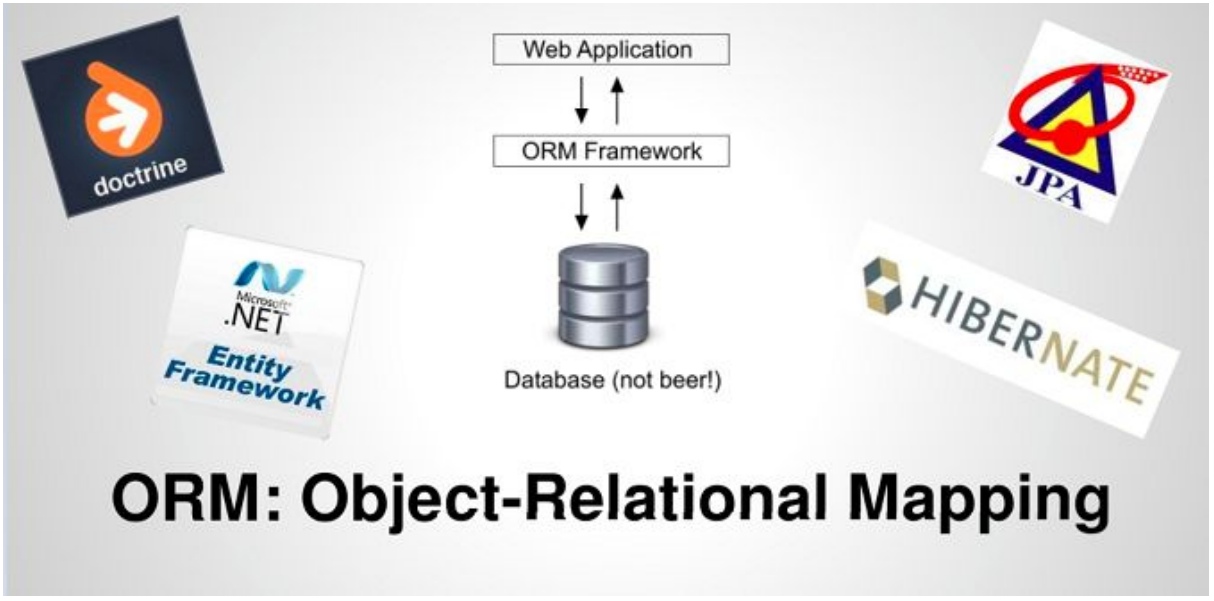
# Persistencia en JAVA

La persistencia es la propiedad de un objeto a través del cual su existencia trasciende en el tiempo y/o espacio. Esto significa que un objeto persistente sigue existiendo después de que ha finalizado el programa que le dio origen y que además puede ser movido de la localidad de memoria en la que fue creado.




## ¿Qué es un ORM?

En el desarrollo de una aplicación suelen estar involucradas dos entidades diferentes, por una parte el código que mueve la aplicación y por otra los datos que se manejan (División en capas).

Los sistemas de Mapeo Objeto-Relacional u ORM ayudan a hacer la transición de lenguaje orientado a objetos a la base de datos que se utilice.

## ¿Qué es un ORM?

Permite ver y modificar los datos almacenados en una base de datos relacional usando el lenguaje de programación elegido y organiza las tablas o procedimientos almacenados en clases, por lo que en vez de utilizar consultas SQL para comunicarse con la base de datos se usan **métodos** y **propiedades** de objetos.

## ¿Qué es un ORM?

La característica más importante de ORM es el **mapeo**.
El mismo se usa para **persistir** objetos almacenados en una base de datos. Un objeto y sus propiedades están típicamente relacionados a una o más tablas y sus campos en la base de datos. El ORM usa la información de esta relación para gestionar el **proceso de conversión** de tablas a objetos y viceversa generando las consultas SQL necesarias para que la base de datos relacional gestione el

## Algunos beneficios de usar un ORM.

- Se encarga de gestionar la conversión de tipos DATE y TIME en forma automática.
- Se encarga de hacer el trabajo “pesado” de asociar los valores de los campos de cada registro de una tabla en los respectivos atributos del objeto.
- Al brindar seguridad en diversos aspectos, previene ataques de inyección SQL.
- Para consultas complejas, muchos ORM brindan

## Algunos beneficios de usar un ORM.

- Generación de código (Entities and Schemas).
- Puede mejorar con creces la velocidad de desarrollo, sobre todo cuando el esquema de la base de datos todavía no se encuentra maduro. Es muy tedioso reescribir una consulta SQL cada vez que se introduce una modificación en dicho esquema, lo cual conlleva a una pérdida de tiempo de producción.
- El uso de caché puede implicar velocidades de acceso

## Algunas desventajas de usar un ORM.

- Se puede llegar a perder la noción acerca de cuánto tiempo se invierte retocando archivos XML o anotaciones, con el objetivo de optimizar la performance del ORM.
- Un mal uso de los sistemas ORMs (debido al desconocimiento del funcionamiento interno) puede reducir la performance de la aplicación en forma drástica. Todo depende del programador y la

## Algunas desventajas de usar un ORM.

- De por sí al agregar una capa más que se encarga abstraer al programador del acceso a los datos agregamos mayor complejidad con lo cual la latencia puede ser mayor.

## ¿Cuándo utilizar un ORM?

- Para aplicaciones transaccionales (CRUD), en las que se realiza una petición, se buscan un conjunto de objetos en la BD, se los procesa y se renderiza una página web. La performance utilizando un ORM pueden ser mucho mejor si dichos objetos se encuentran en la caché del ORM.

# ORM más conocidos: Características y comparaciones. Ejemplos.

## Ejemplos

La mayoría de los ORM se fundan sobre las mismas bases e inclusive suelen trabajar de forma muy similar. Algunos ejemplos de ORMs son:

- Hibernate (Java)
- Entity Framework (.NET)
- Doctrine (PHP)
- SQLAlchemy (Python)
- Active Record (Ruby)

## Algunas Características

- Hibernate (Java) y SQLAlchemy (Python) se basan en el patrón estructural DATA MAPPER.
- Doctrine (PHP) y Active Record (Ruby) se basan en el patrón estructural ActiveRecord.

## Hibernate

Hibernate es una herramienta de Mapeo objeto relacional (ORM) para la plataforma Java (y disponible también para .Net con el nombre de NHibernate) que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los beans de las entidades que permiten establecer estas relaciones.
Hibernate ofrece también un lenguaje de consulta de datos llamado HQL (Hibernate Query Language), al mismo tiempo que una API para construir las consultas programáticamente (conocida como "criteria").