Projeto de Aplicativo Java com Banco SQL Server

Edson Luiz Pacheco Junior - 202308892185

Polo Jardim Goiás
RPG0016 – BackEnd sem banco não tem – 9001 – 2024.3

Objetivo da Prática

Criação de aplicativo Java, com acesso ao banco de dados SQL Server através do middleware JDBC.

1. Implementar persistência com base no middleware JDBC.
2. Utilizar o padrão DAO (Data Access Object) no manuseio de dados.
3. Implementar o mapeamento objeto-relacional em sistemas Java.
4. Criar sistemas cadastrais com persistência em banco relacional.
5. 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

- **Pessoa:**

```
package cadastrobd.model;

/**
*
* @author edson-202308892185
*/
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 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 void exibir() {
        System.out.println("ID: " + id + "\nNome: " + nome + "\nEndereco: " +
logradouro + "\nCidade: " + cidade
                + "\nEstado: " + estado + "\nTelefone: " + telefone + "\nEmail: " +
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;
    }

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

    public String getCidade() {
        return cidade;
    }

    public void setCidade(String cidade) {
```

```
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getEmail() {
        return email;
    }

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

    public String getTelefone() {
        return telefone;
    }

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

}
```

- **PessoaFisica:**

```
package cadastrobd.model;

/**
*
* @author edson-202308892185
*/
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;
    }
```

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

    public String getCpf() {
        return cpf;
    }

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

}
```

- **PessoaJuridica:**

```
package cadastrobd.model;

/**
*
* @author edson-202308892185
*/
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;
    }

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

    public String getCnpj() {
        return cnpj;
    }

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

```
}
```

- **PessoaFisicaDAO:**

```
package cadastrobd.model;

/**
*
* @author edson-202308892185
*/
import cadastrobd.model.util.ConectorBD;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import cadastrobd.model.util.SequenceManager;

public class PessoaFisicaDAO {

   public PessoaFisica getPessoa(int id) throws SQLException {
      String sql = "SELECT * FROM pessoa p JOIN pessoa_fisica pf ON p.idPessoa = pf.idPessoa WHERE p.idPessoa = ?";
      try (Connection conn = ConectorBD.getConnection(); PreparedStatement stmt = ConectorBD.getPrepared(conn, sql)) {
         stmt.setInt(1, id);
         ResultSet rs = stmt.executeQuery();
         if (rs.next()) {
            return new PessoaFisica(
                  rs.getInt("idPessoa"),
                  rs.getString("nome"),
                  rs.getString("logradouro"),
                  rs.getString("cidade"),
                  rs.getString("estado"),
                  rs.getString("telefone"),
                  rs.getString("email"),
                  rs.getString("cpf")
            );
         }
      }
      return null;
   }

   public List<PessoaFisica> getPessoas() throws SQLException {
      List<PessoaFisica> pessoas = new ArrayList<>();
      String sql = "SELECT * FROM pessoa p JOIN pessoa_fisica pf ON p.idPessoa = pf.idPessoa";
      try (Connection conn = ConectorBD.getConnection(); PreparedStatement stmt = ConectorBD.getPrepared(conn, sql); ResultSet rs = stmt.executeQuery()) {
         while (rs.next()) {
```

```
            pessoas.add(new PessoaFisica(
                    rs.getInt("idPessoa"),
                    rs.getString("nome"),
                    rs.getString("logradouro"),
                    rs.getString("cidade"),
                    rs.getString("estado"),
                    rs.getString("telefone"),
                    rs.getString("email"),
                    rs.getString("cpf")
            ));
        }
    }
    return pessoas;
}

public void incluir(PessoaFisica pessoa) throws SQLException {
    String sqlPessoa = "INSERT INTO pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
    String sqlPessoaFisica = "INSERT INTO pessoa_fisica (idPessoaFisica, idPessoa, cpf) VALUES (?, ?, ?)";

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

        int novoIdPessoa = SequenceManager.getValue("seq_pessoa_id", conn);
        pessoa.setId(novoIdPessoa);

        try (PreparedStatement stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa); PreparedStatement stmtFisica = ConectorBD.getPrepared(conn, sqlPessoaFisica)) {

            stmtPessoa.setInt(1, novoIdPessoa);
            stmtPessoa.setString(2, pessoa.getNome());
            stmtPessoa.setString(3, pessoa.getLogradouro());
            stmtPessoa.setString(4, pessoa.getCidade());
            stmtPessoa.setString(5, pessoa.getEstado());
            stmtPessoa.setString(6, pessoa.getTelefone());
            stmtPessoa.setString(7, pessoa.getEmail());
            stmtPessoa.executeUpdate();

            stmtFisica.setInt(1, novoIdPessoa);
            stmtFisica.setInt(2, novoIdPessoa);
            stmtFisica.setString(3, pessoa.getCpf());
            stmtFisica.executeUpdate();

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

