Backend sem banco não tem

Walber do Carmo Farias, 202303752547

Polo de Abaetetuba (PA)
Nivel 3 – Backend sem banco não tem – 2023.1 – 2024.1

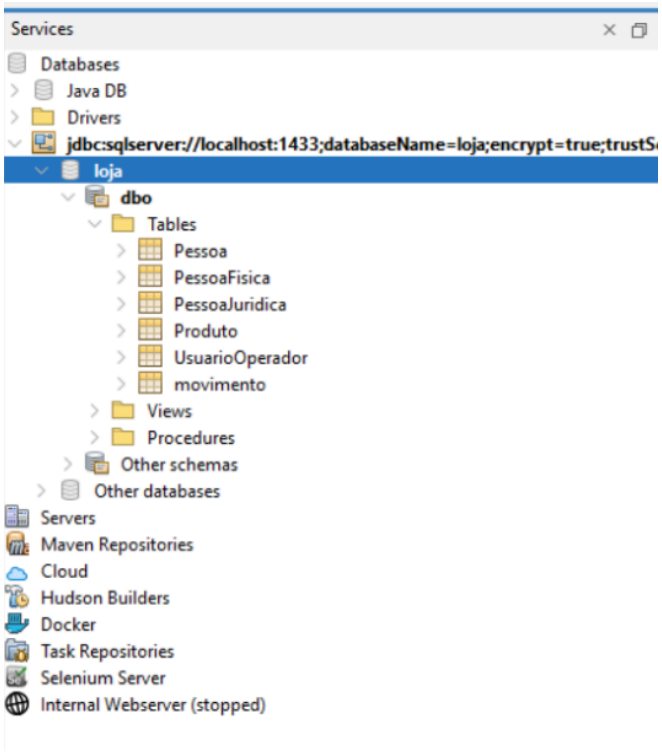
Objetivo da Prática

O objetivo desta missão foi implementar um sistema cadastral com uso de SQL Server e persistência dos dados utilizando middleware JDBC, além de ser feito o mapeamento objeto-relacional e a utilização do padrão DAO (Data Access Object) para o acesso aos dados.

1º Procedimento | Mapeamento Objeto-Relacional e DAO

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 1º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

- Configurando o acesso ao banco de dados SQL Server por meio da aba de serviços do NetBeans:



- Criação da classe Pessoa com os atributos **id, nome, logradouro, cidade, estado, telefone e email**. Além dos **métodos construtores** (padrão e completo) e método **exibir()** para mostrar os dados de forma organizada no console.

```java
 */
package cadastrobd.model;

/**
 *
 * @author Walber
 */
public class Pessoa {
    private int id;
    private String nome;
    private String logradouro;
    private String cidade;
    private String estado;
    private String telefone;
    private String email;

    public int getId() {
        return id;
    }

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

    public String getNome() {
        return nome;
    }

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

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

```java

    public String getTelefone() {
        return telefone;
    }

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

    public String getEmail() {
        return email;
    }

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

    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() {
    }

    public void exibir() {
        System.out.println("id: " + id + "\n nome: " + nome + "\n logradouro: " + logradouro + "\n cidade: " + cidade + "\n
    }

}
```

- Criação da classe Pessoa Física herdando da classe Pessoa todos os atributos e métodos. Além de conter a declaração do atributo específico '**cpf**', os **construtores (padrão e completo)** e a sobrescrita do **método exibir** utilizando a anotação @*Override*

```
5   package cadastrobd.model;
6
7   /**
8    *
9    * @author Walber
10   */
11  public class PessoaFisica extends Pessoa{
12  private String cpf;
13
14      public String getCpf() {
15          return cpf;
16      }
17
18      public void setCpf(String cpf) {
19          this.cpf = cpf;
20      }
21
22      public PessoaFisica() {
23      }
24
25      public PessoaFisica(String cpf, int id, String nome, String logradouro,
26          String cidade, String estado, String telefone, String email) {
27          super(id, nome, logradouro, cidade, estado, telefone, email);
28          this.cpf = cpf;
29      }
30
31      @Override
33      public void exibir() {
34          super.exibir();
35          System.out.println(" cpf: "+ cpf);
36      }
37
38  }
```

- Criação da classe Pessoa Jurídica herdando da classe Pessoa todos os atributos e métodos. Além de conter a declaração do atributo específico '**cnpj**', os **construtores (padrão e completo)** e a sobrescrita do **método exibir** utilizando a anotação @*Override*

