Certified Tech Developer
The Ultimate Degree

Camada de dados (Data layer)

Nesta aula, vamos nos concentrar, entre outras coisas, na persistência de dados e como se conectar e trabalhar com um banco de dados relacional.

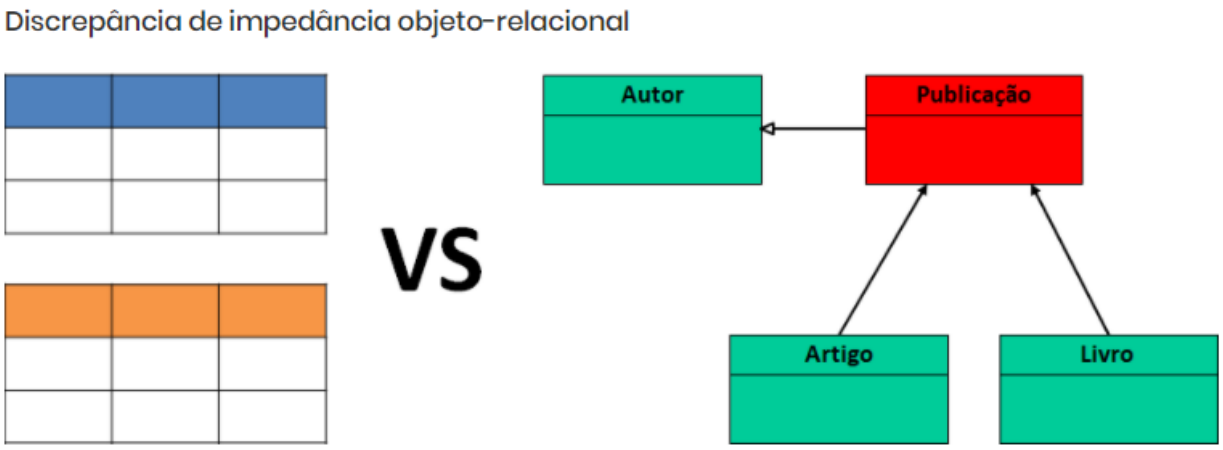
Banco de dados relacionais

É um tipo de banco de dados que segue o modelo relacional. A linguagem mais comum para a construção de consultas em banco de dados relacionais é o SQL (Structured Query Language), um padrão implementado pelas principais engines ou sistemas integrados de banco de dados relacionais. Os bancos de dados relacionais (SQL) seguem as garantias da sigla em inglês ACID: atomicity (atomicidade), consistency (consistência), isolation (isolamento) e durability (durabilidade).

Vamos conhecer suas características:

Essas são excelentes características, mas para garanti-los, esses bancos de dados não podem lidar adequadamente com escalabilidade horizontal (múltiplos nós distribuídos), o que significa que eles não escalam tão bem.

Um sistema de software usado para manter bancos de dados relacionais é um **relational database management system (RDBMS)** ou sistema de gerenciamento de banco de dados relacional. Praticamente, todos os sistemas de banco de dados relacionais usam SQL para consultar e manter as bases de dados.



É apenas uma forma elegante de dizer que modelos de objetos e modelos relacionais não funcionam muito bem juntos. Os **RDBMS** representam dados em um formato de tabela (como uma planilha), enquanto nas linguagens orientadas a objetos, como o **Java**, os representam como um gráfico interconectado de objetos. Carregar e armazenar gráficos de objetos usando um banco de dados relacional com tabelas nos expõe a 5 problemas de discrepância.

Assim, podemos citar que o ORM é um mecanismo que nos permite interagir com nosso banco de dados sem a necessidade de conhecer SQL. Os ORMs são responsáveis por traduzir nossa instrução na linguagem de programação que estamos utilizando em uma sentença SQL que o gerenciador de banco de dados possa entender.

Existem muitos **ORMs** para Java: EJB, JDO, **JPA**. Embora sejam especificações, o **Hibernate**, é uma **implementação**.

Não se preocupe! Alguns conceitos podem ser desconhecidos para você, mas os veremos mais adiante.

Vantagens e Desvantagens

Agora vamos ver quais são as vantagens e desvantagens do ORM.

O que é JPA?

Logo que iniciamos esta aula, mencionamos e dissemos que em breve entraremos neste conceito. E chegou o momento!

JPA ou Java Persistence API é uma coleção de classes e métodos que permitem armazenar de forma persistente grandes quantidades de dados em banco de dados. **Não é um framework**, mas um conjunto de **conceitos que podem ser implementados** em qualquer ferramenta ou **framework**.

