

N_PROG SIST II_A3 - Texto de apoio

Site: [EAD Mackenzie](#)

Tema: PROGRAMAÇÃO DE SISTEMAS II {TURMA 03B} 2023/1

Livro: N_PROG SIST II_A3 - Texto de apoio

Impresso por: FELIPE BALDIM GUERRA .

Data: quarta, 3 mai 2023, 23:58

Descrição

Índice

1. APLICAÇÕES JAVA COM ACESSO A BANCO DE DADOS

1.1. Configurando o ambiente de desenvolvimento para aplicações com acesso a banco de dados

1. APLICAÇÕES JAVA COM ACESSO A BANCO DE DADOS

Evolução no desenvolvimento de aplicações com acesso a banco de dados

Quando desenvolvemos aplicações comerciais, precisamos armazenar as informações para que possamos realizar consultas futuras. O processo de armazenamento de dados (informações) também é chamado de persistência.

Você viu na Aula 2 que uma forma de persistir os dados das variáveis de um programa é a utilização de arquivos, e que outra forma muito mais eficiente de armazenar as informações é utilizar um banco de dados.

Um banco de dados ou base de dados (BD) é uma coleção de dados ou informações relacionadas entre si, essa é uma característica que distingue a utilização de banco de dados em relação ao uso de arquivos no desenvolvimento de aplicações comerciais. Além disso, uma grande vantagem na utilização de um banco de dados é que os dados e registros contidos em tabelas diferentes podem ser facilmente organizados e recuperados utilizando um software de gestão especializado chamado de Sistema Gerenciador de Banco de Dados (SGBD).

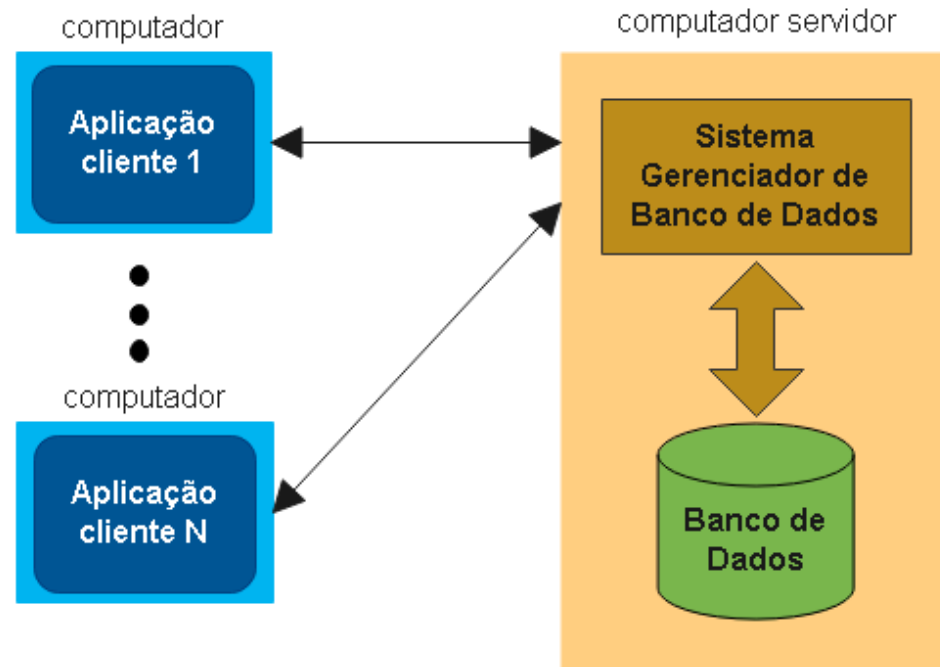
Um SGBD é o conjunto de softwares responsáveis pelo gerenciamento de um banco de dados e seu principal objetivo é retirar da aplicação do cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados.

O SGBD disponibiliza uma interface para que seus clientes possam incluir, alterar ou consultar dados previamente armazenados. A interface é constituída pelas Application Programming Interface (APIs) e drivers do SGBD, que executam comandos da linguagem Structured Query Language (SQL).

A linguagem SQL é a linguagem padrão para trabalhar com bancos de dados e será explicada detalhadamente na disciplina de Banco de Dados que você verá neste semestre. De qualquer forma, se quiser saber um pouquinho mais sobre a linguagem SQL, acesse o link: <<https://becode.com.br/comandos-sql-nao-pode-viver-sem/>>.

Ao longo dos anos, as aplicações comerciais evoluíram na forma como eram desenvolvidas. Na década de 1980, elas eram construídas utilizando a arquitetura cliente-servidor. Nesse modelo, o SGBD ficava em um computador servidor com maior poder de processamento e os clientes executavam um programa que implementava a regra de negócio da aplicação. O programa era escrito em linguagens de programação da época, como Delphi, Visual Basic, C/C++ etc. A maior desvantagem nesse tipo de modelo era manter atualizada as versões dos softwares nos computadores dos clientes (possivelmente centenas). Os sistemas maiores ficavam desatualizados com frequência, já que parte predominante da aplicação era executada no cliente.