```
        }
    }
}

public void alterar(PessoaFisica pessoa) throws SQLException {
    String sqlPessoa = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
    String sqlPessoaFisica = "UPDATE pessoa_fisica SET cpf = ? WHERE idPessoa = ?";

    try (Connection conn = ConectorBD.getConnection()) {
        conn.setAutoCommit(false);
        try (PreparedStatement stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa); PreparedStatement stmtFisica = ConectorBD.getPrepared(conn, sqlPessoaFisica)) {

            stmtPessoa.setString(1, pessoa.getNome());
            stmtPessoa.setString(2, pessoa.getLogradouro());
            stmtPessoa.setString(3, pessoa.getCidade());
            stmtPessoa.setString(4, pessoa.getEstado());
            stmtPessoa.setString(5, pessoa.getTelefone());
            stmtPessoa.setString(6, pessoa.getEmail());
            stmtPessoa.setInt(7, pessoa.getId());
            stmtPessoa.executeUpdate();

            stmtFisica.setString(1, pessoa.getCpf());
            stmtFisica.setInt(2, pessoa.getId());
            stmtFisica.executeUpdate();

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

public void excluir(int id) throws SQLException {
    String sqlPessoaFisica = "DELETE FROM pessoa_fisica WHERE idPessoa = ?";
    String sqlPessoa = "DELETE FROM pessoa WHERE idPessoa = ?";

    try (Connection conn = ConectorBD.getConnection()) {
        conn.setAutoCommit(false);
        try (PreparedStatement stmtFisica = ConectorBD.getPrepared(conn, sqlPessoaFisica); PreparedStatement stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa)) {

            stmtFisica.setInt(1, id);
```

```
            stmtFisica.executeUpdate();

            stmtPessoa.setInt(1, id);
            stmtPessoa.executeUpdate();

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

- **PessoaJuridicaDAO:**

```
package cadastrobd.model;

/**
*
* @author edson-202308892185
*/
import cadastrobd.model.util.ConectorBD;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import cadastrobd.model.util.SequenceManager;

public class PessoaJuridicaDAO {

   public PessoaJuridica getPessoa(int id) throws SQLException {
      String sql = "SELECT * FROM pessoa p JOIN pessoa_juridica pj ON p.idPessoa = pj.idPessoa WHERE p.idPessoa = ?";
      try (Connection conn = ConectorBD.getConnection(); PreparedStatement stmt = ConectorBD.getPrepared(conn, sql)) {
          stmt.setInt(1, id);
          ResultSet rs = stmt.executeQuery();
          if (rs.next()) {
              return new PessoaJuridica(
                      rs.getInt("idPessoa"),
                      rs.getString("nome"),
                      rs.getString("logradouro"),
                      rs.getString("cidade"),
                      rs.getString("estado"),
                      rs.getString("telefone"),
                      rs.getString("email"),
                      rs.getString("cnpj")
              );
          }
      }
```

```
        return null;
    }

    public List<PessoaJuridica> getPessoas() throws SQLException {
        List<PessoaJuridica> pessoas = new ArrayList<>();
        String sql = "SELECT * FROM pessoa p JOIN pessoa_juridica pj ON
p.idPessoa = pj.idPessoa";
        try (Connection conn = ConectorBD.getConnection(); PreparedStatement
stmt = ConectorBD.getPrepared(conn, sql); ResultSet rs = stmt.executeQuery())
{
            while (rs.next()) {
                pessoas.add(new PessoaJuridica(
                        rs.getInt("idPessoa"),
                        rs.getString("nome"),
                        rs.getString("logradouro"),
                        rs.getString("cidade"),
                        rs.getString("estado"),
                        rs.getString("telefone"),
                        rs.getString("email"),
                        rs.getString("cnpj")
                ));
            }
        }
        return pessoas;
    }

    public void incluir(PessoaJuridica pessoa) throws SQLException {
        String sqlPessoa = "INSERT INTO pessoa (idPessoa, nome, logradouro,
cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
        String sqlPessoaJuridica = "INSERT INTO pessoa_juridica
(idPessoaJuridica, idPessoa, cnpj) VALUES (?, ?, ?)";

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

            int novoIdPessoa = SequenceManager.getValue("seq_pessoa_id", conn);
            pessoa.setId(novoIdPessoa);

            try (PreparedStatement stmtPessoa = ConectorBD.getPrepared(conn,
sqlPessoa); PreparedStatement stmtJuridica = ConectorBD.getPrepared(conn,
sqlPessoaJuridica)) {

                stmtPessoa.setInt(1, novoIdPessoa);
                stmtPessoa.setString(2, pessoa.getNome());
                stmtPessoa.setString(3, pessoa.getLogradouro());
                stmtPessoa.setString(4, pessoa.getCidade());
                stmtPessoa.setString(5, pessoa.getEstado());
                stmtPessoa.setString(6, pessoa.getTelefone());
                stmtPessoa.setString(7, pessoa.getEmail());
```

```
            stmtPessoa.executeUpdate();

            stmtJuridica.setInt(1, novoIdPessoa);
            stmtJuridica.setInt(2, novoIdPessoa);
            stmtJuridica.setString(3, pessoa.getCnpj());
            stmtJuridica.executeUpdate();

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

public void alterar(PessoaJuridica pessoa) throws SQLException {
    String sqlPessoa = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade
= ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
    String sqlPessoaJuridica = "UPDATE pessoa_juridica SET cnpj = ? WHERE
idPessoa = ?";

    try (Connection conn = ConectorBD.getConnection()) {
        conn.setAutoCommit(false);
        try (PreparedStatement stmtPessoa = ConectorBD.getPrepared(conn,
sqlPessoa); PreparedStatement stmtJuridica = ConectorBD.getPrepared(conn,
sqlPessoaJuridica)) {

            stmtPessoa.setString(1, pessoa.getNome());
            stmtPessoa.setString(2, pessoa.getLogradouro());
            stmtPessoa.setString(3, pessoa.getCidade());
            stmtPessoa.setString(4, pessoa.getEstado());
            stmtPessoa.setString(5, pessoa.getTelefone());
            stmtPessoa.setString(6, pessoa.getEmail());
            stmtPessoa.setInt(7, pessoa.getId());
            stmtPessoa.executeUpdate();

            stmtJuridica.setString(1, pessoa.getCnpj());
            stmtJuridica.setInt(2, pessoa.getId());
            stmtJuridica.executeUpdate();

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

public void excluir(int id) throws SQLException {
```

```
        String sqlPessoaJuridica = "DELETE FROM pessoa_juridica WHERE idPessoa = ?";
        String sqlPessoa = "DELETE FROM pessoa WHERE idPessoa = ?";

        try (Connection conn = ConectorBD.getConnection()) {
            conn.setAutoCommit(false);
            try (PreparedStatement stmtJuridica = ConectorBD.getPrepared(conn, sqlPessoaJuridica); PreparedStatement stmtPessoa = ConectorBD.getPrepared(conn, sqlPessoa)) {

                stmtJuridica.setInt(1, id);
                stmtJuridica.executeUpdate();

                stmtPessoa.setInt(1, id);
                stmtPessoa.executeUpdate();

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

- **ConectorBD:**

