2024

1-3-2024

MotoSave



Victor Lopez

Israel Ramirez

David Gonález Vilches

Índice

[1 Introducción a Motosave 1](#_Toc157438024)

[1.1 Descripción 1](#_Toc157438025)

[1.2 Funcionalidad 1](#_Toc157438026)

[1.3 Alcance 1](#_Toc157438027)

[2 ORM 1](#_Toc157438028)

[2.1 Definición 1](#_Toc157438029)

[2.2 Valoración de ventajas 1](#_Toc157438030)

[2.3 Herramientas ORM más utilizadas 2](#_Toc157438031)

[2.3.1 Doctrine (PHP) 3](#_Toc157438032)

[2.3.2 Hibernate (Java) 3](#_Toc157438033)

[2.3.3 Entity Framework (C#): 5](#_Toc157438034)

[3 Diferencias entre ORM y JPA 6](#_Toc157438035)

[4 Bases de Datos Documentales vs Bases de Datos Relacionales 6](#_Toc157438036)

[4.1 Ventajas 6](#_Toc157438037)

[4.2 Inconvenientes 7](#_Toc157438038)

[5 UML de Motosave 7](#_Toc157438039)

[6 Entidad-relación de la Base de Datos de Motosave 7](#_Toc157438040)

[7 Uso y funcionamiento de Motosave 8](#_Toc157438041)

[7.1 Uso general 8](#_Toc157438042)

[7.2 Estructura de clases 8](#_Toc157438043)

[7.2.1 Clases (por completar) 8](#_Toc157438044)

[7.3 Funcionamiento detallado 8](#_Toc157438045)

[7.3.1 Método no se que (que sea complejo) descripción detallada y explicación. 8](#_Toc157438046)

[8 Elementos añadidos a los contenidos mínimos 8](#_Toc157438047)

[8.1 Diagrama de flujo. 8](#_Toc157438048)

[8.2 Librerías y frameworks 8](#_Toc157438049)

[8.3 Otros (por si hubiera) 8](#_Toc157438050)

[8.4 Seguridad 8](#_Toc157438051)

[8.4.1 Seguridad de datos 8](#_Toc157438052)

# Introducción a Motosave

## Descripción

## Funcionalidad

## Alcance

# ORM

## Definición

Las herramientas ORM (Object Relational Mapping) son modelos de programación que desempeñan un papel fundamental al simplificar y automatizar las tareas relacionadas con el manejo de datos en aplicaciones. La función principal de un ORM es transformar las estructuras de datos de una base de datos relacional en objetos que pueden ser utilizados en el código de programación. En otras palabras, actúan como un puente entre el mundo de la programación orientada a objetos y las bases de datos relacionales.

Estas herramientas permiten a los desarrolladores trabajar con objetos en el código, abstrayéndolos de las complejidades de la manipulación directa de las tablas de la base de datos. Al utilizar un ORM, las operaciones de acceso a datos, como la creación, lectura, actualización y eliminación (CRUD), se simplifican, lo que ahorra tiempo y esfuerzo en el desarrollo de aplicaciones.

## Valoración de ventajas

**Simplicidad en el Desarrollo:**

Al funcionar como una capa intermedia separada de la base de datos, las herramientas ORM permiten a los desarrolladores centrarse exclusivamente en el desarrollo de la aplicación, simplificando el proceso.

**Facilita Migraciones:**

Simplifica las migraciones, ya que cambiar la base de datos no requiere reconstruir todas las consultas. La aplicación puede migrarse a otra base de datos sin problemas.

Las herramientas ORM facilitan la portabilidad del código entre diferentes sistemas de gestión de bases de datos, ya que el código interactúa con objetos en lugar de consultas SQL específicas de una base de datos.

**Unificación del Lenguaje:**

Elimina la necesidad de que los programadores conozcan un lenguaje específico para cada base de datos, ya que el ORM unifica el lenguaje de acceso a los datos.

