Patrón DAO

DigitalHouse>
Coding School

Certified Tech Developer
The Ultimate Degree

Propósito

Con el patrón DAO desacoplamos los datos propiamente dichos del lugar donde se almacenan o de la tecnología de almacenamiento. Es decir, para nuestro sistema será indiferente si utilizamos PostgreSQL o MySQL ya que en cualquiera de los dos casos la forma de comunicarse será la misma.

Solución

Este patrón nos propone:

1. Crear una interfaz en la que definamos todas las operaciones que queremos realizar, generalmente las más utilizadas son crear, actualizar, eliminar y leer.
2. Crear las diferentes implementaciones de esta interfaz, por ejemplo una implementación para postgreSQL y otra para MySQL.

De esta manera la **capa de negocio**, es decir donde se encuentra la lógica principal de nuestro sistema, se comunicará con la **capa de persistencia**, pero no conocerá los detalles de implementación ni se enterará si estamos utilizando PostgreSQL o MySQL ya que cualquiera sea la implementación, tendrá los mismos métodos que la interfaz.

Solución

Cuando hablamos de **“capas”**, ¿a qué nos referimos?

En un sistema se denomina “capa” a una agrupación de componentes (clases) de iguales responsabilidades.

Diagrama Conceptual

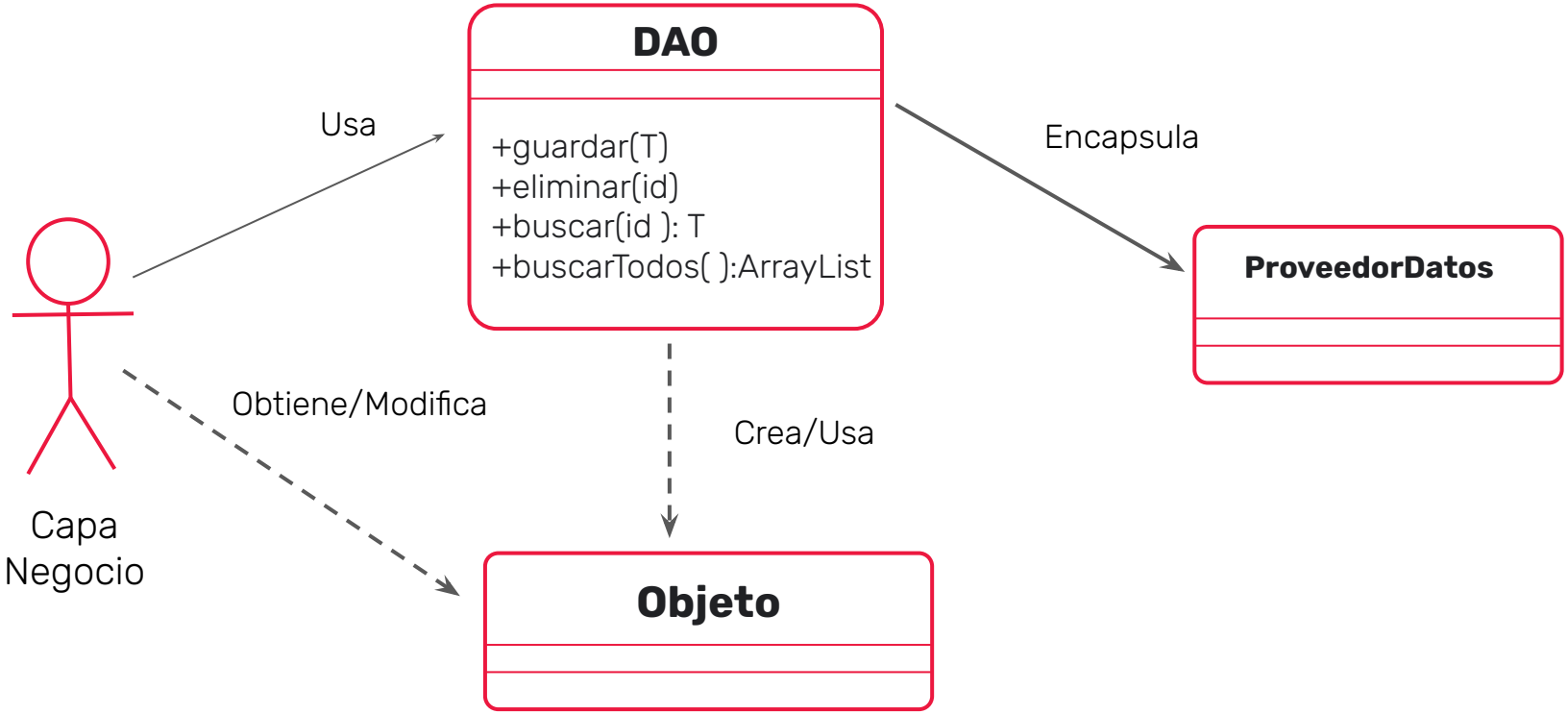
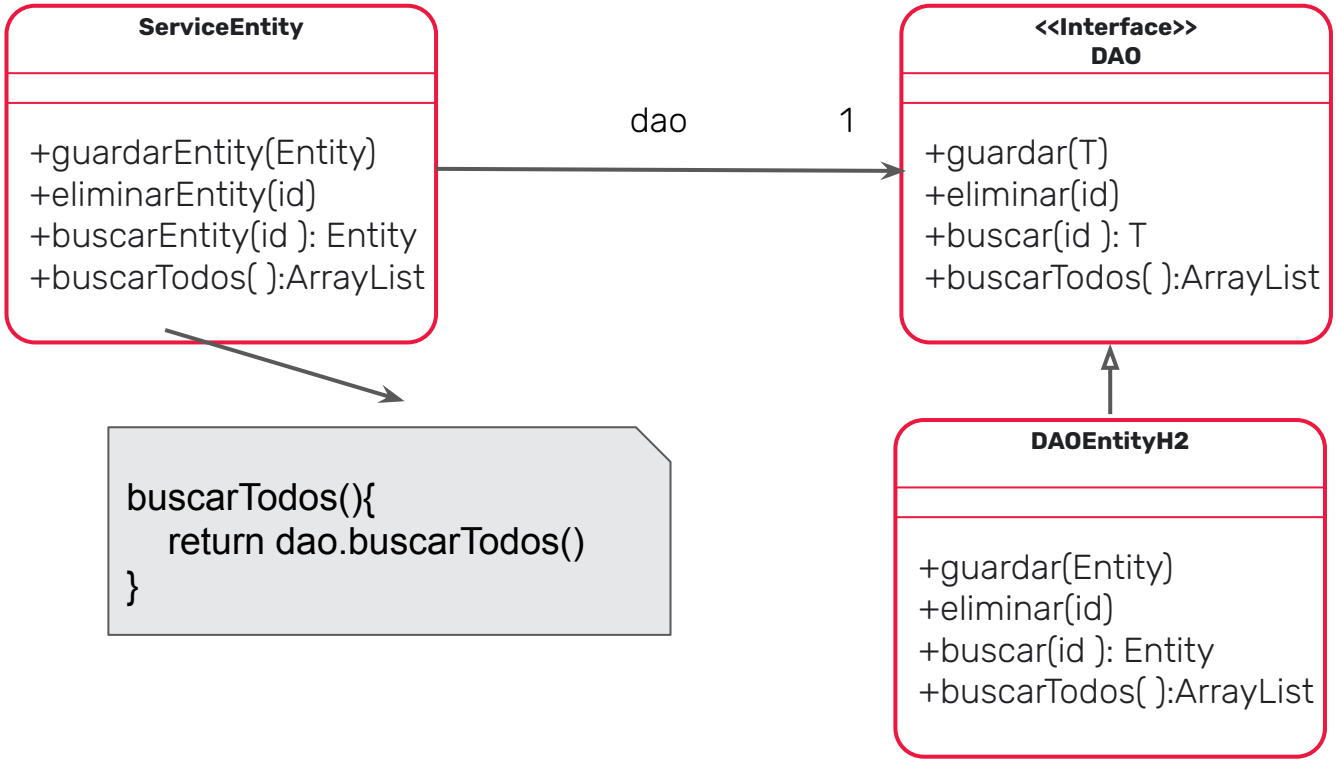


