



# Estácio

Faculdade Estácio

Campus Belford Roxo - RJ

Desenvolvimento Full Stack

Disciplina - Vamos manter as informações

Turma 2023.2

Semestre - 3

Cristian da Silva de Macena

Repositório:

# Mapeamento Objeto-Relacional e DAO

O objetivo dessa prática foi criar um mapeamento objeto-relacional

## Código

```
package cadastrobd.model;

public class Pessoa {

    //Atributos
    int id;
    String nome;
    String logradouro;
    String cidade;
    String telefone;
    String email;

    //Construtor padrão
    public Pessoa(){}

    //Construtor completo
    public Pessoa(int id,String nome,String
logradouro,String cidade,String telefone,String
email){
```

```
    this.id = id;  
    this.nome = nome;  
    this.logradouro = logradouro;  
    this.cidade = cidade;  
    this.telefone = telefone;  
    this.email = email;  
}
```

```
public void setCidade(String cidade) {  
    this.cidade = cidade;  
}
```

```
public void setEmail(String email) {  
    this.email = email;  
}
```

```
public void setId(int id) {  
    this.id = id;  
}
```

```
public void setLogradouro(String logradouro) {  
    this.logradouro = logradouro;  
}
```

```
public void setNome(String nome) {  
    this.nome = nome;  
}
```

```
}
```

```
public void setTelefone(String telefone) {  
    this.telefone = telefone;  
}
```

```
public String getCidade() {  
    return cidade;  
}
```

```
public String getEmail() {  
    return email;  
}
```

```
public int getId() {  
    return id;  
}
```

```
public String getLogradouro() {  
    return logradouro;  
}
```

```
public String getNome() {  
    return nome;  
}
```

```
public String getTelefone() {  
    return telefone;  
}
```

```
//Método exibir  
public void exibir(){  
    System.out.println("Id: " + this.id + ", " +  
"Nome: " + this.nome + ", " + "Logradouro: "  
        + this.logradouro + ", " + "Cidade: " +  
this.cidade + ", " + "Telefone: " + this.telefone + ", "  
+ "Email: " + this.email );  
}  
  
}
```

```
package cadastrobd.model;
```

```
public class PessoaFisica extends Pessoa{
```

```
    //Atributos  
    private String cpf;
```

```
    //Construtor padrão  
    public PessoaFisica(){}  
}
```

```
//Construtor completo
public PessoaFisica(int id,String nome,String
logradouro,String cidade,String telefone,String
email,String cpf){
    super(id, nome, logradouro, cidade, telefone,
email);
    this.cpf = cpf;
}
```

```
public void setCpf(String cpf) {
    this.cpf = cpf;
}
```

```
public String getCpf() {
    return cpf;
}
```

```
//Método exibir
@Override
public void exibir() {
    System.out.println("Id: " + this.id + ", " +
"Nome: " + this.nome + ", " + "Logradouro: "
```

```
        + this.logradouro + ", " + "Cidade: " +  
this.cidade + ", " + "Telefone: " + this.telefone + ", "  
+ "Email: " + this.email + ", " + "Cpf: " + this.cpf);  
    }  
  
}
```

```
package cadastrabd.model;
```

```
public class PessoaJuridica extends Pessoa {  
    private String cnpj;
```

```
    public PessoaJuridica(int id,String nome,String  
logradouro,String cidade,String telefone,String  
email,String cnpj){  
        super(id, nome, logradouro, cidade, telefone,  
email);  
        this.cnpj = cnpj;  
    }
```

```
    public void setCnpj(String cnpj) {  
        this.cnpj = cnpj;  
    }
```

```
    public String getCnpj() {  
        return cnpj;  
    }
```

```

    }

    //Método exibir
    @Override
    public void exibir() {
        System.out.println("Id: " + this.id + ", " +
            "Nome: " + this.nome + ", " + "Logradouro: "
                + this.logradouro + ", " + "Cidade: " +
            this.cidade + ", " + "Telefone: " + this.telefone + ", "
            + "Email: " + this.email + ", " + "Cnpj: " + this.cnpj);
    }
}

```

```

package cadastrobd.model.util;

```

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

```

```

public class ConectorBD {
    private static final String URL = "";
    private static final String USER = "";

```



```
private static final String PASSWORD = "";
```

```
public static Connection getConnection() throws  
SQLException {  
    return DriverManager.getConnection(URL,  
USER, PASSWORD);  
}
```

