

Sistemas

Cliente/Servidor

**MIDDLEWARE JDBC:
USANDO O JAVA DB**

Conteúdo:

- Processamento de banco de dados com JSP
- MySQL
- Connector-J
- Exemplos

- Compreender o conceito do Middleware JDBC
- Capacitar para a criação de JSP e servlets que consultem BDs

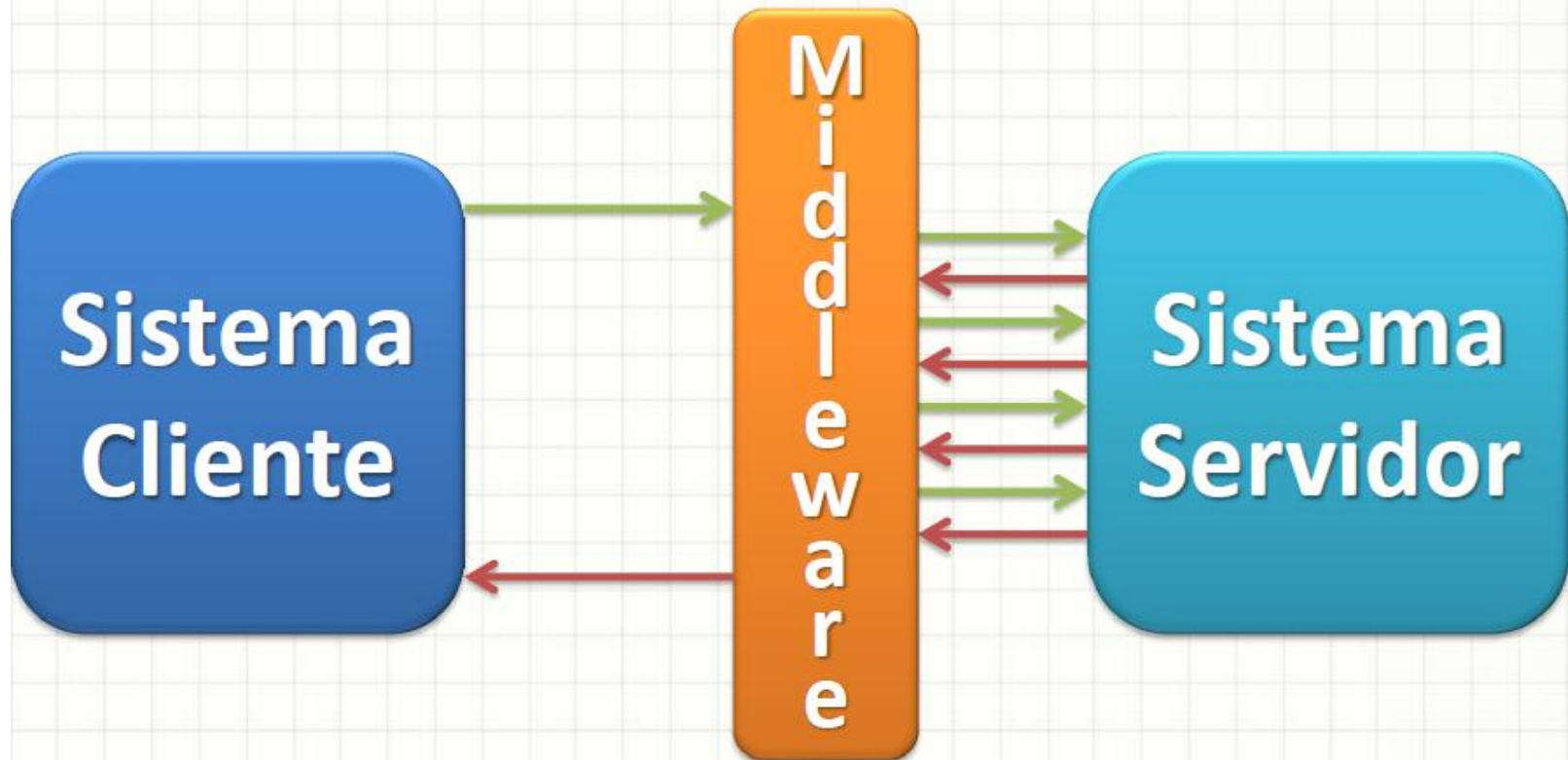
O Que é Middleware?