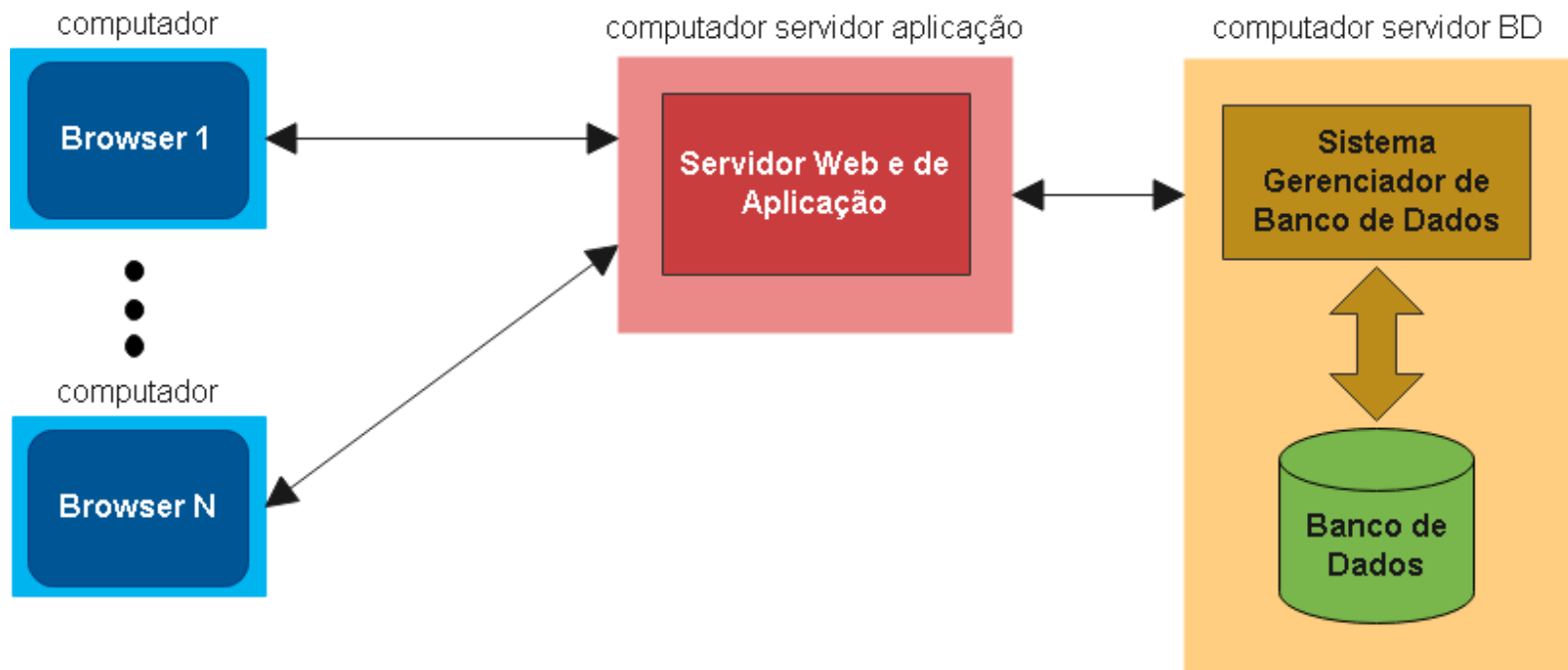
Figura 1 – Arquitetura cliente-servidor



Fonte: Elaborada pelo autor.

Na década de 1990, surgiu a ideia de concentrar a aplicação em uma única máquina nomeada servidor de aplicação, que recebia as solicitações dos clientes e as direcionava para o SGBD, deixando no cliente apenas a parte gráfica da aplicação (telas). Essa solução foi denominada “arquitetura em três camadas” (three tiered architecture) que, além de combinar todas as vantagens da arquitetura cliente-servidor, eliminou a principal desvantagem que era a atualização dos softwares nas máquinas dos clientes. A disponibilização da Internet para empresas comerciais no final da década de 1990 propiciou uma evolução da arquitetura de forma que a aplicação do cliente ficasse armazenada em um servidor Web que também faz o papel do servidor de aplicação, e o único software necessário no cliente é um navegador (browser). Assim, o problema de atualização de software nos clientes praticamente desapareceu (exceto por incompatibilidade entre os navegadores) e, com isso, o desenvolvimento de aplicações Web se tornou um padrão para o desenvolvimento de aplicações comerciais.

Figura 2 – Arquitetura em três camadas



Fonte: Elaborada pelo autor.

No desenvolvimento de aplicações Web, você codificará regras de negócio bastante complexas (requisitos funcionais). Além disso, existem outros requisitos que precisam ser atingidos no desenvolvimento da aplicação: persistência em banco de dados, transação, acesso remoto, Web Services, gerenciamento de threads, gerenciamento de conexões HTTP, cache de objetos, gerenciamento da sessão web, balanceamento de carga, entre outros. Esses requisitos são chamados de não funcionais. Se você tivesse que, além de implementar os requisitos funcionais, implementar os requisitos não funcionais, com certeza teria muito mais trabalho a fazer.

Para simplificar nossa vida, a Sun Microsystems (atualmente essa empresa faz parte da Oracle) criou uma série de especificações (Java EE) que, quando implementadas, podem ser usadas por desenvolvedores para tirar proveito e reutilizar toda essa infraestrutura já pronta.

No Java EE (Java Enterprise Edition), as especificações são escritas de forma detalhada, dando um passo a passo de como deve ser implementado um software que atende cada um dos requisitos não funcionais descrito acima.

Dito isso, como poderíamos então fazer o "download do Java EE"?

Não dá! O Java EE é apenas um grande PDF, um documento, detalhando como deve ser implementado um software para atender às especificações do Java EE.

