



Estácio

Estácio - Mundo 3 - Missão Nível 3

Faculdade Estácio - Polo Centro – Belo Horizonte – MG

Curso: Desenvolvimento Full Stack.

Disciplina: Nível 3: Backend sem banco não tem!

RPG0016.

Semestre Letivo: 3

Integrante: André Luiz Ferreira da Silva

Repositório: <https://github.com/Andre-Luiz22/m3-n3>

IDE: NetBeans

OBS.: As fotos do código foram tiradas no vscode com a extensão CodeSnap para facilitar, mais o código foi feito no netBeans como os resultados.

Título da Prática

Backend sem banco não tem

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

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.

Arquivos do Projeto

Cadastrodb.model

Pessoa

```

1 package cadastradb.model;
2
3 public class Pessoa {
4     private Integer id;
5     private String nome;
6     private String logradouro;
7     private String cidade;
8     private String estado;
9     private String telefone;
10    private String email;
11
12    public Pessoa() {}
13
14    public Pessoa(Integer id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
15        this.id = id;
16        this.nome = nome;
17        this.logradouro = logradouro;
18        this.cidade = cidade;
19        this.estado = estado;
20        this.telefone = telefone;
21        this.email = email;
22    }
23
24    public Integer getId() {
25        return id;
26    }
27
28    public void setId(Integer id) {
29        this.id = id;
30    }
31
32    public String getNome() {
33        return nome;
34    }
35
36    public void setNome(String nome) {
37        this.nome = nome;
38    }
39
40    public String getLogradouro() {
41        return logradouro;
42    }
43
44    public void setLogradouro(String logradouro) {
45        this.logradouro = logradouro;
46    }
47
48    public String getCidade() {
49        return cidade;
50    }
51
52    public void setCidade(String cidade) {
53        this.cidade = cidade;
54    }
55
56    public String getEstado() {
57        return estado;
58    }
59
60    public void setEstado(String estado) {
61        this.estado = estado;
62    }
63
64    public String getTelefone() {
65        return telefone;
66    }
67
68    public void setTelefone(String telefone) {
69        this.telefone = telefone;
70    }
71
72    public String getEmail() {
73        return email;
74    }
75
76    public void setEmail(String email) {
77        this.email = email;
78    }
79
80    public void exibir() {
81        System.out.println("ID: " + id);
82        System.out.println("Nome: " + nome);
83        System.out.println("Logradouro: " + logradouro);
84        System.out.println("Cidade: " + cidade);
85        System.out.println("Estado: " + estado);
86        System.out.println("Telefone: " + telefone);
87        System.out.println("Email: " + email);
88    }
89 }
90

```

Pessoa Fisica

```

1 package cadastrodb.model;
2
3 public class PessoaFisica extends Pessoa {
4     private String cpf;
5
6     public PessoaFisica() {}
7
8     public PessoaFisica(Integer id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cpf) {
9         super(id, nome, logradouro, cidade, estado, telefone, email);
10        this.cpf = cpf;
11    }
12
13    public String getCpf() {
14        return cpf;
15    }
16
17    public void setCpf(String cpf) {
18        this.cpf = cpf;
19    }
20
21    @Override
22    public void exibir() {
23        super.exibir();
24        System.out.println("CPF: " + cpf);
25    }
26 }
27

```

Pessoa Juridica

```

1 package cadastrodb.model;
2
3 public class PessoaJuridica extends Pessoa {
4     private String cnpj;
5
6     public PessoaJuridica() {}
7
8     public PessoaJuridica(Integer id, String nome, String logradouro, String cidade, String estado, String telefone, String email, String cnpj) {
9         super(id, nome, logradouro, cidade, estado, telefone, email);
10        this.cnpj = cnpj;
11    }
12
13    public String getCnpj() {
14        return cnpj;
15    }
16
17    public void setCnpj(String cnpj) {
18        this.cnpj = cnpj;
19    }
20
21    @Override
22    public void exibir() {
23        super.exibir();
24        System.out.println("CNPJ: " + cnpj);
25    }
26 }

```

Cadastro.model.util

Conectordb

```

1 package cadastro.model.util;
2
3 import java.sql.*;
4
5 public class ConectorBD {
6     private Connection connection = null;
7
8     public Connection getConnection() {
9         if(connection != null) return connection;
10
11         try {
12             String url = "jdbc:sqlserver://localhost;databaseName=loja;encrypt=true;trustServerCertificate=true";
13             String usuario = "loja";
14             String senha = "loja";
15
16             connection = DriverManager.getConnection(url, usuario, senha);
17         } catch (SQLException e) {
18             e.printStackTrace();
19         }
20
21         return connection;
22     }
23
24     public PreparedStatement getPrepared(String sql) throws SQLException {
25         Connection conn = getConnection();
26         return conn.prepareStatement(sql);
27     }
28
29     public ResultSet getSelect(String sql) throws SQLException {
30         PreparedStatement statement = getPrepared(sql);
31         return statement.executeQuery();
32     }
33
34     public void close(Statement statement) {
35         try {
36             statement.close();
37         } catch (SQLException e) {
38             e.printStackTrace();
39         }
40     }
41
42     public void close(ResultSet resultSet) {
43         try {
44             resultSet.close();
45         } catch (SQLException e) {
46             e.printStackTrace();
47         }
48     }
49
50     public void close(Connection connection) {
51         try {
52             connection.close();
53         } catch (SQLException e) {
54             e.printStackTrace();
55         }
56     }
57 }
58

```

SequenceManager



```
1 package cadastro.model.util;
2
3 import java.sql.ResultSet;
4 import java.sql.SQLException;
5
6 public class SequenceManager {
7     ConectorBD conectorBD;
8
9     public SequenceManager(ConectorBD conectorBD) {
10         this.conectorBD = conectorBD;
11     }
12
13     public int getValue(String nomeSequencia) throws SQLException {
14         String sql = "SELECT NEXT VALUE FOR " + nomeSequencia;
15         try(ResultSet result = conectorBD.getSelect(sql)) {
16             if(result.next()) {
17                 return result.getInt(1);
18             }
19         }
20         return -1;
21     }
22 }
```

Cadastro.model

PessoaFisicaDAO

```
1 package cadastro.model;
2
3 import cadastro.model.util.ConectorBD;
4 import cadastro.model.util.SequenceManager;
5 import cadastrodb.model.PessoaFisica;
6
7 import java.sql.PreparedStatement;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 public class PessoaFisicaDAO {
14     private ConectorBD conectorBD;
15     private SequenceManager sequenceManager;
16
17     public PessoaFisicaDAO(ConectorBD conectorBD, SequenceManager sequenceManager) {
18         this.conectorBD = conectorBD;
19         this.sequenceManager = sequenceManager;
20     }
21
22     public PessoaFisica getPessoa(int id) {
23         PessoaFisica pessoaFisica = null;
24         String sql = "SELECT * FROM pessoa AS p INNER JOIN pessoaFisica AS pf ON p.idPessoa = pf.idPessoa WHERE p.idPessoa = ?";
25         try {
26             PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
27             preparedStatement.setInt(1, id);
28             ResultSet resultSet = preparedStatement.executeQuery();
29             if(resultSet.next()) {
30                 pessoaFisica = converterPessoa(resultSet);
31             }
32             conectorBD.close(preparedStatement);
33             conectorBD.close(resultSet);
34         } catch (SQLException e) {
35             e.printStackTrace();
36         }
37
38         return pessoaFisica;
39     }
}
```



```

1
2 public List<PessoaFisica> getPessoas() {
3     List<PessoaFisica> pessoas = new ArrayList<PessoaFisica>();
4     String sql = "SELECT * FROM pessoa AS p INNER JOIN pessoaFisica AS pf ON p.idPessoa = pf.idPessoa";
5     try(ResultSet resultSet = conectorBD.getSelect(sql)) {
6         while(resultSet.next()) {
7             pessoas.add(converterPessoa(resultSet));
8         }
9         conectorBD.close(resultSet);
10    } catch (SQLException e) {
11        e.printStackTrace();
12    }
13
14    return pessoas;
15 }
16
17 public PessoaFisica inserir(PessoaFisica pessoa) {
18     try {
19         int idPessoa = sequenceManager.getValue("idPessoa");
20         pessoa.setId(idPessoa);
21         String sql = "INSERT INTO pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
22         PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
23         preparedStatement.setInt(1, idPessoa);
24         preparedStatement.setString(2, pessoa.getNome());
25         preparedStatement.setString(3, pessoa.getLogradouro());
26         preparedStatement.setString(4, pessoa.getCidade());
27         preparedStatement.setString(5, pessoa.getEstado());
28         preparedStatement.setString(6, pessoa.getTelefone());
29         preparedStatement.setString(7, pessoa.getEmail());
30         preparedStatement.executeUpdate();
31         conectorBD.close(preparedStatement);
32
33         String sqlPF = "INSERT INTO pessoaFisica (idPessoa, cpf) VALUES (?, ?)";
34         PreparedStatement preparedStatementPF = conectorBD.getPrepared(sqlPF);
35         preparedStatementPF.setInt(1, idPessoa);
36         preparedStatementPF.setString(2, pessoa.getCpf());
37         preparedStatementPF.executeUpdate();
38         conectorBD.close(preparedStatementPF);
39
40         System.out.println("Pessoa Fisica salva com sucesso.");
41     } catch (SQLException e) {
42         e.printStackTrace();
43     }
44
45     return pessoa;
46 }
47