- Middle: meio
 - Middleware: software intermediário
- “Encapsular” complexidades de uma tarefa



O Que é Middleware?

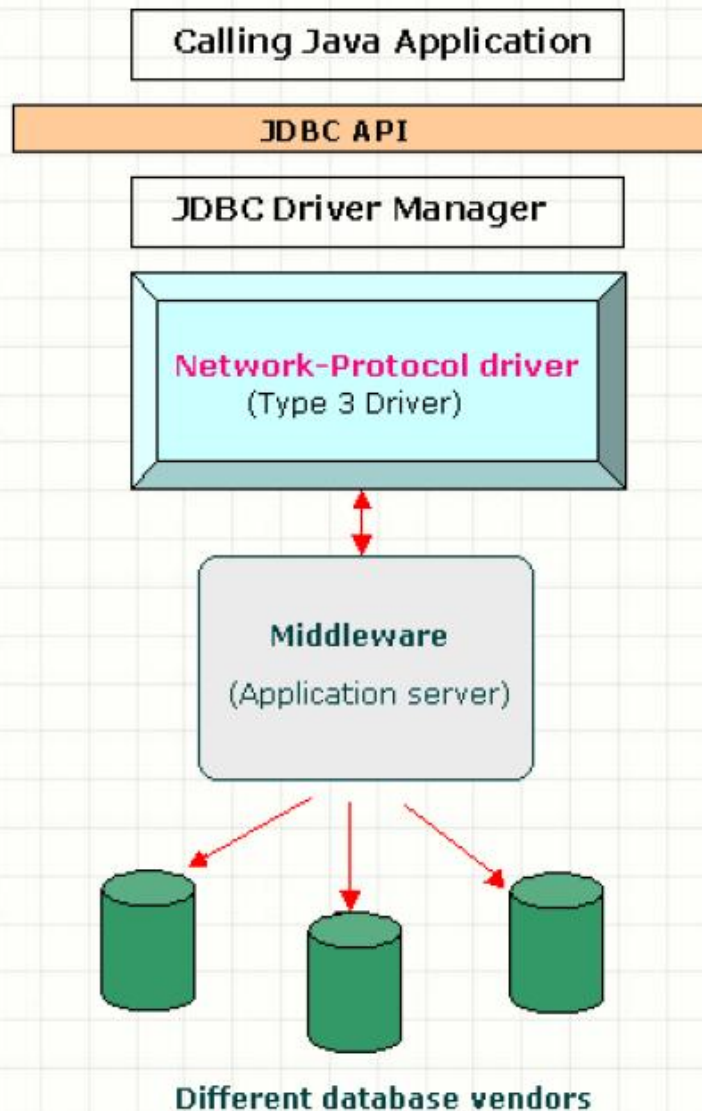
- Middle: meio
 - Middleware: software intermediário
- “Encapsular” complexidades de uma tarefa



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

Estrutura do Middleware JDBC



Principais classes e interfaces de JDBC

(Java Data Base Connectivity)

```
import java.sql.*;
```

```
class DriverManager
```

```
interface Connection
```

```
interface Statement
```

```
interface ResultSet
```

Carregamento do Driver e conexão com a base

Para carregar o manipulador (driver) do BD:

```
Class.forName("nomeDoDriver");
```

Para criar a conexão com a fonte de dados:

```
Connection con =  
DriverManager.getConnection(url, "usuário", "senha");
```


Objeto de comando e objeto de pesquisa

Para criar um objeto de comando:

```
Statement stmt = con.createStatement(...);
```

Para criar um conjunto de resultados:

```
ResultSet rs = stmt.executeQuery(query);
```



Para comandos SQL de pesquisa ou
Comandos que não alteram o banco.