```java
package cadastrobd.model;

/**
 *
 * @author Walber
 */
public class PessoaJuridica extends Pessoa {
    private String cnpj;

    public String getCnpj() {
        return cnpj;
    }

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

    public PessoaJuridica() {
    }

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

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

}
```

- Criação da classe ConectorDB contendo o método ***getConnection*** o qual retorna uma conexão utilizando como parâmetros a URL do banco de dados, o usuário e a senha. O método ***getPrepared*** o qual retorna um objeto PreparedStatement utilizando como parâmetro uma *String sql* para retornar o objeto PreparedStatement. E por fim, o método ***getSelect***, que utiliza como parâmetro um objeto *PreparedStatement ps* para realizar a execução do comando no banco de dados por meio do método *executeQuery* e retornar um ResultSet com os dados consultados.

```java
package cadastro.model.util;

import java.sql.DriverManager;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
/**
 *
 * @author Walber
 */
public class ConectorBD {

static  String usuario = "loja";
static  String senha = "loja";
static  String url = "jdbc:sqlserver://localhost:1433;databaseName=loja;trustServerCertificate=true;";
    public static Connection getConnection() throws SQLException{
        return DriverManager.getConnection(url, usuario, senha);
    }
    public static PreparedStatement getPrepared(String sql) throws SQLException{
        Connection conn=getConnection();
        PreparedStatement ps=conn.prepareStatement(sql);
        return ps;
    }
    public static ResultSet getSelect(PreparedStatement ps) throws SQLException{
        ResultSet rs=ps.executeQuery();
        return rs;
    }
}
```

- Criação da Classe Sequence Manager contendo o método ***getValue*** para buscar o próximo valor de uma sequência, o qual utiliza como parâmetro uma *String nomeSequence* para utilizar na consulta ao banco.

```java
package cadastro.model.util;
import  java.sql.Connection;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
/**
 *
 * @author Walber
 */
public class SequenceManager {
    public static int getValue(String nomeSequence) throws SQLException{
        String sql="select next value for "+ nomeSequence+" as proximoValor";
        Connection conn=ConectorBD.getConnection();
        PreparedStatement ps=ConectorBD.getPrepared(sql);
        ResultSet rs=ConectorBD.getSelect(ps);
        if(rs.next()){
            int proximoValor=rs.getInt("proximoValor");
            return proximoValor;

        }
        return -1;
    }
}

```

- Criação da classe PessoaFisicaDAO contendo os métodos **getPessoa** o qual retorna uma PessoaFisica a partir de um *int id* recebido por parâmetro. Método **getPessoas** o qual retorna um objeto ArrayList<PessoaFisica> contendo todas as pessoas físicas cadastradas no banco. Método incluir que recebe uma

PessoaFisica pf como parâmetro e inclui no banco. Método **alterar** que recebe uma *PessoaFisica pf* como parâmetro e realiza alteração no banco por meio de seu ID. Método excluir que realiza a remoção dos registros de uma PessoaFisica em ambas tabelas por meio de um *int id* recebido no parâmetro.

