



# Estácio

Faculdade Estácio

Campus Belford Roxo - RJ

Desenvolvimento Full Stack

Disciplina - Back-end Sem Banco Não Tem

Turma 2023.2

Semestre - 3

Cristian da Silva de Macena

Repositório:

<https://github.com/CristianS34/cadastro-bd.git>

## Alimentando a Base

O objetivo dessa prática foi inserir os dados no banco através do jdbc

## Código

```
package cadastrobd.model.util;  
import java.sql.Connection;
```

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

public class PessoaFisicaDAO {

    public PessoaFisica getPessoa(long id)
    throws SQLException {
        String sql = "SELECT * FROM Pessoa p
        JOIN PessoaFisica pf ON p.id = pf.id WHERE
        p.id = ?";
        try (Connection conn =
        ConectorBD.getConnection();
            PreparedStatement pstmt =
        ConectorBD.getPrepared(conn, sql)) {

            pstmt.setLong(1, id);
            ResultSet rs =
        ConectorBD.getSelect(pstmt);
            if (rs.next()) {
```

```
        return new
        PessoaFisica(rs.getLong("id"),
        rs.getString("nome"), rs.getString("cpf"));
    }
}
return null;
}
```

```
    public List<PessoaFisica> getPessoas()
    throws SQLException {
        List<PessoaFisica> pessoas = new
        ArrayList<>();
        String sql = "SELECT * FROM Pessoa p
        JOIN PessoaFisica pf ON p.id = pf.id";
        try (Connection conn =
        ConectorBD.getConnection();
            PreparedStatement pstmt =
        ConectorBD.getPrepared(conn, sql);
            ResultSet rs =
        ConectorBD.getSelect(pstmt)) {

            while (rs.next()) {
```

```
        pessoas.add(new
PessoaFisica(rs.getLong("id"),
rs.getString("nome"), rs.getString("cpf")));
    }
}
return pessoas;
}
```

```
    public void incluir(PessoaFisica pessoa)
throws SQLException {
        String sqlPessoa = "INSERT INTO
Pessoa (id, nome) VALUES (?, ?)";
        String sqlPessoaFisica = "INSERT INTO
PessoaFisica (id, cpf) VALUES (?, ?)";
```

```
        try (Connection conn =
ConectorBD.getConnection()) {
            conn.setAutoCommit(false);
```

```
            long id =
SequenceManager.getValue("pessoa_seq");
```

```
            try (PreparedStatement pstmtPessoa
= ConectorBD.getPrepared(conn, sqlPessoa);
```

## PreparedStatement

```
pstmtPessoaFisica =
ConectorBD.getPrepared(conn,
sqlPessoaFisica)) {

    pstmtPessoa.setLong(1, id);
    pstmtPessoa.setString(2,
pessoa.getNome());
    pstmtPessoa.executeUpdate();

    pstmtPessoaFisica.setLong(1, id);
    pstmtPessoaFisica.setString(2,
pessoa.getCpf());

pstmtPessoaFisica.executeUpdate();

    conn.commit();
} catch (SQLException e) {
    conn.rollback();
    throw e;
}
}
```

```
public void alterar(PessoaFisica pessoa)
throws SQLException {
```

```
    String sqlPessoa = "UPDATE Pessoa
SET nome = ? WHERE id = ?";
```

```
    String sqlPessoaFisica = "UPDATE
PessoaFisica SET cpf = ? WHERE id = ?";
```

```
    try (Connection conn =
ConectorBD.getConnection()) {
        conn.setAutoCommit(false);
```

```
        try (PreparedStatement pstmtPessoa
= ConectorBD.getPrepared(conn, sqlPessoa);
```

```
            PreparedStatement
```

```
pstmtPessoaFisica =
```

```
ConectorBD.getPrepared(conn,
sqlPessoaFisica)) {
```

```
            pstmtPessoa.setString(1,
pessoa.getNome());
```

```
            pstmtPessoa.setLong(2,
pessoa.getId());
```

```
            pstmtPessoa.executeUpdate();
```

```
        pstmtPessoaFisica.setString(1,
pessoa.getCpf());
        pstmtPessoaFisica.setLong(2,
pessoa.getId());

pstmtPessoaFisica.executeUpdate();
```

```
        conn.commit();
    } catch (SQLException e) {
        conn.rollback();
        throw e;
    }
}
}
```

