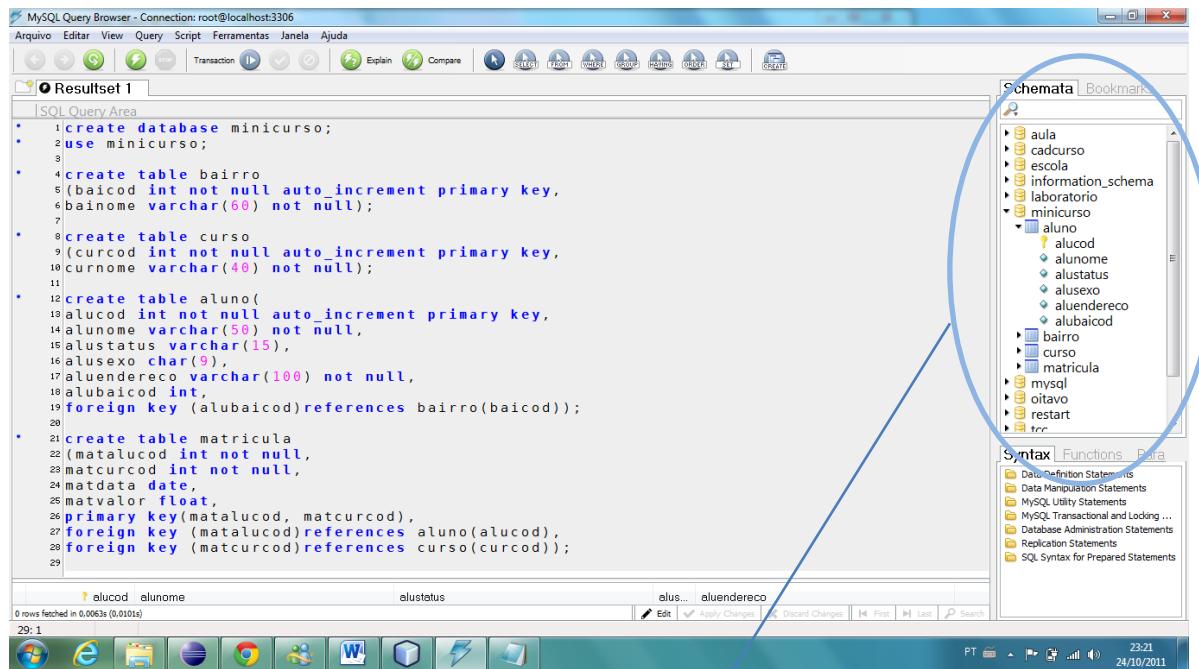


# **TUTORIAL PROJETO WEB**

**PROF.: Esp. Manfrine Tapiramutá**

Primeiro passo é criar um banco de dados. Pois com um esquema montado, utilizaremos o Framework de mapeamento objeto relacional o **Hibernate**, para montar nosso projeto.



The screenshot shows the MySQL Query Browser interface. The SQL Query Area contains the following SQL script:

```

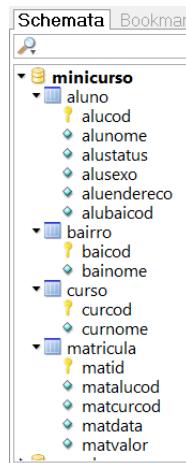
1 create database minicurso;
2 use minicurso;
3
4 create table bairro
5 (baicod int not null auto_increment primary key,
6 bainome varchar(60) not null);
7
8 create table curso
9 (curcod int not null auto_increment primary key,
10 curnome varchar(40) not null);
11
12 create table aluno(
13 alucod int not null auto_increment primary key,
14 alunome varchar(50) not null,
15 alustatus varchar(15),
16 alusexo char(9),
17 aluendereco varchar(100) not null,
18 alubaicod int,
19 foreign key (alubaicod) references bairro(baicod));
20
21 create table matricula
22 (matalucod int not null,
23 matcurcod int not null,
24 matdata date,
25 matvalor float,
26 primary key(matalucod, matcurcod),
27 foreign key (matalucod) references aluno(alucod),
28 foreign key (matcurcod) references curso(curcod));
29

```

The Schema pane on the right shows the database structure:

- aula
- cadcurso
- escola
- information\_schema
- laboratorio
- minicurso
  - aluno
    - alucod
    - alunome
    - alustatus
    - alusexo
    - aluendereco
    - alubaicod
  - bairro
  - curso
  - matricula
- mysql
- oitavo
- restart
- tcc

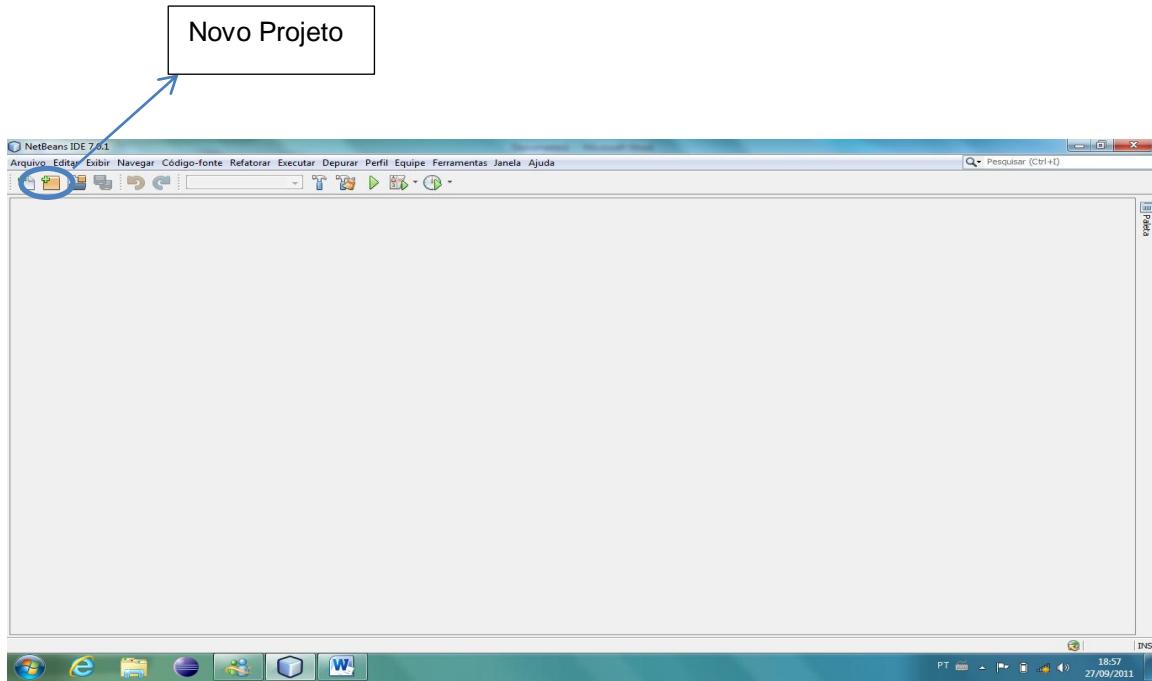
A figura acima mostra o script de criação do banco de dados que chamaremos de minicurso.



Como o foco desse trabalho é o desenvolvimento de um projeto JAVA WEB. Não será a abordado Conceitos mais profundo de banco de dados.

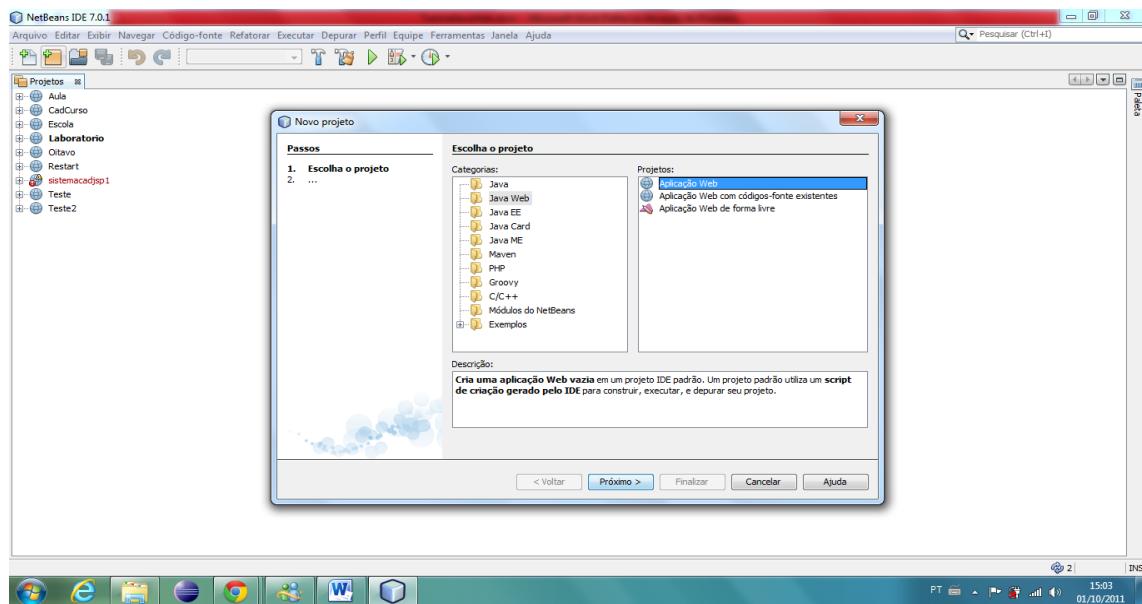
O **NETBEANS 7.0**, é a IDE que iremos utilizar para a criação do nosso projeto JavaWEB.

O **NETBEANS 7.0**, traz consigo um conjunto de plugins (Aplicativos) para os mais diversos tipos de desenvolvimento é uma ferramenta multi - linguagem suporta as linguagens Java (ME, SE e EE), PHP, C/C++ entre outras.



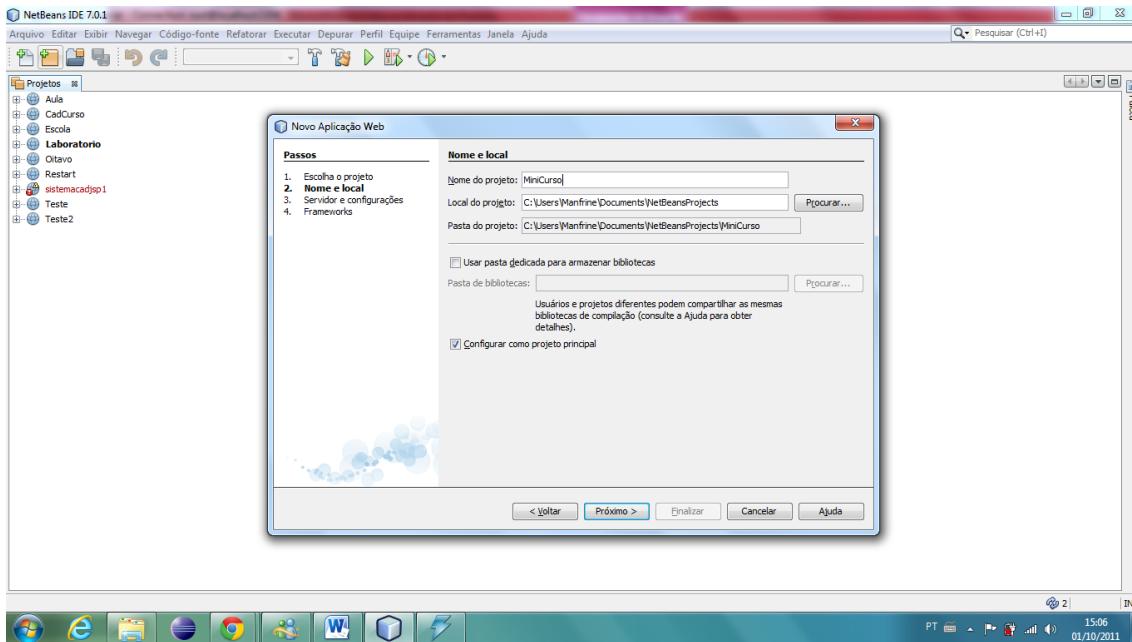
Agora vamos escolha a plataforma que iremos trabalhar.

