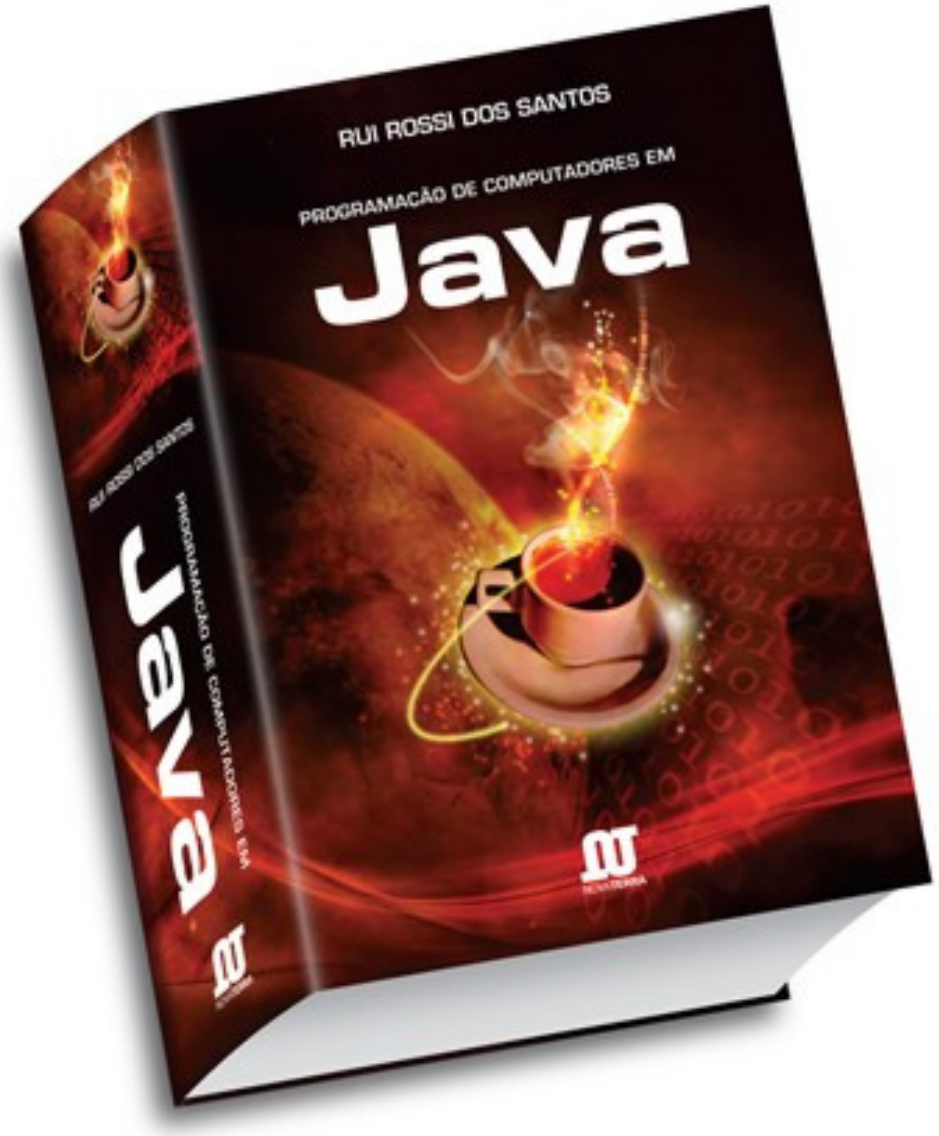


Capítulo 35

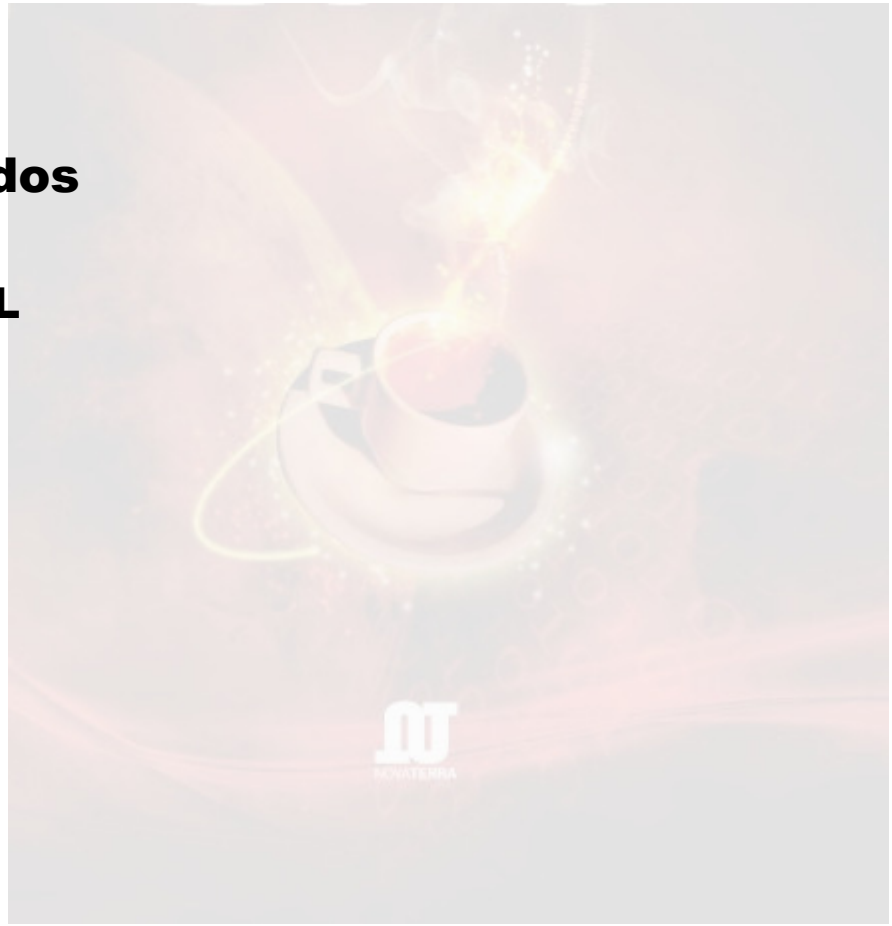
Sistemas de Banco de Dados

Objetivos do Capítulo

- Instalar e configurar um sistema gerenciador de banco de dados relacional.
- Apresentar a API do Java que permite aos aplicativos acessarem bancos de dados de diferentes fornecedores: a JDBC.
- Indicar como realizar operações fundamentais de manipulação de registros: inclusão, consulta, exclusão e alteração.
- Analisar os recursos que permitem extrair dados relativos à estrutura de um banco de dados ou de um resultado produzido por uma consulta.

Introdução

- **Persistência: gravação em mecanismo não volátil**
  - **Arquivos**
    - **TXT**
    - **XML**
  - **Banco de dados**
    - **MySQL**
    - **PostgreSQL**
    - **Oracle**

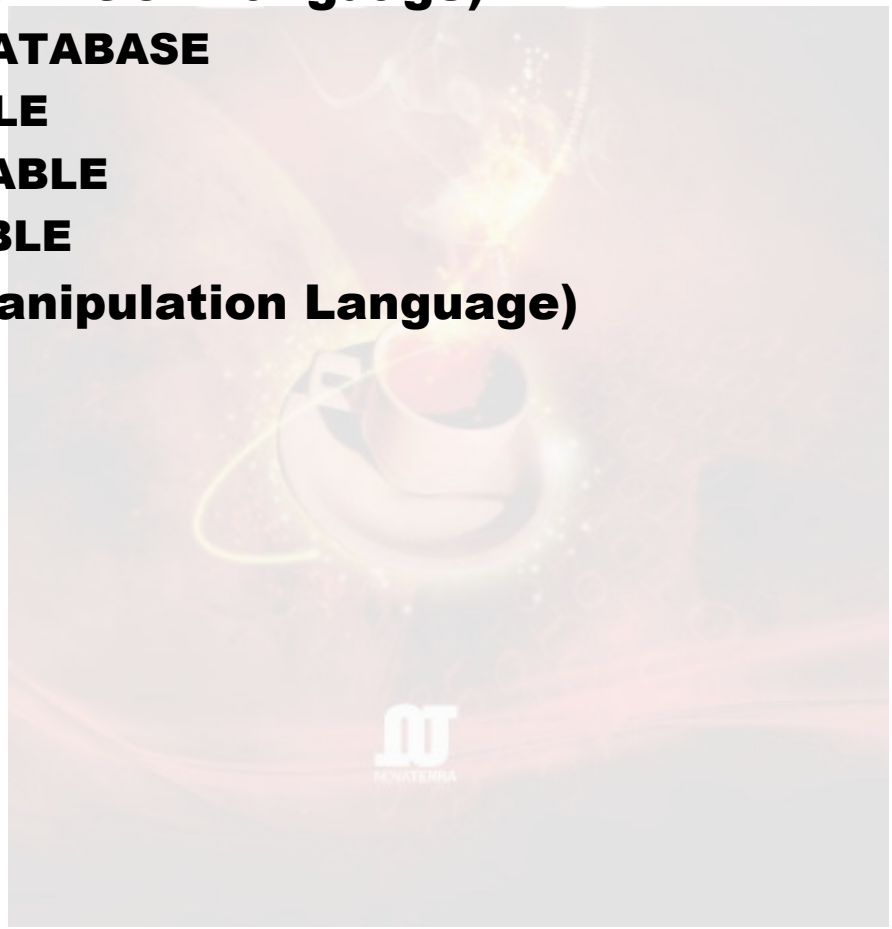


Introdução

- **SGBD**
  - **Relacional (SGBDR)**
    - **Tabelas (Entidades)**
      - **Colunas (Campos ou Atributos)**
      - **Linhas (Registros ou Tuplas)**
    - **Relacionamentos (Chaves Estrangeiras)**
  - **Orientado a Objetos (SGBDOO)**
    - **Objetos**
    - **Ex.: DB4O**