JPA busca solucionar o problema levantado ao tentar traduzir um modelo orientado a objetos em um modelo relacional. Por sua vez, permite armazenar entidades de negócio como entidades relacionais.

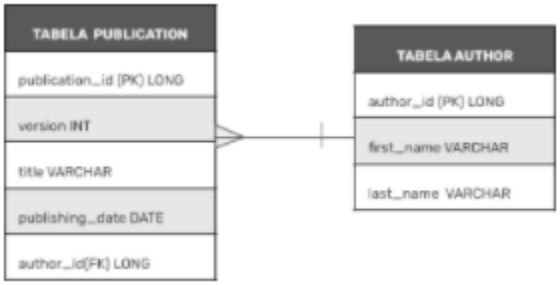
A especificação JPA permite que o desenvolvedor defina quais objetos devem ser persistidos e como esses objetos devem ser persistidos nas aplicações Java.

Ao usar o JPA, um **mapa** do banco de dados é criado para o modelo de objetos da aplicação. A **conexão** entre o **banco de dados relacional e a aplicação** é gerenciada pelo **JDBC** (Java database connectivity API).

JPA é uma API open source e **existem diferentes provedores que a implementam**, sendo usados em produtos como o **Hibernate** e o **Spring Data JPA**.

Exemplo: Mapeando o DER para classes Java

Ao mapear, devemos levar em consideração **os tipos de dados e seus relacionamentos**. Por padrão, o **nome do objeto** persistente se torna o **nome da tabela** e os **atributos** são convertidos em **colunas**. Depois de criar a tabela, cada registro corresponde a um objeto.



author

```
@Entity
public class Author{

@Id
@GeneratedValue(strategy=generationType.SEQUENCE)
private Long id;

@Column(name = “first_name”)
private String firstName;

@Column(name = “last_name”)
private String lastName;

@OnetoMany(mappedBy = “author”)
private Set  publications = HashSet();

}
```

Publication

```
@Entity
public abstract class Publication {

@Id
@GeneratedValue(strategy=generationType.SEQUENCE)
private Long id;
private int version;
private String title;

@ManytoOne(fetch = FetchType.LAZY)
private Author author;


@Column(name = “publishing_date”)
private LocalDate publishingDate;

}
```