```
package cadastrobd.model.util;

/**
*
* @author edson-202308892185
*/
import java.sql.*;

public class ConectorBD {

    private static final String URL = "jdbc:sqlserver://localhost\\loja:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;";
    private static final String USER = "loja";
    private static final String PASSWORD = "loja";

    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 stmt) throws
SQLException {
        return stmt.executeQuery();
    }

    public static void close(Connection conn) throws SQLException {
        if (conn != null) {
            conn.close();
        }
    }

    public static void close(Statement stmt) throws SQLException {
        if (stmt != null) {
            stmt.close();
        }
    }

    public static void close(ResultSet rs) throws SQLException {
        if (rs != null) {
            rs.close();
        }
    }
}
```

- **SequenceManager:**

```
package cadastrobd.model.util;

/**
*
* @author edson-202308892185
*/
import java.sql.*;

public class SequenceManager {

    public static int getValue(String sequenceName, Connection conn) throws
SQLException {
        String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS
next_val";
        try (PreparedStatement stmt = conn.prepareStatement(sql); ResultSet rs =
stmt.executeQuery()) {
            if (rs.next()) {
                return rs.getInt("next_val");
            } else {
                throw new SQLException("Não foi possível obter o próximo valor da
sequência: " + sequenceName);
            }
```

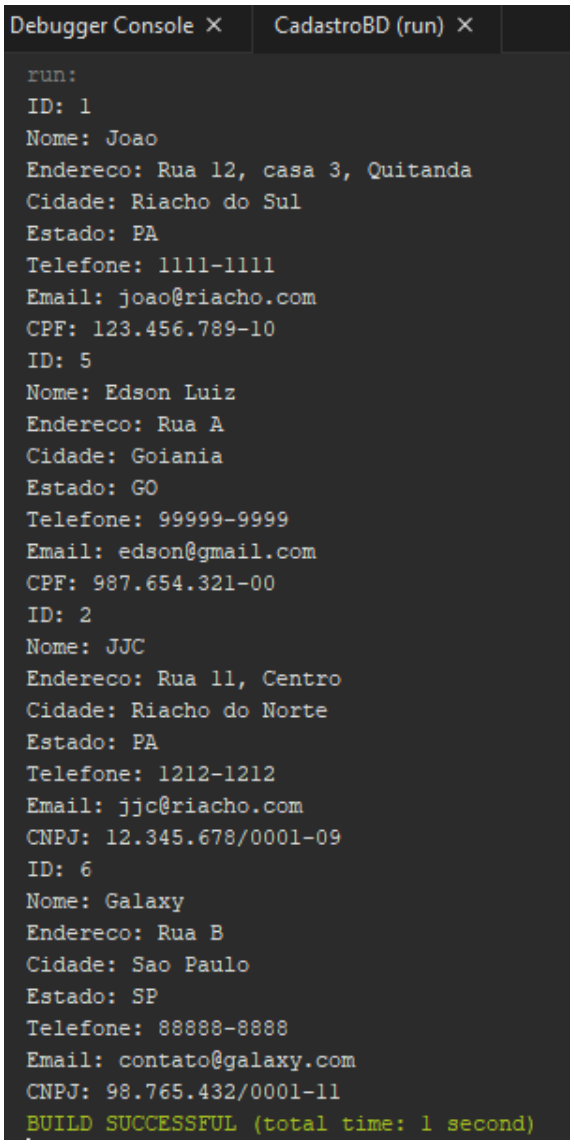
```
        }
    }
}
```

- **CadastroBDTeste:**

```
package cadastrobd;

/**
*
* @author edson-202308892185
*/
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;

import java.util.List;

public class CadastroBDTeste {

    public static void main(String[] args) {
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();

        try {
            PessoaFisica pf = new PessoaFisica(0, "Edson Luiz", "Rua A", "Goiania",
"GO", "99999-9999", "edson@gmail.com", "123.456.789-00");
            pessoaFisicaDAO.incluir(pf);

            pf.setCpf("987.654.321-00");
            pessoaFisicaDAO.alterar(pf);

            List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
            pessoasFisicas.forEach(PessoaFisica::exibir);

            pessoaFisicaDAO.excluir(pf.getId());