```
public void excluir(long id) throws
SQLException {
    String sqlPessoaFisica = "DELETE
FROM PessoaFisica WHERE id = ?";
    String sqlPessoa = "DELETE FROM
Pessoa WHERE id = ?";
```

```
    try (Connection conn =
ConectorBD.getConnection()) {
```

```

        conn.setAutoCommit(false);

        try (PreparedStatement
pstmtPessoaFisica =
ConectorBD.getPrepared(conn,
sqlPessoaFisica);
        PreparedStatement pstmtPessoa =
ConectorBD.getPrepared(conn, sqlPessoa)) {

            pstmtPessoaFisica.setLong(1, id);

pstmtPessoaFisica.executeUpdate();

            pstmtPessoa.setLong(1, id);
            pstmtPessoa.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            conn.rollback();
            throw e;
        }
    }
}
}
}

```



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

```
package cadastro.model;
```

```
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;
```

```
public class PessoaJuridicaDAO {
```

```
    public PessoaJuridica getPessoa(long id)  
    throws SQLException {  
        String sql = "SELECT * FROM Pessoa p  
        JOIN PessoaJuridica pj ON p.id = pj.id  
        WHERE p.id = ?";  
        try (Connection conn =  
            ConectorBD.getConnection();  
            PreparedStatement pstmt =  
            ConectorBD.getPrepared(conn, sql)) {  
            pstmt.setLong(1, id);
```

```

        ResultSet rs =
ConectorBD.getSelect(pstmt);
        if (rs.next()) {
            return new
PessoaJuridica(rs.getLong("id"),
rs.getString("nome"), rs.getString("cnpj"));
        }
    }
    return null;
}

```

```

    public List<PessoaJuridica> getPessoas()
throws SQLException {
        List<PessoaJuridica> pessoas = new
ArrayList<>();
        String sql = "SELECT * FROM Pessoa p
JOIN PessoaJuridica pj ON p.id = pj.id";
        try (Connection conn =
ConectorBD.getConnection();
            PreparedStatement pstmt =
ConectorBD.getPrepared(conn, sql);
            ResultSet rs =
ConectorBD.getSelect(pstmt)) {

```

```
        while (rs.next()) {  
            pessoas.add(new  
PessoaJuridica(rs.getLong("id"),  
rs.getString("nome"), rs.getString("cnpj")));  
        }  
    }  
    return pessoas;  
}
```

```
    public void incluir(PessoaJuridica pessoa)  
throws SQLException {  
        String sqlPessoa = "INSERT INTO  
Pessoa (id, nome) VALUES (?, ?)";  
        String sqlPessoaJuridica = "INSERT  
INTO PessoaJuridica (id, cnpj) VALUES (?,  
?)";
```

```
        try (Connection conn =  
ConectorBD.getConnection()) {  
            conn.setAutoCommit(false);  
  
            long id =  
SequenceManager.getValue("pessoa_seq");
```

```
        try (PreparedStatement pstmtPessoa
= ConectorBD.getPrepared(conn, sqlPessoa);
        PreparedStatement
pstmtPessoaJuridica =
ConectorBD.getPrepared(conn,
sqlPessoaJuridica)) {

            pstmtPessoa.setLong(1, id);
            pstmtPessoa.setString(2,
pessoa.getNome());
            pstmtPessoa.executeUpdate();

            pstmtPessoaJuridica.setLong(1, id);
            pstmtPessoaJuridica.setString(2,
pessoa.getCnpj());

            pstmtPessoaJuridica.executeUpdate();

            conn.commit();
        } catch (SQLException e) {
            conn.rollback();
            throw e;
        }
    }
```

```
}
```

```
    public void alterar(PessoaJuridica pessoa)  
    throws SQLException {
```

```
        String sqlPessoa = "UPDATE Pessoa  
SET nome = ? WHERE id = ?";
```

```
        String sqlPessoaJuridica = "UPDATE  
PessoaJuridica SET cnpj = ? WHERE id = ?";
```

```
        try (Connection conn =  
ConectorBD.getConnection()) {  
            conn.setAutoCommit(false);
```

```
            try (PreparedStatement pstmtPessoa  
= ConectorBD.getPrepared(conn, sqlPessoa);  
                PreparedStatement  
pstmtPessoaJuridica =  
ConectorBD.getPrepared(conn,  
sqlPessoaJuridica)) {
```