- No nosso caso, um **Projeto Java WEB – Aplicação WEB**



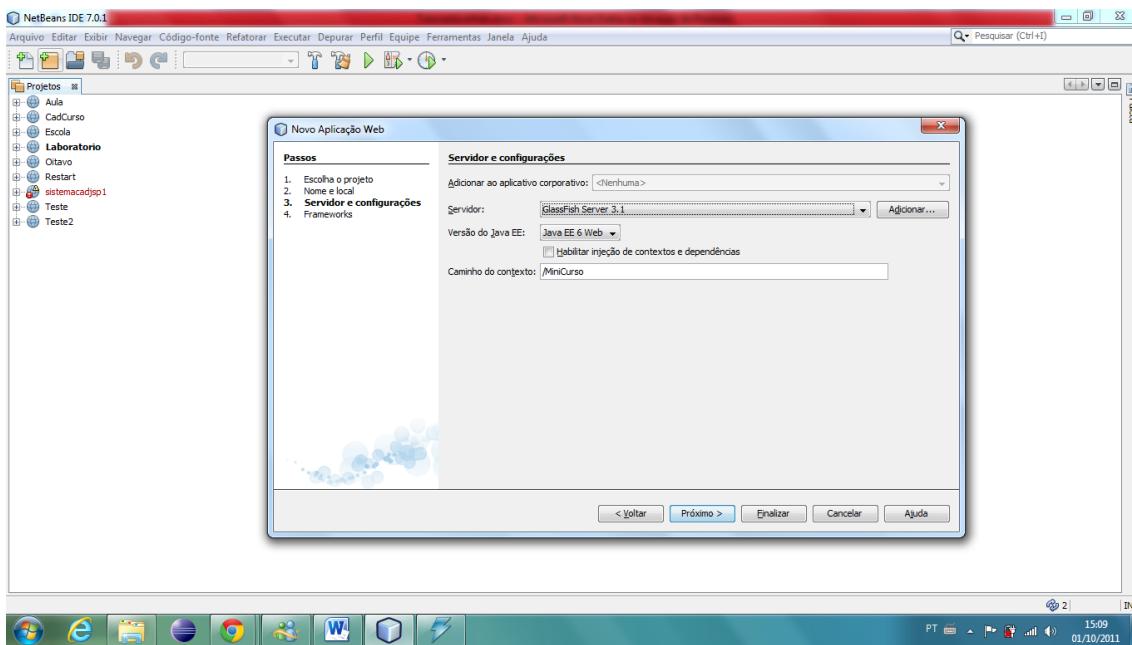
Agora iremos dar nome ao projeto. Optei por colocar o nome do Banco no caso **MiniCurso**.

## Professor Manfrine Tapiramutá – Tecnologia Web com Java



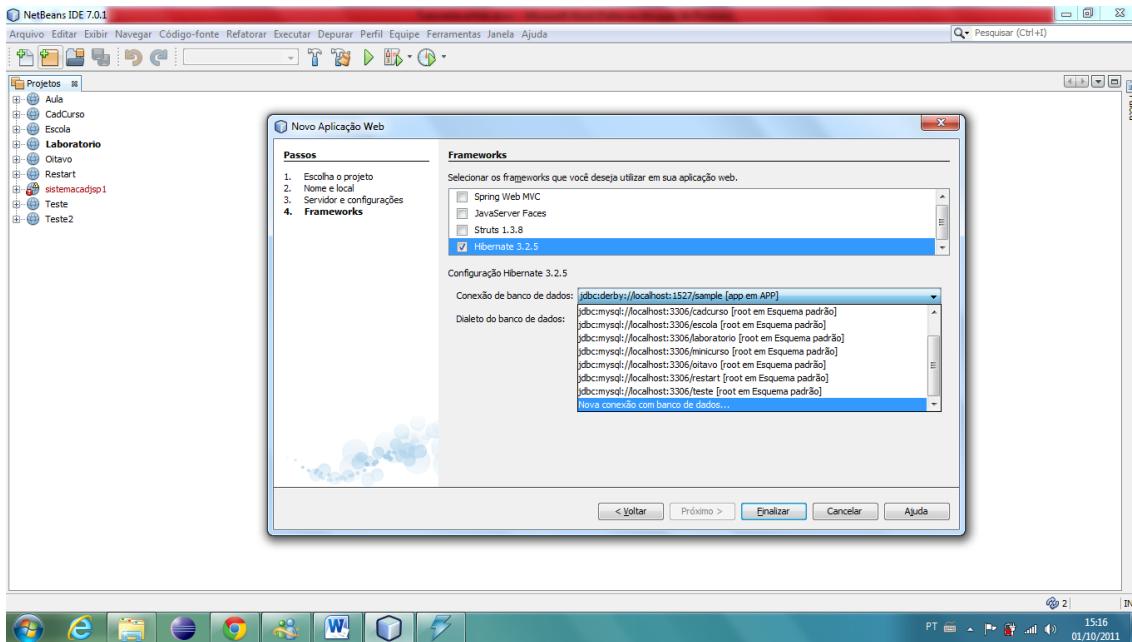
Avança no botão próximo.

A próxima tela é para escolher o servidor de aplicação (Local onde será executado nosso projeto WEB). Fica a critério do desenvolvedor, para esse projeto utilizarei o GlassFish.



A próxima tela requer um pouco de atenção aos passos, é nela que iremos selecionar o framework Hibernate, para controlar toda a camada de dados do sistema.

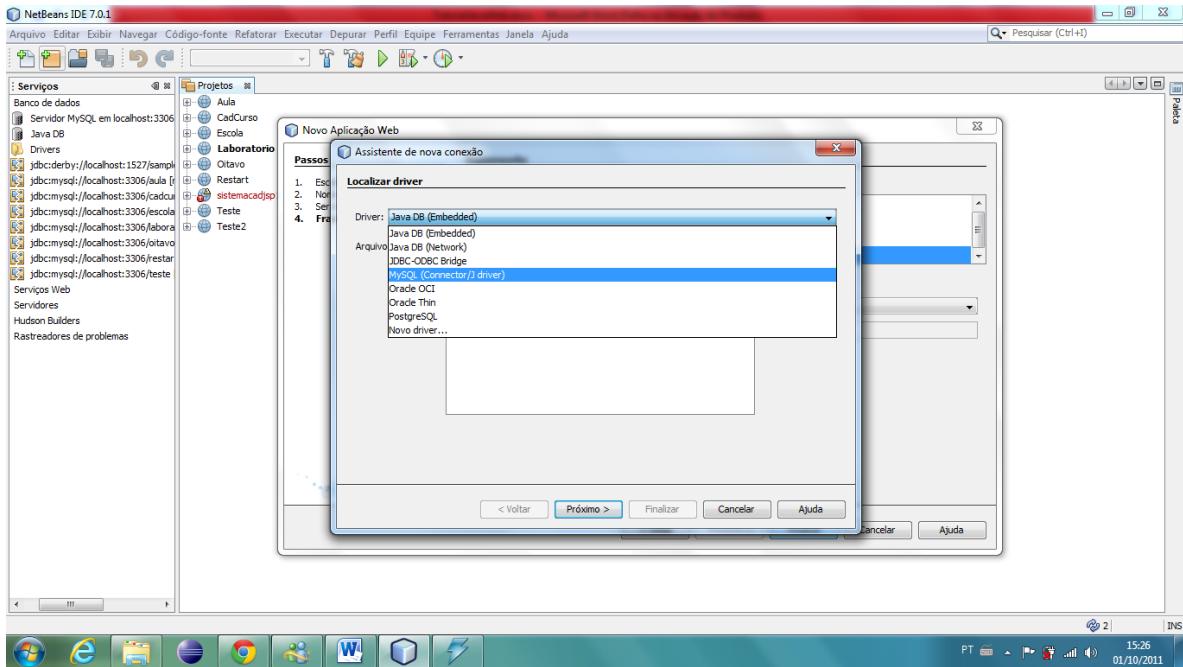
# Professor Manfrine Tapiramutá – Tecnologia Web com Java



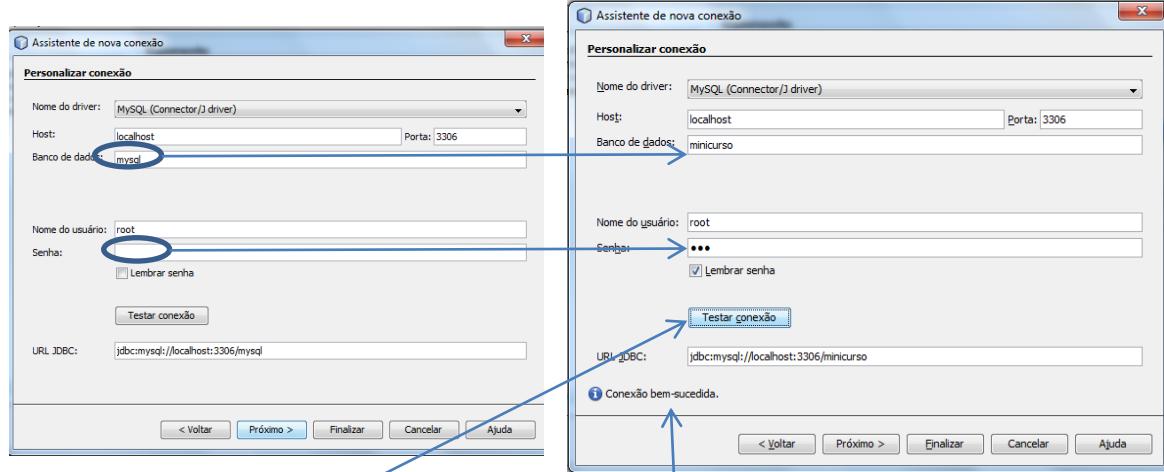
Passos:

- Selecionar a opção Hibernate;
- Seleciona uma nova conexão de banco de dados -> na opção Conexão de banco de dados.

A próxima tela escolhemos o Driver (Software que faz a conexão de um código java com a tecnologia de banco de dados utilizada.). No nosso caso escolheremos o MySQL, pois é o SGBD que utilizaremos.

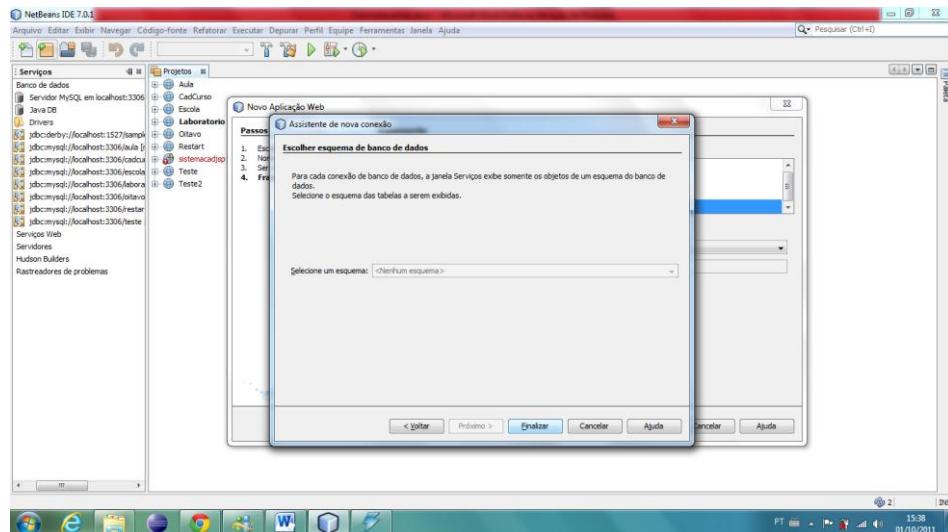


Escolhido o driver Conector, a próxima tela temos que adicionar 2 (Duas informações o banco de dados e a senha).

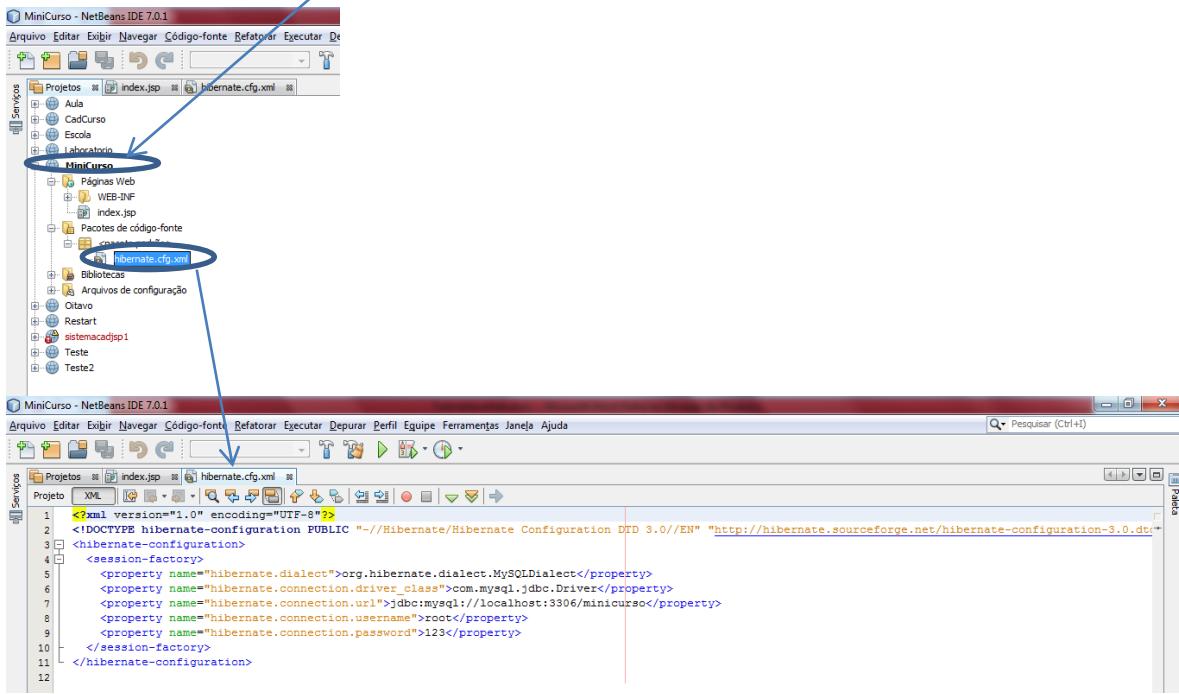


Azione o botão de **Testar conexão**. E observe se a conexão foi efetuada com sucesso.

