POLO MADUREIRA - RIO DE JANEIRO - RJ
NIVEL 3 : BACKEND SEM BANCO NÃO TEM
TURMA 2024.1 - MUNDO 3
ALUNO: MAYLSON SIMOES

Relatório discente de acompanhamento

RPG0016 - BackEnd sem banco não tem

Objetivos da prática

- Implementar persistência com base no middleware JDBC.
- Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
- Implementar o mapeamento objeto-relacional em sistemas Java.
- Criar sistemas cadastrais com persistência em banco relacional.
- No final do exercício, o aluno terá criado um aplicativo cadastral com uso do SQL Server na persistência de dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

```java
public class Pessoa {
    protected int id;
    protected String nome;
    protected String logradouro;
    protected String cidade;
    protected String estado;
    protected String telefone;
    protected String email;
    public Pessoa() { }

    public Pessoa(int id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
        this.id = id;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public Pessoa(String nome, String logradouro, String cidade, String estado, String telefone, String email) {
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public void exibir() {
        System.out.println("ID: " + id);
        System.out.println("Nome: " + nome);
        System.out.println("Logradouro: " + logradouro);
        System.out.println("Cidade: " + cidade);
        System.out.println("Estado: " + estado);
        System.out.println("Telefone: " + telefone);
        System.out.println("Email: " + email);
```

```java
public class PessoaFisica extends Pessoa {
    private String cpf;
    public PessoaFisica() { }

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

    public PessoaFisica(String nome, String logradouro, String cidade, String estado, String telefone, String email, String cpf) {
        super(nome, logradouro, cidade, estado, telefone, email);
        this.cpf = cpf;
    }

    public String getCpf() {...3 lines }
    public void setCpf(String cpf) {...3 lines }
    public int getId() {...3 lines }
    public void setId(int id) {...3 lines }
    public String getNome() {...3 lines }
    public void setNome(String nome) {...3 lines }
    public String getLogradouro() {...3 lines }
    public void setLogradouro(String logradouro) {...3 lines }
    public String getCidade() {...3 lines }
    public void setCidade(String cidade) {...3 lines }
    public String getEstado() {...3 lines }
    public void setEstado(String estado) {...3 lines }
    public String getTelefone() {...3 lines }
    public void setTelefone(String telefone) {...3 lines }
    public String getEmail() {...3 lines }
    public void setEmail(String email) {...3 lines }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CPF: " + cpf);
    }
```

```java
public class PessoaJuridica extends Pessoa {
    private String cnpj;
    public PessoaJuridica() { }

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

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

    public String getCnpj() {...3 lines }
    public void setCnpj(String cnpj) {...3 lines }
    public int getId() {...3 lines }
    public void setId(int id) {...3 lines }
    public String getNome() {...3 lines }
    public void setNome(String nome) {...3 lines }
    public String getLogradouro() {...3 lines }
    public void setLogradouro(String logradouro) {...3 lines }
    public String getCidade() {...3 lines }
    public void setCidade(String cidade) {...3 lines }
    public String getEstado() {...3 lines }
    public void setEstado(String estado) {...3 lines }
    public String getTelefone() {...3 lines }
    public void setTelefone(String telefone) {...3 lines }
    public String getEmail() {...3 lines }
    public void setEmail(String email) {...3 lines }

    @Override
    public void exibir() {
        super.exibir();
        System.out.println("CNPJ: " + cnpj);
    }
```