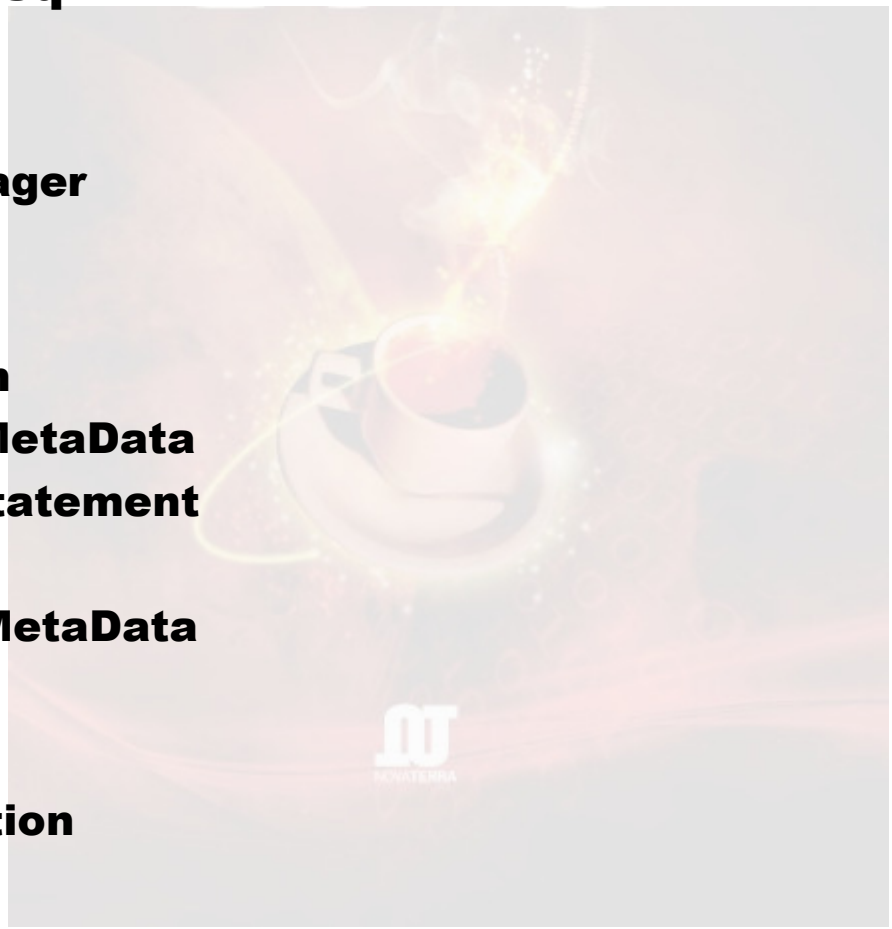
Introdução

- SQL (Structured Query Language)
  - DDL (Data Definition Language)
    - CREATE DATABASE
    - DROP TABLE
    - CREATE TABLE
    - ALTER TABLE
  - DML (Data Manipulation Language)
    - INSERT
    - UPDATE
    - DELETE
    - SELECT



Introdução

- **API: JDBC**
  - **Pacote: java.sql**
  - **Classes:**
    - **Date**
    - **DriverManager**
    - **Time**
  - **Interfaces**
    - **Connection**
    - **DataBaseMetaData**
    - **PreparedStatement**
    - **ResultSet**
    - **ResultSetMetaData**
    - **Statement**
  - **Exceções**
    - **SQLException**



Servidor de Banco de Dados

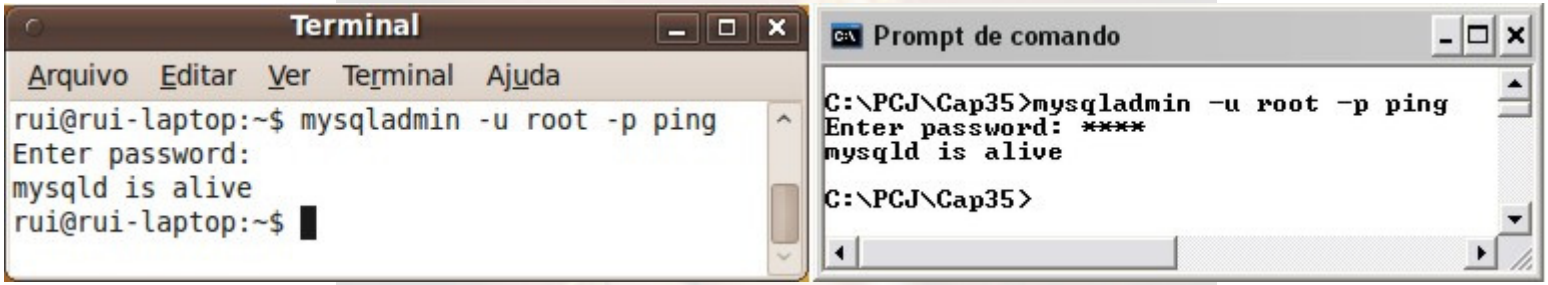
- **MySQL Server**
  - **Web site: http://www.mysql.com**
  - **Uso livre: MySQL Community Server**

- **Instalação**
  - **Pular criação de conta (Skip Sign-Up)**
  - **Configurar a instância do servidor**
    - **Assistente: MySQL Server Instance Configuration Wizard**
    - **Tipo de configuração: Standard Configuration**
    - **Install As Windows Service**
    - **Include Bin Directory in Windows PATH**
    - **Senha do root**

Servidor de Banco de Dados

- **Comando: testando a conectividade**

mysqladmin –u root –p ping



Opções:

- **-u: especifica o login do usuário**
- **-p: especifica que a senha do usuário deverá ser solicitada.**