```
public static PreparedStatement  
getPrepared(Connection conn, String sql) throws  
SQLException {  
    return conn.prepareStatement(sql);  
}
```

```
public static ResultSet  
getSelect(PreparedStatement pstmt) throws  
SQLException {  
    return pstmt.executeQuery();  
}
```

```
public static void close(Statement stmt) {  
    if (stmt != null) {  
        try {  
            stmt.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
    }  
  }  
}
```

```
public static void close(ResultSet rs) {  
    if (rs != null) {  
        try {  
            rs.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
public static void close(Connection conn) {  
    if (conn != null) {  
        try {  
            conn.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
package cadastrabd.model.util;
```

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

public class SequenceManager {
    public static long getValue(String
sequenceName) throws SQLException {
        long nextValue = -1;
        String sql = "SELECT " + sequenceName +
".NEXTVAL FROM DUAL";

        try (Connection conn =
ConectorBD.getConnection();
            PreparedStatement pstmt =
ConectorBD.getPrepared(conn, sql);
            ResultSet rs =
ConectorBD.getSelect(pstmt)) {

            if (rs.next()) {
                nextValue = rs.getLong(1);
            }
        }

        return nextValue;
    }
}
```

```
}  
}
```

```
package cadastrabd.model.util;
```

```
import cadastrabd.model.PessoaFisica;  
import cadastrabd.model.PessoaJuridica;
```

```
import java.sql.SQLException;  
import java.util.List;
```

```
public class CadastroBDTeste {
```

```
    public static void main(String[] args) {  
        // Instanciar os DAOs  
        PessoaFisicaDAO pfDAO = new  
PessoaFisicaDAO();  
        PessoaJuridicaDAO pjDAO = new  
PessoaJuridicaDAO();
```

```
        try {  
            // 1. Instanciar uma pessoa física e persistir  
no banco de dados  
            PessoaFisica pessoaFisica = new  
PessoaFisica(0, "João Silva", "Rua das Flores",
```

```
"São Paulo", "123456789", "joao@example.com",  
"12345678900");  
    pfDAO.incluir(pessoaFisica);  
    System.out.println("Pessoa Física incluída  
com sucesso!");
```

```
// 2. Alterar os dados da pessoa física no  
banco
```

```
    PessoaFisica pessoaFisicaAlterada =  
pfDAO.getPessoa(pessoaFisica.getId());  
    if (pessoaFisicaAlterada != null) {
```

```
pessoaFisicaAlterada.setNome("Marcos");
```

```
pessoaFisicaAlterada.setLogradouro("Rua  
amélia");
```

```
pessoaFisicaAlterada.setCidade("Belford Roxo");
```

```
pessoaFisicaAlterada.setTelefone("219999999");
```

```
pessoaFisicaAlterada.setEmail("email@gmail.com  
");
```

```
pessoaFisicaAlterada.setCpf("11111111111");  
    pfDAO.alterar(pessoaFisicaAlterada);
```

```
        System.out.println("Pessoa Física  
alterada com sucesso!");  
    }
```

// 3. Consultar todas as pessoas físicas do banco de dados e listar no console

```
    List<PessoaFisica> pessoasFisicas =  
pfDAO.getTodos();  
    System.out.println("Lista de Pessoas  
Físicas:");  
    for (PessoaFisica pf : pessoasFisicas) {  
        pf.exibir();  
    }
```

// 4. Excluir a pessoa física criada anteriormente no banco

```
    pfDAO.excluir(pessoaFisica.getId());  
    System.out.println("Pessoa Física excluída  
com sucesso!");
```

