



Estácio

JDBC

OBJETIVOS

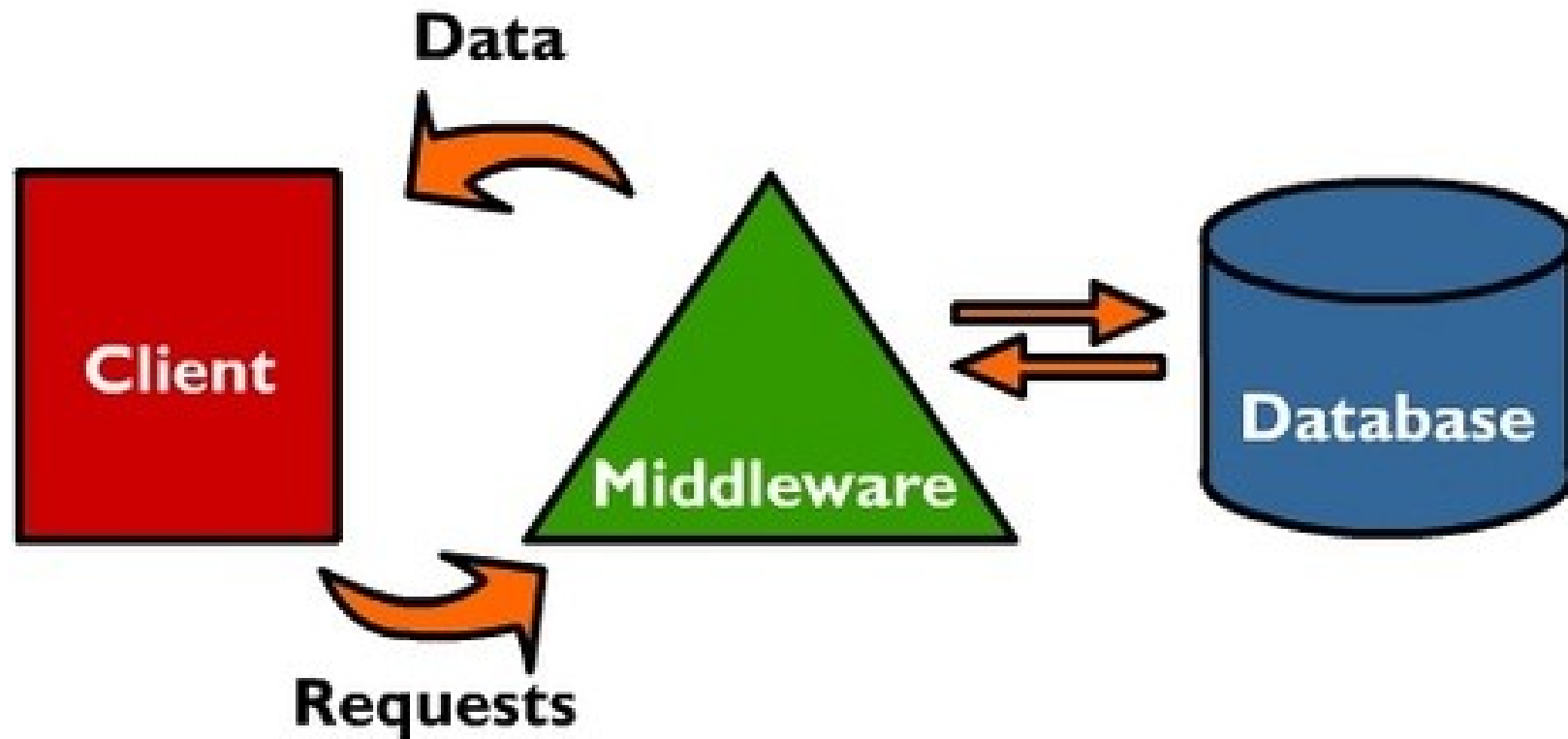
- Tecnologia Middleware JDBC;
- Fundamentos;
- Banco de dados na linguagem Java;
- Uso do NetBeans para gerência do banco JavaDB;
- Servlet para listagem de dados provenientes do banco;

MIDDLEWARE JDBC

- O Que é Middleware?
 - Middle: meio
 - Middleware: software intermediário
 - “Encapsular” complexidades de uma tarefa
 - Middleware ou mediador é um programa de computador que faz a mediação entre software e demais aplicações.

MIDDLEWARE JDBC

- Middleware?
 - Estrutura

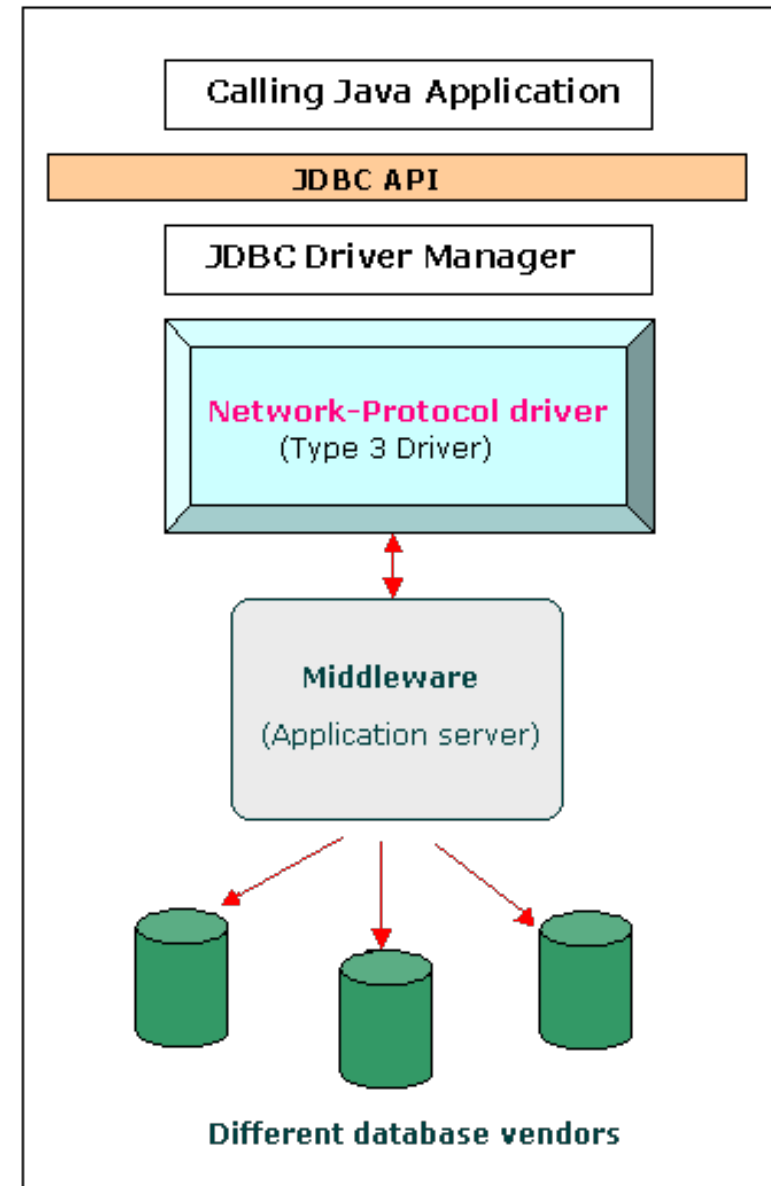


MIDDLEWARE JDBC

- O Que é o Middleware JDBC?
 - JDBC: Java Data Base Connector
 - Função: conectar e se comunicar com SGBD
 - Sistema Gerenciador de Banco de Dados
- Por quê?
 - Programar “na raça” é muito complexo
 - Cada SGDB pode ter detalhes específicos
 - Comandos prontos para tarefas mais comuns
- Resultado:
 - Programação mais simples e uniforme

MIDDLEWARE JDBC

- Middleware JDBC?
 - Estrutura



BANCO DE DADOS: FUNDAMENTOS

- Devido a pluralidade de banco de dados e seus provedores de acesso (drivers de conexão), a SUN criou uma API chamada JDBC (java Data Base Connector) cujo o objetivo é fornecer uma ponte entre a camada do cliente, o driver do fabricante e a fonte de dados.
- Necessária uma interface entre a linguagem Java e outra linguagem que todos os bancos de dados suportem.
 - A linguagem foi o SQL (Structured Query Language).

JDBC: PACOTE JAVA.SQL

- **java.sql.Connection:** Representa uma conexão com um banco de dados
- **java.sql.DriverManager:** Gerencia drivers JDBC usados pela Aplicação
- **java.sql.Statement:** Fornece métodos para o desenvolvedor executar instruções SQL
- **java.sql.PrepareStatement:** Fornece métodos para o desenvolvedor executar instruções SQL pré-compiladas
- **java.sql.ResultSet:** Representa o resultado de uma instrução SQL de Pesquisa

JDBC: PACOTE JAVA.SQL

- Para se obter uma conexão usa-se o método `getConnection` em `DriverManager`
- Objetos da classe *`java.sql.Connection`* representam conexões atuais para o banco de dados
- De posse do objeto que representa a conexão, criamos um objeto da classe *`Statement` ou `PreparedStatement`*, que usamos para executar *queries* SQL através dos seguintes métodos:
 - ***ExecuteQuery***: Utilizada em comandos `SELECT`, retornando o resultado de operações como um objeto `ResultSet`
 - ***ExecuteUpdate***: Utilizada em comandos `INSERT`, `UPDATE` ou `DELETE`, retornando um inteiro que representa o número de colunas afetadas

JDBC: PACOTE JAVA.SQL

- ResultSet
 - são objetos que representam os resultados de uma consulta no banco de dados
 - pode ser visualizado como uma tabela
 - a informação é recuperada um registro (linha) por vez
 - para percorrer as linhas da tabela em um *ResultSet*, usamos o método *next()*

```
while( resultSet.next() ) {  
    // Percorrer a tabela, registro por registro  
}
```