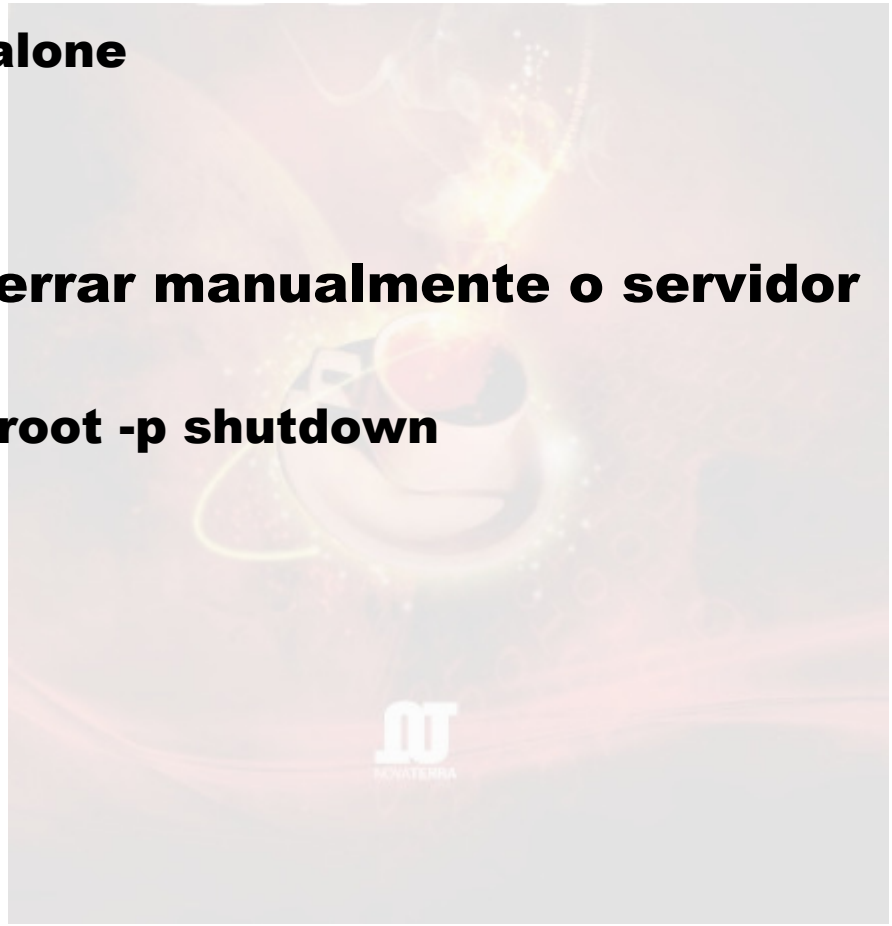
Servidor de Banco de Dados

- **Comando: iniciar manualmente o servidor**

  mysqld --standalone

- **Comando: encerrar manualmente o servidor**

  mysqladmin -u root -p shutdown



Driver JDBC

- **Intermediário: aplicativo Java e banco de dados**

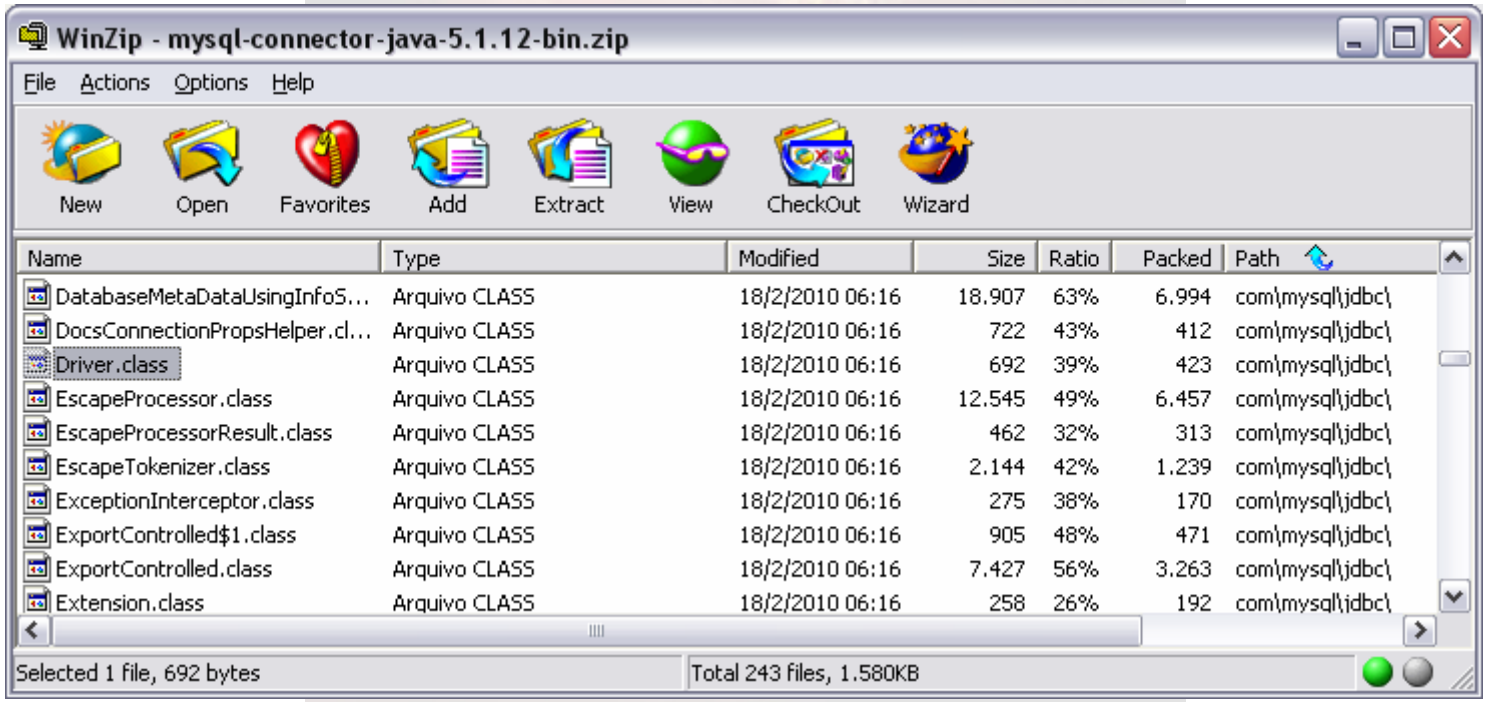
- **Tipos:**
  - **Tipo 1: emprega a API ODBC (Open DataBase Connectivity) para realizar toda a comunicação com o banco de dados e mapeia a chamada a cada método da JDBC para uma função ODBC.**
  - **Tipo 2: utiliza bibliotecas escritas em código nativo da máquina cliente e converte cada chamada de método JDBC para uma chamada a esta biblioteca nativa.**
  - **Tipo 3: converte cada chamada de método JDBC para uma chamada de rede enviada a um servidor que utiliza um protocolo independente de banco de dados. Este servidor exerce a função de middleware, ou seja, ele funciona como um componente intermediário entre a máquina cliente e o banco de dados.**
  - **Tipo 4: é um driver completamente escrito em Java e converte todas as chamadas a métodos JDBC diretamente para o protocolo específico do banco de dados.**

Driver JDBC

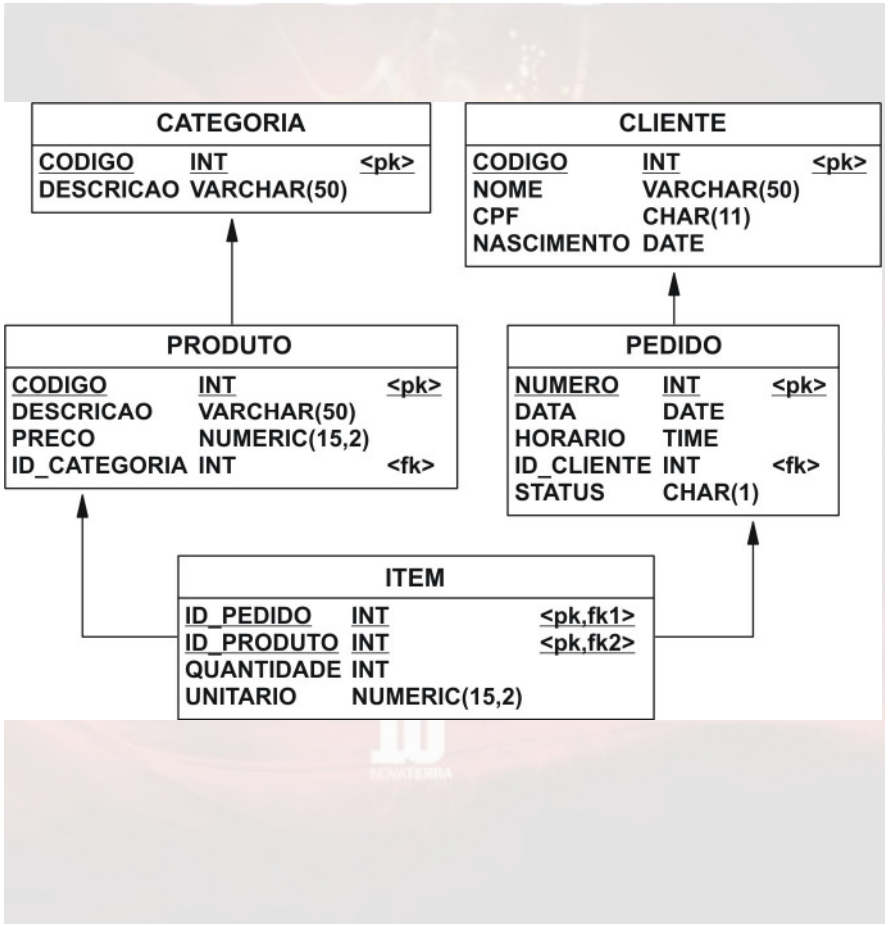
- **Comparativo:**
  - **Tipo 1: perde em eficiência por ter de converter cada chamada para o padrão ODBC e requer que cada máquina cliente tenha o driver ODBC instalado.**
  - **Tipo 2: tem melhor performance que o anterior, mas requer que as bibliotecas do banco de dados estejam instaladas em cada máquina cliente e a disponibilidade delas depende do sistema operacional utilizado.**
  - **Tipo 3: requer a construção de um servidor intermediário e a escrita de código específico do banco de dados utilizado.**
  - **Tipo 4: tem uma performance consideravelmente melhor que as opções anteriores por estabelecer uma comunicação direta com o banco de dados. Para que ele funcione, basta incluí-lo junto à aplicação que será utilizada para o acesso ao banco de dados.**

Driver JDBC

- **MySQL Connector: Driver JDBC Tipo 4**
  - **Arquivo: mysql-connector-java-5.1.12-bin.jar (716KB)**