Depois é só finalizar.



E o projeto acabe de ser gerado.



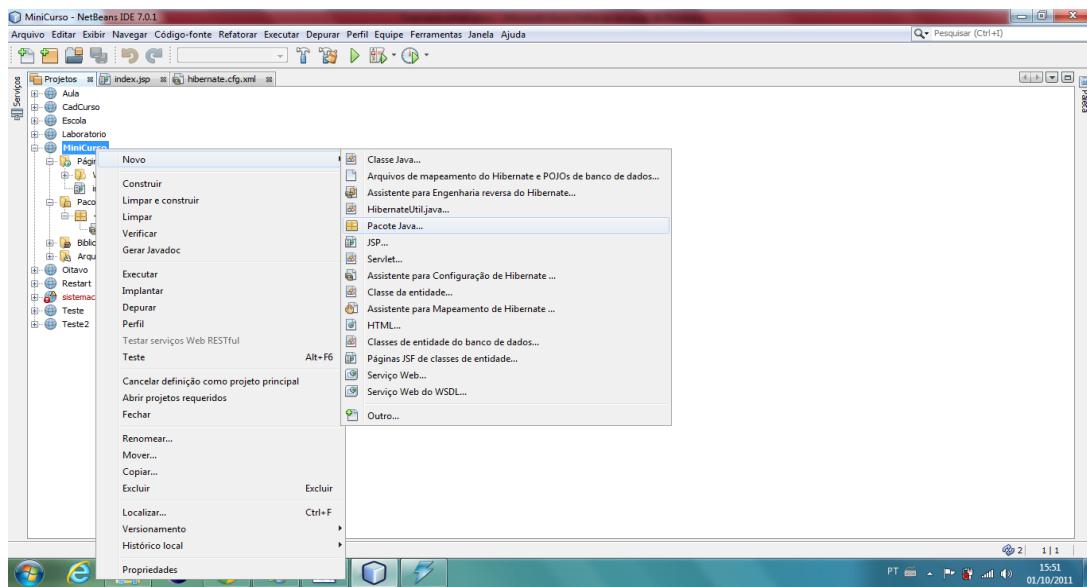
O **hibernate.cfg.xml**, é o arquivo do Framework **Hibernate** que mapeia todos os dados do banco de dados indicado, para realizar a conexão. Nele vem o Driver utilizado, o local, a porta e o banco de dados utilizado, também o usuário e a senha.

Utilizaremos uma nomenclatura de pacote (local onde serão criados os arquivos do nosso projeto) que identificará o local onde estamos trabalhando. Com um dos objetivos do documento é auxiliar os alunos da Fametro na criação dos pacotes utilizaremos a seguinte nomenclatura.

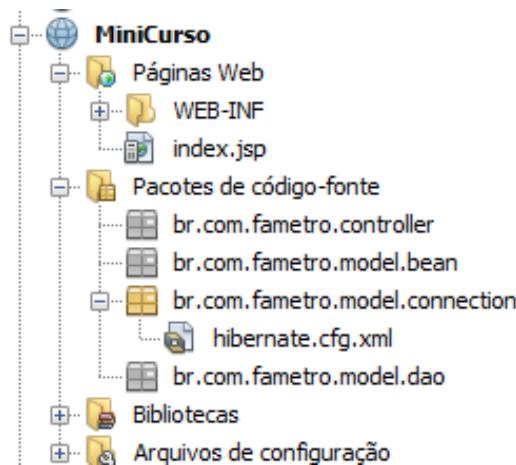
- br.com.fametro.model.connection
- br.com.fametro.model.dao
- br.com.fametro.model.bean
- br.com.fametro.controller

**Os pacotes assumirão esses nomes, pois, trabalharemos com a arquitetura MVC. Model, Controller, View.**

Então seguindo esses passos vamos criar os pacotes botão direito sobre o projeto -> Novo -> Pacote.



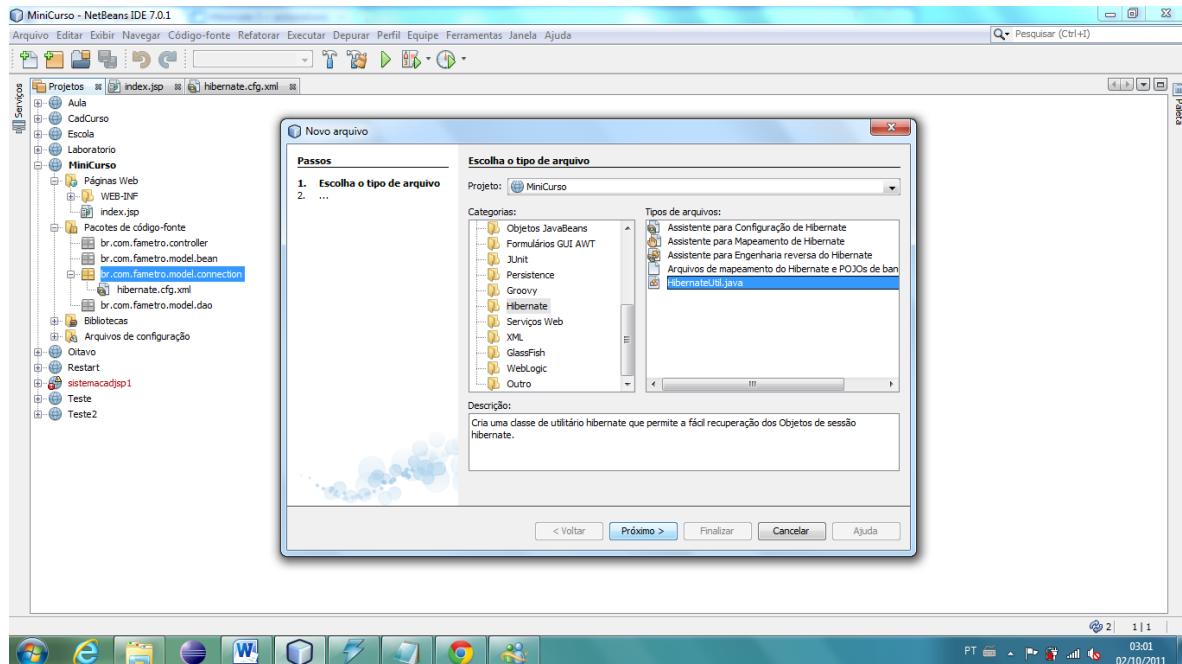
Criado nossos pacote vamos trabalha no pacote **connection**, primeira modificação é jogar o arquivo hibernate.cfg.xml para dentro dele. Segue com o mouse e arraste para dentro do pacote model.connection.



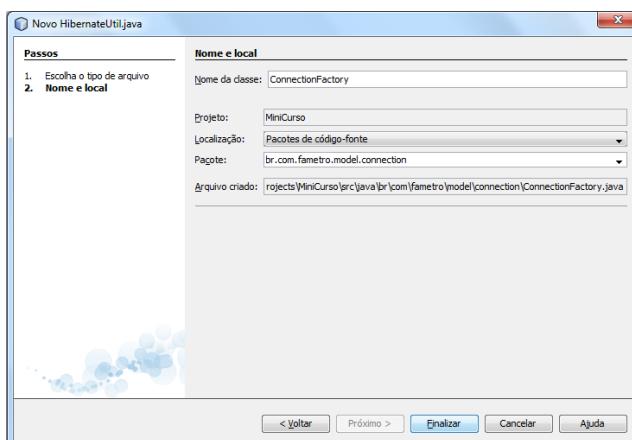
Agora precisamos trabalhar noutro arquivo, o `HibernateUtil.java`. Ele é quem irá criar várias sessões no banco de dados e por esse motivo chamaremos de **ConnectionFactory.java**.

#### Passos:

- Botão direito no pacote connection -> Novo -> Outros -> Pasta Hibernate -> **HibernateUtil.java**



Renomeie a classe para ConnectionFactory. E Finalizar.



Aparecerá essa codificação, precisamos modifica-la.

```

1 /*
2  * To change this template, choose Tools | Templates
3  * and open the template in the editor.
4  */
5 package br.com.fametro.model.connection;
6
7 import org.hibernate.cfg.AnnotationConfiguration;
8 import org.hibernate.SessionFactory;
9
10 /**
11  * Hibernate Utility class with a convenient method to get Session Factory object.
12  *
13  * @author Manfrine
14  */
15 public class ConnectionFactory {
16
17     private static final SessionFactory sessionFactory;
18
19     static {
20         try {
21             // Create the SessionFactory from standard (hibernate.cfg.xml)
22             // config file.
23             sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
24         } catch (Throwable ex) {
25             // Log the exception.
26             System.err.println("Initial SessionFactory creation failed." + ex);
27             throw new ExceptionInInitializerError(ex);
28         }
29     }
30
31     public static SessionFactory getSessionFactory() {
32         return sessionFactory;
33     }
34 }

```

Abaixo as modificações.

The screenshot shows the NetBeans IDE interface with the ConnectionFactory.java file open. Several parts of the code are highlighted with blue circles and arrows pointing to callout boxes containing explanatory text:

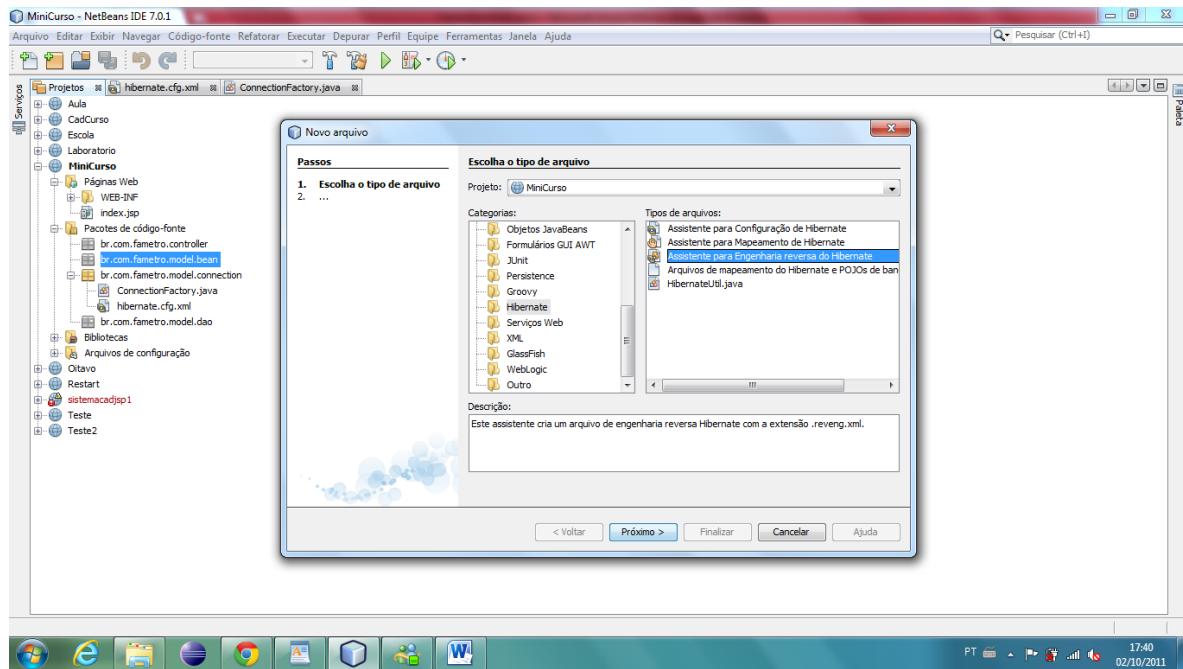
- Line 9:** `import org.hibernate.Session;` is circled and has an arrow pointing to a callout box: "Adiciona a Classe Session do pacote hibernate."
- Line 24:** `sessionFactory = new AnnotationConfiguration().configure("br/com/fametro/model/connection/hibernate.cfg.xml").buildSessionFactory();` is circled and has an arrow pointing to a callout box: "Precisamos apontar a localização do hibernate.cfg.xml, para que a classe ConnectionFactory possa se comunicar com os dados do banco de dados."
- Line 32:** `public static Session getSessionFactory() {` and `return sessionFactory.openSession();` are circled and have an arrow pointing to a callout box: "Do método SessionFactory, retiramos o Factory, precisaremos só do Session."
- Line 32:** `return sessionFactory.openSession();` has an arrow pointing to another callout box: "O tipo de retorno será uma sessionFactory.openSession()."