FECHANDO OS OBJETOS: LIBERANDO RECURSOS

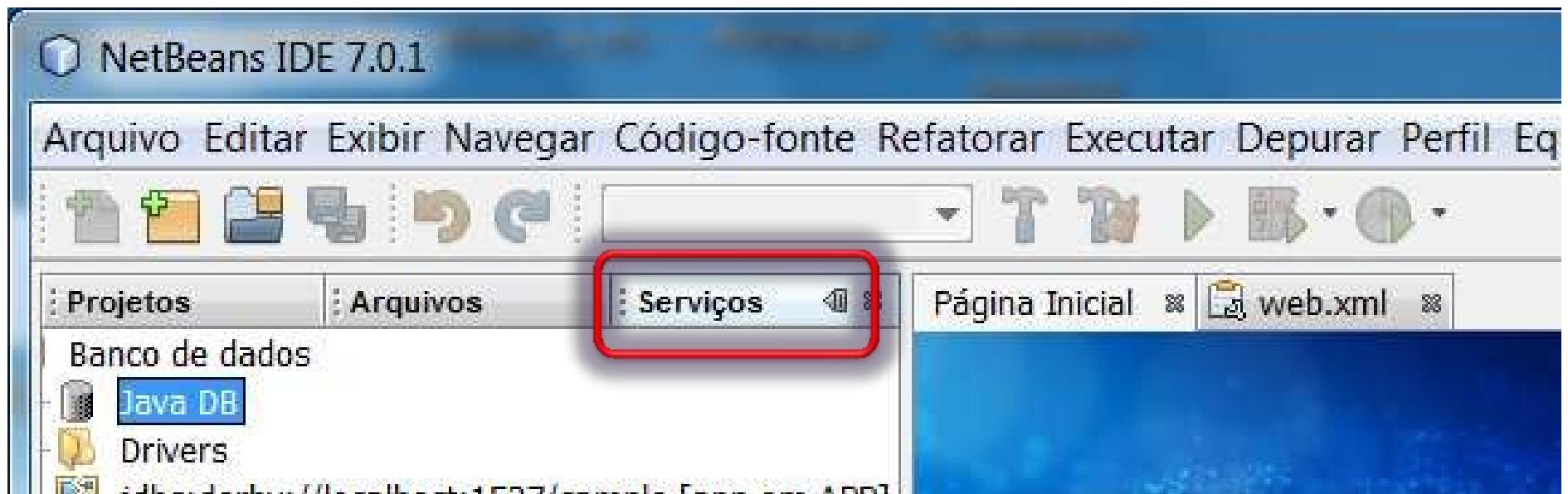
- Este é um passo muito importante que frequentemente é negligenciado após a(s) consulta(s) completada(s).
- Deve ser feita explicitamente e é uma responsabilidade do programador.
- Sem executar tal liberação, os recursos utilizados pela operação não podem ser novamente alocados.
- Para aplicações muito grandes, rapidamente resultaria na perda de conexões disponíveis
- Executa-se chamando o método *close()* disponível em cada objeto das classes:
 - *Connection*
 - *Statement*
 - *ResultSet*
- Existe uma ordem específica envolvida: a ordem inversa de criação
- Recomenda-se a colocação do código dentro de uma cláusula **finally**

JAVADB (DERBY)

- O Que é o JavaDB?
 - Java DB é um SGBD 100% em Java
 - Faz parte do projeto Apache Derby
 - Instalado junto com o suporte Java EE no NetBeans
 - Facilmente utilizável por meio do NetBeans

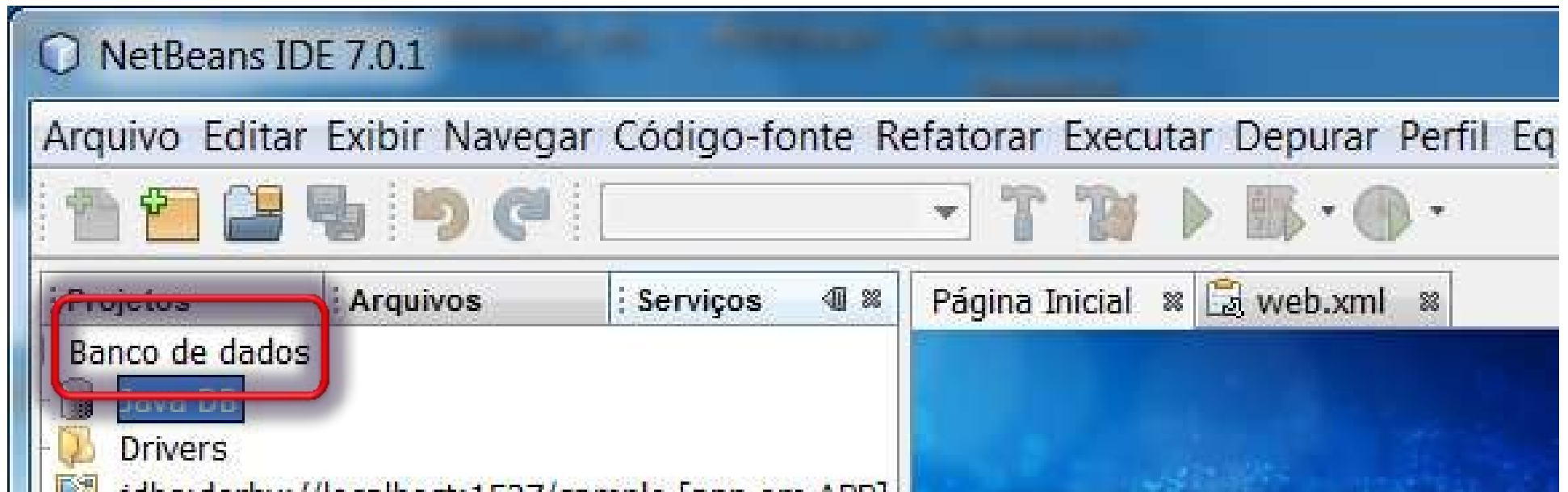
BANCO COM O JAVA DB

- Abra o NetBeans e selecione a “aba” Serviços



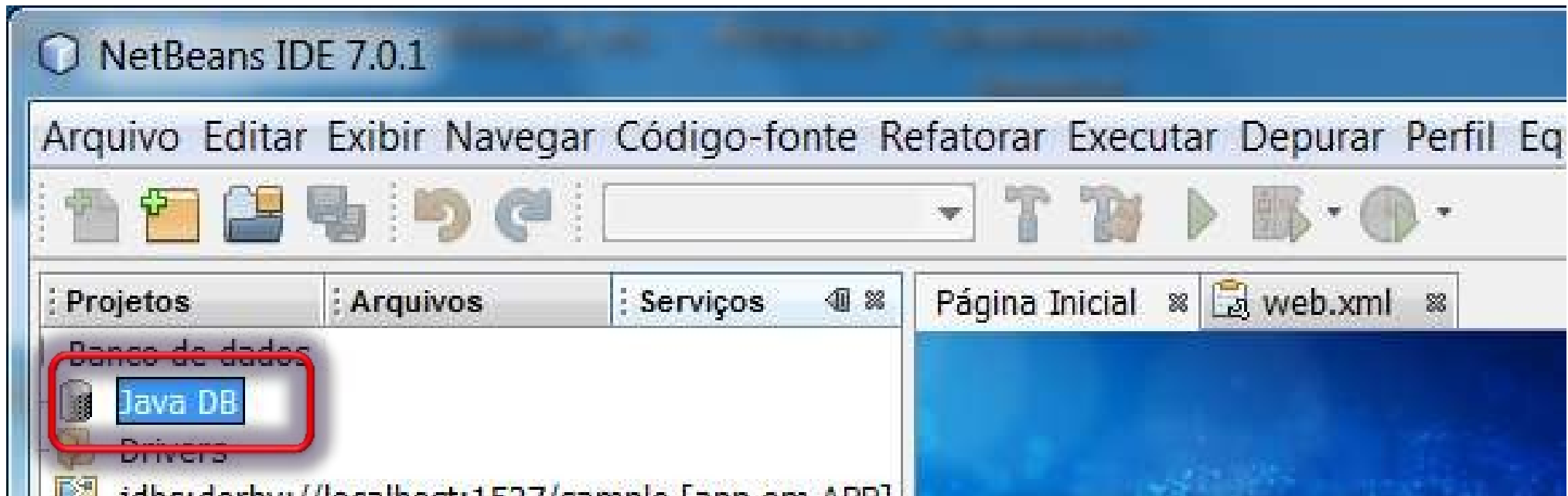
BANCO COM O JAVA DB

- “Abra” a opção “Banco de Dados”