```

```

1
2 public void alterar(PessoaFisica pessoa) {
3     try {
4         String sql = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
5         PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
6         preparedStatement.setString(1, pessoa.getNome());
7         preparedStatement.setString(2, pessoa.getLogradouro());
8         preparedStatement.setString(3, pessoa.getCidade());
9         preparedStatement.setString(4, pessoa.getEstado());
10        preparedStatement.setString(5, pessoa.getTelefone());
11        preparedStatement.setString(6, pessoa.getEmail());
12        preparedStatement.setInt(7, pessoa.getId());
13        preparedStatement.executeUpdate();
14        conectorBD.close(preparedStatement);
15
16        String sqlPF = "UPDATE pessoaFisica SET cpf = ? WHERE idPessoa = ?";
17        PreparedStatementPF preparedStatementPF = conectorBD.getPrepared(sqlPF);
18        preparedStatementPF.setString(1, pessoa.getCpf());
19        preparedStatementPF.setInt(2, pessoa.getId());
20        preparedStatementPF.executeUpdate();
21        conectorBD.close(preparedStatementPF);
22
23        System.out.println("Pessoa Fisica atualizada com sucesso.");
24    } catch (SQLException e) {
25        e.printStackTrace();
26    }
27 }
28
29 public void excluir(int id) {
30     try {
31         String sqlPF = "DELETE FROM pessoaFisica WHERE idPessoa = ?";
32         PreparedStatementPF preparedStatementPF = conectorBD.getPrepared(sqlPF);
33         preparedStatementPF.setInt(1, id);
34         preparedStatementPF.executeUpdate();
35
36         String sql = "DELETE FROM pessoa WHERE idPessoa = ?";
37         PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
38         preparedStatement.setInt(1, id);
39         preparedStatement.executeUpdate();
40         conectorBD.close(preparedStatement);
41
42         System.out.println("Pessoa Fisica excluida com sucesso.");
43     } catch (SQLException e) {
44         e.printStackTrace();
45     }
46 }
47
48 private PessoaFisica converterPessoa(ResultSet resultSet) throws SQLException {
49     PessoaFisica pessoa = new PessoaFisica();
50     pessoa.setId(resultSet.getInt("idPessoa"));
51     pessoa.setNome(resultSet.getString("nome"));
52     pessoa.setLogradouro(resultSet.getString("logradouro"));
53     pessoa.setCidade(resultSet.getString("cidade"));
54     pessoa.setEstado(resultSet.getString("estado"));
55     pessoa.setTelefone(resultSet.getString("telefone"));
56     pessoa.setEmail(resultSet.getString("email"));
57     pessoa.setCpf(resultSet.getString("cpf"));
58     return pessoa;
59 }
60 }
61

```

PessoaJuridicaDAO

```
1 package cadastro.model;
2
3 import cadastro.model.util.ConectorBD;
4 import cadastro.model.util.SequenceManager;
5 import cadastrodb.model.PessoaJuridica;
6
7 import java.sql.PreparedStatement;
8 import java.sql.ResultSet;
9 import java.sql.SQLException;
10 import java.util.ArrayList;
11 import java.util.List;
12
13 public class PessoaJuridicaDAO {
14     private ConectorBD conectorBD;
15     private SequenceManager sequenceManager;
16
17     public PessoaJuridicaDAO(ConectorBD conectorBD, SequenceManager sequenceManager) {
18         this.conectorBD = conectorBD;
19         this.sequenceManager = sequenceManager;
20     }
21
22     public PessoaJuridica getPessoa(int id) {
23         PessoaJuridica pessoaJuridica = null;
24         String sql = "SELECT * FROM pessoa AS p INNER JOIN pessoaJuridica AS pj ON p.idPessoa = pj.idPessoa WHERE p.idPessoa = ?";
25         try {
26             PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
27             preparedStatement.setInt(1, id);
28             ResultSet resultSet = preparedStatement.executeQuery();
29             if(resultSet.next()) {
30                 pessoaJuridica = converterPessoa(resultSet);
31             }
32             conectorBD.close(preparedStatement);
33             conectorBD.close(resultSet);
34         } catch (SQLException e) {
35             e.printStackTrace();
36         }
37
38         return pessoaJuridica;
39     }
}
```

```

1
2 public List<PessoaJuridica> getPessoas() {
3     List<PessoaJuridica> pessoas = new ArrayList<PessoaJuridica>();
4     String sql = "SELECT * FROM pessoa AS p INNER JOIN pessoaJuridica AS pj ON p.idPessoa = pj.idPessoa";
5     try(ResultSet resultSet = conectorBD.getSelect(sql)) {
6         while(resultSet.next()) {
7             pessoas.add(converterPessoa(resultSet));
8         }
9         conectorBD.close(resultSet);
10    } catch (SQLException e) {
11        e.printStackTrace();
12    }
13
14    return pessoas;
15 }
16
17 public PessoaJuridica inserir(PessoaJuridica pessoa) {
18     try {
19         int idPessoa = sequenceManager.getValue("idPessoa");
20         pessoa.setId(idPessoa);
21         String sql = "INSERT INTO pessoa (idPessoa, nome, logradouro, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?, ?)";
22         PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
23         preparedStatement.setInt(1, idPessoa);
24         preparedStatement.setString(2, pessoa.getNome());
25         preparedStatement.setString(3, pessoa.getLogradouro());
26         preparedStatement.setString(4, pessoa.getCidade());
27         preparedStatement.setString(5, pessoa.getEstado());
28         preparedStatement.setString(6, pessoa.getTelefone());
29         preparedStatement.setString(7, pessoa.getEmail());
30         preparedStatement.executeUpdate();
31         conectorBD.close(preparedStatement);
32
33         String sqlPJ = "INSERT INTO pessoaJuridica (idPessoa, cnpj) VALUES (?, ?)";
34         PreparedStatement preparedStatementPJ = conectorBD.getPrepared(sqlPJ);
35         preparedStatementPJ.setInt(1, idPessoa);
36         preparedStatementPJ.setString(2, pessoa.getCnpj());
37         preparedStatementPJ.executeUpdate();
38         conectorBD.close(preparedStatementPJ);
39
40         System.out.println("Pessoa Jurídica salva com sucesso");
41     } catch (SQLException e) {
42         e.printStackTrace();
43     }
44
45     return pessoa;
46 }