            PessoaJuridica pj = new PessoaJuridica(0, "Galaxy", "Rua B", "Sao
Paulo", "SP", "88888-8888", "contato@galaxy.com", "12.345.678/0001-99");
            pessoaJuridicaDAO.incluir(pj);

            pj.setCnpj("98.765.432/0001-11");
            pessoaJuridicaDAO.alterar(pj);

            List<PessoaJuridica>                pessoasJuridicas                =
pessoaJuridicaDAO.getPessoas();
            pessoasJuridicas.forEach(PessoaJuridica::exibir);
```

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

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Execução do código **CadastroBDTeste**:



Conclusão:

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

São muito importantes pois eles ajudam diferentes partes de um sistema (como APIs, banco de dados e outros programas) a se comunicarem de forma mais fácil e organizada. Também são importantes para abstrair a complexidade escondendo detalhes técnicos; boa portabilidade permitindo trocar partes do sistema sem precisar mudar todo o código; facilita a integração entre sistemas diferentes, fazendo com que trabalhem juntos de forma eficiente.

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

A principal diferença é que o PreparedStatement é mais seguro e eficiente. Evita SQL Injection pois os valores são passados separadamente, ele compila a consulta uma vez e reaproveita para várias execuções o que o torna mais eficiente.

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

Ele separa a lógica de acesso do banco de dados da lógica da aplicação, trazendo vantagens como: separação de responsabilidades, facilidade de manutenção, reutilização de código e testes mais fáceis.

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

Tem que ser feita uma adaptação, pois os bancos de dados não suportam herança diretamente. Usamos então uma tabela por subclasse, onde cada classe terá sua tabela e as tabelas das subclasses terão uma chave estrangeira para a tabela da classe base.

2º Procedimento | Alimentando a Base

- **Códigos e resultados**

CadastroBD:

```
package cadastrobd;

/**
*
* @author edson-202308892185
*/
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.PessoaJuridicaDAO;

import java.util.List;
import java.util.Scanner;

public class CadastroBD {
```

```
public static void main(String[] args) {
    PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO();
    PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO();
    Scanner scanner = new Scanner(System.in);

    boolean continuar = true;

    while (continuar) {
        try {
            System.out.println("\n================================");
            System.out.println("1 - Incluir Pessoa");
            System.out.println("2 - Alterar Pessoa");
            System.out.println("3 - Excluir Pessoa");
            System.out.println("4 - Buscar pelo Id");
            System.out.println("5 - Exibir Todos");
            System.out.println("0 - Finalizar Programa");
            System.out.println("================================");
            int opcao = scanner.nextInt();
            scanner.nextLine();

            switch (opcao) {
                case 1 ->
                    incluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                case 2 ->
                    alterar(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                case 3 ->
                    excluir(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                case 4 ->
                    obter(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                case 5 ->
                    obterTodos(scanner, pessoaFisicaDAO, pessoaJuridicaDAO);
                case 0 -> {
```

```
                    continuar = false;
                    System.out.println("Programa finalizado.");
                }
                default ->
                    System.out.println("Opcao invalida. Tente novamente.");
            }
        } catch (Exception e) {
            System.err.println("Erro: " + e.getMessage());
            e.printStackTrace();
        }
    }

    scanner.close();
}

private static void incluir(Scanner scanner, PessoaFisicaDAO pfDAO,
PessoaJuridicaDAO pjDAO) throws Exception {
    char tipo = lerTipo(scanner);

    switch (tipo) {
        case 'F' -> {
            PessoaFisica pf = lerDadosPessoaFisica(scanner);
            pfDAO.incluir(pf);
            System.out.println("Pessoa Fisica incluida com sucesso.");
        }
        case 'J' -> {
            PessoaJuridica pj = lerDadosPessoaJuridica(scanner);
            pjDAO.incluir(pj);
            System.out.println("Pessoa Juridica incluida com sucesso.");
        }
        default ->
            System.out.println("Tipo invalido.");
    }
```

```
    }

    private static void alterar(Scanner scanner, PessoaFisicaDAO pfDAO,
PessoaJuridicaDAO pjDAO) throws Exception {
        char tipo = lerTipo(scanner);

        System.out.print("Informe o ID da pessoa: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        switch (tipo) {
            case 'F':
                PessoaFisica pf = pfDAO.getPessoa(id);
                if (pf != null) {
                    System.out.println("Dados atuais: ");
                    pf.exibir();
                    System.out.println("Informe os novos dados:");
                    PessoaFisica novosDados = lerDadosPessoaFisica(scanner);
                    novosDados.setId(pf.getId());
                    pfDAO.alterar(novosDados);
                    System.out.println("Pessoa Fisica alterada com sucesso.");
                } else {
                    System.out.println("Pessoa Fisica nao encontrada.");
                }
                break;
            case 'J':
                PessoaJuridica pj = pjDAO.getPessoa(id);
                if (pj != null) {
                    System.out.println("Dados atuais: ");
                    pj.exibir();
                    System.out.println("Informe os novos dados:");
                    PessoaJuridica novosDados = lerDadosPessoaJuridica(scanner);
                    novosDados.setId(pj.getId());
```

```
                pjDAO.alterar(novosDados);
                System.out.println("Pessoa Juridica alterada com sucesso.");
            } else {
                System.out.println("Pessoa Juridica nao encontrada.");
            }
            break;
        default:
            System.out.println("Tipo invalido.");
            break;
    }
}

private static void excluir(Scanner scanner, PessoaFisicaDAO pfDAO,
PessoaJuridicaDAO pjDAO) throws Exception {
    char tipo = lerTipo(scanner);