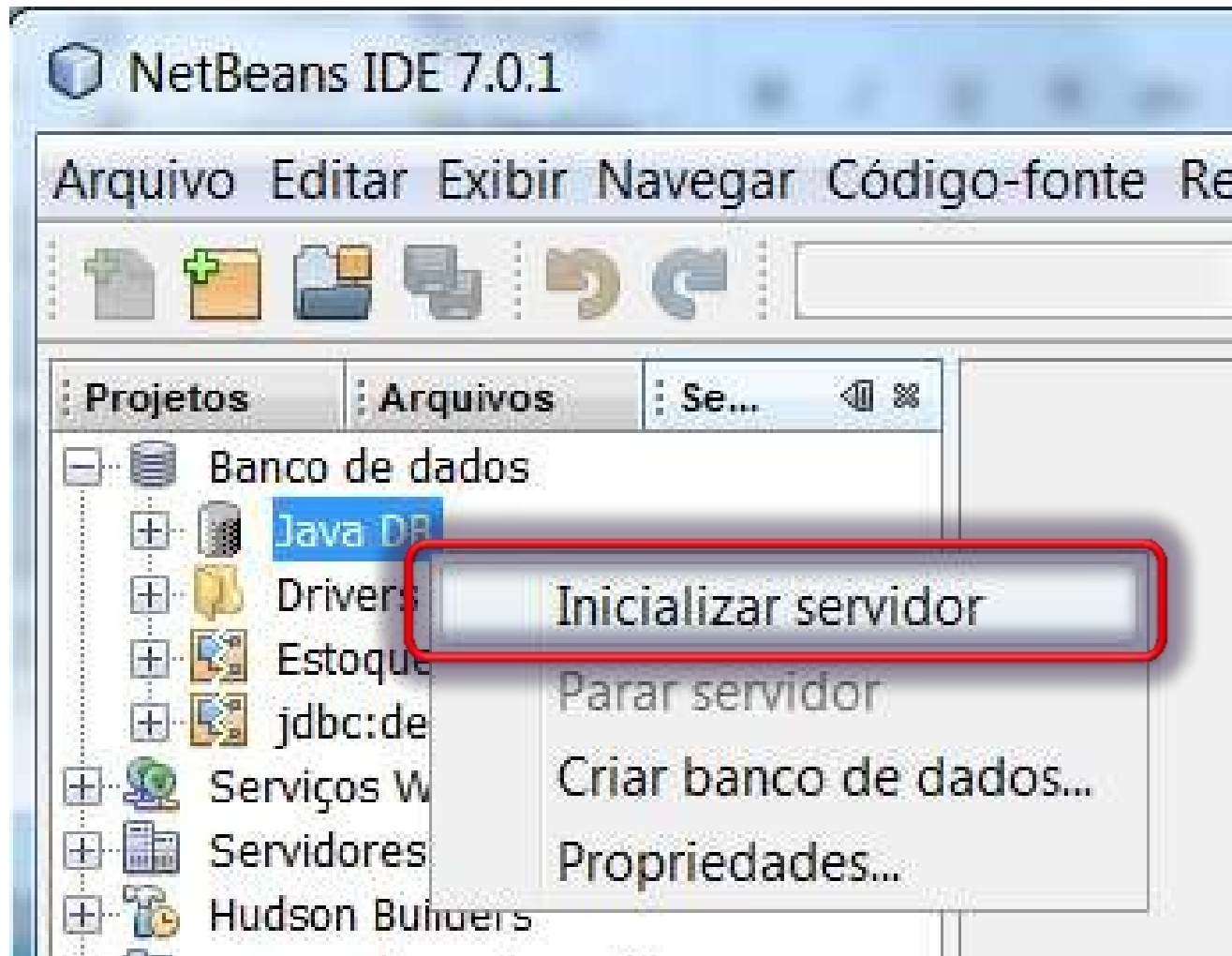
BANCO COM O JAVA DB

- E procure pelo item “Java DB”



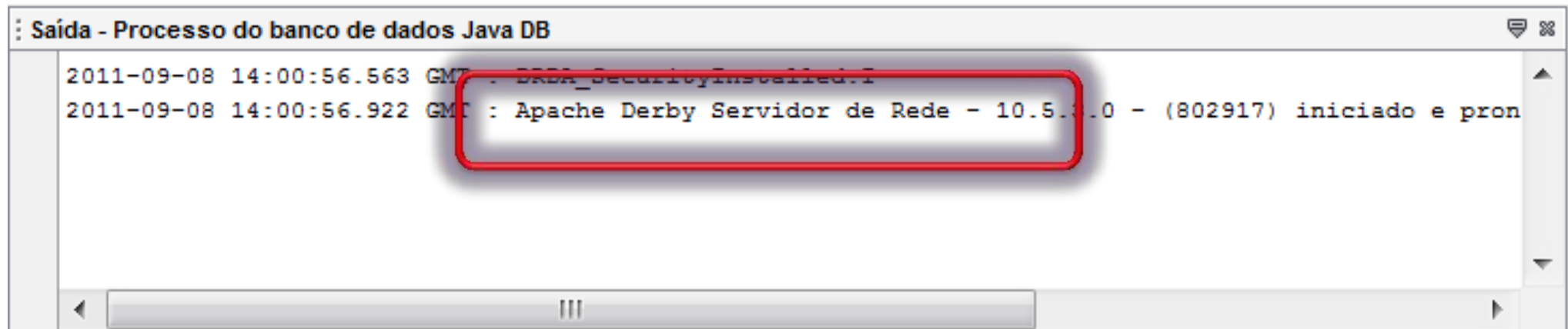
BANCO COM O JAVA DB

- Clique com o botão direito em “Java DB” e selecione “Iniciar servidor”



BANCO COM O JAVA DB

- Observe na área de mensagens, a informação de que o banco de dados foi carregado e está pronto para receber conexões.

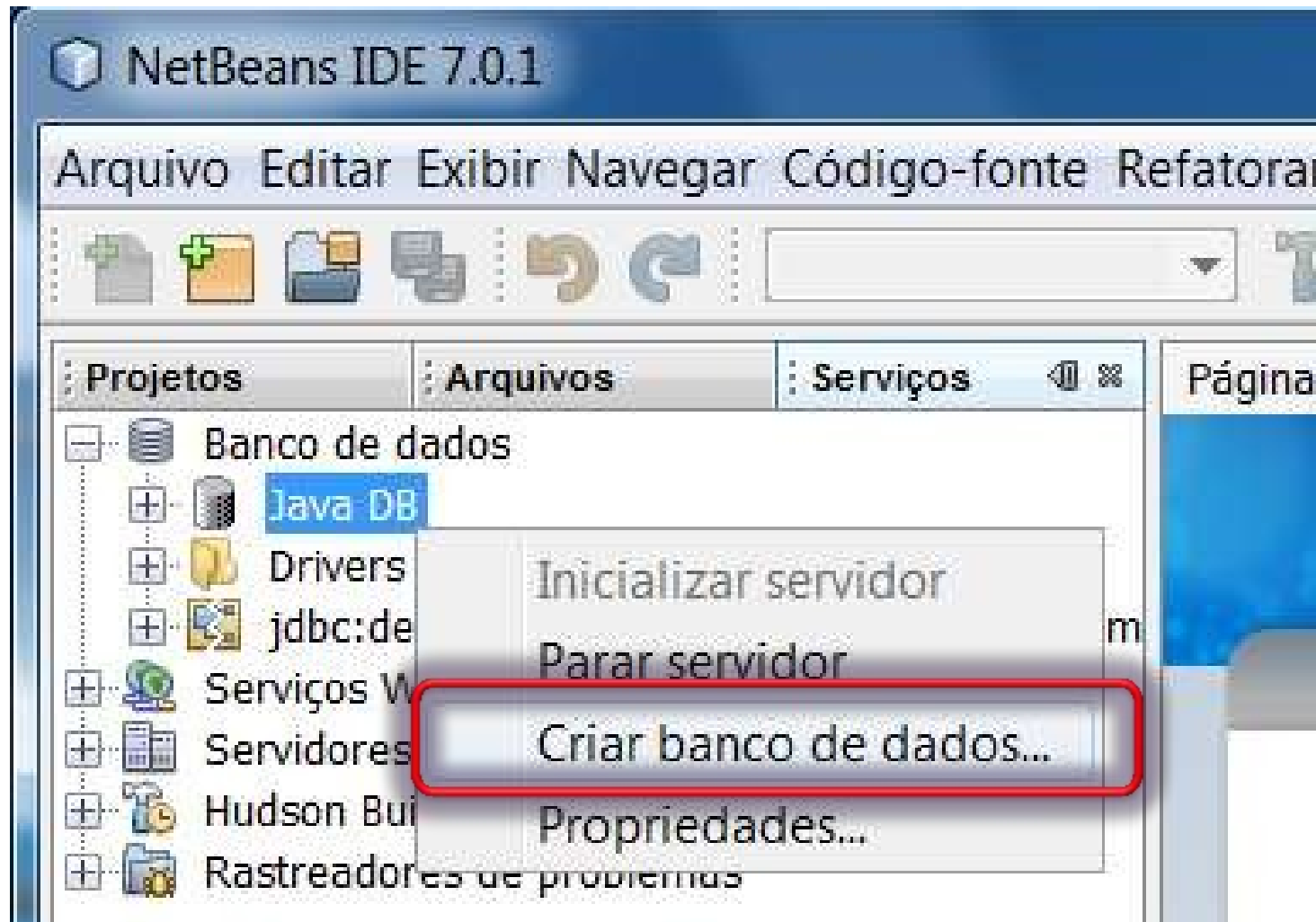


The screenshot shows a console window titled "Saída - Processo do banco de dados Java DB". It contains two lines of log output. The second line, "2011-09-08 14:00:56.922 GMT : Apache Derby Servidor de Rede - 10.5.3.0 - (802917) iniciado e pronto para aceitar conexões.", is highlighted with a red rectangular box. The first line is partially obscured by the box.

```
2011-09-08 14:00:56.563 GMT : DERBY_SecurityInstalled.T
2011-09-08 14:00:56.922 GMT : Apache Derby Servidor de Rede - 10.5.3.0 - (802917) iniciado e pronto para aceitar conexões.
```

BANCO COM O JAVA DB

- Clique com o botão direito em “Java DB” e selecione “Criar banco de dados...”



BANCO COM O JAVA DB

- Na janela, configure o banco e depois clique em “OK”:
 - Nome: estoque
 - Usuário / Senha / Confirmação: nbuser

Criar banco de dados Java DB

Nome do banco de dados: estoque

Nome do usuário: nbuser

Senha: nbuser

Confirmar senha: nbuser

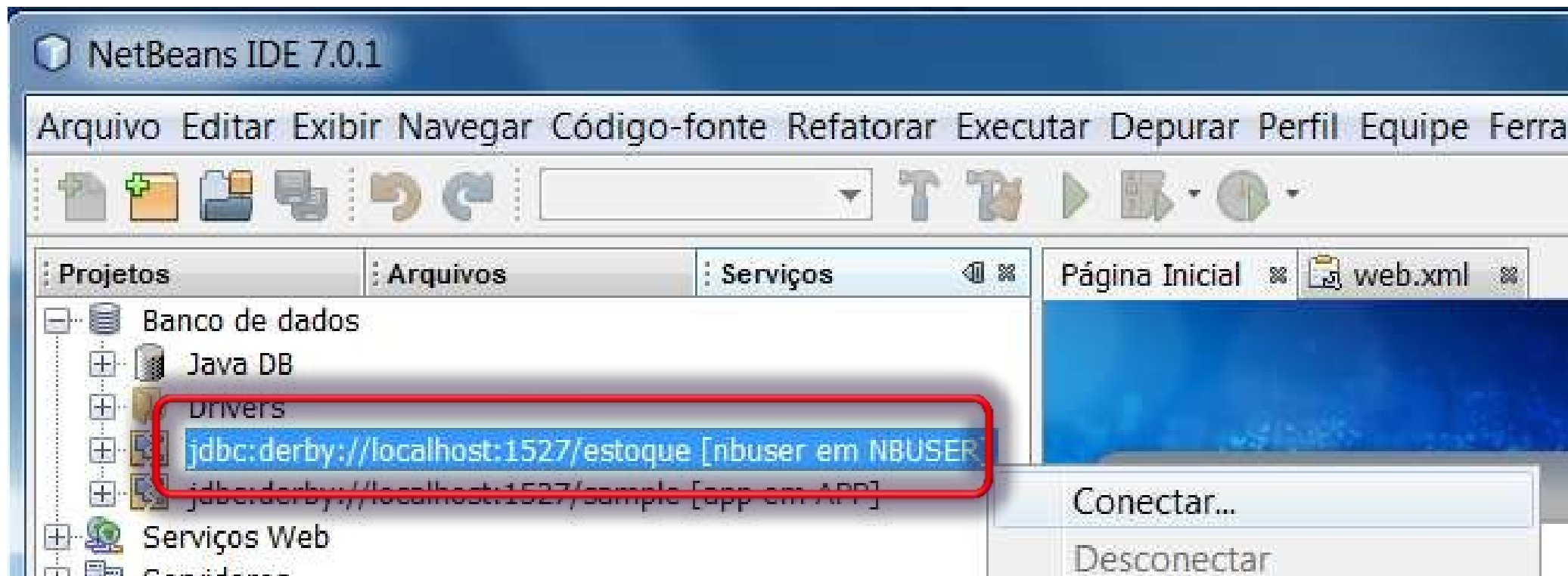
Localização do banco de dados: C:\Users\djcaetano\.netbeans-derby

Propriedades...