## MODIFICAÇÕES:

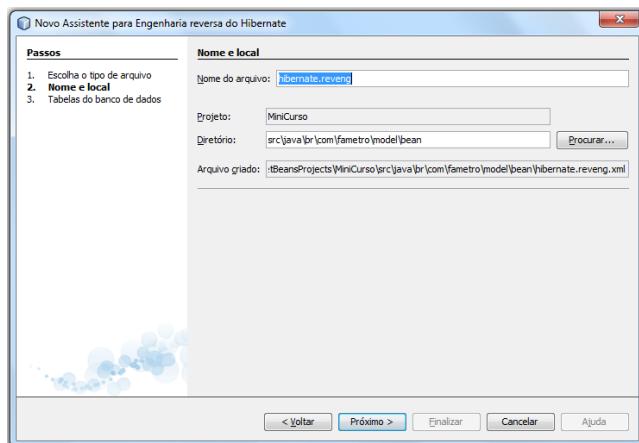
- Adicionar o `import.java.hibernateSession`.
- Adicionar o método `getSessionFactory` da classe `Session`.
- Adicionar o tipo de Retorno do método, no caso o `OpenSession()`.
- Adicionar "`br/com/fametro/model/connection/hibernate.cfg.xml`" no `configure`.

Com a configuração do banco pronta, vamos utilizar um dos melhores recursos do Hibernate, o engenharia versá. O arquin=vo de engenharia reversa, mapeia todos os atributos da tabelas do banco e transforma em classe java.

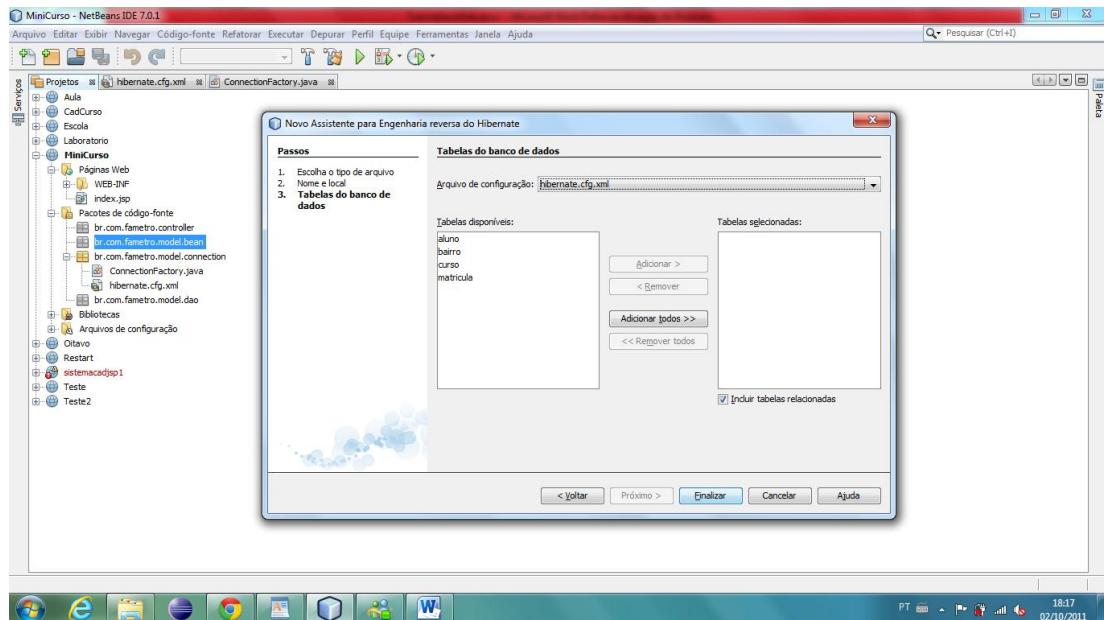
Seguindo os passos vamos lá, clique com o botão direito sobre o pacote model.bean -> Novo -> Outro -> Pasta Hibernate – Assistente de engenharia reversa.



### Próximo



## Próximo



O Hibernate mapeará todas as tabelas do banco de dados, próximo passo é adicionar todos e finalizar.

O arquivo virá assim.

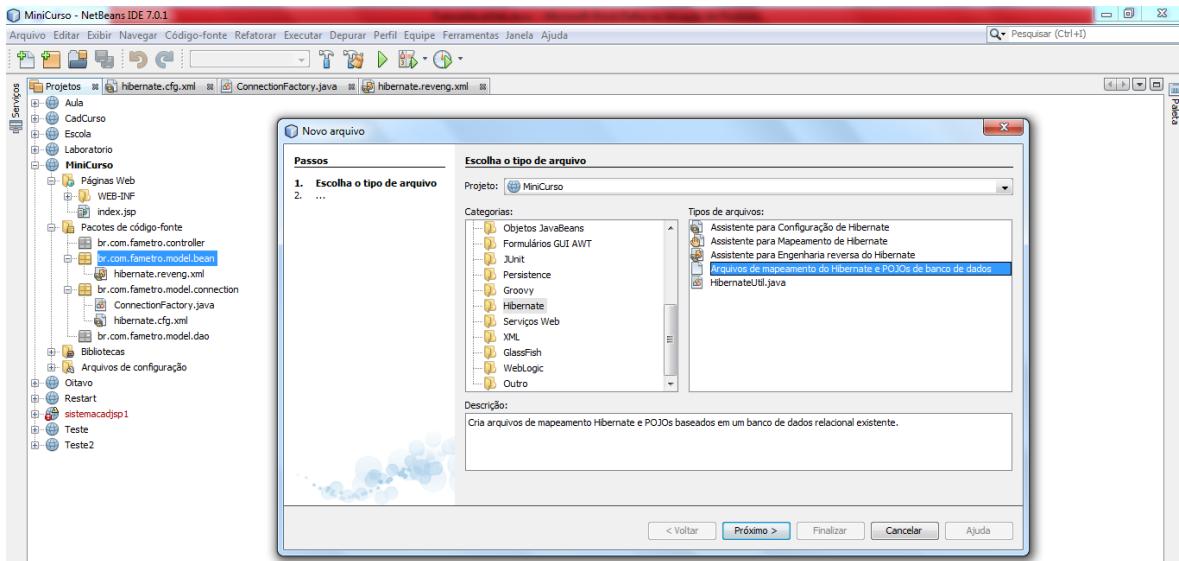
```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-reverse-engineering PUBLIC "-//Hibernate/Hibernate Reverse Engineering DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-reverse->
<hibernate-reverse-engineering>
    <schema-selection match-catalog="minicurso"/>
    <table-filter match-name="matricula"/>
    <table-filter match-name="aluno"/>
    <table-filter match-name="curso"/>
    <table-filter match-name="bairro"/>
</hibernate-reverse-engineering>

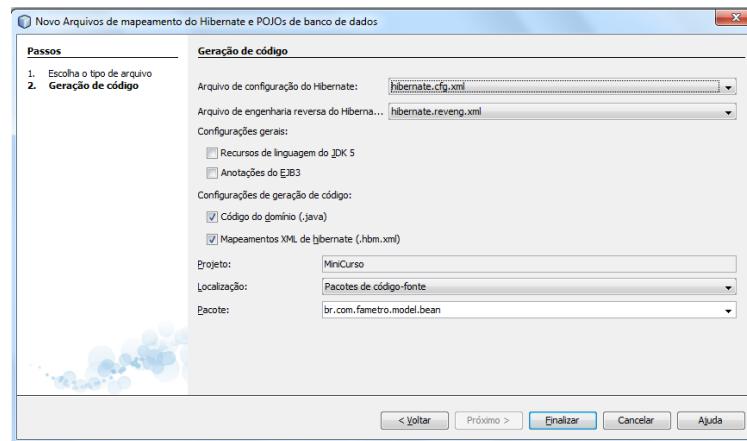
```

Agora precisamos gerar os POJOS. Pojos é uma termos referenciado a classe básicas do Java a quais conterá os atributos e o gets e sets.

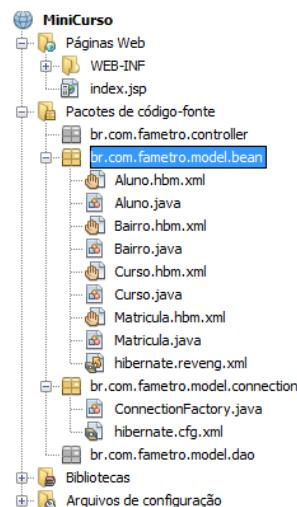
Ainda no pacote model.bean, clique com botão direito -> novo -> Pasta Hibernate -> **Arquivos de Mapeamento do Hibernate e POJOS do Banco de Dados.**



**Próximo:**



**Finalizar**

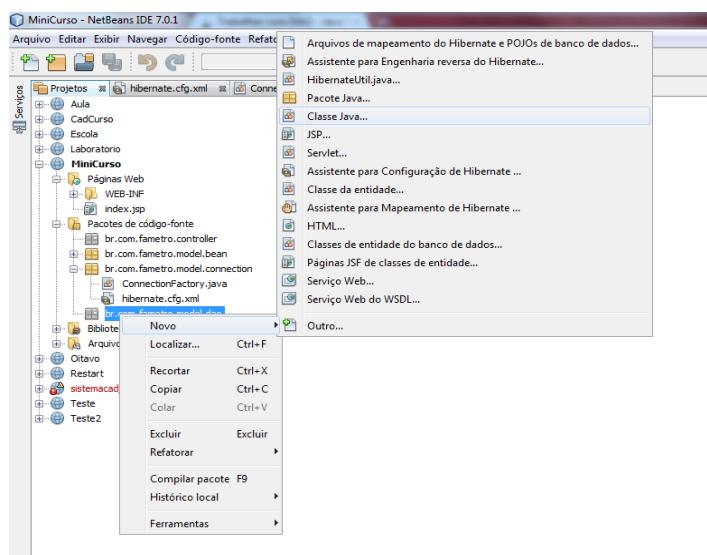


Mapeada as classes básicas, vamos trabalhar no pacote **model.dao**.

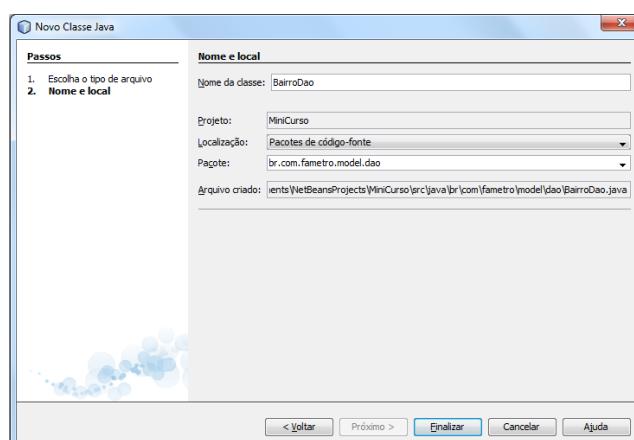
O Dao é também uma nomenclatura que utilizaremos para identificar os arquivo que trarão das transações com o banco de dados, com esses arquivos estaremos transferindo toda a responsabilidade gerenciar todos os trabalhos com o banco de dados, salvar buscar, deletar e realizar as buscas na base.

Vamos utilizar uma classe básica para realizar a criação dos métodos, para esse documento utilizaremos um Dao para cada classe. Mas para o próximo será implementado o modelo DaoGeneric.

Vamos lá, botão direito do mouse sobre o pacote **model.dao** -> Novo -> Classe Java.



O primeiro arquivo que será criado é o classe **BairroDao.java**



## Finalizar

```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  package br.com.fametro.model.dao;
6
7  /**
8   * @author Manfrine
9   */
10 public class BairroDao {
```

The screenshot shows the Java code editor with the generated template for the 'BairroDao' class. The code includes standard JavaDoc comments and imports. The class definition starts with a package declaration 'br.com.fametro.model.dao;' followed by a blank class body.

Agora vamos implementar as características da classe.

```
package br.com.fametro.model.dao;
import br.com.fametro.model.bean.Bairro;
import br.com.fametro.model.connection.ConnectionFactory;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;
```

Todo os pacotes com as Classes que iremos utilizar para esse método.

