Nome: Wallace Felipe Tavares Moreira

Matrícula: 202109237331

Universidade: UNIVERSIDADE ESTÁCIO DE SÁ

Curso: Desenvolvimento Full Stack

Campus: Jardim América – Itaguaí/RJ

Disciplina: Nível 3 – BeckEnd sem banco não tem

Semestre Letivo: Terceiro Semestre

Objetivo da Prática

1 - Implementar persistência com base no middleware JDBC.

2 - Utilizar o padrão DAO (Data Access Object) no manuseio de dados.

3 - Implementar o mapeamento objeto-relacional em sistemas Java.

4 - Criar sistemas cadastrais com persistência em banco relacional.

5 - criar um aplicativo cadastral com uso do SQL Server na persistência de dados.

Códigos Solicitados:

3.1. Classe ConectorBD:

A classe ConectorBD é responsável por estabelecer a conexão com o banco de dados. Ela contém métodos para abrir e fechar a conexão, bem como para obter um objeto Statement que será usado para executar consultas SQL no banco de dados.

3.2. Classe SequenceManager:

A classe SequenceManager é responsável por gerenciar sequências numéricas no banco de dados. Ela utiliza a função nextval do SQL para obter o próximo valor de uma sequência especificada no banco de dados.

3.3. PessoaFisica e PessoaJuridica:

As classes PessoaFisica e PessoaJuridica são classes que representam entidades no sistema, ou seja, as informações de uma pessoa física e pessoa jurídica, respectivamente. Cada uma dessas classes possui atributos que correspondem aos campos do banco de dados, como nome, CPF, CNPJ, etc. Essas classes também possuem métodos getter e setter para acessar e modificar esses atributos.

3.4. Classe PessoaFisicaDAO:

A classe PessoaFisicaDAO é uma classe que implementa a lógica de acesso aos dados para a entidade PessoaFisica. Ela possui métodos para incluir, alterar, excluir e buscar informações de pessoas físicas no banco de dados. A classe utiliza a classe ConectorBD para estabelecer a conexão com o banco de dados e realizar as operações SQL.

3.5. Classe PessoaJuridicaDAO:

A classe PessoaJuridicaDAO é semelhante à classe PessoaFisicaDAO, mas é específica para a entidade PessoaJuridica. Ela também possui métodos para incluir, alterar, excluir e buscar informações de pessoas jurídicas no banco de dados, utilizando a classe ConectorBD para a conexão e operações SQL.

3.6. Classe InterfaceCadastro:

A classe InterfaceCadastro contém métodos para interagir com o usuário através da linha de comando e realizar as operações de inclusão, alteração, exclusão, obtenção por ID e listagem de pessoas físicas e jurídicas. Ela utiliza as classes PessoaFisicaDAO e PessoaJuridicaDAO para realizar essas operações no banco de dados.

3.7. Classe CadastroBD:

A classe CadastroBD é a classe principal do aplicativo. Ela contém o método main, que é o ponto de entrada do programa. Essa classe é responsável por criar instâncias dos DAOs e realizar a interação com o usuário através da classe InterfaceCadastro.

Esses códigos trabalham em conjunto para criar um aplicativo de cadastro que permite ao usuário interagir com o banco de dados, realizando operações de inclusão, alteração, exclusão, obtenção e listagem de pessoas físicas e jurídicas. O uso do padrão DAO ajuda a manter a separação de responsabilidades e a melhorar a manutenibilidade do código, enquanto o JDBC facilita a conexão com o banco de dados e a execução de consultas SQL. A interface InterfaceCadastro fornece uma experiência amigável ao usuário para realizar essas operações de forma intuitiva e eficiente.

Análise e Conclusão:

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

Os componentes de middleware, como o JDBC (Java Database Connectivity), desempenham um papel crucial na comunicação entre aplicativos Java e bancos de dados. Eles agem como uma camada de tradução que permite que o código Java interaja com diferentes sistemas de gerenciamento de banco de dados (DBMS) de forma independente. O JDBC facilita a conexão com o banco de dados, o envio de consultas SQL, o processamento de resultados e o controle de transações. Isso torna o desenvolvimento de aplicativos Java mais eficiente e portátil, pois não é necessário escrever código específico para cada DBMS.

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

O Statement e o PreparedStatement são interfaces do JDBC usadas para executar consultas SQL em um banco de dados. A principal diferença entre eles é a forma como tratam os parâmetros de consulta:

Statement: É usado para executar consultas estáticas sem parâmetros. As consultas SQL são fornecidas diretamente no código Java, e quaisquer valores inseridos na consulta precisam ser concatenados como strings, tornando o código suscetível a ataques de injeção de SQL.

