

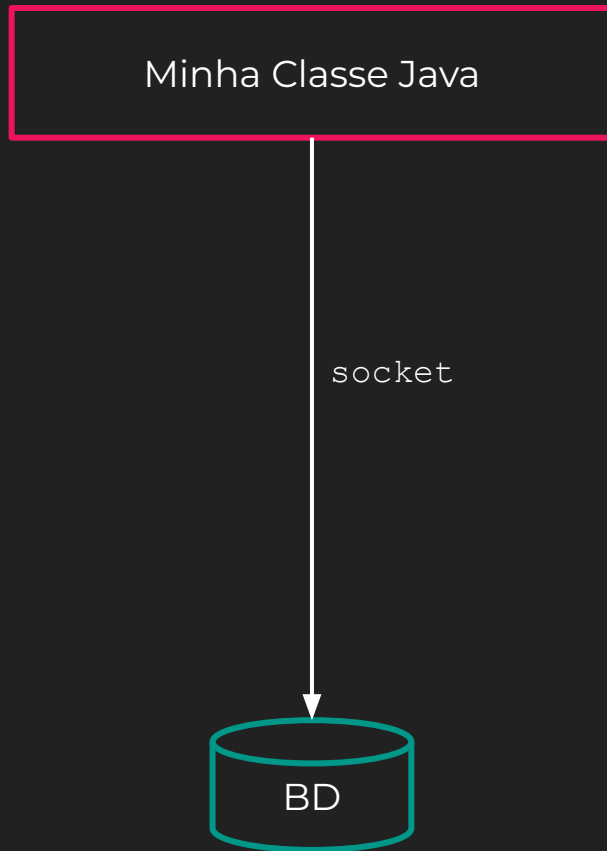
JAVA ADVANCED

**#03**

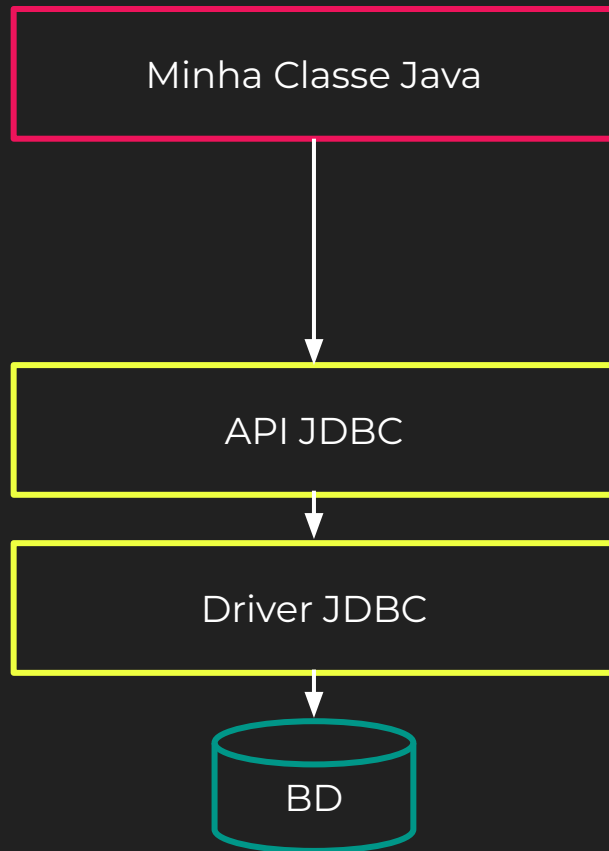
**SPRING DATA JPA**

João Carlos Lima

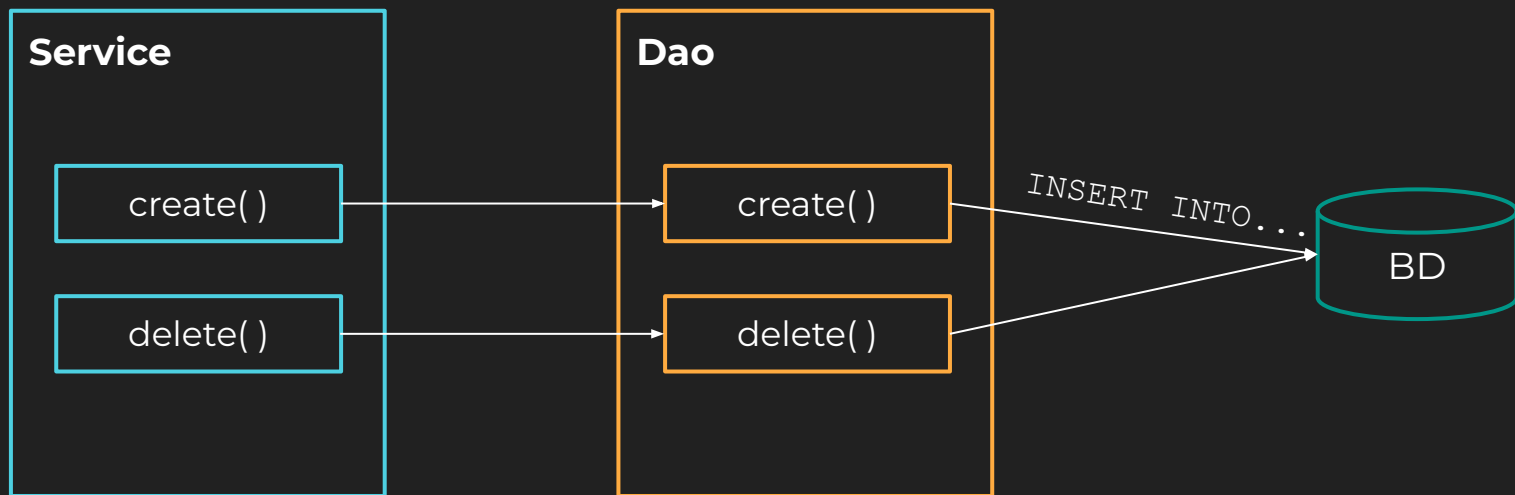
# ACESSO AO BANCO DE DADOS



# ACESSO AO BANCO DE DADOS



# DESIGN PATTERN DAO



# DESIGN PATTERN DAO



```
public class ProdutoService{  
  
    private ProdutoDao dao;  
  
    public ProdutoService(ProdutoDao dao){  
        this.dao = dao;  
    }  
  
    public void cadastrarProduto(Produto produto){  
        this.dao.cadastrar(produto);  
    }  
}
```

# DESIGN PATTERN DAO

```
public class ProdutoDao{

    private Connection conexao;

    public ProdutoDao(Connection conexao){
        this.conexao = conexao;
    }

    public void cadastrar(Produto produto){
        String sql = "INSERT INTO produtos VALUES (?, ?, ?)";

        PreparedStatement ps = conexao.prepareStatement(sql);
        ps.setString(1, produto.getNome());
        ps.setString(2, produto.getDescricao());
        ps.setBigDecimal(3, produto.getPreco());

        ps.execute();
        ps.close();
    }
}
```

# PROBLEMAS DO JDBC

```
public class ProdutoDao{

    private Connection conexao;

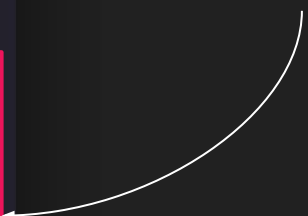
    public ProdutoDao(Connection conexao){
        this.conexao = conexao;
    }

    public void cadastrar(Produto produto){
        String sql = "INSERT INTO produtos VALUES (?, ?, ?)";

        PreparedStatement ps = conexao.prepareStatement(sql);
        ps.setString(1, produto.getNome());
        ps.setString(2, produto.getDescricao());
        ps.setBigDecimal(3, produto.getPreco());

        ps.execute();
        ps.close();
    }
}
```