```

```

1 public void alterar(PessoaJuridica pessoa) {
2     try {
3         String sql = "UPDATE pessoa SET nome = ?, logradouro = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE idPessoa = ?";
4         PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
5         preparedStatement.setString(1, pessoa.getNome());
6         preparedStatement.setString(2, pessoa.getLogradouro());
7         preparedStatement.setString(3, pessoa.getCidade());
8         preparedStatement.setString(4, pessoa.getEstado());
9         preparedStatement.setString(5, pessoa.getTelefone());
10        preparedStatement.setString(6, pessoa.getEmail());
11        preparedStatement.setInt(7, pessoa.getId());
12        preparedStatement.executeUpdate();
13        conectorBD.close(preparedStatement);
14
15        String sqlPJ = "UPDATE pessoaJuridica SET cnpj = ? WHERE idPessoa = ?";
16        PreparedStatement preparedStatementPJ = conectorBD.getPrepared(sqlPJ);
17        preparedStatementPJ.setString(1, pessoa.getCnpj());
18        preparedStatementPJ.setInt(2, pessoa.getId());
19        preparedStatementPJ.executeUpdate();
20        conectorBD.close(preparedStatementPJ);
21
22        System.out.println("Pessoa Juridica atualizada com sucesso.");
23    } catch (SQLException e) {
24        e.printStackTrace();
25    }
26 }
27
28 public void excluir(int id) {
29     try {
30         String sqlPf = "DELETE FROM pessoaJuridica WHERE idPessoa = ?";
31         PreparedStatement preparedStatementPJ = conectorBD.getPrepared(sqlPf);
32         preparedStatementPJ.setInt(1, id);
33         preparedStatementPJ.executeUpdate();
34
35         String sql = "DELETE FROM pessoa WHERE idPessoa = ?";
36         PreparedStatement preparedStatement = conectorBD.getPrepared(sql);
37         preparedStatement.setInt(1, id);
38         preparedStatement.executeUpdate();
39         conectorBD.close(preparedStatement);
40
41         System.out.println("Pessoa Juridica excluida com sucesso.");
42     } catch (SQLException e) {
43         e.printStackTrace();
44     }
45 }
46
47 private PessoaJuridica converterPessoa(ResultSet resultSet) throws SQLException {
48     PessoaJuridica pessoa = new PessoaJuridica();
49     pessoa.setId(resultSet.getInt("idPessoa"));
50     pessoa.setNome(resultSet.getString("nome"));
51     pessoa.setLogradouro(resultSet.getString("logradouro"));
52     pessoa.setCidade(resultSet.getString("cidade"));
53     pessoa.setEstado(resultSet.getString("estado"));
54     pessoa.setTelefone(resultSet.getString("telefone"));
55     pessoa.setEmail(resultSet.getString("email"));
56     pessoa.setCnpj(resultSet.getString("cnpj"));
57     return pessoa;
58 }
59 }
60

```

CadastroDB

```

1 import cadastro.model.PessoaFisicaDAO;
2 import cadastro.model.PessoaJuridicaDAO;
3 import cadastro.model.util.ConectorBD;
4 import cadastro.model.util.SequenceManager;
5 import cadastrodb.model.PessoaFisica;
6 import cadastrodb.model.PessoaJuridica;
7
8 import java.util.List;
9
10 public class CadastroDB {
11     public static void main(String[] args) {
12         ConectorBD conectorBD = new ConectorBD();
13         SequenceManager sequenceManager = new SequenceManager(conectorBD);
14         PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD, sequenceManager);
15         PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conectorBD, sequenceManager);
16
17         // Instanciar uma pessoa física e persistir no banco de dados.
18         PessoaFisica pessoaFisica = new PessoaFisica(0, "Ribeiro", "Rua 30, Centro", "São Paulo", "SP", "1111-1111", "ribeiro@sp.com.br", "111111111111");
19         pessoaFisica = pessoaFisicaDAO.inserir(pessoaFisica);
20
21         // Alterar os dados da pessoa física no banco.
22         pessoaFisica.setNome("Ribeiro Alberto");
23         pessoaFisica.setCidade("Guarulhos");
24         pessoaFisicaDAO.alterar(pessoaFisica);
25
26         // Consultar todas as pessoas físicas do banco de dados e listar no console.
27         List<PessoaFisica> listaPF = pessoaFisicaDAO.getPessoas();
28         listaPF.forEach(PessoaFisica::exibir);
29
30         // Excluir a pessoa física criada anteriormente no banco.
31         pessoaFisicaDAO.excluir(pessoaFisica.getId());
32
33         // Instanciar uma pessoa jurídica e persistir no banco de dados.
34         PessoaJuridica pessoaJuridica = new PessoaJuridica(0, "Ribeiro LTDA", "Rua 30, Centro", "São Paulo", "SP", "1111-1111", "ribeiro-ltda@sp.com.br", "11111111111111");
35         pessoaJuridica = pessoaJuridicaDAO.inserir(pessoaJuridica);
36
37         // Alterar os dados da pessoa jurídica no banco.
38         pessoaJuridica.setNome("Ribeiro Soares LTDA");
39         pessoaJuridica.setLogradouro("Rua 31, Centro");
40         pessoaJuridicaDAO.alterar(pessoaJuridica);
41
42         // Consultar todas as pessoas jurídicas do banco e listar no console.
43         List<PessoaJuridica> listaPJ = pessoaJuridicaDAO.getPessoas();
44         listaPJ.forEach(PessoaJuridica::exibir);
45
46         // Excluir a pessoa jurídica criada anteriormente no banco.
47         pessoaJuridicaDAO.excluir(pessoaJuridica.getId());
48
49         conectorBD.close(conectorBD.getConnection());
50     }
51 }
52
53

```

Análise e Conclusão

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

Os componentes de middleware são responsáveis por permitir a integração simplificada, de forma transparente. Os middlewares como o JDBC são responsáveis pela comunicação com uma fonte de dados, fornecendo uma interface padronizada, que muitas vezes requer pouca ou nenhuma alteração na implementação para alterar o fornecedor, além de abstrair a forma como as consultas são realizadas, criando um padrão que pode ser utilizado independente do banco de dados utilizado.

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

O statement executa uma instrução SQL estática, onde todos os dados que compõem a consulta devem ser informados, ele é indicado apenas para consultas sem parâmetros,

pois não oferece um tratamento nos dados informados, tornando seu uso uma vulnerabilidade para SQL Injection.

O PreparedStatement recebe seus parâmetros após a definição da instrução SQL definida, que previne que dados de tipos não esperados sejam informados ou instruções que explorem o SQL Injection, além de facilitar a escrita do comando SQL, pois não é necessário o uso de apóstrofe ou qualquer outro delimitador.

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

Ele tem como objetivo organizar e centralizar todos os comandos SQL que são utilizados pela aplicação, é indicado possuir uma classe DAO para cada classe de entidade relevante para o sistema. Utilizar o padrão DAO tem como principais benefícios a reutilização de comandos sem duplicação de código e a facilidade na manutenção do código, pois todas as consultas estão centralizadas em classes DAO.

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

Na modelagem de herança em um banco de dados relacional, a superclasse e a classe são representadas por duas tabelas distintas, relacionadas por meio de uma chave primária e uma chave estrangeira, estabelecendo um relacionamento 1 para 1. Durante a inserção de dados, a tabela que representa a classe depende de uma chave primária existente na tabela que representa a superclasse. Esse tipo de relacionamento simplifica a recuperação de dados e se assemelha muito à estrutura de herança encontrada em Java.