    System.out.print("Informe o ID da pessoa: ");
    int id = scanner.nextInt();
    scanner.nextLine();

    switch (tipo) {
        case 'F' -> {
            pfDAO.excluir(id);
            System.out.println("Pessoa Fisica excluida com sucesso.");
        }
        case 'J' -> {
            pjDAO.excluir(id);
            System.out.println("Pessoa Juridica excluida com sucesso.");
        }
        default ->
            System.out.println("Tipo invalido.");
    }
}
```

```
    private static void obter(Scanner scanner, PessoaFisicaDAO pfDAO,
PessoaJuridicaDAO pjDAO) throws Exception {
        char tipo = lerTipo(scanner);

        System.out.print("Informe o ID da pessoa: ");
        int id = scanner.nextInt();
        scanner.nextLine();

        switch (tipo) {
            case 'F':
                PessoaFisica pf = pfDAO.getPessoa(id);
                if (pf != null) {
                    pf.exibir();
                } else {
                    System.out.println("Pessoa Fisica nao encontrada.");
                }
                break;
            case 'J':
                PessoaJuridica pj = pjDAO.getPessoa(id);
                if (pj != null) {
                    pj.exibir();
                } else {
                    System.out.println("Pessoa Juridica nao encontrada.");
                }
                break;
            default:
                System.out.println("Tipo invalido.");
                break;
        }
    }
```

```
    private static void obterTodos(Scanner scanner, PessoaFisicaDAO pfDAO,
PessoaJuridicaDAO pjDAO) throws Exception {
        char tipo = lerTipo(scanner);

        switch (tipo) {
            case 'F' -> {
                List<PessoaFisica> pessoas = pfDAO.getPessoas();
                pessoas.forEach(PessoaFisica::exibir);
            }
            case 'J' -> {
                List<PessoaJuridica> pessoas = pjDAO.getPessoas();
                pessoas.forEach(PessoaJuridica::exibir);
            }
            default ->
                System.out.println("Tipo invalido.");
        }
    }

    private static char lerTipo(Scanner scanner) {
        char tipo;
        do {
            System.out.print("F - Pessoa Fisica | J - Pessoa Juridica: ");
            String entrada = scanner.nextLine().toUpperCase();

            if (entrada.length() == 1 && (entrada.equals("F") ||
entrada.equals("J"))) {
                tipo = entrada.charAt(0);
                break;
            } else {
                System.out.println("Entrada invalida. Digite apenas 'F' ou 'J'.");
            }
        } while (true);
```

```
    return tipo;
}

private static PessoaFisica lerDadosPessoaFisica(Scanner scanner) {
    System.out.print("Nome: ");
    String nome = scanner.nextLine();
    System.out.print("Logradouro: ");
    String logradouro = scanner.nextLine();
    System.out.print("Cidade: ");
    String cidade = scanner.nextLine();
    String estado;
    do {
        System.out.print("Estado: ");
        estado = scanner.nextLine().toUpperCase();
        if (estado.length() != 2) {
            System.out.println("O estado deve ter exatamente 2 caracteres.");
        }
    } while (estado.length() != 2);
    String telefone;
    do {
        System.out.print("Telefone: ");
        telefone = scanner.nextLine();
        if (telefone.length() > 11) {
            System.out.println("O telefone deve ter no maximo 11 caracteres.");
        }
    } while (telefone.length() > 11);
    System.out.print("Email: ");
    String email = scanner.nextLine();
    String cpf;
    do {
        System.out.print("CPF: ");
        cpf = scanner.nextLine();
        if (cpf.length() > 14) {
```

```
                System.out.println("O CPF deve ter no maximo 14 caracteres.");
            }
        } while (cpf.length() > 14);
        return new PessoaFisica(0, nome, logradouro, cidade, estado, telefone,
email, cpf);
    }

    private static PessoaJuridica lerDadosPessoaJuridica(Scanner scanner) {
        System.out.print("Nome: ");
        String nome = scanner.nextLine();
        System.out.print("Logradouro: ");
        String logradouro = scanner.nextLine();
        System.out.print("Cidade: ");
        String cidade = scanner.nextLine();
        String estado;
        do {
            System.out.print("Estado: ");
            estado = scanner.nextLine().toUpperCase();
            if (estado.length() != 2) {
                System.out.println("O estado deve ter exatamente 2 caracteres.");
            }
        } while (estado.length() != 2);
        String telefone;
        do {
            System.out.print("Telefone: ");
            telefone = scanner.nextLine();
            if (telefone.length() > 11) {
                System.out.println("O telefone deve ter no maximo 11 caracteres.");
            }
        } while (telefone.length() > 11);
        System.out.print("Email: ");
        String email = scanner.nextLine();
        String cnpj;
```

```
            do {
                System.out.print("CNPJ: ");
                cnpj = scanner.nextLine();
                if (cnpj.length() > 18) {
                    System.out.println("O CNPJ deve ter no maximo 14 caracteres.");
                }
            } while (cnpj.length() > 18);
            return new PessoaJuridica(0, nome, logradouro, cidade, estado, telefone,
email, cnpj);
        }
    }
```

1. **Incluir:**

```
==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
1
F - Pessoa Fisica | J - Pessoa Juridica: F
Nome: Edson
Logradouro: Rua dos Sonhos
Cidade: Goiania
Estado: GO
Telefone: 9999-8888
Email: edson@mail
CPF: 21345677900
Pessoa Fisica incluida com sucesso.