muito  
verboso



# PROBLEMAS DO JDBC

dependência  
do SGBD

```
public class ProdutoDao{

    private Connection conexao;

    public ProdutoDao(Connection conexao){
        this.conexao = conexao;
    }

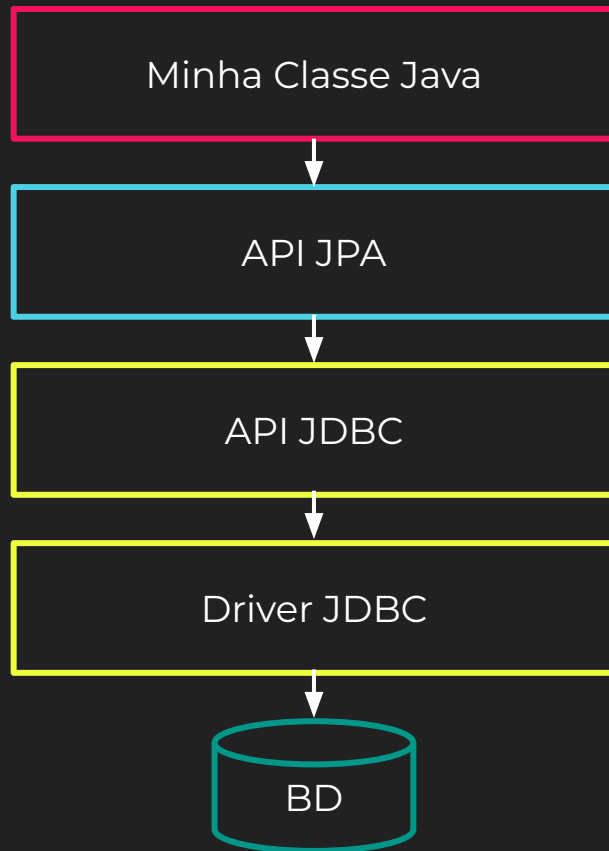
    public void cadastrar(Produto produto){
        String sql = "INSERT INTO produtos VALUES (?, ?, ?)";

        PreparedStatement ps = conexao.prepareStatement(sql);
        ps.setString(1, produto.getNome());
        ps.setString(2, produto.getDescricao());
        ps.setBigDecimal(3, produto.getPreco());

        ps.execute();
        ps.close();
    }
}
```

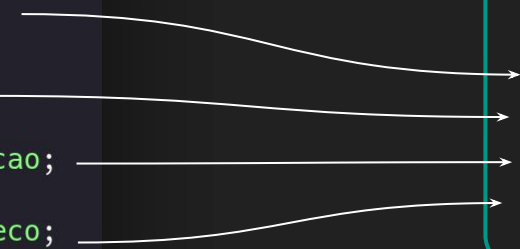


# A SOLUÇÃO JPA



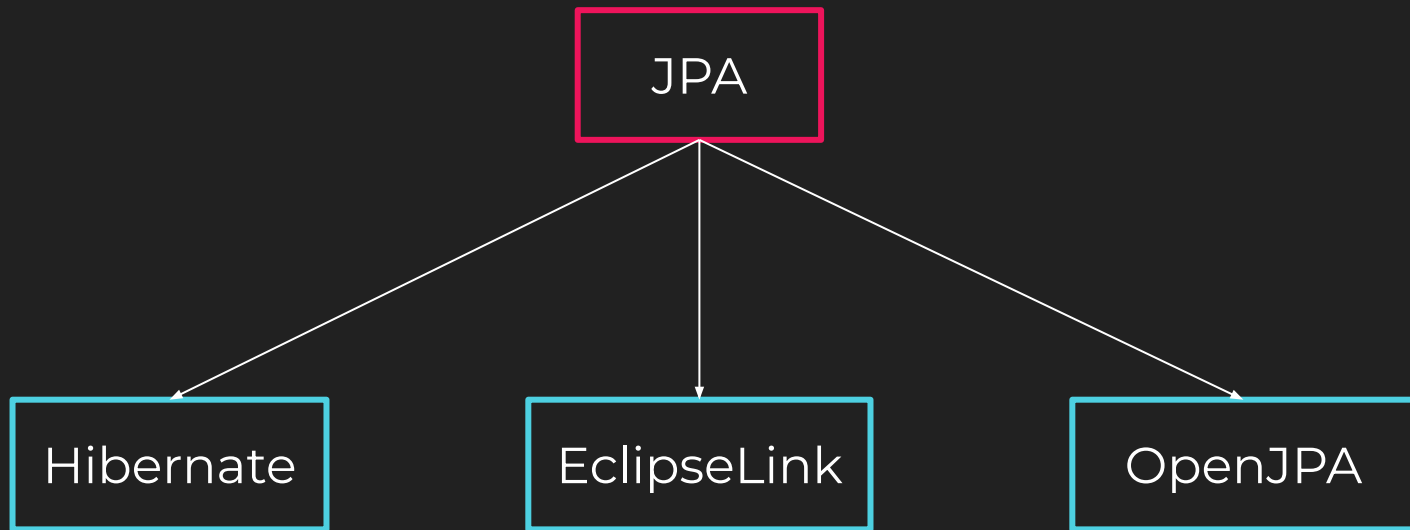
# ORM

```
public class Produto{  
    private int id;  
    private String nome,  
    private String descricao;  
    private BigDecimal preco;  
}
```