2º Procedimento | Alimentando a Base

CadastroDB2

```
1 import cadastro.model.PessoaFisicaDAO;
2 import cadastro.model.PessoaJuridicaDAO;
3 import cadastro.model.util.ConectorBD;
4 import cadastro.model.util.SequenceManager;
5 import cadastro.model.PessoaFisica;
6 import cadastro.model.PessoaJuridica;
7
8 import java.util.List;
9 import java.util.Scanner;
10
11 public class CadastroDBTeste2 {
12     public static void main(String[] args) {
13         ConectorBD conectorBD = new ConectorBD();
14         SequenceManager sequenceManager = new SequenceManager(conectorBD);
15         PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conectorBD, sequenceManager);
16         PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conectorBD, sequenceManager);
17
18         Scanner scanner = new Scanner(System.in);
19
20         while(true) {
21             try {
22                 int opcaoMenu = mostrarMenu(scanner);
23
24                 scanner.nextLine();
25
26                 if (opcaoMenu == 0) {
27                     break;
28                 }
29
30                 String tipoPessoa = tipoPessoa(scanner);
31
32                 switch (opcaoMenu) {
33                     case 1:
34                         inserir(pessoaFisicaDAO, pessoaJuridicaDAO, scanner, tipoPessoa);
35                         break;
36                     case 2:
37                         alterar(pessoaFisicaDAO, pessoaJuridicaDAO, scanner, tipoPessoa);
38                         break;
39                     case 3:
40                         excluirPorId(pessoaFisicaDAO, pessoaJuridicaDAO, scanner, tipoPessoa);
41                         break;
42                     case 4:
43                         buscarPorId(pessoaFisicaDAO, pessoaJuridicaDAO, scanner, tipoPessoa);
44                         break;
45                     case 5:
46                         exibirTodos(pessoaFisicaDAO, pessoaJuridicaDAO, tipoPessoa);
47                         break;
48                 }
49             } catch (Exception err) {
50                 System.out.println("Opção inválida, tente novamente.");
51             }
52         }
53     }
54
55     public static int mostrarMenu(Scanner scanner) {
56         System.out.println("=====");
57         System.out.println("1 - Incluir Pessoa");
58         System.out.println("2 - Alterar Pessoa");
59         System.out.println("3 - Excluir Pessoa");
60         System.out.println("4 - Buscar pelo Id");
61         System.out.println("5 - Exibir Todos");
62         System.out.println("0 - Finalizar Programa");
63         System.out.println("=====");
64
65         return scanner.nextInt();
66     }
67 }
```



```

1
2 public static String tipoPessoa(Scanner scanner) throws Exception {
3     System.out.println("F - Pessoa Física | J - Pessoa Jurídica");
4     String tipoPessoa = scanner.nextLine();
5     if(tipoPessoa.equals("F") || tipoPessoa.equals("J")) return tipoPessoa;
6     throw new Exception("Tipo de Pessoa inválida");
7 }
8
9 public static void inserir(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner, String tipoPessoa) {
10     exibirTitulo("Insira os dados da Pessoa " + tipoPessoaString(tipoPessoa));
11
12     System.out.println("Digite o nome:");
13     String nome = scanner.nextLine();
14
15     System.out.println("Digite o logradouro:");
16     String logradouro = scanner.nextLine();
17
18     System.out.println("Digite a cidade:");
19     String cidade = scanner.nextLine();
20
21     System.out.println("Digite o estado (2 caracteres):");
22     String estado = scanner.nextLine();
23
24     System.out.println("Digite o telefone:");
25     String telefone = scanner.nextLine();
26
27     System.out.println("Digite o email:");
28     String email = scanner.nextLine();
29
30     if(tipoPessoa.equals("F")) {
31         PessoaFisica pessoaFisica = criarPF(scanner, 0, nome, logradouro, cidade, estado, telefone, email);
32         pessoaFisicaDAO.inserir(pessoaFisica);
33     } else {
34         PessoaJuridica pessoaJuridica = criarPJ(scanner, 0, nome, logradouro, cidade, estado, telefone, email);
35         pessoaJuridicaDAO.inserir(pessoaJuridica);
36     }
37     System.out.println();
38 }
39
40 private static void alterar(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner, String tipoPessoa) {
41     exibirTitulo("Insira os dados da " + tipoPessoaString(tipoPessoa));
42
43     System.out.println("Digite o ID:");
44     int id = scanner.nextInt();
45
46     scanner.nextLine();
47
48     System.out.println("Digite o nome:");
49     String nome = scanner.nextLine();
50
51     System.out.println("Digite o logradouro:");
52     String logradouro = scanner.nextLine();
53
54     System.out.println("Digite a cidade:");
55     String cidade = scanner.nextLine();
56
57     System.out.println("Digite o estado (2 caracteres):");
58     String estado = scanner.nextLine();
59
60     System.out.println("Digite o telefone:");
61     String telefone = scanner.nextLine();
62
63     System.out.println("Digite o email:");
64     String email = scanner.nextLine();
65
66     if(tipoPessoa.equals("F")) {
67         PessoaFisica pessoaFisica = criarPF(scanner, id, nome, logradouro, cidade, estado, telefone, email);
68         pessoaFisicaDAO.alterar(pessoaFisica);
69     } else {
70         PessoaJuridica pessoaJuridica = criarPJ(scanner, id, nome, logradouro, cidade, estado, telefone, email);
71         pessoaJuridicaDAO.alterar(pessoaJuridica);
72     }
73     System.out.println();
74 }

```

```

1
2 private static void excluirPorId(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner, String tipoPessoa) {
3     System.out.println("Digite o ID:");
4     int id = scanner.nextInt();
5     scanner.nextLine();
6
7     if(tipoPessoa.equals("F")) {
8         pessoaFisicaDAO.excluir(id);
9     } else {
10        pessoaJuridicaDAO.excluir(id);
11    }
12    System.out.println();
13 }
14
15 private static void buscarPorId(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner, String tipoPessoa) {
16     System.out.println("Digite o ID:");
17     int id = scanner.nextInt();
18     scanner.nextLine();
19
20     exibirTitulo("Exibindo Pessoa " + tipoPessoaString(tipoPessoa));
21
22     if(tipoPessoa.equals("F")) {
23         PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);
24         if(pessoaFisica == null) {
25             System.out.println("Nenhuma Pessoa Física encontrada com o id " + id);
26         } else {
27             pessoaFisica.exibir();
28         }
29     }
30     else {
31         PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
32         if(pessoaJuridica == null) {
33             System.out.println("Nenhuma Pessoa Jurídica encontrada com o id " + id);
34         } else {
35             pessoaJuridica.exibir();
36         }
37     }
38     System.out.println();
39 }
40
41 private static void exibirTodos(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO pessoaJuridicaDAO, String tipoPessoa) {
42     exibirTitulo("Exibindo dados de Pessoas " + tipoPessoaString(tipoPessoa) + "s");
43     if(tipoPessoa.equals("F")) {
44         List<PessoaFisica> pessoaFisicaList = pessoaFisicaDAO.getPessoas();
45         pessoaFisicaList.forEach(pessoaFisica -> {
46             pessoaFisica.exibir();
47             System.out.println();
48         });
49     } else {
50         List<PessoaJuridica> pessoaJuridicaList = pessoaJuridicaDAO.getPessoas();
51         pessoaJuridicaList.forEach(pessoaJuridica -> {
52             pessoaJuridica.exibir();
53             System.out.println();
54         });
55     }
56 }
57
58 private static PessoaFisica criarPF(Scanner scanner, int id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
59     System.out.println("Digite o CPF:");
60     String cpf = scanner.nextLine();
61
62     return new PessoaFisica(id, nome, logradouro, cidade, estado, telefone, email, cpf);
63 }
64
65 private static PessoaJuridica criarPJ(Scanner scanner, int id, String nome, String logradouro, String cidade, String estado, String telefone, String email) {
66     System.out.println("Digite o CNPJ:");
67     String cnpj = scanner.nextLine();
68
69     return new PessoaJuridica(id, nome, logradouro, cidade, estado, telefone, email, cnpj);
70 }
71
72 private static void exibirTitulo(String msg) {
73     System.out.println(msg);
74     System.out.println("=====");
75 }
76
77 private static String tipoPessoaString(String tipoPessoa) {
78     return tipoPessoa.equals("F") ? "Física" : "Jurídica";
79 }
80 }
81

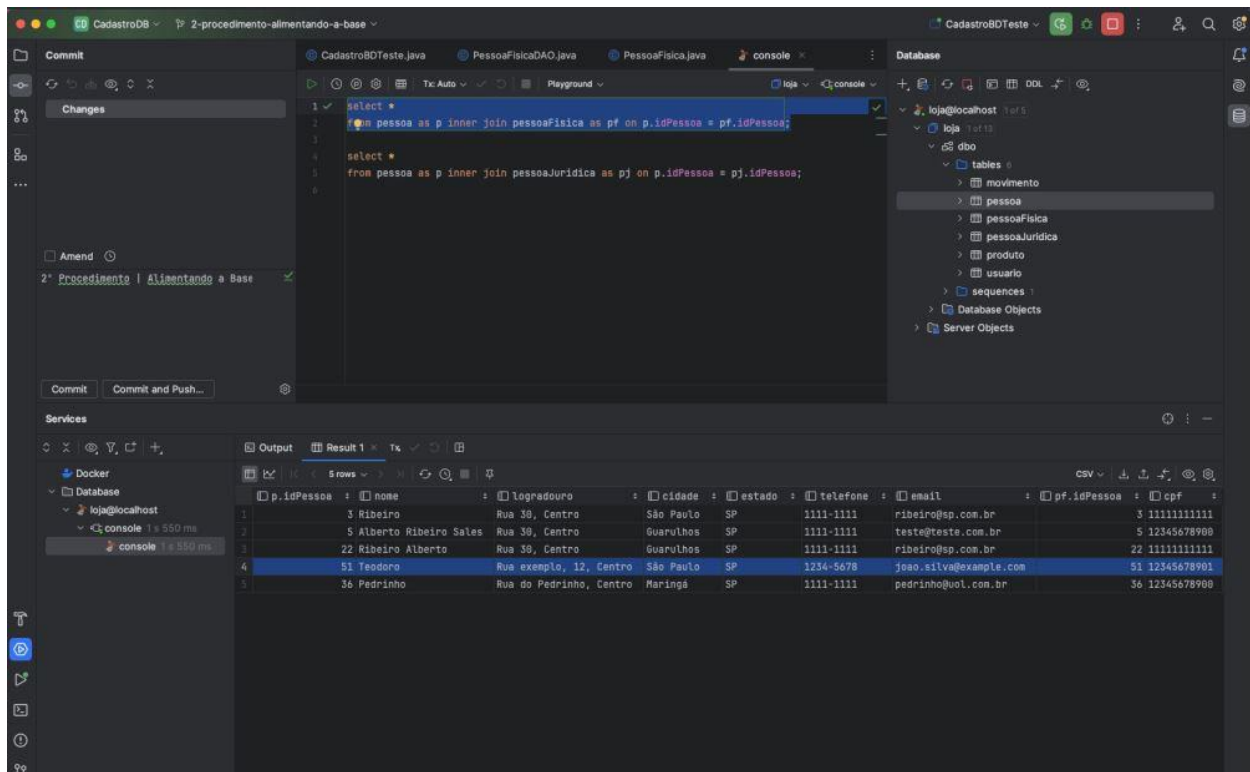
```