```
19  public class PessoaFisicaDAO {
20
21      public void inserir(PessoaFisica pf) throws SQLException {
22          int idPessoa = SequenceManager.getValue("idPessoa_Sequence");
23          int idPessoaFisica = SequenceManager.getValue("idPessoaFisica");
24          if (idPessoa != -1 && idPessoaFisica != -1) {
25              String sqlPessoa = "insert into Pessoa(idPessoa,nome,logradouro, cidade, estado, telefone, email) values(?,?,?
27              Connection conn = ConectorBD.getConnection();
28              PreparedStatement psPessoa = ConectorBD.getPrepared(sqlPessoa);
29              psPessoa.setInt(1, idPessoa);
30              psPessoa.setString(2, pf.getNome());
31              psPessoa.setString(3, pf.getLogradouro());
32              psPessoa.setString(4, pf.getCidade());
33              psPessoa.setString(5, pf.getEstado());
34              psPessoa.setString(6, pf.getTelefone());
35              psPessoa.setString(7, pf.getEmail());
36              psPessoa.execute();
37
38              String sqlPessoaFisica = "insert into PessoaFisica(idPessoaFisica, cpf, idPessoa)values(?,?,?)";
39              PreparedStatement psPessoaFisica = ConectorBD.getPrepared(sqlPessoaFisica);
40              psPessoaFisica.setInt(1, idPessoaFisica);
41              psPessoaFisica.setString(2, pf.getCpf());
42              psPessoaFisica.setInt(3, idPessoa);
43              psPessoaFisica.execute();
44              psPessoa.close();
45              psPessoaFisica.close();
46              conn.close();
47          }
48
49      public void excluir(int id) throws SQLException {
50          String sqlPessoaFisica = "delete from PessoaFisica where idPessoa=?";
52          Connection conn = ConectorBD.getConnection();
53          PreparedStatement psPessoaFisica = ConectorBD.getPrepared(sqlPessoaFisica);
54          psPessoaFisica.setInt(1, id);
          psPessoaFisica.execute();
```

```
49      public void excluir(int id) throws SQLException {
50          String sqlPessoaFisica = "delete from PessoaFisica where idPessoa=?";
52          Connection conn = ConectorBD.getConnection();
53          PreparedStatement psPessoaFisica = ConectorBD.getPrepared(sqlPessoaFisica);
54          psPessoaFisica.setInt(1, id);
55          psPessoaFisica.execute();
56
57          String sqlPessoa = "delete from Pessoa where idPessoa=?";
58
59          PreparedStatement psPessoa = ConectorBD.getPrepared(sqlPessoa);
60          psPessoa.setInt(1, id);
61          psPessoa.execute();
62
63          psPessoa.close();
64          psPessoaFisica.close();
65          conn.close();
66      }
67      public void alterar(PessoaFisica pf) throws SQLException {
68          String sqlPessoa = "update Pessoa set nome=?,logradouro=?, cidade=?,estado=?,telefone=?,email=? where idPessoa=?";
70          Connection conn = ConectorBD.getConnection();
71          PreparedStatement psPessoa = ConectorBD.getPrepared(sqlPessoa);
72          psPessoa.setString(1, pf.getNome());
73          psPessoa.setString(2, pf.getLogradouro());
74          psPessoa.setString(3, pf.getCidade());
75          psPessoa.setString(4, pf.getEstado());
76          psPessoa.setString(5, pf.getTelefone());
77          psPessoa.setString(6, pf.getEmail());
78          psPessoa.setInt(7, pf.getId());
79
80          psPessoa.execute();
81          String sqlPessoaFisica = "update PessoaFisica set cpf=? where idPessoa=?";
82          PreparedStatement psPessoaFisica = ConectorBD.getPrepared(sqlPessoaFisica);
83
            psPessoaFisica.setString(1, pf.getCpf());
```

```
82
83          psPessoaFisica.setString(1, pf.getCpf());
84
85          psPessoaFisica.setInt(2, pf.getId());
86
87          psPessoaFisica.execute();
88          psPessoa.close();
89          psPessoaFisica.close();
90          conn.close();
91      }
92
93      public PessoaFisica getPessoa(int id) throws SQLException {
94          String sql = "select * from Pessoa p inner join PessoaFisica pf on p.idPessoa=pf.idPessoa where p.id=?";
96          Connection conn = ConectorBD.getConnection();
97          PreparedStatement ps = ConectorBD.getPrepared(sql);
98          ps.setInt(1, id);
99          ResultSet rs = ConectorBD.getSelect(ps);
100         PessoaFisica pf = null;
102         if (rs.next() == true) {
103             int idPessoa = rs.getInt("idPessoa");
104             String nome = rs.getString("nome");
105             String logradouro = rs.getString("logradouro");
106             String cidade = rs.getString("cidade");
107             String estado = rs.getString("estado");
108             String telefone = rs.getString("telefone");
109             String email = rs.getString("email");
110             String cpf = rs.getString("cpf");
111             pf = new PessoaFisica(cpf, id, nome, logradouro, cidade, estado, telefone, email);
112
113         }
114         return pf;
115     }
116
117     public ArrayList<PessoaFisica> getPessoas() throws SQLException {
118         String sql = "select * from Pessoa p inner join PessoaFisica pf on p.idPessoa=pf.idPessoa ";
120         Connection conn = ConectorBD.getConnection();
```

