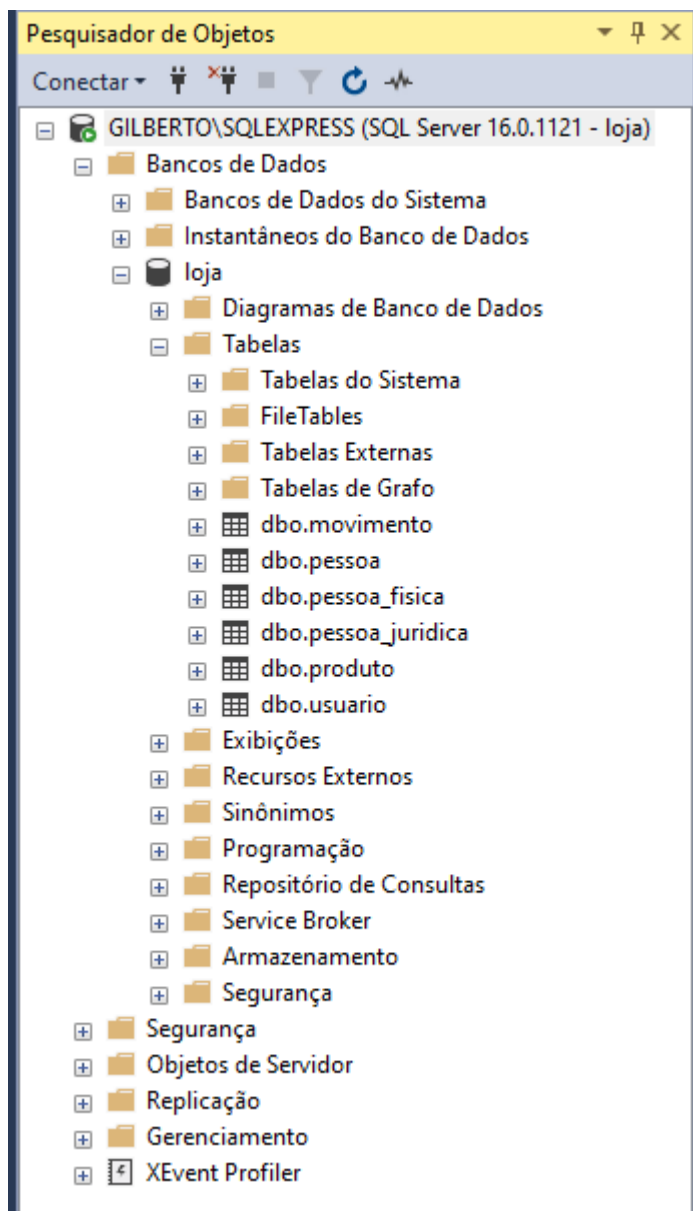


Disciplina: RPG0016 – Backend sem banco de dados não tem
Nome: João Gilberto dos Santos
Turma: 2022.4

1º Título da Prática: Criando o Banco de Dados

2º Objetivo da Prática

O objetivo deste relatório é demonstrar os resultados obtidos durante a criação de um banco de dados e toda sua funcionalidade.



3º Códigos Solicitados:

```
USE Loja;  
GO
```

```
CREATE TABLE [pessoa] (  
    id_pessoa integer NOT NULL IDENTITY(1,1),  
    nome varchar(255) NOT NULL,  
    endereco varchar(255) NOT NULL,  
    cidade varchar(255) NOT NULL,  
    estado char(2) NOT NULL,  
    telefone varchar(11) NOT NULL,  
    email varchar(255) NOT NULL,  
    CONSTRAINT [PK_PESSOA] PRIMARY KEY CLUSTERED ([id_pessoa] ASC)  
)  
GO
```

```
CREATE TABLE [pessoa_fisica] (  
    id_pessoa integer NOT NULL,  
    cpf varchar(11) NOT NULL,  
    CONSTRAINT [PK_PESSOA_FISICA] PRIMARY KEY CLUSTERED ([id_pessoa] ASC),  
    CONSTRAINT [FK_PESSOA_FISICA_PESSOA] FOREIGN KEY ([id_pessoa])  
REFERENCES [pessoa] ([id_pessoa])  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
)  
GO
```

```
CREATE TABLE [pessoa_juridica] (  
    id_pessoa integer NOT NULL,  
    cnpj varchar(20) NOT NULL,  
    CONSTRAINT [PK_PESSOA_JURIDICA] PRIMARY KEY CLUSTERED ([id_pessoa] ASC),  
    CONSTRAINT [FK_PESSOA_JURIDICA_PESSOA] FOREIGN KEY ([id_pessoa])  
REFERENCES [pessoa] ([id_pessoa])  
    ON UPDATE CASCADE  
    ON DELETE CASCADE  
)  
GO
```

```
CREATE TABLE [produto] (  
    id_produto integer NOT NULL IDENTITY(1,1),  
    nome varchar(255) NOT NULL,
```

```
    quantidade integer NOT NULL,  
    precoVenda numeric(10,2) NOT NULL,  
    CONSTRAINT [PK_PRODUTO] PRIMARY KEY CLUSTERED ([id_produto] ASC)  
)  
GO
```

```
CREATE TABLE [usuario] (  
    id_usuario integer NOT NULL IDENTITY(1,1),  
    login varchar(25) NOT NULL,  
    senha varchar(25) NOT NULL,  
    CONSTRAINT [PK_USUARIO] PRIMARY KEY CLUSTERED ([id_usuario] ASC)  
)  
GO
```

```
CREATE TABLE [movimento] (  
    id_movimento integer NOT NULL IDENTITY(1,1),  
    id_pessoa integer NOT NULL,  
    id_produto integer NOT NULL,  
    id_usuario integer NOT NULL,  
    quantidade integer NOT NULL,  
    tipo char(1) NOT NULL,  
    valor_unitario numeric(10,2) NOT NULL,  
    CONSTRAINT [PK_MOVIMENTO] PRIMARY KEY CLUSTERED ([id_movimento] ASC)  
)  
GO
```

```
ALTER TABLE [movimento] WITH CHECK ADD CONSTRAINT [movimento_fk0] FOREIGN  
KEY ([id_pessoa])  
REFERENCES [pessoa] ([id_pessoa])  
ON UPDATE CASCADE  
ON DELETE CASCADE  
GO  
ALTER TABLE [movimento] CHECK CONSTRAINT [movimento_fk0]  
GO
```

```
ALTER TABLE [movimento] WITH CHECK ADD CONSTRAINT [movimento_fk1] FOREIGN  
KEY ([id_produto])  
REFERENCES [produto] ([id_produto])  
ON UPDATE CASCADE  
GO  
ALTER TABLE [movimento] CHECK CONSTRAINT [movimento_fk1]  
GO
```

```
ALTER TABLE [movimento] WITH CHECK ADD CONSTRAINT [movimento_fk2] FOREIGN  
KEY ([id_usuario])
```

```
REFERENCES [usuario] ([id_usuario])
ON UPDATE CASCADE
GO
ALTER TABLE [movimento] CHECK CONSTRAINT [movimento_fk2]
GO
```

4º Os resultados da execução dos códigos:

4.1 Tabela dbo.pessoa

Gilberto\SQLEXPRESS.loja - dbo.pessoa			
	Nome da Coluna	Tipo de Dados	Permitir Nul...
▶	id_pessoa	int	<input type="checkbox"/>
	nome	varchar(255)	<input type="checkbox"/>
	endereço	varchar(255)	<input type="checkbox"/>
	cidade	varchar(255)	<input type="checkbox"/>
	estado	char(2)	<input type="checkbox"/>
	telefone	varchar(11)	<input type="checkbox"/>
	email	varchar(255)	<input type="checkbox"/>
			<input type="checkbox"/>

4.2 Tabela dbo.pessoa_fisica

Gilberto\SQLEXPRES...dbo.pessoa_fisica			
create_db (1).sql - G...RESS.loja (loja (59))			
	Nome da Coluna	Tipo de Dados	Permitir Nul...
▶	id_pessoa	int	<input type="checkbox"/>
	cpf	varchar(11)	<input type="checkbox"/>
			<input type="checkbox"/>

4.3 Tabela dbo.pessoa_juridica

Gilberto\SQLEXPRES...bo.pessoa_juridica			
create_db (1).sql - G...RESS.loja (loja (59))			
	Nome da Coluna	Tipo de Dados	Permitir Nul...
▶	id_pessoa	int	<input type="checkbox"/>
	cnpj	varchar(20)	<input type="checkbox"/>
			<input type="checkbox"/>

4.4 Tabela dbo.produto

	Nome da Coluna	Tipo de Dados	Permitir Nul...
▶	id_produto	int	<input type="checkbox"/>
	nome	varchar(255)	<input type="checkbox"/>
	quantidade	int	<input type="checkbox"/>
	precoVenda	numeric(10, 2)	<input type="checkbox"/>
			<input type="checkbox"/>

4.5 Tabela dbo.usuario

	Nome da Coluna	Tipo de Dados	Permitir Nul...
▶	id_usuario	int	<input type="checkbox"/>
	login	varchar(25)	<input type="checkbox"/>
	senha	varchar(25)	<input type="checkbox"/>
			<input type="checkbox"/>

5° Análise e Conclusão:

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

R: Em termos simples, o middleware é um software que atua como uma ponte entre diferentes aplicativos e sistemas. Ele facilita a comunicação e a integração entre diversas tecnologias, como bancos de dados, sistemas operacionais e linguagens de programação.

Abstração: O middleware esconde a complexidade da infraestrutura subjacente, permitindo que os desenvolvedores se concentrem na lógica de negócios da aplicação.

Reutilização: Componentes de middleware podem ser reutilizados em diversos projetos, acelerando o desenvolvimento e reduzindo custos.

Interoperabilidade: O middleware facilita a integração de sistemas heterogêneos, permitindo que diferentes aplicações se comuniquem de forma eficiente.

Escalabilidade: Muitos componentes de middleware são projetados para suportar cargas de trabalho crescentes, garantindo a performance e a disponibilidade dos sistemas.

O JDBC (Java Database Connectivity) é um API (Interface de Programação de Aplicativos) para conectar aplicações Java a bancos de dados. Ele é um dos componentes de middleware mais utilizados no mundo.