### Método salvar();

```
/*
 * @author Manfrine
 */
public class BairroDao {

    Session session; // variável do tipo session
    Transaction transaction; // variável do tipo Transaction
    // Objeto do tipo list para ser utilizados por outras classes
    List<Bairro> listaBairro = new ArrayList<Bairro>();

    //método salvar
    public void salvar(Bairro bairro){

        try{
            //Envoca o método estático getSession ele traz todos os atributos
            //da classe Session
            session = new ConnectionFactory().getSessionFactory();
            //Envoca o método para iniciar a transação no banco
            transaction = session.beginTransaction();
            //transfere para o Hibernate a responsabilidade de salvar os dados
            session.save(bairro);
            //efetiva a transação no banco
            transaction.commit();
            //fecha a sessão do banco
            session.close();
        }catch(Exception e){
            //caso ocorra algum erro ele vai mostrar onde.
            JOptionPane.showMessageDialog(null, "+e.getMessage());
            //desfaz toda a transação iniciada
            transaction.rollback();
            //fechar a sessão no banco
            session.close();
        }
    }
}
```

### Método alterar();

```
public void alterar(Bairro bairro){

    try {
        session = new ConnectionFactory().getSessionFactory();
        transaction = session.beginTransaction();

        session.update(bairro); //realiza a alteração

        transaction.commit();
        session.close();
    } catch (Exception e) {

        JOptionPane.showMessageDialog(null, "+e.getMessage());
        transaction.rollback();

        session.close();
    }
}
```

### Método consultarPorNome();

```
public Bairro consultarPorNome(String nome){

    session = new ConnectionFactory().getSessionFactory();
    //como o hibernate não sabe todas as características do banco por exemplo o nome que adicionamos lá
    // precisamos criar a query para fazer isso
    Query query = session.createSQLQuery("SELECT * FROM bairro WHERE bainome = '"+nome+"'").addEntity(Bairro.class);
    System.out.println("SELECT * FROM bairro WHERE bainome = '"+nome+"'");
    //captura uma lista
    listaBairro = query.list();
    //retorna o primeiro nome listado
    return listaBairro.get(0);
}
```

### Método listar();

```
public List<Bairro> listar() {  
  
    session = new ConnectionFactory().getSessionFactory();  
    //lista lista todos os bairros salvos no banco  
    // é o mesmo que select * from  
    listaBairro = session.createCriteria(Bairro.class).list();  
    //retorna toda a lista de dados.  
    return listaBairro;  
}
```

### Método consultarPorId();

```
public Bairro consultarPorId(int codigo) {  
  
    session = new ConnectionFactory().getSessionFactory();  
    //monta-se a query também pois não se pode adivinhar o código que foi criado no banco  
    Query query = session.createSQLQuery("SELECT * FROM bairro WHERE baicod = "+codigo+"").addEntity(Bairro.class);  
    listaBairro = query.list();  
  
    session.close();  
    return listaBairro.get(0);  
}
```

### Método listarAproximado();

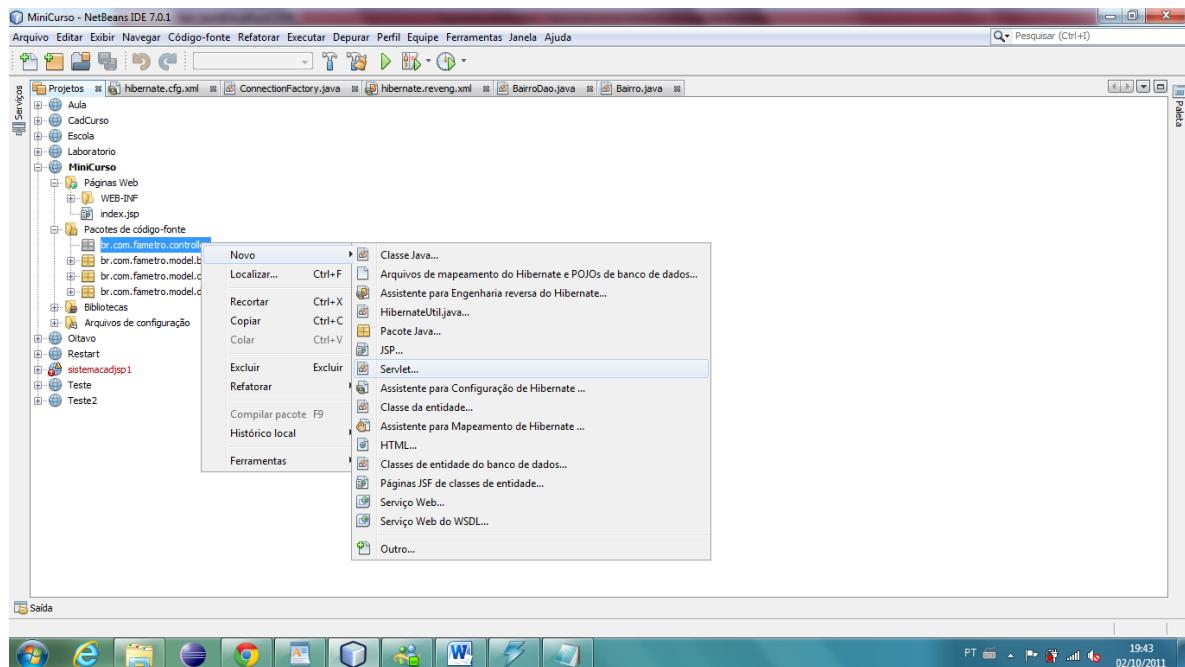
```
public List<Bairro> listarAproximado(String nome) {  
  
    session = new ConnectionFactory().getSessionFactory();  
    //montar a query com o like  
    Query query = session.createSQLQuery("SELECT * FROM bairro WHERE bainome like '%"+nome+"%'").addEntity(Bairro.class);  
    listaBairro = query.list();  
  
    session.close();  
    return listaBairro;  
}
```

### Método deletar();

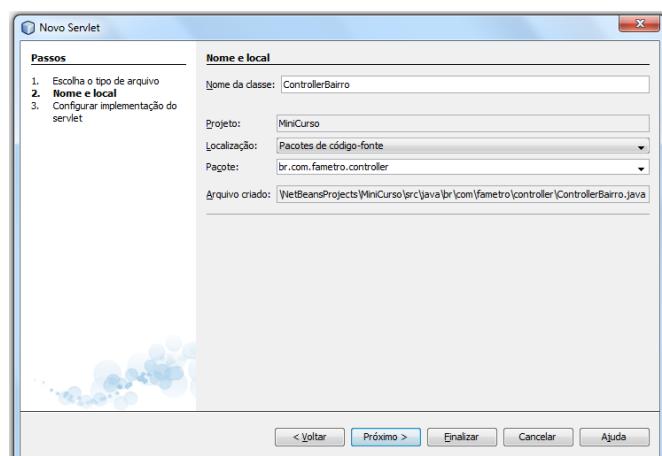
```
public void excluir(Bairro bairro){  
  
    try {  
        session = new ConnectionFactory().getSessionFactory();  
        transaction = session.beginTransaction();  
        //deleta um objeto  
        session.delete(bairro);  
  
        transaction.commit();  
        session.close();  
    } catch (Exception e) {  
  
        JOptionPane.showMessageDialog(null, ""+e.getMessage());  
        transaction.rollback();  
  
        session.close();  
    }  
}
```

Agora precisamos criar a classe de controller. O Controller é a classe que irá gerenciar todas as requisições que venha da tela e comunica com os arquivos do DAO.

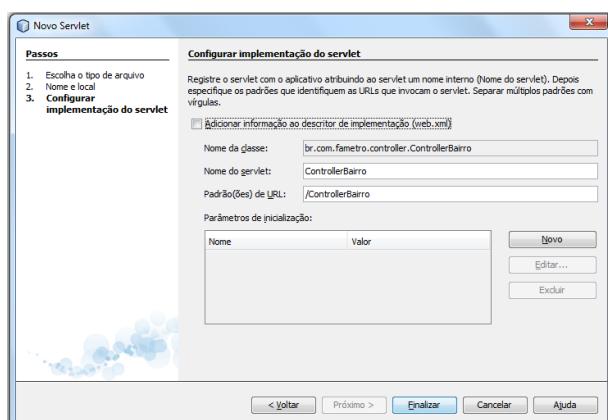
Vamos lá, botão direito do mouse sobre o pacote **controller** -> Novo -> Servlet.



Renomeia para **ControllerBairro**.



**Próximo**



**Finalizar**

```

/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package br.com.fametro.controller;

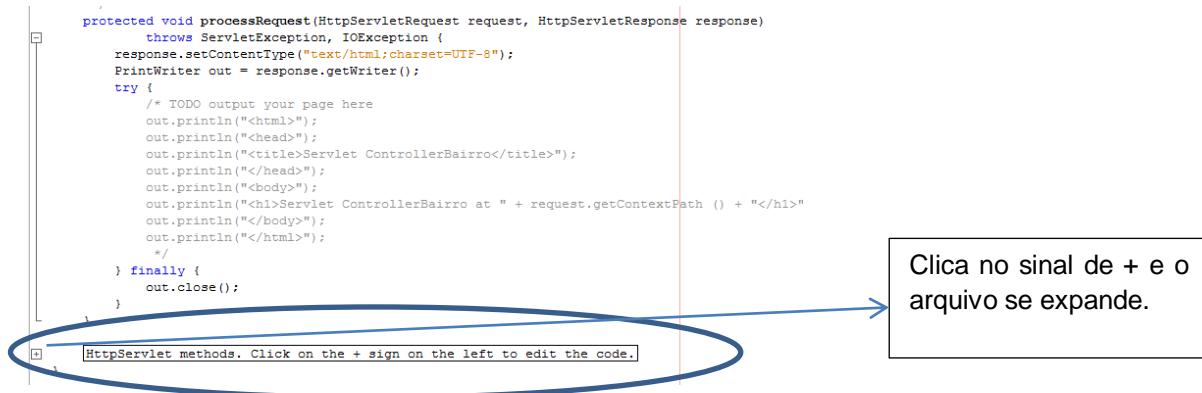
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author Manfrine
 */
@WebServlet(name = "ControllerBairro", urlPatterns = {"/*ControllerBairro"})
public class ControllerBairro extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        ...
}

```

Precisamos localizar o método **doPost**



Clica no sinal de + e o arquivo se expande.

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);
}

}

```

Faça as seguintes alterações.

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    processRequest(request, response);

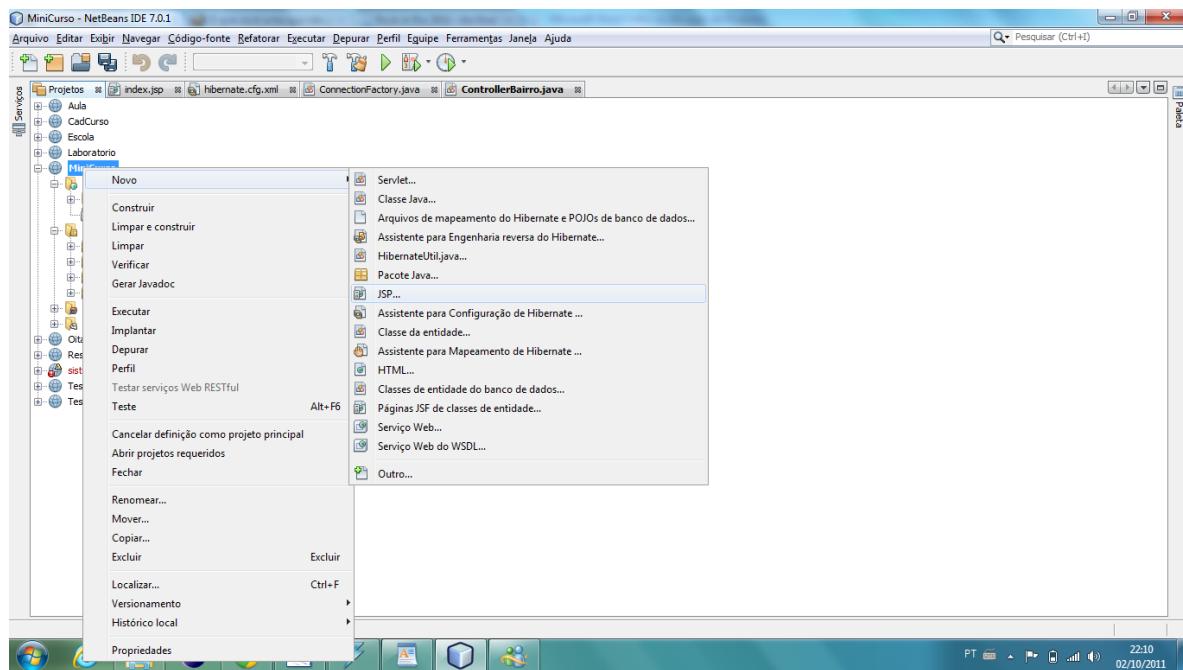
    //instancia da Classe Bairro
    Bairro bairro = new Bairro();
    //instancia da Classe BairroDAO
    BairroDao bairroDao = new BairroDao();
    //seta os valores que serão capturados da tela, no metodo da classe Bairro
    bairro.setBainome(request.getParameter("txt nome"));
    //utiliza o DAO para salvar os dados da tela
    bairroDao.salvar(bairro);
    //retorna o sistema para a tela de cadastro
    request.getRequestDispatcher("CadastroBairro.jsp").include(request, response);
}