**Velocidad en Tareas Básicas:**

Proporciona mayor velocidad en tareas básicas relacionadas con el acceso a los datos, mejorando la eficiencia en operaciones comunes.

**Código Más Legible y Conciso:**

El código generado por las herramientas ORM suele ser más legible y requerir menos líneas en comparación con la escritura manual de consultas SQL.

**Mayor Seguridad:**

Al ser una capa independiente de los datos, las herramientas ORM contribuyen a la seguridad al protegerlos de ciberataques al operar en un nivel separado.

**Evita la Escritura Manual de Consultas SQL:**

Elimina la necesidad de escribir a mano las consultas SQL necesarias, simplificando el desarrollo y reduciendo posibles errores.

**Facilita el Trabajo con Operaciones Básicas (CRUD):**

Simplifica el manejo de operaciones CRUD (Create, Read, Update, Delete), facilitando la interacción con la base de datos.

**Guarda y Carga Información Automáticamente:**

Una de las ventajas más atractivas es que un ORM guarda y carga automáticamente toda la información de una base de datos relacional.

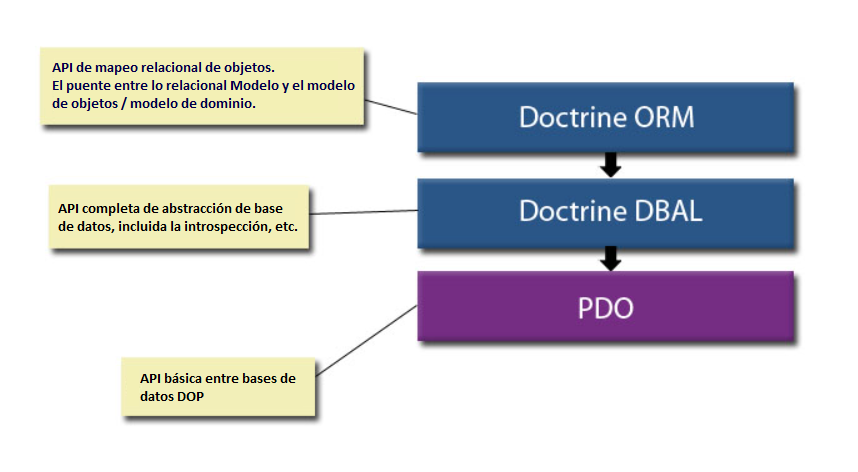
## Herramientas ORM más utilizadas

### Doctrine (PHP)



**Descripción**: Doctrine es una herramienta ORM para PHP que destaca por su código de alta calidad y estabilidad. Ofrece un conjunto de funciones poderosas y flexibles para la manipulación de datos en bases de datos relacionales.

**Características Clave**:

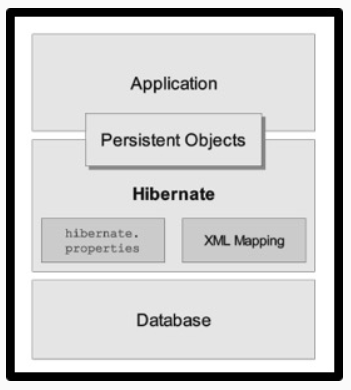
* Consultas flexibles y potentes.
* Capacidad para trabajar con modelos de datos complejos.
* Facilita la gestión de esquemas de bases de datos.