PreparedStatement: É usado para consultas parametrizadas, onde os valores dos parâmetros são definidos usando marcadores de posição (por exemplo, "?") na consulta. Isso evita a concatenação direta de valores e torna o código mais seguro contra ataques de injeção de SQL, pois os valores são tratados como parâmetros separados.

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

O padrão DAO (Data Access Object) é um padrão de projeto que separa a lógica de negócios da lógica de acesso aos dados em um aplicativo. Ao utilizar o padrão DAO, as operações de acesso ao banco de dados são encapsuladas em classes DAO específicas para cada entidade, como PessoaFisicaDAO e PessoaJuridicaDAO. Isso melhora a manutenibilidade do software de várias maneiras:

Separação de Responsabilidades: O padrão DAO permite separar claramente as operações do banco de dados da lógica de negócios, facilitando a compreensão e a manutenção do código.

Flexibilidade: Com o DAO, é mais fácil modificar ou trocar o sistema de gerenciamento de banco de dados subjacente sem afetar outras partes do código do aplicativo. Isso torna o software mais flexível e adaptável a mudanças futuras.

Reutilização de Código: As operações de acesso ao banco de dados são encapsuladas em métodos dentro das classes DAO, permitindo sua reutilização em diferentes partes do aplicativo, evitando a duplicação de código.

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

Em um modelo estritamente relacional, a herança é geralmente implementada usando uma estratégia conhecida como "herança por tabelas" ou "tabelas de junção". Nessa abordagem, cada classe filha herda os atributos e comportamentos da classe pai, e essas classes são mapeadas para tabelas separadas no banco de dados.

Por exemplo, temos as classes PessoaFisica e PessoaJuridica, que herdam da classe Pessoa. No banco de dados, elas seriam mapeadas para tabelas diferentes, com cada tabela contendo os atributos específicos da classe, além dos atributos herdados da classe Pessoa.

Essa abordagem garante que cada tabela no banco de dados represente apenas uma entidade específica, mantendo a integridade do modelo relacional.

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

As diferenças entre a persistência em arquivo e a persistência em banco de dados são:

Estrutura de Dados: Na persistência em arquivo, os dados são armazenados em formatos específicos (como texto ou binário) em arquivos no sistema de arquivos do sistema operacional. Em contraste, a persistência em banco de dados organiza os dados em tabelas relacionais com colunas e linhas, seguindo um esquema definido.

Recursos: O acesso a dados em arquivo requer operações de leitura e gravação de arquivos no sistema de arquivos, o que pode ser mais lento para grandes volumes de dados. Os bancos de dados oferecem mecanismos otimizados para consultas e manipulação de dados, tornando o acesso mais eficiente.

Recuperação e Segurança: Os bancos de dados oferecem recursos de backup e recuperação, garantindo a segurança dos dados em caso de falhas. Além disso, os bancos de dados geralmente têm recursos de controle de acesso para proteger os dados contra acesso não autorizado. Em comparação, os arquivos podem ser mais suscetíveis a perda de dados e têm menos recursos de segurança.

Gerenciamento de Concorrência: Os bancos de dados são projetados para lidar com múltiplas conexões e garantir consistência e integridade dos dados em ambientes concorrentes. O acesso a arquivos pode ser mais suscetível a problemas de concorrência.

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

Nas versões mais recentes do Java (a partir do Java 8), o uso de operadores lambda introduziu uma forma mais concisa e expressiva de lidar com iterações em coleções de dados. No código o operador lambda é utilizado para simplificar a impressão dos valores contidos nas entidades PessoaFisica e PessoaJuridica nas operações de listagem. Em vez de criar um laço explícito e escrever um código de impressão personalizado, o operador lambda permite a impressão dos valores de forma mais concisa e legível, reduzindo a quantidade de código necessário.

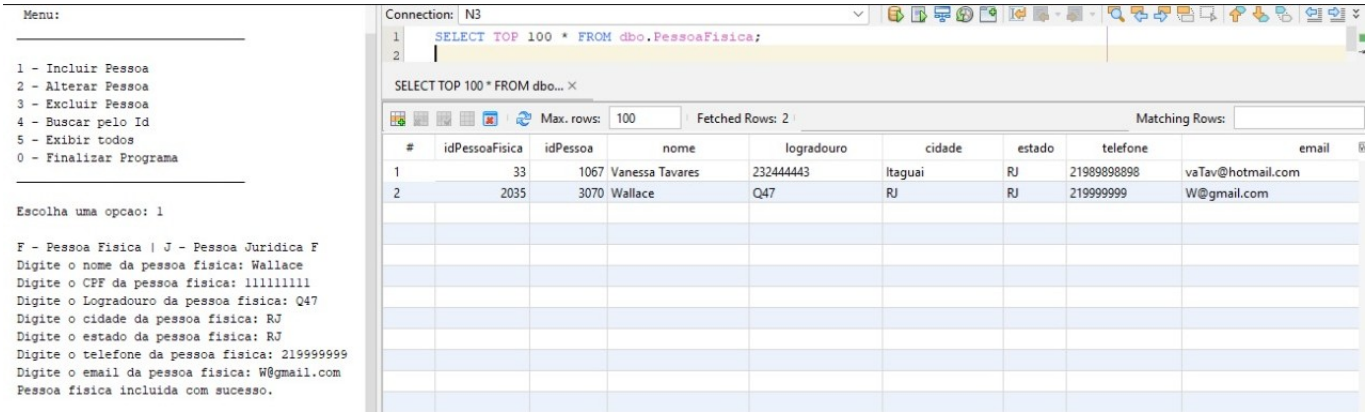
G. Por que métodos acionados diretamente pelo método main, sem o uso de um objeto, precisam ser marcados como static?