Métodos de acesso aos campos

Para pegar valores de campos:

```
int a = rs.getInt("nomeDeUmCampoInteiro");  
String b = rs.getString("nomeDeUmCampoString");
```

Para executar um comando SQL de alteração:

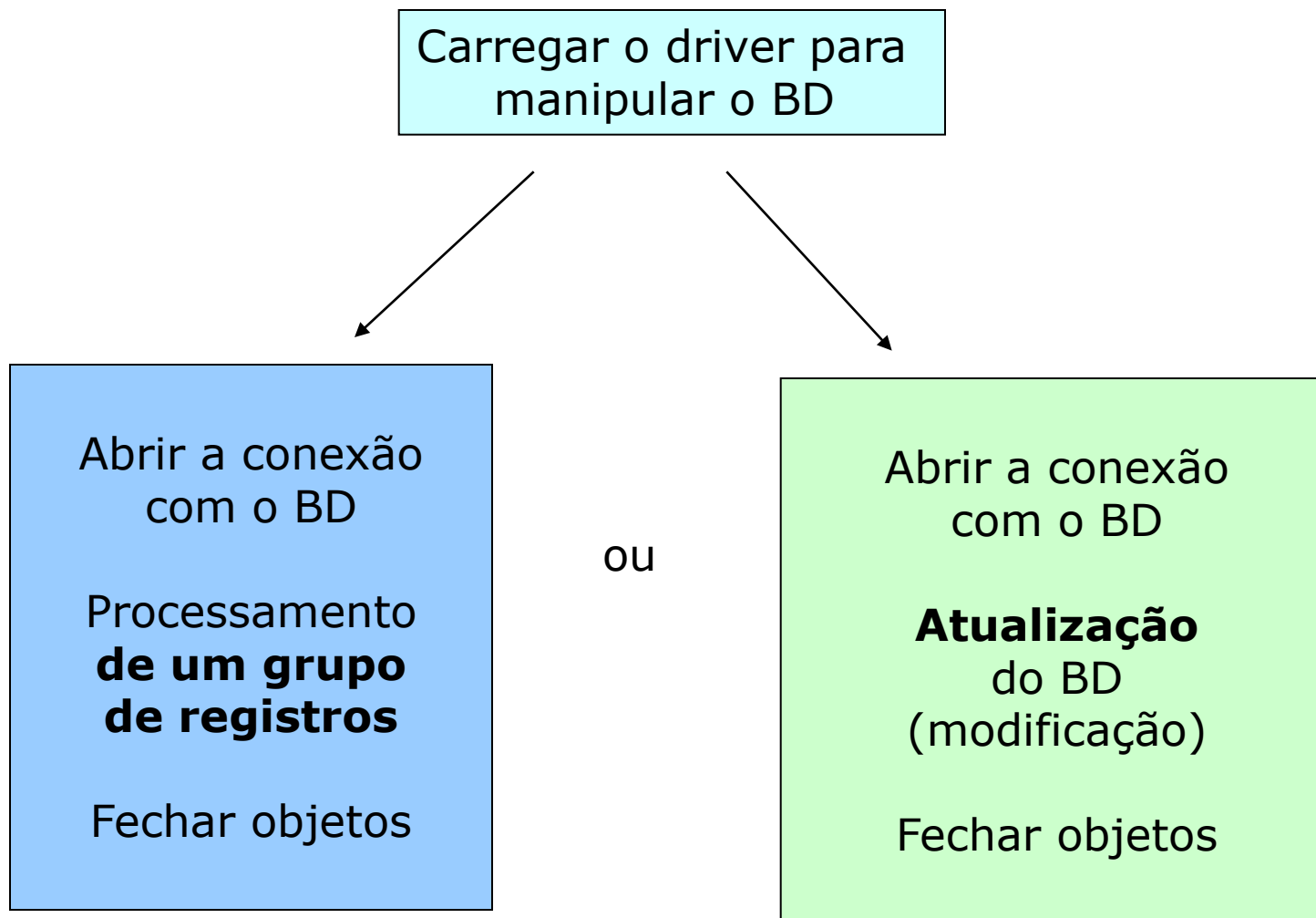
Método para alterações de dados

```
stmt.executeUpdate(query);
```