Criação de um Banco de Dados



Criação de um Banco de Dados

- **Procedimento via linha de comando:**
  - **No Windows:**

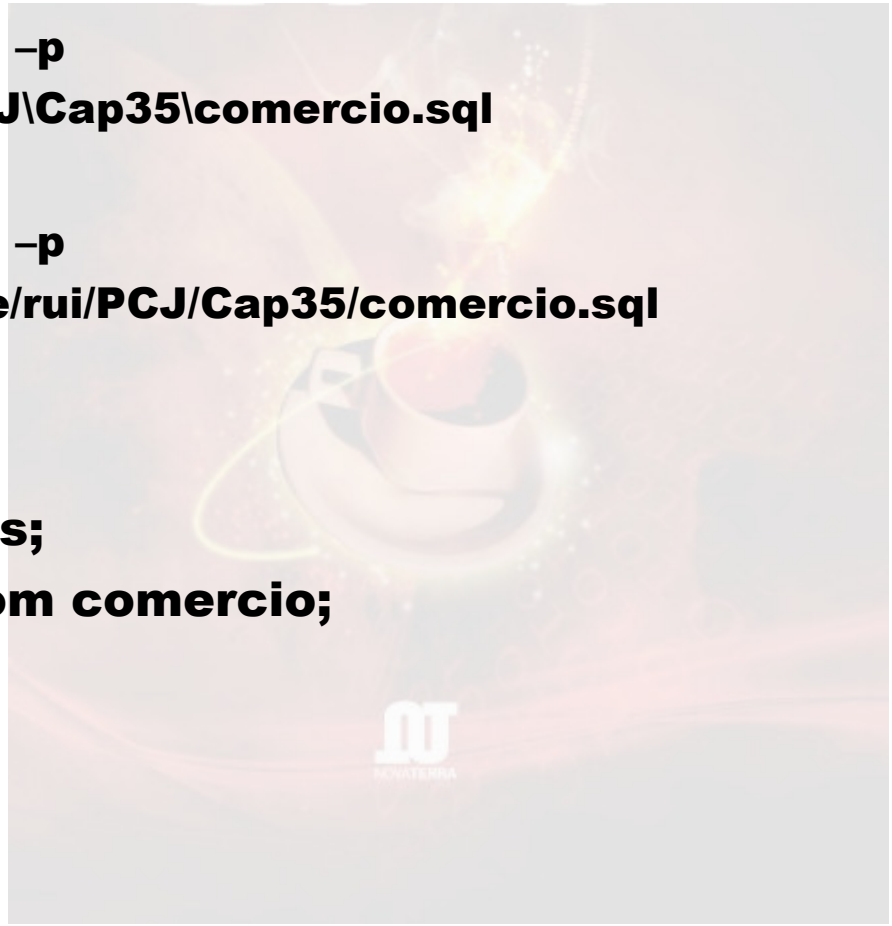
    ```
    mysql -u root –p
    source C:\PCJ\Cap35\comercio.sql
    ```

  - **No Linux:**

    ```
    mysql -u root –p
    source /home/rui/PCJ/Cap35/comercio.sql
    ```

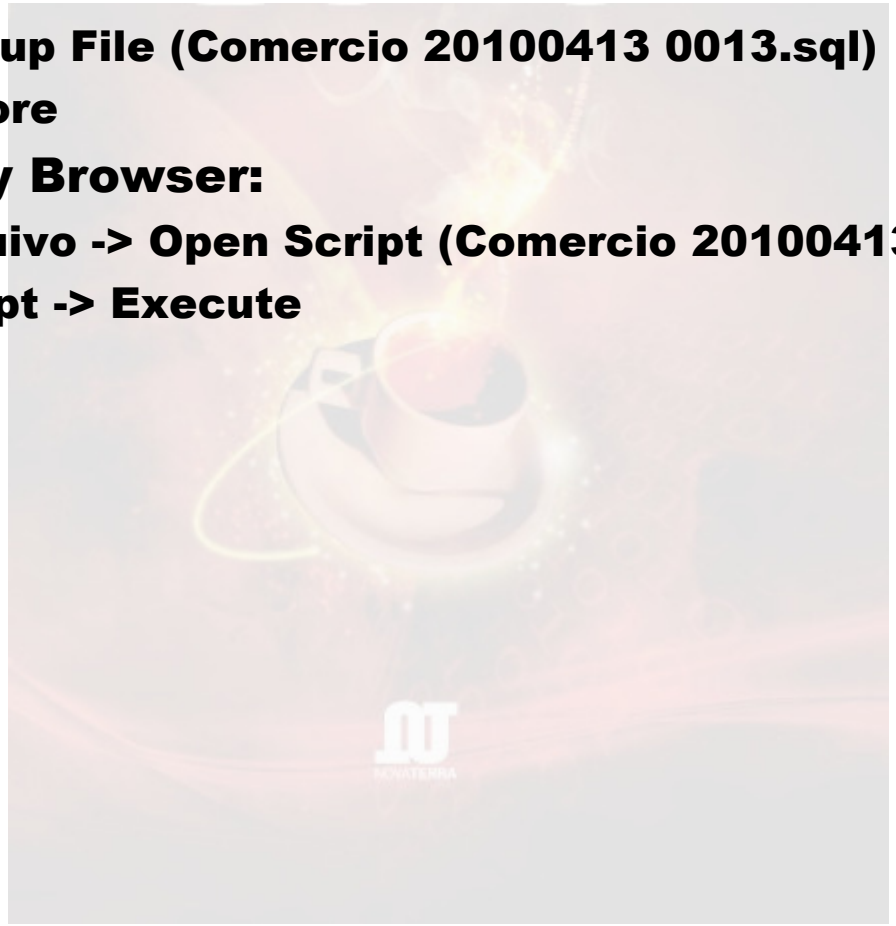
- **Conferência:**

  ```
  show databases;
  show tables from comercio;
  ```



Criação de um Banco de Dados

- **Procedimento com ferramentas visuais:**
  - **MySQL Administrator:**
    - **Open Backup File (Comercio 20100413 0013.sql)**
    - **Start Restore**
  - **MySQL Query Browser:**
    - **Menu: Arquivo -> Open Script (Comercio 20100413 0013.sql)**
    - **Menu: Script -> Execute**



Conexão com o Banco de Dados

- **Registrar o driver JDBC**

```
Class.forName("com.mysql.jdbc.Driver");
```

- **Abrir uma conexão com o banco de dados**

```
Connection conexao = DriverManager.getConnection(
  "jdbc:mysql://localhost/comercio","root","root");
```

- **Configurar a conexão**

```
conexao.setAutoCommit(false);
conexao.setTransactionIsolation(
    Connection.TRANSACTION_READ_COMMITTED);
```

Conexão com o Banco de Dados

- **Fechar a conexão**

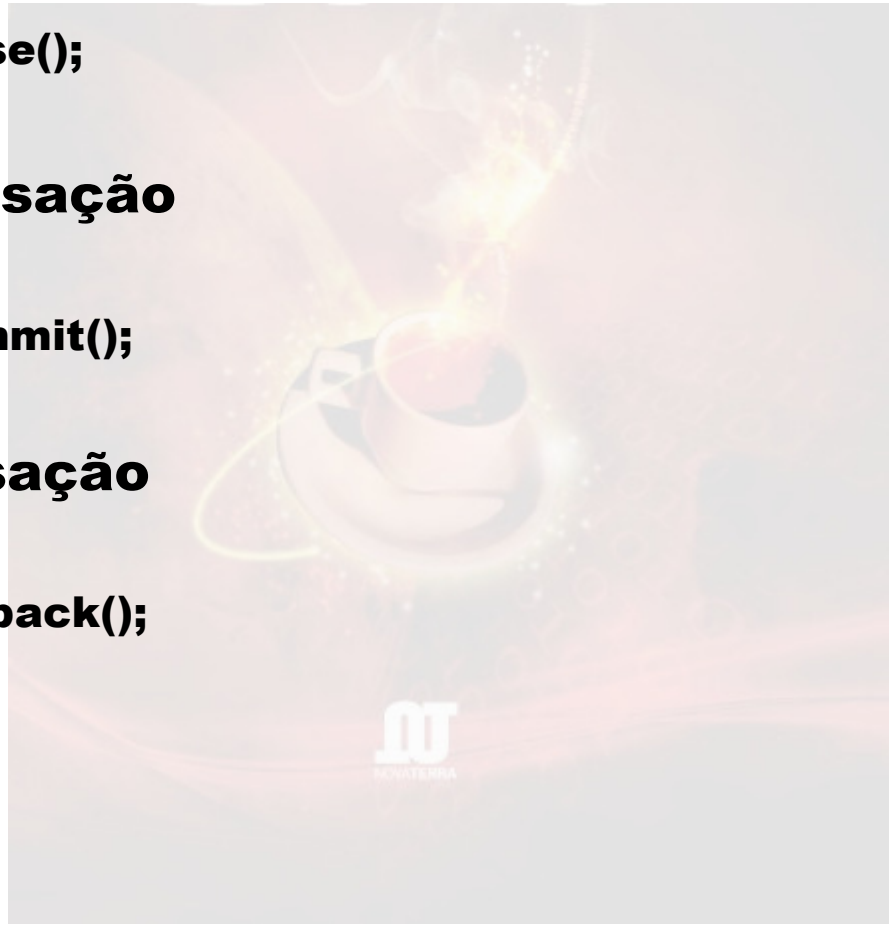
```
conexao.close();
```

- **Confirmar transação**

```
conexao.commit();
```

- **Cancelar transação**

```
conexao.rollback();
```



Conexão com o Banco de Dados

- **Interface java.sql.Connection**
  - **Métodos:**
    - **void close( ) throws SQLException**
    - **boolean isClosed( ) throws SQLException**
    - **void setAutoCommit(boolean autoCommit) throws SQLException**
    - **boolean getAutoCommit( ) throws SQLException**
    - **void commit( ) throws SQLException**
    - **void rollback( ) throws SQLException**
    - **void setTransactionIsolation(int level) throws SQLException**
    - **int getTransactionIsolation( ) throws SQLException**
    - **Statement createStatement( ) throws SQLException**
    - **PreparedStatement prepareStatement(String sql, int autoGeneratedKeys) throws SQLException**
    - **CallableStatement prepareCall(String sql) throws SQLException**
    - **DataBaseMetaData getMetaData( ) throws SQLException**