```

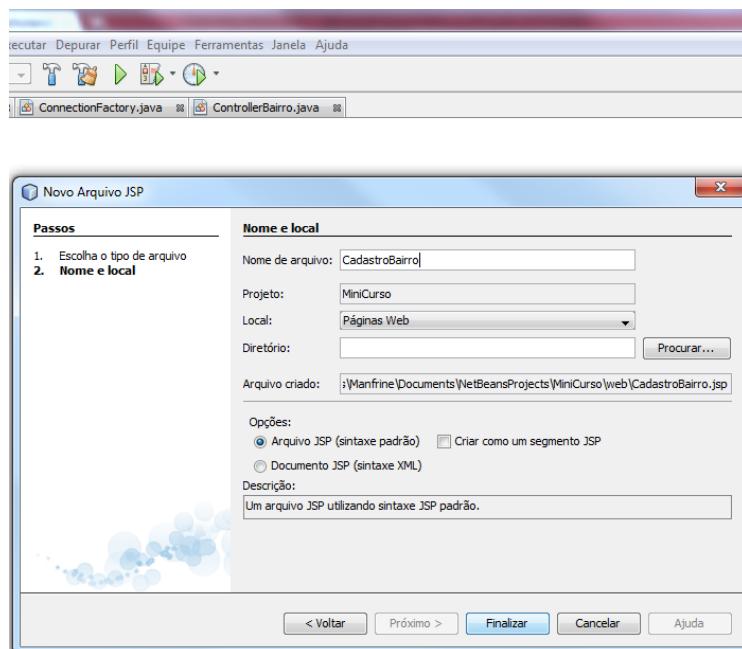
Agora vamos trabalhar na tela do sistema. Utilizaremos a tecnologia JSP (Java Server Pages), para criar as tela WEB.

Utilizaremos também o recurso do NetBeans **Paleta**.

Mas antes vamos criar o arquivo JSP. Clique com o botão direto do mouse sobre o Projeto – > Novo -> JSP.

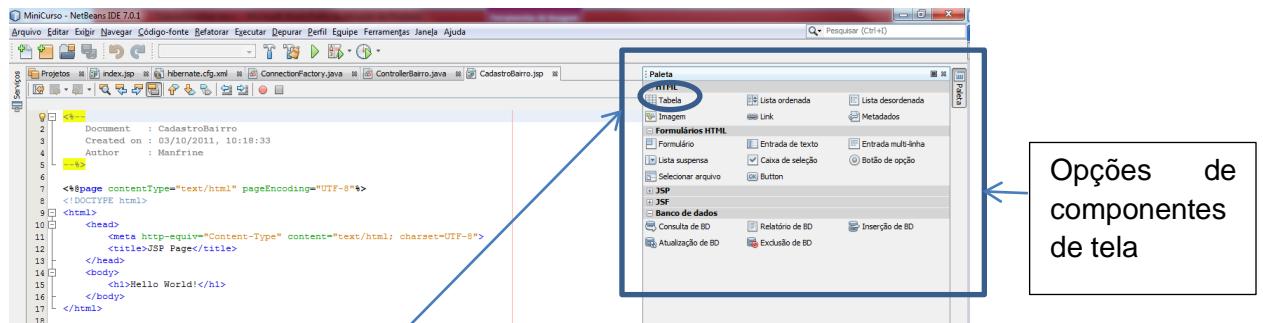
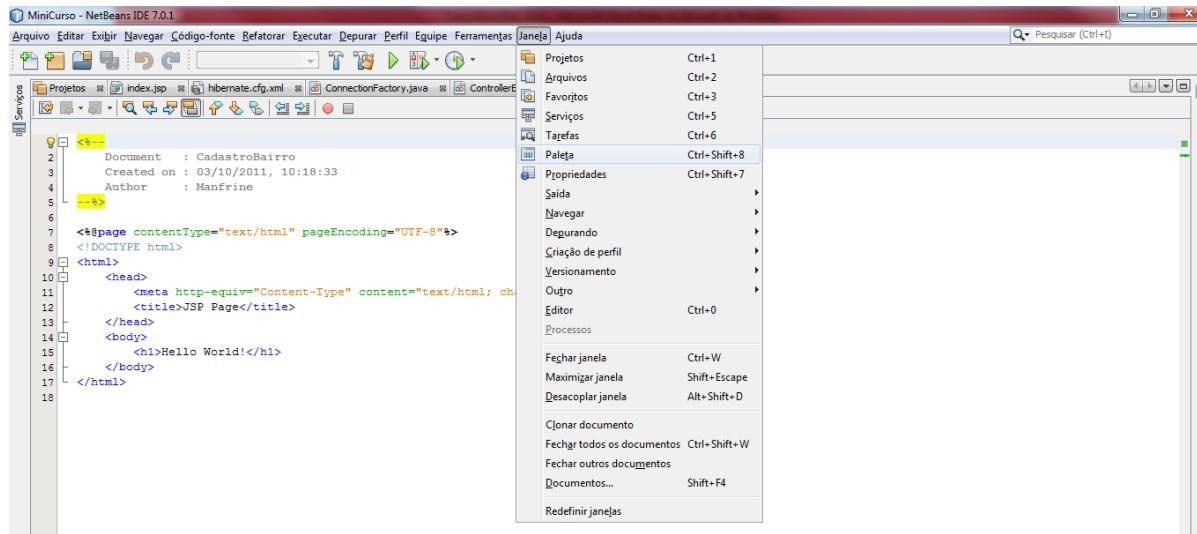


Renomeei o arquivo para **CadastroBairro**.



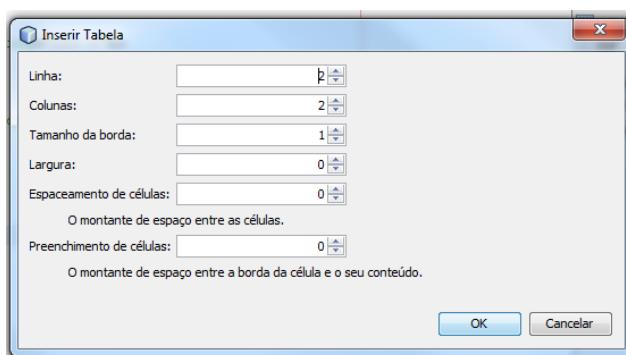
**Clique em Finalizar.**

Com o auxilio do NetBeans, vamos utilizar o recurso de paleta (é um plugin do NetBeans que auxilia na criação dinâmica de janela.). Menu Janela – Paleta.



Vamos utilizar a opção de **Tabelas**.

Segure a opção de tabelas e arraste para dentro do código, especificamente dentro do **<body> </body>**. Uma janela será aberta e precisamos informar a quantidade de colunas e linhas que precisaremos para os campos da tela. Como mostrado na figura abaixo.



Optei por deixar 2(duas) linhas e 2(duas) colunas. Ainda é possível personalizar a borda da tabela e a largura. O seguinte código será gerado.

```

6   html body
7     <%@page contentType="text/html" pageEncoding="UTF-8"%>
8     <!DOCTYPE html>
9   <html>
10    <head>
11      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12      <title>JSP Page</title>
13    </head>
14    <body>
15      <h1>Hello World!</h1>
16
17      <table border="1">
18        <thead>
19          <tr>
20            <th></th>
21            <th></th>
22          </tr>
23        </thead>
24        <tbody>
25          <tr>
26            <td></td>
27            <td></td>
28          </tr>
29          <tr>
30            <td></td>
31            <td></td>
32          </tr>
33        </tbody>
34      </table>
35
36    </body>
37  </html>
38

```

Precisamos modificá-lo.

```

6   html body
7     <%@page contentType="text/html" pageEncoding="UTF-8"%>
8     <!DOCTYPE html>
9   <html>
10    <head>
11      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12      <title>JSP Page</title>
13    </head>
14    <body>
15      <form name="form" action="ControllerBairro" method="post">
16        <table border="1" height="150" width="200">
17          <thead>
18            <tr>
19              <th colspan="2">Cadastro de Bairro</th>
20            </tr>
21          </thead>
22          <tbody>
23            <tr>
24              <td>Nome: </td>
25              <td><input type="text" name="txtnome" value="" /></td>
26            </tr>
27            <tr>
28              <td colspan="2" align="center">
29                <input type="submit" value="Cadastrar" name="btCadastrar" />
30              </td>
31            </tr>
32          </tbody>
33        </table>
34
35      </form>
36
37    </body>
38  </html>

```

Preciso de um form para poder enviar os dados da tela.

Terei que invocar o Controller que é quem vai tratar todas as requisições que venha da tela.

Campo onde eu vou digitar. É o campo de entrada de texto.

Botão.

**Shift +F6** – é o comando para executar o arquivo.

### Tela de Cadastro de Bairro.



E a primeira tela de cadastro está pronta. Feito isso, para todos os cadastros só iremos adicionar mais campo de texto e criar o controller e os dao para a tela, pois os beans, o mapeamento não vai mudar.

Agora vamos Listar o que nós adicionamos na Tela de Bairro.

Para Isso iremos utilizar a tecnologia Scriptlet, que nos dá a possibilidade de escrever código Java na tela JSP. Iremos utilizar a tag `<% %>` para inserir o código e não precisaremos de um controller para listar os dados.

Precisamos de um arquivo JSP, então vamos lá **botão direito sobre o projeto, Novo -> JSP.**

Você terá que alterar o código adicionando os dados do arquivo abaixo.

```

<%@page import="java.util.ArrayList"%>
<%@page import="java.util.List"%>
<%@page import="br.com.fametro.model.dao.BairroDao"%>
<%@page import="br.com.fametro.model.bean.Bairro"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>JSP Page</title>
    </head>
    <body>
        <%
            List<Bairro> listBairro = new ArrayList<Bairro>();
            BairroDao bairroDao = new BairroDao();
            String nome = request.getParameter("txtnome");
        %>
        <table border="1">
            <thead>
                <tr>
                    <th>Codigo</th>
                    <th>Nome</th>
                    <th>&nbsp;</th>
                </tr>
            </thead>
            <tbody>
                <%listBairro = bairroDao.listar();
                for(Bairro b: listBairro) {
                %>
                <tr>
                    <td><%= b.getBaicod() %></td>
                    <td><%= b.getBainome() %></td>
                </tr>
            
```

Importando os pacote onde estão localizados os arquivos que iremos precisar.

A tecnologia Scriptlet, possibilita que façamos isso, utilizar código java numa pag.

Cria na table, os campos que receberam o dados.

Chama o método listar e chama a estrutura de repetição para imprimir a lista.

Enquanto houver nome, ele retorna o código e o nome do bairro

```

<td>
    <input type="button" value="Alterar" name="btAlterar" onclick="location='AlterarBairro.jsp?codigo=<%
b.getBaicod() %>'/>
    <input type="button" value="Excluir" name="btExcluir" onclick="location='ExcluirBairro.jsp?codigo=<%
b.getBaicod() %>'/>
</td>
</tr>
<%>
}
%>
</tbody>
</table>
</body>
</html>

```

**SHIFT + F6**

## Tela de Listar Bairro

Codigo	Nome		
1	Centro	Alterar	Excluir
3	Dom Pedro5555565	Alterar	Excluir

## Implementando um alterar

Para alterar vai seguir a mesma lógica do cadastro, alterando somente o método. Para alterar é preciso antes de mais nada ter listado um objeto, para que a o objeto seja preenchido com informações.

Vamos criar o método Controller do Alterar.

**Botão direito sobre o pacote Controller -> new Servlet.**

Renomear para ControllerAlterarBairro.