Para fechar objetos:

```
if(rs!=null) {rs.close(); rs=null;}  
if(stmt!=null) {stmt.close(); stmt=null;}  
if(con!=null) {con.close(); con=null;}
```

Estrutura típica do processamento de BDs com JDBC



Carregando o driver do Banco de dados

//**Carregar o driver** para manipulação do BD.
//Está lógica poderia ser colocada no método construtor da classe ou pelo menos
//sempre antes de abrir a conexão com o banco de dados.

```
try {  
    Class.forName("nomeDoDriverParaEsteFormatoDeBD"); //carregamos o driver  
}  
catch(ClassNotFoundException ex){  
    JOptionPane.showMessageDialog(null,  
        "Erro carregando o driver do banco de dados -> " + ex.getMessage(),  
        "Erro", JOptionPane.PLAIN_MESSAGE );  
}
```



dependerá do formato de BD que queremos processar

Processando um grupo de registros

//Processamento de um grupo de registros:

```
Connection con=null;
Statement stmt=null;
ResultSet rs=null;
String query="....."; //tipicamente será um comando SQL do tipo SELECT
String url = "....."; //especifique o banco com uma sintaxe adequada

try {
    con = DriverManager.getConnection(url,"loginUsuario", "senhaUsuario"); //cria a conexão
    //criamos o objeto para executar comandos SQL:
    stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                               ResultSet.CONCUR_UPDATABLE); //permite atualizar e "scrollable"
    rs = stmt.executeQuery(query); //agora criamos o conjunto de registros
    while (rs.next()){ //percorrendo os registros e processando seus valores
        int a = rs.getInt("nomeDeUmCampoInteiro");
        String b = rs.getString("nomeDeUmCampoString");
        float c = rs.getFloat("nomeDeUmCampoFloat");
        //.....processar os dados anteriores.....
    }
}
catch(SQLException ex){
    JOptionPane.showMessageDialog(null,
    "Erro abrindo conexão ou criando o objeto para comandos -> " + ex.getMessage(),
    "Erro", JOptionPane.PLAIN_MESSAGE );
}
finally{
    try{
        if(rs!=null) {rs.close(); rs=null;}
        if(stmt!=null) {stmt.close(); stmt=null;}
        if(con!=null) {con.close(); con=null;}
    }
    catch(SQLException exfecha){
    }
}
```

Atualizando o Banco de Dados

//Processamento: comandos SQL que atualizam o BD:

```
Connection con=null;
Statement stmt=null;
String query="....."; // um comando SQL de alteração do BD (INSERT INTO, DELETE, UPDATE)
String url = "....."; //especificamos o banco com uma sintaxe adequada

try {
    //criamos a conexão:
    con = DriverManager.getConnection(url,"loginUsuario", "senhaUsuario");
    //criamos o objeto para executar comandos SQL:
    stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                               ResultSet.CONCUR_UPDATABLE); //permite atualizar e "scrollable"
    stmt.executeUpdate(query); //executamos o comando SQL de atualização
}
catch(SQLException ex){
    JOptionPane.showMessageDialog(null,
    "Erro abrindo conexão ou criando o objeto para comandos -> " + ex.getMessage(),
    "Erro", JOptionPane.PLAIN_MESSAGE );
}
finally{
    try{
        if(stmt!=null) {stmt.close(); stmt=null;}
        if(con!=null) {con.close(); con=null;}
    }
    catch(SQLException exfecha){
    }
}
```

Drivers para acesso a bancos de dados

Para baixar o driver para acessar bancos de dados MySQL via ODBC você deve baixar o driver atualizado do site da MySQL: www.mysql.com

Lembre-se que deverá cadastrar uma fonte de dados ODBC no *Painel de Controle / Ferramentas administrativas / ODBC* para cada banco de dados que deseje acessar.