Os métodos acionados diretamente pelo método main (o ponto de entrada do programa) precisam ser marcados como static porque o método main também é static. O método main é o ponto de partida para a execução do programa Java, e ele pertence à classe que contém o método main, não a uma instância específica dessa classe.

Quando o programa é iniciado, a JVM (Java Virtual Machine) chama o método main diretamente, sem criar um objeto da classe que o contém. Portanto, para que o método main possa chamar outros métodos dentro da mesma classe (que também são acionados diretamente pelo main), esses métodos também precisam ser static, pois eles pertencem à classe, não a um objeto.

Ao marcar um método como static, ele se torna um método de classe, o que significa que ele pode ser chamado diretamente usando o nome da classe, sem a necessidade de criar uma instância dessa classe. Isso permite que o método main e outros métodos estáticos sejam chamados sem a necessidade de criar um objeto da classe, o que é apropriado para o ponto de entrada do programa.

Incluir Pessoa Fisica



```
Menu:
_____________________

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

Escolha uma opcao: 1

F - Pessoa Fisica | J - Pessoa Juridica F
Digite o nome da pessoa fisica: Wallace
Digite o CPF da pessoa fisica: 111111111
Digite o Logradouro da pessoa fisica: Q47
Digite o cidade da pessoa fisica: RJ
Digite o estado da pessoa fisica: RJ
Digite o telefone da pessoa fisica: 219999999
Digite o email da pessoa fisica: W@gmail.com
Pessoa fisica incluida com sucesso.
```

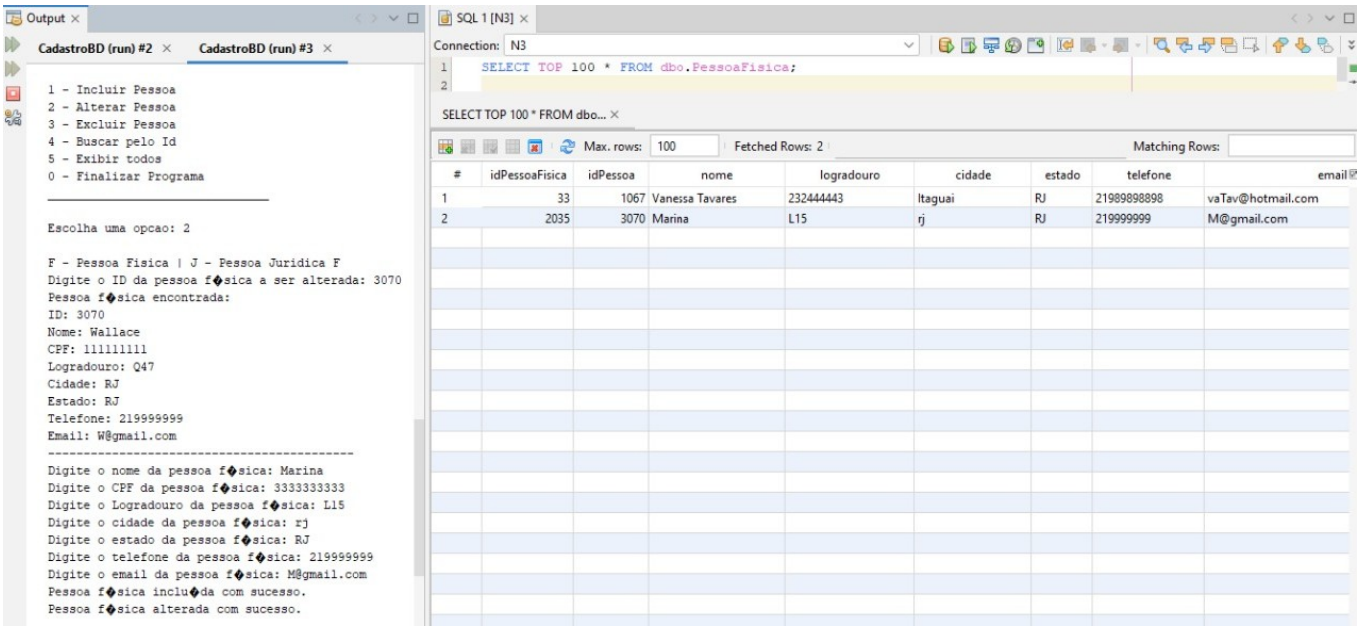
```
Connection: N3
1  SELECT TOP 100 * FROM dbo.PessoaFisica;
2
```