Abstração de Bancos de Dados: O JDBC fornece uma interface padrão para acessar diferentes bancos de dados (MySQL, Oracle, PostgreSQL, etc.), permitindo que os desenvolvedores escrevam código que seja independente do banco de dados utilizado.

Simplificação do Acesso a Dados: O JDBC simplifica tarefas comuns como a execução de consultas SQL, a manipulação de resultados e o tratamento de exceções.

Integração com Outras Tecnologias: O JDBC se integra facilmente com outras tecnologias Java, como frameworks de persistência (Hibernate, JPA) e servidores de aplicação (Tomcat, JBoss).

B° Qual a diferença no uso de Statement ou PreparedStatement para a manipulação de dados?

R: A escolha entre Statement e PreparedStatement para manipulação de dados em Java, especialmente quando se trabalha com bancos de dados relacionais via JDBC, é crucial para otimizar o desempenho e a segurança das suas aplicações.

Statement:

Criação: A cada execução de uma consulta SQL, um novo objeto Statement é criado.

Preenchimento: A consulta SQL é concatenada diretamente com os valores dos parâmetros, o que pode levar a problemas de injeção de SQL.

PreparedStatement:

Criação: Um único objeto PreparedStatement é criado para uma determinada consulta SQL, com placeholders para os parâmetros.

Preenchimento: Os valores dos parâmetros são definidos separadamente, utilizando métodos como `setString()`, `setInt()`, etc., evitando problemas de injeção de SQL.

C° Como o padrão DAO melhora a manutenibilidade do software?

R: O padrão DAO (Data Access Object) é uma ferramenta poderosa na construção de aplicações, especialmente quando se trata de manter a organização e a flexibilidade do código. Ao separar a lógica de acesso a dados da lógica de negócio, o DAO contribui significativamente para a manutenibilidade do software.

Principais Benefícios para a Manutenibilidade:

Abstração da Fonte de Dados: Isolamento: O DAO cria uma camada de abstração entre o código da aplicação e a fonte de dados (banco de dados, arquivo, etc.). Isso significa que você pode mudar a fonte de dados sem afetar a lógica de negócios.

Flexibilidade: Ao trocar a fonte de dados, é necessário apenas modificar a implementação do DAO, mantendo o restante do código intacto.

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

R: A herança, um conceito fundamental na programação orientada a objetos, não tem um mapeamento direto para o modelo relacional de dados. Isso ocorre porque os bancos relacionais são baseados em tabelas, enquanto a herança implica em hierarquias de classes.

No entanto, existem algumas estratégias comuns para modelar a herança em bancos relacionais:

Em Resumo, modelar a herança em bancos relacionais exige um cuidadoso planejamento e pode envolver trade-offs entre diferentes estratégias. A escolha da melhor abordagem dependerá das necessidades específicas da aplicação e do banco de dados.

1º Título da Prática: Conectando ao Banco de Dados

2º Objetivo da Prática

O objetivo deste relatório é demonstrar os resultados obtidos após a implementação de um banco de dados e seu funcionamento conforme foi solicitado pelo docente desta materia.

The screenshot displays an IDE interface for a Java project named 'CadastroBD'. The 'Projects' pane on the left shows the project structure, including 'Source Packages' and 'Test Packages'. The 'Navigator' pane shows the 'Members' of the 'CadastroBD' package, listing various methods like 'alterarPessoa', 'buscarPessoaPelold', 'cadastrarPessoaFisica', etc. The 'Source' pane shows the code for 'CadastroBDTeste.java', which includes imports for 'conexaobd.model.util.ConectorBD', 'java.sql.Connection', and 'java.sql.SQLException'. The 'Output - CadastroBD (run)' pane shows the execution results, including a successful database connection message and a menu of options for the application.

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Comment
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Comment
4  */
5  package conexaobd;
6
7  import conexaobd.model.util.ConectorBD;
8  import java.sql.Connection;
9  import java.sql.SQLException;
10
11  /**
12   *
```

run:
Banco de Dados Conectado com Sucesso

1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair

Escolha uma opcao:
1
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
f
Digite o nome para a pessoa fisica:

3º Códigos Solicitados:

3.1

```
package cadastrobd;

import cadastrobd.model.PessoaFisicaDAO;
import cadastrobd.model.PessoaJuridicaDAO;
import cadastrobd.model.PessoaFisica;
import cadastrobd.model.PessoaJuridica;
import cadastrobd.model.util.ConectorBD;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;
import java.util.Scanner;

/**
 *
 * @author mrjoa
 */
public class CadastroBD {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        Connection conn = ConectorBD.getConnection();
        PessoaFisicaDAO pessoaFisicaDAO = new PessoaFisicaDAO(conn);
        PessoaJuridicaDAO pessoaJuridicaDAO = new PessoaJuridicaDAO(conn);

        int escolha;

        do {
            System.out.println("#####");
            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 - Listar todos");
            System.out.println("6 - Lista de pessoas fisicas");
```



```

System.out.println("7 - Lista de pessoas juridicas");
System.out.println("0 - Sair");
System.out.println("#####");
System.out.println("Escolha uma opcao: ");
escolha = scanner.nextInt();
scanner.nextLine();

try {
    switch (escolha) {
        case 1:
            System.out.println("(F) - Pessoa Fisica | (J) - Pessoa juridica");
            System.out.println("Escolha uma opcao: ");
            char tipoInclusao = scanner.next().charAt(0);
            scanner.nextLine();

            if (tipoInclusao == 'F' || tipoInclusao == 'f') {
                cadastrarPessoaFisica(pessoaFisicaDAO, scanner);
            } else if (tipoInclusao == 'J' || tipoInclusao == 'j') {
                cadastrarPessoaJuridica(pessoaJuridicaDAO, scanner);
            } else {
                System.out.println("## > A opcao escolhida invalida.");
            }
            break;

        case 2:
            alterarPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
            break;

        case 3:
            excluirPessoa(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
            break;

        case 4:
            buscarPessoaPeloid(pessoaFisicaDAO, pessoaJuridicaDAO, scanner);
            break;

        case 5:
            exibirTodasPessoas(pessoaFisicaDAO, pessoaJuridicaDAO);
            break;

        case 6:
            exibirPessoasFisicas(pessoaFisicaDAO);
            break;

        case 7:

```

```

        exibirPessoasJuridicas(pessoaJuridicaDAO);
        break;
    case 0:
        System.out.println("## > Saindo....");
        break;
    default:
        System.out.println("## > Opcao invalida. Verifique a opcao escolhida e tente novamente");
    }
} catch (SQLException e) {
    System.out.println("## > Erro de banco de dados: " + e.getMessage());
}
} while (escolha != 0);
}

```

// Métods para cadastrar Pessoa Fisica

```

private static void cadastrarPessoaFisica(PessoaFisicaDAO pessoaFisicaDAO, Scanner scanner) throws SQLException {
    System.out.println("Digite o nome para a pessoa fisica:");
    String nome = scanner.nextLine();
    System.out.println("Digite o endereco:");
    String logradouro = scanner.nextLine();
    System.out.println("Digite a cidade:");
    String cidade = scanner.nextLine();
    System.out.println("Digite o estado:");
    String estado = scanner.nextLine();
    System.out.println("Digite o telefone:");
    String telefone = scanner.nextLine();
    System.out.println("Digite o e-mail:");
    String email = scanner.nextLine();
    System.out.println("Digite o CPF:");
    String cpf = scanner.nextLine();

    PessoaFisica novaPessoaFisica = new PessoaFisica(0, nome, logradouro, cidade, estado, telefone, email, cpf);
    pessoaFisicaDAO.inserirPessoaFisica(novaPessoaFisica);
    System.out.println("## > Pessoa Fisica cadastrada com sucesso!");
}

```

// Métods para cadastrar Pessoa Juridica

```

private static void cadastrarPessoaJuridica(PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner) throws SQLException {
    System.out.println("Digite o nome para a pessoa juridica:");
}

```

```

String nome = scanner.nextLine();
System.out.println("Digite o endereço:");
String logradouro = scanner.nextLine();
System.out.println("Digite a cidade:");
String cidade = scanner.nextLine();
System.out.println("Digite o estado:");
String estado = scanner.nextLine();
System.out.println("Digite o telefone:");
String telefone = scanner.nextLine();
System.out.println("Digite o e-mail:");
String email = scanner.nextLine();
System.out.println("Digite o CNPJ:");
String cnpj = scanner.nextLine();

```

```

PessoaJuridica novaPessoaJuridica = new PessoaJuridica(0, nome, logradouro, cidade,
estado, telefone, email, cnpj);
pessoaJuridicaDAO.incluir(novaPessoaJuridica);
System.out.println("## > Pessoa Jurídica cadastrada com sucesso.");
}

```

// Métodos para alterar Pessoas

```

private static void alterarPessoa(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO
pessoaJuridicaDAO, Scanner scanner) throws SQLException {

```

```

    System.out.println("F - Alterar pessoa fisica | J - Alterar pessoa juridica");
    System.out.println("Escolha uma opção: ");
    char tipoPessoa = scanner.next().charAt(0);
    scanner.nextLine();
    int id;

```

```

    if (tipoPessoa == 'F' || tipoPessoa == 'f') {
        System.out.println("Digite o ID da pessoa fisica que deseja alterar:");
        id = scanner.nextInt();
        scanner.nextLine();
    }

```

```

    PessoaFisica pessoaExistente = pessoaFisicaDAO.getPessoa(id);

```

```

    if (pessoaExistente != null) {
        System.out.println("Digite o novo nome da pessoa fisica:");
        String novoNome = scanner.nextLine();
        System.out.println("Digite o novo endereço:");
        String novoLogradouro = scanner.nextLine();
        System.out.println("Digite a nova cidade:");
        String novaCidade = scanner.nextLine();
    }

```

```
System.out.println("Digite o novo estado:");
String novoEstado = scanner.nextLine();
System.out.println("Digite o novo telefone:");
String novoTelefone = scanner.nextLine();
System.out.println("Digite o novo e-mail:");
String novoEmail = scanner.nextLine();
System.out.println("Digite o novo CPF:");
String novoCpf = scanner.nextLine();
```