Para baixar o Connector-J, driver para acessar bancos de dados MySQL diretamente em programas Java, sem necessidade de instalar fontes ODBC:

Coloque o arquivo .jar da pasta Bibliotecas no Netbeans para que fique disponível para todos os aplicativos Web.

Baixe a versão atualizada de:

<http://www.mysql.com/downloads/connector/j>

Para baixar outros drivers

Outras empresas poderão fornecer drivers gratuitos para acessar bancos de dados com Java e JDBC. As informações e orientação serão fornecidas pelo fabricante. Também, na instalação do sistema operacional, alguns drivers de BDs são instalados automaticamente.

Abrindo a conexão com o banco de dados (1)

Via fonte ODBC (para qualquer banco com suporte a ODBC)

```
//Carregamos o driver genérico para acesso via ODBC:  
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");  
String url = "jdbc:odbc:NomeDaFonteODBC"; //especificamos a fonte ODBC cadastrada  
con = DriverManager.getConnection(url,"", ""); //banco de dados com ou sem senha  
Lembre-se, que deverá instalar antes o "driver ODBC" específico para o BD a ser  
acessado. O acesso ao instalador para ODBC MySQL foi comentado antes.
```

Via Connector-J para programas Java e bancos de dados MySQL

O arquivo JAR deve estar dentro do diretório: %TOMCAT_HOME%\lib

```
//Observe o nome do driver Connector-J para MySQL:  
Class.forName("com.mysql.jdbc.Driver"); //ou poderia ser: org.gjt.mm.mysql.Driver  
//Observe a sintaxe especial para acessar o servidor de MySQL e que especificaremos o  
//nome do BD e não o nome da fonte de dados ODBC (no exemplo seguinte universidade  
//é o nome do BD):  
String url = "jdbc:mysql://localhost/universidade";  
con = DriverManager.getConnection(url,"root", "123"); //banco de dados com senha
```

Abrindo a conexão com o banco de dados (2)

Acesso direto a bancos de dados Access no Windows, sem necessidade de cadastro de uma fonte de dados ODBC:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

//Especifique caminho e nome do BD no elemento path:

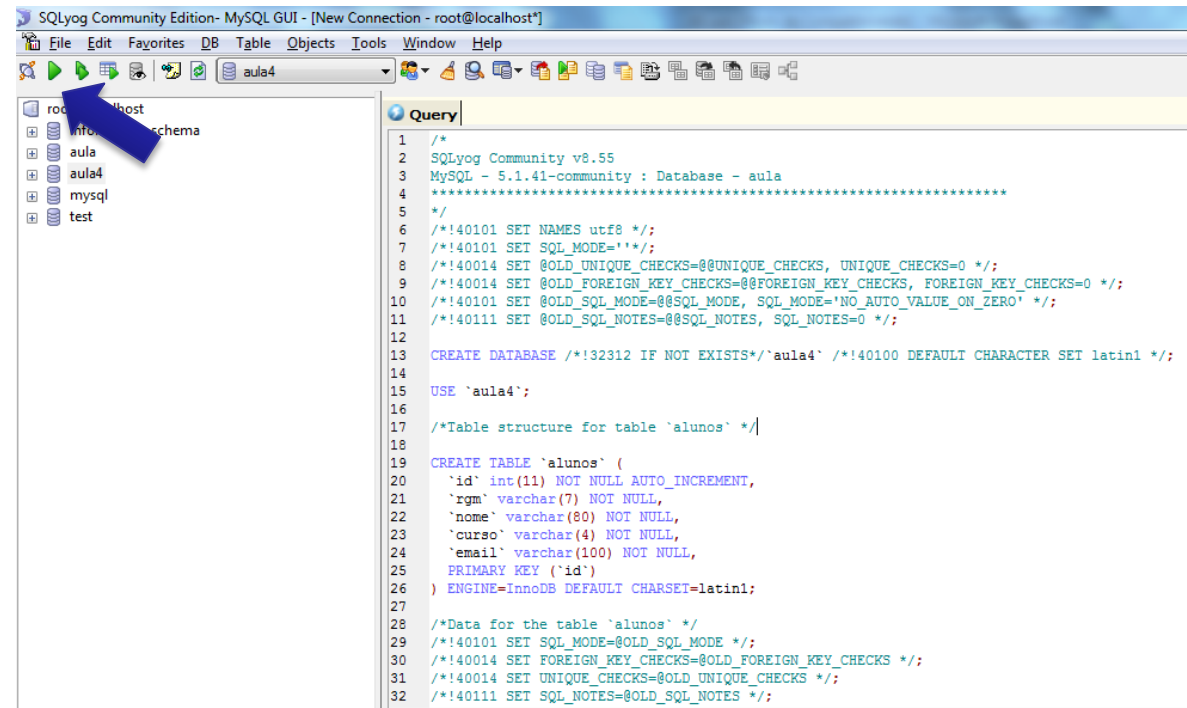
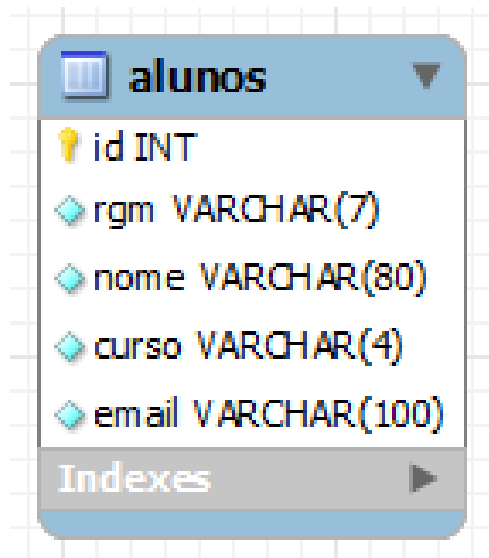
```
url = "jdbc:odbc:DBQ=" + path + ";Driver={Microsoft Access Driver (*.mdb)}";  
con = DriverManager.getConnection(url,"", "");
```

Exemplo

- Vamos criar um aplicativo web para cadastro de alunos, este cadastro deverá:
 - Cadastrar novos alunos
 - Listar os alunos cadastrados
 - Apagar um cadastro existente

Modelando o banco

- Antes de fazer a modelagem, abra o SQLyog e crie um banco de dados com o nome **aula** (Utilize o script **BD.sql**)
- Como este é um exemplo introdutório, iremos criar somente uma tabela para armazenar as informações dos alunos
- Após abrir a ferramenta SQLyog, e executar o script.



SQLyog Community Edition - MySQL GUI - [New Connection - root@localhost]

Query

```
1  /*
2  SQLyog Community v8.55
3  MySQL - 5.1.41-community : Database - aula
4  *****
5  */
6  /*!40101 SET NAMES utf8 */;
7  /*!40101 SET SQL_MODE=''; */;
8  /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
9  /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
10 /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
11 /*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
12
13 CREATE DATABASE /*!32312 IF NOT EXISTS*/ `aula4` /*!40100 DEFAULT CHARACTER SET latin1 */;
14
15 USE `aula4`;
16
17 /*Table structure for table `alunos` */
18
19 CREATE TABLE `alunos` (
20   `id` int(11) NOT NULL AUTO_INCREMENT,
21   `rgm` varchar(7) NOT NULL,
22   `nome` varchar(80) NOT NULL,
23   `curso` varchar(4) NOT NULL,
24   `email` varchar(100) NOT NULL,
25   PRIMARY KEY (`id`)
26 ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
27
28 /*Data for the table `alunos` */
29 /*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
30 /*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
31 /*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
32 /*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

Criando a aplicação

- Após a criação do banco, iremos criar as páginas JSP para a aplicação.
- Crie primeiro uma página index.html com os links
 - Cadastrar Novo aluno (link para cad_novo.html)
 - Listar alunos cadastrados (link para lista.jsp)
- Em seguida crie a página cad_novo.html com um formulário contendo os campos (rgm, nome, curso e email)
- O formulário deverá chamar no atributo action a página grava_novo.jsp

Insira nomes para cada campo



RGM:

Nome:

Curso: CCP

E-mail:

JSP para gravação