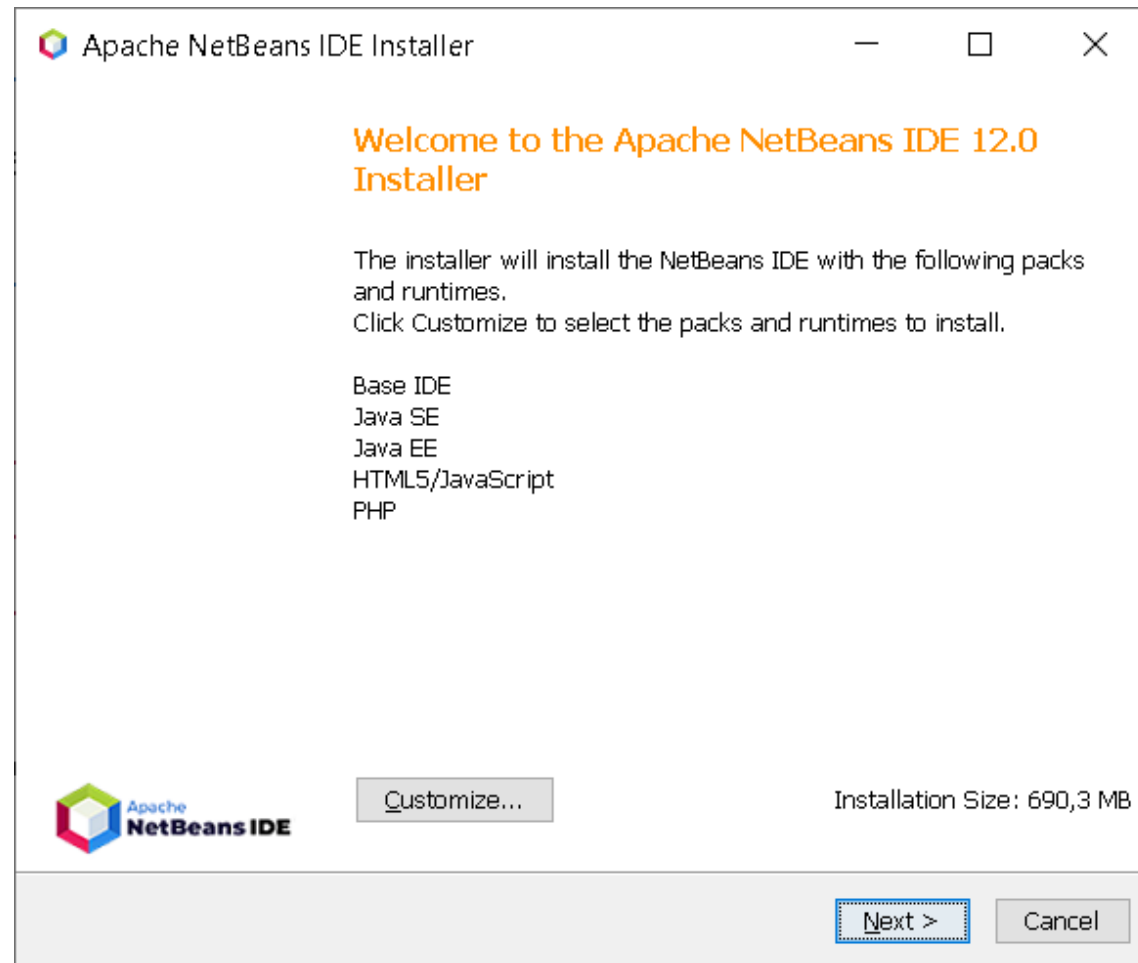
Existem diversas implementações de servidores de aplicação Web que atendem às especificações do Java EE, entre elas: Sun/GlassFish, RedHat/JBoss, Apache, IBM/WebSphere.

Utilizaremos o servidor de aplicação Web desenvolvido pela própria Sun (Oracle), o Glassfish, que é open source e gratuito. Em nosso desenvolvimento de aplicações Web, o Glassfish terá o papel de servir sua aplicação, disponibilizando um ambiente para o desenvolvimento e a execução de aplicações, centralizando e dispensando a instalação em computadores de clientes, para que você apenas se preocupe com as regras de negócio do problema.

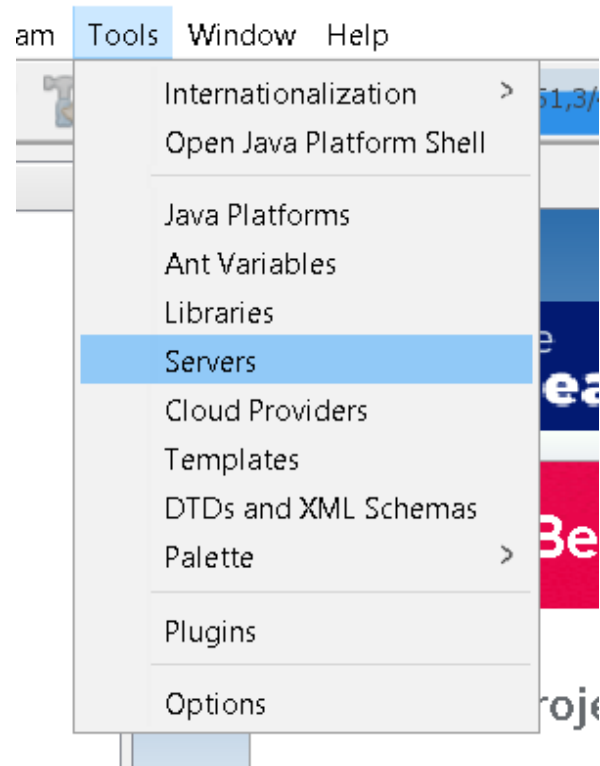
1.1. Configurando o ambiente de desenvolvimento para aplicações com acesso a banco de dados

Para facilitar o desenvolvimento de aplicações Web, trabalharemos com um Integrated Development Environment (IDE) ou Ambiente de Desenvolvimento Integrado que permita a integração com um servidor de aplicação GlassFish. O IDE NetBeans 12.0 LTS oferece excelente suporte para o desenvolvimento de aplicações Web integrado com o servidor GlassFish. Para fazer o download do NetBeans, acesse: <<https://netbeans.apache.org/download/nb120/nb120.html>>.