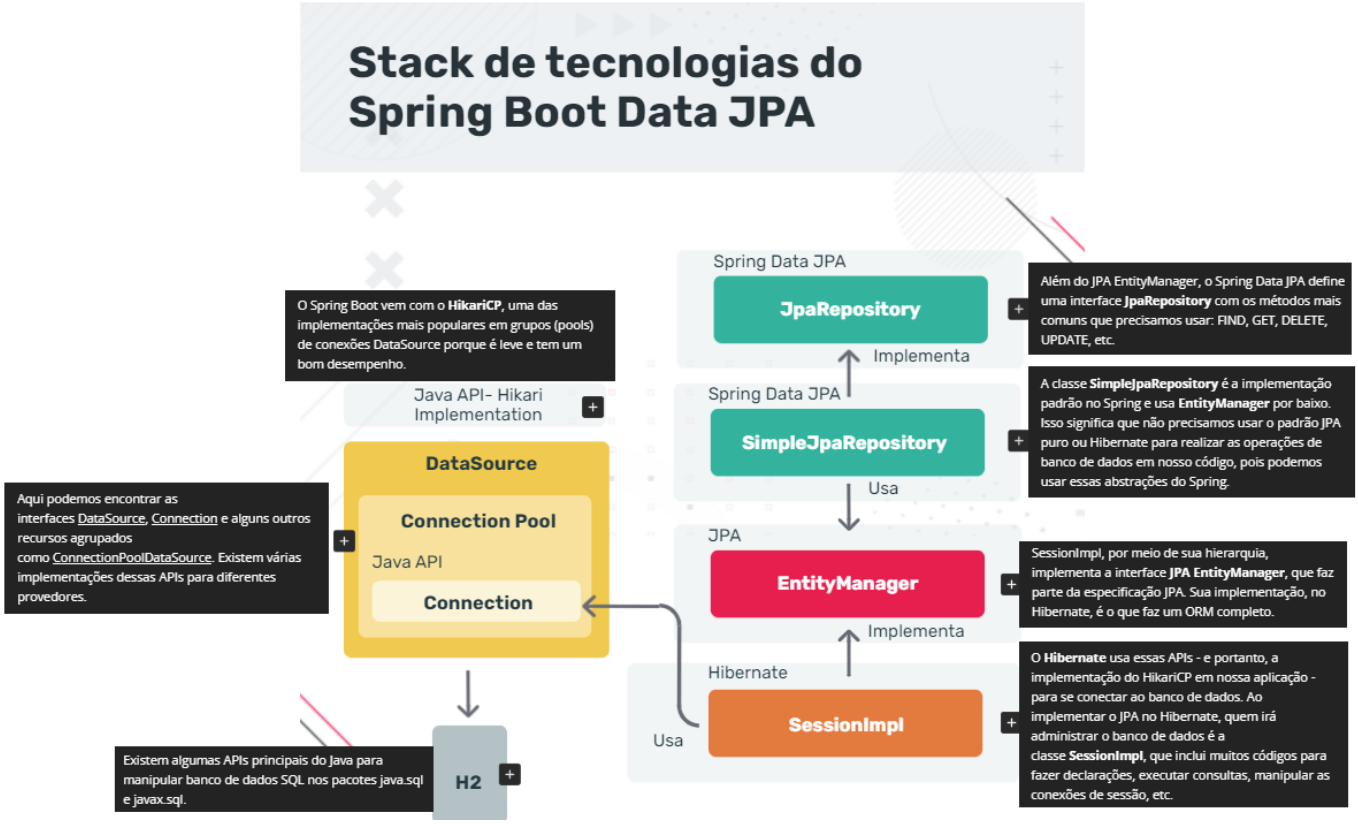
Spring Data

O **Spring Framework** disponibiliza diversos módulos para trabalhar com banco de dados, agrupados por família **Spring Data**: JDBC, Cassandra, Hadoop, Elasticsearch, entre outros. Um deles é o **Spring Data JPA**, que abstrai o acesso ao banco de dados usando a **API de persistência do Java** no modelo de programação baseada em Spring.

O **Spring Boot** dá um passo adicional com um inicializador dedicado que usa configuração automática e algumas ferramentas adicionais para iniciar rapidamente o acesso ao banco de dados: **o módulo spring-boot-startedata-jpa**. Você também pode configurar automaticamente banco de dados integrados como o H2.

```
<dependencies>
  <dependency>
   <groupid>org.springframework.boot</groupid>
   <artifactid>spring-boot-starter-data-jpa</artifactid>
  </dependency>
  <dependency>
     <groupid>com.h2database</groupid>
     <artifactid>h2</artifactid>
     <scope>runtime</scope>
  </dependency>
</dependencies>
```

O **Hibernate** é a implementação de referência para JPA no Spring Boot. Isso significa que o inicializador traz as dependências do Hibernate para dentro. Ele também inclui os principais artefatos JPA e a dependência de seu módulo principal, **Spring Data JPA**.


Stack de tecnologias do Spring Boot Data JPA
Spring Data JPA
JpaRepository
Além do JPA EntityManager, o Spring Data JPA define uma interface JpaRepository com os métodos mais comuns que precisamos usar: FIND, GET, DELETE, UPDATE, etc.
Implementa
Spring Data JPA
SimpleJpaRepository
A classe SimpleJpaRepository é a implementação padrão no Spring e usa EntityManager por baixo. Isso significa que não precisamos usar o padrão JPA puro ou Hibernate para realizar as operações de banco de dados em nosso código, pois podemos usar essas abstrações do Spring.
Usa
JPA
EntityManager
SessionImpl, por meio de sua hierarquia, implementa a interface JPA EntityManager, que faz parte da especificação JPA. Sua implementação, no Hibernate, é o que faz um ORM completo.
Implementa
Hibernate
SessionImpl
O Hibernate usa essas APIs - e portanto, a implementação do HikariCP em nossa aplicação - para se conectar ao banco de dados. Ao implementar o JPA no Hibernate, quem irá administrar o banco de dados é a classe SessionImpl, que inclui muitos códigos para fazer declarações, executar consultas, manipular as conexões de sessão, etc.
Usa
O Spring Boot vem com o HikariCP, uma das implementações mais populares em grupos (pools) de conexões DataSource porque é leve e tem um bom desempenho.
Java API- Hikari Implementation
DataSource
Connection Pool
Java API
Connection
Aqui podemos encontrar as interfaces DataSource, Connection e alguns outros recursos agrupados como ConnectionPoolDataSource. Existem várias implementações dessas APIs para diferentes provedores.
Existem algumas APIs principais do Java para manipular banco de dados SQL nos pacotes java.sql e javax.sql.
H2