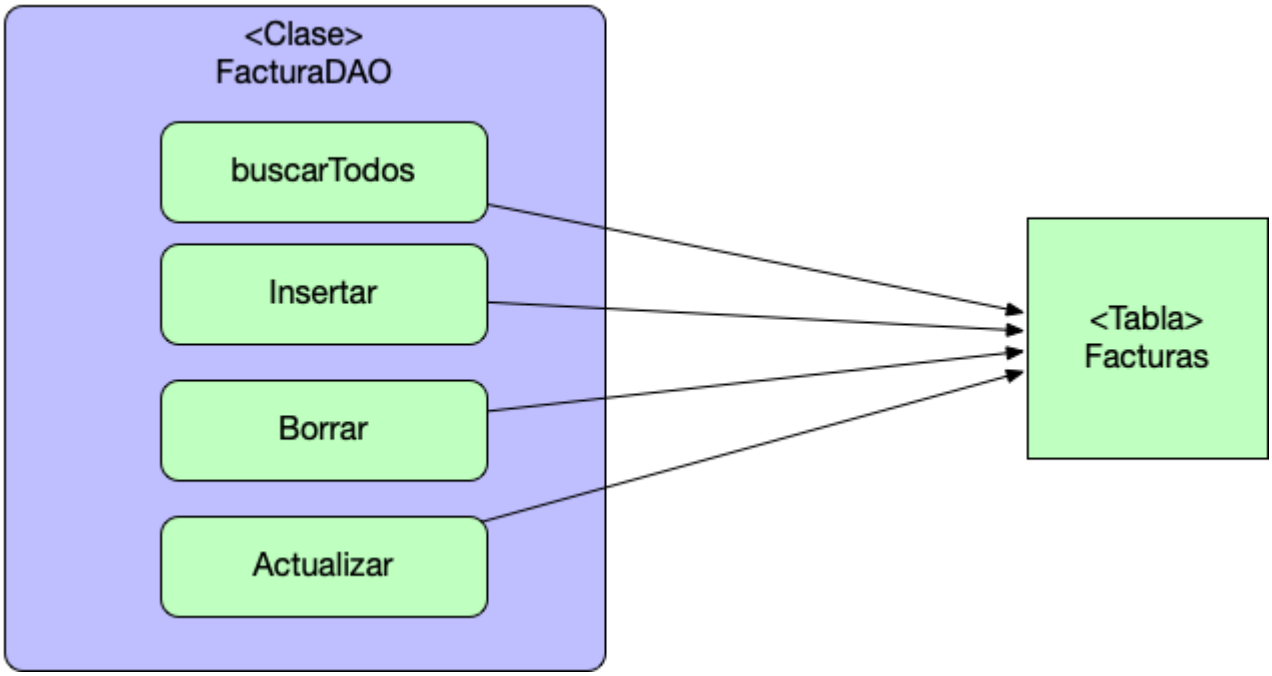
Tabla de Contenidos

- DAO vs Repository
- Repositorios y Flexibilidad
- DAO vs Repository y programación funcional
- Conclusiones
- Otros artículos relacionados

DAO vs Repository . Muchas veces cuando hablamos de estos patrones la mayor para de los desarrolladores consideran que son el mismo patrón y la verdad es que aunque no son el mismo se parecen mucho a nivel de lo que habitualmente queremos construir. Cuando hablamos de DAO o Data Access Object estamos hablando de un patrón de diseño muy clásico en el cual una clase se encarga de las operaciones de persistencia contra una tabla de la base de datos.