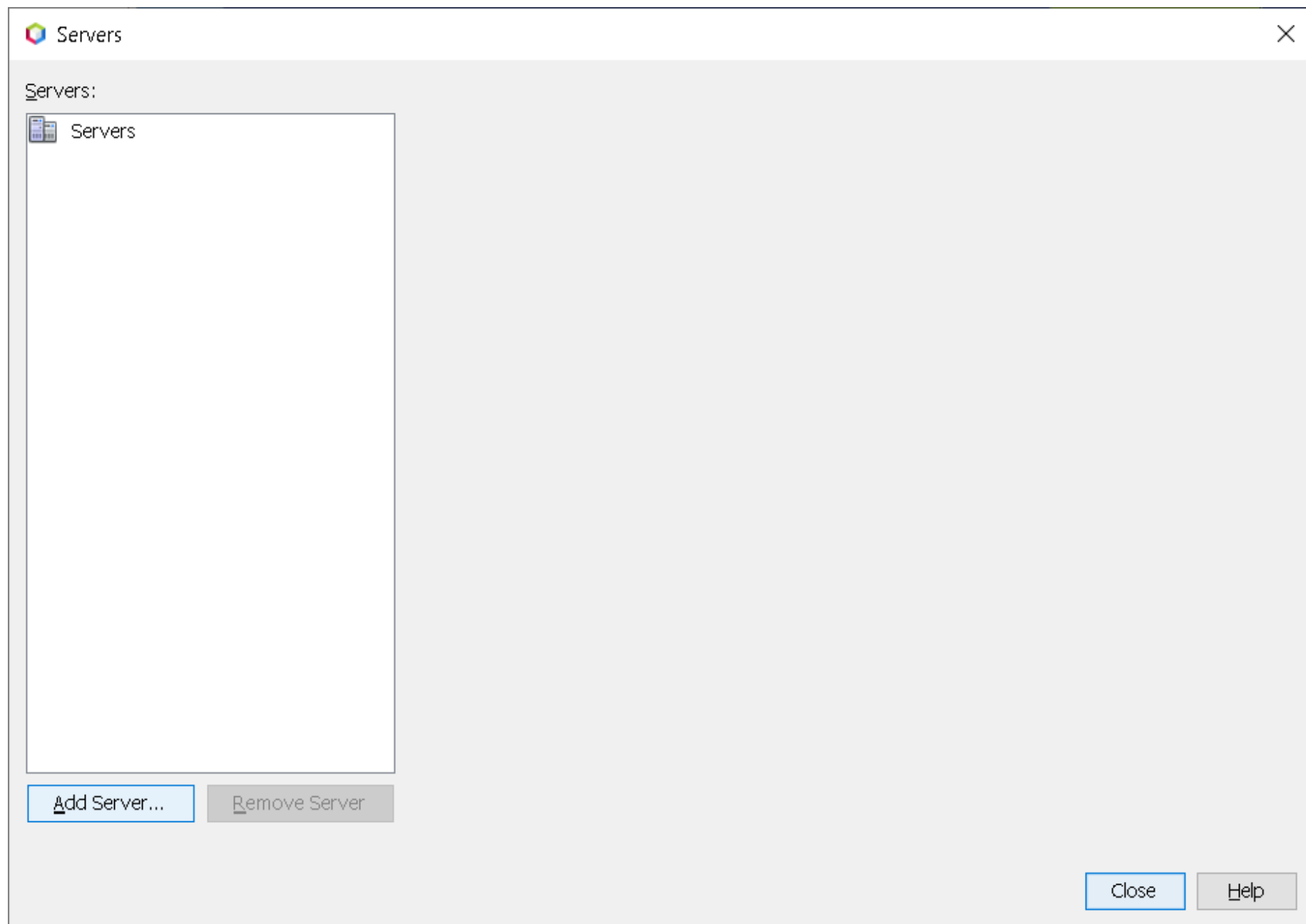
A partir dessa página, você pode selecionar qual instalador é o mais adequado em função de seu sistema operacional (Windows, Linux ou Mac). Como estou usando o Windows 10 de 64 bits, baixei o arquivo Apache-NetBeans-12.0-bin-windows-x64.exe com aproximadamente 366 MB. Clicando no executável, temos a tela abaixo; perceba que a instalação consumirá 630 MB do disco rígido. Clique em Next e siga os passos do instalador.



Após a instalação do NetBeans 12.0 LTS, é necessário instalar e configurar o servidor de aplicação GlassFish. Nesse caso, utilizaremos a versão 5.1 (<https://javaee.github.io/glassfish/>). No NetBeans, basta clicar na barra de menu e acessar a opção Tools à Servers:



Em seguida, após aparecer a janela abaixo, selecione a botão Add Server:



Selecione o servidor GlassFish Server e clique em Next.

Add Server Instance

Steps

1. **Choose Server**
2. ...

Choose Server

Server: Apache Tomcat or TomEE
GlassFish Server
Payara Server
WildFly Application Server

Name: GlassFish Server

< Back Next > Finish Cancel Help

Na próxima janela, é definida a pasta na qual será instalado o GlassFish, veja a caixa de texto Installation Location; no meu caso, instalei em C:\Users\fabio\GlassFish_Server. A versão selecionada deve ser a GlassFish Server 5.1.0 e, então, selecione o checkbox referente à leitura e aceite o contrato de licença (I have read and accept the license agreement). Em seguida, faça o download do GlassFish Server, clicando no botão Download Now. Veja a próxima janela:

Add Server Instance

×

Steps

1. Choose Server

2. **Server Location**

3. Domain Name/Location

Server Location

Installation Location:

C:\Users\fabio\GlassFish_Server

Browse...

☒ Local Domain

☐ Remote Domain

Choose server to download:

GlassFish Server 5.1.0

▼

Download Now...

☒ I have read and accept the license agreement... (click)

C:\Users\fabio\GlassFish_Server does not exist but server can be downloaded and installed there.

< Back

Next >

Finish

Cancel

Help

Após a realização do download e a instalação do GlassFish, clique no botão Next para ter acesso à janela e, depois, clique no botão Finish habilitado para finalizar o processo:

Steps

1. Choose Server
2. Server Location
- 3. Domain Name/Location**

Domain Location

Domain: ▾

Host: ▾ ☒ Loopback

DAS Port: HTTP Port: ☒ Default

Target:

User Name:

Password:

 Register existing embedded domain: domain1

< Back

Next >

Finish

Cancel

Help

Por fim, você terá a tela final já com o GlassFish Server configurado.