OK Cancelar

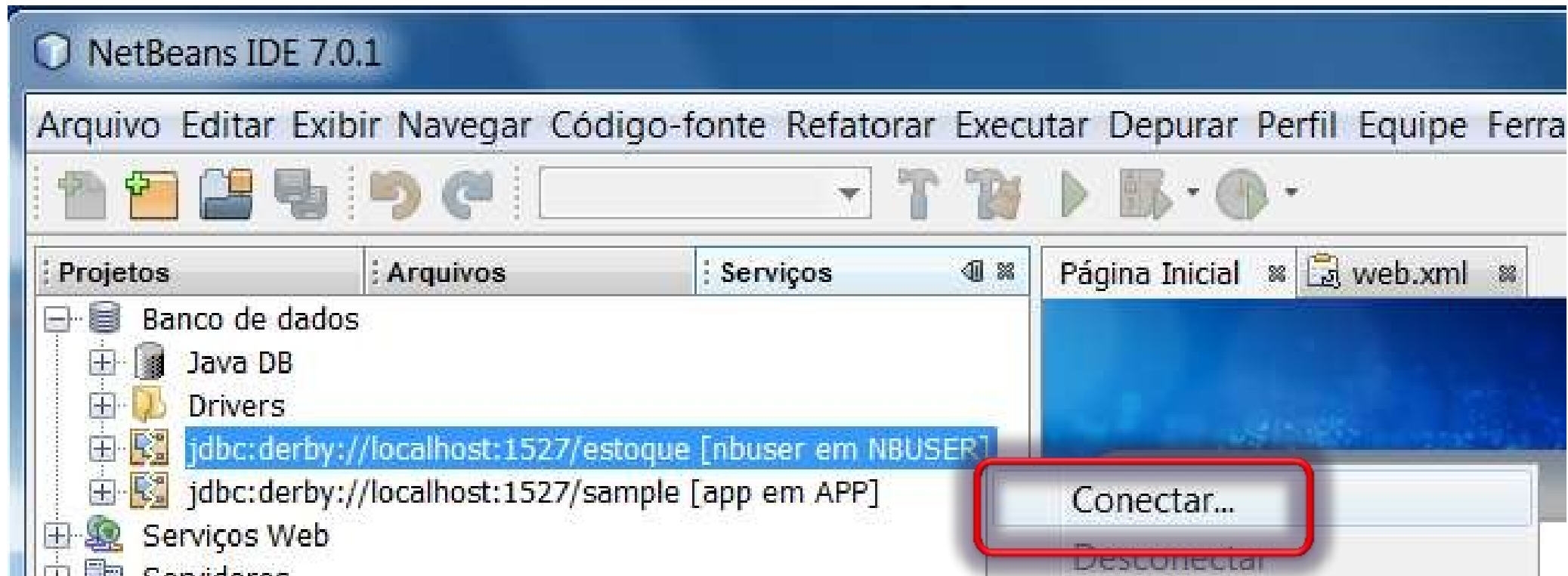
BANCO COM O JAVA DB

- Clique com o botão direito em:
 - jdbc:derby://localhost:1527/estoque



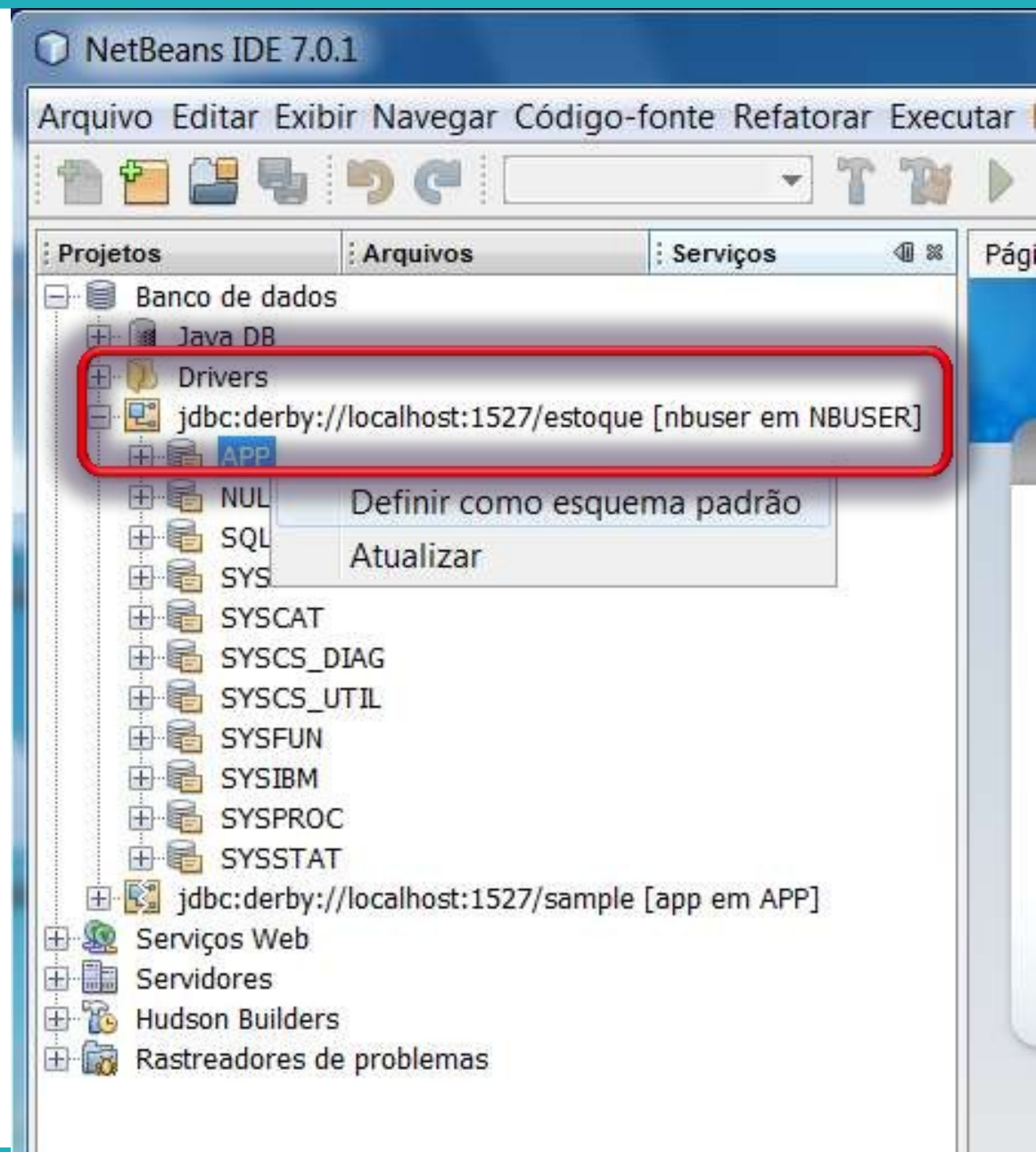
BANCO COM O JAVA DB

- E selecione a opção Conectar...