// 5. Instanciar uma pessoa jurídica e persistir no banco de dados

```
    PessoaJuridica pessoaJuridica = new  
PessoaJuridica(0, "Empresa XYZ LTDA", "Avenida  
Central", "São Paulo", "1122334455",  
"empresa@example.com", "12345678000195");
```

```
        pjDAO.incluir(pessoaJuridica);  
        System.out.println("Pessoa Jurídica  
incluída com sucesso!");
```

```
        // 6. Alterar os dados da pessoa jurídica no  
        banco
```

```
        PessoaJuridica pessoaJuridicaAlterada =  
        pjDAO.getPessoa(pessoaJuridica.getId());  
        if (pessoaJuridicaAlterada != null) {  
            pessoaJuridicaAlterada.setNome("Ana");
```

```
        pessoaJuridicaAlterada.setLogradouro("Rua  
Santos");
```

```
        pessoaJuridicaAlterada.setCidade("Nova iguaçu");
```

```
        pessoaJuridicaAlterada.setTelefone("21999999");
```

```
        pessoaJuridicaAlterada.setEmail("email@gamil.co  
m");
```

```
        pessoaJuridicaAlterada.setCnpj("111111111111111111  
");
```

```
        pjDAO.alterar(pessoaJuridicaAlterada);  
        System.out.println("Pessoa Jurídica  
alterada com sucesso!");
```

```
}
```

```
// 7. Consultar todas as pessoas jurídicas  
do banco de dados e listar no console
```

```
List<PessoaJuridica> pessoasJuridicas =  
pjDAO.getTodos();
```

```
System.out.println("Lista de Pessoas  
Jurídicas:");
```

```
for (PessoaJuridica pj : pessoasJuridicas) {  
    pj.exibir();  
}
```

```
// 8. Excluir a pessoa jurídica criada  
anteriormente no banco
```

```
    pjDAO.excluir(pessoaJuridica.getId());
```

```
    System.out.println("Pessoa Jurídica  
excluída com sucesso!");
```

```
} catch (SQLException e) {
```

```
    // Tratamento de exceções de banco de  
dados
```

```
    System.out.println("Erro ao acessar o  
banco de dados: " + e.getMessage());
```

```
    e.printStackTrace();
```

```
} catch (Exception e) {
```

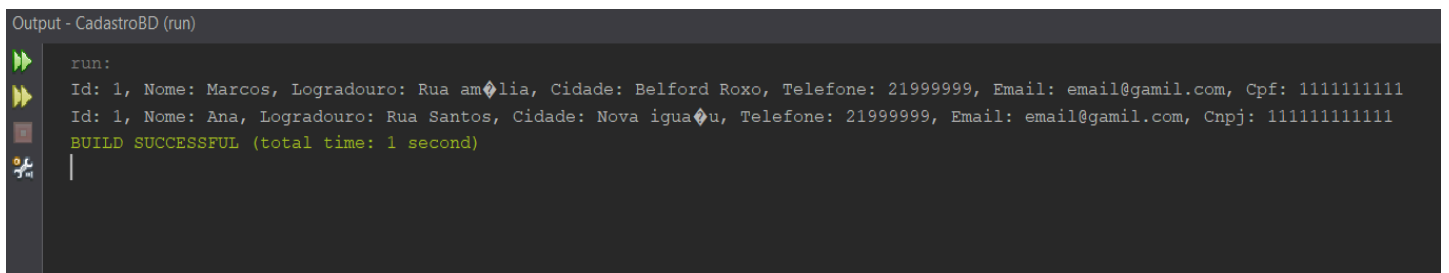
```
    // Tratamento de outras exceções
```



```

        System.out.println("Erro: " +
e.getMessage());
        e.printStackTrace();
    }
}
}

```



```

Output - CadastroBD (run)
run:
Id: 1, Nome: Marcos, Logradouro: Rua amélia, Cidade: Belford Roxo, Telefone: 21999999, Email: email@gamil.com, Cpf: 1111111111
Id: 1, Nome: Ana, Logradouro: Rua Santos, Cidade: Nova iguaçu, Telefone: 21999999, Email: email@gamil.com, Cnpj: 111111111111
BUILD SUCCESSFUL (total time: 1 second)

```

## Análise e Conclusão

Qual a importância dos componentes de middleware, como o JDBC?

Os componentes de middleware, como o JDBC, são essenciais para a integração e comunicação entre sistemas. O JDBC permite que aplicações Java se conectem a diversos bancos de dados, padronizando a comunicação e melhorando a eficiência na execução de comandos SQL.

Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

A principal diferença entre Statement e PreparedStatement no JDBC está na forma como lidam com consultas SQL.

## Como o padrão DAO melhora a manutenibilidade do software?

O padrão DAO melhora a manutenibilidade do software ao isolar a lógica de acesso a dados da lógica de negócios, facilitando manutenção, testes e reutilização de código.

## Como a herança é refletida no banco de dados, quando lidamos com um modelo estritamente relacional?

Quando lidamos com um modelo relacional, a herança geralmente é refletida criando tabelas separadas para cada classe ou uma única tabela que inclui colunas para todos os atributos das classes. Isso ajuda a manter a estrutura e as relações entre os dados de forma organizada e eficiente.