es más fácil escribir pruebas unitarias para esta capa sin necesidad de interactuar con la base de datos real. Puedes crear implementaciones simuladas o de prueba del DAO para las pruebas unitarias.

Mejora del rendimiento: Un DAO bien diseñado puede implementar estrategias de almacenamiento en caché o de optimización de consultas para mejorar el rendimiento de las operaciones de acceso a datos sin afectar la lógica de negocio.

**Capas**

Alumno: Esta clase representa la entidad Alumno y contiene los atributos nombre y edad. También se incluye los métodos getters y setters, constructores y toString.

AlumnoDAO: Esta interfaz define los métodos que se utilizarán para acceder a los datos relacionados con los alumnos. Contiene las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) relacionadas con la entidad Alumno.

AlumnoDAOImpl: Esta clase implementa la interfaz AlumnoDAO y contiene la lógica de acceso a datos específica para la base de datos MySQL. Aquí es donde se ejecutan consultas SQL y se realizan operaciones con la base de datos.

Conexion: Esta clase puede manejar la conexión a la base de datos MySQL. Puede ser una clase que tenga métodos para conectar y desconectar con la base de datos, así como para obtener instancias de Connection.

En cuanto a las reglas de negocio, depende de la complejidad de la aplicación. Si la aplicación tiene reglas de negocio significativas, podríamos crear una clase separada llamada ***AlumnoService*** o algo similar. Esta clase podría contener métodos que implementen las reglas de negocio específicas de nuestra aplicación, como validar datos de alumnos antes de realizar operaciones de inserción o actualización en la base de datos.