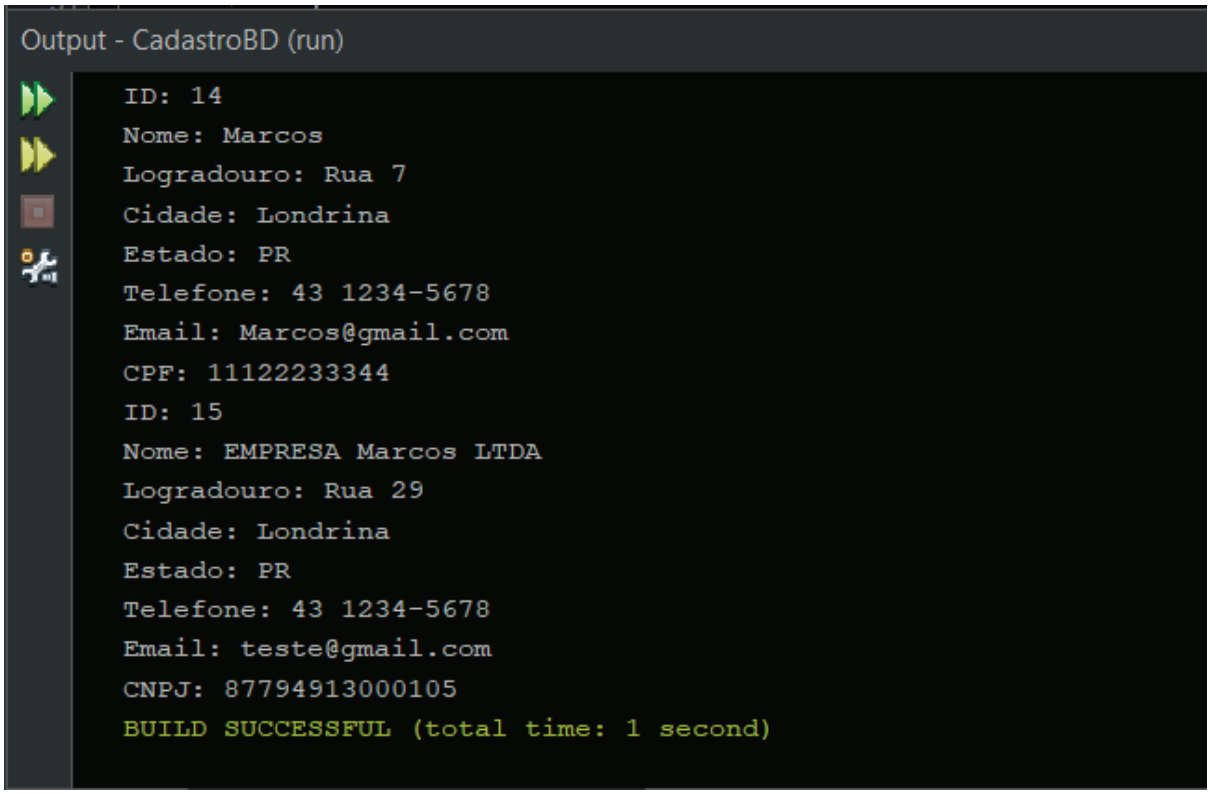
```java
public class ConectorBD {
    private static final String URL = "jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;TrustServerCertificate=true";
    private static final String USER = "loja";
    private static final String PASSWORD = "loja";

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

    public PreparedStatement getPrepared(String sql) throws SQLException {
        return getConnection().prepareStatement(sql);
    }

    public ResultSet getSelect(String sql) throws SQLException {
        return getPrepared(sql).executeQuery();
    }

    public static void close(Connection connection) {...9 lines }

    public static void close(Statement statement) {...9 lines }

    public static void close(ResultSet resultSet) {...9 lines }
}
```

```java
public class SequenceManager {
    public int getValue(String sequenceName){
        ConectorBD cb = new ConectorBD();
        int nextValue = -1;
        try {
            ResultSet rs = cb.getSelect("select NEXT VALUE for " + sequenceName);
            if (rs.next()) {
                nextValue = rs.getInt(1);
            }
            ConectorBD.close(rs);
            ConectorBD.close(cb.getConnection());
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return nextValue;
    }
}
```

```java
public class PessoaFisicaDAO {

    ConectorBD cb = new ConectorBD();
    SequenceManager sequenceManager = new SequenceManager();

    public PessoaFisica getPessoa(int id) {...22 lines }

    public List<PessoaFisica> getPessoas() {...23 lines }

    public void incluir(PessoaFisica pessoa){...27 lines }

    public void alterar(PessoaFisica pessoa) {...26 lines }

    public boolean excluir(int id) {...20 lines }
}
```

```java
public class PessoaJuridicaDAO {

    ConectorBD cb = new ConectorBD();
    SequenceManager sequenceManager = new SequenceManager();

    public PessoaJuridica getPessoa(int id) {...22 lines }

    public List<PessoaJuridica> getPessoas() {...23 lines }

    public void incluir(PessoaJuridica pessoa){...26 lines }

    public void alterar(PessoaJuridica pessoa) {...26 lines }

    public boolean excluir(int id) {...19 lines }
}
```

```java
public class CadastroBDTeste {

    public static void main(String[] args) {
        PessoaFisica pessoaFisica = new PessoaFisica("Marcos", "Rua 7", "Londrina", "PR", "43 1234-5678", "teste@gmail.com", "11122233344");
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();

        pessoaFisicaDAO.incluir(pessoaFisica);

        pessoaFisica.setEmail("Marcos@gmail.com");

        pessoaFisicaDAO.alterar(pessoaFisica);

        List<PessoaFisica> listaPessoasFisicas = pessoaFisicaDAO.getPessoas();