```
PessoaFisica novaPessoa = new PessoaFisica(id, novoNome, novoLogradouro,
novaCidade, novoEstado, novoTelefone, novoEmail, novoCpf);
```

```
    pessoaFisicaDAO.alterar(novaPessoa);
```

```
    System.out.println("## > Pessoa fisica atualizada com sucesso.");
```

```
    } else {
```

```
        System.out.println("## > Pessoa fisica não encontrada.");
```

```
    }
```

```
    } else if (tipoPessoa == 'J' || tipoPessoa == 'j') {
```

```
        System.out.println("Digite o ID da pessoa juridica que deseja alterar:");
```

```
        if (scanner.hasNextInt()) {
```

```
            id = scanner.nextInt();
```

```
            System.out.println("Voce adicionou o ID: " + id);
```

```
        } else {
```

```
            String input = scanner.next();
```

```
            System.out.println("Nao e permitido Strings: " + input);
```

```
            return;
```

```
        }
```

```
        scanner.nextLine();
```

```
PessoaJuridica pessoaExistente = pessoaJuridicaDAO.getPessoa(id);
```

```
if (pessoaExistente != null) {
```

```
    System.out.println("Digite o novo nome da pessoa juridica:");
```

```
    String novoNome = scanner.nextLine();
```

```
    System.out.println("Digite o novo endereco:");
```

```
    String novoLogradouro = scanner.nextLine();
```

```
    System.out.println("Digite a nova cidade:");
```

```
    String novaCidade = scanner.nextLine();
```

```
    System.out.println("Digite o novo estado:");
```

```
    String novoEstado = scanner.nextLine();
```

```
    System.out.println("Digite o novo telefone:");
```

```
    String novoTelefone = scanner.nextLine();
```

```
System.out.println("Digite o novo email:");
String novoEmail = scanner.nextLine();
System.out.println("Digite o novo CNPJ:");
String novoCnpj = scanner.nextLine();
```

```
PessoaJuridica novaPessoa = new PessoaJuridica(id, novoNome, novoLogradouro,
novaCidade, novoEstado, novoTelefone, novoEmail, novoCnpj);
pessoaJuridicaDAO.alterar(novaPessoa);
System.out.println("## > ## >Pessoa juridica atualizada com sucesso.");
} else {
    System.out.println("## > Pessoa juridica não encontrada.");
}
} else {
    System.out.println("## > Opcao invalida.");
}
}
```

// Métods para excluir Pessoas

```
private static void excluirPessoa(PessoaFisicaDAO pessoaFisicaDAO, PessoaJuridicaDAO
pessoaJuridicaDAO, Scanner scanner) throws SQLException {
    System.out.println("(F) - Excluir pessoa fisica | (J) - Excluir pessoa juridica");
    System.out.println("Escolha uma opcao: ");
    char tipoPessoa = scanner.next().charAt(0);
    scanner.nextLine();

    if (tipoPessoa == 'F' || tipoPessoa == 'f') {
        System.out.println("Digite o ID da pessoa fisica para ser removida:");
        int id = scanner.nextInt();
        pessoaFisicaDAO.excluir(id);
        System.out.println("## > Pessoa fisica removida com sucesso.");
    } else if (tipoPessoa == 'J' || tipoPessoa == 'j') {
        System.out.println("Digite o ID da pessoa juridica para ser removida:");
        int id = scanner.nextInt();
        pessoaJuridicaDAO.excluir(id);
        System.out.println("## > Pessoa juridica excluida com sucesso.");
    } else {
        System.out.println("## > Opcao invalida");
    }
}
```

// Métods para buscar Pessoas

```
private static void buscarPessoaPeloid(PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO, Scanner scanner) throws SQLException {
    System.out.println("(F) - Buscar pessoa fisica | (J) - Buscar pessoa juridica");
```

```
System.out.println("Escolha uma opcao: ");
char tipoPessoa = scanner.next().charAt(0);
scanner.nextLine();
```

```
if (tipoPessoa == 'F' || tipoPessoa == 'f') {
    System.out.println("Digite o ID da pessoa fisica:");
    int id = scanner.nextInt();
    scanner.nextLine();
    PessoaFisica pessoaFisica = pessoaFisicaDAO.getPessoa(id);

    if (pessoaFisica != null) {
        System.out.println("Informacoes da pessoa fisica:");
        System.out.println("ID: " + pessoaFisica.getId());
        System.out.println("Nome: " + pessoaFisica.getNome());
        System.out.println("Endereco: " + pessoaFisica.getEndereco());
        System.out.println("Cidade: " + pessoaFisica.getCidade());
        System.out.println("Estado: " + pessoaFisica.getEstado());
        System.out.println("Telefone: " + pessoaFisica.getTelefone());
        System.out.println("E-mail: " + pessoaFisica.getEmail());
        System.out.println("CPF: " + pessoaFisica.getCpf());
    } else {
        System.out.println("## > Nao foi possivel encontrar a pessoa fisica");
    }
} else if (tipoPessoa == 'J' || tipoPessoa == 'j') {
    System.out.println("Digite o ID da pessoa juridica:");
    int id = scanner.nextInt();
    scanner.nextLine();
```

```
PessoaJuridica pessoaJuridica = pessoaJuridicaDAO.getPessoa(id);
```

```
if (pessoaJuridica != null) {
    System.out.println("Informacoes da pessoa juridica:");
    System.out.println("ID: " + pessoaJuridica.getId());
    System.out.println("Nome: " + pessoaJuridica.getNome());
    System.out.println("Endereco: " + pessoaJuridica.getEndereco());
    System.out.println("Cidade: " + pessoaJuridica.getCidade());
    System.out.println("Estado: " + pessoaJuridica.getEstado());
    System.out.println("Telefone: " + pessoaJuridica.getTelefone());
    System.out.println("E-mail: " + pessoaJuridica.getEmail());
    System.out.println("CNPJ: " + pessoaJuridica.getCnpj());
} else {
    System.out.println("## > Nao foi possivel encontrar a pessoa juridica.");
}
```

```

    } else {
        System.out.println("## > Opcao invalida.");
    }
}

```

// Métods para exibir todas as Pessoas

```

private static void exibirTodasPessoas(PessoaFisicaDAO pessoaFisicaDAO,
PessoaJuridicaDAO pessoaJuridicaDAO) throws SQLException {

```

```

    System.out.println("Relacao de pessoas fisicas cadastradas:");
    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
    if (pessoasFisicas.isEmpty()) {
        System.out.println("## > Nenhuma cadastro de pessoa fisica.");
    } else {

```

```

        for (PessoaFisica pf : pessoasFisicas) {
            System.out.println("ID: " + pf.getId());
            System.out.println("Nome: " + pf.getNome());
            System.out.println("Endereco: " + pf.getEndereco());
            System.out.println("Cidade: " + pf.getCidade());
            System.out.println("Estado: " + pf.getEstado());
            System.out.println("Telefone: " + pf.getTelefone());
            System.out.println("E-mail: " + pf.getEmail());
            System.out.println("CPF: " + pf.getCpf());
            System.out.println();
        }
    }
}

```

```

    System.out.println("Relacao de pessoas juridicas cadastradas:");
    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoasJuridicas();
    if (pessoasJuridicas.isEmpty()) {
        System.out.println("## > Nenhuma pessoa juridica cadastrada.");
    } else {

```

```

        for (PessoaJuridica pj : pessoasJuridicas) {
            System.out.println("ID: " + pj.getId());
            System.out.println("Nome: " + pj.getNome());
            System.out.println("Endereco: " + pj.getEndereco());
            System.out.println("Cidade: " + pj.getCidade());
            System.out.println("Estado: " + pj.getEstado());
            System.out.println("Telefone: " + pj.getTelefone());
            System.out.println("E-mail: " + pj.getEmail());
            System.out.println("CNPJ: " + pj.getCnpj());
            System.out.println();
        }
    }
}

```

```
}
```

```
// Métods para exibir Pessoas Fisicas
```

```
private static void exibirPessoasFisicas(PessoaFisicaDAO pessoaFisicaDAO) throws  
SQLException {
```

```
    List<PessoaFisica> pessoasFisicas = pessoaFisicaDAO.getPessoas();
```

```
    if (pessoasFisicas.isEmpty()) {
```

```
        System.out.println("## > Nenhuma pessoa fisica cadastrada.");
```

```
    } else {
```

```
        for (PessoaFisica pf : pessoasFisicas) {
```

```
            System.out.println("ID: " + pf.getId());
```

```
            System.out.println("Nome: " + pf.getNome());
```

```
            System.out.println("Endereco: " + pf.getEndereco());
```

```
            System.out.println("Cidade: " + pf.getCidade());
```

```
            System.out.println("Estado: " + pf.getEstado());
```

```
            System.out.println("Telefone: " + pf.getTelefone());
```

```
            System.out.println("E-mail: " + pf.getEmail());
```

```
            System.out.println("CPF: " + pf.getCpf());
```

```
            System.out.println();
```

```
        }
```

```
    }
```

```
}
```

```
// Métods para exibir Pessoas Fisicas
```

```
private static void exibirPessoasJuridicas(PessoaJuridicaDAO pessoaJuridicaDAO) throws  
SQLException {
```

```
    List<PessoaJuridica> pessoasJuridicas = pessoaJuridicaDAO.getPessoasJuridicas();
```

```
    if (pessoasJuridicas.isEmpty()) {
```

```
        System.out.println("## > Nenhuma pessoa juridica cadastrada.");
```

```
    } else {
```

```
        for (PessoaJuridica pj : pessoasJuridicas) {
```

```
            System.out.println("ID: " + pj.getId());
```

```
            System.out.println("Nome: " + pj.getNome());
```

```
            System.out.println("Endereco: " + pj.getEndereco());
```

```
            System.out.println("Cidade: " + pj.getCidade());
```

```
            System.out.println("Estado: " + pj.getEstado());
```

```
            System.out.println("Telefone: " + pj.getTelefone());
```

```
            System.out.println("E-mail: " + pj.getEmail());
```

```
            System.out.println("CNPJ: " + pj.getCnpj());
```

```
            System.out.println();
```

```
        }
```

```
    }
```

```
}
```


}

3.2

```
package cadastrabd.model;
```

```
/**
```

```
*
```

```
* @author mrjoa
```

```
*/
```

```
public class Pessoa {
```

```
    private int id;
```

```
    private String nome;
```

```
    private String endereco;
```

```
    private String cidade;
```

```
    private String estado;
```

```
    private String telefone;
```

```
    private String email;
```

```
    public Pessoa() {}
```

```
    public Pessoa(int id, String nome, String endereco, String cidade, String estado, String  
telefone, String email) {
```