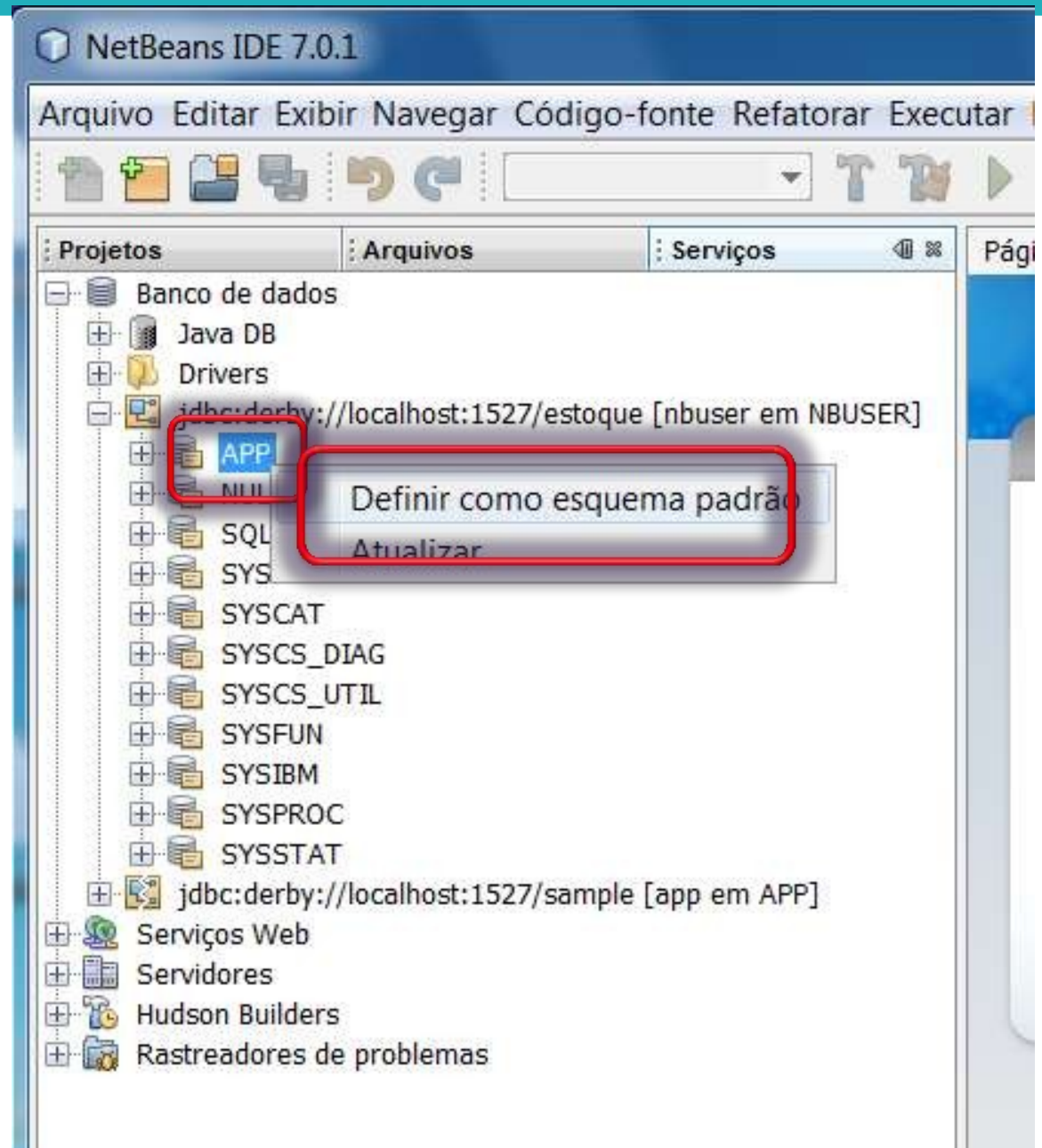
BANCO COM O JAVA DB

- “Expanda” os ícones do banco estoque



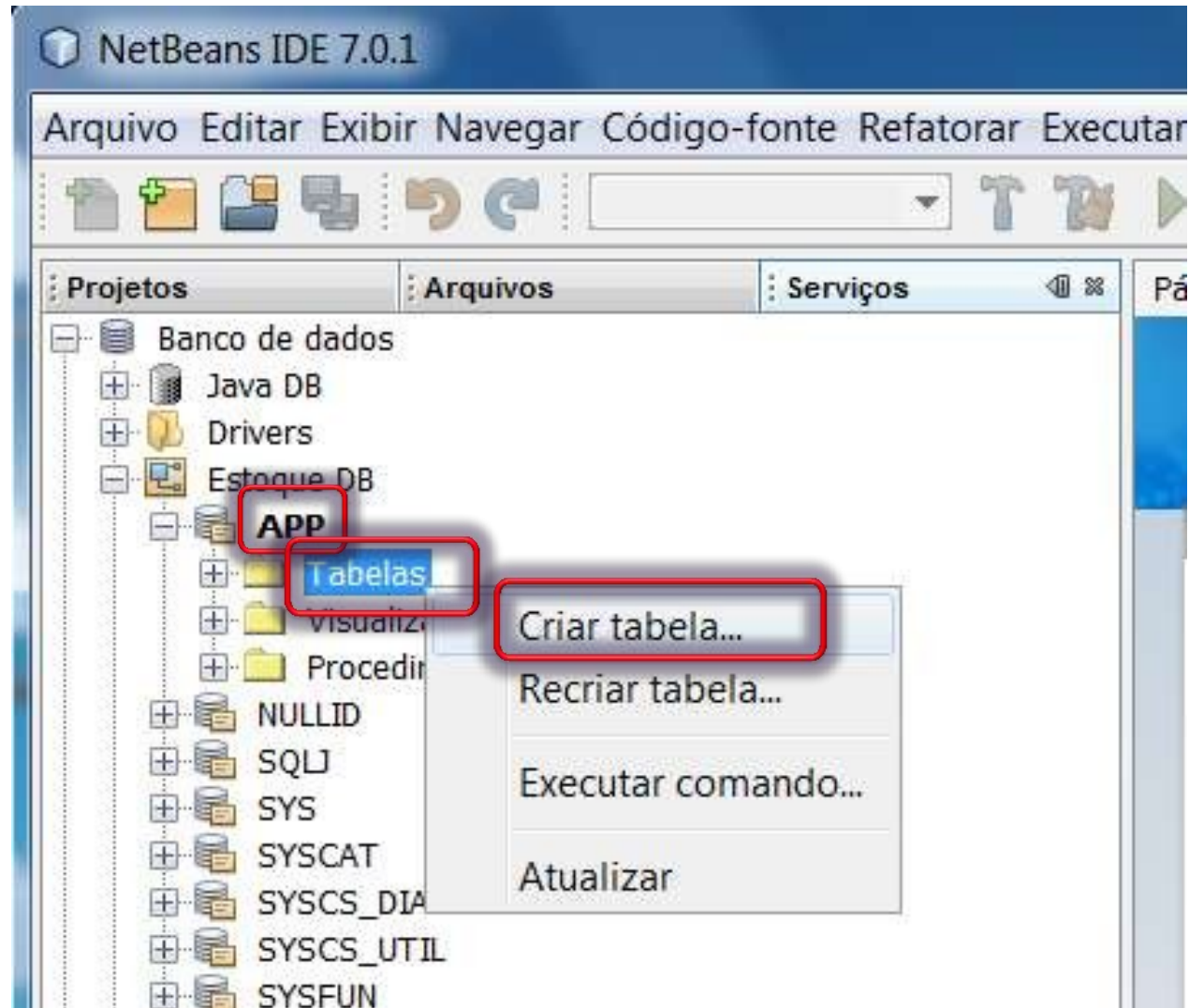
BANCO COM O JAVA DB

- “Clique com o botão direito em APP
- E selecione Definir como esquema padrão



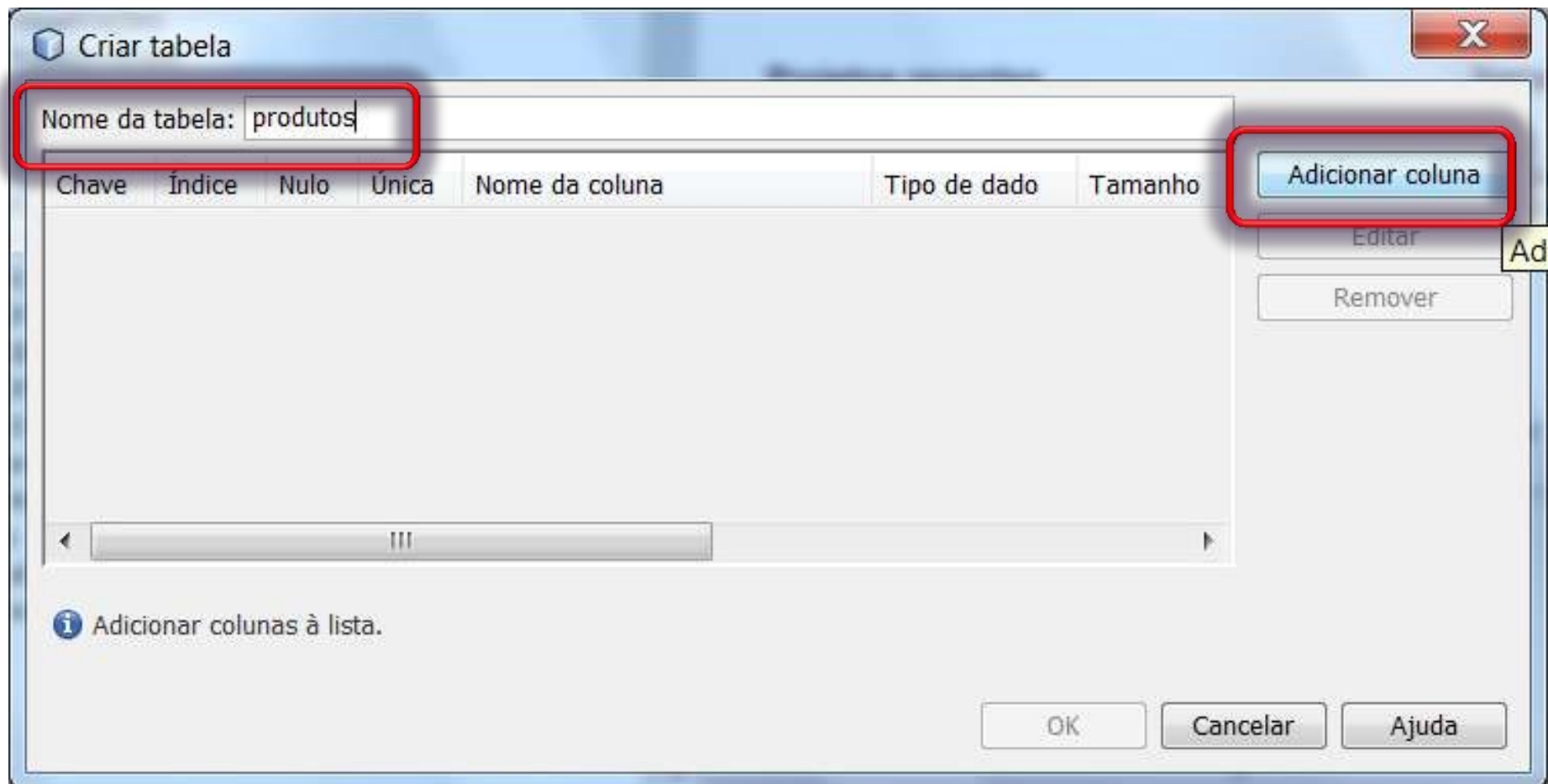
BANCO COM O JAVA DB

- “Expanda” a pasta APP
- Clique com o botão direito em Tabelas
- Selecione “Criar Tabela”



BANCO COM O JAVA DB

- Nessa janela, dê o nome produtos à tabela
- Clique em “Adicionar coluna”



BANCO COM O JAVA DB

- Adicione as colunas da tabela:

Adicionar coluna

Nome: id

Tipo: INTEGER

Tamanho: Escala:

Padrão:

Restrições

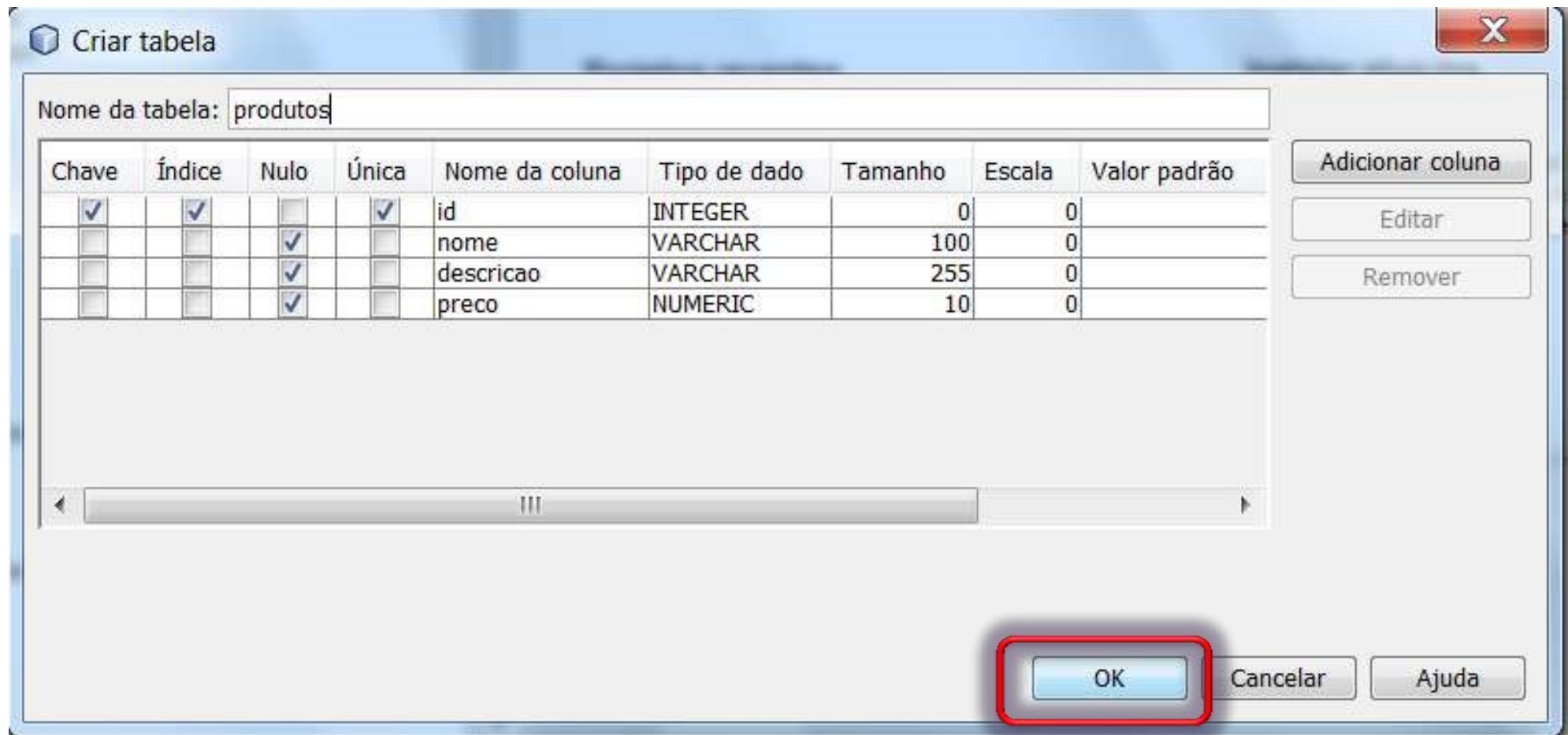
☒ Chave primária ☒ Única ☐ Nulo ☒ Índice

☐ Verificar:

OK Cancelar

BANCO COM O JAVA DB

- Após adicionar todas as colunas, uma de cada vez:

A screenshot of a 'Criar tabela' (Create Table) dialog box. The title bar says 'Criar tabela' with a close button. The 'Nome da tabela:' field contains 'produtos'. Below it is a table with columns: Chave, Índice, Nulo, Única, Nome da coluna, Tipo de dado, Tamanho, Escala, and Valor padrão. The table has four rows: 'id' (INTEGER, 0, 0, primary key, indexed, unique), 'nome' (VARCHAR, 100, 0, not unique), 'descricao' (VARCHAR, 255, 0, not unique), and 'preco' (NUMERIC, 10, 0, not unique). To the right of the table are buttons: 'Adicionar coluna', 'Editar', and 'Remover'. At the bottom right are 'OK', 'Cancelar', and 'Ajuda' buttons. The 'OK' button is highlighted with a red rectangle.

Nome da tabela: produtos

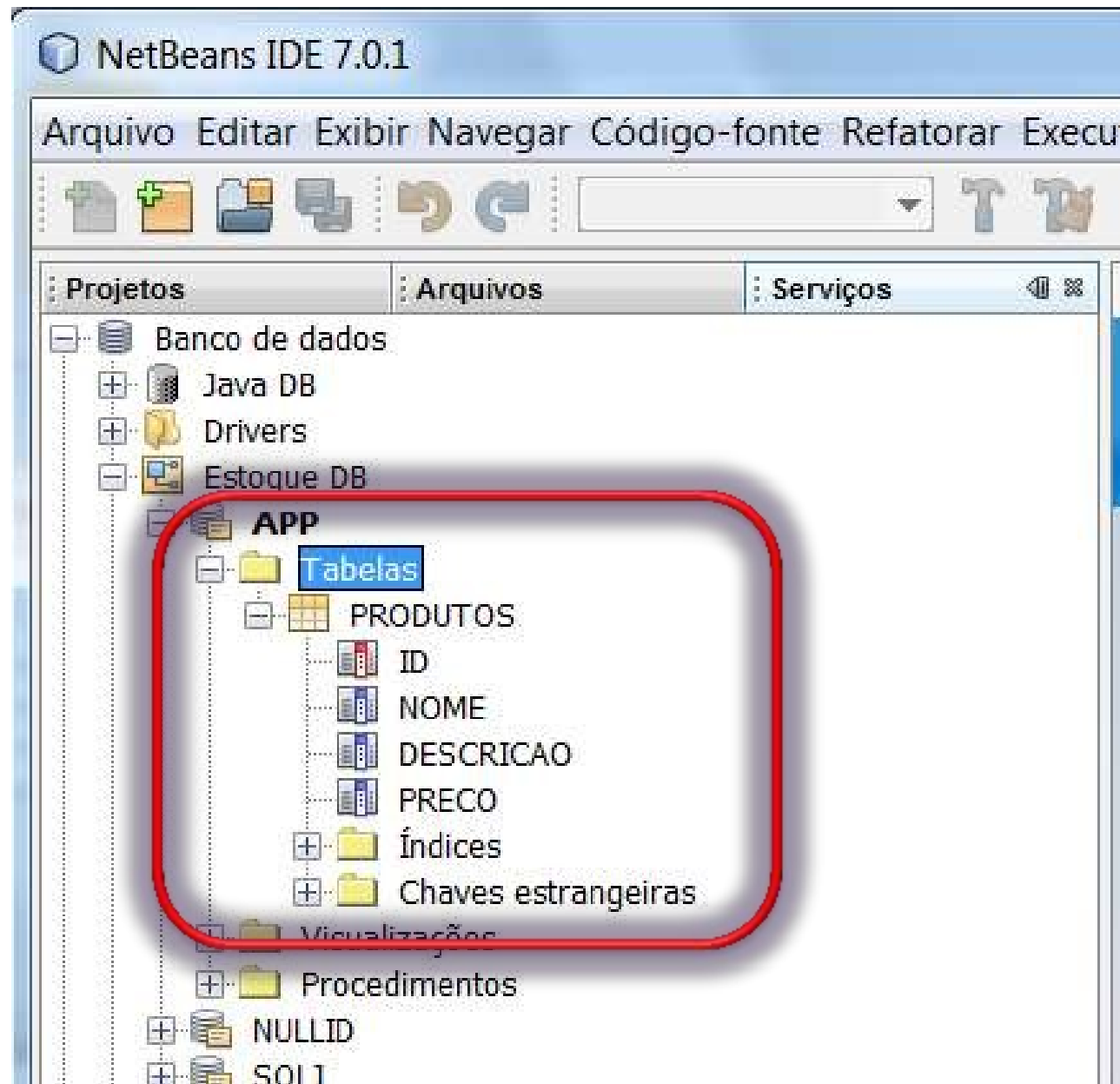
| Chave | Índice | Nulo | Única | Nome da coluna | Tipo de dado | Tamanho | Escala | Valor padrão |
|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|----------------|--------------|---------|--------|--------------|
| <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | id | INTEGER | 0 | 0 | |
| <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | nome | VARCHAR | 100 | 0 | |
| <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | descricao | VARCHAR | 255 | 0 | |
| <input type="checkbox"/> | <input type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/> | preco | NUMERIC | 10 | 0 | |

