RESUMO ALLEPH:

ORM é você pega seus dados e transforma em outros, exemplo: no seu código você pega seus dados orientado a objetos e ele transforma no modelo relacional(nosso banco de dados relacional) vice e versa.

Padrões para usar o ORM

ActiveRecord

Temos um código mais simples, tudo é feito de forma dinâmica, ele faz o “trabalho” pra você

Data Mapper

Imagina que temos uma classe padrão e essa classe vai ser mapeada para um banco de dados.

https://www.youtube.com/watch?time\_continue=57&v=NvdGYdOAt5Q&embeds\_referring\_euri=https%3A%2F%2Fcursos.alura.com.br%2F&source\_ve\_path=MTM5MTE3LDEzOTExNywxMzkxMTcsMTM5MTE3LDI4NjY2&feature=emb\_logo

Ao trabalhar com programação Web, frequentemente é necessário utilizar bancos de dados para armazenar os dados da aplicação desenvolvida, sejam estes dados de usuários, endereços ou quaisquer dados que precisem ser guardados para o funcionamento do programa.

Sendo assim, é muito interessante que as linguagens de programação que são voltadas para Web tenham maneiras de se conectar com o banco de dados para que seja possível cadastrar, deletar, consultar e atualizar registros guardados no banco. Ou seja, que possibilitem fazer um **CRUD** diretamente a partir da linguagem.

Com isso, a maioria das linguagens passou a adotar soluções nativas para lidar com conexões em bancos de dados, como é o caso do **JDBC** (Java DataBase Connector) no Java. No entanto, a utilização desse tipo de solução começou a ser um pouco maçante, uma vez que a configuração não era simples, nem pequena e tinha de ser feita repetidas vezes de forma idêntica.

Por este motivo, passaram a aparecer alternativas ao uso dessas soluções nativas, que ao invés de apresentarem uma conexão "crua" com o banco de dados, isso é, que exija a escrita de consultas e comandos SQL na mão, passaram a apresentar código encapsulado que já apresenta grande parte do que fazemos no dia a dia utilizando banco de dados de maneira pronta, com funções disponibilizadas na própria linguagem, sem a necessidade de misturá-la com SQL a cada nova conexão com o banco de dados. A esse tipo de ferramenta, damos o nome de **ORM**, em português: Mapeamento Objeto-Relacional:

Dentro do ecossistema Java, a ferramenta ORM mais conhecida é a JPA, ou Java Persistance API, que é uma especificação criada com o intuito de simplificar a conexão de banco de dados em aplicações Java, a partir da definição de uma interface comum a ser utilizada para persistência de dados na linguagem.

O JPA surge, justamente com o objetivo de oferecer uma maneira mais fácil e padronizada e mapear objetos e tabelas de um banco de dados relacional, e fornecer uma API para realizar as operações **CRUD**, utilizando o Java.

Para entender melhor seu funcionamento, é importante familiarizar-se com alguns conceitos-chave da especificação, como:

* Entidade: é uma classe Java criada para representar um objeto armazenado em um banco de dados. Cada entidade é mapeada para uma tabela do banco.
* EntityManager: é a interface central do JPA, que é usada para realizar as operações de persistência, ou seja, o CRUD. É a EntityManager que gerencia o ciclo de vida das entidades dentro de uma aplicação Java.
* JPQL: é a linguagem de consulta da JPA, que permite escrever consultas customizadas se aproveitando da orientação a objetos, ou seja, é como se fosse uma mistura de Java e SQL, ou uma linguagem SQL orientada a objetos. A utilização da JPQL é interessante pois ela encapsula a chamada real para o banco de dados, e permite inclusive a troca do banco de dados da aplicação sem a necessidade de alteração no código-fonte.

É importante observar, no entanto, que a JPA é apenas uma especificação entregue pela linguagem e não uma implementação real de conexão a banco de dados, isto é, a JPA fornece uma interface que deve ser seguida para conexões de banco de dados utilizando ORMs, de maneira que esteja encapsulada a conexão com o banco de dados para reduzir o acoplamento do código.

Existem, portanto, várias implementações da JPA, que fazem conexões com bancos de dados baseando-se na especificação e que tornam a vida da pessoa desenvolvedora mais fácil, pois basta se preocupar em entender a JPA, e não necessariamente como funciona cada detalhe de todas as suas implementações.

Uma das implementações mais populares da JPA é o **Hibernate**, que será utilizada durante o decorrer deste curso. O Hibernate fornece uma biblioteca baseada no JPA e que permite o acesso ao banco de dados de forma simplificada, o que melhora a produtividade no ciclo de desenvolvimento, além de oferecer recursos avançados para modelagem de dados complexos e gerenciamento de persistência, como o suporte à herança e associações e a integração com o projeto Spring.

Além disso, o Hibernate é distribuído em Open Source, e conta com uma grande comunidade de pessoas desenvolvedoras que mantém a ferramenta, além de muita [documentação](https://hibernate.org/orm/documentation/6.3/) disponível para consulta.