ORM

Lenguaje	Herramienta que Implementa ORM	ORM	Framework Asociado	Anotaciones de Interés
Java	JPA	Hibernate	Spring Framework (Spring Boot)	Soporte completo para OO; Hibernate permite extensiones avanzadas.
Python	SQLAlchemy	Django ORM	Flask/Django	SQLAlchemy es flexible; Django ORM integrado con Django.
Ruby	ActiveRecord	ActiveRecord	Ruby on Rails	Estándar integrado con Rails.
C#	ADO.NET Entity Framework	Entity Framework	.NET (ASP.NET Core)	ORM oficial de Microsoft, integración perfecta.
C++	-	ODB	Ninguno predominante	Requiere herramientas externas.
JavaScript	Sequelize	Sequelize	Express (Node.js)	Popular para aplicaciones Node.js.
PHP	Laravel	Eloquent	Laravel	Altamente integrado con Laravel.
Kotlin	Hibernate, Exposed	Hibernate, Exposed	Ktor/Spring Boot	Exposed es más idiomático; Hibernate para JPA.

Ejemplos de uso de los diferentes ORM:

Java (Hibernate)

- Definir una clase Java que represente una tabla en la base de datos. Por ejemplo, una clase `User` con campos como `id` y `name`.
- Configurar Hibernate para conectar con la base de datos.
- Usar una interfaz de repositorio para realizar operaciones como insertar, actualizar o consultar usuarios.

Python (SQLAlchemy o Django ORM)

- Definir una clase Python que actúe como modelo y esté vinculada a una tabla en la base de datos, como una tabla `users` con columnas `id` y `name`.
- Configurar la conexión a la base de datos usando SQLAlchemy o Django ORM.
- Crear instancias del modelo para agregar datos y luego guardar los cambios en la base de datos.

Ruby (ActiveRecord)

- Crear un modelo Ruby llamado `User` que corresponda a una tabla en la base de datos.
- Utilizar métodos de ActiveRecord para insertar nuevos registros, como creando un nuevo usuario con el nombre "Alice".
- Realizar consultas como buscar todos los usuarios o actualizar sus datos directamente desde el modelo.

C# (Entity Framework)

- Definir una clase que represente una tabla, como una clase `User` con propiedades `Id` y `Name`.
- Configurar el contexto de base de datos para conectar con una base de datos específica.
- Usar el contexto para realizar operaciones como agregar nuevos usuarios o actualizar registros existentes.

C++ (ODB)

- Crear una clase para definir la estructura de una tabla, como una clase `User` que tenga un identificador y un nombre.
- Usar ODB para generar automáticamente el código que conecta esta clase con la base de datos.
- Insertar nuevos usuarios y realizar consultas a través del API generado por ODB.

JavaScript (Sequelize)

- Definir un modelo para una tabla en la base de datos, como una tabla `users` con columnas para `id` y `name`.
- Configurar Sequelize para conectarse a una base de datos específica, como SQLite o PostgreSQL.
- Utilizar métodos del modelo para insertar un usuario llamado "Alice" o consultar los registros existentes.

PHP (Eloquent)

- Crear un modelo que represente una tabla en la base de datos, como un modelo `User` para la tabla `users`.

- Usar métodos de Eloquent para agregar registros, como crear un usuario con el nombre "Alice".
- Realizar consultas, como obtener todos los usuarios o actualizar los nombres de algunos registros.

Kotlin (Exposed o Hibernate)

- Definir un esquema para la tabla usando un objeto en Kotlin, como `Users` con columnas `id` y `name`.
- Establecer una conexión a la base de datos utilizando Exposed o Hibernate.
- Insertar nuevos datos en la tabla `users` o realizar consultas para recuperar y actualizar registros.