SELECT TOP 100 * FROM dbo... ×

Max. rows: 100 | Fetched Rows: 2 | Matching Rows:

#	idPessoaFisica	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	33	1067	Vanessa Tavares	232444443	Itaguai	RJ	21989898898	vaTav@hotmail.com
2	2035	3070	Wallace	Q47	RJ	RJ	219999999	W@gmail.com

Alterar Pessoa Fisica



Output ×

CadastroBD (run) #2 × CadastroBD (run) #3 ×

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

Escolha uma opcao: 2

F - Pessoa Fisica | J - Pessoa Juridica F
Digite o ID da pessoa f◆sica a ser alterada: 3070
Pessoa f◆sica encontrada:
ID: 3070
Nome: Wallace
CPF: 111111111
Logradouro: Q47
Cidade: RJ
Estado: RJ
Telefone: 219999999
Email: W@gmail.com
-------------------------------------------
Digite o nome da pessoa f◆sica: Marina
Digite o CPF da pessoa f◆sica: 3333333333
Digite o Logradouro da pessoa f◆sica: L15
Digite o cidade da pessoa f◆sica: rj
Digite o estado da pessoa f◆sica: RJ
Digite o telefone da pessoa f◆sica: 219999999
Digite o email da pessoa f◆sica: M@gmail.com
Pessoa f◆sica inclu◆da com sucesso.
Pessoa f◆sica alterada com sucesso.
```

SQL 1 [N3] ×

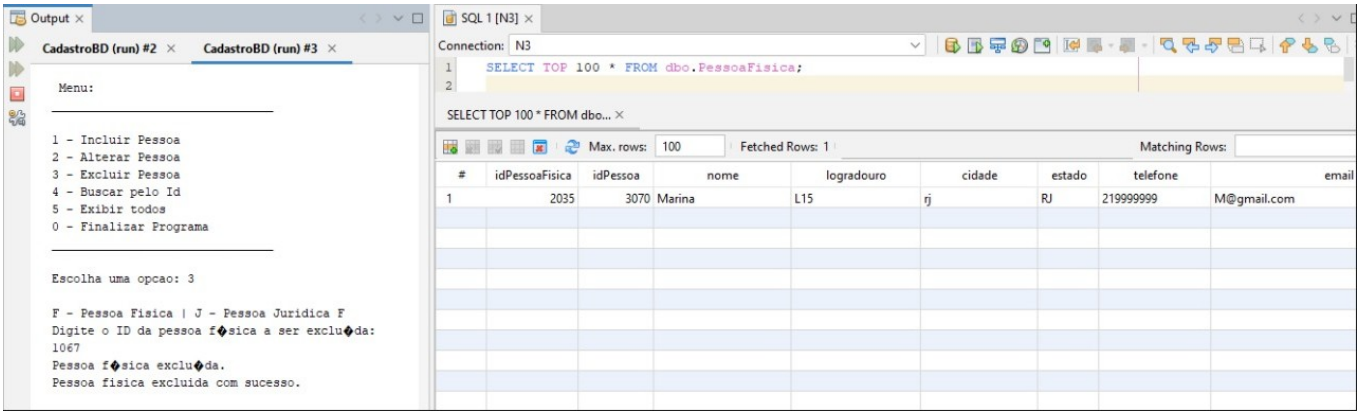
```
Connection: N3
1  SELECT TOP 100 * FROM dbo.PessoaFisica;
2
```

SELECT TOP 100 * FROM dbo... ×