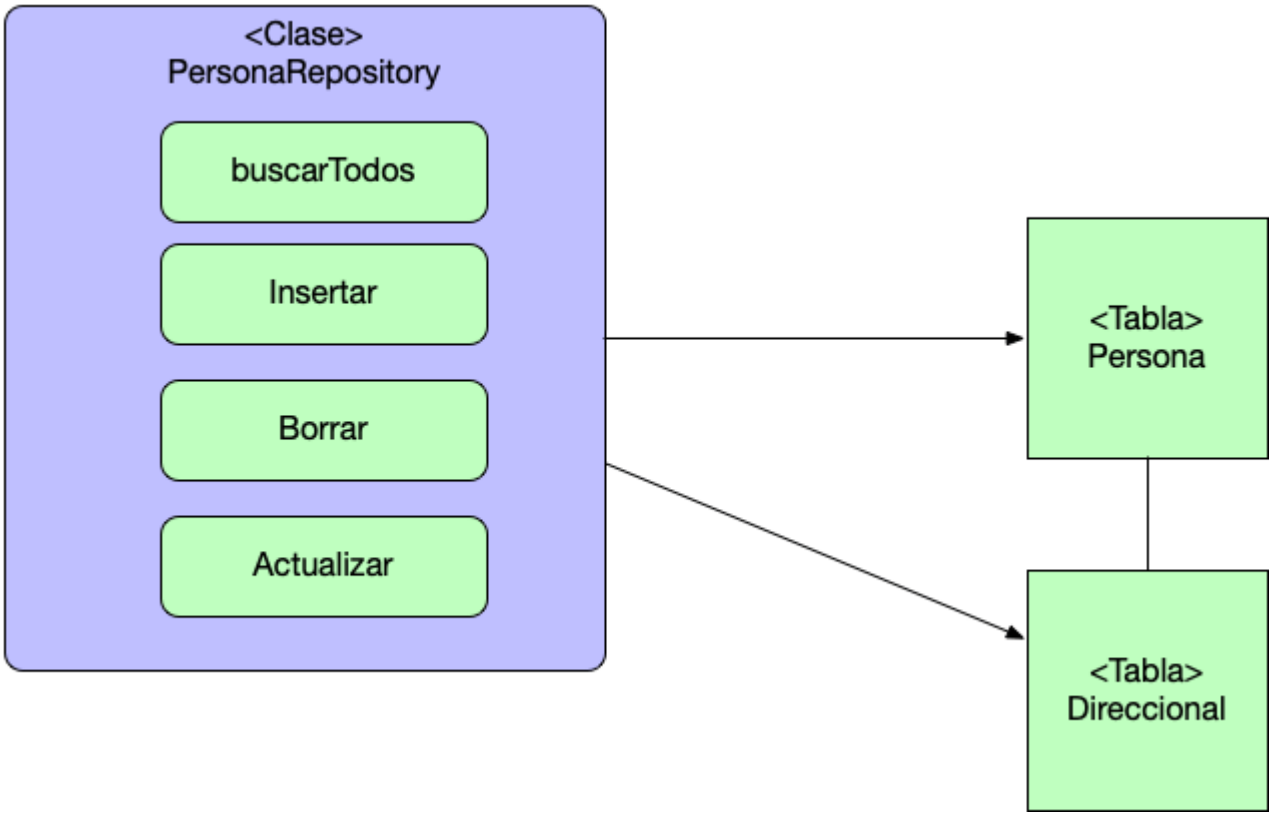
Esa clase normalmente dispone de un conjunto de métodos que se encargan de salvar borrar, editar y actualizar la tabla de la bases de datos .



Este patrón de diseño es muy antiguo y ya era muy común su uso hace 25 años cuando Java estaba empezando a hacerse conocido.