```
109             pf = new PessoaFisica(cpf, id, nome, logradouro, cidade, estado, telefone, email);
110
111         }
112         return pf;
113     }
114
115     public ArrayList<PessoaFisica> getPessoas() throws SQLException {
116         String sql = "select * from Pessoa p inner join PessoaFisica pf on p.idPessoa=pf.idPessoa ";
118         Connection conn = ConectorBD.getConnection();
119         PreparedStatement ps = ConectorBD.getPrepared(sql);
120         ResultSet rs = ConectorBD.getSelect(ps);
121         PessoaFisica pf = null;
122         ArrayList<PessoaFisica> pessoas = new ArrayList();
123         while (rs.next() == true) {
124             int idPessoa = rs.getInt("idPessoa");
125             String nome = rs.getString("nome");
126             String logradouro = rs.getString("logradouro");
127             String cidade = rs.getString("cidade");
128             String estado = rs.getString("estado");
129             String telefone = rs.getString("telefone");
130             String email = rs.getString("email");
131             String cpf = rs.getString("cpf");
132             pf = new PessoaFisica(cpf, idPessoa, nome, logradouro, cidade, estado, telefone, email);
133             pessoas.add(pf);
134
135         }
136         return pessoas;
137     }
}
```

- Criação da classe PessoaJuridicaDAO contendo os métodos **getPessoa** o qual retorna uma PessoaJuridica a partir de um *int id* recebido por parâmetro. Método **getPessoas** o qual retorna um objeto ArrayList<PessoaJuridica > contendo todas as pessoas físicas cadastradas no banco. Método incluir que recebe uma PessoaJuridica *pj* como parâmetro e inclui no banco. Método **alterar** que recebe uma PessoaJuridica *pj* como parâmetro e realiza alteração no banco por meio de

seu ID. Método excluir que realiza a remoção dos registros de uma PessoaJuridica em ambas tabelas por meio de um *int id* recebido no parâmetro.

```
19  public class PessoaJuridicaDAO {
20      public void inserir(PessoaJuridica pj) throws SQLException {
21          int idPessoa = SequenceManager.getValue("idPessoa_Sequence");
22          int idPessoaJuridica = SequenceManager.getValue("idPessoaJuridica");
23          if (idPessoa != -1 && idPessoaJuridica != -1) {
24              String sqlPessoa = "insert into Pessoa(idPessoa,nome,logradouro, cidade, estado, telefone, email) values(?,?,'
25              Connection conn = ConectorBD.getConnection();
26              PreparedStatement psPessoa = ConectorBD.getPrepared(sqlPessoa);
27              psPessoa.setInt(1, idPessoa);
28              psPessoa.setString(2, pj.getNome());
29              psPessoa.setString(3, pj.getLogradouro());
30              psPessoa.setString(4, pj.getCidade());
31              psPessoa.setString(5, pj.getEstado());
32              psPessoa.setString(6, pj.getTelefone());
33              psPessoa.setString(7, pj.getEmail());
34              psPessoa.execute();
35
36              String sqlPessoaJuridica = "insert into PessoaJuridica(idPessoaJuridica, cnpj, idPessoa)values(?,?,?)";
37              PreparedStatement psPessoaJuridica = ConectorBD.getPrepared(sqlPessoaJuridica);
38              psPessoaJuridica.setInt(1, idPessoaJuridica);
39              psPessoaJuridica.setString(2, pj.getCnpj());
40              psPessoaJuridica.setInt(3, idPessoa);
41              psPessoaJuridica.execute();
42              psPessoa.close();
43              psPessoaJuridica.close();
44              conn.close();
45          }
46      }
47
48      public void excluir(int id) throws SQLException {
49          String sqlPessoaJuridica = "delete from PessoaJuridica where idPessoa=?";
50          Connection conn = ConectorBD.getConnection();
51          PreparedStatement psPessoaJuridica = ConectorBD.getPrepared(sqlPessoaJuridica);
52          psPessoaJuridica.setInt(1, id);
53          psPessoaJuridica.execute();
54
```