Conexão com o Banco de Dados

- **Inconsistências comuns em ambiente concorrente:**
  - **Dirty read (leitura suja): uma transação lê informações que ainda não foram confirmadas por outra transação.**
  - **Non-repeatable read (leitura não-repetida): uma transação recupera um mesmo registro por duas vezes e os valores obtidos são diferentes em função de mudanças realizadas no mesmo por outras transações.**
  - **Phantom read (leitura fantasma): uma transação seleciona um conjunto de registros por duas vezes consecutivas e a quantidade de registros recuperados em cada uma delas é diferente em função de terem sido excluídos ou acrescentados registros por outras transações.**

Conexão com o Banco de Dados

- **Interface java.sql.Connection**
  - **Atributos usados no o método setTransactionIsolation( ):**
    - **TRANSACTION_READ_UNCOMMITED: permite que os três tipos de inconsistências.**
    - **TRANSACTION_READ_COMMITED: previne dirty reads, mas permite que non-repeatable reads e phantom reads ocorram.**
    - **TRANSACTION_REPEATABLE_READ: previne dirty reads e non-repeatable reads, mas permite que phantom reads ocorram.**
    - **TRANSACTION_SERIALIZABLE: previne os três tipos de inconsistências supracitadas.**

Conexão com o Banco de Dados

- Código 35.1 – comercio.sql
- Código 35.2 – ConexaoComercio.java
- Código 35.3 – TesteConexao.java

Inclusão de Registros

- **Instrução INSERT**
  - **Sintaxe**

```
INSERT INTO <TABELA> VALUES(<Valor1>,<Valor2>,..,<ValorN>)
```

  - **Exemplos**

```
INSERT INTO CATEGORIA VALUES(1,'Livros')
INSERT INTO CATEGORIA(DESCRICAO) VALUES('Livros')
```

Inclusão de Registros

- **Passos:**
  - **Abrir uma conexão**
  - **Enviar uma instrução INSERT**
    - **Preparar a instrução (PreparedStatement)**
    - **Definir valores para os parâmetros**
    - **Executar a instrução**
  - **Confirmar a transação**
  - **Recuperar a chave gerada (opcional)**

Inclusão de Registros

- **Exemplo**

```
PreparedStatement pst = conexao.prepareStatement(
  "INSERT INTO CATEGORIA (DESCRICAO) VALUES(?)",
  Statement.RETURN_GENERATED_KEYS);
pst.setString(1, “Livro”);
pst.executeUpdate();
conexao.commit();
ResultSet rs = pst.getGeneratedKeys();
rs.next();
int chave_primaria = rs.getInt(1);
```

- **java.sql.Connection**

```
PreparedStatement prepareStatement(String sql,
  int autoGenerateKeys) throws SQLException
```

- **java.sql.Statement**
  - **Atributos:**
    - **RETURN_GENERATED_KEYS**
    - **NO_GENERATED_KEYS**

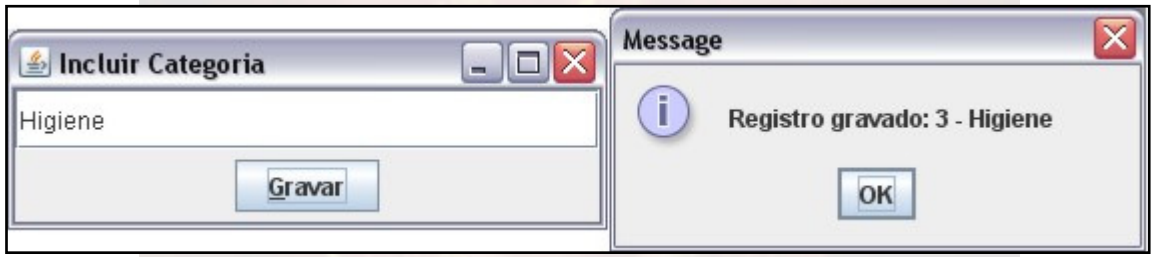
Inclusão de Registros

- **Métodos de java.sql.PreparedStatement**
  - **void setString(int parameterIndex, String x)**
  - **void setDate(int parameterIndex, java.sql.Date x)**
  - **void setTime(int parameterIndex, java.sql.Time x)**
  - **void setInt(int parameterIndex, int x)**
  - **void setDouble(int parameterIndex, double x)**
  - **int executeUpdate( ) throws SQLException**
  - **ResultSet getGeneratedKeys( ) throws SQLException**

- **Métodos de java.sql.ResultSet**
  - **boolean next( ) throws SQLException**
  - **int getInt(int columnIndex) throws SQLException**
  - **int getString(int columnIndex) throws SQLException**

Inclusão de Registros

- **Código 35.4 – JFCategoriaIncluir.java**



Recuperação de Registros

❑ Instrução SELECT

➢ Sintaxe

```
SELECT <CAMPOS> FROM <TABELA>
WHERE <CONDIÇÃO>
ORDER BY <CAMPO>
```

➢ Exemplos

```
SELECT * FROM CATEGORIA
ORDER BY DESCRICAO
```

```
SELECT * FROM CATEGORIA
WHERE (CODIGO <= 5) AND (DESCRICAO LIKE '%im%')
ORDER BY DESCRICAO
```

Recuperação de Registros

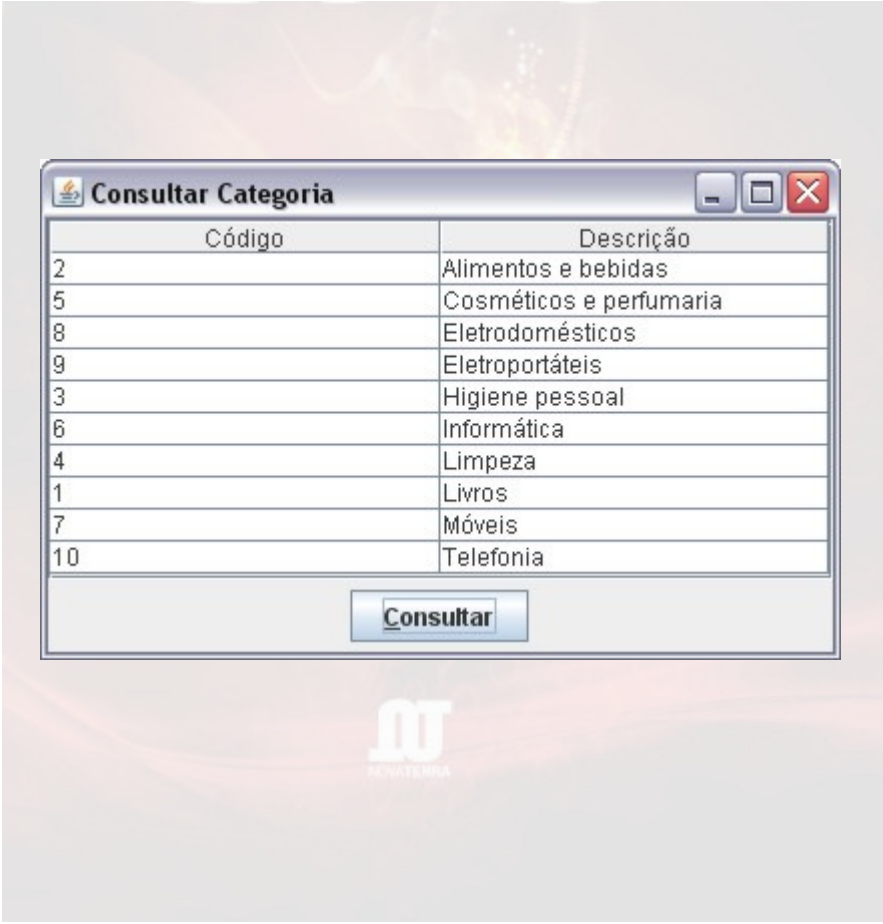
- **Passos:**
  - **Abrir uma conexão**
  - **Enviar uma instrução SELECT**
    - **Criar o objeto responsável pelo envio da instrução (Statement)**
    - **Executar a instrução**
  - **Percorrer os registros recuperados**