DAO vs Repository

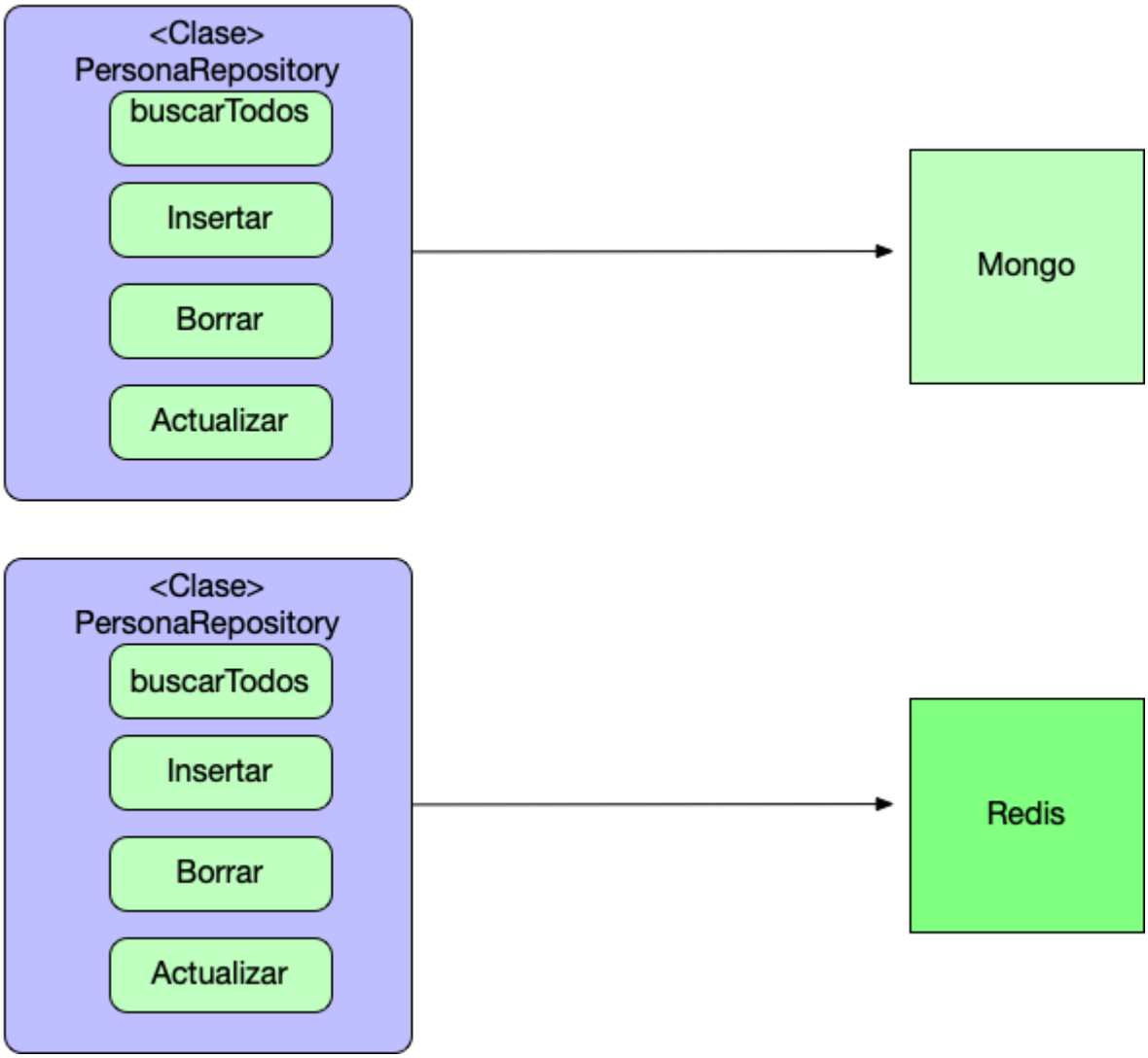
Las cosas han evolucionado mucho en los últimos años y poco a poco el patrón Repository ha ido sustituyendo al patrón DAO. ¿Que diferencias existen? . Probablemente la más clara es que el patrón Repository hace referencia a una clase que se encarga de almacenar un objeto y no especifica con claridad que tipo de persistencia se ha de usar o si el mapeo debe ser uno a uno con la base de datos . Por ejemplo podríamos tener un Repository como el siguiente.



En este caso el Repositorio guarda los datos de Persona pero los separa entre dos tablas y el mapeo no es de uno a uno. Sino de uno a varios. El repositorio aporta mayor flexibilidad y no esta tan fuertemente acoplado a la base de datos. De hecho este es el patrón que usa Spring Framework a nivel de componentes.

Repositorios y Flexibilidad

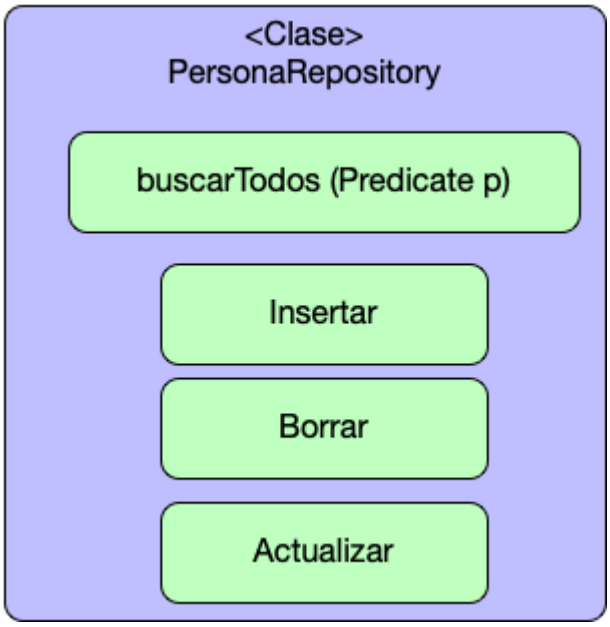
El patrón repository no solo vale para guardar los datos en una base de datos clásica sino que permite también el operar contra otros sistemas de persistencia.



O incluso sistemas que consideremos que son externos y que puedan servir de almacen de datos . Una situación sería llamar a través de un repositorio a un servicio REST y que lo almacene en otra aplicación. Lo importante es que al final la clase de Repositorio se encarga de guardar la información en el destino que se sugiera.