```
        this.id = id;
```

```
        this.nome = nome;
```

```
        this.endereco = endereco;
```

```
        this.cidade = cidade;
```

```
        this.estado = estado;
```

```
        this.telefone = telefone;
```

```
        this.email = email;
```

```
    }
```

```
    public void exibir() {
```

```
        System.out.println("ID: " + id);
```

```
        System.out.println("Nome: " + nome);
```

```
        System.out.println("Endereco: " + endereco);
```

```
        System.out.println("Cidade: " + cidade);
```

```
        System.out.println("Estado: " + estado);
```

```
        System.out.println("Telefone: " + telefone);
```

```
        System.out.println("Email: " + email);
```

```
    }
```

```
    public Integer getId() {
```

```
        return id;
    }

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

    public String getNome() {
        return nome;
    }

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

    public String getEndereco() {
        return endereco;
    }

    public void setEndereco(String endereco) {
        this.endereco = endereco;
    }

    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 getTelefone() {
        return telefone;
    }
}
```

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

```
public String getEmail() {  
    return email;  
}
```

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

```
}
```

3.3

```
package cadastrabd.model;
```

```
/**
```

```
*
```

```
* @author mrjoa
```

```
*/
```

```
public class PessoaFisica extends Pessoa {
```

```
    private String cpf;
```

```
    // Construtor padrão
```

```
    public PessoaFisica() {}
```

```
    // Construtor completo
```

```
    public PessoaFisica(int id, String nome, String logradouro, String cidade, String estado,  
String telefone, String email, String cpf) {
```

```
        super(id, nome, logradouro, cidade, estado, telefone, email);
```

```
        this.cpf = cpf;
```

```
    }
```

```
    public String getCpf() {
```

```
        return cpf;
```

```
    }
```

```
    public void setCpf(String cpf) {
```

```
        this.cpf = cpf;
```

```
    }
```

```

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

```

3.4

```

package cadastrabd.model;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author mrjoa
 */
public class PessoaFisicaDAO {

    private final Connection conn;

    public PessoaFisicaDAO(Connection conexao) {
        this.conn = conexao;
    }

    public void inserirPessoaFisica(PessoaFisica pf) throws SQLException {
        String sqlPessoa = "INSERT INTO Pessoa (nome, endereco, cidade, estado, telefone, email) VALUES (?, ?, ?, ?, ?, ?)";
        String sqlPessoaFisica = "INSERT INTO pessoa_fisica (id_pessoa, cpf) VALUES (?, ?)";

        try {
            conn.setAutoCommit(false);
            int pessoald = 0;

            // Inserir na tabela Pessoa
            try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa, Statement.RETURN_GENERATED_KEYS)) {
                stPessoa.setString(1, pf.getNome());
                stPessoa.setString(2, pf.getEndereco());
                stPessoa.setString(3, pf.getCidade());
                stPessoa.setString(4, pf.getEstado());
            }
        }
    }
}

```

```

        stPessoa.setString(5, pf.getTelefone());
        stPessoa.setString(6, pf.getEmail());
        stPessoa.executeUpdate();

        // Obter o ID gerado
        try (ResultSet rs = stPessoa.getGeneratedKeys()) {
            if (rs.next()) {
                pessoald = rs.getInt(1);
            }
        }

        // Inserir na tabela PessoaFisica
        try (PreparedStatement stPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
            stPessoaFisica.setInt(1, pessoald);
            stPessoaFisica.setString(2, pf.getCpf());
            stPessoaFisica.executeUpdate();
        }

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

```

```

public void alterar(PessoaFisica pf) throws SQLException {
    String sqlPessoa = "UPDATE Pessoa SET nome = ?, endereco = ?, cidade = ?, estado = ?, telefone = ?, email = ? WHERE id_pessoa = ?";
    String sqlPessoaFisica = "UPDATE pessoa_fisica SET cpf = ? WHERE id_pessoa = ?";

    try {
        conn.setAutoCommit(false);

        // Atualizar na tabela Pessoa
        try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)) {
            stPessoa.setString(1, pf.getNome());
            stPessoa.setString(2, pf.getEndereco());
            stPessoa.setString(3, pf.getCidade());
            stPessoa.setString(4, pf.getEstado());
            stPessoa.setString(5, pf.getTelefone());

```

```

        stPessoa.setString(6, pf.getEmail());
        stPessoa.setInt(7, pf.getId());
        stPessoa.executeUpdate();
    }

    // Atualizar na tabela PessoaFisica
    try (PreparedStatement stPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
        stPessoaFisica.setString(1, pf.getCpf());
        stPessoaFisica.setInt(2, pf.getId());
        stPessoaFisica.executeUpdate();
    }

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

public void excluir(Integer id) throws SQLException {
    String sqlPessoaFisica = "DELETE FROM pessoa_fisica WHERE id_pessoa = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE id_pessoa = ?";

    try {
        conn.setAutoCommit(false);

        // Excluir da tabela PessoaFisica
        try (PreparedStatement stPessoaFisica = conn.prepareStatement(sqlPessoaFisica)) {
            stPessoaFisica.setInt(1, id);
            stPessoaFisica.executeUpdate();
        }

        // Excluir da tabela Pessoa
        try (PreparedStatement stPessoa = conn.prepareStatement(sqlPessoa)) {
            stPessoa.setInt(1, id);
            stPessoa.executeUpdate();
        }

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

```

```

        throw e;
    } finally {
        conn.setAutoCommit(true);
    }
}

```

```

public PessoaFisica getPessoa(Integer id) throws SQLException {
    String sql = "SELECT Pessoa.id_pessoa, Pessoa.nome, Pessoa.endereco,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email, PF.cpf "
        + "FROM Pessoa JOIN pessoa_fisica PF ON Pessoa.id_pessoa = PF.id_pessoa
WHERE Pessoa.id_pessoa = ?";
    try (PreparedStatement st = conn.prepareStatement(sql)) {
        st.setInt(1, id);
        try (ResultSet rs = st.executeQuery()) {
            if (rs.next()) {
                return new PessoaFisica(
                    rs.getInt("id_pessoa"),
                    rs.getString("nome"),
                    rs.getString("endereco"),
                    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> list = new ArrayList<>();
    String sql = "SELECT Pessoa.id_pessoa, Pessoa.nome, Pessoa.endereco,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email, PF.cpf "
        + "FROM Pessoa JOIN pessoa_fisica PF ON Pessoa.id_pessoa =
PF.id_pessoa";
    try (PreparedStatement st = conn.prepareStatement(sql); ResultSet rs =
st.executeQuery()) {
        while (rs.next()) {
            list.add(new PessoaFisica(
                rs.getInt("id_pessoa"),
                rs.getString("nome"),
                rs.getString("endereco"),

```

```

        rs.getString("cidade"),
        rs.getString("estado"),
        rs.getString("telefone"),
        rs.getString("email"),
        rs.getString("cpf")
    ));
    }
}
return list;
}
}

```

3.5

```
package cadastrabd.model;
```

```
/**
```

```
*
```

```
* @author mrjoa
```

```
*/
```

```
public class PessoaJuridica extends Pessoa {
```

```
    private String cnpj;
```

```
    public PessoaJuridica() {}
```

```
    public PessoaJuridica(int id, String nome, String endereco, String cidade, String estado,
String telefone, String email, String cnpj) {
```

```
        super(id, nome, endereco, cidade, estado, telefone, email);
```

```
        this.cnpj = cnpj;
```

```
    }
```

```
    public String getCnpj() {
```

```
        return cnpj;
```

```
    }
```

```
    public void setCnpj(String cnpj) {
```

```
        this.cnpj = cnpj;
```

```
    }
```

```
@Override
```

```
public void exibir() {
```

```
    super.exibir();
```



```
        System.out.println("CNPJ: " + cnpj);
    }
}
```

3.6

```
package cadastrbd.model;

import cadastrbd.model.util.ConectorBD;
import cadastrbd.model.util.SequenceManager;
import java.sql.Connection;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

/**
 *
 * @author mrjoa
 */
public class PessoaJuridicaDAO {

    private Connection conn;

    public PessoaJuridicaDAO(Connection conn) {
        this.conn = conn;
    }

    private PessoaJuridica extrairPessoaJuridica(ResultSet rs) throws SQLException {
        return new PessoaJuridica(
            rs.getInt("id_pessoa"),
            rs.getString("nome"),
            rs.getString("endereco"),
            rs.getString("cidade"),
            rs.getString("estado"),
            rs.getString("telefone"),
            rs.getString("email"),
            rs.getString("cnpj")
        );
    }
}
```

```

public PessoaJuridica getPessoa(int id) throws SQLException {
    final String sql = "SELECT Pessoa.id_pessoa, Pessoa.nome, Pessoa.endereco,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email, PJ.cnpj\n"
        + "FROM Pessoa AS Pessoa JOIN pessoa_juridica AS PJ ON Pessoa.id_pessoa =
PJ.id_pessoa WHERE Pessoa.id_pessoa = ?;";
    try (PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        try (ResultSet rs = stmt.executeQuery()) {
            if (rs.next()) {
                return extrairPessoaJuridica(rs);
            }
        }
    }
    return null;
}

```

```

public List<PessoaJuridica> getPessoasJuridicas() throws SQLException {
    List<PessoaJuridica> list = new ArrayList<>();
    final String sql = "SELECT Pessoa.id_pessoa, Pessoa.nome, Pessoa.endereco,
Pessoa.cidade, Pessoa.estado, Pessoa.telefone, Pessoa.email, PJ.cnpj\n"
        + "FROM Pessoa AS Pessoa JOIN pessoa_juridica AS PJ ON Pessoa.id_pessoa =
PJ.id_pessoa;";
    try (PreparedStatement stmt = conn.prepareStatement(sql); ResultSet rs =
stmt.executeQuery()) {
        while (rs.next()) {
            list.add(extrairPessoaJuridica(rs));
        }
    }
    return list;
}

```