The screenshot shows the 'Servers' configuration window in NetBeans. On the left, a tree view shows 'Servers' and 'GlassFish Server'. The main area is divided into 'Common' and 'Java' tabs. The 'Common' tab is active, showing fields for 'Server Name' (GlassFish Server), 'Server Type' (GlassFish Server 5.1.0), 'Installation Location' (C:\Users\fabio\GlassFish_Server\glassfish), 'Domains Folder' (C:\Users\fabio\GlassFish_Server\glassfish\domains), 'Host' (localhost), 'DAS Port' (4848), 'HTTP Port' (8080), 'Domain' (domain1), 'Target', 'User Name', and 'Password'. There are also checkboxes for 'Enable Comet Support', 'Enable HTTP Monitor', 'Enable JDBC Driver Deployment', 'Show Password in this Form', 'Preserve Sessions Across Redeployment', and 'Start Registered Derby Server'. At the bottom, there are buttons for 'Add Server...', 'Remove Server', 'Close', and 'Help'.

Servers

Servers:

- Servers
- GlassFish Server

Server Name: GlassFish Server

Server Type: GlassFish Server 5.1.0

Common Java

Installation Location: C:\Users\fabio\GlassFish_Server\glassfish

Domains Folder: C:\Users\fabio\GlassFish_Server\glassfish\domains

Host: localhost ☐ Loopback

DAS Port: 4848 HTTP Port: 8080

Domain: domain1 Target:

User Name: Password:

☐ Enable Comet Support ☐ Show Password in this Form

☐ Enable HTTP Monitor ☒ Preserve Sessions Across Redeployment

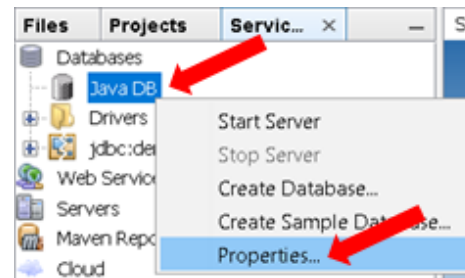
☒ Enable JDBC Driver Deployment ☒ Start Registered Derby Server

Add Server... Remove Server

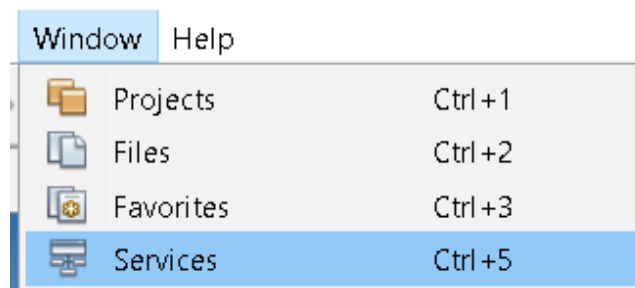
Close Help

Agora, podemos criar um banco de dados para começarmos a desenvolver nossa aplicação Web. Para tanto, utilizaremos o Java DB (Apache Derby) que é um servidor de banco de dados encapsulado no servidor de aplicações GlassFish já instalado.

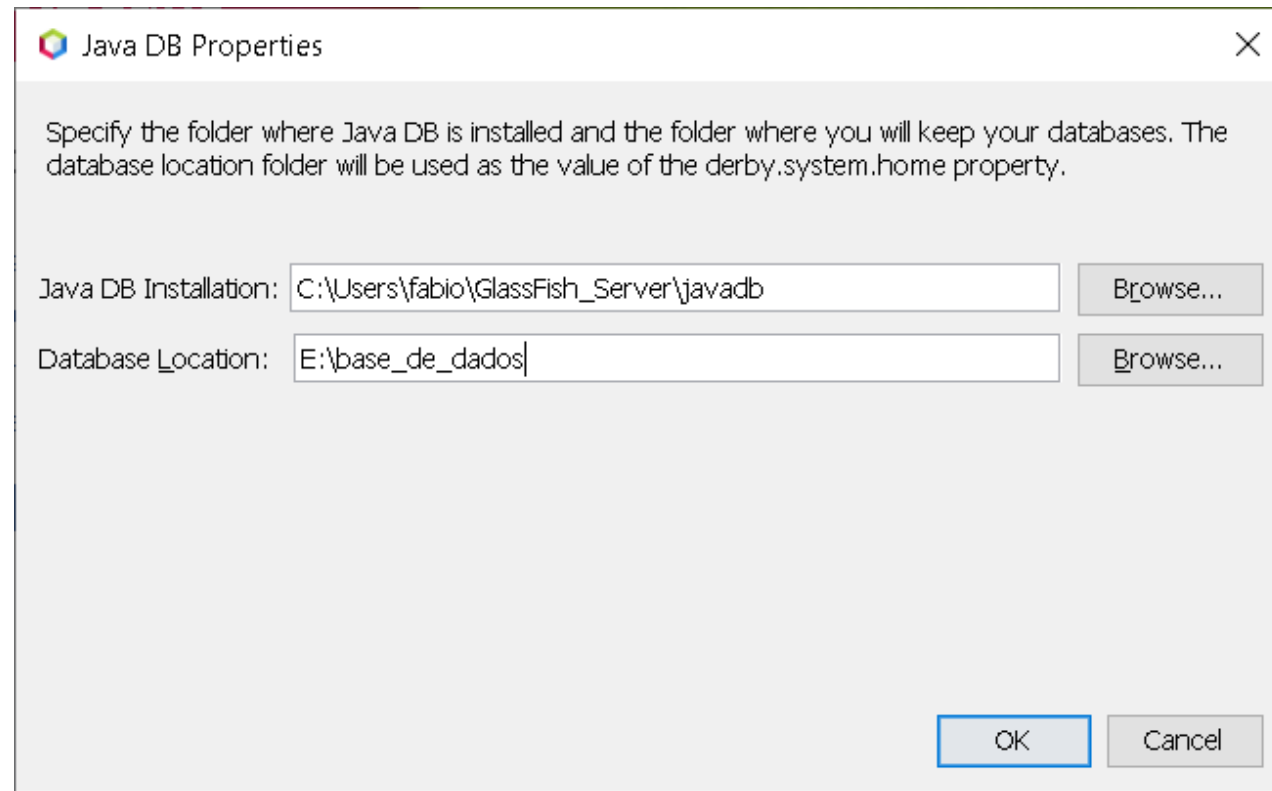
O Java DB é um SGDB seguro e totalmente transacional escrito inteiramente em Java, com suporte à linguagem de consulta SQL e disponibilização da API Java DataBase Connectivity (JDBC), que fornece um conjunto de funções de conexão e manipulação de banco de dados para o desenvolvimento de aplicações comerciais em Java.