No métodos doPost.

```

/*
 * Handles the HTTP <code>POST</code> method.
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
    Bairro bairro = new Bairro();
    BairroDao bairroDao = new BairroDao();

    bairro = bairroDao.consultarPorId(Integer.valueOf(request.getParameter("txtid")));
    bairro.setBairrone(request.getParameter("txtname"));

    bairroDao.alterar(bairro);

    JOptionPane.showMessageDialog(null, "ALTERAÇÃO FEITA COM SUCESSO !");
    request.getRequestDispatcher("ListarBairro.jsp").forward(request, response);
}

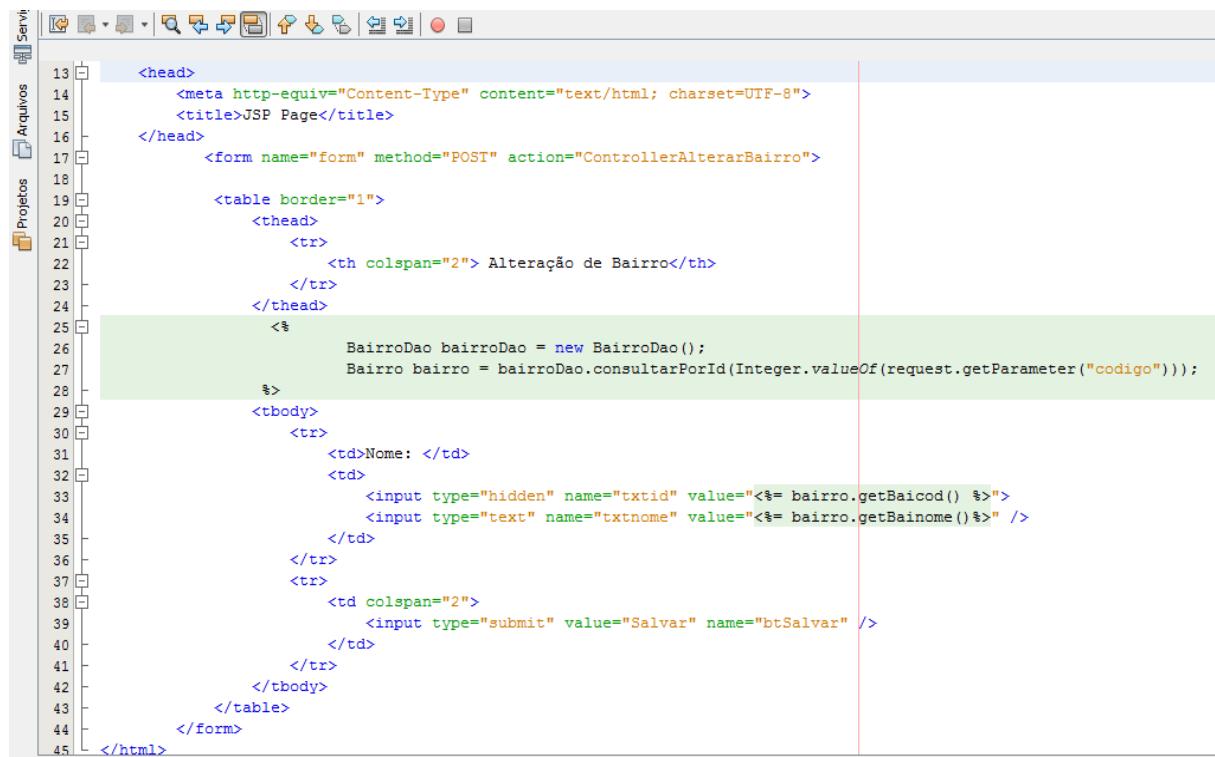
```

## Criando a tela de alterar

Botão direito sobre o projeto Novo - > JSP

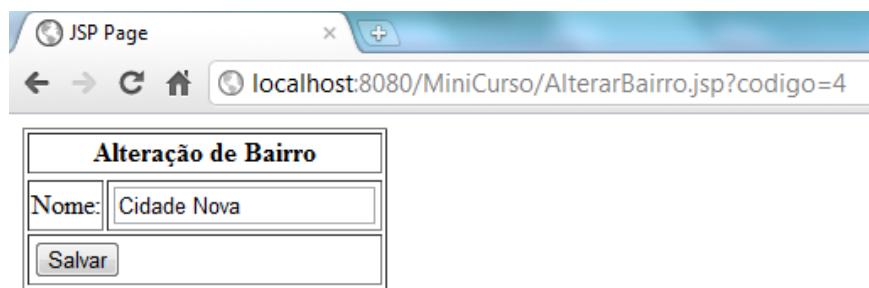
E adiciona o código abaixo.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="br.com.fametro.model.bean.Bairro" %>
<%@page import="br.com.fametro.model.dao.BairroDao" %>
```



```
13 <head>
14     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15     <title>JSP Page</title>
16 </head>
17 <form name="form" method="POST" action="ControllerAlterarBairro">
18
19     <table border="1">
20         <thead>
21             <tr>
22                 <th colspan="2"> Alteração de Bairro</th>
23             </tr>
24         </thead>
25         <%
26             BairroDao bairroDao = new BairroDao();
27             Bairro bairro = bairroDao.consultarPorId(Integer.valueOf(request.getParameter("codigo")));
28         %>
29         <tbody>
30             <tr>
31                 <td>Nome: </td>
32                 <td>
33                     <input type="hidden" name="txtid" value="<% bairro.getBaicod() %>">
34                     <input type="text" name="txtnome" value="<% bairro.getBainome() %>" />
35                 </td>
36             </tr>
37             <tr>
38                 <td colspan="2">
39                     <input type="submit" value="Salvar" name="btSalvar" />
40                 </td>
41             </tr>
42         </tbody>
43     </table>
44 </form>
45 </html>
```

Tela



## Excluindo

Criar um controller do excluir.

```

70 * @throws ServletException if a servlet-specific error occurs
71 * @throws IOException if an I/O error occurs
72 */
73 @Override
74 protected void doPost(HttpServletRequest request, HttpServletResponse response)
75     throws ServletException, IOException {
76     processRequest(request, response);
77
78     processRequest(request, response);
79     Bairro bairro = new Bairro();
80     BairroDao bairroDao = new BairroDao();
81
82     bairro = bairroDao.consultarPorId(Integer.valueOf(request.getParameter("txtid")));
83     bairro.setBainome(request.getParameter("txtname"));
84
85     bairroDao.excluir(bairro);
86
87     JOptionPane.showMessageDialog(null, "EXCLUIU COM SUCESSO !");
88
89     request.getRequestDispatcher("ListarBairro.jsp").forward(request, response);
90
91 }

```

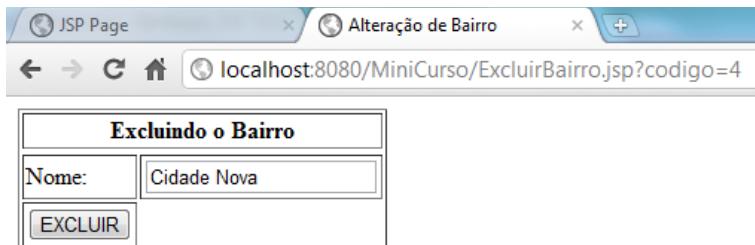
Agora a tela.

```

11 <html>
12   <head>
13     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14     <title>Alteração de Bairro</title>
15   </head>
16   <body>
17     <form name="form" method ="POST" action="ControllerExcluirBairro">
18       <table border="1">
19         <%
20           Bairro bairro = new Bairro();
21           BairroDao bairroDao = new BairroDao();
22           bairro = bairroDao.consultarPorId(Integer.valueOf(request.getParameter("codigo")));
23         %>
24
25         <tbody>
26           <tr>
27             <td>Nome:</td>
28             <td>
29               <input type="hidden" name="txtid" value="<%=(bairro.getBaicod())%>" />
30               <input type="text" name="txtname" value="<%=(bairro.getBainome())%>" />
31             </td>
32           </tr>
33           <tr>
34             <td>
35               <input type="submit" value="EXCLUIR" name="btSalvar"/>
36             </td>
37           </tr>
38         </tbody>
39       </table>
40     </form>
41   </body>
42 </html>

```

Tela



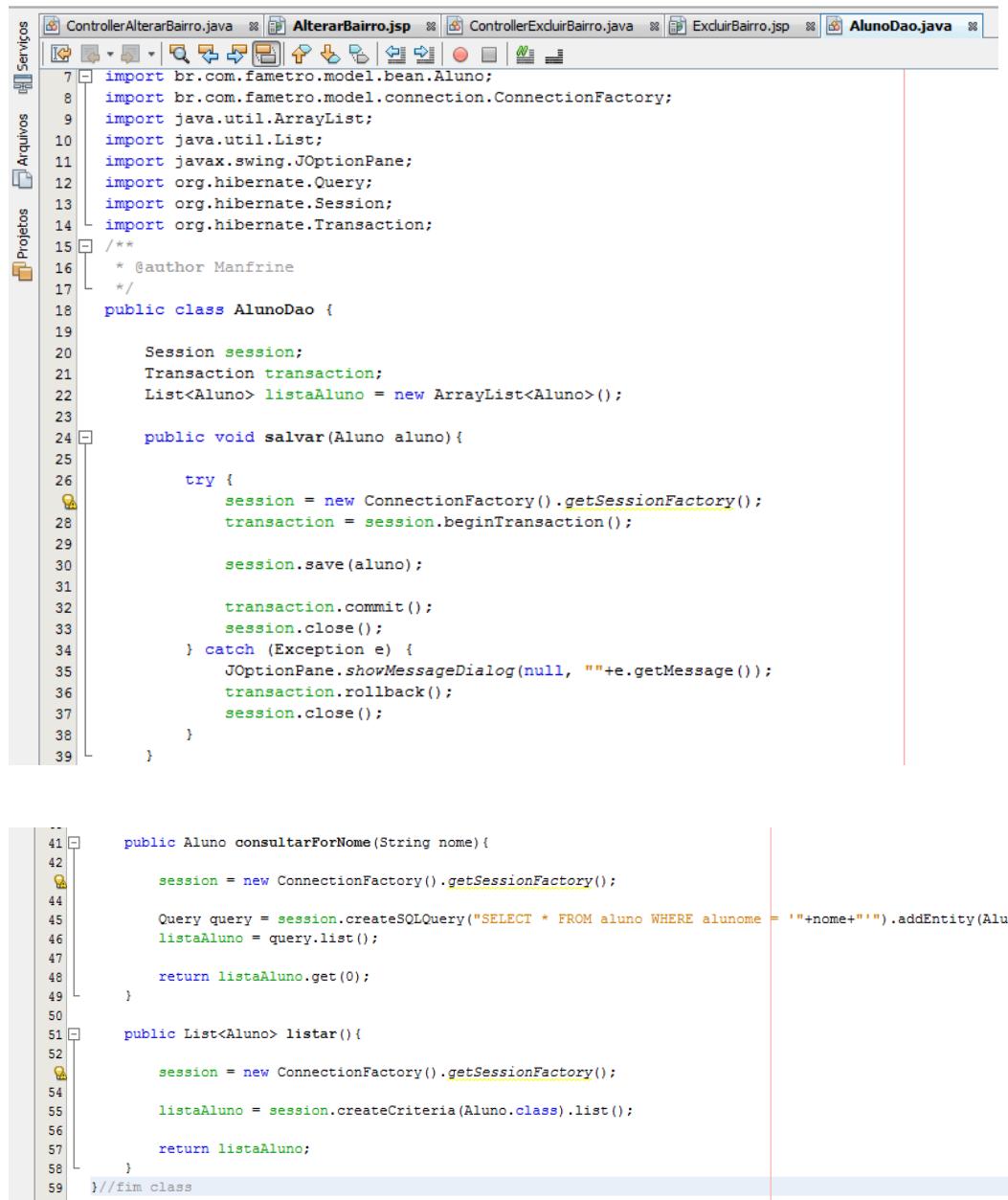


Cadastrar numa tabela que envolva uma relação

Para cadastrar uma tabela que possui uma FK(foreign key) precisamos de um detalhe a mais do que os cadastros que vínhamos fazendo.

Vamos utilizar a tabela/classe **Aluno**.

Precisando criar o Dao do aluno, pois até agora só trabalhamos com o bairro.



The screenshot shows a Java IDE interface with several tabs at the top: ControllerAlterarBairro.java, AlterarBairro.jsp, ControllerExcluirBairro.java, ExcluirBairro.jsp, and AlunoDao.java. The AlunoDao.java tab is active. The left sidebar shows project navigation with 'Arquivos' (Files), 'Projetos' (Projects), and 'Servicos' (Services). The main area displays the AlunoDao.java code:

```
import br.com.fametro.model.bean.Aluno;
import br.com.fametro.model.connection.ConnectionFactory;
import java.util.ArrayList;
import java.util.List;
import javax.swing.JOptionPane;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.Transaction;

/**
 * @author Manfrine
 */
public class AlunoDao {

    Session session;
    Transaction transaction;
    List<Aluno> listaAluno = new ArrayList<Aluno>();

    public void salvar(Aluno aluno){

        try {
            session = new ConnectionFactory().getSessionFactory();
            transaction = session.beginTransaction();

            session.save(aluno);

            transaction.commit();
            session.close();
        } catch (Exception e) {
            JOptionPane.showMessageDialog(null, ""+e.getMessage());
            transaction.rollback();
            session.close();
        }
    }

    public Aluno consultarForNome(String nome){

        session = new ConnectionFactory().getSessionFactory();