```
                pstmtPessoa.setString(1,  
pessoa.getNome());  
                pstmtPessoa.setLong(2,  
pessoa.getId());
```

```

        pstmtPessoa.executeUpdate();

        pstmtPessoaJuridica.setString(1,
pessoa.getCnpj());
        pstmtPessoaJuridica.setLong(2,
pessoa.getId());

pstmtPessoaJuridica.executeUpdate();

        conn.commit();
    } catch (SQLException e) {
        conn.rollback();
        throw e;
    }
}
}
}

```

```

    public void excluir(long id) throws
SQLException {
        String sqlPessoaJuridica = "DELETE
FROM PessoaJuridica WHERE id = ?";
        String sqlPessoa = "DELETE FROM
Pessoa WHERE id = ?";

```

```
try (Connection conn =
ConectorBD.getConnection()) {
    conn.setAutoCommit(false);

    try (PreparedStatement
pstmtPessoaJuridica =
ConectorBD.getPrepared(conn,
sqlPessoaJuridica);
        PreparedStatement pstmtPessoa =
ConectorBD.getPrepared(conn, sqlPessoa)) {

        pstmtPessoaJuridica.setLong(1, id);

pstmtPessoaJuridica.executeUpdate();

        pstmtPessoa.setLong(1, id);
        pstmtPessoa.executeUpdate();

        conn.commit();
    } catch (SQLException e) {
        conn.rollback();
        throw e;
    }
}
```

```
}  
}
```

```
import cadastrobd.model.PessoaFisica;  
import cadastrobd.model.PessoaJuridica;  
import cadastrobd.model.util.PessoaFisicaDAO;  
import cadastrobd.model.util.PessoaJuridicaDAO;
```

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

```
public class CadastroBD {
```

```
    public static void main(String[] args) {  
        // Scanner para leitura de entradas do usuário  
        Scanner scanner = new Scanner(System.in);
```

```
        // Instâncias dos DAOs para gerenciar  
        operações de banco de dados
```

```
        PessoaFisicaDAO pfDAO = new  
        PessoaFisicaDAO();
```

```
        PessoaJuridicaDAO pjDAO = new  
        PessoaJuridicaDAO();
```

```
        while (true) {
```



```
// Apresentação do menu de opções para o  
usuário
```

```
System.out.println("Escolha uma opção:");  
System.out.println("1. Incluir");  
System.out.println("2. Alterar");  
System.out.println("3. Excluir");  
System.out.println("4. Exibir pelo ID");  
System.out.println("5. Exibir todos");  
System.out.println("0. Sair");
```

```
int opcao =
```

```
Integer.parseInt(scanner.nextLine());
```

```
try {
```

```
// Tratamento das opções escolhidas  
pelo usuário
```

```
switch (opcao) {
```

```
case 1:
```

```
    incluir(scanner, pfDAO, pjDAO);  
    break;
```

```
case 2:
```

```
    alterar(scanner, pfDAO, pjDAO);  
    break;
```

```
case 3:
```

```
    excluir(scanner, pfDAO, pjDAO);  
    break;
```

```
case 4:
```

```
        exibirPeloid(scanner, pfDAO,
pjDAO);
        break;
    case 5:
        exibirTodos(scanner, pfDAO,
pjDAO);
        break;
    case 0:
        // Encerramento do programa
        System.out.println("Encerrando o
programa.");
        scanner.close();
        return;
    default:
        // Opção inválida
        System.out.println("Opção inválida.
Tente novamente.");
    }
} 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();
    }
}
}
```

// Método para incluir uma nova pessoa no banco de dados

```
private static void incluir(Scanner scanner,
PessoaFisicaDAO pfDAO, PessoaJuridicaDAO
pjDAO) throws SQLException, Object {
    System.out.println("Incluir Pessoa:");
    System.out.println("1. Física");
    System.out.println("2. Jurídica");
    int tipo = Integer.parseInt(scanner.nextLine());

    if (tipo == 1) {
        // Inclusão de pessoa física
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Logradouro: ");
        String logradouro = scanner.nextLine();
        System.out.print("Cidade: ");
        String cidade = scanner.nextLine();
        System.out.print("Telefone: ");
```

```
String telefone = scanner.nextLine();
System.out.print("Email: ");
String email = scanner.nextLine();
System.out.print("CPF: ");
String cpf = scanner.nextLine();
```