Diagrama de Clases UML



Clases que Intervienen

Entity: Son las clases de negocio Ej Estudiante, Curso, Cuenta, etc.

Clases ServiceEntity: Por cada entidad tendremos una clase de servicio. Las clases de servicio serán utilizadas por la capa de presentación y nos permite desacoplar el acceso a datos de la vista.

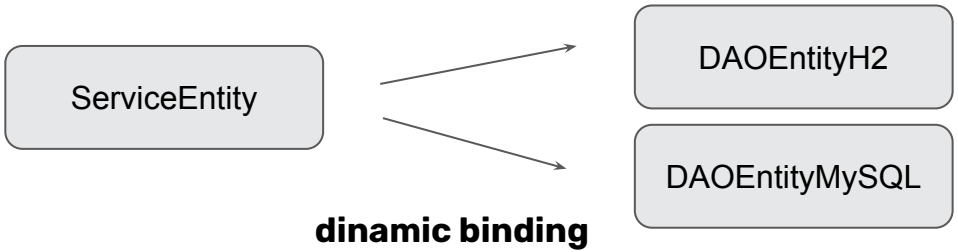
Interface DAO: Es la interface que obligará a las clases DAOs que la implementen a implementar las operaciones que necesitamos realizar sobre la base de datos.

Clases DAO Entity: Implementa la interface DAO y realizará todas estas operaciones en una base de datos en particular ej H2. Si el día de mañana debemos utilizar en lugar de H2, MySQL o MongoDB, crearemos una nueva clase por ejemplo DAOEntityMongoDB que implementará la misma interface pero el código de cada método será diferente al de DAOEntityH2.

Beneficios

Las clases ServiceEntity como indica el diagrama de clases tienen una referencia, es decir, un atributo llamado dao del tipo DAO. Este atributo puede ser cualquier clase que implemente dicha interface y esto nos permite el día de mañana cambiar el mecanismo de persistencia mucho más fácil y de manera dinámica, simplemente apuntando a nuestra clase service al nuevo DAO (dinamic binding).

Te habrás dado cuenta que DAO implementa el patrón Strategy, donde tenemos diferentes estrategias de persistencia que son nuestras clases DAO.



Conclusión

El patrón DAO nos permite abstraer la lógica de negocio de nuestra capa de persistencia, logrando que el sistema sea mucho más fácil de evolucionar.


DigitalHouse>
Coding School