==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
1
F - Pessoa Fisica | J - Pessoa Juridica: J
Nome: Galaxy
Logradouro: Rua das Galáxias
Cidade: Sao Paulo
Estado: SP
Telefone: 8888-9999
Email: contato@galaxy
CNPJ: 12345678000199
Pessoa Juridica incluida com sucesso.
```

2. **Alterar:**

```
==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
2
F - Pessoa Fisica | J - Pessoa Juridica: f
Informe o ID da pessoa: 10
Dados atuais:
ID: 10
Nome: Edson
Endereco: Rua dos Sonhos
Cidade: Goiania
Estado: GO
Telefone: 9999-8888
Email: edson@mail
CPF: 21345677900
Informe os novos dados:
Nome: Edson Luiz
Logradouro: Rua dos Sonhos
Cidade: Goiania
Estado: GO
Telefone: 9876-5421
Email: edson@mail.com
CPF: 87394175689
Pessoa Fisica alterada com sucesso.
```

```
============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
============================
2
F - Pessoa Fisica | J - Pessoa Juridica: j
Informe o ID da pessoa: 11
Dados atuais:
ID: 11
Nome: Galaxy
Endereco: Rua das Gal◆xias
Cidade: Sao Paulo
Estado: SP
Telefone: 8888-9999
Email: contato@galaxy
CNPJ: 12345678000199
Informe os novos dados:
Nome: Galaxy Universe
Logradouro: Rua das Galaxias
Cidade: Sao Paulo
Estado: SP
Telefone: 8764-9876
Email: contato@galaxy.com
CNPJ: 12897465000190
Pessoa Juridica alterada com sucesso.
```

3. **Excluir:**

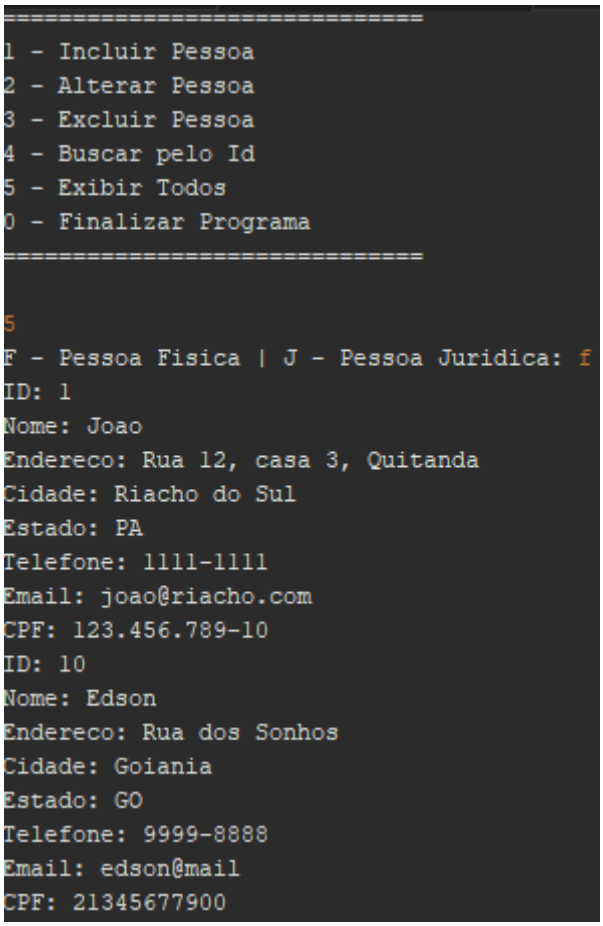
```
==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
3
F - Pessoa Fisica | J - Pessoa Juridica: f
Informe o ID da pessoa: 10
Pessoa Fisica excluida com sucesso.

==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
3
F - Pessoa Fisica | J - Pessoa Juridica: j
Informe o ID da pessoa: 11
Pessoa Juridica excluida com sucesso.
```

4. **Obter:**

```
==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
4
F - Pessoa Fisica | J - Pessoa Juridica: f
Informe o ID da pessoa: 10
ID: 10
Nome: Edson Luiz
Endereco: Rua dos Sonhos
Cidade: Goiania
Estado: GO
Telefone: 9876-5421
Email: edson@mail.com
CPF: 87394175689