Teste das funcionalidades

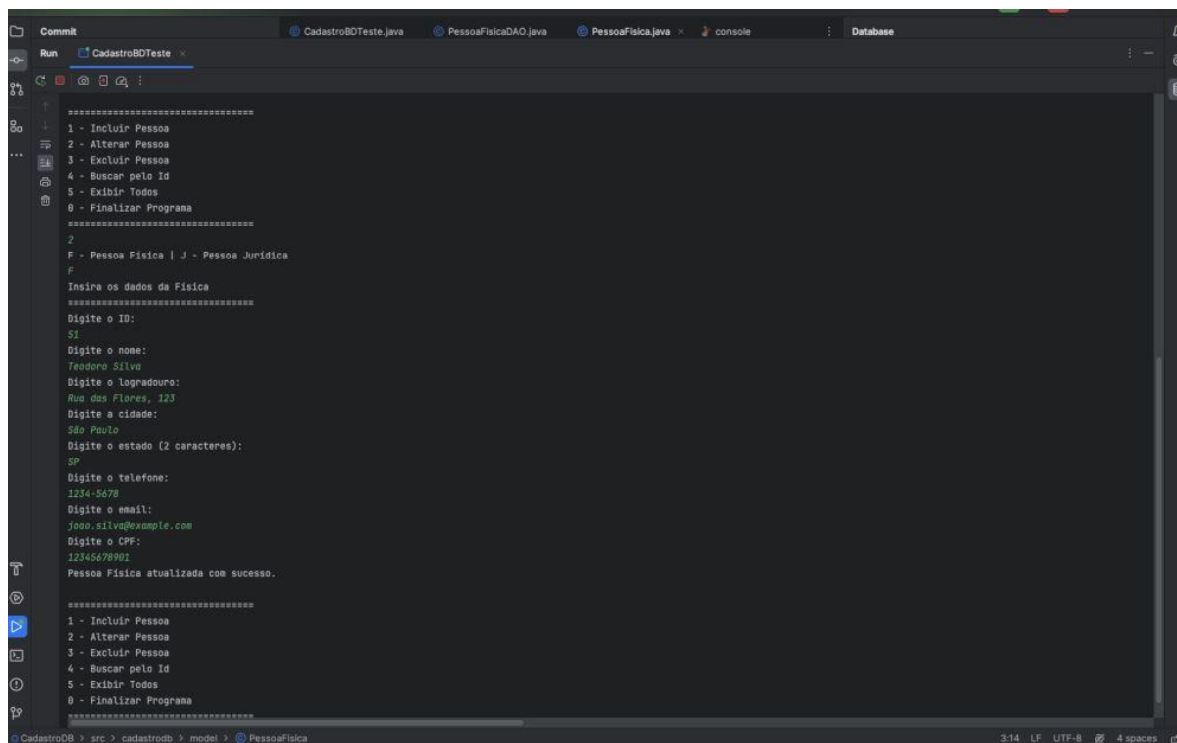
Pessoa Fisica

- Inserir

```
Run CadastroBDTeste
/opt/homebrew/Cellar/openjdk/21.0.2/libexec/openjdk.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=49654:/Applications/IntelliJ IDEA.app/Contents
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Finalizar Programa
=====
I
F - Pessoa Fisica | J - Pessoa Juridica
F
Insira os dados da Pessoa Fisica
=====
Digite o nome:
Teodoro
Digite o logradouro:
Rua exemplo, 12, Centro
Digite a cidade:
São Paulo
Digite o estado (2 caracteres):
SP
Digite o telefone:
1234-5678
Digite o email:
joao.silva@example.com
Digite o CPF:
12345678901
Pessoa Fisica salva com sucesso.
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
6 - Finalizar Programa
=====
```



- Alterar



The screenshot shows an IDE with a SQL query editor and a database table view. The query is:

```
1 select *
2 from pessoa as p inner join pessoaFisica as pf on p.idPessoa = pf.idPessoa;
3
4 select *
5 from pessoa as p inner join pessoaJuridica as pj on p.idPessoa = pj.idPessoa;
6
```

The database table view shows the following data:

p.idPessoa	nome	logradouro	cidade	estado	telefone	email	pf.idPessoa	cpf
3	Ribeiro	Rua 38, Centro	São Paulo	SP	1111-1111	ribeiro@sp.com.br	3	11111111111
5	Alberto Ribeiro Sales	Rua 38, Centro	Guarulhos	SP	1111-1111	teste@teste.com.br	5	12345678900
22	Ribeiro Alberto	Rua 38, Centro	Guarulhos	SP	1111-1111	ribeiro@sp.com.br	22	11111111111
51	Teodoro Silva	Rua das Flores, 123	São Paulo	SP	1234-5678	joao.silva@example.com	51	12345678901
36	Pedrinho	Rua do Pedrinho, Centro	Maringá	SP	1111-1111	pedrinho@uol.com.br	36	12345678900

- Excluir

The screenshot shows an IDE with a menu-driven application interface. The menu options are:

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

The application is running and displaying the following text:

```
=====
3
F - Pessoa Fisica | J - Pessoa Juridica
F
Digite o ID:
51
Pessoa Fisica excluida com sucesso.
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====
```

The screenshot shows an IDE with a SQL editor and a database table view. The SQL editor contains two queries:

```
1 select *
2 from pessoa as p inner join pessoaFisica as pf on p.idPessoa = pf.idPessoa;
3
4 select *
5 from pessoa as p inner join pessoaJuridica as pj on p.idPessoa = pj.idPessoa;
6
```

The database view shows a table with the following data:

p.idPessoa	nome	logradouro	cidade	estado	telefone	email	pf.idPessoa	cpf
3	Ribeiro	Rua 30, Centro	São Paulo	SP	1111-1111	ribeiro@sp.com.br	3	11111111111
5	Alberto Ribeiro Sales	Rua 30, Centro	Guarulhos	SP	1111-1111	teste@teste.com.br	5	12345678900
22	Ribeiro Alberto	Rua 30, Centro	Guarulhos	SP	1111-1111	ribeiro@sp.com.br	22	11111111111
36	Pedrinho	Rua do Pedrinho, Centro	Maringá	SP	1111-1111	pedrinho@uol.com.br	36	12345678900

- Buscar por ID

The screenshot shows a Java application running in the console. The application is a menu-driven program for managing people. The console output shows the following sequence of events:

```
51
 Pessoa Fisica excluida com sucesso.
=====
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
Digite o ID:
22
Exibindo Pessoa Fisica
=====
ID: 22
Nome: Ribeiro Alberto
Logradouro: Rua 30, Centro
Cidade: Guarulhos
Estado: SP
Telefone: 1111-1111
Email: ribeiro@sp.com.br
CPF: 11111111111
=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====
```

- Exibir todos

```

Commit
CadastroBDTeste.java PessoaFisicaDAO.java PessoaFisica.java console Database

Run CadastroBDTeste
=====
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
F
Exibindo dados de Pessoas Fisicas
=====
ID: 3
Nome: Ribeiro
Logradouro: Rua 30, Centro
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: ribeiro@sp.com.br
CPF: 111111111111

ID: 5
Nome: Alberto Ribeiro Sales
Logradouro: Rua 30, Centro
Cidade: Guarulhos
Estado: SP
Telefone: 1111-1111
Email: teste@teste.com.br
CPF: 12345678900

ID: 22
Nome: Ribeiro Alberto
Logradouro: Rua 30, Centro
Cidade: Guarulhos
Estado: SP
Telefone: 1111-1111
Email: ribeiro@sp.com.br
CPF: 111111111111

```

```

ID: 36
Nome: Pedrinho
Logradouro: Rua do Pedrinho, Centro
Cidade: Maringá
Estado: SP
Telefone: 1111-1111
Email: pedrinho@uol.com.br
CPF: 12345678900

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

```

Pessoa Juridica

- Inserir


```
Commit
Run CadastroBDTeste.java PessoaFisicaDAO.java PessoaFisica.java console Database

/opt/homebrew/Cellar/openjdk/21.0.2/libexec/openjdk.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=49883:/Applications/IntelliJ IDEA.app/Contente

=====
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
Insira os dados da Pessoa Juridica
=====
Digite o nome:
Maria Oliveira LTDA
Digite o logradouro:
Avenida dos Girassóis, 456
Digite a cidade:
Rio de Janeiro
Digite o estado (2 caracteres):
RJ
Digite o telefone:
9876-5432
Digite o email:
maria.oliveira@example.com
Digite o CNPJ:
12345678912345
Pessoa Juridica salva com sucesso

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

Changes

Amend

2* Procedimento | Alimentando a Base

Commit Commit and Push...

```
1 select *
2 from pessoa as p inner join pessoaFisica as pf on p.idPessoa = pf.idPessoa;
3
4 select *
5 from pessoa as p inner join pessoaJuridica as pj on p.idPessoa = pj.idPessoa;
6
```

Database

loja@localhost 1 of 5

loja 1 of 53

dbo

tables

movimento

pessoa

pessoaFisica

pessoaJuridica

produto

usuario

sequences

Database Objects

Server Objects

Services

Output

Result 6

CSV

4 rows

p.idPessoa	nome	logradouro	cidade	estado	telefone	email	pj.idPessoa	cnpj
2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	2	22222222222222
21	Ribeiro LTDA	Rua 30, Centro	São Paulo	SP	1111-1111	ribeiro-ltda@sp.com.br	21	11111111111111
25	Ribeiro LTDA	Rua 30, Centro	São Paulo	SP	1111-1111	ribeiro-ltda@sp.com.br	25	11111111111111
52	Maria Oliveira LTDA	Avenida dos Girassóis, 456	Rio de Janeiro	RJ	9876-5432	maria.oliveira@example.com	52	12345678912345

Database Consoles > loja@localhost > console

SUM: 12345678912449 9 cells, 1 row 4:1 4:1 (86 chars, 1 line break) LF UTF-8 4 spaces n

- Alterar


```
Commit
Run
CadastroBDTeste.java PessoaFisicaDAO.java PessoaFisica.java console Database

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

F - Pessoa Fisica | J - Pessoa Juridica
J
Insira os dados da Juridica
Digite o ID:
52
Digite o nome:
Maria Oliveira ME
Digite o logradouro:
Avenida dos Girassóis, 380
Digite a cidade:
Rio de Janeiro
Digite o estado (2 caracteres):
RJ
Digite o telefone:
1234-5678
Digite o email:
maria.oliveira@example.com
Digite o CNPJ:
12345678912345
Pessoa Juridica atualizada com sucesso.

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

Changes

Amend

2* Procedimento | Alimentando a Base

Commit Commit and Push...

```
1. select *
2. from pessoa as p inner join pessoaFisica as pf on p.idPessoa = pf.idPessoa;
3.
4. select *
5. from pessoa as p inner join pessoaJuridica as pj on p.idPessoa = pj.idPessoa;
6.
```