```
55              String sqlPessoa = "delete from Pessoa where idPessoa=?";
56
57              PreparedStatement psPessoa = ConectorBD.getPrepared(sqlPessoa);
58              psPessoa.setInt(1, id);
59              psPessoa.execute();
60
61              psPessoa.close();
62              psPessoaJuridica.close();
63              conn.close();
64      }
65
66      public void alterar(PessoaJuridica pj) throws SQLException {
67          String sqlPessoa = "update Pessoa set nome=?,logradouro=?, cidade=?,estado=?,telefone=?,email=? where idPessoa=?";
68          Connection conn = ConectorBD.getConnection();
69          PreparedStatement psPessoa = ConectorBD.getPrepared(sqlPessoa);
70          psPessoa.setString(1, pj.getNome());
71          psPessoa.setString(2, pj.getLogradouro());
72          psPessoa.setString(3, pj.getCidade());
73          psPessoa.setString(4, pj.getEstado());
74          psPessoa.setString(5, pj.getTelefone());
75          psPessoa.setString(6, pj.getEmail());
76          psPessoa.setInt(7, pj.getId());
77
78          psPessoa.execute();
79          String sqlPessoaJuridica = "update PessoaJuridica set cnpj=? where idPessoa=?";
80          PreparedStatement psPessoaJuridica = ConectorBD.getPrepared(sqlPessoaJuridica);
81
82          psPessoaJuridica.setString(1, pj.getCnpj());
83
84          psPessoaJuridica.setInt(2, pj.getId());
85
86          psPessoaJuridica.execute();
87          psPessoa.close();
88          psPessoaJuridica.close();
89          conn.close();
90      }
```

```
91
92      public PessoaJuridica getPessoa(int id) throws SQLException {
93          String sql = "select * from Pessoa p inner join PessoaJuridica pj on p.idPessoa=pj.idPessoa where p.id=?";
94          Connection conn = ConectorBD.getConnection();
95          PreparedStatement ps = ConectorBD.getPrepared(sql);
96          ps.setInt(1, id);
97          ResultSet rs = ConectorBD.getSelect(ps);
98          PessoaJuridica pj = null;
99          if (rs.next() == true) {
100             int idPessoa = rs.getInt("idPessoa");
101             String nome = rs.getString("nome");
102             String logradouro = rs.getString("logradouro");
103             String cidade = rs.getString("cidade");
104             String estado = rs.getString("estado");
105             String telefone = rs.getString("telefone");
106             String email = rs.getString("email");
107             String cnpj = rs.getString("cnpj");
108             pj = new PessoaJuridica(cnpj, id, nome, logradouro, cidade, estado, telefone, email);
109
110         }
111         return pj;
112     }
113
114     public ArrayList<PessoaJuridica> getPessoas() throws SQLException {
115         String sql = "select * from Pessoa p inner join PessoaJuridica pj on p.idPessoa=pj.idPessoa ";
116         Connection conn = ConectorBD.getConnection();
117         PreparedStatement ps = ConectorBD.getPrepared(sql);
118         ResultSet rs = ConectorBD.getSelect(ps);
119         PessoaJuridica pj = null;
120         ArrayList<PessoaJuridica> pessoas = new ArrayList();
121         while (rs.next() == true) {
122             int idPessoa = rs.getInt("idPessoa");
123             String nome = rs.getString("nome");
124             String logradouro = rs.getString("logradouro");
125             String cidade = rs.getString("cidade");
126             String estado = rs.getString("estado");
```

```
public ArrayList<PessoaJuridica> getPessoas() throws SQLException {
    String sql = "select * from Pessoa p inner join PessoaJuridica pj on p.idPessoa=pj.idPessoa ";
    Connection conn = ConectorBD.getConnection();
    PreparedStatement ps = ConectorBD.getPrepared(sql);
    ResultSet rs = ConectorBD.getSelect(ps);
    PessoaJuridica pj = null;
    ArrayList<PessoaJuridica> pessoas = new ArrayList();
    while (rs.next() == true) {
        int idPessoa = rs.getInt("idPessoa");
        String nome = rs.getString("nome");
        String logradouro = rs.getString("logradouro");
        String cidade = rs.getString("cidade");
        String estado = rs.getString("estado");
        String telefone = rs.getString("telefone");
        String email = rs.getString("email");
        String cnpj = rs.getString("cnpj");
        pj = new PessoaJuridica(cnpj, idPessoa, nome, logradouro, cidade, estado, telefone, email);
        pessoas.add(pj);