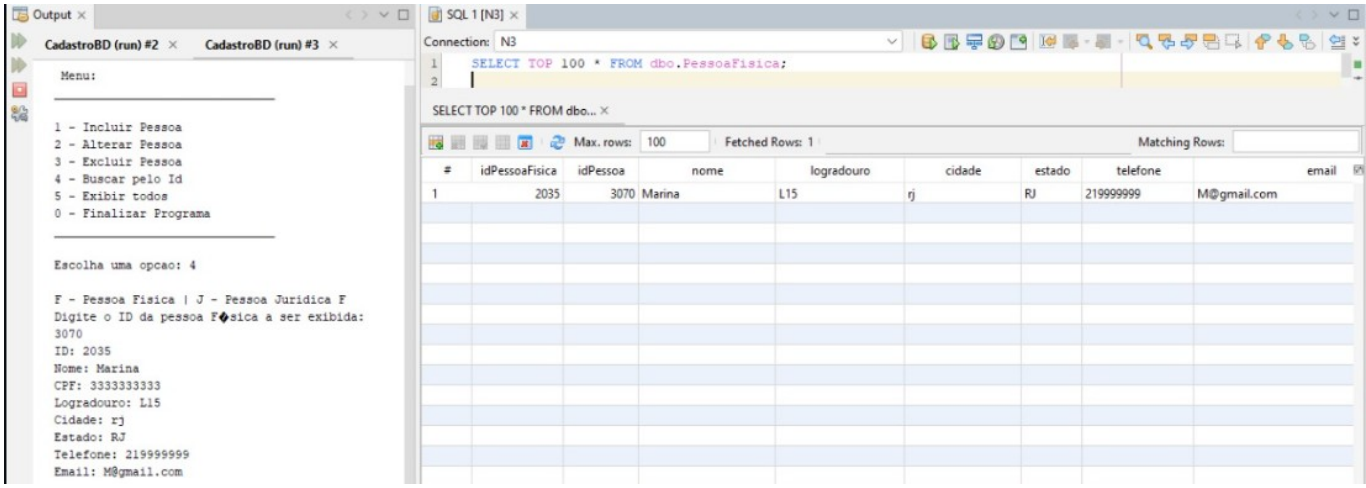
Max. rows: 100 | Fetched Rows: 2 | Matching Rows:

#	idPessoaFisica	idPessoa	nome	logradouro	cidade	estado	telefone	email
1	33	1067	Vanessa Tavares	232444443	Itaguai	RJ	21989898898	vaTav@hotmail.com
2	2035	3070	Marina	L15	rj	RJ	219999999	M@gmail.com

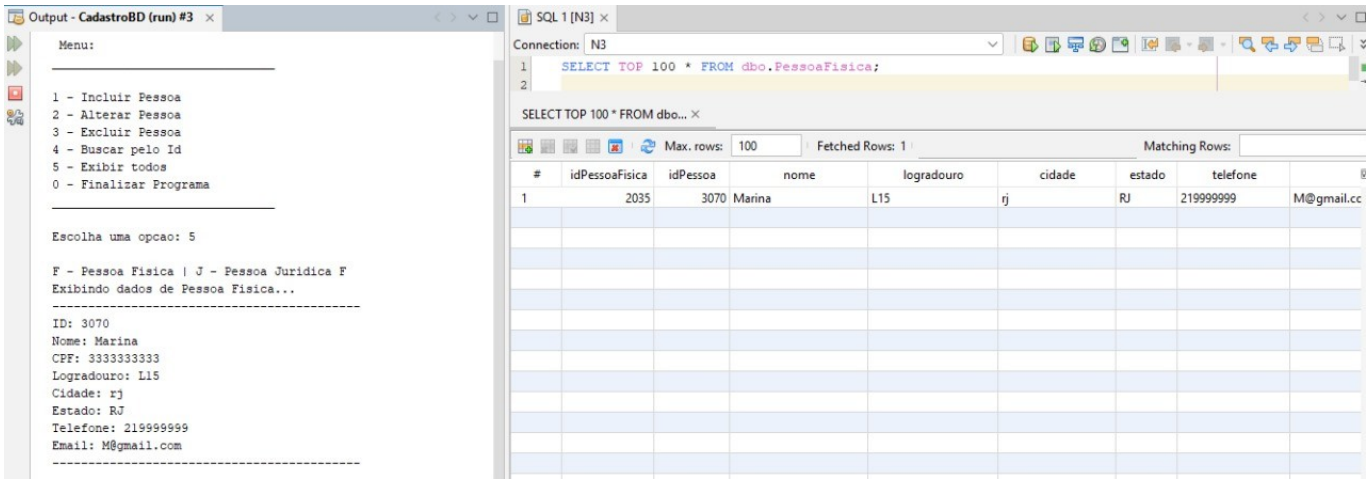
Excluir Pessoa Fisica



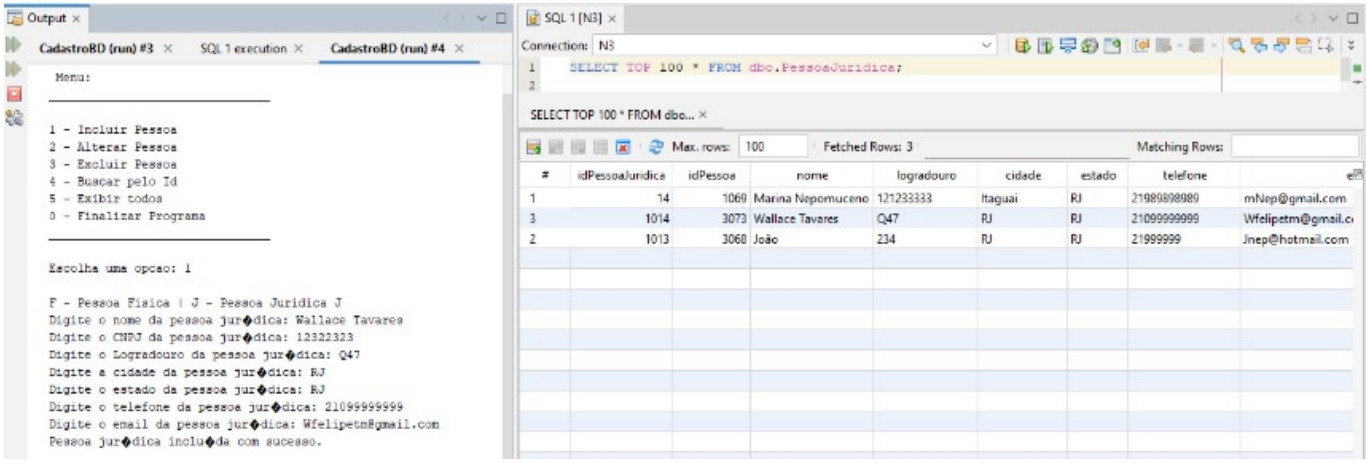
Buscar Pessoa Fisic por ID:



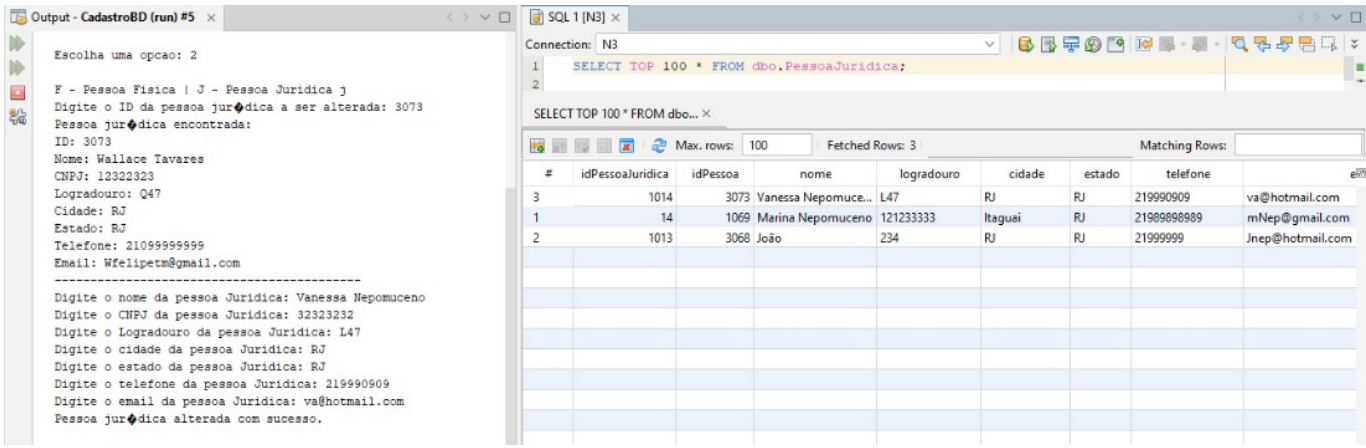
Exibir todas Pessoa Fisica



Incluir Pessoa Juridica



Alterar Pessoa Juridica

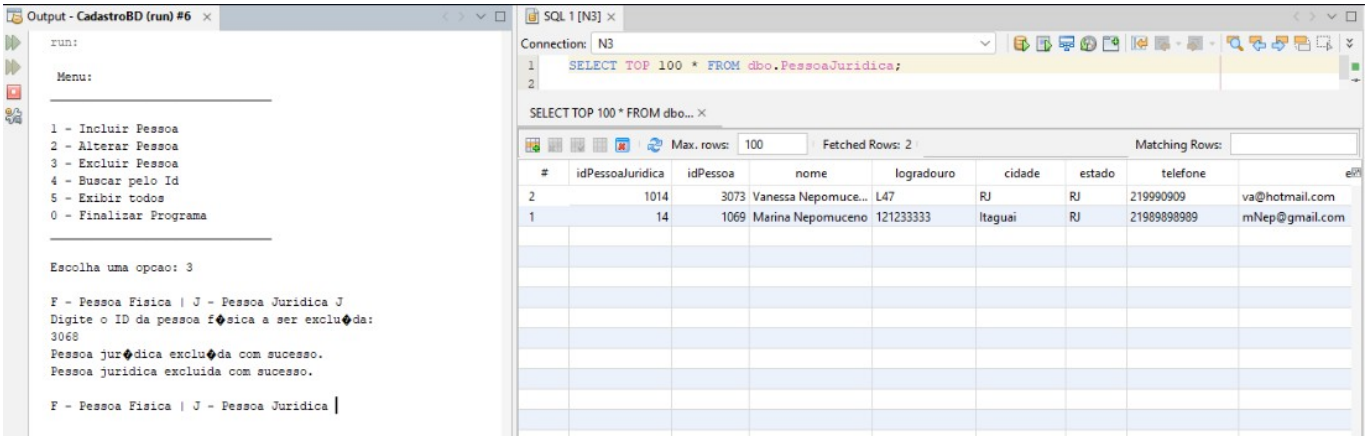