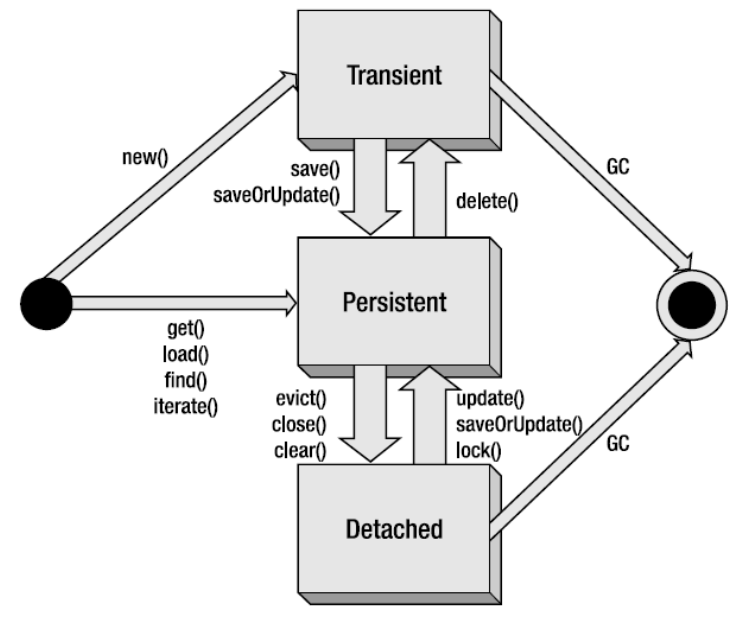
### Hibernate (Java)



**Descripción**: Hibernate es una herramienta ORM ampliamente utilizada en el entorno Java. Utiliza XML o anotaciones para mapear objetos a las tablas de la base de datos y se destaca por su capacidad para gestionar la persistencia de datos de manera eficiente.

**Características Clave:**

* Mapeo objeto-relacional mediante XML o anotaciones.
* Integración con Java Persistence API (JPA).
* Gestión eficiente de la persistencia de objetos en bases de datos relacionales.



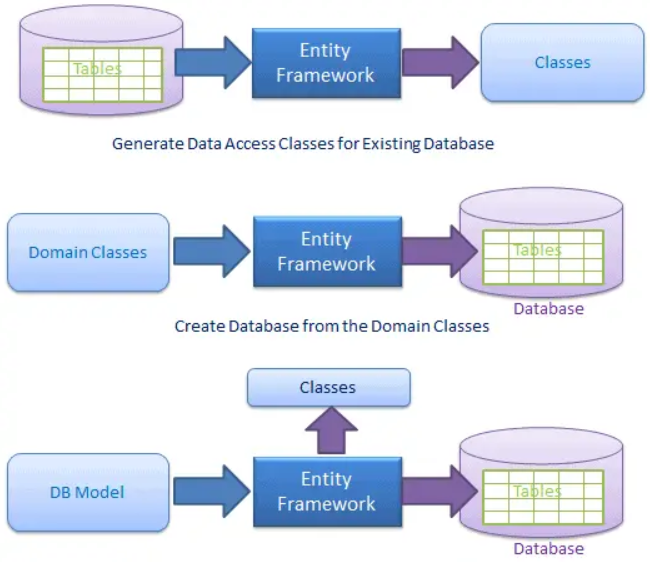
### Entity Framework (C#):



**Descripción**: Entity Framework es la herramienta ORM de Microsoft para el entorno .NET. Permite a los desarrolladores trabajar con datos relacionales utilizando objetos específicos de C#.

**Características Clave:**

* Modelado de datos a través de Entity Framework Designer.
* Admite la creación de nuevos modelos o la vinculación a bases de datos existentes.
* Facilita operaciones CRUD y manipulación de esquemas de bases de datos.



# Diferencias entre ORM y JPA

ORM es el concepto general de mapear objetos a tablas de base de datos, mientras que JPA es una especificación en Java que sigue el enfoque ORM. JPA proporciona estándares para el mapeo objeto-relacional en Java, y Hibernate es una de las implementaciones más populares de JPA. En otras palabras, JPA es un conjunto de especificaciones y estándares, mientras que Hibernate es una implementación concreta de esas especificaciones.

# Bases de Datos Documentales vs Bases de Datos Relacionales

**Las bases de datos documentales:** Son un tipo de sistema de gestión de bases de datos (DBMS) que almacenan, recuperan y gestionan datos en formato de documentos, generalmente en formato JSON o BSON. Cada documento puede contener datos semiestructurados o no estructurados.