```
    PessoaFisica pessoaFisica = new
PessoaFisica(0, nome, logradouro, cidade,
telefone, email, cpf);
    pfDAO.incluir(pessoaFisica);
    System.out.println("Pessoa Física incluída
com sucesso!");
```

```
    } else if (tipo == 2) {
        // Inclusão de pessoa jurídica
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Logradouro: ");
        String logradouro = scanner.nextLine();
        System.out.print("Cidade: ");
        String cidade = scanner.nextLine();
        System.out.print("Telefone: ");
        String telefone = scanner.nextLine();
        System.out.print("Email: ");
        String email = scanner.nextLine();
        System.out.print("CNPJ: ");
        String cnpj = scanner.nextLine();
```

```
        PessoaJuridica pessoaJuridica = new
PessoaJuridica(0, nome, logradouro, cidade,
telefone, email, cnpj);
        pjDAO.incluir(pessoaJuridica);
        System.out.println("Pessoa Jurídica
incluída com sucesso!");
    } else {
        // Tipo inválido
        System.out.println("Tipo inválido.");
    }
}
```

// Método para alterar os dados de uma pessoa  
no banco de dados

```
private static void alterar(Scanner scanner,
PessoaFisicaDAO pfDAO, PessoaJuridicaDAO
pjDAO) throws SQLException {
    System.out.println("Alterar Pessoa:");
    System.out.println("1. Física");
    System.out.println("2. Jurídica");
    int tipo = Integer.parseInt(scanner.nextLine());

    System.out.print("ID: ");
    int id = Integer.parseInt(scanner.nextLine());
```

```
        if (tipo == 1) {  
            // Alteração de pessoa física  
            PessoaFisica pessoa =  
pfDAO.getPessoa(id);  
            if (pessoa == null) {  
                System.out.println("Pessoa Física não  
encontrada.");  
                return;  
            }  
            System.out.println("Nome atual: " +  
pessoa.getNome());  
            System.out.println("Logradouro atual: " +  
pessoa.getLogradouro());  
            System.out.println("Cidade atual: " +  
pessoa.getCidade());  
            System.out.println("Telefone atual: " +  
pessoa.getTelefone());  
            System.out.println("Email atual: " +  
pessoa.getEmail());  
            System.out.println("CPF atual: " +  
pessoa.getCpf());
```

```
        System.out.print("Novo Nome: ");  
        String nome = scanner.nextLine();  
        System.out.print("Novo Logradouro: ");  
        String logradouro = scanner.nextLine();
```

```
System.out.print("Nova Cidade: ");
String cidade = scanner.nextLine();
System.out.print("Novo Telefone: ");
String telefone = scanner.nextLine();
System.out.print("Novo Email: ");
String email = scanner.nextLine();
System.out.print("Novo CPF: ");
String cpf = scanner.nextLine();
```

```
    pessoa.setNome(nome);
    pessoa.setLogradouro(logradouro);
    pessoa.setCidade(cidade);
    pessoa.setTelefone(telefone);
    pessoa.setEmail(email);
    pessoa.setCpf(cpf);
```

```
    pfDAO.alterar(pessoa);
    System.out.println("Pessoa Física alterada
com sucesso!");
```

```
    } else if (tipo == 2) {
        // Alteração de pessoa jurídica
        PessoaJuridica pessoa =
        pjDAO.getPessoa(id);
        if (pessoa == null) {
            System.out.println("Pessoa Jurídica não
encontrada.");
```

```
        return;
    }
    System.out.println("Nome atual: " +
    pessoa.getNome());
    System.out.println("Logradouro atual: " +
    pessoa.getLogradouro());
    System.out.println("Cidade atual: " +
    pessoa.getCidade());
    System.out.println("Telefone atual: " +
    pessoa.getTelefone());
    System.out.println("Email atual: " +
    pessoa.getEmail());
    System.out.println("CNPJ atual: " +
    pessoa.getCnpj());
```

```
System.out.print("Novo Nome: ");
String nome = scanner.nextLine();
System.out.print("Novo Logradouro: ");
String logradouro = scanner.nextLine();
System.out.print("Nova Cidade: ");
String cidade = scanner.nextLine();
System.out.print("Novo Telefone: ");
String telefone = scanner.nextLine();
System.out.print("Novo Email: ");
String email = scanner.nextLine();
System.out.print("Novo CNPJ: ");
```