==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================
4
F - Pessoa Fisica | J - Pessoa Juridica: j
Informe o ID da pessoa: 11
ID: 11
Nome: Galaxy Universe
Endereco: Rua das Galaxias
Cidade: Sao Paulo
Estado: SP
Telefone: 8764-9876
Email: contato@galaxy.com
CNPJ: 12897465000190
```

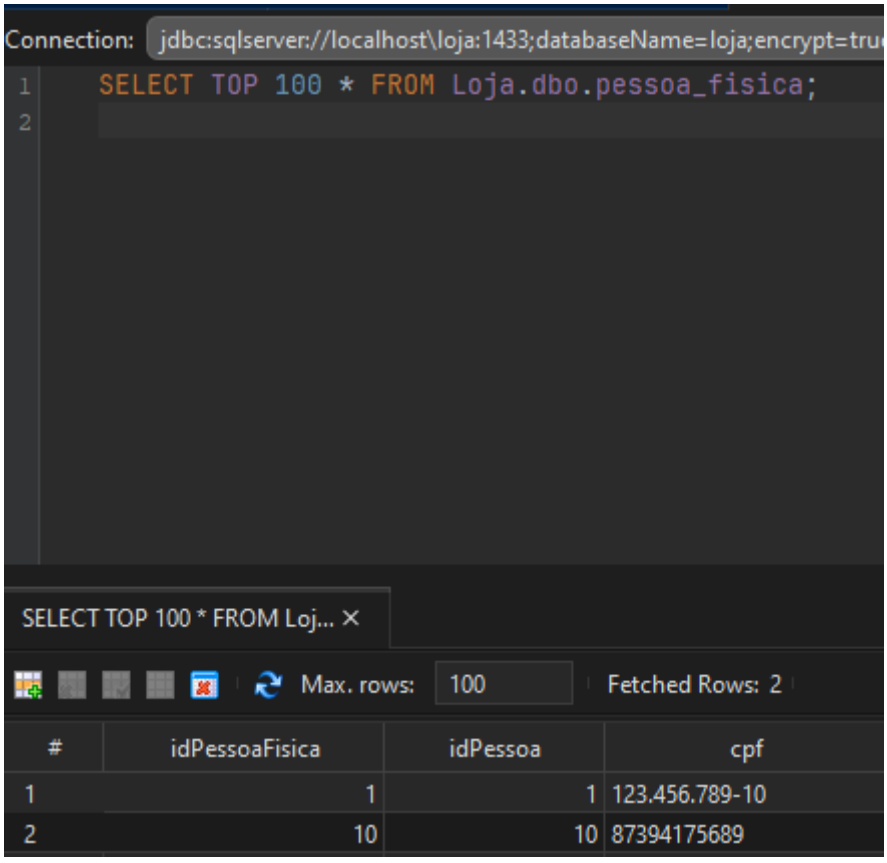
5. **ObterTodos:**

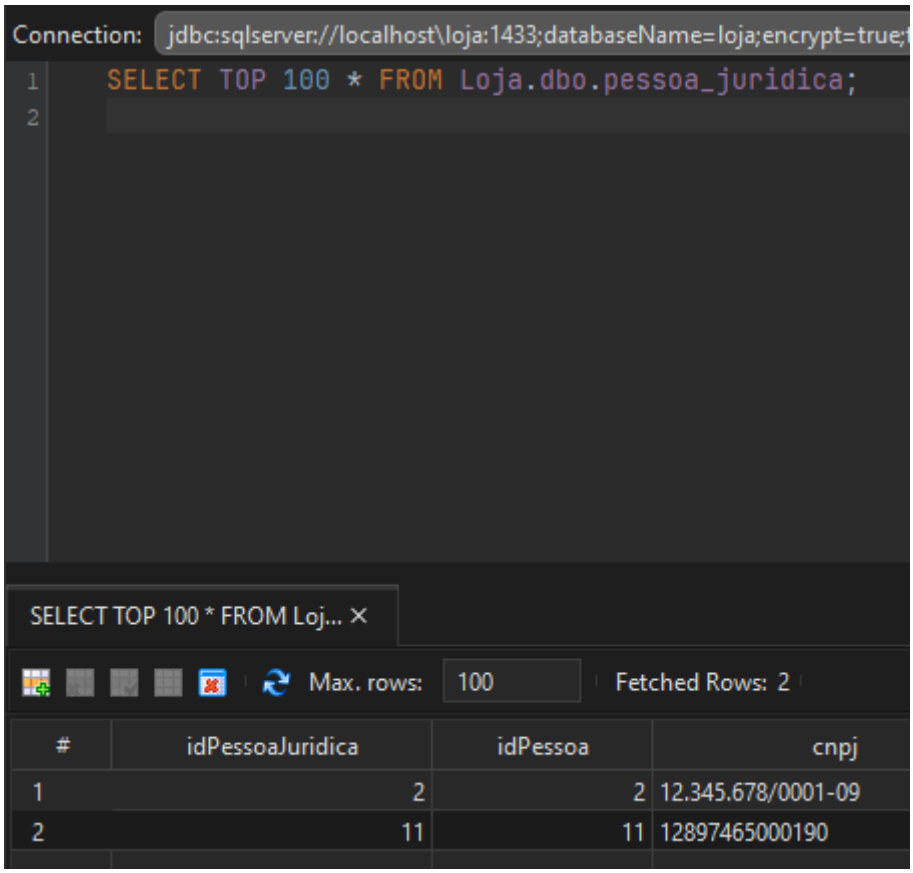
Obs: Fiz esses prints após as inclusões de Pessoa Física e Jurídica.

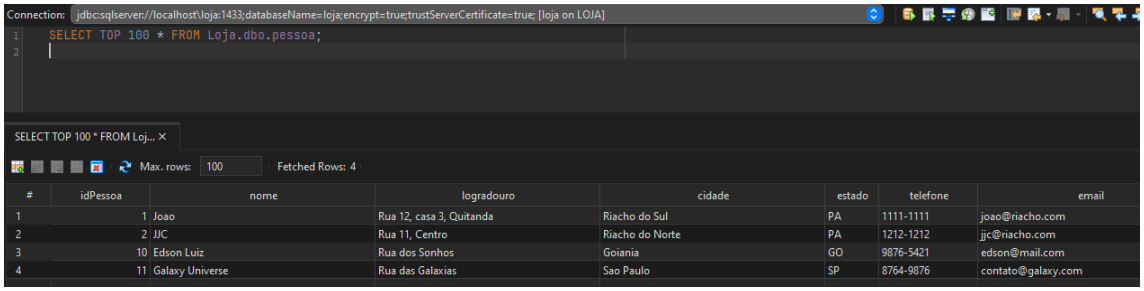