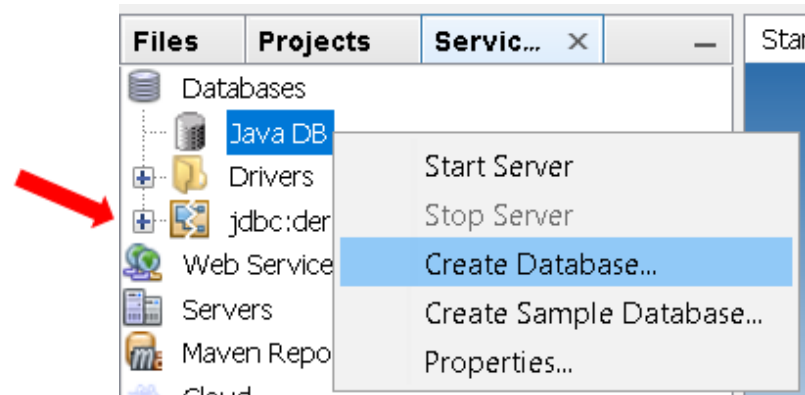
Para configurar o servidor Java DB (Apache Derby) no NetBeans, basta acessar no menu Window à Services. Em seguida, após aparecer a janela de serviços (Services), clique com o botão direito do mouse em Java DB e escolha a opção Properties.



Então, você terá acesso à janela a seguir com o caminho para onde está instalado o Java DB. Em Database Location, selecione a pasta na qual serão criados seus bancos de dados (por exemplo, E:\base_de_dados) e pressione OK.



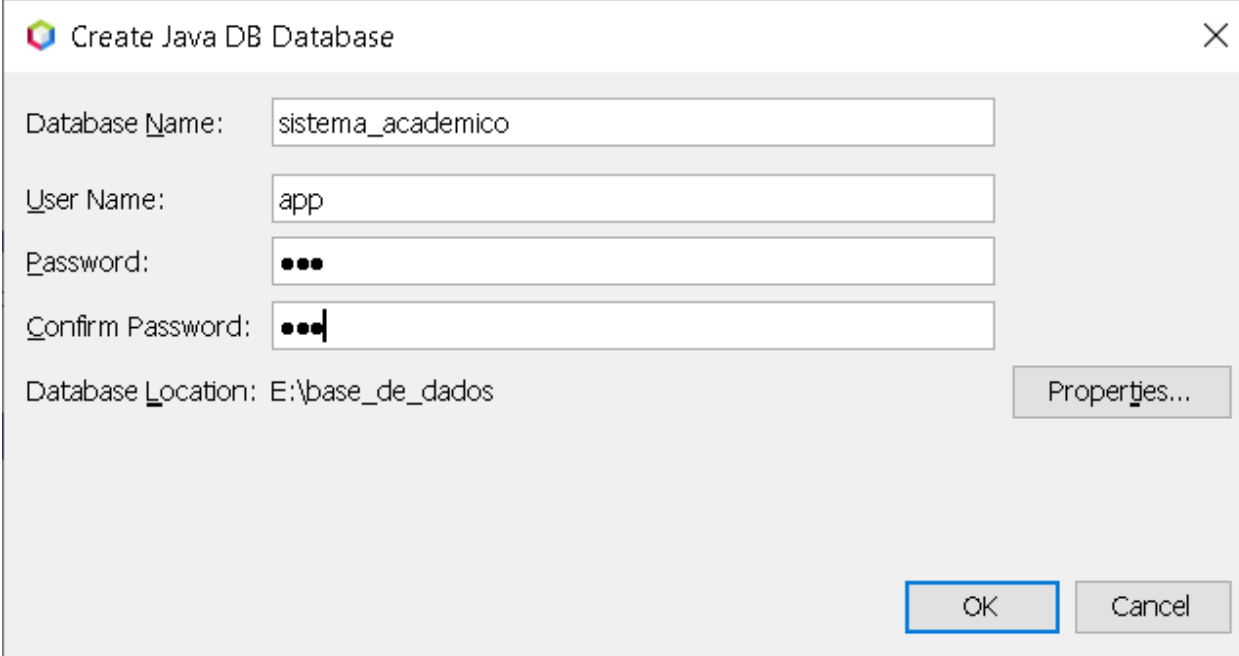
Para criar um banco de dados, clique novamente com o botão direito do mouse em Java DB e escolha a opção Create Database.



Na janela seguinte, preencha as caixas de textos com as seguintes informações:

Database Name: sistema_academico

User Name/Password: app



Create Java DB Database

Database Name: sistema_academico

User Name: app

Password: ...

Confirm Password: ...

Database Location: E:\base_de_dados

Properties...

OK Cancel

Agora que temos o banco de dados sistema_academico, podemos praticar um pouco o acesso ao banco de dados utilizando a linguagem SQL. Você se lembra da atividade APLICANDO CONHECIMENTO da Aula 2?

Nessa atividade, você tinha de fazer a leitura de um conjunto de alunos armazenado em um arquivo texto. Os alunos tinham as seguintes informações: id aluno, nome e nota. Veja o exemplo abaixo:

111;Jose da Silva;10.0

222;Manoel Pereira;5.0

333;Carlos Dias;7.5

122;Ana Andrade;6.5

441;Mario Souza;3.5

Como estamos trabalhando agora com um banco de dados, as informações dos alunos serão persistidas em uma tabela, que é uma estrutura de armazenamento de dados na qual as informações dos alunos são armazenadas em linhas (registro), cada linha é dividida em várias colunas (campos) e cada coluna tem um nome para que possamos acessar os dados dos alunos como se fossem variáveis em um programa.