DAO vs Repository y programación funcional

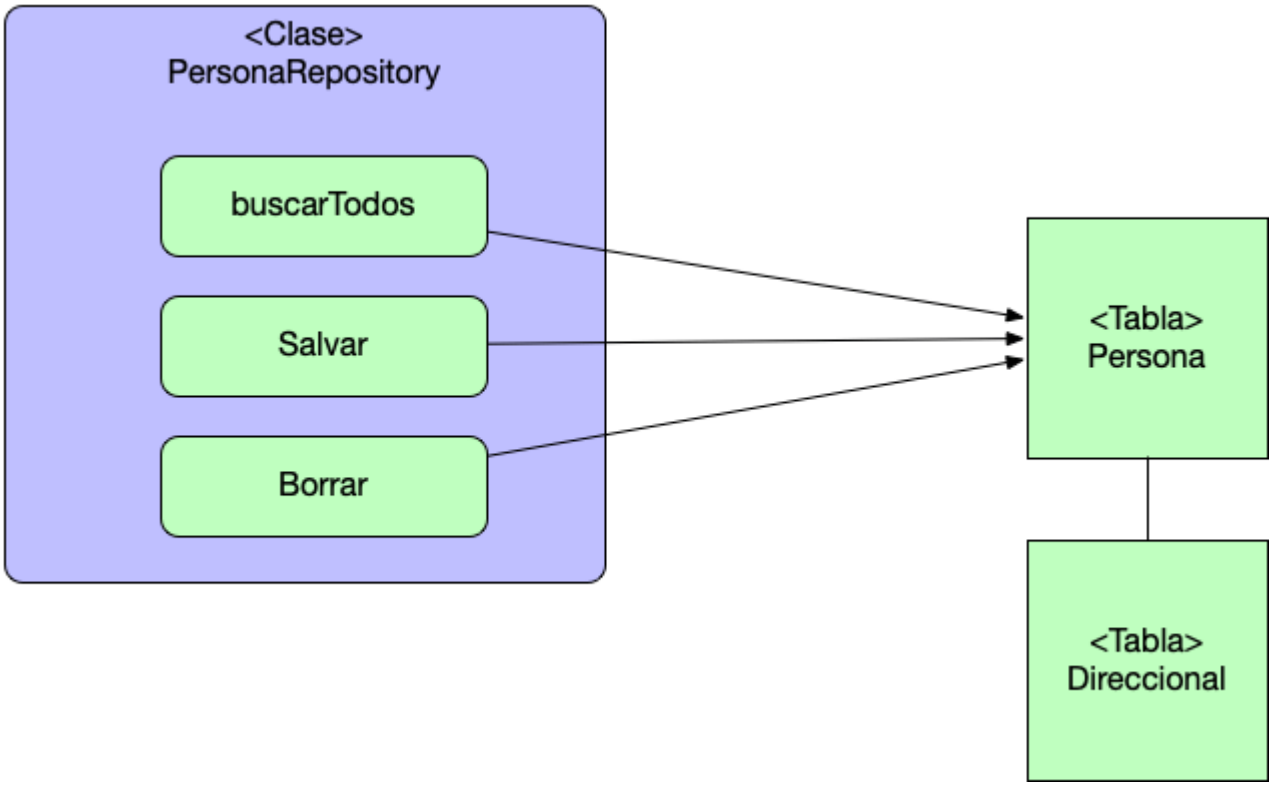
Otro tema muy muy común en los Repositorios es que cuando ya no estamos ligados a una persistencia de base de datos concreta y se trata de una persistencia “generica” es muy común tener métodos de busqueda que admitan Predicates .



Un ejemplo puede ser el caso de Spring Data Specifications que se encarga de usar el api de Criteria para realizar búsquedas a través de Predicados . Por lo tanto en un Repository nos podemos encontrar métodos orientados a programación funcional que aporten flexibilidad y se abstraigan del tipo de persistencia.

Conclusiones

El patrón repositorio esta mas ligado a guardar una colección de objetos sin importar mucho el sistema de persistencia de destino . Mientras que el patrón DAO siempre ha estado mas ligado a guardar datos de una tabla muy concreta de una base de datos SQL. Por ejemplo un DAO contendría un método update e insert y un repositorio solo un método save. Así pues nuestro repositorio final tendría una forma más similar a :



Otros artículos relacionados

- Java Optional Repository y JPA
- Java Generic Repository y JPA
- Webinar: Generic Repository JPA y buenas prácticas
- El patron Repository y la explosión de métodos
- El patrón MVC , arquitectura cliente vs servidor