Adicionar coluna
Editar
Remover

OK Cancelar Ajuda

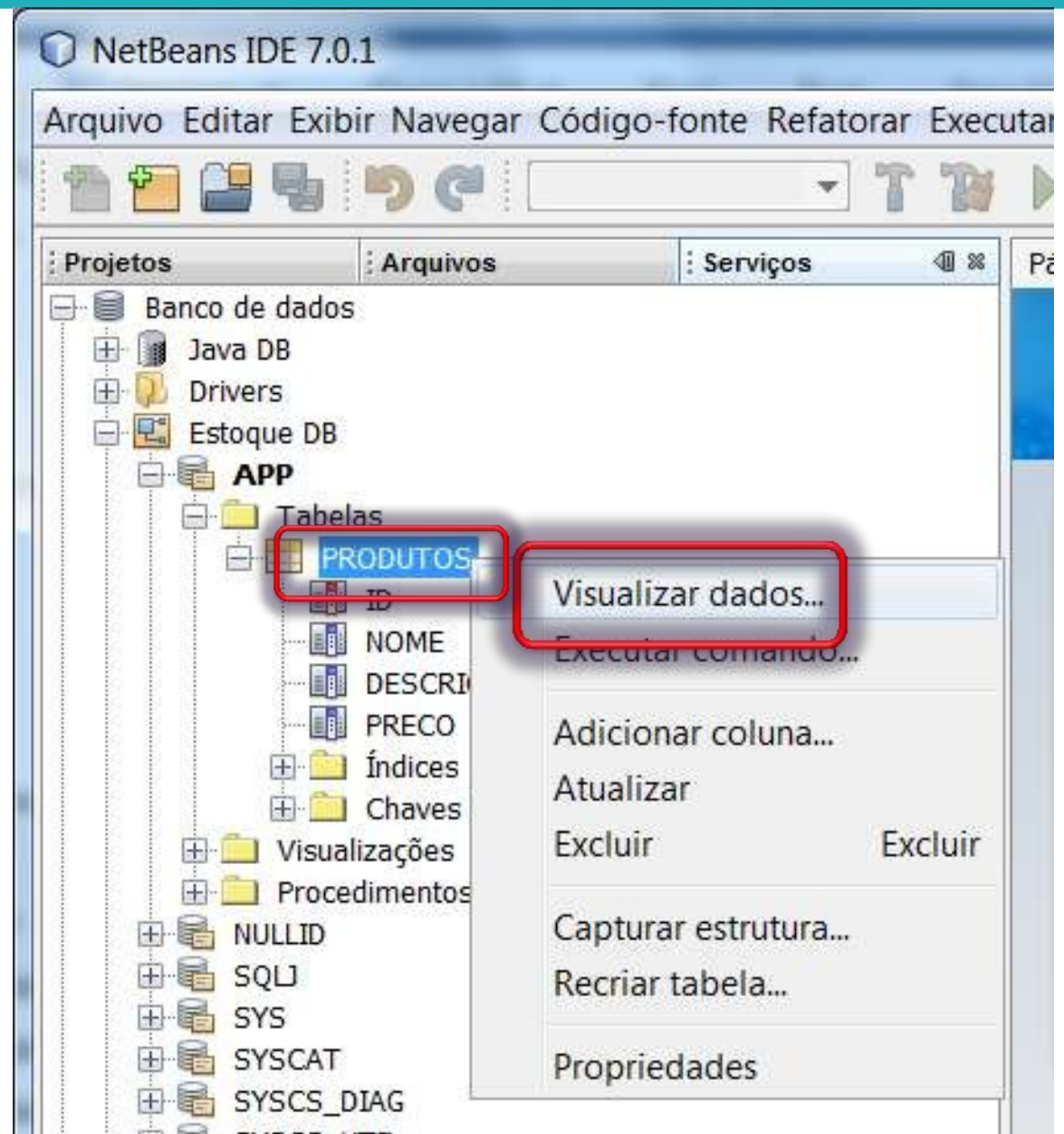
BANCO COM O JAVA DB

- Observe a tabela e colunas criadas



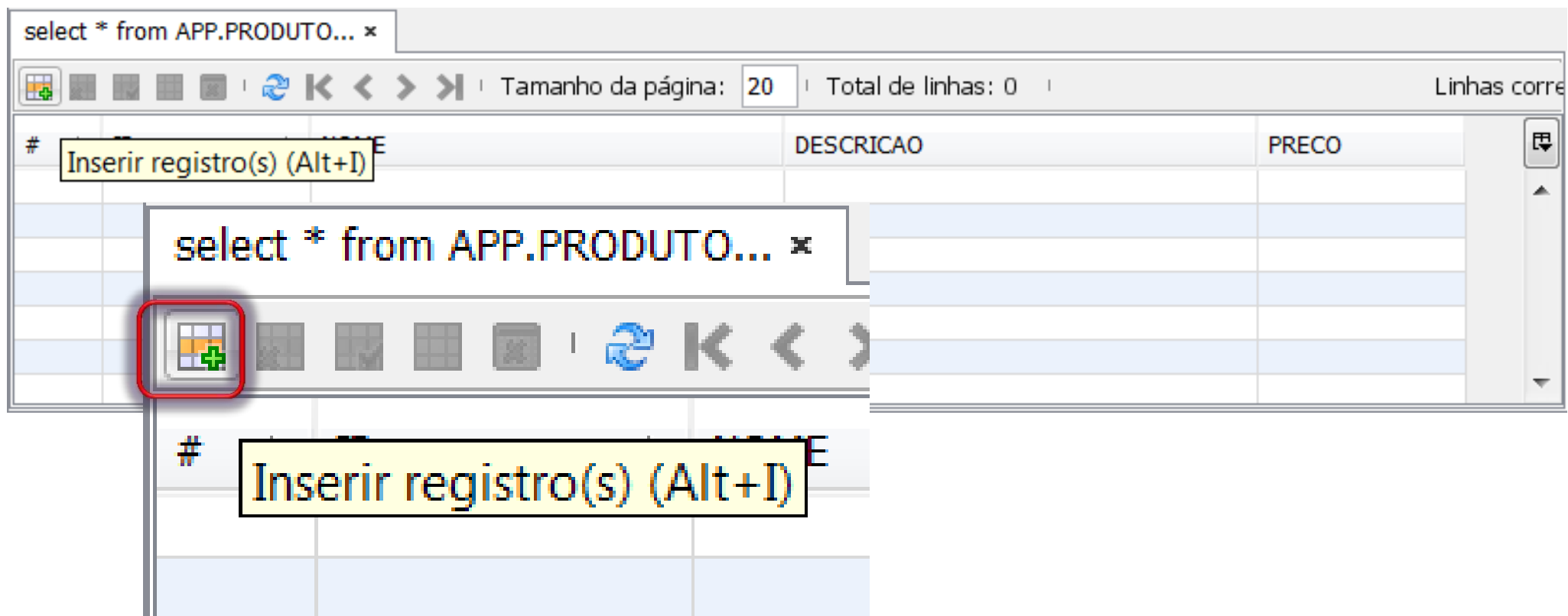
BANCO COM O JAVA DB

- Clique com o botão direito em “PRODUTOS”
- Selecione “Visualizar dados...”



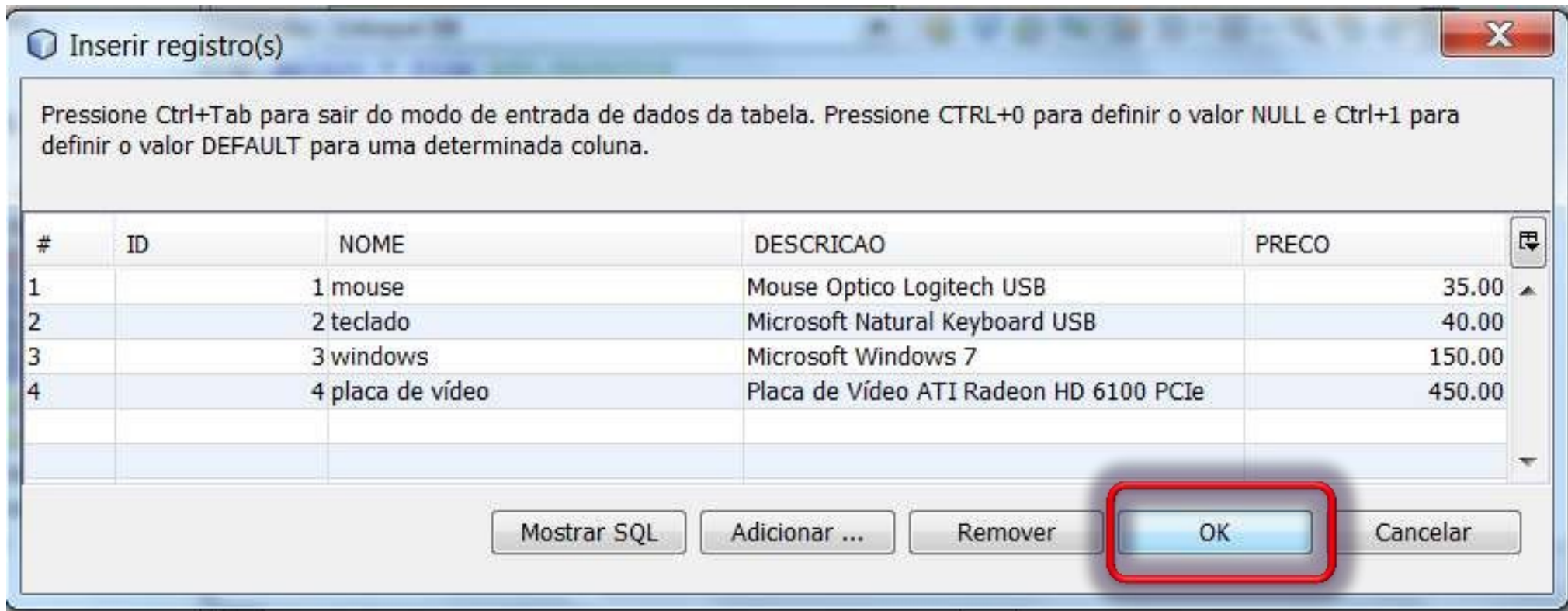
BANCO COM O JAVA DB

- Observe a tabela vazia...
- Clique em “Inserir registro(s)”



BANCO COM O JAVA DB

- Preencha os dados
- Clique em “OK”