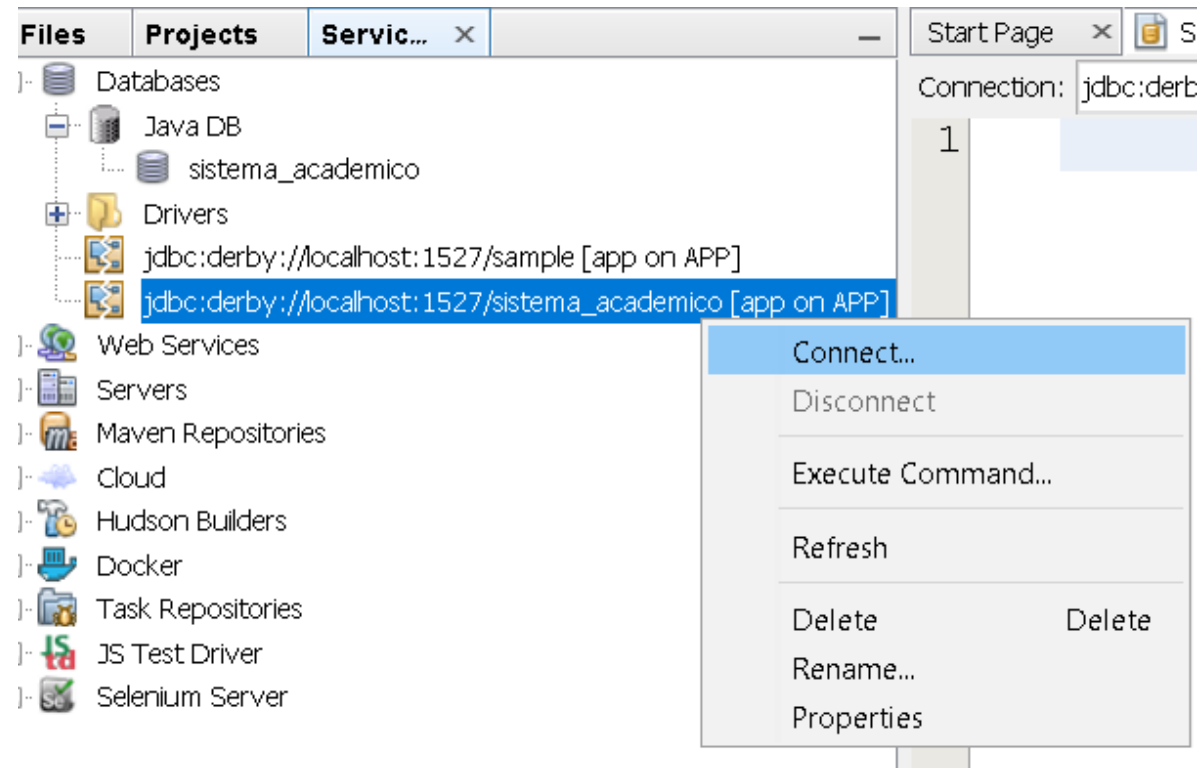
id_aluno	nome	nota
111	Jose da Silva	10.0
222	Manoel Pereira	5.0
333	Carlos Dias	7.5
122	Ana Andrade	6.5
441	Mario Souza	3.5

Para criar uma tabela no banco de dados, usamos o comando CREATE TABLE, seguido pelo nome da tabela. Para informar os nomes das colunas, escrevemos entre parênteses o nome do campo, seguido do tipo de informação que será armazenada. As definições das colunas são separadas por vírgula, conforme o exemplo:

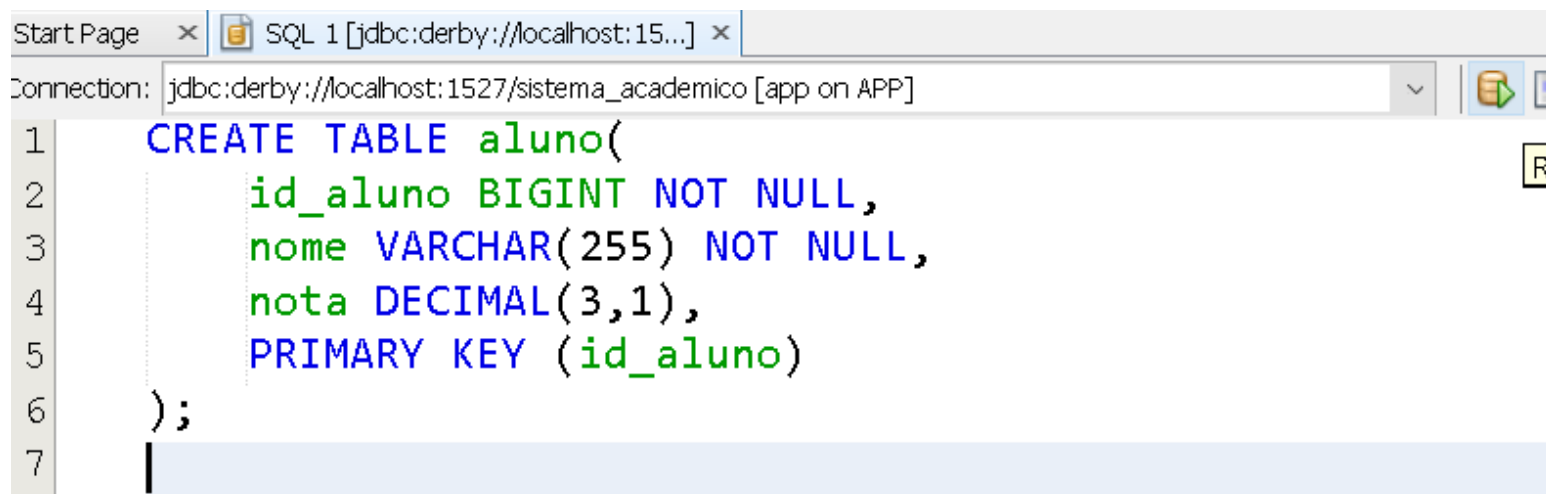
```
CREATE TABLE aluno(  
    id_aluno BIGINT NOT NULL,  
    nome VARCHAR(255) NOT NULL,  
    nota DECIMAL(3,1),  
    PRIMARY KEY (id_aluno)
```

);

Perceba que definimos o campo `id_aluno` como chave primária na tabela, ou seja, o valor dessa coluna não deve se repetir. Para executar o comando SQL `CREATE TABLE` no NetBeans, clique com o botão direito do mouse no banco de dados `sistema_academico` dentro de Database e selecione `Connect`. Para digitar os comandos SQL, selecione a opção `Execute Command` no menu pop-up.