        for(PessoaFisica p : listaPessoasFisicas){
            p.exibir();
        }

        pessoaFisicaDAO.excluir(pessoaFisica.getId());

        PessoaJuridica pessoaJuridica = new PessoaJuridica("EMPRESA Marcos LTDA", "Rua 29", "Londrina", "PR", "43 1234-5678", "teste@gmail.com", "52775121000109");
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        pessoaJuridicaDAO.incluir(pessoaJuridica);

        pessoaJuridica.setCnpj("87794913000105");

        pessoaJuridicaDAO.alterar(pessoaJuridica);

        List<PessoaJuridica> listaPessoasJuridicas = pessoaJuridicaDAO.getPessoas();

        for(PessoaJuridica p : listaPessoasJuridicas){ p.exibir(); }

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



Análise e Conclusão:

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

Desempenham um papel crucial na arquitetura de software, servindo como intermediários que facilitam a comunicação e a interação entre diferentes sistemas ou camadas de uma aplicação. Suas principais razões que explicam a importância são: Abstração da Complexidade, Gerenciamento de Conexões, Segurança, Facilidade de Desenvolvimento, Desempenho e Eficiência, Manutenção e Evolução.

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

Ambas têm propósitos semelhantes, mas diferem em suas funcionalidades e no impacto que podem ter em termos de desempenho e segurança.

- **Statement** é usado para executar consultas SQL estáticas e simples que não exigem parâmetros.
- **PreparedStatement** é usado para executar consultas SQL pré-compiladas e parametrizadas.

3. Como o padrão DAO melhora a manutenibilidade do software?

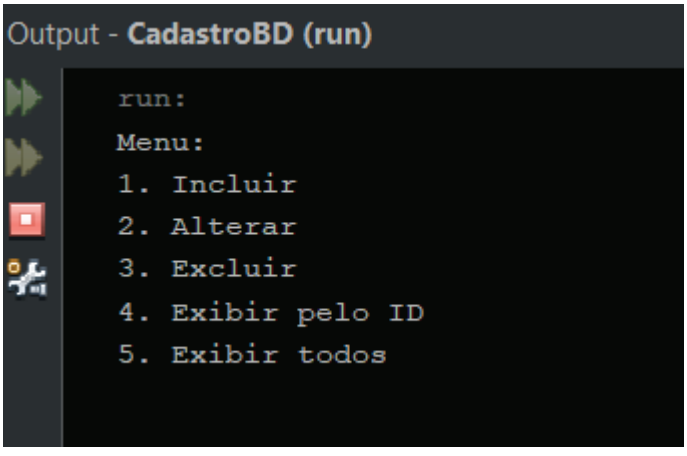
Como as operações de acesso a dados são centralizadas em classes DAO, qualquer mudança na forma como os dados são acessados ou armazenados precisa ser feita apenas na camada DAO, dessa forma simplifica a manutenção e a atualização do código. Por exemplo, se houver necessidade de mudar de um banco de dados SQL para NoSQL, as mudanças podem ser feitas nas classes DAO sem impactar outras partes do sistema.

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

Quando trabalhamos com herança em um banco de dados relacional, a hierarquia de classes de um modelo orientado a objetos precisa ser mapeada para tabelas relacionais. Esse mapeamento pode ser feito usando várias estratégias, como usado nesse projeto, a tabela de subclasse, nesta abordagem, cada classe na hierarquia de herança é mapeada para uma tabela separada. A tabela da superclasse contém as colunas comuns e cada tabela de subclasse contém colunas específicas junto com uma chave estrangeira para a tabela da superclasse.

2º Procedimento | Alimentando a Base