    }
    return pessoas;
}
```

- Criação da classe CadastroBD contendo as ações solicitadas, sendo elas:
    - instanciar e persistir;
    - alterar os dados e salvá-los no banco de dados;
    - consultar todas as pessoas e listá-las;
    - excluir uma pessoa criada anteriormente no banco;
    - Obs: todos essas ações tanto para pessoa física quanto para pessoa jurídica.

```
    public static void main(String[] args) throws SQLException {
        PessoaFisica pf=new PessoaFisica("1111111111", 0, "Walber", "Rua lauro sodre", "abaetetuba", "Pa","1234-456", "walb
        PessoaFisicaDAO pfDAO=new PessoaFisicaDAO();
        pfDAO.inserir(pf);
        System.out.println("Pessoa Fisica Inserida");
        PessoaFisica pfEditada=new PessoaFisica("22222222", 5,"Lucca", "Rua lauro sobre","Abaetetuba", "Pa", "33456578", "w
        pfDAO.alterar(pfEditada);
        System.out.println("Pessoa Fisica Alterada");
        ArrayList<PessoaFisica> pessoasFisicas = pfDAO.getPessoas();
        for (int i = 0; i < pessoasFisicas.size(); i++) {
            pessoasFisicas.get(i).exibir();

        }
        pfDAO.excluir(4);
        System.out.println("Pessoa Fisica Excluida");

        PessoaJuridica pj = new PessoaJuridica("33333333333333", 0, "Fintech","Rua lauro sodre", "abaetetuba", "Pa","1234-4
        PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();
        pjDAO.inserir(pj);
        System.out.println("Pessoa Juridica Inserida");
        PessoaJuridica pjEditada = new PessoaJuridica("44444444444444444", 1,"Inovation 2D", "Rua lauro sobre","Abaetetuba
        pjDAO.alterar(pjEditada);
        System.out.println("Pessoa Juridica alterada");
        ArrayList<PessoaJuridica> pessoasJuridicas = pjDAO.getPessoas();
        for (int i = 0; i < pessoasJuridicas.size(); i++) {
            pessoasJuridicas.get(i).exibir();

        }
        pjDAO.excluir(14);
        System.out.println("Pessoa Juridica excluida");
    }

}
```

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

Resposta:

- São essenciais para simplificar o desenvolvimento dos softwares, pois abstraem a complexidade promovendo portabilidade, segurança e agindo como uma camada intermediária, o que facilita a comunicação entre diferentes partes do sistema.

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

Resposta:

- O Statement é adequado para consultas simples e com valores concatenados.O PreparedStatement é mais seguro e eficiente para consultas parametrizadas, utilizando marcadores de posição “ ? ”, especialmente em ambientes onde a segurança e o desempenho são prioritárias.

c) Como o padrão DAO melhora a manutenibilidade do software?

Resposta:

- O Statement é adequado para consultas simples e com valores concatenados.O PreparedStatement é mais seguro e eficiente para

consultas parametrizadas, utilizando marcadores de posição “ ? ”, especialmente em ambientes onde a segurança e o desempenho são prioritárias.

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

Resposta:

- Pode ser refletida de duas maneiras : Herança por tabela separada em que cada classe da hierarquia tem sua própria tabela e herança por tabela juntas em que todas as classes da hierarquia compartilham uma única tabela.

2º Procedimento | Alimentando a Base

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

- Alteração da classe **main** para a implementação de cadastro em modo texto, contendo os métodos: **incluir, alterar, excluir, obter, obterTodos e sair**. Todos os métodos contém a seleção de escolha de Pessoa ‘F - Fisica ou J - Jurídica’.