```

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

    try {
        conn.setAutoCommit(false);

        int pessoald = 0;
        try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa,
Statement.RETURN_GENERATED_KEYS)) {
            stmtPessoa.setString(1, pessoa.getNome());
            stmtPessoa.setString(2, pessoa.getEndereco());

```

```

stmtPessoa.setString(3, pessoa.getCidade());
stmtPessoa.setString(4, pessoa.getEstado());
stmtPessoa.setString(5, pessoa.getTelefone());
stmtPessoa.setString(6, pessoa.getEmail());
stmtPessoa.executeUpdate();

```

```

try (ResultSet generatedKeys = stmtPessoa.getGeneratedKeys()) {
    if (generatedKeys.next()) {
        pessoald = generatedKeys.getInt(1);
    }
}
}

```

```

if (pessoald == 0) {
    throw new SQLException("Falha ao inserir pessoa, nenhum ID foi gerado.");
}

```

```

try (PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {
    stmtPessoaJuridica.setInt(1, pessoald);
    stmtPessoaJuridica.setString(2, pessoa.getCnpj());
    stmtPessoaJuridica.executeUpdate();
}

```

```

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

```

```

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

```

```

try {
    conn.setAutoCommit(false);
    try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
        stmtPessoa.setString(1, pessoa.getNome());
        stmtPessoa.setString(2, pessoa.getEndereco());

```

```

        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();
    }
    try (PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {
        stmtPessoaJuridica.setString(1, pessoa.getCnpj());
        stmtPessoaJuridica.setInt(2, pessoa.getId());
        stmtPessoaJuridica.executeUpdate();
    }
    conn.commit();
} catch (SQLException e) {
    conn.rollback();
    throw e;
} finally {
    conn.setAutoCommit(true);
}
}

```

public void excluir(int id) throws SQLException {

```

    String sqlPessoaJuridica = "DELETE FROM pessoa_juridica WHERE id_pessoa = ?";
    String sqlPessoa = "DELETE FROM Pessoa WHERE id_pessoa = ?";

```

```

    try {
        conn.setAutoCommit(false);
        try (PreparedStatement stmtPessoaJuridica =
conn.prepareStatement(sqlPessoaJuridica)) {
            stmtPessoaJuridica.setInt(1, id);
            stmtPessoaJuridica.executeUpdate();
        }
        try (PreparedStatement stmtPessoa = conn.prepareStatement(sqlPessoa)) {
            stmtPessoa.setInt(1, id);
            stmtPessoa.executeUpdate();
        }
        conn.commit();
    } catch (SQLException e) {
        conn.rollback();
        throw e;
    } finally {
        conn.setAutoCommit(true);
    }
}

```

```
}  
}
```

```
}
```

3.7

```
package cadastrobd.model.util;
```

```
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.sql.Statement;
```

```
/**
```

```
 *
```

```
 * @author mrjoa
```

```
 */
```

```
public class ConectorBD {
```

```
    public static void main(String[] args) {  
        getConnection();
```

```
    }
```

```
    private static Connection conn = null;
```

```
    public static Connection getConnection() {
```

```
        if (conn == null) {
```

```
            try {
```

```
                // URL de conexão atualizada para incluir o banco de dados
```

```
                String url = "jdbc:sqlserver://localhost\\DESKTOP-  
3F3K0EB:1433;databaseName=loja;encrypt=true;trustServerCertificate=true;";
```

```
                String user = "loja";
```

```
                String password = "loja";
```

```
                conn = DriverManager.getConnection(url, user, password);
```

```
                System.out.println("Banco de Dados Conectado com Sucesso");
```

```
            } catch (SQLException e) {
```

```
                System.out.println("Erro no Banco de Dados");
```

```
                throw new RuntimeException("Erro ao obter conexao com o banco de dados: " +  
e.getMessage(), e);
```

```
            }
```

```

    }
    return conn;
}

public static void closeStatement(Statement statement) {
    try {
        if (statement != null) {
            statement.close();
        }
    } catch (SQLException e) {
        throw new RuntimeException("Erro ao fechar o statement: " + e.getMessage(), e);
    }
}

public static void closeResultSet(ResultSet resultSet) {
    try {
        if (resultSet != null) {
            resultSet.close();
        }
    } catch (SQLException e) {
        throw new RuntimeException("Erro ao fechar o resultSet: " + e.getMessage(), e);
    }
}

public static void closeConnection() {
    try {
        if (conn != null) {
            conn.close();
        }
    } catch (SQLException e) {
        throw new RuntimeException("Erro ao fechar a conexão com o banco de dados: " +
e.getMessage(), e);
    }
}

void close(ResultSet resultSet) {
    throw new UnsupportedOperationException("Not supported yet."); // Generated from
nbfs://nbhost/SystemFileSystem/Templates/Classes/Code/GeneratedMethodBody
}
}

```

```

package cadastrabd.model.util;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

/**
 *
 * @author mrjoa
 */
public class SequenceManager {

    private final ConectorBD conectorBD;

    public SequenceManager() {
        this.conectorBD = new ConectorBD();
    }

    public int getNextValue(String sequenceName) throws SQLException {
        int proximoValor = 0;
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet resultSet = null;

        try {
            connection = conectorBD.getConnection();
            String sql = "SELECT NEXT VALUE FOR " + sequenceName + " AS NextVal";
            preparedStatement = connection.prepareStatement(sql);
            resultSet = preparedStatement.executeQuery();

            if (resultSet.next()) {
                proximoValor = resultSet.getInt("NextVal");
            }
        } finally {
            conectorBD.closeResultSet(resultSet);
            conectorBD.closeStatement(preparedStatement);
            conectorBD.closeConnection();
        }

        return proximoValor;
    }
}

```

4º Os resultados da execução dos códigos:

4.1 Incluir Pessoa Fisica:

```
Output - CadastroBD (run) #4 x
Banco de Dados Conectado com Sucesso
#####
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
#####
Escolha uma opcao:
1
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
f
Digite o nome para a pessoa fisica:
Jairo
Digite o endereco:
taguai
Digite a cidade:
Recife
Digite o estado:
PE
Digite o telefone:
33043304
Digite o e-mail:
mr.jojo@gmail.com
Digite o CPF:
52080595
## > Pessoa Fisica cadastrada com sucesso!
```

Resultado:

CadastroBD.java x SQL 1 [jdbc:sqlserver:// localho...] x

Connection: jdbc:sqlserver:// localhost\Gilberto\SQLEXPRESS:1433;databaseNa...

```
1 SELECT TOP 100 * FROM dbo.pessoa;
```

SELECT TOP 100 * FROM dbo... x

Max. rows: 100 | Fetched Rows: 4 | Matching Rows:

#	id_pessoa	nome	endereco	cidade	estado	telefone	email
1	1	jo	tuguai	hellcife	PE	33093309	mr.jo
2	3	joao	casa	recife	PE	44440000	mr.jo
3	4	jorge					popo
4	6	Jairo	taguai	Recife	PE	33043304	mr.jojo@gmail.com

4.1 Incluir Pessoa Juridica:

CadastroBD.java x

Source History

Output - CadastroBD (run) x