- **Exemplo**

```
Statement stm = conexao.createStatement();
ResultSet rs = stm.executeQuery("SELECT * FROM CATEGORIA");
while(rs.next( ))
  System.out.println(rs.getInt(1) + " - " + rs.getString(2));
```

Recuperação de Registros

- **Código 35.5 – JFCategoriaConsultar.java**



Alteração de Registros

❑ Instrução UPDATE

➢ Sintaxe

```
UPDATE <TABELA>
SET <CAMPO1> = <VALOR1>,<CAMPO2> = <VALOR2>
WHERE <CONDIÇÃO>
```

➢ Exemplos

```
UPTADE CATEGORIA
SET DESCRICAO = 'Perfumaria e beleza'
WHERE CODIGO = 5
```

Alteração de Registros

- **Passos:**
  - **Abrir uma conexão**
  - **Enviar uma instrução UPDATE**
    - **Preparar a instrução (PreparedStatement)**
    - **Definir valores para os parâmetros**
    - **Executar a instrução**
  - **Confirmar a transação**
  - **Recuperar a quantidade de registros afetada (opcional)**

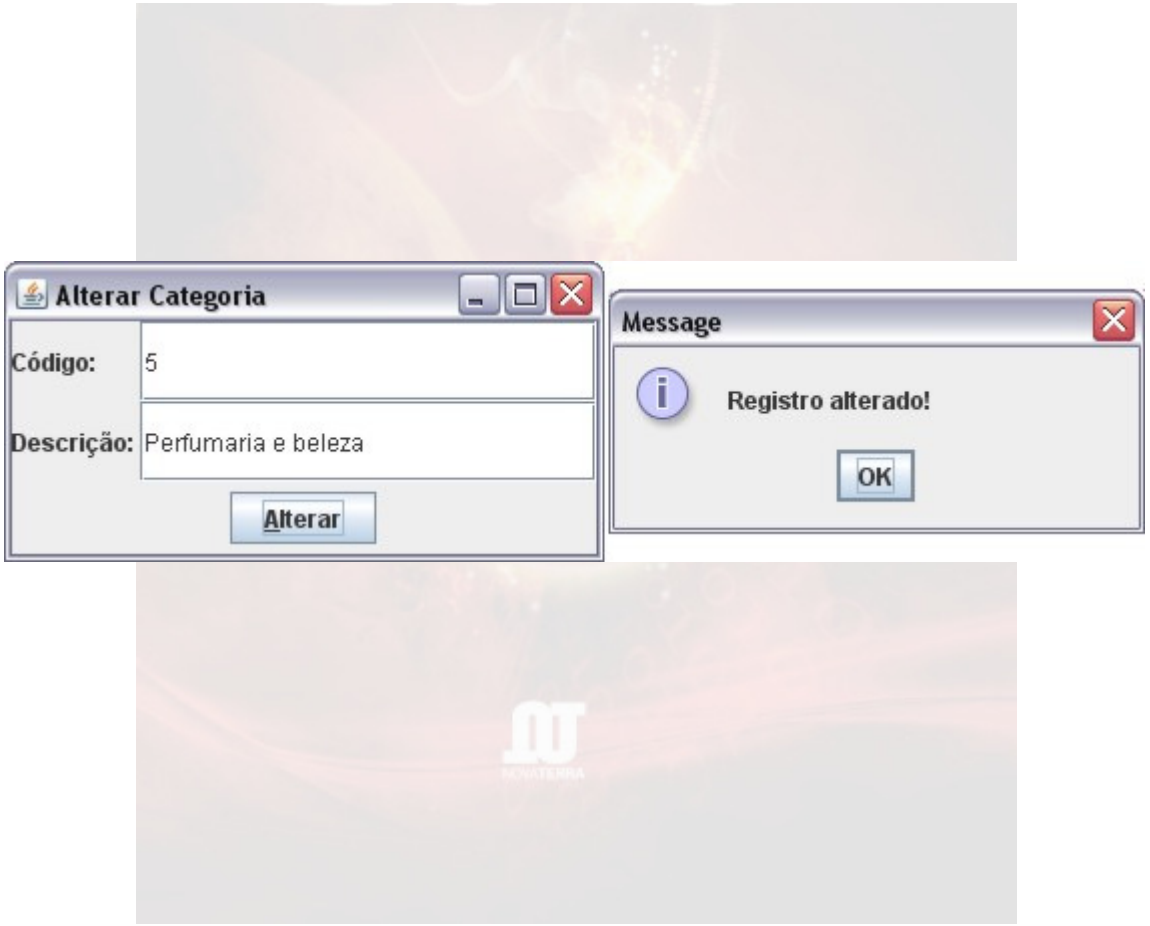
Alteração de Registros

- **Exemplo**

```
PreparedStatement pst = conexao.prepareStatement(
  "UPDATE CATEGORIA SET DESCRICAO = ? WHERE CODIGO = ?");
pst.setString(1, “Perfumaria e beleza”);
pst.setInt(2, 5);
pst.executeUpdate();
conexao.commit();
if (pst.getUpdateCount() > 0) System.out.println(“Alterado!”);
else System.out.println(“Registro não existe!”);
```

Alteração de Registros

- **Código 35.6 – JFCategoriaAlterar.java**



Exclusão de Registros

- **Instrução DELETE**

  - **Sintaxe**

```
DELETE FROM <TABELA>
WHERE <CONDIÇÃO>
```

  - **Exemplos**

```
DELETE FROM CATEGORIA
WHERE CODIGO = 1
```

Exclusão de Registros

- **Passos:**
  - **Abrir uma conexão**
  - **Enviar uma instrução DELETE**
    - **Preparar a instrução (PreparedStatement)**
    - **Definir valores para os parâmetros**
    - **Executar a instrução**
  - **Confirmar a transação**
  - **Recuperar a quantidade de registros afetada (opcional)**

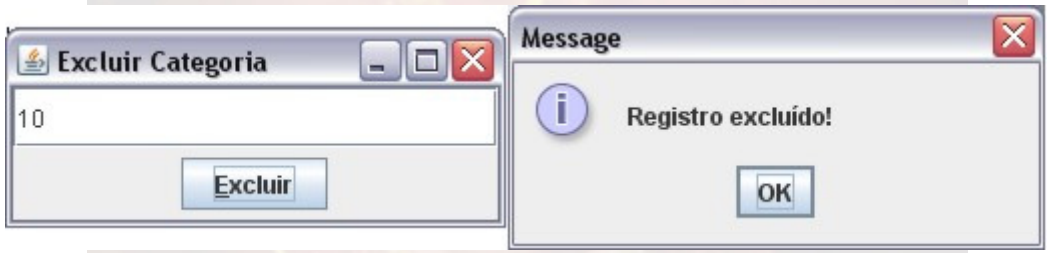
Exclusão de Registros

- **Exemplo**

```
PreparedStatement pst = conexao.prepareStatement(
  "DELETE FROM CATEGORIA WHERE CODIGO = ?");
pst.setInt(1, 10);
pst.executeUpdate();
conexao.commit();
if (pst.getUpdateCount() > 0) System.out.println("Excluído!");
else System.out.println("Registro não existe!");
```

Exclusão de Registros

- **Código 35.7 – JFCategoriaExcluir.java**



Recuperação de Metadados

- **Metadados**
  - **Dados sobre dados**
  - **Descrições dos dados**
  - **Estrutura dos dados**

- **Objetivo**
  - **Análise dinâmica sobre a estrutura do banco**
  - **Análise dinâmica sobre um conjunto de resultados**