Database

loja@localhost 1 of 5

dbo

tables

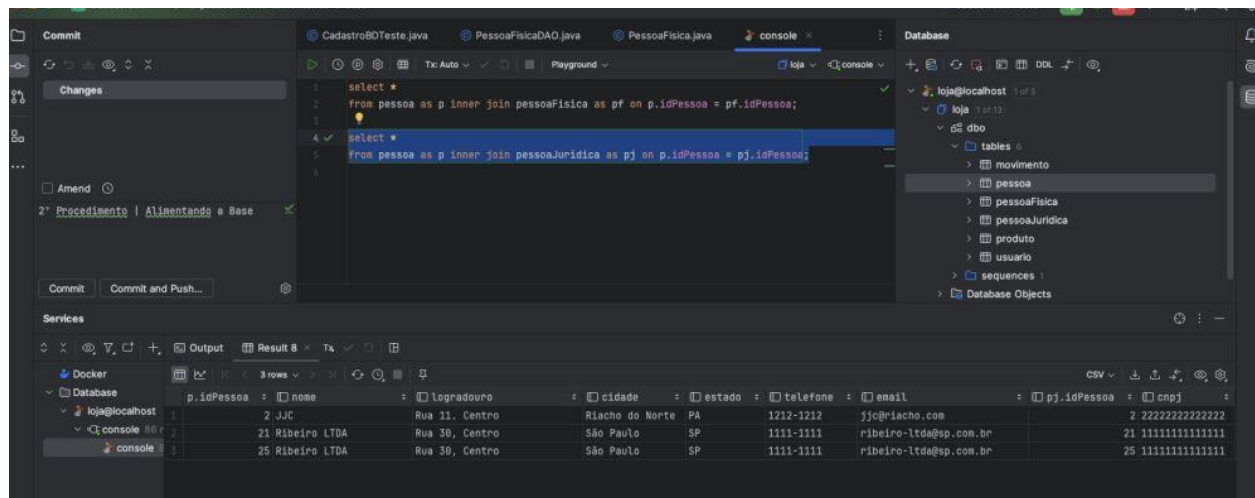
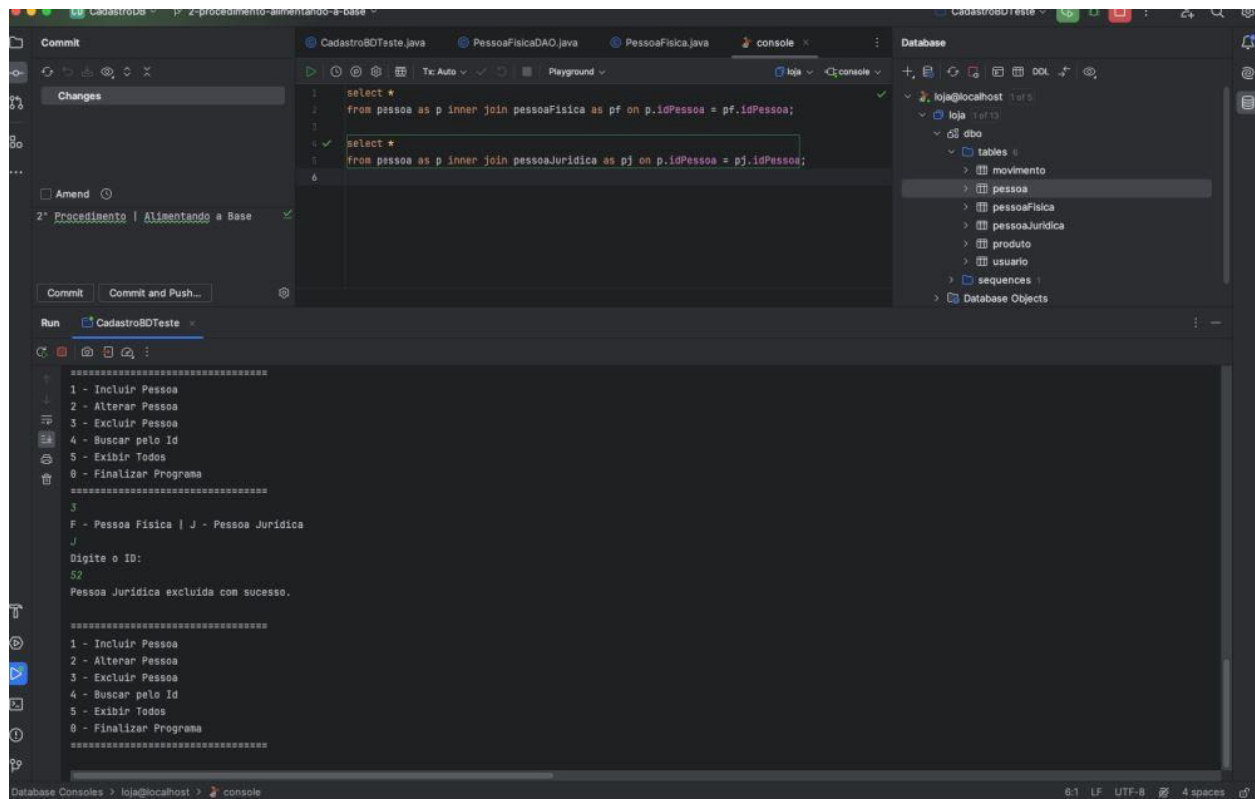
- movimento
- pessoa
- pessoaFisica
- pessoaJuridica
- produto
- usuario
- sequences 1
- Database Objects
- Server Objects

Services

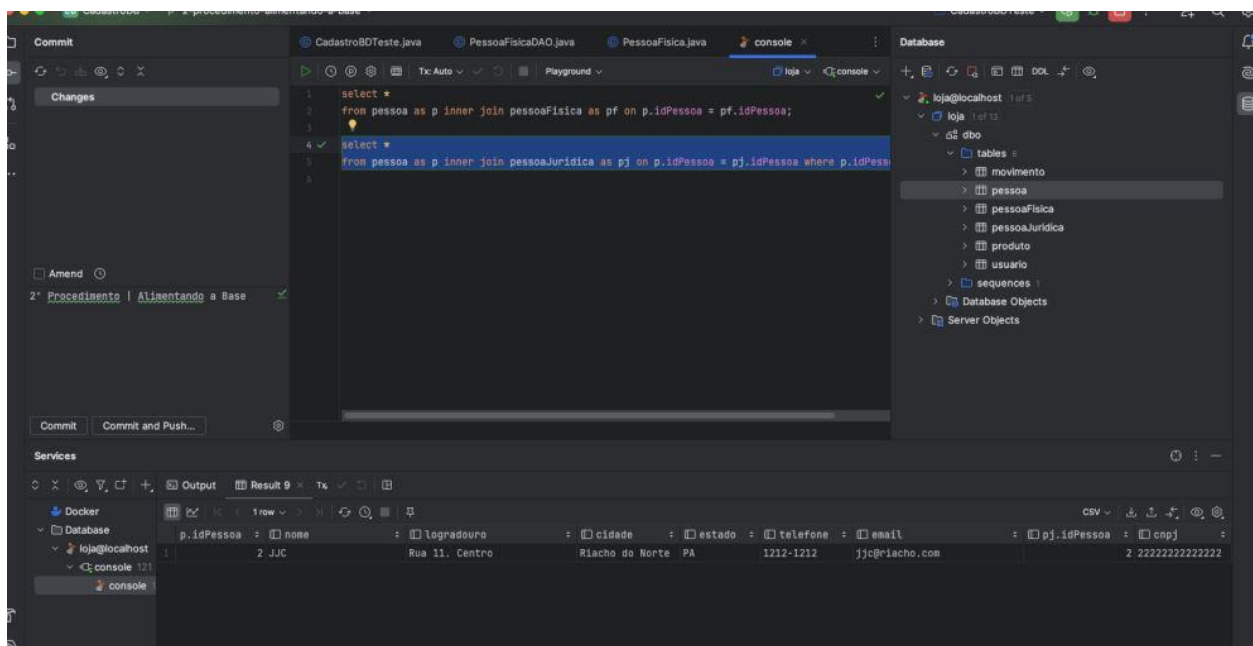
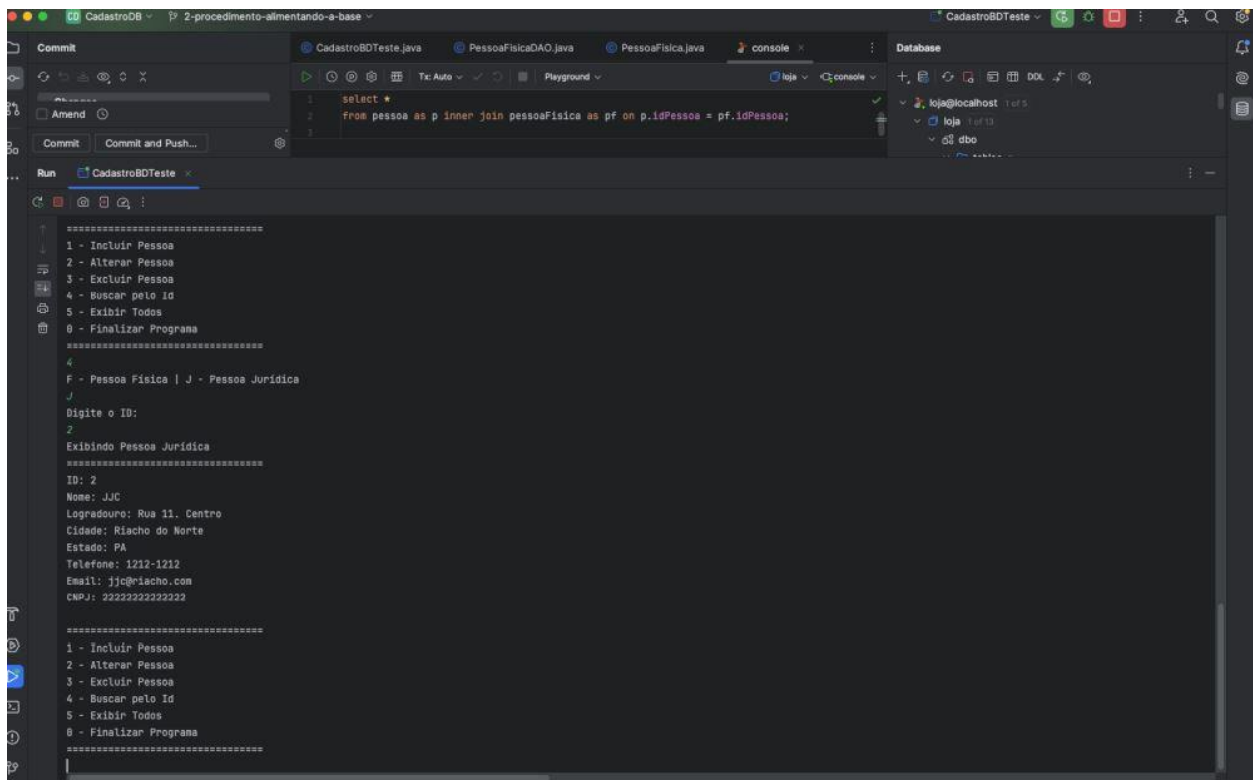
Output Result 7 Tx