```java
PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
Scanner scanner = new Scanner(System.in);
while (true) {
    System.out.println("Menu:");
    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");
    int opcao = scanner.nextInt();
    scanner.nextLine(); // Consumir a nova linha
    try {
        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:
                System.out.println("Finalizando...");
                scanner.close();
                return;
            default:
                System.out.println("Opção inválida.");
        }
    } catch (Exception e) { ...4 lines }
```



```
Tipo (1 - Física, 2 - Jurídica): 1
Nome: Vinicius
Logradouro: Rua teste
Cidade: cidade Teste
Estado: MA
Telefone: 12345678
Email: teste@gmail.com
CPF: 11122233344
Pessoa física incluída com sucesso.
```

```java
private static void excluir(Scanner scanner, PessoaFisicaDAO pfDAO, PessoaJuridicaDAO pjDAO) {
    System.out.print("Tipo (1 - Física, 2 - Jurídica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();
    if (tipo == 1) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (pfDAO.excluir(id)) {
            System.out.println("Pessoa física excluída com sucesso.");
        } else {
            System.out.println("Erro ao excluir pessoa física.");
        }
    } else if (tipo == 2) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (pjDAO.excluir(id)) {
            System.out.println("Pessoa jurídica excluída com sucesso.");
        } else {
            System.out.println("Erro ao excluir pessoa jurídica.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}
```

```java
private static void alterar(Scanner scanner, PessoaFisicaDAO pfDAO, PessoaJuridicaDAO pjDAO) {
    System.out.print("Tipo (1 - Física, 2 - Jurídica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();
    if (tipo == 1) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        PessoaFisica pf = pfDAO.getPessoa(id);
        if (pf == null) {
            System.out.println("Pessoa física não encontrada.");
            return;
        }
        System.out.println("Dados atuais:");
        pf.exibir();
        System.out.print("Nome: ");
        pf.setNome(scanner.nextLine());
        System.out.print("Logradouro: ");
        pf.setLogradouro(scanner.nextLine());
        System.out.print("Cidade: ");
        pf.setCidade(scanner.nextLine());
        System.out.print("Estado: ");
        pf.setEstado(scanner.nextLine());
        System.out.print("Telefone: ");
        pf.setTelefone(scanner.nextLine());
        System.out.print("Email: ");
        pf.setEmail(scanner.nextLine());
        System.out.print("CPF: ");
        pf.setCpf(scanner.nextLine());
        pfDAO.alterar(pf);
        System.out.println("Pessoa física alterada com sucesso.");
    } else if (tipo == 2) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();
```

```
2
Tipo (1 - Física, 2 - Jurídica): 1
ID: 19
Dados atuais:
ID: 19
Nome: Vinicius
Logradouro: Rua teste
Cidade: cidade Teste
Estado: MA
Telefone: 12345678
Email: teste@gmail.com
CPF: 11122233344
Nome: Vinicius Pereira
Logradouro: Rua teste
Cidade: Cidade teste
Estado: MA
Telefone: 12345678
Email: teste@gmail.com
CPF: 1112223344
Pessoa física alterada com sucesso.
Menu:
```