Recuperação de Metadados

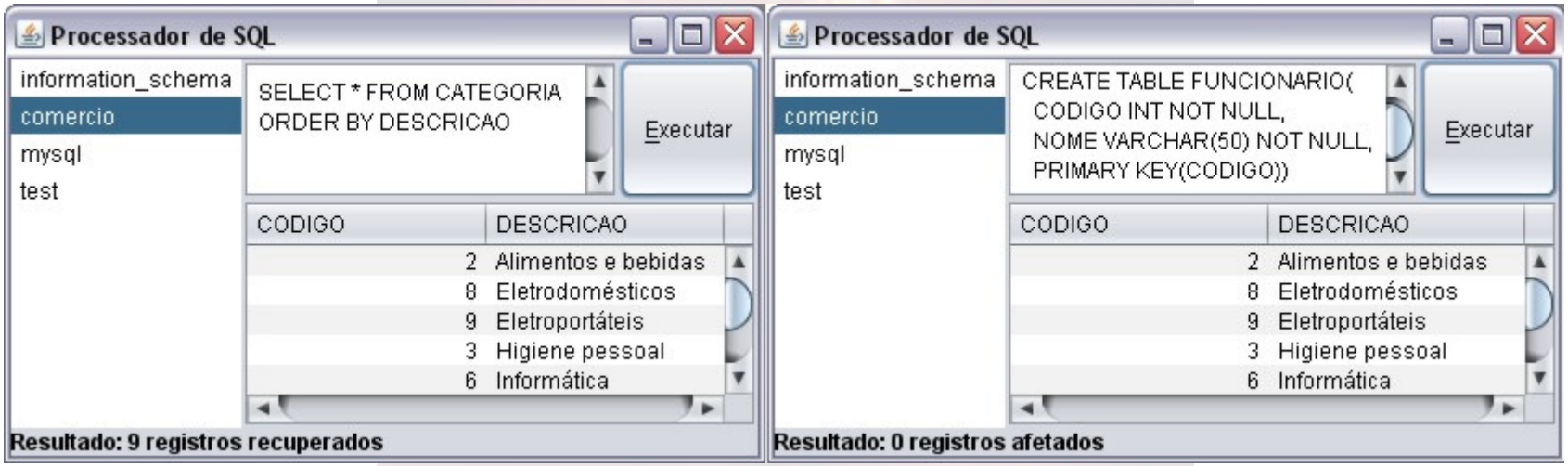
- **API:**
  - **java.sql.Connection**
    - DatabaseMetaData getMetaData( )
  - **java.sql.DatabaseMetaData**
    - ResultSet getCatalogs( )
    - ResultSet getTables(String catalog, String schemaPattern, String tableNamePattern, String[] types)
    - ResultSet getColumns(String catalog, String schemaPattern, String tableNamePattern, String columnNamePattern)
  - **java.sql.ResultSet**
    - ResultSetMetaData getMetaData( )
  - **java.sql.ResultSetMetaData**
    - int getColumnCount( )
    - String getColumnLabel(int column)
    - int getColumnType(int column)

Recuperação de Metadados

- Código 35.8 – ModeloGrade.java
- Código 35.9 – DBExplorer.java

Processamento de Instruções SQL

- **Código 35.10 – ProcessadorSQL.java**



Estudo de Caso: Registro de Pedidos

- **Camada de Transferência:**
  - **Categoria.java**
  - **Produto.java**
  - **Cliente.java**
  - **Item.java**
  - **Pedido.java**

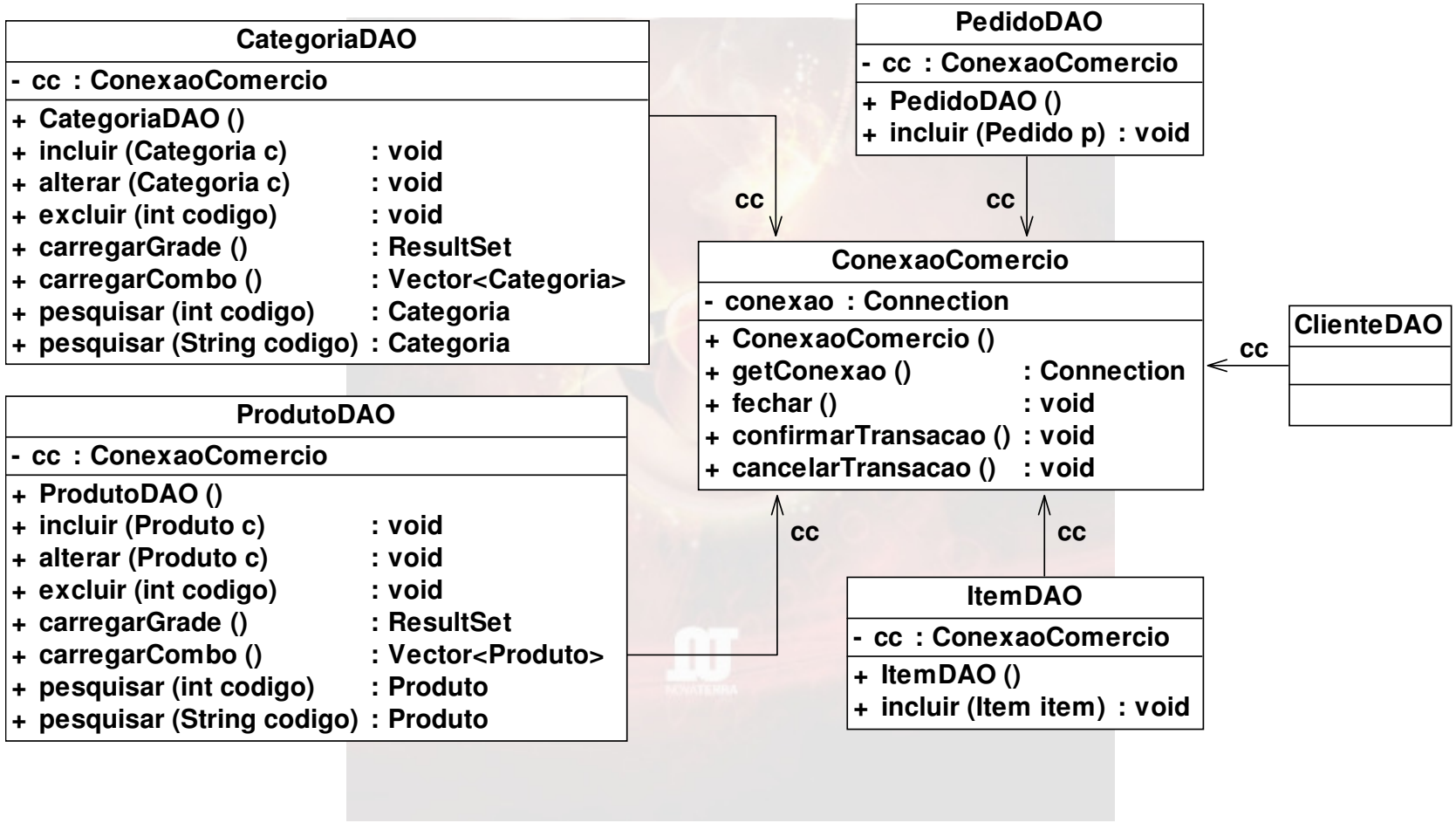
Categoria	
- codigo	: int
- descricao	: String
+ Categoria ()	
+ Categoria (int codigo, String descricao)	
+ getCodigo ()	: int
+ getDescricao ()	: String
+ setCodigo (int codigo)	: void
+ setCodigo (String codigo)	: void
+ setDescricao (String descricao)	: void
+ toString ()	: String
+ equals (Object obj)	: boolean
+ hashCode ()	: int

Estudo de Caso: Registro de Pedidos

- **Camada de Persistência**
  - **ConexaoComercio.java**
  - **CategoriaDAO.java**
  - **ProdutoDAO.java**
  - **ClienteDAO.java**
  - **ItemDAO**
  - **PedidoDAO**

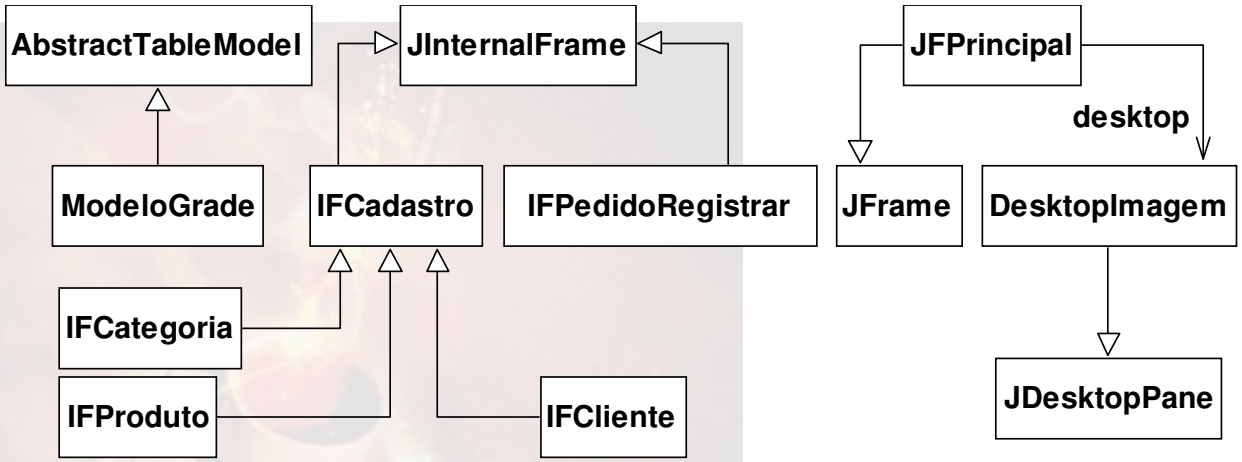
Estudo de Caso: Registro de Pedidos

❑ Camada de Persistência



Estudo de Caso: Registro de Pedidos

- **Camada de Apresentacao:**
  - **ModeloGrade**
  - **IFCadastro**
  - **IFCategoria**
  - **IFProduto**
  - **IFCliente**
  - **IFPedidoRegistrar**
  - **DesktopImagem**
  - **JFPrincipal**