```
1
(F) - Pessoa Fisica | (J) - Pessoa juridica
Escolha uma opcao:
j
Digite o nome para a pessoa juridica:
Mercadinho Lambelambe
Digite o endereco:
rua to perdido
Digite a cidade:
Tilambuco
Digite o estado:
PE
Digite o telefone:
21212121
Digite o e-mail:
merc@mercado.com.br
Digite o CNPJ:
12345678901234
## > Pessoa Juridica cadastrada com sucesso.
#####
```

Resultado:

CadastroBD.java x SQL 1 [jdbc:sqlserver:// localho...] x SQL 2 [jdbc:sqlserver:// localho...]

Connection: jdbc:sqlserver:// localhost\Gilberto\SQLEXPRESS:1433;databaseNam...

1 SELECT TOP 100 * FROM dbo.pessoa;

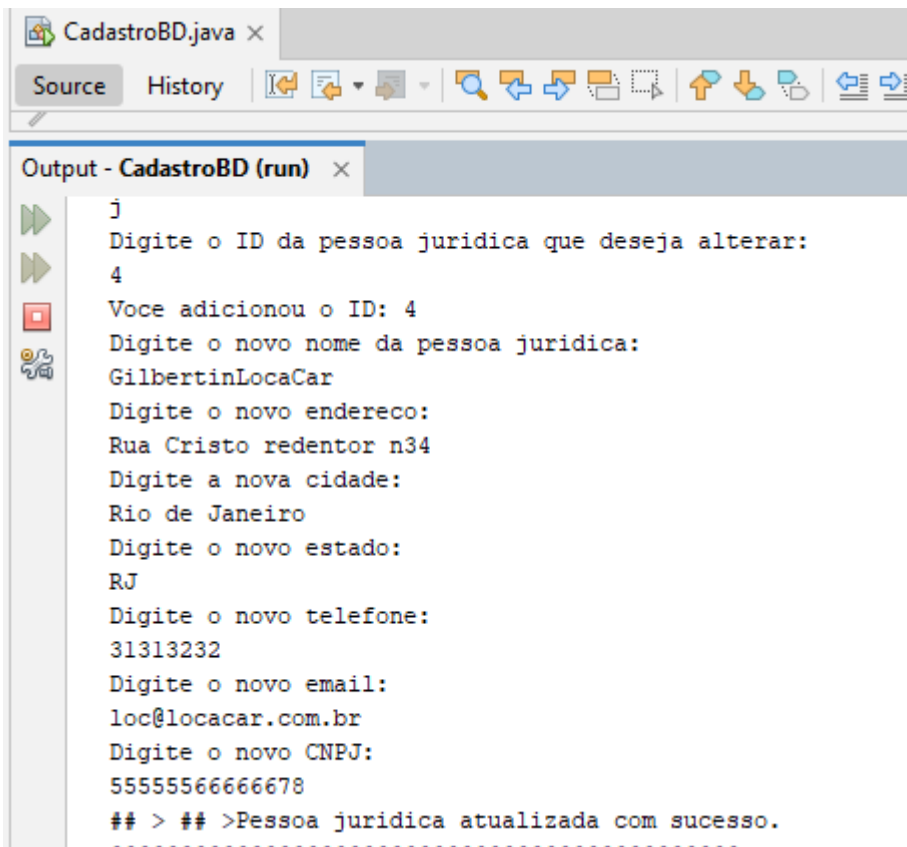
2

SELECT TOP 100 * FROM dbo...

Max. rows: 100 | Fetched Rows: 5 | Matching Rows:

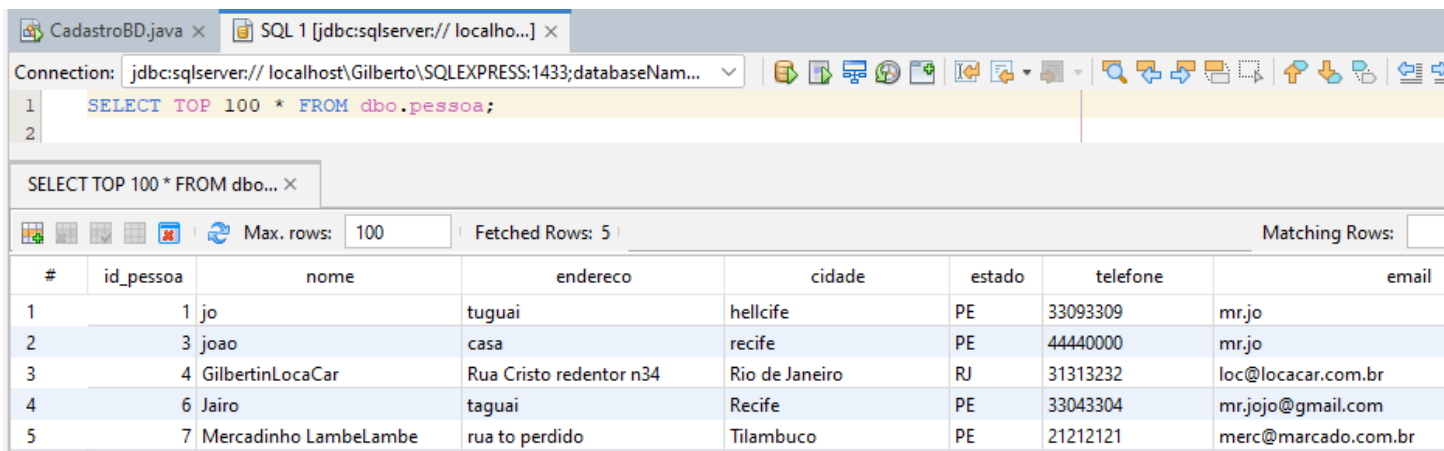
#	id_pessoa	nome	endereco	cidade	estado	telefone	email
1	1	jo	tuguai	hellcife	PE	33093309	mr.jo
2	3	joao	casa	recife	PE	44440000	mr.jo
3	4	jorge					popo
4	6	Jairo	taguai	Recife	PE	33043304	mr.jojo@gmail.com
5	7	Mercadinho Lambelambe	rua to perdido	Tilambuco	PE	21212121	merc@mercado.com.br

4.2 Alterar Pessoa



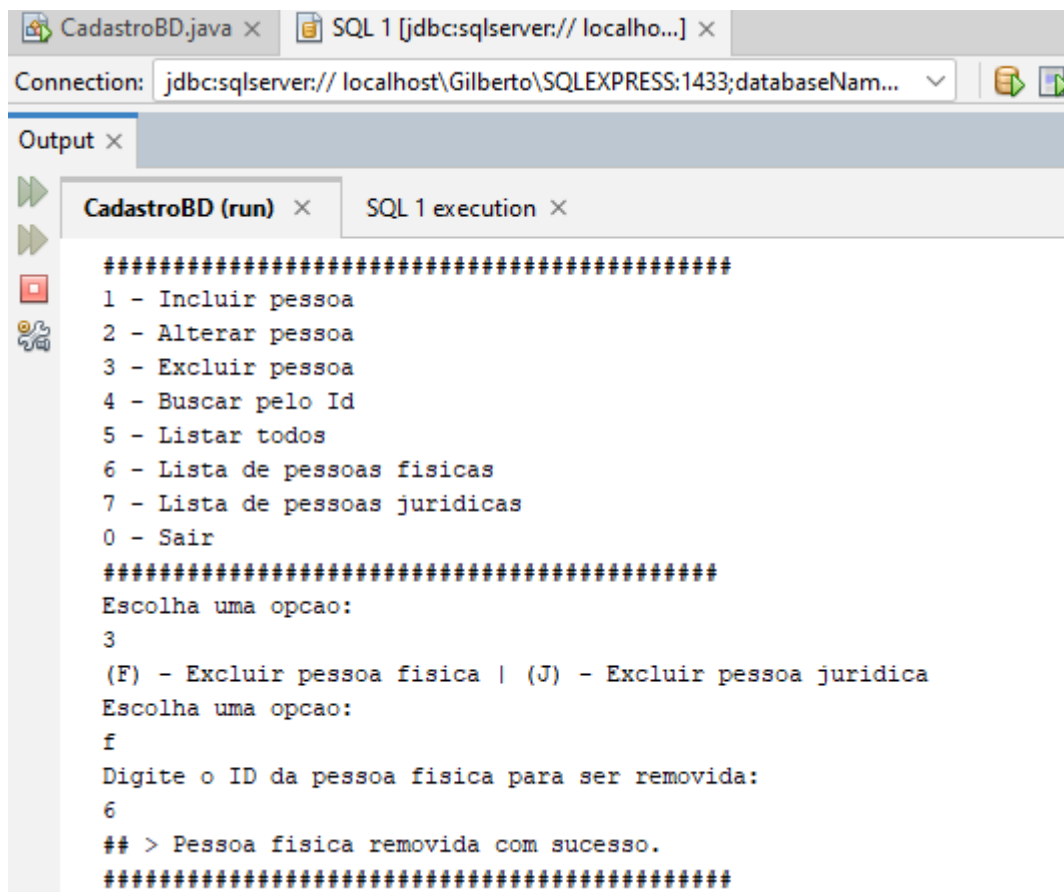
```
j
Digite o ID da pessoa juridica que deseja alterar:
4
Voce adicionou o ID: 4
Digite o novo nome da pessoa juridica:
GilbertinLocaCar
Digite o novo endereco:
Rua Cristo redentor n34
Digite a nova cidade:
Rio de Janeiro
Digite o novo estado:
RJ
Digite o novo telefone:
31313232
Digite o novo email:
loc@locacar.com.br
Digite o novo CNPJ:
55555566666678
## > ## >Pessoa juridica atualizada com sucesso.
.....
```

Resultado: O ID: 4 foi alterado com o novo nome GilbertinLocaCar e os demais dados.



#	id_pessoa	nome	endereco	cidade	estado	telefone	email
1	1	jo	tuguai	hellcife	PE	33093309	mr.jo
2	3	joao	casa	recife	PE	44440000	mr.jo
3	4	GilbertinLocaCar	Rua Cristo redentor n34	Rio de Janeiro	RJ	31313232	loc@locacar.com.br
4	6	Jairo	taguai	Recife	PE	33043304	mr.jojo@gmail.com
5	7	Mercadinho LambeLambe	rua to perdido	Tilambuco	PE	21212121	merc@mercado.com.br

4.3 Excluir Pessoa



The screenshot shows an IDE with two tabs: 'CadastroBD.java' and 'SQL 1 [jdbc:sqlserver:// localhost...]'.

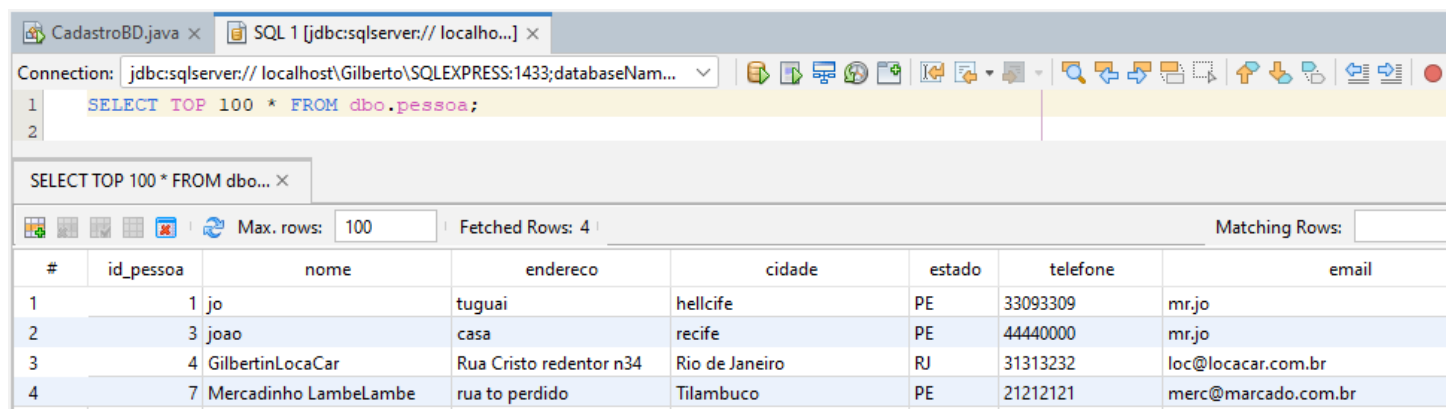
The 'CadastroBD (run)' tab displays the following output:

```
#####
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
#####
Escolha uma opcao:
3
(F) - Excluir pessoa fisica | (J) - Excluir pessoa juridica
Escolha uma opcao:
f
Digite o ID da pessoa fisica para ser removida:
6
## > Pessoa fisica removida com sucesso.
#####
```

The 'SQL 1 execution' tab shows the following SQL query:

```
SELECT TOP 100 * FROM dbo.pessoa;
```

Resultado: Foi excluido o ID 6 .



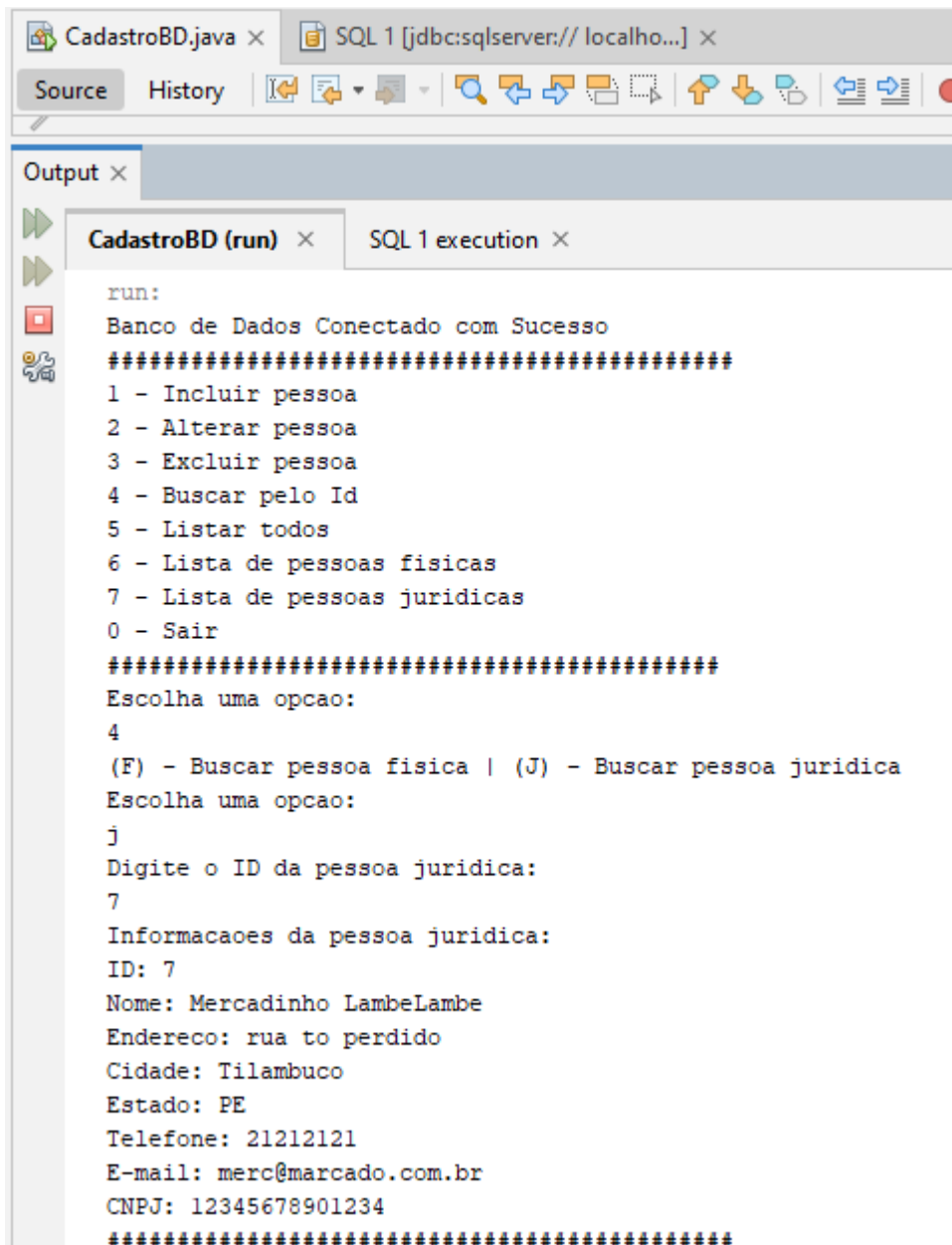
The screenshot shows the 'SQL 1 [jdbc:sqlserver:// localhost...]' tab with the following SQL query:

```
SELECT TOP 100 * FROM dbo.pessoa;
```

The results are displayed in a table with 8 columns: #, id_pessoa, nome, endereco, cidade, estado, telefone, and email. The table contains 4 rows of data.

#	id_pessoa	nome	endereco	cidade	estado	telefone	email
1	1	jo	tuguai	hellcife	PE	33093309	mr.jo
2	3	joao	casa	recife	PE	44440000	mr.jo
3	4	GilbertinLocaCar	Rua Cristo redentor n34	Rio de Janeiro	RJ	31313232	loc@locacar.com.br
4	7	Mercadinho LambelLambe	rua to perdido	Tilambuco	PE	21212121	merc@mercado.com.br

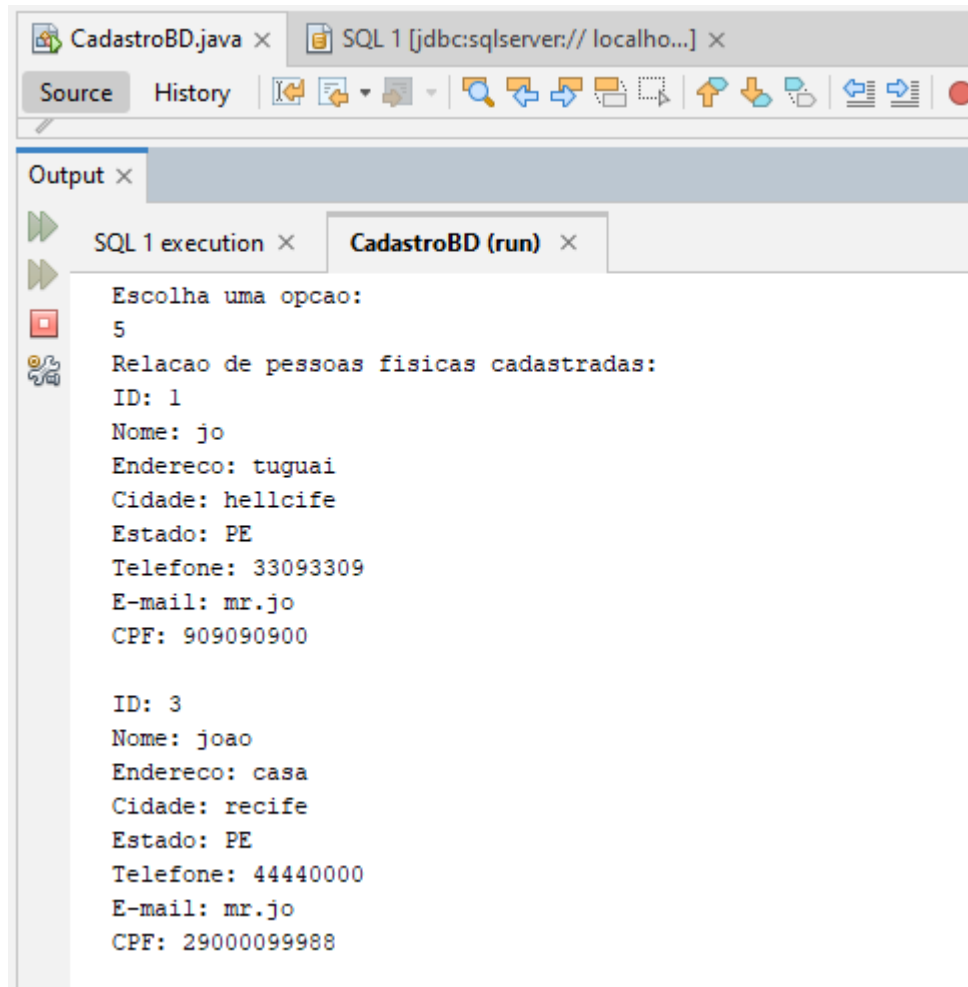
4.4 Buscar ID



```
run:
Banco de Dados Conectado com Sucesso
#####
1 - Incluir pessoa
2 - Alterar pessoa
3 - Excluir pessoa
4 - Buscar pelo Id
5 - Listar todos
6 - Lista de pessoas fisicas
7 - Lista de pessoas juridicas
0 - Sair
#####
Escolha uma opcao:
4
(F) - Buscar pessoa fisica | (J) - Buscar pessoa juridica
Escolha uma opcao:
j
Digite o ID da pessoa juridica:
7
Informacoes da pessoa juridica:
ID: 7
Nome: Mercadinho LambeLambe
Endereco: rua to perdido
Cidade: Tilambuco
Estado: PE
Telefone: 21212121
E-mail: merc@mercado.com.br
CNPJ: 12345678901234
#####
```

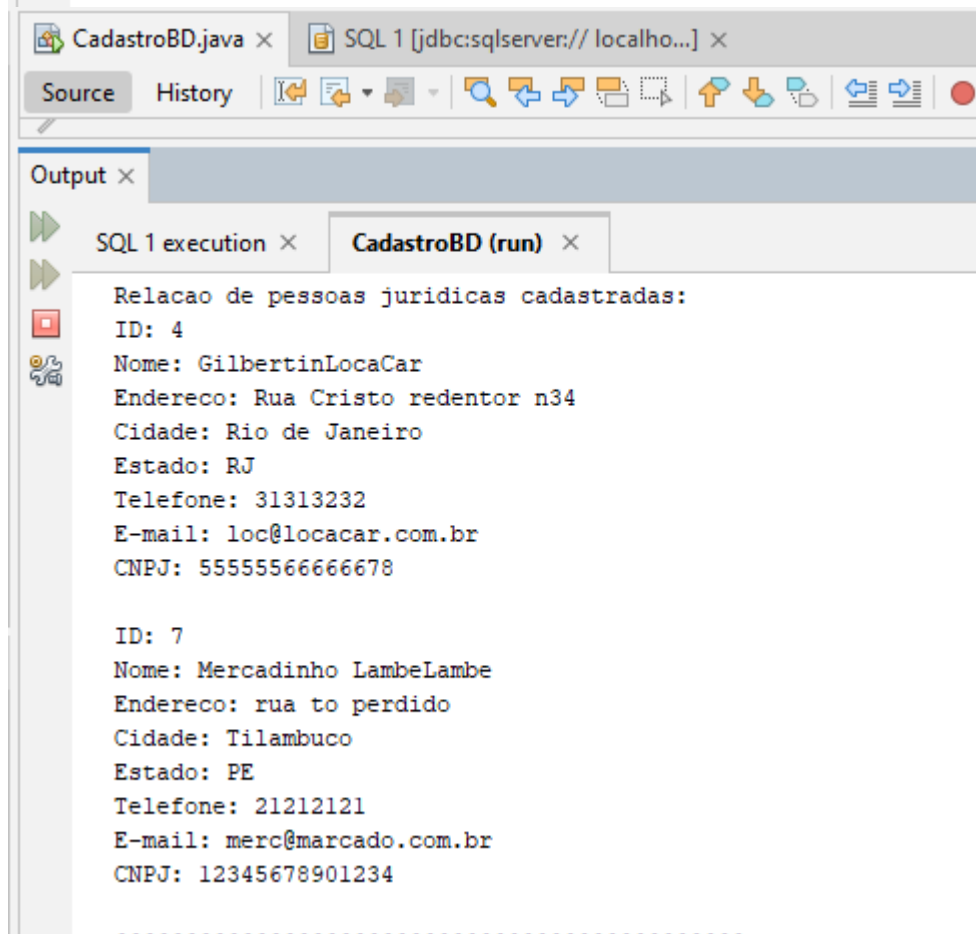
Resultado: Neste caso, foi mostrado em tela a busca pelo ID 7 e o resultado sao os dados que foi solicitado.

4.5 Listar Todos