```
==============================
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
==============================

5
F - Pessoa Fisica | J - Pessoa Juridica: j
ID: 2
Nome: JJC
Endereco: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 12.345.678/0001-09
ID: 11
Nome: Galaxy
Endereco: Rua das Gal◆xias
Cidade: Sao Paulo
Estado: SP
Telefone: 8888-9999
Email: contato@galaxy
CNPJ: 12345678000199
```

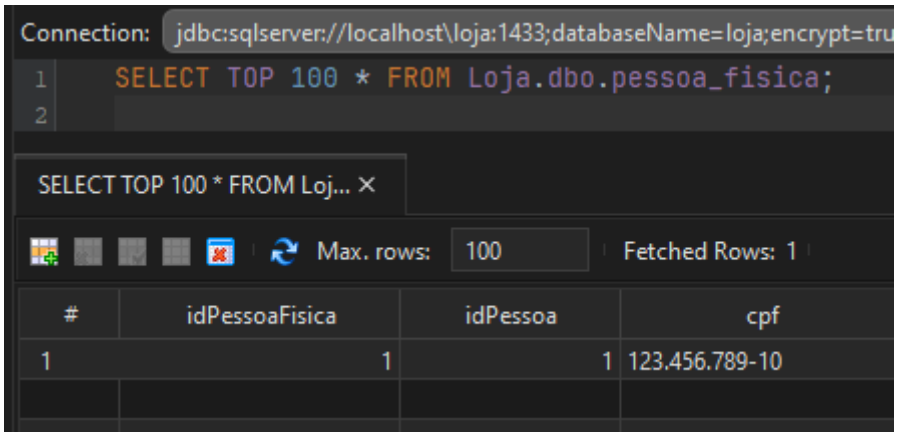
6. **Consultas:**

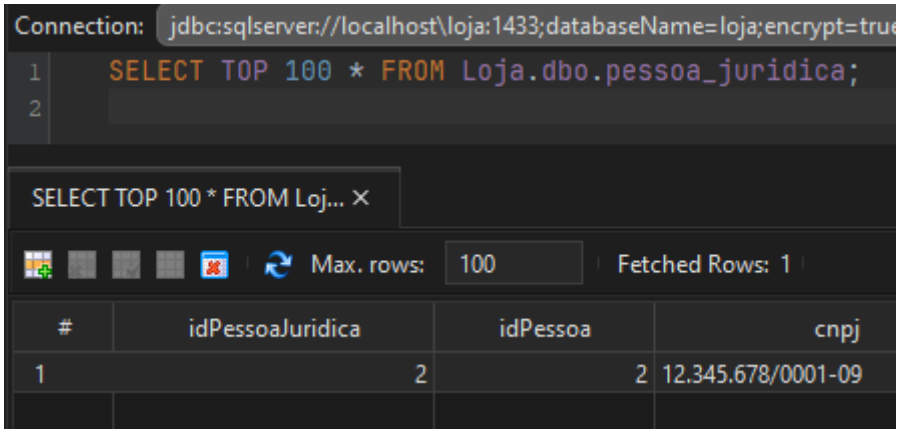

Connection: jdbc:sqlserver://localhost\loja:1433;databaseName=loja;encrypt=tru
1 SELECT TOP 100 * FROM Loja.dbo.pessoa_fisica;
2
SELECT TOP 100 * FROM Loj... ×
Max. rows: 100
Fetched Rows: 2
idPessoaFisica idPessoa cpf
1 1 1 123.456.789-10
2 10 10 87394175689


Connection: jdbc:sqlserver://localhost\loja:1433;databaseName=loja;encrypt=true;
1 SELECT TOP 100 * FROM Loja.dbo.pessoa_juridica;
2
SELECT TOP 100 * FROM Loj... ×
Max. rows: 100
Fetched Rows: 2
idPessoaJuridica idPessoa cnpj
1 2 2 12.345.678/0001-09
2 11 11 12897465000190



Pós exclusões:





Conclusão:

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

   Em arquivo: Os dados são armazenados localmente; é mais simples de implementar; não é eficiente para uso em grande escala e para uso mais complexo; é menos seguro pois os dados podem facilmente ser manipulados.

   Em banco de dados: Os dados são armazenados em um sistema de gerenciamento de banco de dados (MySQL, SQL Server ou PostgreSQL); Tem desempenho melhor e suporta operações complexas; é escalável e mais seguro possuindo até backups e controle de acesso; requer mais configurações e recursos.

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

   Por fazer de forma mais simples e direta, o código ficou mais enxuto e fácil de ler, sem precisar ter que criar loops longos ou complicados para realizar operações como a de impressão de valores.

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

Pois o main não cria uma instância da classe.

Conclusão

Elabore uma análise crítica da sua Missão Prática.

Gostei da missão prática, teve uma coisa ou outra que tive em dificuldade em entender o que foi pedido, mas creio que consegui entregar corretamente. E com essa missão pude praticar e fazer uso de banco de dados diretamente no NeBeans algo que eu não tinha conhecimento anteriormente.