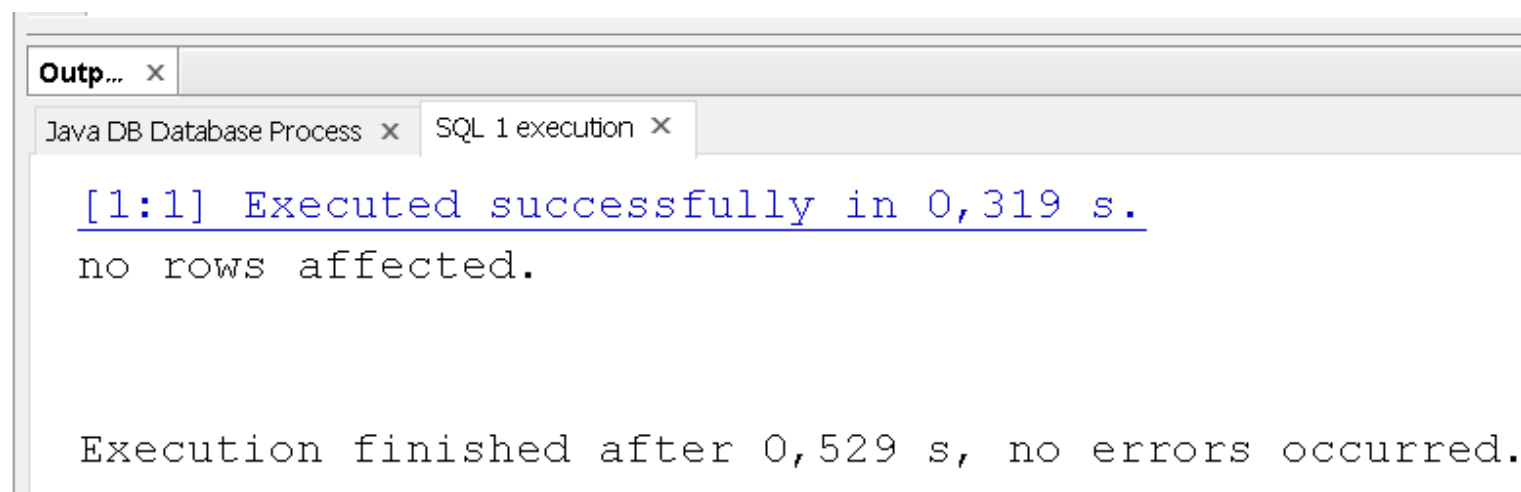
Ao clicar em `Execute Command`, aparecerá uma aba conforme exemplo abaixo, e nela você pode digitar um comando da linguagem SQL. Para executar o comando SQL, clique no ícone



The screenshot shows a SQL IDE window with a single tab titled 'SQL 1 [jdbc:derby://localhost:15...]' and a close button. Below the tab bar, the connection string is displayed as 'jdbc:derby://localhost:1527/sistema_academico [app on APP]'. The main editor area contains the following SQL code:

```
1 CREATE TABLE aluno(  
2     id_aluno BIGINT NOT NULL,  
3     nome VARCHAR(255) NOT NULL,  
4     nota DECIMAL(3,1),  
5     PRIMARY KEY (id_aluno)  
6 );  
7
```

Caso o comando seja executado, a seguinte mensagem será exibida na janela do output:



The screenshot shows the output window with two tabs: 'Java DB Database Process' and 'SQL 1 execution'. The output text is as follows:

```
[1:1] Executed successfully in 0,319 s.  
no rows affected.  
  
Execution finished after 0,529 s, no errors occurred.
```






Para inserir os dados dos alunos na tabela aluno, usaremos o comando INSERT. Para isso, devemos escrever INSERT INTO, seguido do nome da tabela. Depois, devemos escrever VALUES e, entre parênteses, os valores que serão inseridos nas colunas na ordem em que eles apareceram quando criamos a tabela usando o comando CREATE TABLE. Agora, clique novamente no ícone para executar um comando SQL:

```
SQL 1 [jdbc:derby://localhost:15...] x
Connection: jdbc:derby://localhost:1527/sistema_academico [app on APP]
1  INSERT INTO aluno VALUES (111, 'Jose da Silva', 10.0);
2  INSERT INTO aluno VALUES (222, 'Manoel Pereira', 5.0);
3  INSERT INTO aluno VALUES (333, 'Carlos Dias', 7.0);
4  INSERT INTO aluno VALUES (122, 'Ana Andrade', 6.5);
5  INSERT INTO aluno VALUES (441, 'Mario Souza', 3.5);
```

Para conferir o que foi inserido, podemos selecionar todos os campos da tabela aluno (asterisco) com consulta (query) usando o comando `SELECT * FROM aluno`, conforme exemplo abaixo:

```
SQL 1 [jdbc:derby://localhost:15...] x
Connection: jdbc:derby://localhost:1527/sistema_academico [app on APP]
1  SELECT * FROM aluno;
2
```

E, em seguida, serão apresentados todos os alunos cadastrados na tabela aluno.

SELECT * FROM aluno x			
     Max. rows: 100 Fetched Rows: 5			
#	ID_ALUNO	NOME	NOTA
1	111	Jose da Silva	10.0
2	222	Manoel Pereira	5.0
3	333	Carlos Dias	7.0
4	122	Ana Andrade	6.5
5	441	Mario Souza	3.5

Neste momento, não se preocupe com os detalhes da linguagem SQL, isso será explicado na disciplina de Banco de Dados deste semestre. O mais importante é que os comandos sejam executados corretamente.