```
CadastroBD.java x SQL 1 [jdbc:sqlserver:// localho...] x
Source History
Output x
SQL 1 execution x CadastroBD (run) x
Escolha uma opcao:
5
Relacao de pessoas fisicas cadastradas:
ID: 1
Nome: jo
Endereco: tuguai
Cidade: hellcife
Estado: PE
Telefone: 33093309
E-mail: mr.jo
CPF: 909090900

ID: 3
Nome: joao
Endereco: casa
Cidade: recife
Estado: PE
Telefone: 44440000
E-mail: mr.jo
CPF: 29000099988
```



```
CadastroBD.java x SQL 1 [jdbc:sqlserver:// localho...] x
Source History
Output x
SQL 1 execution x CadastroBD (run) x
Relacao de pessoas juridicas cadastradas:
ID: 4
Nome: GilbertinLocaCar
Endereco: Rua Cristo redentor n34
Cidade: Rio de Janeiro
Estado: RJ
Telefone: 31313232
E-mail: loc@locacar.com.br
CNPJ: 55555566666678

ID: 7
Nome: Mercadinho Lambelambe
Endereco: rua to perdido
Cidade: Tilambuco
Estado: PE
Telefone: 21212121
E-mail: merc@mercado.com.br
CNPJ: 12345678901234

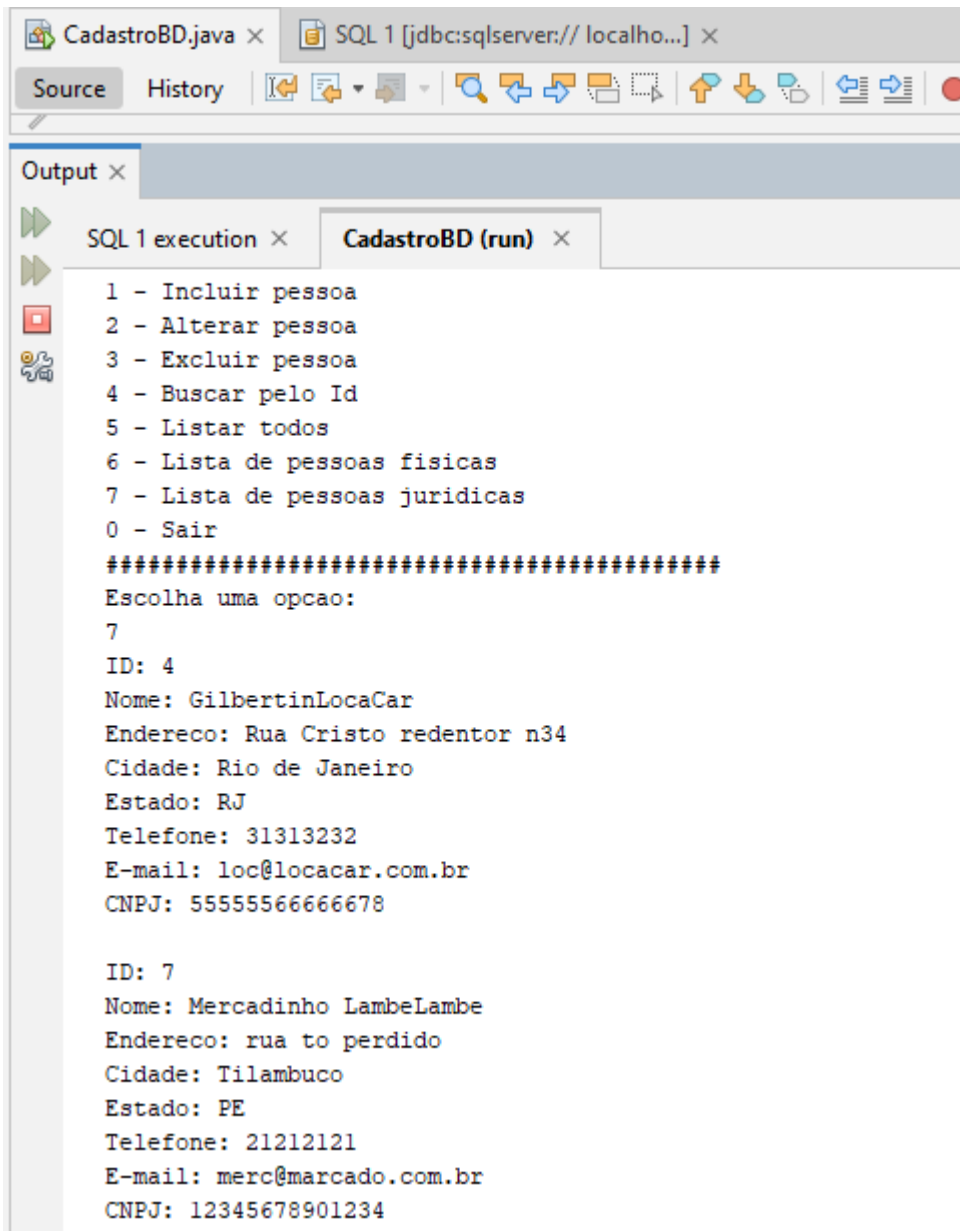
*****
```

4.6 Listar Pessoas Fisicas

```
Relacao de pessoas juridicas cadastradas:  
ID: 4  
Nome: GilbertinLocaCar  
Endereco: Rua Cristo redentor n34  
Cidade: Rio de Janeiro  
Estado: RJ  
Telefone: 31313232  
E-mail: loc@locacar.com.br  
CNPJ: 55555566666678  
  
ID: 7  
Nome: Mercadinho Lambelambe  
Endereco: rua to perdido  
Cidade: Tilambuco  
Estado: PE  
Telefone: 21212121  
E-mail: merc@mercado.com.br  
CNPJ: 12345678901234  
  
*****
```

Resultado: Neste caso foi listados todas Pessoas Fisicas cadastradas

4.7 Listar Pessoas Juridicas



```
CadastroBD.java x SQL 1 [jdbc:sqlserver:// localho...]  
Source History  
Output x  
SQL 1 execution x CadastroBD (run) x  
1 - Incluir pessoa  
2 - Alterar pessoa  
3 - Excluir pessoa  
4 - Buscar pelo Id  
5 - Listar todos  
6 - Lista de pessoas fisicas  
7 - Lista de pessoas juridicas  
0 - Sair  
#####  
Escolha uma opcao:  
7  
ID: 4  
Nome: GilbertinLocaCar  
Endereco: Rua Cristo redentor n34  
Cidade: Rio de Janeiro  
Estado: RJ  
Telefone: 31313232  
E-mail: loc@locacar.com.br  
CNPJ: 55555566666678  
  
ID: 7  
Nome: Mercadinho LambeLambe  
Endereco: rua to perdido  
Cidade: Tilambuco  
Estado: PE  
Telefone: 21212121  
E-mail: merc@mercado.com.br  
CNPJ: 12345678901234
```

Resultado: Neste caso foi listados todas Pessoas Juridicas cadastrada

5º Análise e Conclusão:

Aº Quais as diferenças entre a persistência em arquivo e a persistência em bancos de dados?

R: A decisão de como persistir os dados de uma aplicação, seja em arquivos ou em um banco de dados, é crucial e pode influenciar significativamente o desempenho, a escalabilidade e a manutenibilidade do sistema.

persistência em arquivo:

Conceito: Nessa abordagem, os dados são armazenados em arquivos no sistema de arquivos, podendo ser em formatos simples como texto ou em formatos mais complexos como JSON ou XML.

persistência em banco de dados:

Conceito: Os dados são armazenados em um sistema gerenciador de banco de dados (SGBD), que oferece ferramentas para organizar, armazenar e recuperar dados de forma eficiente.

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

R: Os operadores lambda, introduzidos no Java 8, trouxeram uma nova forma de escrever código mais conciso e expressivo, especialmente quando se trata de lidar com coleções e expressões funcionais. No contexto da impressão de valores contidos em entidades, os lambdas oferecem uma sintaxe mais elegante e simplificada em comparação com as abordagens tradicionais.

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

R: • **Pertence à classe, não ao objeto:** Um método estático está associado à classe em si, e não a uma instância específica daquela classe. Isso significa que ele pode ser chamado diretamente pelo nome da classe, sem a necessidade de criar um objeto.

• **Acesso a membros estáticos:** Métodos estáticos só podem acessar outros membros estáticos da classe, como variáveis estáticas e outros métodos estáticos. Eles não podem acessar diretamente membros não estáticos (variáveis de instância e métodos de instância) porque estes estão associados a objetos específicos.

Ponto de entrada: O método `main` é o ponto de partida da execução de um programa Java. É chamado diretamente pela máquina virtual Java (JVM) quando um programa é executado.

Método estático: O `main` é sempre declarado como `static` por dois motivos principais:

1. **Acessibilidade:** A JVM precisa chamar o `main` sem criar uma instância da classe, já que ainda não existe nenhum objeto no início da execução.
2. **Convenção:** É uma convenção estabelecida em Java que o método `main` seja sempre estático.