**Las bases de datos relacionales**: Son sistemas de gestión de bases de datos que organizan los datos en tablas relacionadas entre sí. Utilizan un esquema predefinido y un lenguaje estructurado de consulta (SQL) para acceder y manipular los datos.

## Ventajas

|  |  |
| --- | --- |
| **Bases de Datos Documentales** | **Bases de Datos Relacionales** |
| **Flexibilidad en la Estructura de Datos**:  Las bases de datos documentales permiten almacenar datos no estructurados, como documentos JSON o BSON, lo que brinda flexibilidad en la estructura de los datos. | **Estructura de Datos Rigurosa**:  Ofrecen una estructura de datos rigurosa con relaciones definidas entre tablas, lo que facilita la representación de datos complejos y relaciones entre entidades. |
| **Escalabilidad Horizonta**l:  Son más adecuadas para escalar horizontalmente, ya que permiten agregar nodos de forma eficiente, facilitando la gestión de grandes volúmenes de datos distribuidos | **Soporte para Transacciones ACID**:  Las bases de datos relacionales garantizan transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad), proporcionando coherencia y seguridad en operaciones críticas. |
| **Manejo de Datos No Estructurados**:  Eficientes para almacenar y recuperar datos no estructurados como documentos, imágenes y videos, comunes en entornos web y aplicaciones modernas. | **Optimización para Consultas Relacionales**:  Son altamente eficientes para consultas que involucran relaciones complejas, ya que están optimizadas para operaciones join y consultas relacionadas. |
| **Rendimiento en Operaciones de Lectura y Escritura**:  Suelen ser eficientes en operaciones de lectura y escritura, especialmente cuando se trata de documentos completos. | **Madurez y Experiencia**:  Las bases de datos relacionales tienen décadas de experiencia y una amplia adopción en la industria, con una abundancia de herramientas y recursos disponibles. |

## Inconvenientes

|  |  |
| --- | --- |
| **Bases de Datos Documentales** | **Bases de Datos Relacionales** |
| **Complejidad en Consultas Relacionadas**:  Pueden ser menos eficientes para consultas que involucran relaciones complejas entre diferentes tipos de documentos, ya que no están optimizadas para operaciones join. | **Menos Flexibilidad Estructural**:  Pueden resultar menos flexibles al tratar con datos no estructurados o con estructuras de datos que pueden cambiar con el tiempo. |
| **Consistencia y Transacciones**:  Algunas bases de datos documentales sacrifican la consistencia en favor de la disponibilidad y la tolerancia a la partición, lo que puede afectar la integridad de los datos en ciertos escenarios. | **Dificultades en Escalabilidad Horizontal**:  Escalar horizontalmente puede ser más desafiante, ya que la coherencia de los datos en múltiples nodos puede ser difícil de mantener. |
| **Mayor Overhead en Operaciones No Relacionales**:  Para operaciones que involucran datos no relacionales o estructuras más complejas, las bases de datos relacionales pueden tener un mayor overhead en comparación con las bases de datos documentales. |

# UML de Motosave

# Entidad-relación de la Base de Datos de Motosave

# Uso y funcionamiento de Motosave

## Uso general

## Estructura de clases

### Clases (por completar)

## Funcionamiento detallado

### Método no se que (que sea complejo) descripción detallada y explicación.

# Elementos añadidos a los contenidos mínimos

## Diagrama de flujo.

## Librerías y frameworks

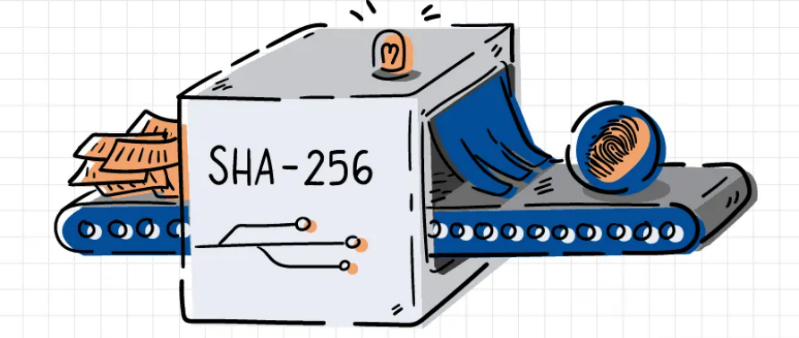
## Otros (por si hubiera)