```
<%  
Connection con;  
Statement stm;  
String sql;  
String url = "jdbc:mysql://localhost/aula";  
try{  
    Class.forName("com.mysql.jdbc.Driver");  
    con = DriverManager.getConnection(url,"root","root");  
    stm = con.createStatement();  
    sql = "INSERT INTO alunos (rgm,nome,curso,email) VALUES (" +  
        request.getParameter("rgm")+"",""+  
        request.getParameter("nome")+"",""+  
        request.getParameter("curso")+"",""+  
        request.getParameter("email")+"")";  
  
    int retorno = stm.executeUpdate(sql);  
    if (retorno > 0)  
        out.print("Dados gravados com sucesso");  
    else  
        out.print("Erro. Tente novamente");  
}  
catch(SQLException ex){  
    out.print(ex.getMessage());  
}  
//fechar objetos abertos  
%>
```

JSP para listagem

- Crie o arquivo lista.jsp com uma tabela conforme exemplo abaixo.

ID	RGM	NOME	CURSO	EMAIL	
					X

- Iremos inserir o código JSP que irá consultar a tabela alunos e mostrar cada aluno em uma linha da tabela anterior.

JSP para listagem

```
<table...>
```

```
.....
```

```
<%
```

```
Connection con;
```

```
Statement stm;
```

```
ResultSet rs; //objeto que irá guardar o retorno da consulta
```

```
String sql;
```

```
String url = "jdbc:mysql://localhost/aula4";
```

```
try{
```

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

```
    con = DriverManager.getConnection(url,"root","123");
```

```
    stm = con.createStatement();
```

```
    sql = "SELECT * FROM alunos ORDER BY nome";
```

```
    rs = stm.executeQuery(sql);
```

```
    while (rs.next()){
```

```
        %>
```

```
    }
    <%
```

```
    }
```

```
}
```

```
catch(SQLException ex){
```

```
    out.print(ex.getMessage());
```

```
%>
```

```
</table>
```

```
<tr>
```

```
    <td><%= rs.getString("id") %></td>
```

```
    <td><%= rs.getString("rgm") %></td>
```

```
    <td><%= rs.getString("nome") %></td>
```

```
    <td><%= rs.getString("curso") %></td>
```

```
    <td><%= rs.getString("email") %></td>
```

```
    <td><a href="apagar.jsp?id=<%= rs.getString("id")%>">X</a></td>
```

```
</tr>
```

JSP para apagar registros

- Nome do arquivo apagar.jsp

```
<%  
Connection con;  
Statement stm;  
String sql;  
String url = "jdbc:mysql://localhost/aula";  
try  
{  
    Class.forName("com.mysql.jdbc.Driver");  
    con = DriverManager.getConnection(url,"root","123");  
    stm = con.createStatement();  
    sql = "DELETE FROM alunos WHERE id = "+request.getParameter("id");  
    stm.executeUpdate(sql);  
    response.sendRedirect("lista.jsp");  
}  
catch(SQLException ex)  
{  
    out.print(ex.getMessage());  
}  
%>
```

Banco de dados com java bean!!

Exemplo 3 e 4

Exercício

- Com base no exemplo de aula, crie um cadastro de produtos conforme diagrama abaixo. Você deverá implementar todos os arquivos necessários para:
 - Cadastrar um novo produto
 - Listar os produtos cadastrados ordenados por nome
 - Apagar um produto cadastrado