	p.idPessoa	nome	logradouro	cidade	estado	telefone	email	pj.idPessoa	cnpj
1	2	JJC	Rua 11, Centro	Riacho do Norte	PA	1212-1212	jjc@riacho.com	2	222222222222222
2	21	Ribeiro LTDA	Rua 30, Centro	São Paulo	SP	1111-1111	ribeiro-ltda@sp.com.br	21	111111111111111
3	25	Ribeiro LTDA	Rua 30, Centro	São Paulo	SP	1111-1111	ribeiro-ltda@sp.com.br	25	111111111111111
4	52	Maria Oliveira ME	Avenida dos Girassóis, 380	Rio de Janeiro	RJ	1234-5678	maria.oliveira@example.com	52	12345678912345

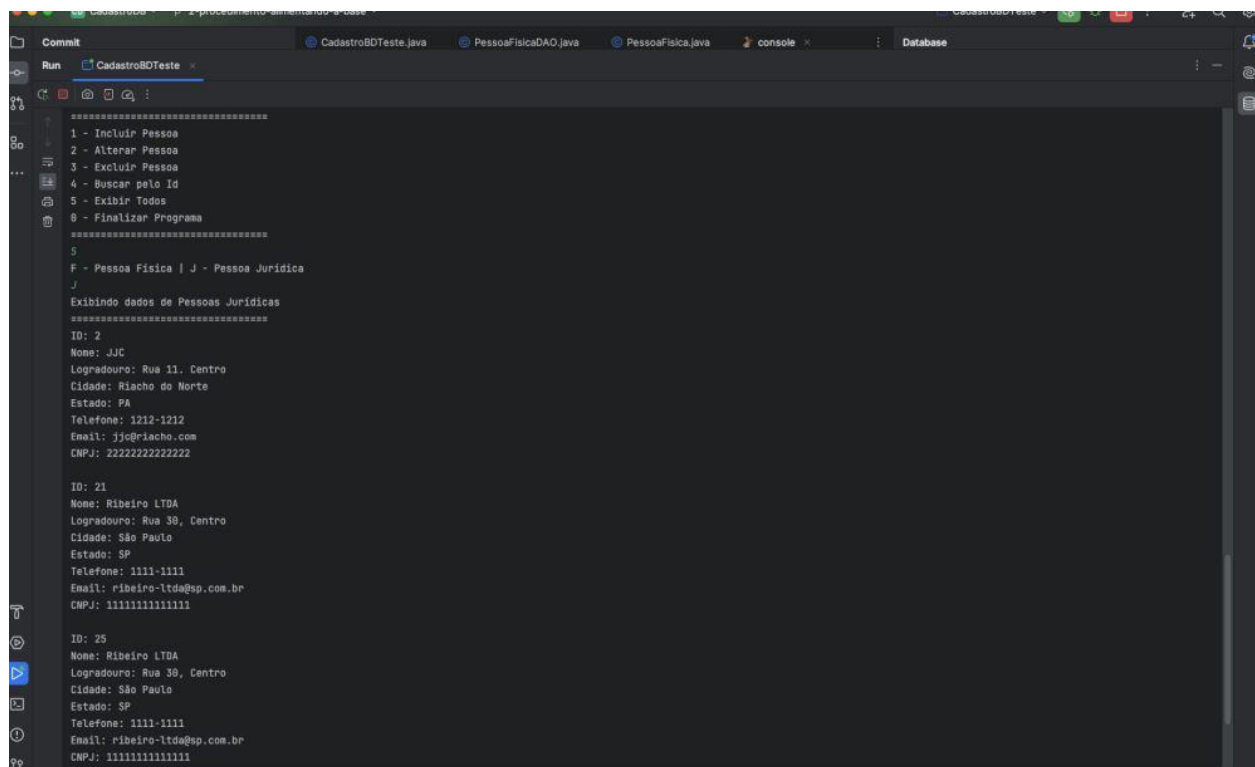
- Excluir



- Buscar por ID



- Exibir todos



```
Commit
Run CadastroBDTeste
Database

=====
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir Todos
0 - Finalizar Programa
=====
0
P - Pessoa Fisica | J - Pessoa Juridica
J
Exibindo dados de Pessoas Juridicas
=====
ID: 2
Nome: JJC
Logradouro: Rua 11, Centro
Cidade: Riacho do Norte
Estado: PA
Telefone: 1212-1212
Email: jjc@riacho.com
CNPJ: 2222222222222222

ID: 21
Nome: Ribeiro LTDA
Logradouro: Rua 38, Centro
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: ribeiro-ltda@sp.com.br
CNPJ: 1111111111111111

ID: 25
Nome: Ribeiro LTDA
Logradouro: Rua 38, Centro
Cidade: São Paulo
Estado: SP
Telefone: 1111-1111
Email: ribeiro-ltda@sp.com.br
CNPJ: 1111111111111111
```

Análise e Conclusão

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

A persistência em arquivos é uma prática que dificulta o processo de leitura, alteração e gravação dos dados, visto que depende da integridade de um arquivo gravado em disco, que pode ser corrompido ou até mesmo lido por qualquer outra fonte que tenha acesso ao disco rígido, além de tornar o processo de backup mais moroso, pois envolve processos manuais de versionamento dos backups, ele deve ser utilizado em cenários onde o volume de dados e volume de acessos é bem baixo, já o armazenamento em banco de dados permite uma flexibilidade maior, pois qualquer aplicação pode acessar o banco utilizando as credenciais de acesso, o que deixa o acesso e o controle do acesso aos dados mais protegido e possibilita a integração com diversas aplicações alimentadas pelas mesmas informações, além de permitir ler, gravar e manipular os dados com maior facilidade, ou executar filtragem avançada dos dados por possuir métodos nativos, o banco de dados também possui na maioria das vezes um próprio sistema de backup que pode ser configurado e seu versionamento é auto gerenciado

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

Nas versões mais recentes do Java é possível imprimir valores utilizando menos código, que permite informar para um loop `forEach` por exemplo, como cada dado deve ser impresso na tela, ou até mesmo somente informar qual método de uma classe deve ser invocado para cada item do loop, isso facilita a compressão e leitura do código e reduz de forma significável o boilerplate que era causado por loops tradicionais como `for` ou `foreach`

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

O java inicia a aplicação invocando diretamente o método `main` da classe principal sem instanciar a classe, se o método `main` não for do tipo `static` a JVM não irá conseguir iniciar a aplicação. O `static` indica que o método faz parte da classe e pode ser invocado diretamente sem estar vinculado a um objeto em específico, assim como todos os métodos que são invocados diretamente, sem instâncias de objetos, precisam ser do tipo `static`, para que possam ser executados sem objetos vinculados.