```java
public class CadastroBD {

    static PessoaFisicaDAO pfDAO = new PessoaFisicaDAO();
    static PessoaJuridicaDAO pjDAO = new PessoaJuridicaDAO();

    public static void main(String[] args) throws SQLException {
        menu();
    }

    public static void menu() throws SQLException {
        System.out.println("==================\n 1-Incluir Pessoa\n 2-Alterar Pessoa \n 3-Excluir Pessoa \n 4-Buscar pe
        Scanner leitor = new Scanner(System.in);
        int opcaoDoUsuario = leitor.nextInt();
        if (opcaoDoUsuario >= 1 && opcaoDoUsuario <= 5) {

            System.out.println(" F-Pessoa Fisica \n J-Pessoa Juridica");
            String opcaoDoUsuario2 = leitor.next();
            if (opcaoDoUsuario == 1 && opcaoDoUsuario2.equals("F")) {
                incluirPessoaFisica();
                menu();
            } else if (opcaoDoUsuario == 1 && opcaoDoUsuario2.equals("J")) {
                incluirPessoaJuridica();
                menu();
            } else if (opcaoDoUsuario == 2 && opcaoDoUsuario2.equals("F")) {
                alterarPessoaFisica();
                menu();
            } else if (opcaoDoUsuario == 2 && opcaoDoUsuario2.equals("J")) {
                alterarPessoaJuridica();
                menu();
            } else if (opcaoDoUsuario == 3 && opcaoDoUsuario2.equals("F")) {
                excluirPessoaFisica();
                menu();
            } else if (opcaoDoUsuario == 3 && opcaoDoUsuario2.equals("J")) {
                excluirPessoaJuridica();
                menu();
            } else if (opcaoDoUsuario == 4 && opcaoDoUsuario2.equals("F")) {


    }

    public static void incluirPessoaFisica() throws SQLException {
        Scanner leitor = new Scanner(System.in);
        System.out.println("nome:");
        String nomeProprio = leitor.next();
        System.out.println("logradouro:");
        String logradouro = leitor.next();
        System.out.println("cidade:");
        String cidade = leitor.next();
        System.out.println("estado:");
        String estado = leitor.next();
        System.out.println("telefone");
        String telefone = leitor.next();
        System.out.println("email:");
        String email = leitor.next();
        System.out.println("CPF:");
        String cpf = leitor.next();

        PessoaFisica pessoa = new PessoaFisica(cpf, 0, nomeProprio, logradouro, cidade, estado, telefone, email);
        pfDAO.inserir(pessoa);
        System.out.println("Pessoa Fisica incluida");
    }
```

```java
public static void incluirPessoaJuridica() throws SQLException {

    Scanner leitor = new Scanner(System.in);
    System.out.println("nome:");
    String nomeProprio = leitor.next();
    System.out.println("logradouro:");
    String logradouro = leitor.next();
    System.out.println("cidade:");
    String cidade = leitor.next();
    System.out.println("estado:");
    String estado = leitor.next();
    System.out.println("telefone");
    String telefone = leitor.next();
    System.out.println("email:");
    String email = leitor.next();
    System.out.println("CNPJ:");
    String cnpj = leitor.next();
    PessoaJuridica pessoa = new PessoaJuridica(cnpj, 0, nomeProprio, logradouro, cidade, estado, telefone, email);
    pjDAO.inserir(pessoa);
    System.out.println("Pessoa Juridica incluida");
}
```

```java
public static void alterarPessoaFisica() throws SQLException {
    System.out.println("id:");
    Scanner leitor = new Scanner(System.in);
    int numeroId = leitor.nextInt();
    PessoaFisica pessoa = pfDAO.getPessoa(numeroId);
    pessoa.exibir();
    System.out.println("nome:");
    String nomeProprio = leitor.next();
    System.out.println("logradouro:");
    String logradouro = leitor.next();
    System.out.println("cidade:");
    String cidade = leitor.next();
    System.out.println("estado:");
    String estado = leitor.next();
    System.out.println("telefone");
    String telefone = leitor.next();
    System.out.println("email:");
    String email = leitor.next();
    System.out.println("CPF:");
    String cpf = leitor.next();

    PessoaFisica pessoaAlterada = new PessoaFisica(cpf, numeroId, nomeProprio, logradouro, cidade, estado, telefone,
    pfDAO.alterar(pessoaAlterada);
    System.out.println(" A pessoa foi alterada");