```

        String cnpj = scanner.nextLine();

        pessoa.setNome(nome);
        pessoa.setLogradouro(logradouro);
        pessoa.setCidade(cidade);
        pessoa.setTelefone(telefone);
        pessoa.setEmail(email);
        pessoa.setCnpj(cnpj);

        pjDAO.alterar(pessoa);
        System.out.println("Pessoa Jurídica
alterada com sucesso!");
    } else {
        // Tipo inválido
        System.out.println("Tipo inválido.");
    }
}

// Método para excluir uma pessoa do banco de
dados
private static void excluir(Scanner scanner,
PessoaFisicaDAO pfDAO, PessoaJuridicaDAO
pjDAO) throws SQLException {
    System.out.println("Excluir Pessoa:");
    System.out.println("1. Física");
    System.out.println("2. Jurídica");
}

```

```
int tipo = Integer.parseInt(scanner.nextLine());

System.out.print("ID: ");
int id = Integer.parseInt(scanner.nextLine());

if (tipo == 1) {
    // Exclusão de pessoa física
    pfDAO.excluir(id);
    System.out.println("Pessoa Física excluída
com sucesso!");
} else if (tipo == 2) {
    // Exclusão de pessoa jurídica
    pjDAO.excluir(id);
    System.out.println("Pessoa Jurídica
excluída com sucesso!");
} else {
    // Tipo inválido
    System.out.println("Tipo inválido.");
}
}
```

// Método para exibir os dados de uma pessoa pelo ID

```
private static void exibirPeloid(Scanner scanner,
PessoaFisicaDAO pfDAO, PessoaJuridicaDAO
pjDAO) throws SQLException {
```

```
System.out.println("Exibir Pessoa pelo ID:");
System.out.println("1. Física");
System.out.println("2. Jurídica");
int tipo = Integer.parseInt(scanner.nextLine());

System.out.print("ID: ");
int id = Integer.parseInt(scanner.nextLine());

if (tipo == 1) {
    // Exibição de pessoa física
    PessoaFisica pessoa =
pfDAO.getPessoa(id);
    if (pessoa == null) {
        System.out.println("Pessoa Física não
encontrada.");
        return;
    }
    pessoa.exibir();
} else if (tipo == 2) {
    // Exibição de pessoa jurídica
    PessoaJuridica pessoa =
pjDAO.getPessoa(id);
    if (pessoa == null) {
        System.out.println("Pessoa Jurídica não
encontrada.");
        return;
    }
}
```

```
    }  
    pessoa.exibir();  
} else {  
    // Tipo inválido  
    System.out.println("Tipo inválido.");  
}  
}
```

```
// Método para exibir todos os registros  
private static void exibirTodos(Scanner scanner,  
PessoaFisicaDAO pfDAO, PessoaJuridicaDAO  
pjDAO) throws SQLException {  
    System.out.println("Exibir Todos:");  
    System.out.println("1. Pessoas Físicas");  
    System.out.println("2. Pessoas Jurídicas");  
    int tipo = Integer.parseInt(scanner.nextLine());  
  
    if (tipo == 1) {  
        // Exibição de todas as pessoas físicas  
        List<PessoaFisica> pessoasFisicas =  
pfDAO.getTodos();  
        for (PessoaFisica pessoa : pessoasFisicas)  
{  
            pessoa.exibir();  
        }  
    } else if (tipo == 2) {
```

```

        // Exibição de todas as pessoas jurídicas
        List<PessoaJuridica> pessoasJuridicas =
pjDAO.getTodos();
        for (PessoaJuridica pessoa :
pessoasJuridicas) {
            pessoa.exibir();
        }
    } else {
        // Tipo inválido
        System.out.println("Tipo inválido.");
    }
}
}
}

```

run:

=====

```

1 - Incluir pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos

```

## **Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?**

A persistência em arquivo armazena dados em arquivos físicos simples, enquanto a persistência em banco de dados usa sistemas gerenciadores para armazenar e consultar dados de forma estruturada e eficiente.

## **Como o uso de operador lambda simplificou a impressão dos valores contidos nas entidades, nas versões mais recentes do Java?**

O uso de operadores lambda no Java simplifica a impressão de valores ao permitir o uso de expressões concisas e funções anônimas em streams, facilitando operações como foreach diretamente sobre coleções.

## **Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?**

Métodos acionados diretamente pelo método main precisam ser static porque o método main é estático e não pode acessar métodos de instância sem criar um objeto da classe.