## Seguridad

### Seguridad de datos

**SHA256**

SHA256 se deriva de las palabras ***"Secure Hash Algorithm 256 bits"***, lo que significa que es un algoritmo de hash seguro de 256 bits.



Su tarea principal es tomar una cantidad arbitraria de información, ya sea un mensaje de texto, una contraseña o incluso un archivo completo, y convertirla en una secuencia de 256 bits, un conjunto de números y letras aparentemente aleatorios.

Esta secuencia conocida como ***"hash"***, es única para cada conjunto de datos que se le pasa al algoritmo, lo que significa que si se cambia una sola letra en el mensaje original, el hash resultante será completamente diferente. Es como la ***"huella digital"*** de los datos, y es imposible que dos conjuntos de datos diferentes produzcan el mismo hash.

**¿Cómo funciona el algoritmo SHA256?**[**​**](https://cuevaneander.tech/blog/seguridad-informatica-algoritmo-sha256/#c%C3%B3mo-funciona-el-algoritmo-sha256)

Ahora que ya tenemos un poco más de contexto general sobre lo que es este algoritmo, es hora de profundizar un poco más en cómo funciona. Como se mencionó anteriormente, el algoritmo SHA256 toma una cantidad arbitraria de datos y los convierte en una secuencia de 256 bits. Pero, ¿cómo lo hace exactamente? ¿Cómo puede tomar cualquier cantidad de datos y producir un hash único para cada uno de ellos?

**Mensaje “M**[**​**](https://cuevaneander.tech/blog/seguridad-informatica-algoritmo-sha256/#mensaje-m)”

El mensaje "M" o "datos de entrada" es la información que la función SHA256 procesará para calcular el hash (también conocido como "digest"). M puede tener una longitud variable, lo que significa que puede consistir en un archivo de datos o una cadena de texto, como por ejemplo "Hola mundo", incluso puede ser una cadena de texto vacía.

El resultado de este proceso es una cadena ω de 256 bits de longitud, que se expresa en un formato de 64 caracteres hexadecimales(base16). Este proceso de transformación garantiza que, independientemente de la forma o contenido de la entrada, obtendremos un hash de 2556 bits que actúa como una huella digital única de los datos de entrada.

La función SHA256 se puede representar como SHA256(M) = ω, donde M es el mensaje de entrada y ω es el hash de salida.



Aplicando esa función a una cadena de texto como "Hola mundo" obtendríamos el siguiente resultado:

**"Hola mundo" | ca8f60b2cc7f05837d98b208b57fb6481553fc5f1219d59618fd025002a66f5c**

La cadena hexadecimal generada se obtiene mediante una serie de cálculos en los que se emplean entre otros datos los bits de entrada, es decir, para un mismo mensaje M siempre se obtendrá el mismo hash ω. Por el contrario, si modifica un solo bit, ya sea cambiando o agregando un caracter (espacios y saltos de línea incluidos), el hash resultante será completamente diferente.

1. echo -n "abc" | sha256sum  
   ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb410ff61f20015ad
2. echo -n "Abc" | sha256sum  
   06d90109c8cce34ec0c776950465421e176f08b831a938b3c6e76cb7bee8790b

De este modo, se garantiza la integridad de los datos. Si alguien modifica el mensaje, el hash resultante será diferente, por lo que no coincidirá con el hash original. Esto permite detectar si los datos han sido modificados.