- **Classe Principal:**
  - **Principal**

Estudo de Caso: Registro de Pedidos

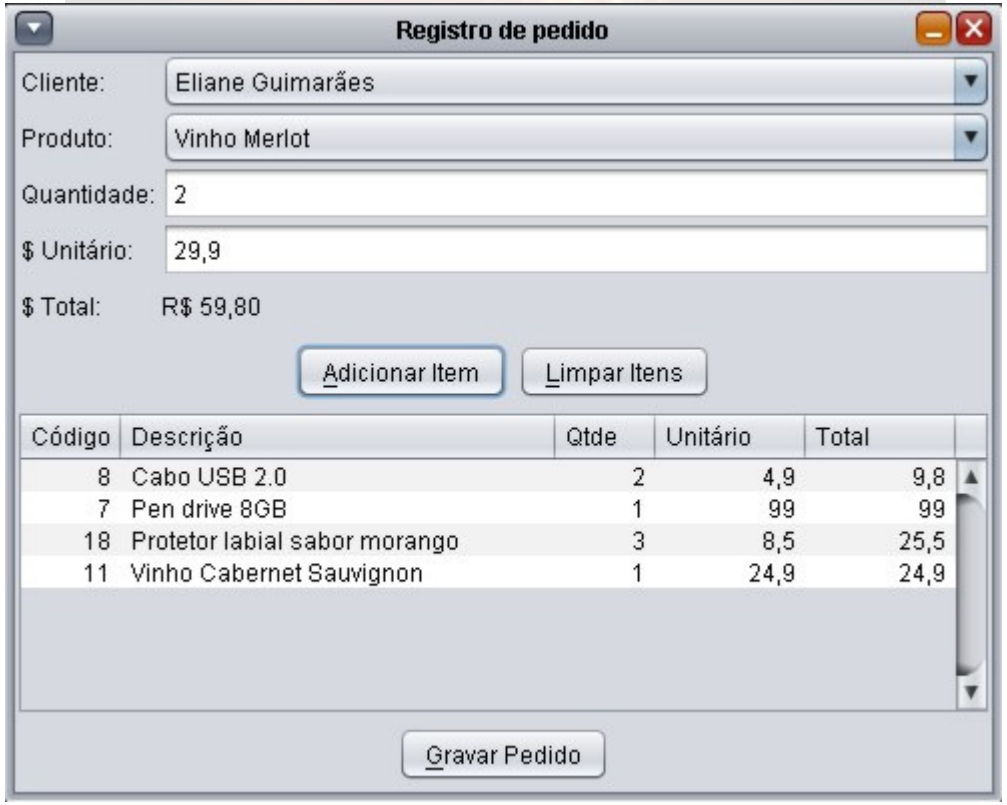
- **Janelas:**
  - **IFCategoria**
  - **IFProduto**

Estudo de Caso: Registro de Pedidos

- **Janelas:**
  - **IFCliente**

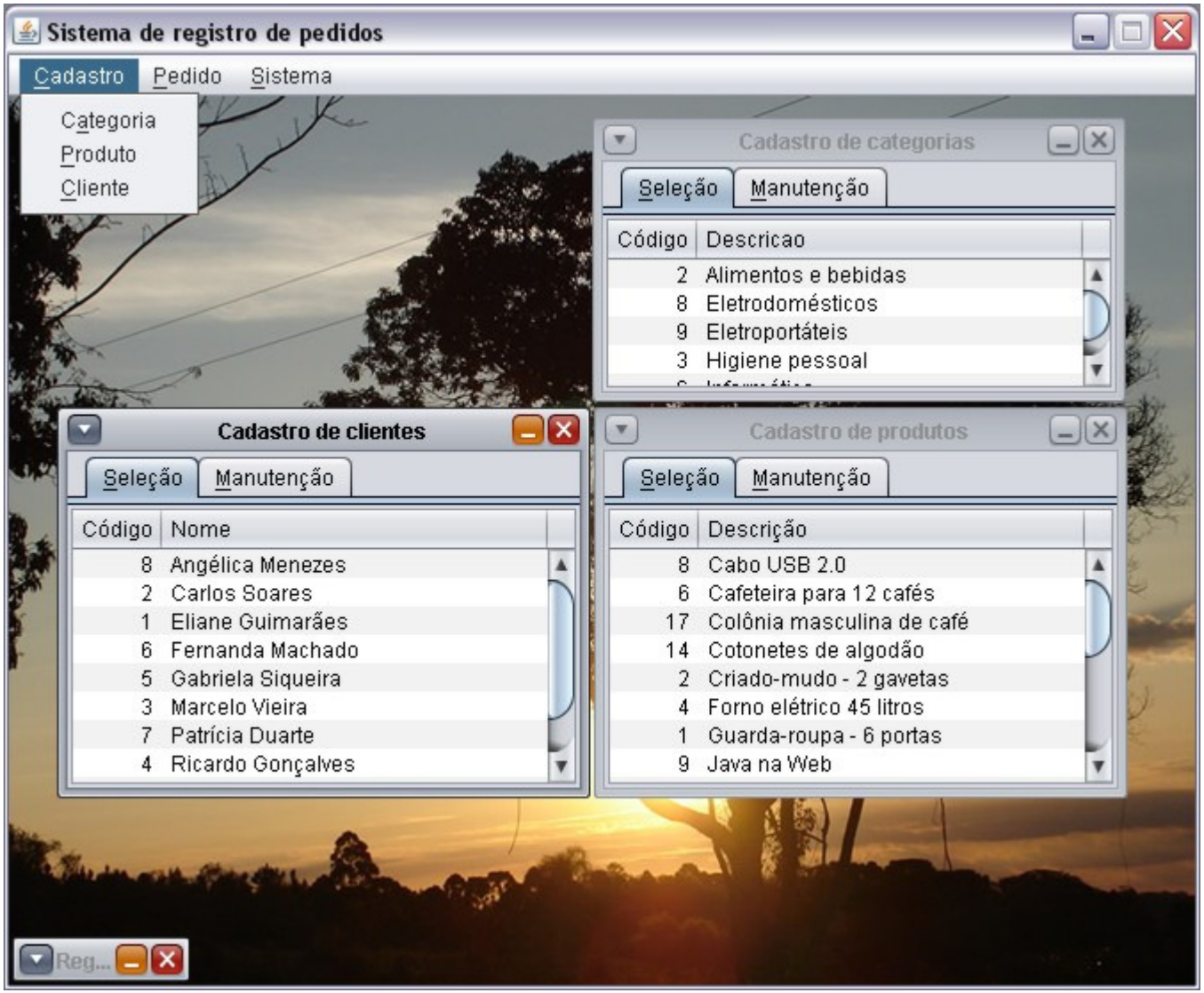
Estudo de Caso: Registro de Pedidos

- **Janelas:**
  - **IFPedidoRegistrar**



Estudo de Caso: Registro de Pedidos

- **Janelas:**
  - **JFPrincipal**



Exercício 1

- **Amplie o sistema de registro de pedidos que fora apresentado neste capítulo e acrescente uma janela que permita registrar a entrega de um pedido.**
  - **Esta janela deve ter apenas uma grade e um botão.**
  - **A grade deve apresentar todos os pedidos que ainda se encontram abertos (com o status igual à 'A').**
  - **O botão deverá ser rotulado como "Fechar Pedido" e servirá para marcar o pedido como entregue.**
  - **Quando um pedido for marcado como entregue, seu status deve ser modificado para 'F' (Fechado) e ele deve ser eliminado imediatamente da grade.**

Exercício 2

- **Amplie o sistema de registro de pedidos que fora apresentado neste capítulo e acrescente uma janela que permita consultar todos os pedidos que foram registrados entre duas datas.**
  - **Esta janela deve conter dois campos de texto formatados para captar a data inicial e a data final do período desejado, deve ter uma grade para apresentar os registros e um botão para acionar a consulta.**
  - **Esta janela também deve ter uma caixa de combinação que permita selecionar uma dentre três opções: visualizar apenas os pedidos que encontram-se abertos, apenas os pedidos fechados ou ambos.**

Exercício 3

- **Amplie o sistema de registro de pedidos que fora apresentado neste capítulo e acrescente uma janela que permita realizar o cadastro de funcionários.**
  - **O registro de cada funcionário deverá ser composto pelos seguintes dados: um código numérico, seu nome completo, seu CPF, seu RG, seu telefone, seu e-mail, sua data de nascimento e seu salário.**
  - **Antes de criar esta janela, acrescente uma nova tabela no banco de dados e a chame de FUNCIONARIO.**
    - **Esta tabela deverá ter um campo para cada um dos dados supracitados.**
    - **Defina o código do funcionário como um campo que seja incrementado pelo próprio banco de dados.**

Contato

Com o autor:

Rui Rossi dos Santos
E-mail: livros@ruirossi.pro.br
Web Site: http://www.ruirossi.pro.br

Com a editora:

Editora NovaTerra
Telefone: (21) 2218-5314
Web Site: http://www.editoranovaterra.com.br