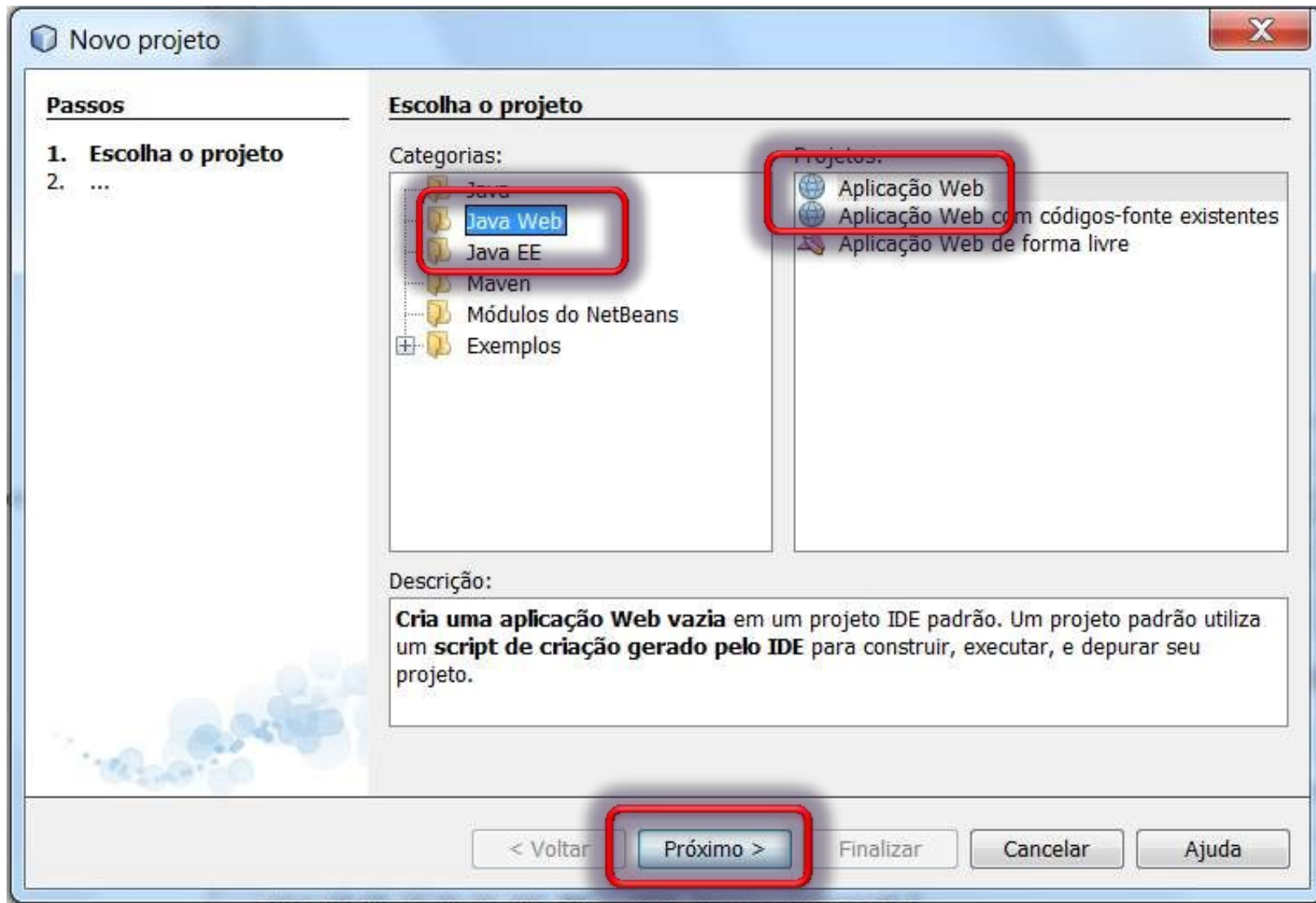
Pressione Ctrl+Tab para sair do modo de entrada de dados da tabela. Pressione CTRL+0 para definir o valor NULL e Ctrl+1 para definir o valor DEFAULT para uma determinada coluna.

| # | ID | NOME | DESCRICAO | PRECO |
|---|----|------------------|--|--------|
| 1 | | 1 mouse | Mouse Optico Logitech USB | 35.00 |
| 2 | | 2 teclado | Microsoft Natural Keyboard USB | 40.00 |
| 3 | | 3 windows | Microsoft Windows 7 | 150.00 |
| 4 | | 4 placa de vídeo | Placa de Vídeo ATI Radeon HD 6100 PCIe | 450.00 |
| | | | | |
| | | | | |

Mostrar SQL Adicionar ... Remover **OK** Cancelar

ACESSANDO O BANCO DE DADOS

- Inicie um novo projeto “Java Web”



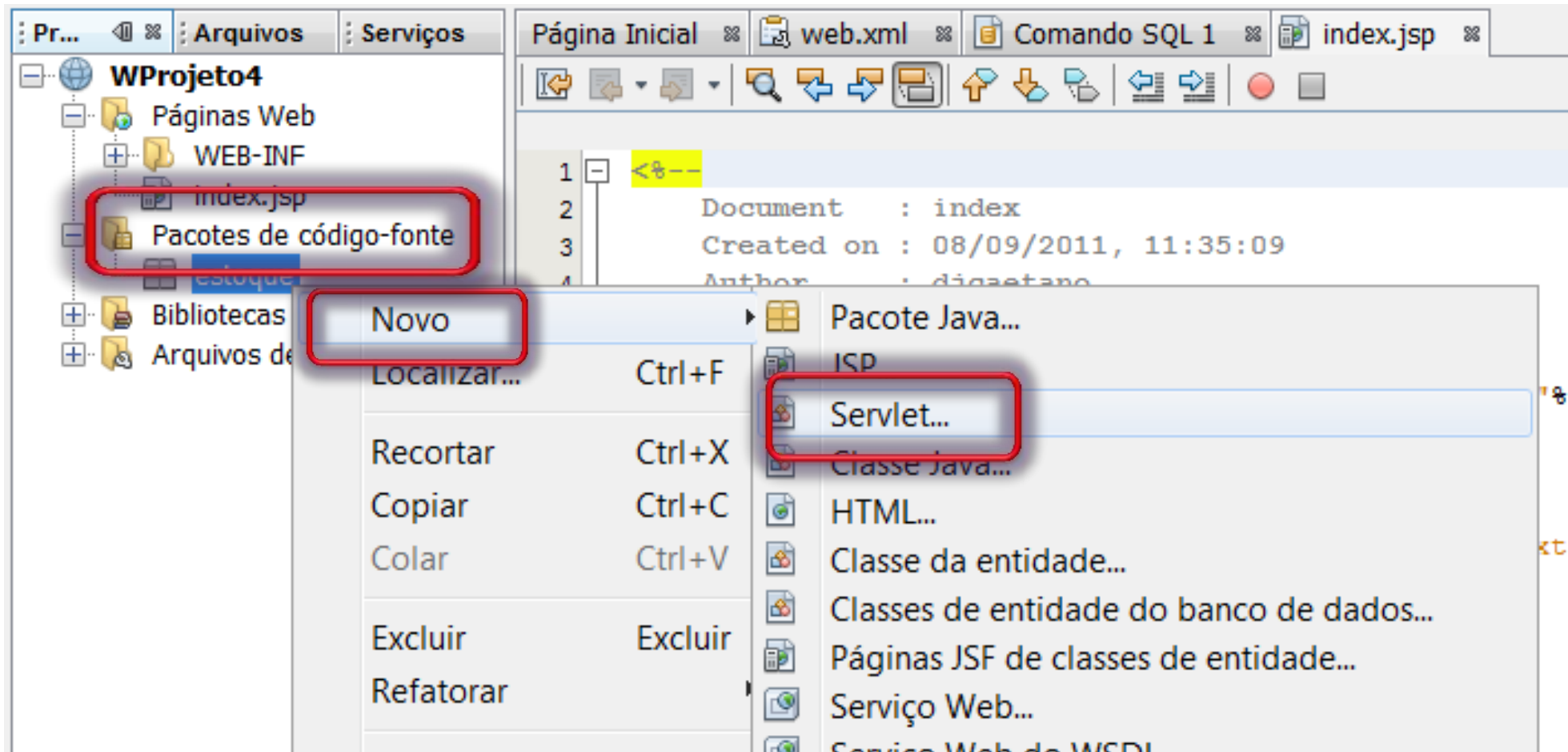
ACESSANDO O BANCO DE DADOS

- Modifique o index.html desta forma

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
    <title>JSP Page</title>
  </head>
  <body>
    <h1><a href="estoque">Estoque</a></h1>
  </body>
</html>
```

ACESSANDO O BANCO DE DADOS

- Crie um Servlet para atender as requisições, com nome “Estoque” mapeado para “/estoque” no web.xml



ACESSANDO O BANCO DE DADOS

- Vamos carregar o driver em memória
- Criar uma conexão para uso futuro

```
try {  
    Class.forName("org.apache.derby.jdbc.ClientDriver");  
    c = DriverManager.getConnection(  
        "jdbc:derby://localhost:1527/estoque", "nbuser","nbuser");  
  
    } catch (ClassNotFoundException ex) {  
        System.err.println("Falta o driver!");  
    } finally{  
        return c;  
    }  
}
```

ACESSANDO O BANCO DE DADOS

- JavaDB:

```
Class.forName("org.apache.derby.jdbc.ClientDriver");  
con = DriverManager.getConnection( "jdbc:derby://localhost:1527/database", "usuario", "senha");
```

- MySQL:

```
Class.forName("com.mysql.jdbc.Driver");  
Con = DriverManager.getConnection( "jdbc:mysql://localhost:3306/database", "usuario", "senha");
```

- PostgreSQL:

```
Class.forName("org.postgresql.Driver");  
con = DriverManager.getConnection( "jdbc:postgresql://localhost:5432/database", "usuario", "senha");
```

- SQLServer:

```
Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");  
con =  
DriverManager.getConnection( "jdbc:sqlserver://localhost:1433;databaseName=database;", "usuario", "senha");
```

ACESSANDO O BANCO DE DADOS

- Vamos preparar uma transação (Statement)
- Executar uma query

```
Statement trans = c.createStatement();
```

```
String query = "SELECT * FROM APP.PRODUTOS";
```

```
ResultSet res = trans.executeQuery(query);
```

ACESSANDO O BANCO DE DADOS

- Vamos agora navegar nos registros ou tuplas retornados

```
while (res.next()){  
    Integer id = res.getInt("id");  
  
    String nome = res.getString("nome");  
  
    String descricao = res.getString("descricao");  
  
    Double preco = res.getDouble("preco");  
  
    Produto p = new Produto(id, nome, descricao,preco);  
  
    produtos.add(p);  
}
```

ACESSANDO O BANCO DE DADOS

- Lembrar sempre de liberar recursos ao final da utilização
 - Obrigação do desenvolvedor

```
// Fechando o ResultSet  
res.close();
```

```
// Fechando a transação  
trans.close();
```

```
// Fechando a conexão  
con.close();
```

EXERCICIOS DE LABORATORIO

- Pratique no Laboratório e em casa
 - Crie um banco de dados no SGBD
 - Crie as tabelas do banco e popule, caso necessário
 - Crie um projeto JavaWeb no Netbeans
 - Manipule os dados da tabela

Até a Próxima Aula!