        Query query = session.createSQLQuery("SELECT * FROM aluno WHERE alunome = '" + nome + "'").addEntity(Aluno.class);
        listaAluno = query.list();

        return listaAluno.get(0);
    }

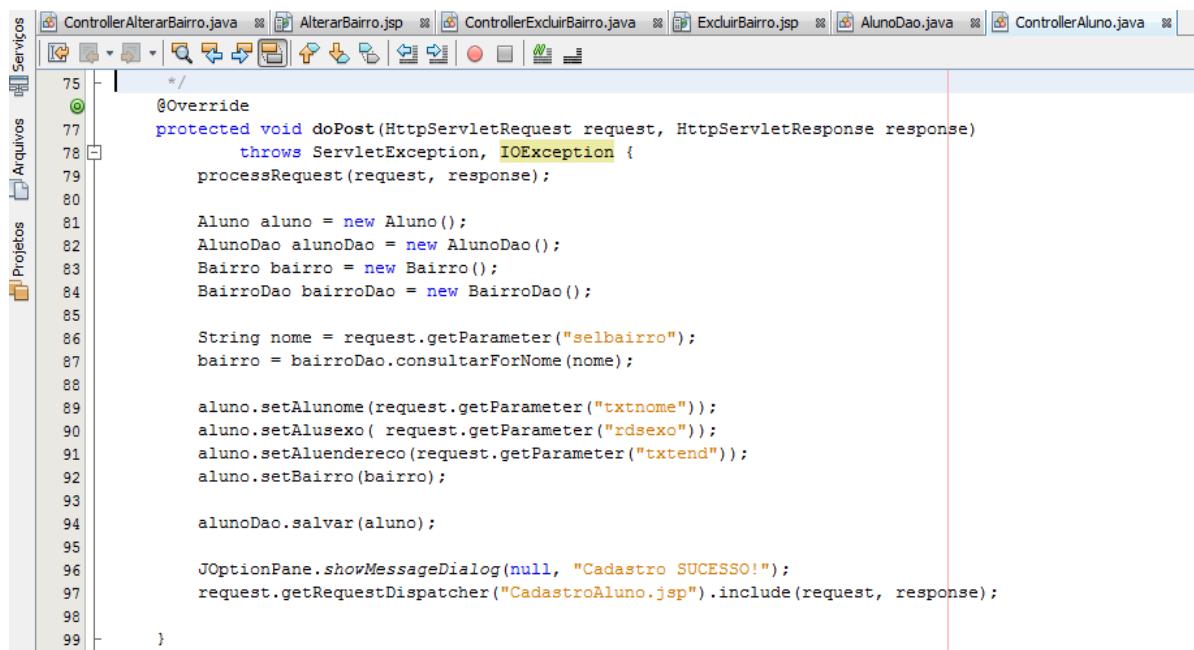
    public List<Aluno> listar(){

        session = new ConnectionFactory().getSessionFactory();

        listaAluno = session.createCriteria(Aluno.class).list();

        return listaAluno;
    }
}
```

## Criando o Controller do Aluno

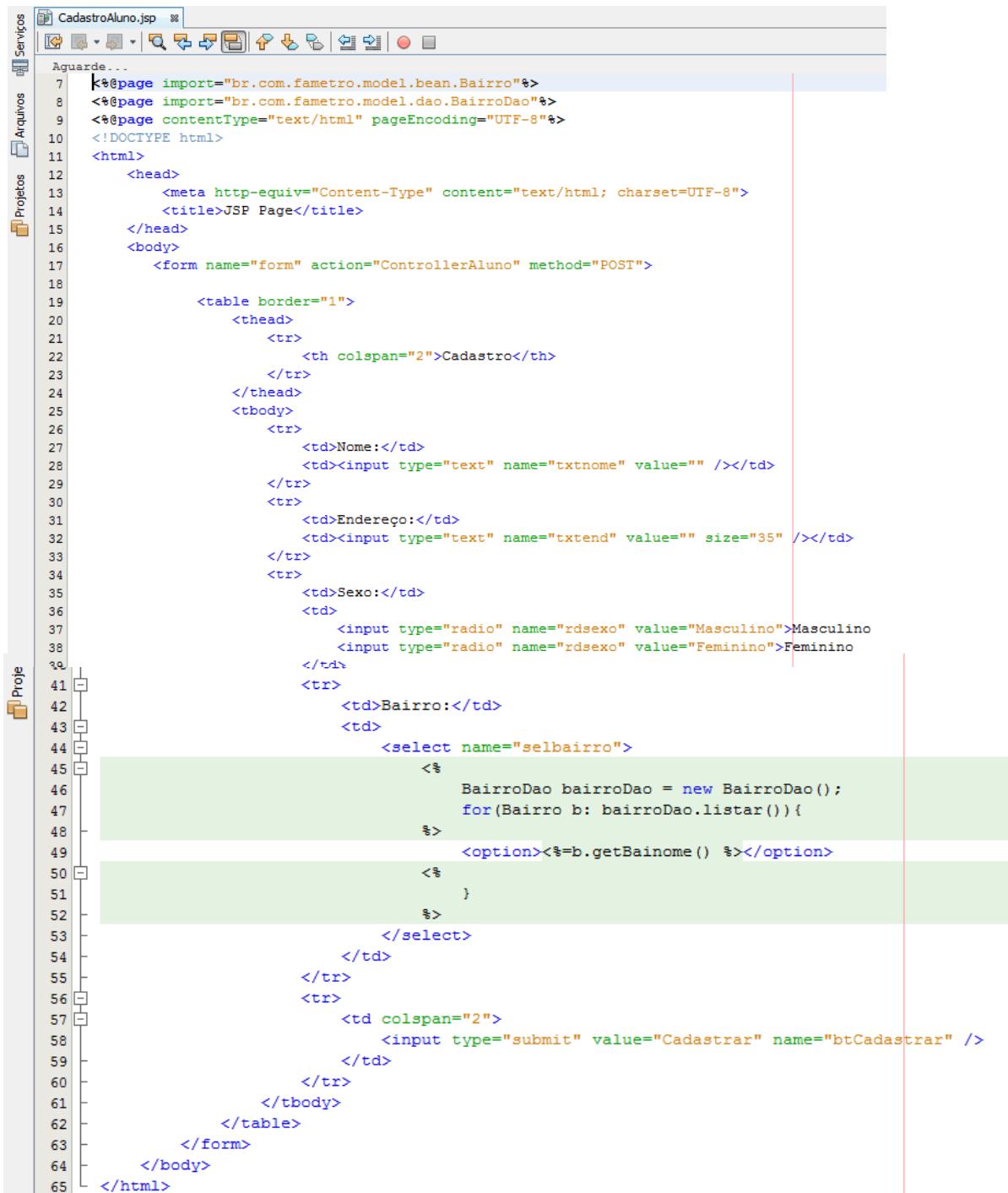


The screenshot shows a Java IDE interface with the following details:

- Toolbar:** Includes icons for file operations (New, Open, Save, Print, Find, Copy, Paste, Cut, Undo, Redo), search, and other common functions.
- Sidebar:** Shows navigation tabs for "Serviços", "Arquivos", and "Projetos".
- Code Editor:** Displays the code for `ControllerAluno.java`. The code is a doPost method implementation. It starts with an annotation `@Override`, followed by the method definition with parameters `HttpServletRequest request, HttpServletResponse response`. Inside the method, it handles exceptions `ServletException` and `IOException`, calls `processRequest`, creates `Aluno`, `AlunoDao`, `Bairro`, and `BairroDao` objects, retrieves a parameter `nome`, sets properties for `aluno`, and saves it to the database via `alunoDao.salvar`. Finally, it shows a success message and includes the response with `JOptionPane.showMessageDialog` and `request.getRequestDispatcher`.

```
75 -    */
76 -    @Override
77 -    protected void doPost(HttpServletRequest request, HttpServletResponse response)
78 -        throws ServletException, IOException {
79 -        processRequest(request, response);
80 -
81 -        Aluno aluno = new Aluno();
82 -        AlunoDao alunoDao = new AlunoDao();
83 -        Bairro bairro = new Bairro();
84 -        BairroDao bairroDao = new BairroDao();
85 -
86 -        String nome = request.getParameter("selbairro");
87 -        bairro = bairroDao.consultarForNome(nome);
88 -
89 -        aluno.setAlunome(request.getParameter("txtnome"));
90 -        aluno.setAlusexo( request.getParameter("rdsexo"));
91 -        aluno.setAluendereco(request.getParameter("txtend"));
92 -        aluno.setBairro(bairro);
93 -
94 -        alunoDao.salvar(aluno);
95 -
96 -        JOptionPane.showMessageDialog(null, "Cadastro SUCESSO!");
97 -        request.getRequestDispatcher("CadastroAluno.jsp").include(request, response);
98 -
99 -    }
```

## Tela o Cadastro de Aluno



```
Aguarde...
7  %@page import="br.com.fametro.model.bean.Bairro"%>
8  %@page import="br.com.fametro.model.dao.BairroDao"%>
9  %@page contentType="text/html" pageEncoding="UTF-8"%>
10 <!DOCTYPE html>
11 <html>
12     <head>
13         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
14         <title>JSP Page</title>
15     </head>
16     <body>
17         <form name="form" action="ControllerAluno" method="POST">
18
19             <table border="1">
20                 <thead>
21                     <tr>
22                         <th colspan="2">Cadastro</th>
23                     </tr>
24                 </thead>
25                 <tbody>
26                     <tr>
27                         <td>Nome:</td>
28                         <td><input type="text" name="txtnome" value="" /></td>
29                     </tr>
30                     <tr>
31                         <td>Endereço:</td>
32                         <td><input type="text" name="txtend" value="" size="35" /></td>
33                     </tr>
34                     <tr>
35                         <td>Sexo:</td>
36                         <td>
37                             <input type="radio" name="rdsexo" value="Masculino">Masculino
38                             <input type="radio" name="rdsexo" value="Feminino">Feminino
39                         </td>
40                     <tr>
41                         <td>Bairro:</td>
42                         <td>
43                             <select name="selbairro">
44                                 <%
45                                     BairroDao bairroDao = new BairroDao();
46                                     for(Bairro b: bairroDao.listar()){
47                                         %>
48                                         <option><%=b.getBainome() %></option>
49                                     }
50                                 %>
51                             </select>
52                         </td>
53                     </tr>
54                     <tr>
55                         <td colspan="2">
56                             <input type="submit" value="Cadastrar" name="btCadastrar" />
57                         </td>
58                     </tr>
59                 </tbody>
60             </table>
61         </form>
62     </body>
63 </html>
```

## Tela

Cadastro	
Nome:	<input type="text" value="Manfrine"/>
Endereço:	<input type="text" value="Rua M"/>
Sexo:	<input checked="" type="radio"/> Masculino <input type="radio"/> Feminino
Bairro:	<input type="button" value="Centro"/>
<input type="button" value="Cadastrar"/>	

## Listar de Alunos

```

1  html body table thead tr th
2  <%-->
3  Document      : ListarAluno
4  Created on   : 19/11/2011, 14:04:25
5  Author        : Manfrine
6
7  <%@page import="java.util.ArrayList"%>
8  <%@page import="java.util.List"%>
9  <%@page import="br.com.fametro.model.dao.AlunoDao"%>
10 <%@page import="br.com.fametro.model.bean.Aluno"%>
11 <%@page contentType="text/html" pageEncoding="UTF-8"%>
12 <!DOCTYPE html>
13 <html>
14   <head>
15     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
16     <title>JSP Page</title>
17   </head>
18   <body>
19     <thead>
20       <tr>
21         <th colspan="2">Listando os Alunos</th>
22       </tr>
23     </thead>
24     <%
25       List<Aluno> listAluno = new ArrayList<Aluno>();
26       AlunoDao alunoDao = new AlunoDao();
27       String nome = request.getParameter("txtnome");
28       String endereco = request.getParameter("txtendereco");
29       String bairro = request.getParameter("txtbairro");
30     %>
31     <table border="1">
32       <thead>
33         <tr>
34           <th>Codigo</th>
35           <th>Nome</th>
36           <th>Endereco</th>
37           <th>Bairro</th>
38           <th>&nbsp;</th>
39         </tr>
40       </thead>
41       <tbody>
42         <%listAluno = alunoDao.listar();>
43         // listBairro = bairroDao.listarAproximado(nome);
44         for(Aluno b: listAluno){>

```

```
45 -    %>
46 -    <tr>
47 -        <td><%= b.getAlucod() %></td>
48 -        <td><%= b.getAlunome() %></td>
49 -        <td><%= b.getAluendereco() %></td>
50 -        <td><%= b.getBairro().getBainome() %></td>
51 -        <td>
52 -            <input type="button" value="Alterar" name="btAlterar" onclick="location.href='?id=<%= b.getId() %>'"/>
53 -            <input type="button" value="Excluir" name="btExcluir" onclick="location.href='?excluir=<%= b.getId() %>'"/>
54 -        </td>
55 -    </tr>
56 -    <%
57 -    }
58 -    %>
59 -    </tbody>
60 -    </table>
61 -
62 -    </body>
63 -
64 -</html>
```