```
Output - CadastroBD (run) #5

Escolha uma opcao: 2

F - Pessoa Fisica | J - Pessoa Juridica j
Digite o ID da pessoa jur◆dica a ser alterada: 3073
Pessoa jur◆dica encontrada:
ID: 3073
Nome: Wallace Tavares
CNPJ: 12322323
Logradouro: Q47
Cidade: RJ
Estado: RJ
Telefone: 21099999999
Email: Wfelipetm@gmail.com
-------------------------------------------
Digite o nome da pessoa Juridica: Vanessa Nepomuceno
Digite o CNPJ da pessoa Juridica: 32323232
Digite o Logradouro da pessoa Juridica: L47
Digite o cidade da pessoa Juridica: RJ
Digite o estado da pessoa Juridica: RJ
Digite o telefone da pessoa Juridica: 219990909
Digite o email da pessoa Juridica: va@hotmail.com
Pessoa jur◆dica alterada com sucesso.
```

```
SQL 1 [N3]
Connection: N3
1  SELECT TOP 100 * FROM dbo.PessoaJuridica;
2
SELECT TOP 100 * FROM dbo...
Max. rows: 100   Fetched Rows: 3   Matching Rows:
```

#	idPessoaJuridica	idPessoa	nome	logradouro	cidade	estado	telefone	
3	1014	3073	Vanessa Nepomuce...	L47	RJ	RJ	219990909	va@hotmail.com
1	14	1069	Marina Nepomuceno	121233333	Itaguai	RJ	21989898989	mNep@gmail.com
2	1013	3068	João	234	RJ	RJ	21999999	Jnep@hotmail.com

Excluir Pessoa Juridica

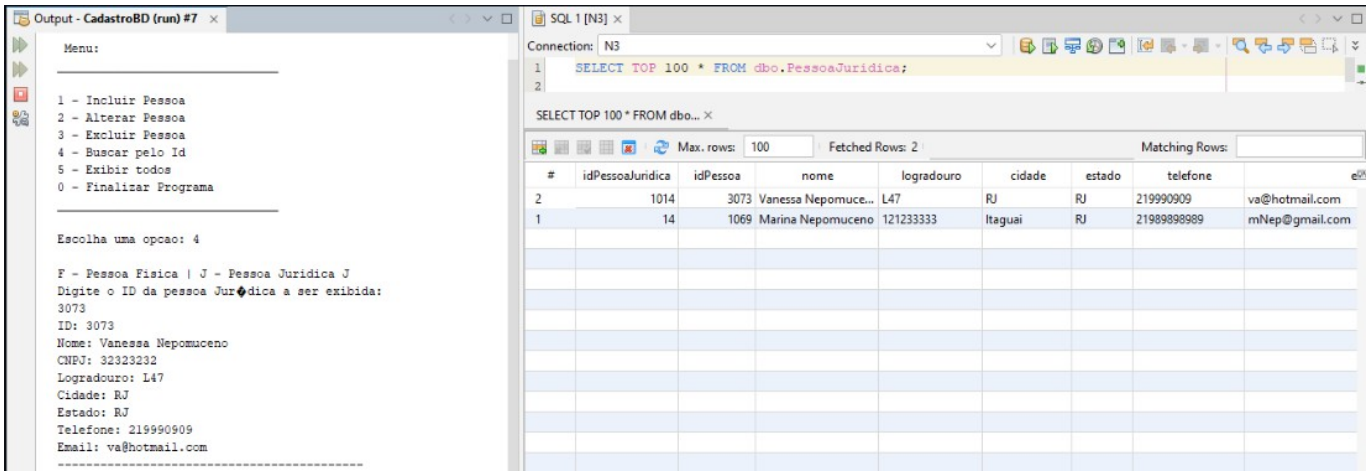


```
Output - CadastroBD (run) #6

run:

 Menu:
_____________________

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

Escolha uma opcao: 3

F - Pessoa Fisica | J - Pessoa Juridica J
Digite o ID da pessoa f◆sica a ser exclu◆da:
3068
Pessoa jur◆dica exclu◆da com sucesso.
Pessoa juridica excluida com sucesso.