```java
private static void exibirPeloId(Scanner scanner, PessoaFisicaDAO pfDAO, PessoaJuridicaDAO pjDAO) {
    System.out.print("Tipo (1 - Física, 2 - Jurídica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        PessoaFisica pf = pfDAO.getPessoa(id);
        if (pf != null) {
            pf.exibir();
        } else {
            System.out.println("Pessoa física não encontrada.");
        }
    } else if (tipo == 2) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        PessoaJuridica pj = pjDAO.getPessoa(id);
        if (pj != null) {
            pj.exibir();
        } else {
            System.out.println("Pessoa jurídica não encontrada.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}
```

```
Menu:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
4
Tipo (1 - Física, 2 - Jurídica): 1
ID: 19
ID: 19
Nome: Vinicius Pereira
Logradouro: Rua teste
Cidade: Cidade teste
Estado: MA
Telefone: 12345678
Email: teste@gmail.com
CPF: 1112223344
```

```java
private static void excluir(Scanner scanner, PessoaFisicaDAO pfDAO, PessoaJuridicaDAO pjDAO) {
    System.out.print("Tipo (1 - Física, 2 - Jurídica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();
    if (tipo == 1) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (pfDAO.excluir(id)) {
            System.out.println("Pessoa física excluída com sucesso.");
        } else {
            System.out.println("Erro ao excluir pessoa física.");
        }
    } else if (tipo == 2) {
        System.out.print("ID: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        if (pjDAO.excluir(id)) {
            System.out.println("Pessoa jurídica excluída com sucesso.");
        } else {
            System.out.println("Erro ao excluir pessoa jurídica.");
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}
```

```
Menu:
1. Incluir
2. Alterar
3. Excluir
4. Exibir pelo ID
5. Exibir todos
3
Tipo (1 - Física, 2 - Jurídica): 1
ID: 19
Pessoa física excluída com sucesso.
```

```java
private static void exibirTodos(Scanner scanner, PessoaFisicaDAO pfDAO, PessoaJuridicaDAO pjDAO) {
    System.out.print("Tipo (1 - Física, 2 - Jurídica): ");
    int tipo = scanner.nextInt();
    scanner.nextLine();

    if (tipo == 1) {
        List<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
        for (PessoaFisica pessoa : pessoasFisicas) {
            pessoa.exibir();
        }
    } else if (tipo == 2) {
        List<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
        for (PessoaJuridica pessoa : pessoasJuridicas) {
            pessoa.exibir();
        }
    } else {
        System.out.println("Tipo inválido.");
    }
}
```

Análise e Conclusão:

1. Quais as diferenças entre a persistência em arquivo e a persistência em banco de dados?

- **Persistência em Arquivo**:
  - Os dados são armazenados em arquivos no sistema de arquivos.
  - A estrutura dos dados pode ser simples (como texto ou CSV) ou complexa (como JSON, XML ou formatos binários).
  - A organização dos dados e o acesso dependem do formato do arquivo e das convenções usadas na aplicação.
- **Persistência em Banco de Dados**:
  - Os dados são armazenados em tabelas dentro de um banco de dados relacional ou em documentos/coleções em um banco de dados NoSQL.
  - A estrutura dos dados é definida por esquemas (no caso de bancos de dados relacionais) ou por documentos flexíveis (no caso de bancos de dados NoSQL).
  - Bancos de dados fornecem mecanismos robustos para a organização e acesso aos dados.

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

O uso de expressões lambda e a API de Streams no Java 8 e versões posteriores tornaram operações comuns, como a impressão de valores de uma coleção, muito mais simples e intuitivas. Isso promoveu um estilo de programação mais funcional e expressivo no ecossistema Java, melhorando tanto a produtividade dos desenvolvedores quanto a legibilidade do código.

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

Métodos chamados diretamente pelo método **main** precisam ser marcados como **static** porque **main** é um método estático e só pode interagir diretamente com outros membros estáticos. Isso é um aspecto fundamental do design do Java que assegura que o ponto de entrada da aplicação possa ser executado sem necessidade de criar instâncias de classes, permitindo a inicialização simples e direta do programa.

Repositório github: https://github.com/VascoMay/CadastroBD