}
```

```java
public static void alterarPessoaJuridica() throws SQLException {
    System.out.println("id:");
    Scanner leitor = new Scanner(System.in);
    int numeroId = leitor.nextInt();
    PessoaJuridica pessoa = pjDAO.getPessoa(numeroId);
    pessoa.exibir();
    System.out.println("nome:");
    String nomeProprio = leitor.next();
    System.out.println("logradouro:");
    String logradouro = leitor.next();
    System.out.println("cidade:");
    String cidade = leitor.next();
    System.out.println("estado:");
    String estado = leitor.next();
    System.out.println("telefone");
    String telefone = leitor.next();
    System.out.println("email:");
    String email = leitor.next();
    System.out.println("CNPJ:");
    String cnpj = leitor.next();
    PessoaJuridica pessoaAlterada = new PessoaJuridica(cnpj, numeroId, nomeProprio, logradouro, cidade, estado, telef
    pjDAO.alterar(pessoaAlterada);
    System.out.println(" A pessoa foi alterada");
}
```

```java
public static void excluirPessoaFisica() throws SQLException {
    System.out.println("id:");
    Scanner leitor = new Scanner(System.in);
    int numeroId = leitor.nextInt();
    pfDAO.excluir(numeroId);
    System.out.println(" A pessoa foi excluida");
}

public static void excluirPessoaJuridica() throws SQLException {
    System.out.println("id:");
    Scanner leitor = new Scanner(System.in);
    int numeroId = leitor.nextInt();
    pjDAO.excluir(numeroId);
    System.out.println(" A pessoa foi excluida");
}

public static void obterPeloIdPessoaFisica() throws SQLException {
    System.out.println("id:");
    Scanner leitor = new Scanner(System.in);
    int numeroId = leitor.nextInt();
    PessoaFisica pessoa = pfDAO.getPessoa(numeroId);
    pessoa.exibir();

}

public static void obterPeloIdPessoaJuridica() throws SQLException {
    System.out.println("id:");
    Scanner leitor = new Scanner(System.in);
    int numeroId = leitor.nextInt();
    PessoaJuridica pessoa = pjDAO.getPessoa(numeroId);
    pessoa.exibir();
}

public static void obterTodosPessoaFisica() throws SQLException {
    ArrayList<PessoaFisica> pessoas = pfDAO.getPessoas();
    for (int i = 0; i < pessoas.size(); i++) {
        pessoas.get(i).exibir();

    }

}

public static void obterTodosPessoaJuridica() throws SQLException {
    ArrayList<PessoaJuridica> pessoas = pjDAO.getPessoas();
    for (int i = 0; i < pessoas.size(); i++) {
        pessoas.get(i).exibir();
    }
}
```

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

Resposta:

- Persistência em arquivo: Dados armazenados em arquivos simples (binários, JSON, etc) no sistema de arquivos, com organização menos estruturada e com capacidade de consulta limitada. Persistência em banco de dados: Dados armazenados em bancos de dados estruturados, permitindo consultas complexas, controle de transações e melhor gerenciamento de dados em sistemas robustos.

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

Resposta:

- O uso de operadores lambda simplificou a impressão dos valores das entidades em coleções no Java, reduzindo a quantidade de código necessária e tornando-o mais legível e expressivo.

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

Resposta:

- Métodos acionados diretamente pelo método main em Java precisam ser marcados como static porque o main é estático e pode ser chamado sem criar uma instância da classe. O uso do static permite o acesso direto ao método sem a necessidade de uma instância da classe.

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

Conclusão

A implementação deste sistema ajudou não só a entender melhor os conceitos teóricos, mas também a desenvolver habilidades práticas na construção de sistemas backend. Usando uma abordagem de acesso a dados com DAOs, além de uma comunicação eficiente com o banco de dados via JDBC, foi possível criar uma base sólida para desenvolver aplicações que podem crescer e são fáceis de manter. O projeto mostrou claramente a importância de seguir boas práticas de programação e design de software. Ele ajudou a entender melhor os componentes essenciais no desenvolvimento

backend. Esta prática será uma base valiosa para projetos futuros, oferecendo um modelo claro e organizado para construir sistemas complexos e eficientes.