F - Pessoa Fisica | J - Pessoa Juridica
```

```
SQL 1 [N3]
Connection: N3
1  SELECT TOP 100 * FROM dbo.PessoaJuridica;
2
SELECT TOP 100 * FROM dbo...
Max. rows: 100   Fetched Rows: 2   Matching Rows:
```

#	idPessoaJuridica	idPessoa	nome	logradouro	cidade	estado	telefone	
2	1014	3073	Vanessa Nepomuce...	L47	RJ	RJ	219990909	va@hotmail.com
1	14	1069	Marina Nepomuceno	121233333	Itaguai	RJ	21989898989	mNep@gmail.com

Buscar Pessoa Juridica por ID



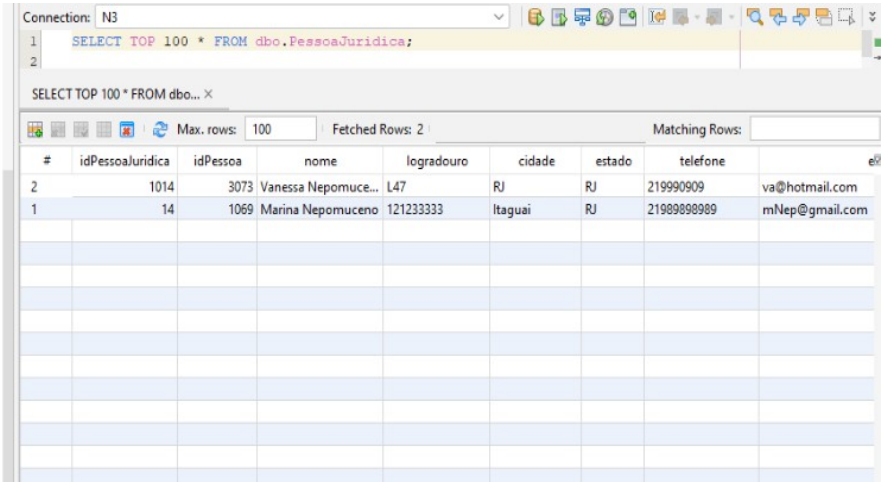
```
Output - CadastroBD (run) #7
Menu:
1 - Incluir Pessoa
2 - Alterar Pessoa
3 - Excluir Pessoa
4 - Buscar pelo Id
5 - Exibir todos
0 - Finalizar Programa
Escolha uma opcao: 4
F - Pessoa Fisica | J - Pessoa Juridica J
Digite o ID da pessoa Jur?dica a ser exibida:
3073
ID: 3073
Nome: Vanessa Nepomuceno
CNPJ: 32323232
Logradouro: L47
Cidade: RJ
Estado: RJ
Telefone: 219990909
Email: va@hotmail.com
```

```
SQL 1 [N3]
Connection: N3
SELECT TOP 100 * FROM dbo.PessoaJuridica;
SELECT TOP 100 * FROM dbo...
Max. rows: 100  Fetched Rows: 2  Matching Rows:
```

#	idPessoaJuridica	idPessoa	nome	logradouro	cidade	estado	telefone	
2	1014	3073	Vanessa Nepomuce...	L47	RJ	RJ	219990909	va@hotmail.com
1	14	1069	Marina Nepomuceno	121233333	Itaguai	RJ	21989898989	mNep@gmail.com

Exibir todas Pessoa Juridica

```
Escolha uma opcao: 5
F - Pessoa Fisica | J - Pessoa Juridica J
Exibindo dados de Pessoa Juridica...
ID: 1069
Nome: Marina Nepomuceno
CNPJ: 12213323443
Logradouro: 121233333
Cidade: Itaguai
Estado: RJ
Telefone: 21989898989
Email: mNep@gmail.com
ID: 3073
Nome: Vanessa Nepomuceno
CNPJ: 32323232
Logradouro: L47
Cidade: RJ
Estado: RJ
Telefone: 219990909
Email: va@hotmail.com
```



```
Connection: N3
SELECT TOP 100 * FROM dbo.PessoaJuridica;
SELECT TOP 100 * FROM dbo...
Max. rows: 100  Fetched Rows: 2  Matching Rows:
```

#	idPessoaJuridica	idPessoa	nome	logradouro	cidade	estado	telefone	
2	1014	3073	Vanessa Nepomuce...	L47	RJ	RJ	219990909	va@hotmail.com
1	14	1069	Marina Nepomuceno	121233333	Itaguai	RJ	21989898989	mNep@gmail.com