Field	Type
id	int
nome	varchar(200)
descricao	varchar(200)
preco	decimal(5,2)

# ESPECIFICAÇÃO VS IMPLEMENTAÇÃO



# PROPRIEDADES DO BD

#DATASOURCE

```
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.url=jdbc:h2:mem:epictask  
spring.datasource.username=sa  
spring.datasource.password=
```

#JPA

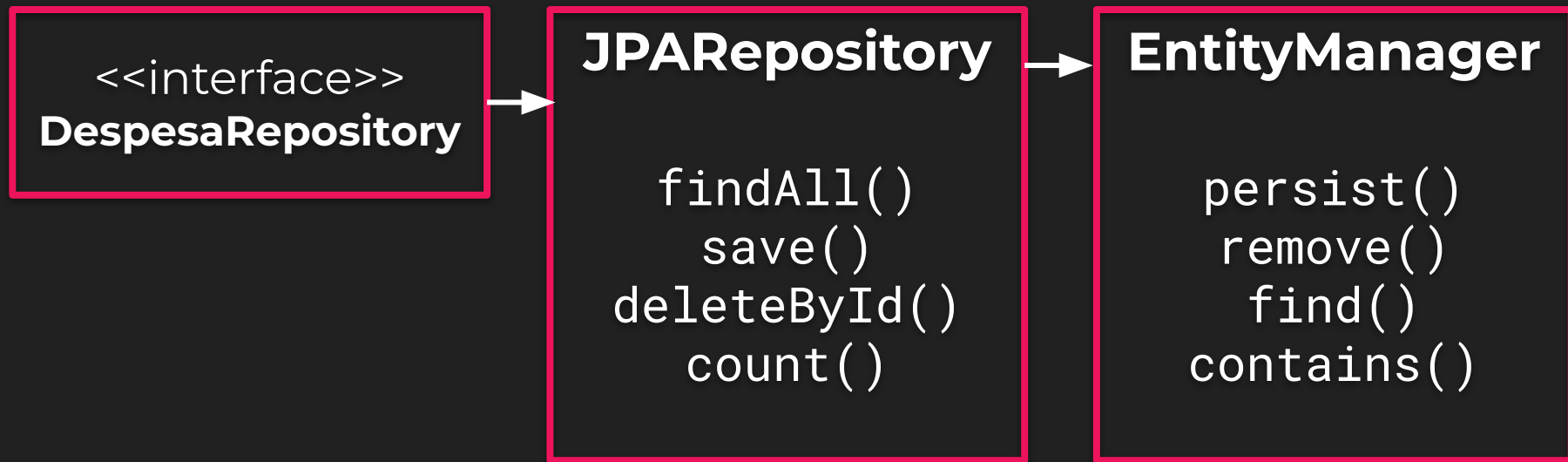
```
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect  
spring.jpa.hibernate.ddl-auto=update
```

#H2-CONSOLE

```
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console
```

application.properties

# SPRING DATA JPA



# SPRING DATA JPA

## JPQL

```
... where x.lastname = ?1 and x.firstname = ?2
```

## Spring Data JPA

```
findByLastnameAndFirstname
```

# SPRING DATA JPA

## JPQL

```
... where x.startDate between ?1 and ?2
```

## Spring Data JPA

```
findByStartDateBetween
```

# SPRING DATA JPA

## JPQL

```
... where x.age not null
```

## Spring Data JPA

```
findByAgeIsNotNull
```



# SPRING DATA JPA

## Criar sua query personalizada

```
public interface UserRepository extends JpaRepository<User, Long>
{
    @Query("select u from User u where u.emailAddress = ?")
    User findByEmailAddress(String emailAddress);
}
```